Release

Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair

Department of Statistics / Rice University

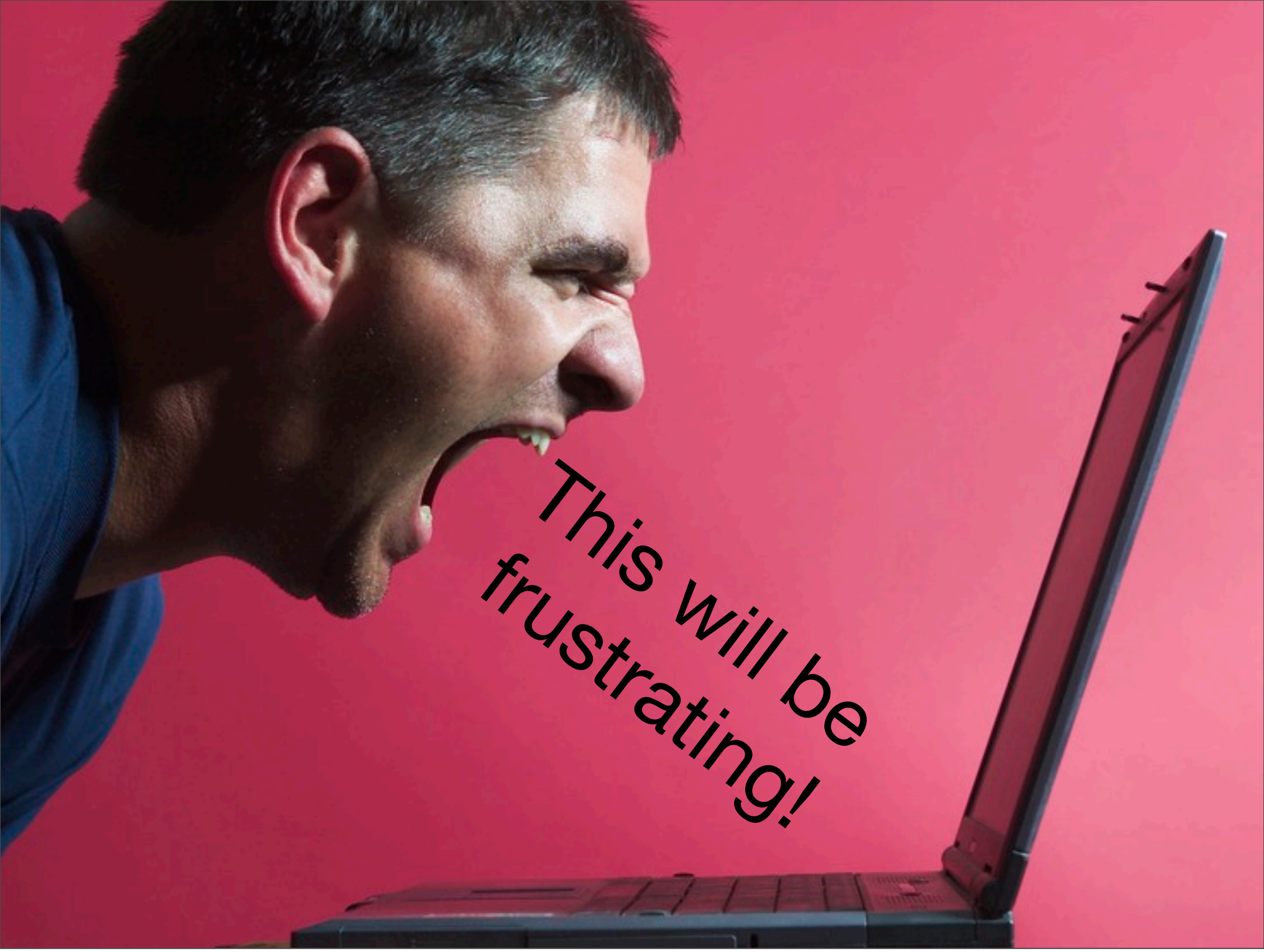
June 2012

CC BY NC

1. `R CMD check`
2. The release process
3. Post-release marketing
4. Source code control
5. Learning more

R CMD

check


This will be
frustrating!

```
check("check-me")

# Fix problems
# Repeat until done
```

```
* using log directory ‘/Users/hadley/Documents/devtools/profr.Rcheck’
* using R version 2.13.0 (2011-04-13)
* using platform: x86_64-apple-darwin9.8.0 (64-bit)
* using session charset: UTF-8
* checking for file ‘profr/DESCRIPTION’ ... OK
* checking extension type ... Package
* this is package ‘profr’ version ‘0.3’
* checking package name space information ... OK
* checking package dependencies ... OK
* checking if this is a source package ... OK
* checking whether package ‘profr’ can be installed ... OK
* checking installed package size ... OK
* checking package directory ... OK
* checking for portable file names ... OK
* checking for sufficient/correct file permissions ... OK
* checking DESCRIPTION meta-information ... OK
* checking top-level files ... OK
* checking index information ... OK
* checking package subdirectories ... OK
* checking R files for non-ASCII characters ... OK
* checking R files for syntax errors ... OK
* checking whether the package can be loaded ... OK
```

```
* checking whether the package can be loaded with stated dependencies ... OK
* checking whether the package can be unloaded cleanly ... OK
* checking whether the name space can be loaded with stated dependencies ... OK
* checking whether the name space can be unloaded cleanly ... OK
* checking for unstated dependencies in R code ... OK
* checking S3 generic/method consistency ... OK
* checking replacement functions ... OK
* checking foreign function calls ... OK
* checking R code for possible problems ... NOTE
.simplify: no visible binding for global variable ‘f’
.simplify: no visible binding for global variable ‘leaf’
ggplot.profr: no visible binding for global variable ‘level’
ggplot.profr: no visible binding for global variable ‘f’
* checking Rd files ... OK
* checking Rd metadata ... OK
* checking Rd cross-references ... OK
* checking for missing documentation entries ... OK
* checking for code/documentation mismatches ... OK
* checking Rd \usage sections ... WARNING
Undocumented arguments in documentation object 'explore'
  df

Functions with \usage entries need to have the appropriate \alias
entries, and all their arguments documented.
The \usage entries must correspond to syntactically valid R code.
See the chapter 'Writing R documentation files' in manual 'Writing R
Extensions'.
```

```
* checking Rd contents ... OK
* checking for unstated dependencies in examples ... OK
* checking contents of 'data' directory ... OK
* checking data for non-ASCII characters ... OK
* checking data for ASCII and uncompressed saves ... OK
* checking examples ... ERROR
Running examples in ‘profr-Ex.R’ failed
The error most likely occurred in:

> ### Name: parse_rprof
> ### Title: Parse Rprof output.
> ### Aliases: parse_rprof
> ### Keywords: debugging
>
> ### ** Examples
> nesting_ex <- system.file("samples", "nesting.rprof", package="profr")
> nesting <- parse_rprof(nesting_ex)
>
> reshape_ex <- system.file("samples", "reshape.rprof", package="profr")
> diamonds <- parse_rprof(reshape_ex)
> p <- profr(parse_rprof(reshape_ex))
```

Types of problem

- **ERROR**: must fix
- **WARNING**: probably should fix
- **NOTE**: might need to fix. Won't prevent CRAN submission

Some tips

- Most common problems are documentation and namespace related. `check_doc()` will allow you to fix these more rapidly.
- Next most common problems are examples. Use `run_examples()`
- The more often you check, the easier it gets. (Eventual aim of devtools is continuous automated checking)

```
document("check-me")
check_doc("check-me")
run_examples("check-me")
run_examples("check-me", "rv.Rd")
check("check-me")
```

Your turn

Run `check("check-me")` and fix the problems that you find.

There are a number of problems for you to find and fix. Don't look at the answers until you've made an attempt at finding them all.

1. Already a package called rv on CRAN:
I renamed to rv2 (also need to change in test-all.R)

2. print generic is function(x, ...)

3. Missing "Suggests: testthat" in description

4. Missing @examples for rif

5. Missing closing } in probability.r

6. # instead of #' in rv.r

7. @returns instead of @return in probability.r

Release process

To CRAN

- Update `NEWS`, checking that dates are correct. Use `show_news` to check that it's in the correct format
- Build package
- Upload to `cran`
- Email `cran@r-project.org`

```
# Or, more simply:
release()

# Don't worry - it asks you a whole lot of questions
# before uploading anything to CRAN.
```

Next

- You'll get an email back from Kurt saying either that your package is now on CRAN, or there were problems you need to fix.
- A day or two later you'll get an email from the windows builder
- If all is well, you can now publicise your package

Post-release

Publicity

- No one will use your package if they don't know it exists.
- You can send a release announcement to `r-packages@stat.math.ethz.ch`. This should include your `README` and `NEWS` for this version
- Also announce on your blog or twitter, if appropriate. (I've recently started using tumblr for this)

Other resources

- **Website**: github pages a good place to start. Include links to papers, presentations etc *(I'm bad at this)*
- **Mailing list**: create a mailing list on google groups
- **Google alerts**: Set up a google alert so you can find out when people are using your package

Source code control

git + github

Makes collaboration much much easier.

Allow you to track changes, and rollback mistakes.

Steep learning curve, but worth it.

Resources

http://help.github.com/

http://gitref.org/

Rstudio git support

Learning more

The command line

Mastery of the command line requires some upfront investment but will save you a lot of time in the long run.

You'll notice I run R from the command line.

Try to avoid any menu items in the gui.

Automate, automate, automate.

```
# .profile
alias R='R --no-save --no-restore-data --quiet'

# .Rprofile
.First <- function() {
  library(grDevices)
  options(
    repos = c(CRAN = "http://cran.r-project.org/"),
    device = "quartz")
}

# .inputrc
"\e[A": history-search-backward
"\e[B": history-search-forward
```

RStudio

The obvious first choice for new R users. Becoming increasing compelling as a development environment.

Learn the keyboard shortcuts!

Future versions will hopefully have tight integration with `testthat` and `devtools`

Where next

http://cran.r-project.org/doc/manuals/R-exts.html#Creating-R-packages

Subscribe to r-devel. Skim emails with interesting topics.

Download and inspect package source code. What makes packages easy to understand? What makes them hard to understand?

Feedback

http://bit.ly/QcRCqV

This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/us/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.