MY FiRST R PACKAGE

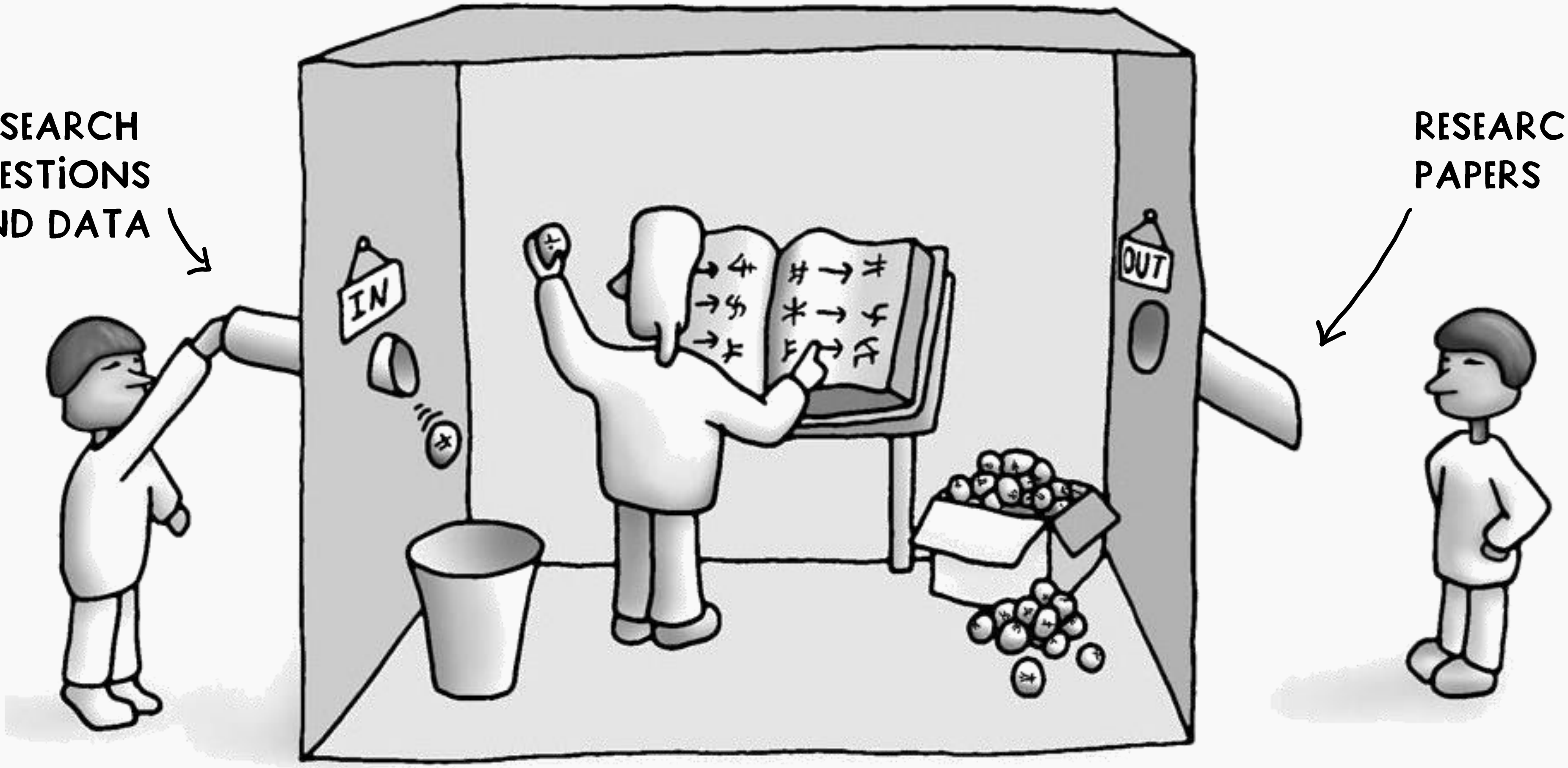
TOBiAS BUSCH, PHD / @TOBiLOTTii / NORDiC RSE GET-TOGETHER / DECEMBER 2020

teebusch.github.io/noah

```
library(noah)
pseudonymize(1:10)

#  [1] "Warm Anaconda"    "Nimble Ptarmigan"
#  [3] "Tired Bear"       "Soft Jellyfish"
#  [5] "Little Tern"      "Elderly Crow"
#  [7] "Excited Seahorse" "Chubby Cricket"
#  [9] "Rough Lobster"    "Chilly Lemming"

pseudonymize(1:10, .alliterate = TRUE)

#  [1] "Private Porcupine" "Vacuous Vulture"
#  [3] "Aware Antelope"    "Strange Snake"
#  [5] "Plucky Pinniped"   "Fancy Falcon"
#  [7] "Low Llama"         "Teeny Tarantula"
#  [9] "Meek Magpie"       "Prickly Panda"
```

THE ~~CHINESE~~ RSE ROOM

Is the person in the room a research software engineer?



RESEARCH QUESTIONS AND DATA

RESEARCH PAPERS

Image from Wikipedia

HOW TO BECOME A (BETTER) RSE?

BUiLD A PACKAGE!

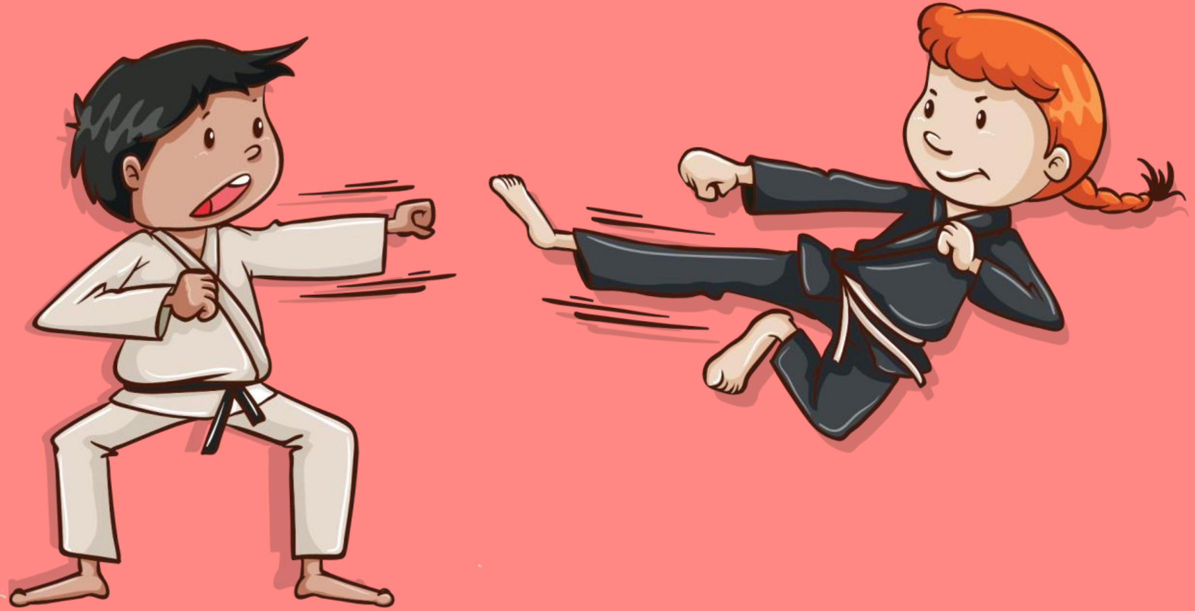


Image by brgfx @ freepik.com

WELL-STRUCTURED

ROBUST

SHAREABLE & EXTENDABLE

DIFFERENT

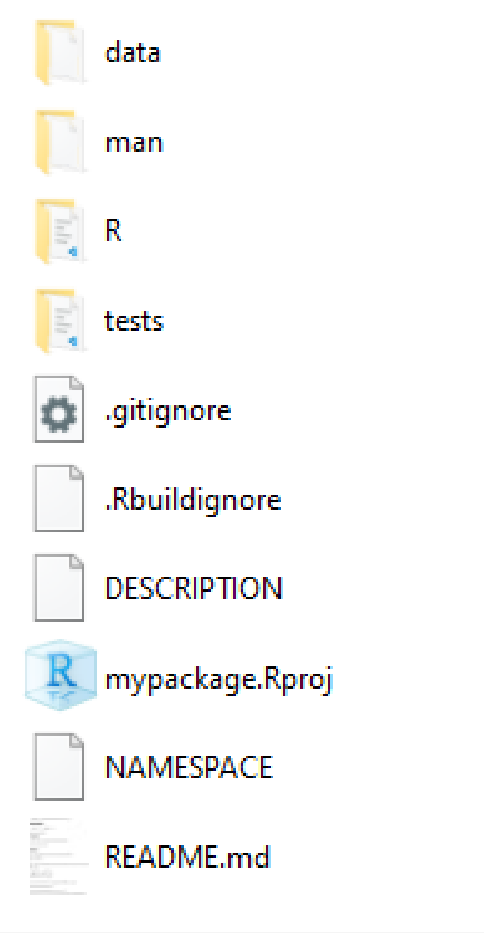
BUILDING PACKAGES HELPS YOU WRITE CODE THAT'S ...

R AND RSTUDIO MAKE BUILDING PACKAGES FUN AND EASY!

Image by brgfx @ freepik.com

File Structure

```
usethis::create_package("mycoolpackage")
```

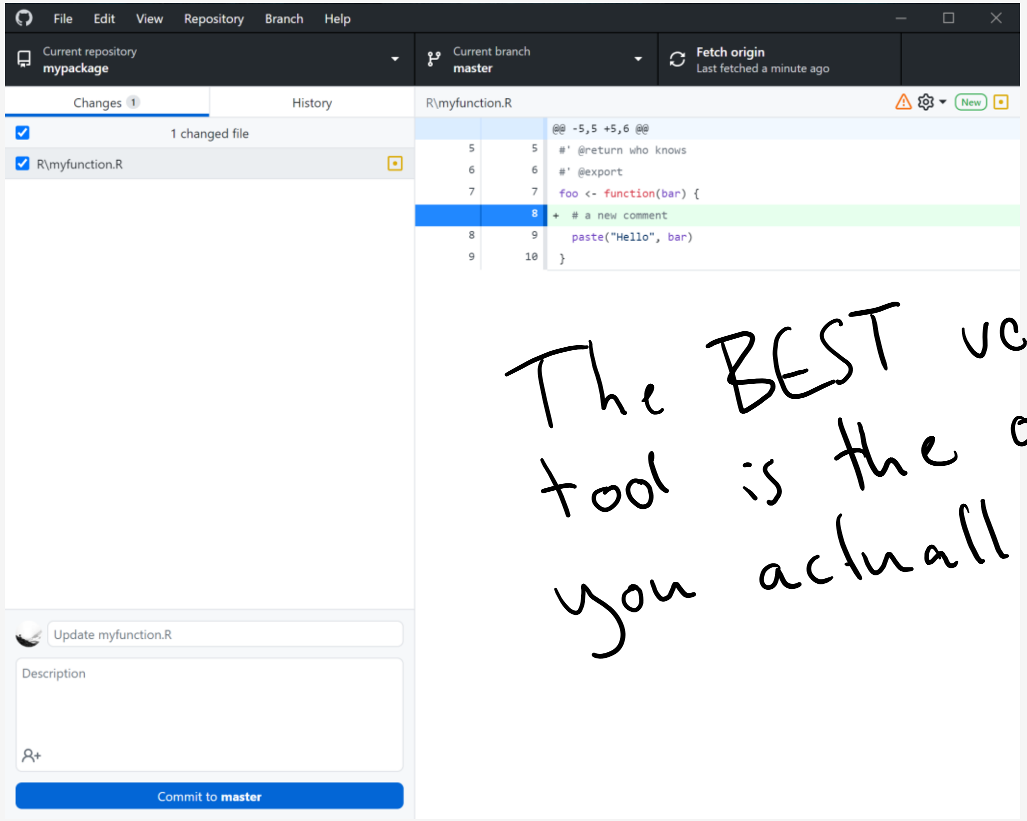


data
man
R
tests
.gitignore
.Rbuildignore
DESCRIPTION
mypackage.Rproj
NAMESPACE
README.md

File Structure

Version Control

```
usethis::use_git()
usethis::use_github()
```



```
@@ -5,5 +5,6 @@
 #' @return who knows
 #' @export
 foo <- function(bar) {
+  # a new comment
   paste("Hello", bar)
 }
```

The BEST version control tool is the one that you actually use!

File Structure

Version Control

Workflow

```r
usethis::use_r("foo")
devtools::load_all(".")   # or Ctrl + Shift + L
```

foo.R

```r
greeting <- function(name) {
  paste("Hello", name, "!")
}
```

File Structure

Version Control

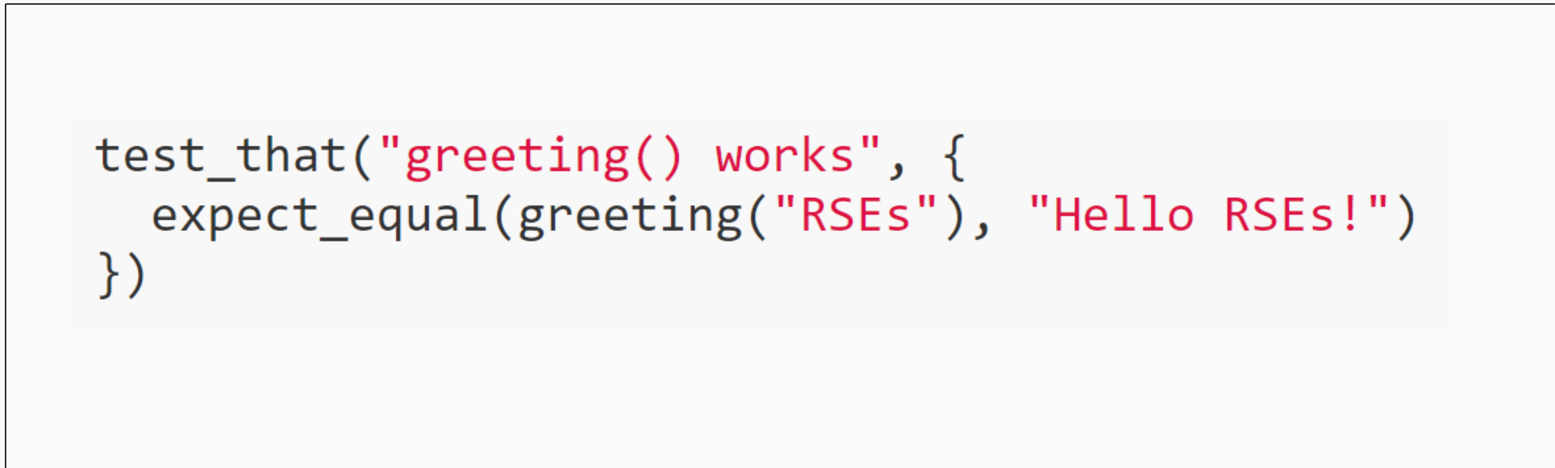
Workflow

Testing

```
usethis::use_test()
devtools::test()     # or Ctrl + Shift + T
devtools::check()    # or Ctrl + Shift + E
```

checks like CRAN would

test-foo.R



```
test_that("greeting() works", {
  expect_equal(greeting("RSEs"), "Hello RSEs!")
})
```

WELL-STRUCTURED **ROBUST** SHAREABLE & EXTENDABLE DIFFERENT

File Structure

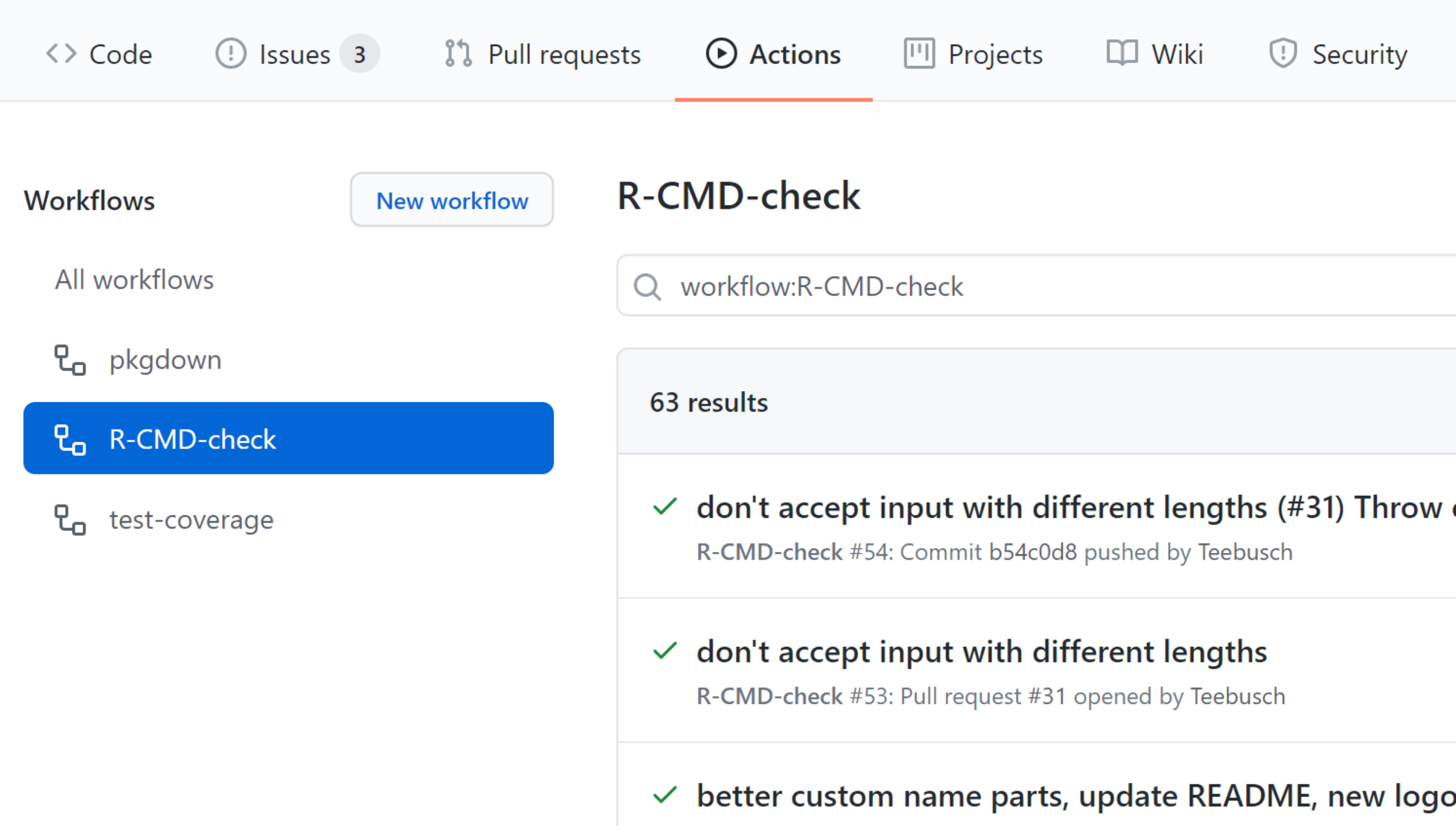
Version Control

Workflow

Testing

Continuous Integration

```
usethis::use_github_action_check_standard()
usethis::use_coverage("codecov")
```



R-CMD-check

workflow:R-CMD-check

63 results

✓ don't accept input with different lengths (#31) Throw
R-CMD-check #54: Commit b54c0d8 pushed by Teebusch

✓ don't accept input with different lengths
R-CMD-check #53: Pull request #31 opened by Teebusch

✓ better custom name parts, update README, new logo

For more Github actions, see https://github.com/r-lib/actions/tree/master/examples

- File Structure
- Version Control
- Workflow
- Testing
- Continuous Integration
- **Documentation**

```r
# create a roxygen skeleton
# with Ctrl + Alt + Shift + R

devtools::document()  # or Ctrl + Shift + D
```

foo.R

```r
#' Greets someone
#'
#' @param name a person to greet
#'
#' @return a greeting
#' @export
greeting <- function(name) {
  paste("Hello", name, "!")
}
```

WELL-STRUCTURED ROBUST **SHAREABLE & EXTENDABLE** DIFFERENT

File Structure

Version Control

Workflow

Testing

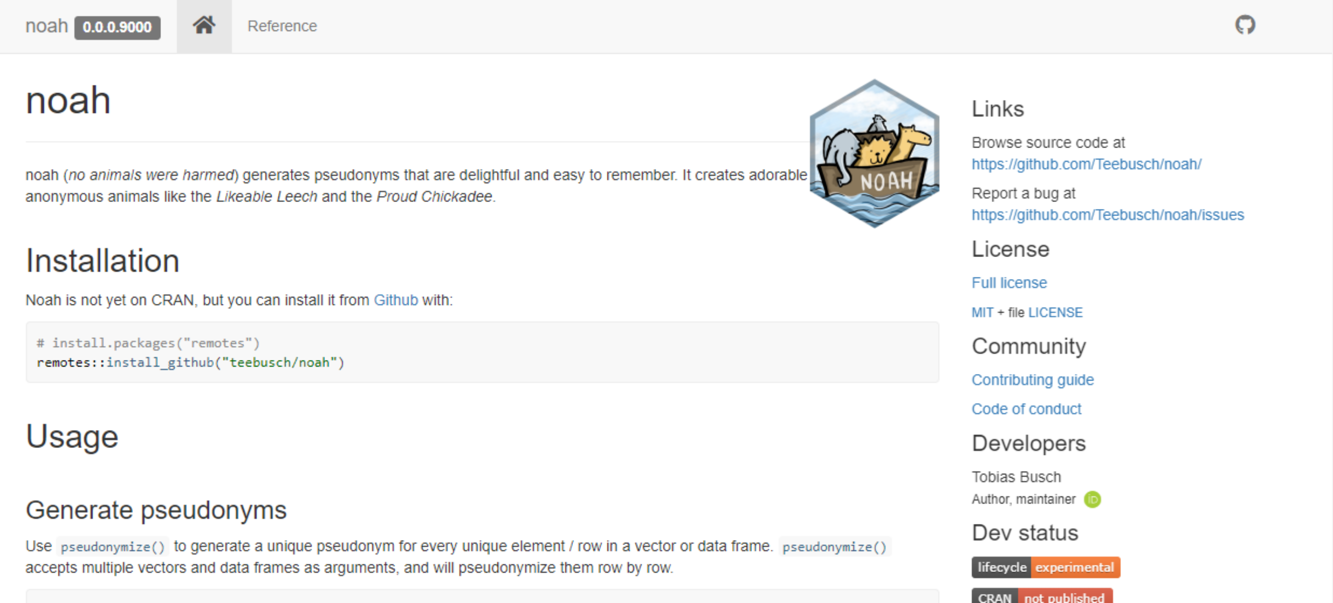
Continuous Integration

Documentation

create an empty branch

```
git checkout --orphan gh-pages
git rm -rf .
git commit --allow-empty -m 'Initial gh-pages commit'
git push origin gh-pages
git checkout master
```

```
usethis::use_pkgdown()
usethis::use_github_action("pkgdown")
```



noah 0.0.0.9000 Reference

noah

noah (*no animals were harmed*) generates pseudonyms that are delightful and easy to remember. It creates adorable anonymous animals like the *Likeable Leech* and the *Proud Chickadee*.

Installation

Noah is not yet on CRAN, but you can install it from Github with:

```
# install.packages("remotes")
remotes::install_github("teebusch/noah")
```

Usage

Generate pseudonyms

Use `pseudonymize()` to generate a unique pseudonym for every unique element / row in a vector or data frame. `pseudonymize()` accepts multiple vectors and data frames as arguments, and will pseudonymize them row by row.

Links

Browse source code at
https://github.com/Teebusch/noah/

Report a bug at
https://github.com/Teebusch/noah/issues

License

Full license

MIT + file LICENSE

Community

Contributing guide

Code of conduct

Developers

Tobias Busch
Author, maintainer

Dev status

More about this here:
https://www.rostrum.blog/2020/08/09/ghactions-pkgs/

- File Structure
- Version Control
- Workflow
- Testing
- Continuous Integration
- **Documentation**

```
# create a roxygen skeleton
# with Ctrl + Alt + Shift + R

devtools::document()  # or Ctrl + Shift + D
```

```
usethis::use_readme_rmd()
devtools::build_readme()
```

File Structure

Version Control

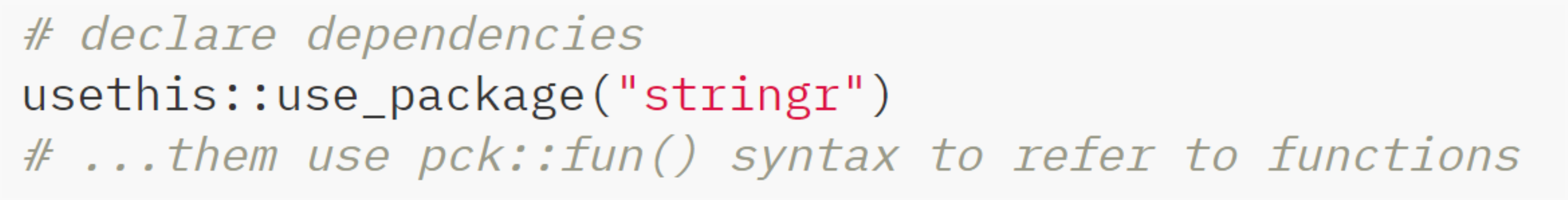
Workflow

Testing

Continuous Integration

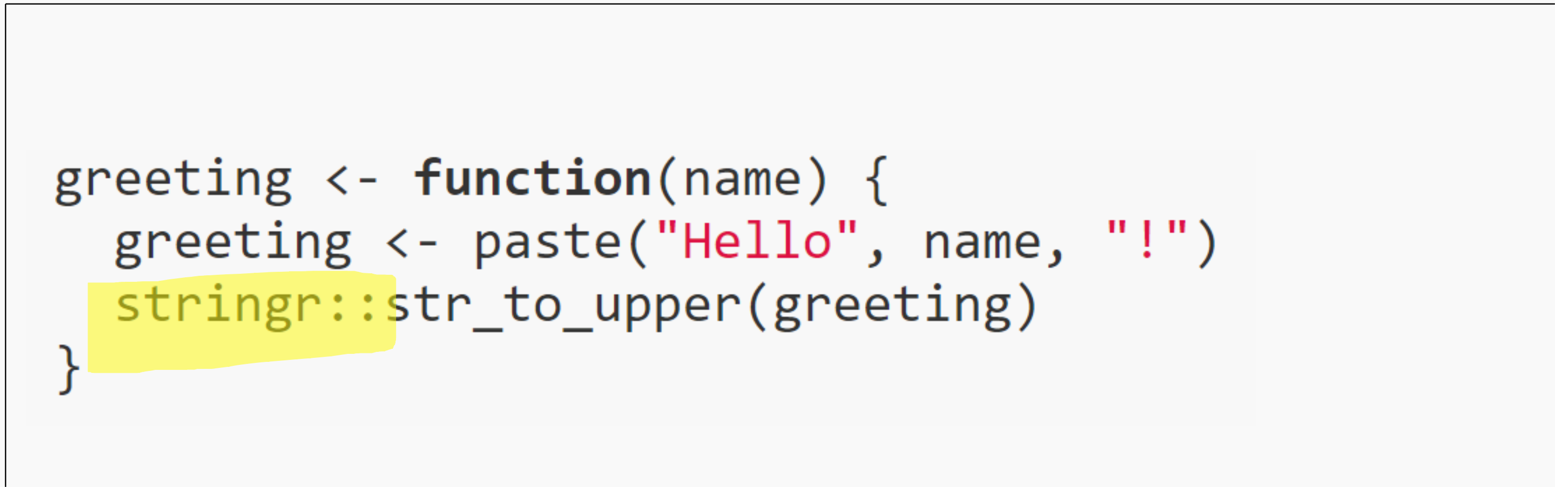
Documentation

Dependency Management



```r
# declare dependencies
usethis::use_package("stringr")
# ...them use pck::fun() syntax to refer to functions
```

foo.R



```r
greeting <- function(name) {
  greeting <- paste("Hello", name, "!")
  stringr::str_to_upper(greeting)
}
```

WELL-STRUCTURED   ROBUST

SHAREABLE & EXTENDABLE

DIFFERENT

- File Structure
- Version Control
- Workflow
- Testing
- Continuous Integration
- Documentation
- Dependency Management
- **Licensing**

```r
usethis::use_mit_license("Tobias Busch")
```

File Structure

Version Control

Workflow

Testing

Continuous Integration

Documentation

Dependency Management

Licensing

Publishing

```
devtools::install("path/package")
devtools::install_github("user/repo")

install.packages("mycoolpackage")
```

install from CRAN

(one day ...)

File Structure

Version Control

Workflow

Testing

Continuous Integration

Documentation

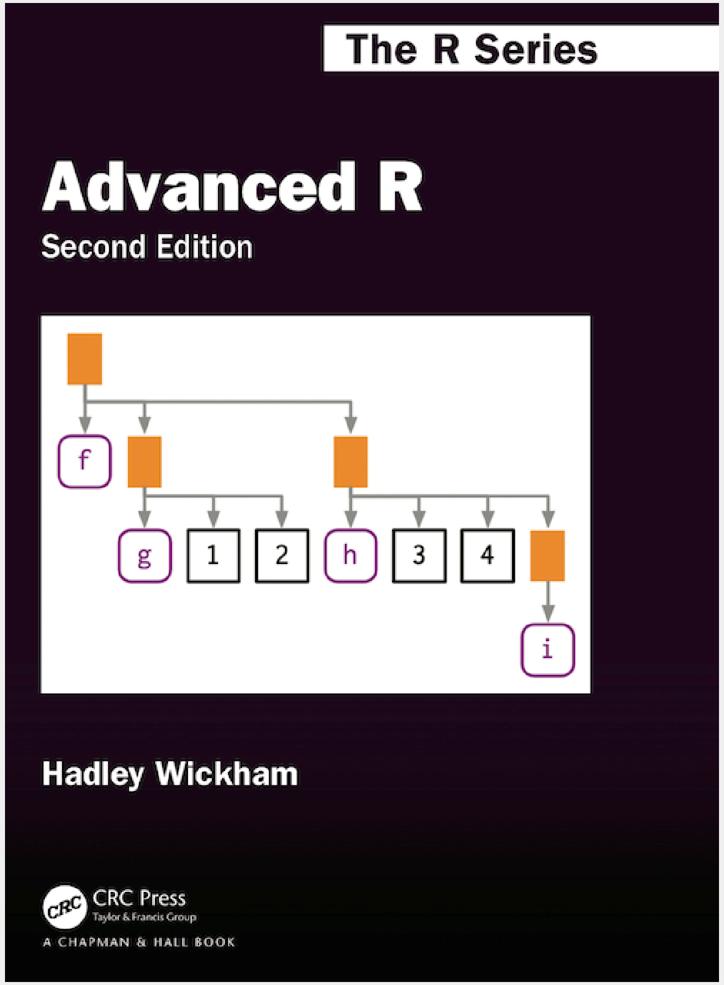
Dependency Management

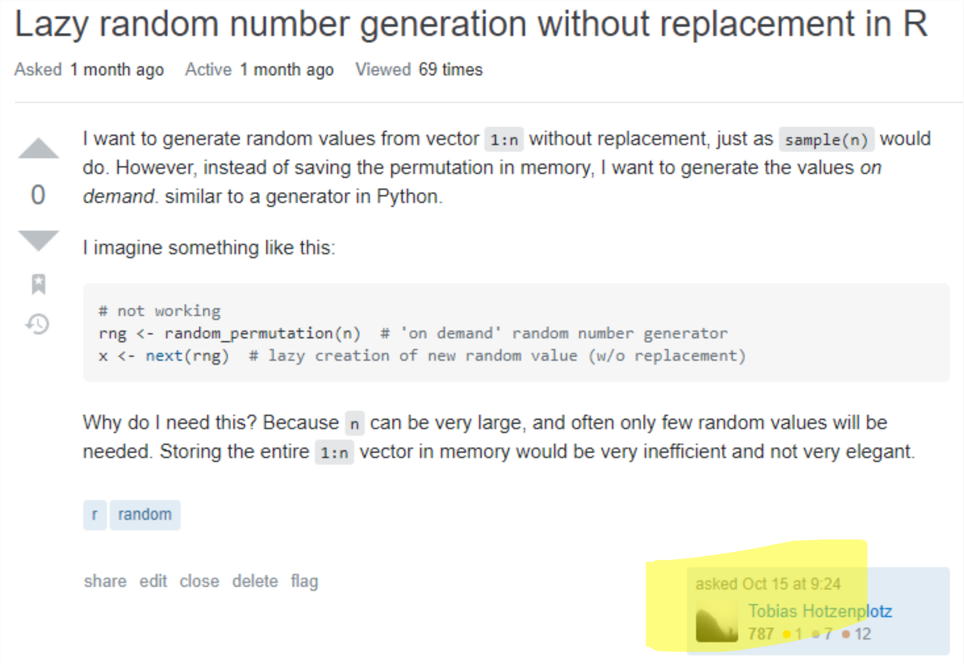
Licensing

Publishing

Uncomfortable Code

adv-r.hadley.nz





Lazy random number generation without replacement in R

Asked 1 month ago Active 1 month ago Viewed 69 times

0

I want to generate random values from vector `1:n` without replacement, just as `sample(n)` would do. However, instead of saving the permutation in memory, I want to generate the values *on demand*. similar to a generator in Python.

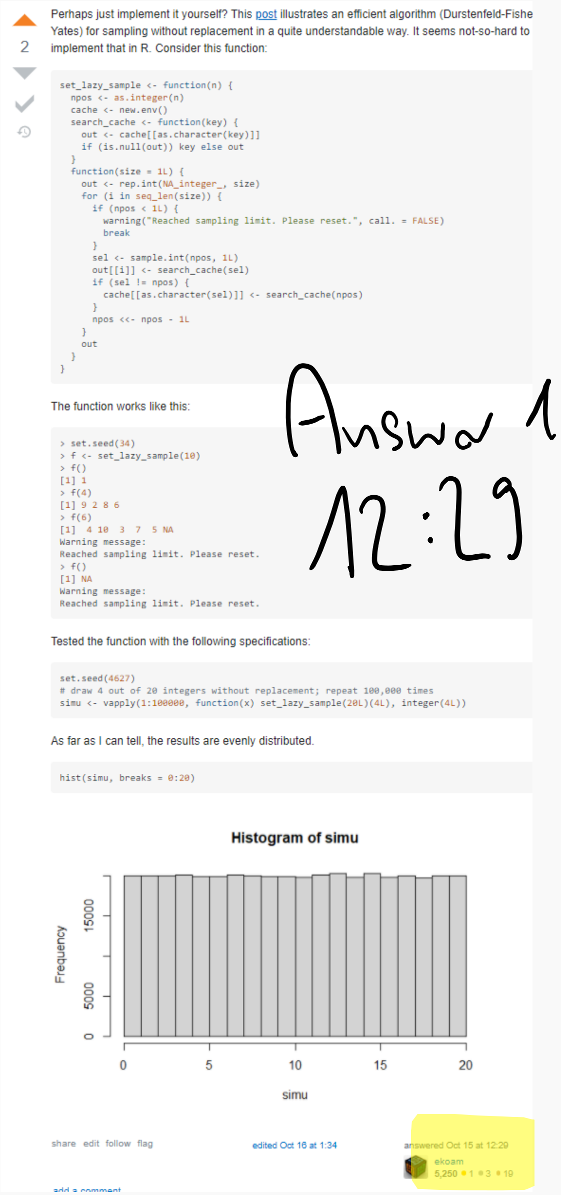
I imagine something like this:

```
# not working
rng <- random_permutation(n)  # 'on demand' random number generator
x <- next(rng)  # lazy creation of new random value (w/o replacement)
```

Why do I need this? Because `n` can be very large, and often only few random values will be needed. Storing the entire `1:n` vector in memory would be very inefficient and not very elegant.

asked Oct 15 at 9:24
Tobias Hotzenplotz
787 ●1 ●7 ●12

Question asked: Thursday 9:24



2

Perhaps just implement it yourself? This post illustrates an efficient algorithm (Durstenfeld-Fisher-Yates) for sampling without replacement in a quite understandable way. It seems not-so-hard to implement that in R. Consider this function:

```
set_lazy_sample <- function(n) {
  npos <- as.integer(n)
  cache <- new.env()
  search_cache <- function(key) {
    out <- cache[[as.character(key)]]
    if (is.null(out)) key else out
  }
  function(size = 1L) {
    out <- rep.int(NA_integer_, size)
    for (i in seq_len(size)) {
      if (npos < 1L) {
        warning("Reached sampling limit. Please reset.", call. = FALSE)
        break
      }
      sel <- sample.int(npos, 1L)
      out[[i]] <- search_cache(sel)
      if (sel != npos) {
        cache[[as.character(sel)]] <- search_cache(npos)
      }
      npos <<- npos - 1L
    }
    out
  }
}
```

The function works like this:

```
> set.seed(34)
> f <- set_lazy_sample(10)
> f()
[1] 1
> f(4)
[1] 9 2 8 6
> f(6)
[1]  4 10  3  7  5 NA
Warning message:
Reached sampling limit. Please reset.
> f()
[1] NA
Warning message:
Reached sampling limit. Please reset.
```

Tested the function with the following specifications:

```
set.seed(4627)
# draw 4 out of 20 integers without replacement; repeat 100,000 times
simu <- vapply(1:100000, function(x) set_lazy_sample(20L)(4L), integer(4L))
```

As far as I can tell, the results are evenly distributed.

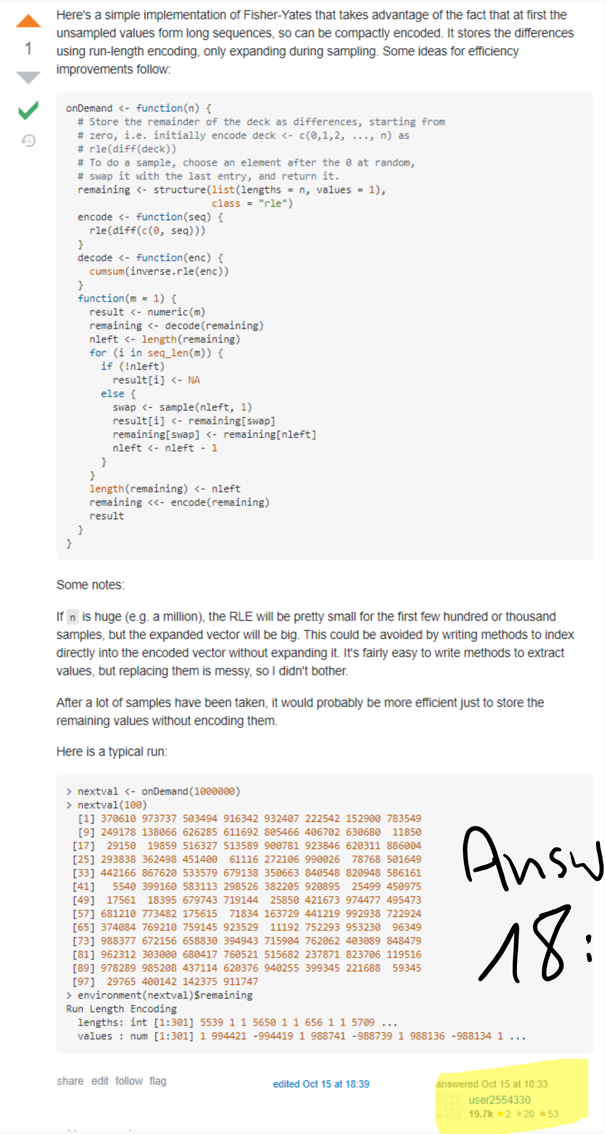
```
hist(simu, breaks = 0:20)
```

Histogram of simu

edited Oct 16 at 1:34

answered Oct 15 at 12:29
ekoam
5,250 ●1 ●3 ●19

Answer 1 12:29



1

Here's a simple implementation of Fisher-Yates that takes advantage of the fact that at first the unsampled values form long sequences, so can be compactly encoded. It stores the differences using run-length encoding, only expanding during sampling. Some ideas for efficiency improvements follow:

```
onDemand <- function(n) {
  # Store the remainder of the deck as differences, starting from
  # zero, i.e. initially encode deck <- c(0,1,2, ..., n) as
  # rle(diff(deck))
  # To do a sample, choose an element after the 0 at random,
  # swap it with the last entry, and return it.
  remaining <- structure(list(lengths = n, values = 1),
                         class = "rle")
  encode <- function(seq) {
    rle(diff(c(0, seq)))
  }
  decode <- function(enc) {
    cumsum(inverse.rle(enc))
  }
  function(m = 1) {
    result <- numeric(m)
    remaining <- decode(remaining)
    nleft <- length(remaining)
    for (i in seq_len(m)) {
      if (!nleft)
        result[i] <- NA
      else {
        swap <- sample(nleft, 1)
        result[i] <- remaining[swap]
        remaining[swap] <- remaining[nleft]
        nleft <- nleft - 1
      }
    }
    length(remaining) <- nleft
    remaining <<- encode(remaining)
    result
  }
}
```

Some notes:

If `n` is huge (e.g. a million), the RLE will be pretty small for the first few hundred or thousand samples, but the expanded vector will be big. This could be avoided by writing methods to index directly into the encoded vector without expanding it. It's fairly easy to write methods to extract values, but replacing them is messy, so I didn't bother.

After a lot of samples have been taken, it would probably be more efficient just to store the remaining values without encoding them.

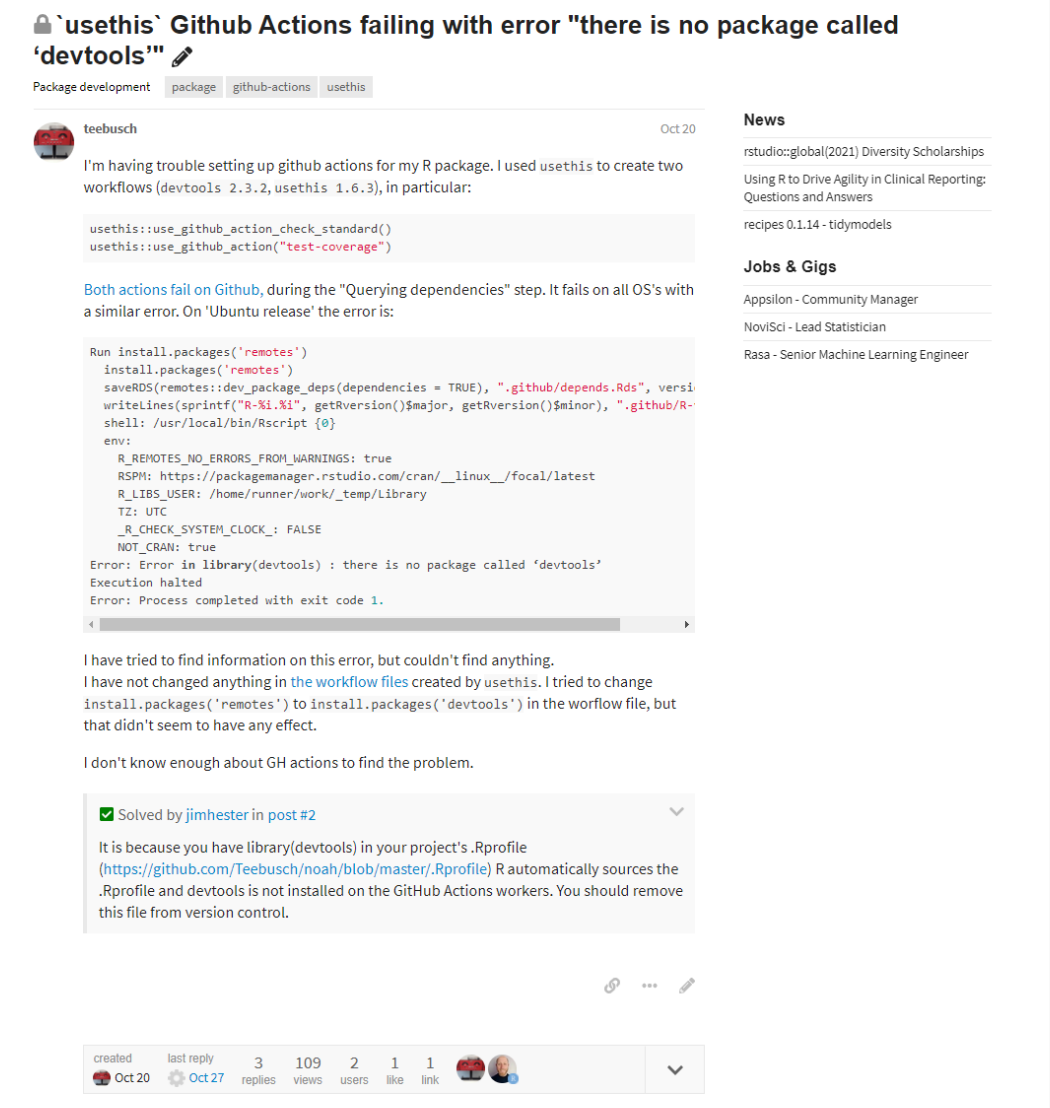
Here is a typical run:

```
> nextval <- onDemand(1000000)
> nextval(100)
  [1] 370610 973737 503494 916342 932407 222542 152900 783549
  [9] 249178 138066 626285 611692 805466 406702 630680  11850
 [17]  29150  19859 516327 513589 900781 923846 620311 886004
 [25] 293838 362498 451400  61116 272106 990026  78768 501649
 [33] 442166 867620 533579 679138 350663 840548 820948 586161
 [41]   5540 399160 583113 298526 382205 920895  25499 450975
 [49]  17561  18395 679743 719144  25850 421673 974477 495473
 [57] 681210 773482 175615  71834 163729 441219 992938 722924
 [65] 374084 769210 759145 923529  11192 752293 953230  96349
 [73] 988377 672156 658830 394943 715904 762062 403089 848479
 [81] 962312 303000 680417 760521 515682 237871 823706 119516
 [89] 978289 985208 437114 620376 940255 399345 221688  59345
 [97]  29765 400142 142375 911747
> environment(nextval)$remaining
Run Length Encoding
  lengths: int [1:301] 5539 1 1 5650 1 1 656 1 1 5709 ...
  values : num [1:301] 1 994421 -994419 1 988741 -988739 1 988136 -988134 1 ...
```

edited Oct 15 at 18:39

answered Oct 15 at 18:33
user2554330
19.7k ●2 ●20 ●53

Answer 2 18:33



`usethis` Github Actions failing with error "there is no package called 'devtools'"

Package development | package | github-actions | usethis

teebusch Oct 20

I'm having trouble setting up github actions for my R package. I used `usethis` to create two workflows (`devtools 2.3.2`, `usethis 1.6.3`), in particular:

```
usethis::use_github_action_check_standard()
usethis::use_github_action("test-coverage")
```

Both actions fail on Github, during the "Querying dependencies" step. It fails on all OS's with a similar error. On 'Ubuntu release' the error is:

```
Run install.packages('remotes')
  install.packages('remotes')
  saveRDS(remotes::dev_package_deps(dependencies = TRUE), ".github/depends.Rds", versi
  writeLines(sprintf("R-%i.%i", getRversion()$major, getRversion()$minor), ".github/R-
  shell: /usr/local/bin/Rscript {0}
  env:
    R_REMOTES_NO_ERRORS_FROM_WARNINGS: true
    RSPM: https://packagemanager.rstudio.com/cran/__linux__/focal/latest
    R_LIBS_USER: /home/runner/work/_temp/Library
    TZ: UTC
    _R_CHECK_SYSTEM_CLOCK_: FALSE
    NOT_CRAN: true
Error: Error in library(devtools) : there is no package called 'devtools'
Execution halted
Error: Process completed with exit code 1.
```

I have tried to find information on this error, but couldn't find anything.
I have not changed anything in the workflow files created by `usethis`. I tried to change `install.packages('remotes')` to `install.packages('devtools')` in the worflow file, but that didn't seem to have any effect.

I don't know enough about GH actions to find the problem.

> ✓ Solved by jimhester in post #2
>
> It is because you have library(devtools) in your project's .Rprofile (https://github.com/Teebusch/noah/blob/master/.Rprofile) R automatically sources the .Rprofile and devtools is not installed on the GitHub Actions workers. You should remove this file from version control.

created Oct 20 | last reply Oct 27 | 3 replies | 109 views | 2 users | 1 like | 1 link

News

rstudio::global(2021) Diversity Scholarships

Using R to Drive Agility in Clinical Reporting: Questions and Answers

recipes 0.1.14 - tidymodels

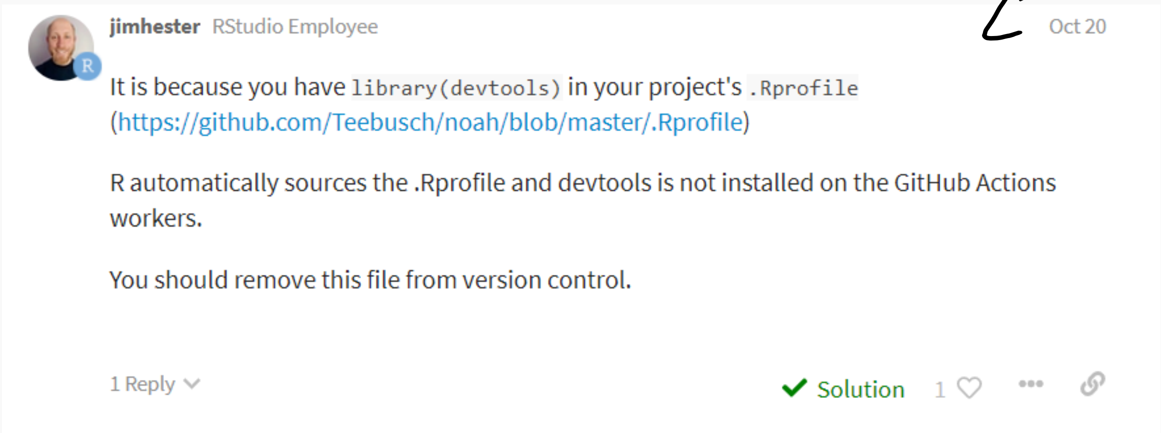
Jobs & Gigs

Appsilon - Community Manager

NoviSci - Lead Statistician

Rasa - Senior Machine Learning Engineer

Same day response by devtools maintainer Jim Hester on RStudio Community!



jimhester RStudio Employee Oct 20

It is because you have `library(devtools)` in your project's `.Rprofile` (https://github.com/Teebusch/noah/blob/master/.Rprofile)

R automatically sources the .Rprofile and devtools is not installed on the GitHub Actions workers.

You should remove this file from version control.

ALL TOGETHER NOW...

```r
install.packages(c("devtools", "usethis"))

usethis::create_package("mycoolpackage") # edit DESCRIPTION!
usethis::use_git()
usethis::use_github()
usethis::use_r("foo")        # create a file, write function(s)
devtools::load_all(".")      # load all functions (Ctrl + Shift + L)
usethis::use_test()          # write test
devtools::test()             # run all tests
devtools::check()            # run CRAN check (Ctrl + Shift + E)
devtools::document()         # build documentation (Ctrl + Shift + D)
usethis::use_package("bar")  # declare dependencies, then use pck::fun()

usethis::use_github_action_check_standard()  # set up CI
usethis::use_coverage("codecov")             # set up test coverage

usethis::use_readme_rmd()              # edit readme, use R code
devtools::build_readme()               # convert Rmd to md
usethis::use_mit_license("Your Name")  # pick a license
usethis::use_pkgdown()                 # build package website
usethis::use_github_action("pkgdown")  # deploy site to Github pages

devtools::install("path/package")      # install from GitHub
devtools::install_github("user/repo")  # install from local
install.packages("mycoolpackage")      # install from CRAN
```

Main Packages

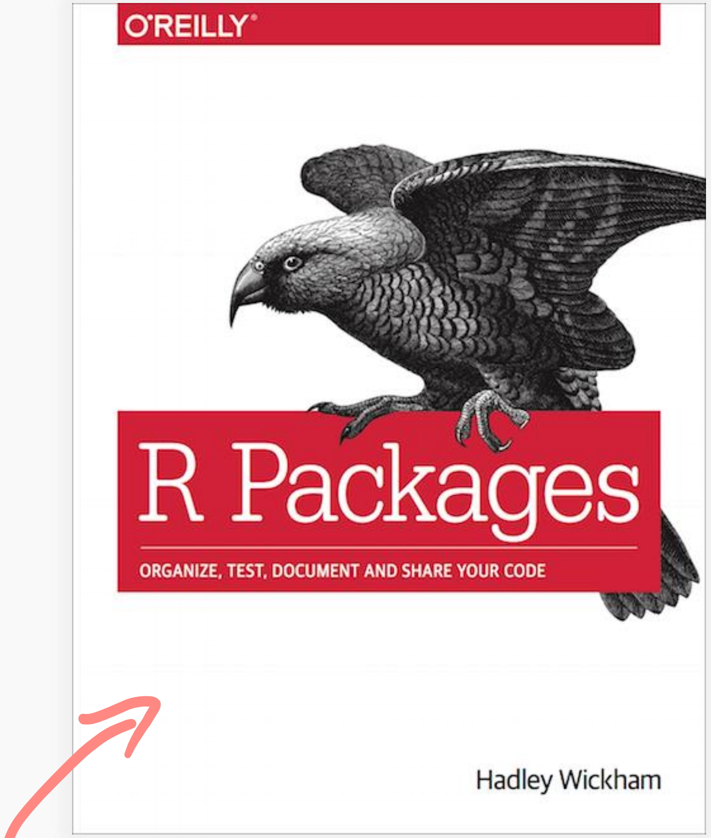
- devtools
- usethis
- roxygen2
- testthat
- pckdown

Supporting Packages

- reprex
- profvis

Free Book

r-pkgs.org



R4DS Book Club
youtu.be/FR6NsbkYhcw

Talk

Zen and the Art of R package development

youtu.be/d6JPRyp0bzY

Blog Post

Your first R package in 1 hour

pipinghotdata.com/posts/2020-10-25-your-first-r-package-in-1-hour/

Thank You!

Tobias Busch

@tobilottii

tobiasbusch.xyz

noah

teebusch.github.io/noah