

Package Development:

The Rest of the Owl

GitHub repo: pos.it/pkg_dev_24

Wifi: Posit Conf 2024 | conf2024

Discord: #workshop-pkg-dev

How to draw an owl

1.

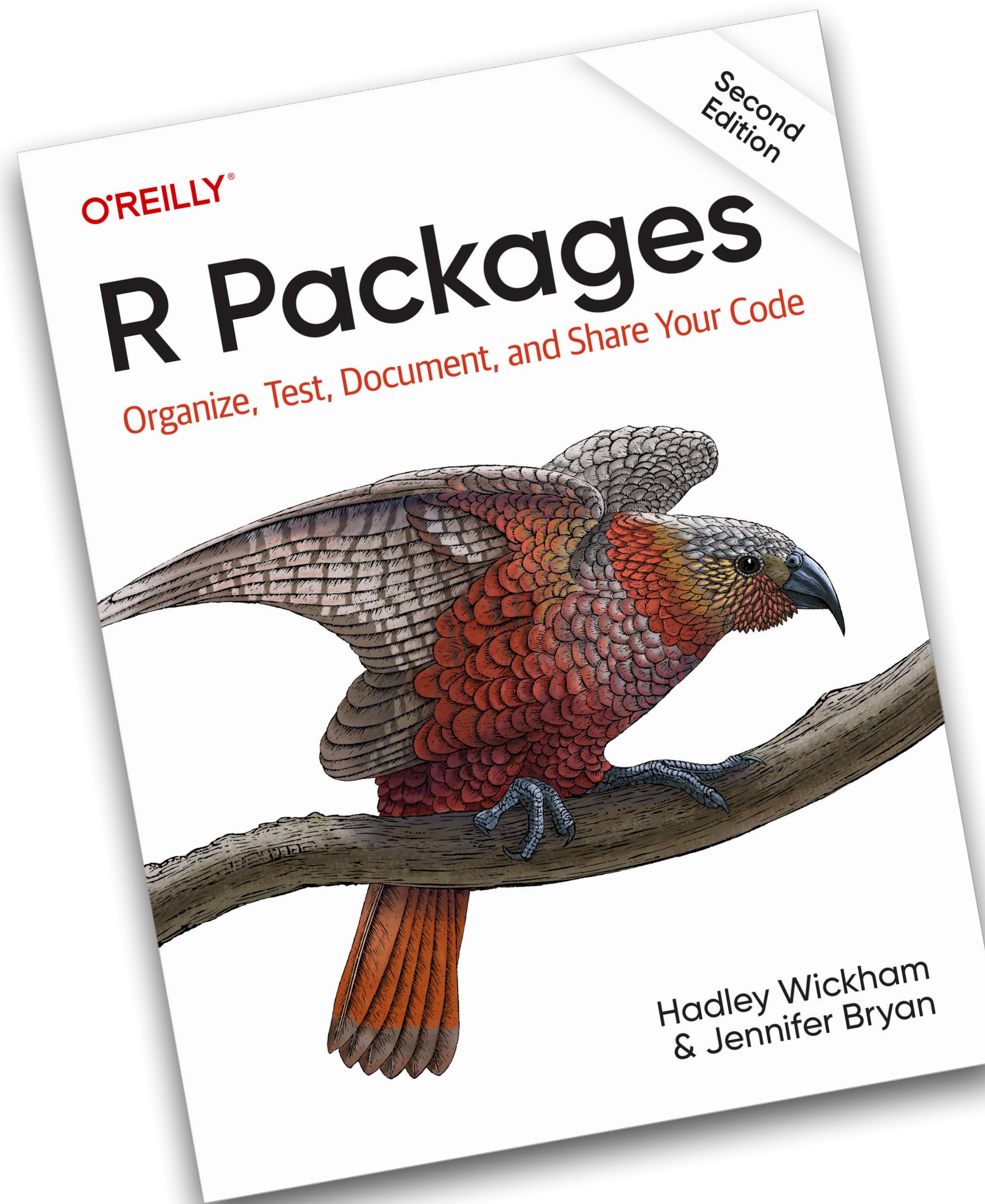


2.



1. Draw some circles

2. Draw the rest of the @#\$% owl



<https://r-pkgs.org/>

Introductions

- Jenny Bryan, instructor
- Tomasz Kalinowski, TA
- Kevin Ushey, TA
- Introduce yourself to your neighbors.
 - Did you bring a package with you today? If so, what are you most hoping to learn today?
 - Are you just starting out? What do you feel the most confused or curious about?

Outline

Kick the tires of your setup

Core workflows

Let me GitHub search that for you

Testing

Documentation

GitHub Actions

Beautiful UI with cli

Topic by popular request?

Kicking the tires

Prework

```
install.packages(c("devtools", "knitr"))  
# devtools also installs other important packages, such as
```

```
# usethis
```

```
# roxygen2
```

```
# testthat
```

```
# pkgdown
```

```
install.packages("pak")
```

```
# extremely useful for package installation, as a developer
```

```
# pak is the successor to remotes
```

```
library(devtools)  
#> Loading required package: usethis
```



From this point on, I assume you're attaching devtools, and therefore usethis, in today's interactive work.

Don't expect devtools::this() or usethis::that().

Prework

```
# you might want this in .Rprofile  
# to make devtools always available in all interactive sessions  
if (interactive()) {  
  suppressMessages(require(devtools))  
}  
  
# helps you get ^^^ the code above ^^^ into your .Rprofile  
use_devtools()  
# won't take affect until you restart R
```

Prework

```
options(  
  usethis.description = list(  
    "Authors@R" = utils::person(  
      "Jennifer", "Bryan",  
      email = "jenny@posit.co",  
      role = c("aut", "cre"),  
      comment = c(ORCID = "0000-0002-6983-2759"))  
,  
  License = "MIT + file LICENSE"  
,  
  usethis.destdir = "~/rrr",  
  usethis.overwrite = TRUE  
)
```

Customize your own defaults for DESCRIPTION.

Tell usethis where to put new projects.

Rely on Git to help you review (and undo?) unwanted changes.

Prework

```
# package development "situation report"  
dev_sitrep()
```

```
# git/github "situation report"  
git_sitrep()
```

(Quickly) get a source package

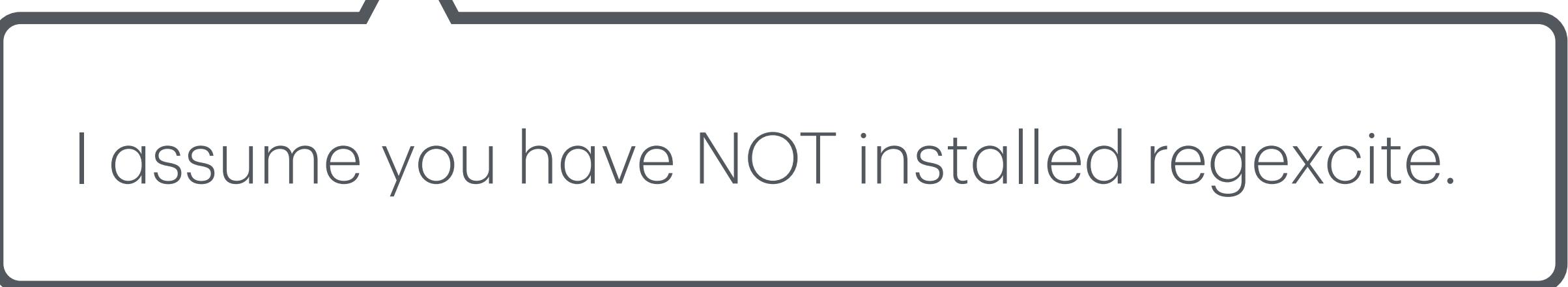
- Recommended workflow:
 - `create_from_github("jennybc/regexcite")`
 - `create_from_github("jennybc/regexcite", destdir = "some/folder")`
 - Does "fork and clone" of a Git repo.
- Alternative workflow:
 - `use_course("jennybc/regexcite")`
 - Does a one-time download and unpacking of a ZIP archive.

Core workflows

Let's run a little code from `regexcite`'s README, as if you were going to do some development on it.

```
(x ← "alfa,bravo,charlie,delta")
#> [1] "alfa,bravo,charlie,delta"
```

```
str_split_one(x, pattern = ",")  
#> Error in str_split_one(x, pattern = ","): could not find function "str_split_one"
```

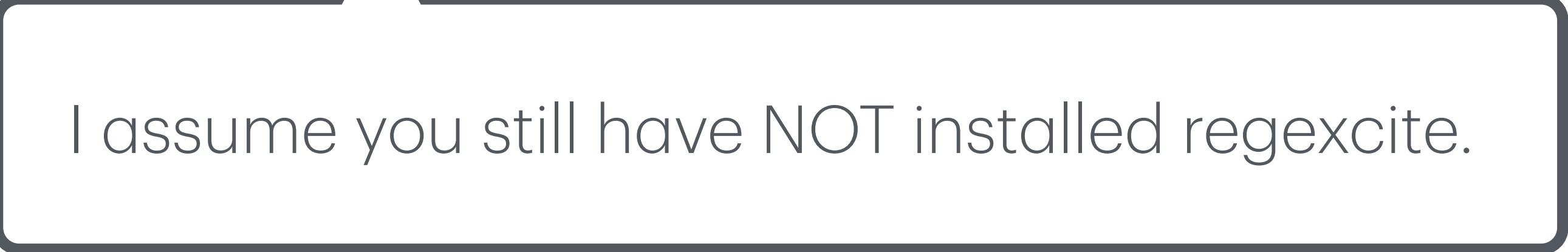


I assume you have NOT installed `regexcite`.

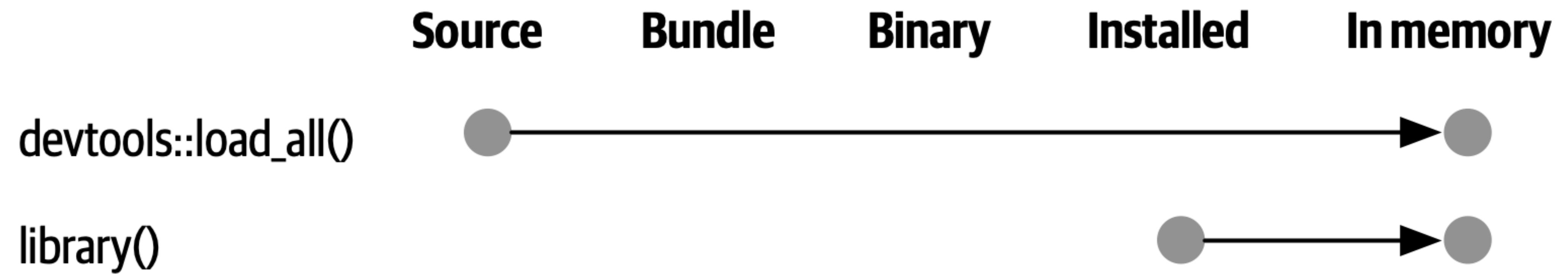
load_all()

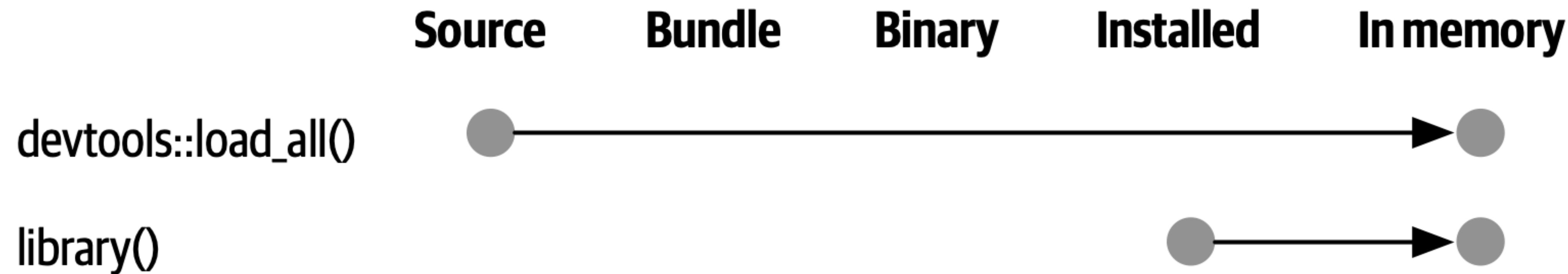
Let's run a little code from `regexcite`'s README, as if you were going to do some development on it.

```
load_all()  
#> i Loading regexcite  
  
(x ← "alfa,bravo,charlie,delta")  
#> [1] "alfa,bravo,charlie,delta"  
  
str_split_one(x, pattern = ",")  
#> [1] "alfa"    "bravo"   "charlie" "delta"
```

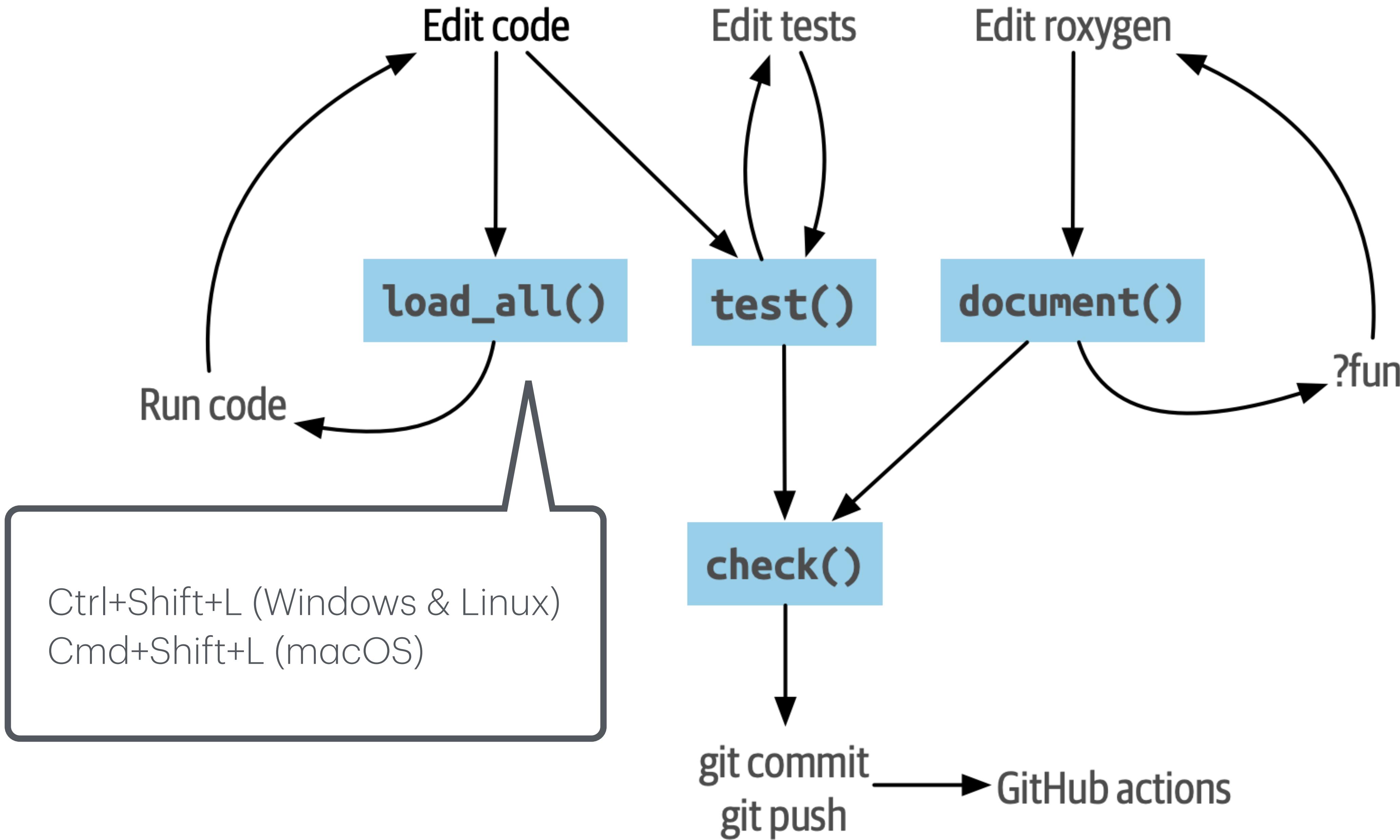


I assume you still have NOT installed `regexcite`.





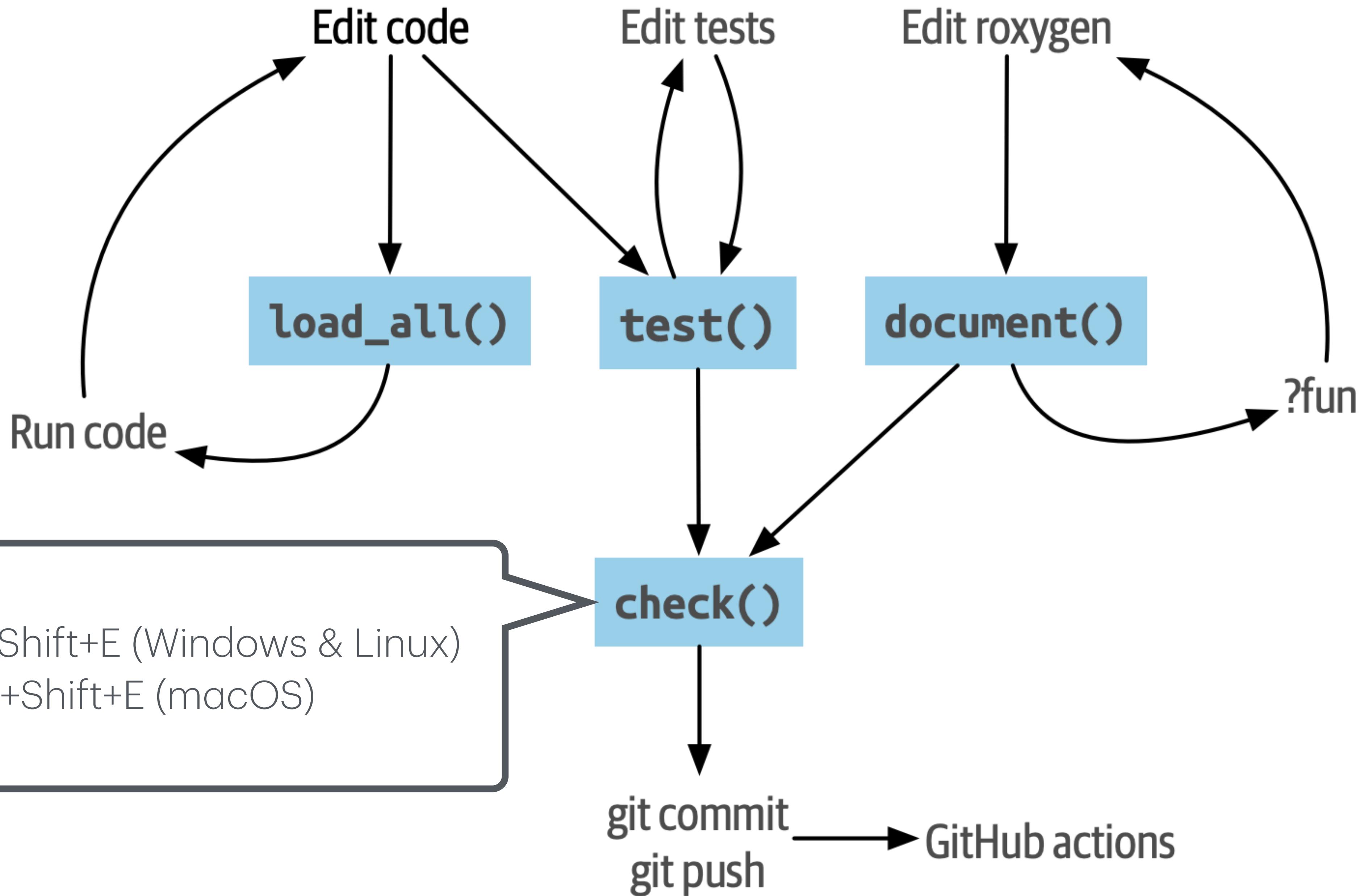
- Simulates building, installing, and attaching your package
- Makes all of the functions in your package available to use
- Makes anything you've imported available to use
- Allows fast iteration of editing and test-driving your functions



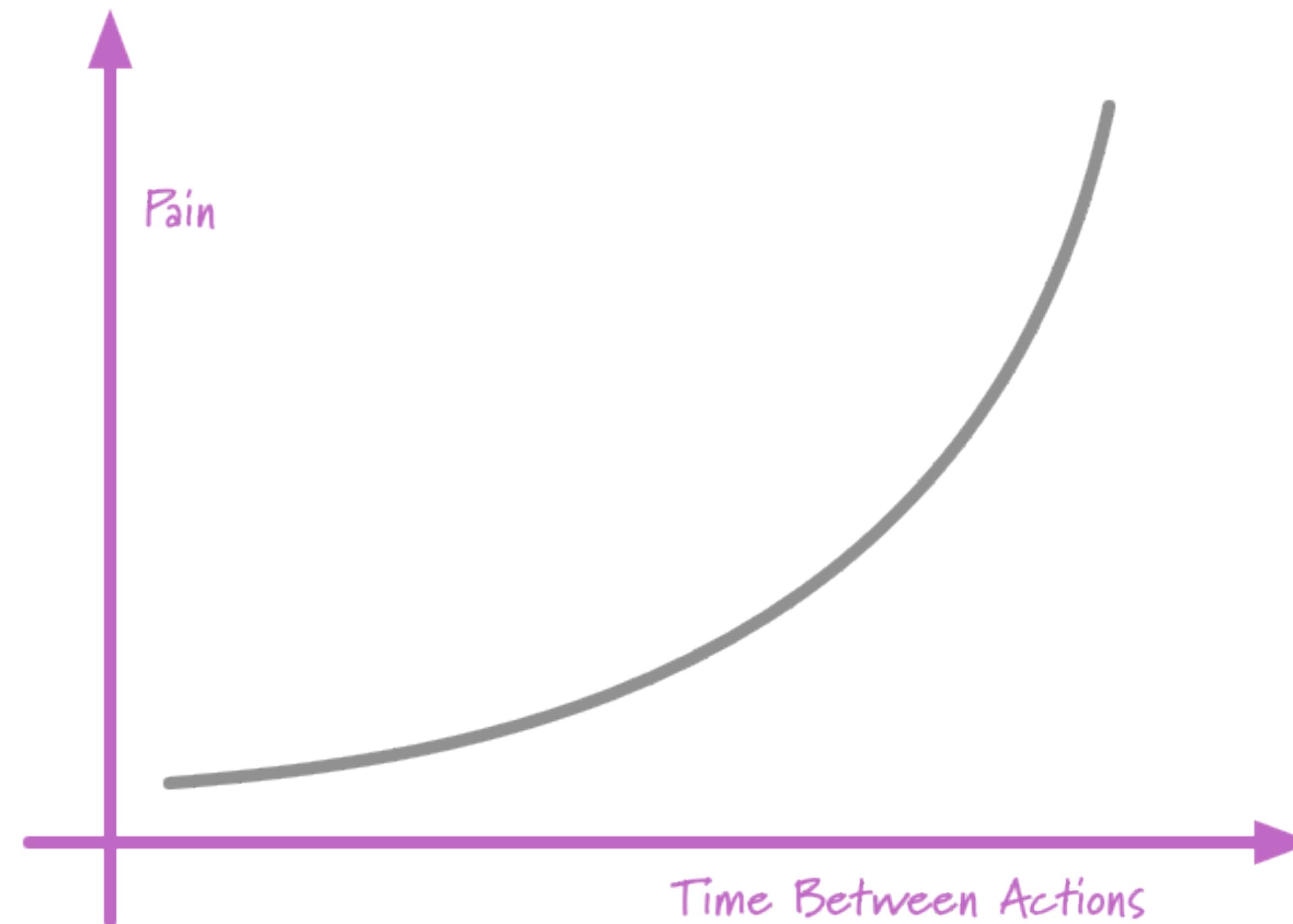
check()

```
check()

#> == Documenting ==
#> i Installed roxygen2 version (7.3.2) doesn't match required (7.1.2)
#> x `check()` will not re-document this package
#> == Building ==
...
#> — R CMD build —
...
#>   — building 'regexcite_0.0.0.9000.tar.gz'
#>
#> == Checking ==
...
#> — R CMD check —
...
#> — R CMD check results ————— regexcite 0.0.0.9000 —————
#> Duration: 11.2s
#>
#> 0 errors ✓ | 0 warnings ✓ | 0 notes ✓
```

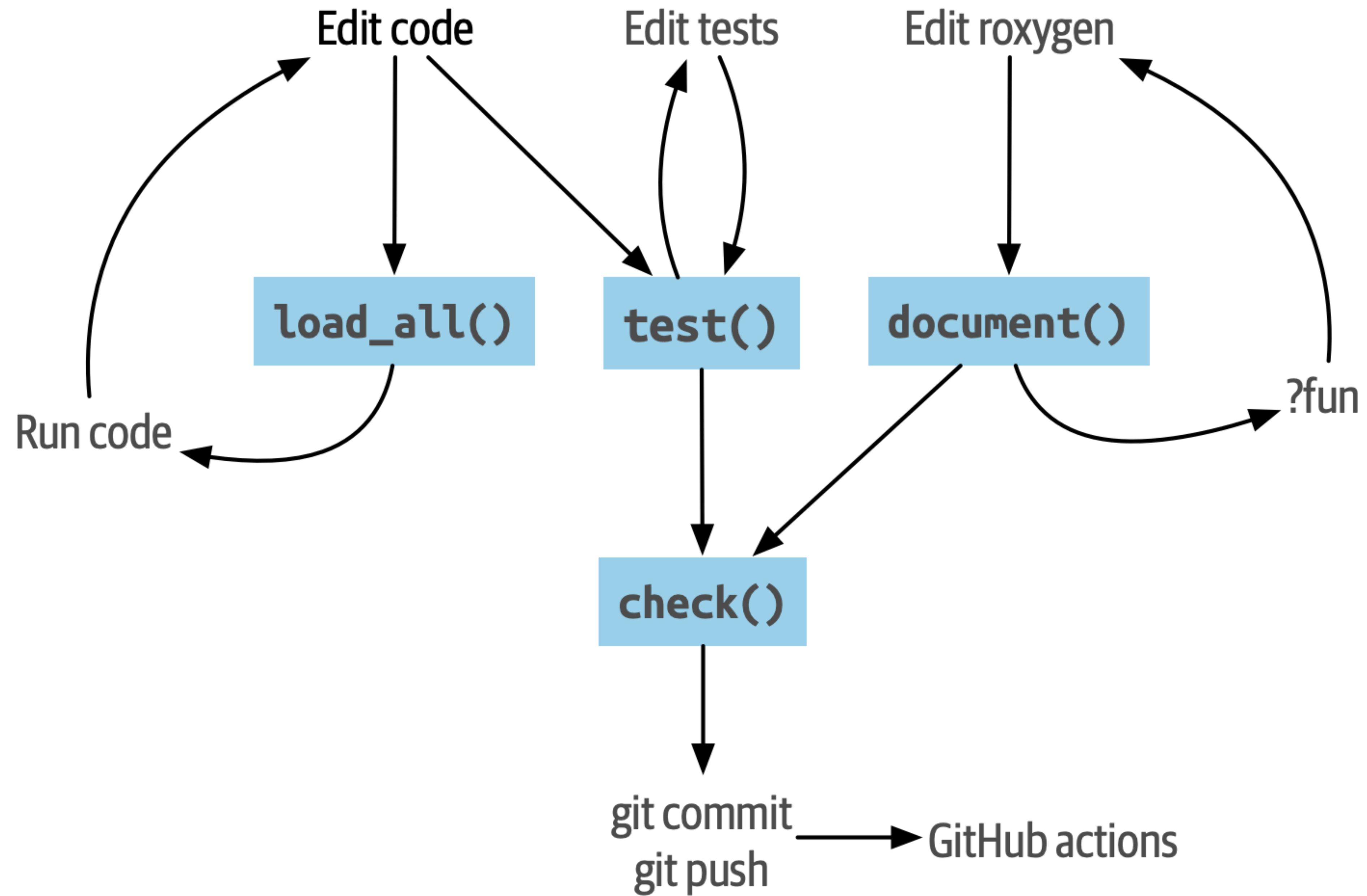


"If it hurts, do it more often."



"If it hurts, do it more often."

- If `check()` goes from passing to failing after you've tinkered with one function or touched 10 lines of code, it's clear what to troubleshoot.
- If you've added 15 functions that all call each other and touched 100s of lines of code, bless your heart.
- Better to learn of a regrettable choice before you've built upon it for days or weeks.



Your turn

- Grab the source of some other package and take it out for a test drive.
 - `create_from_github("OWNER/REPO")` or `use_course("OWNER/REPO")`
 - Does it pass R CMD check locally for you? `pak :: local_install_dev_deps()` is very useful!
- Ideas
 - r-lib/usethis
 - r-lib/rematch2
 - r-lib/sessioninfo
 - r-lib/desc
 - r-lib/withr

GitHub Code Search



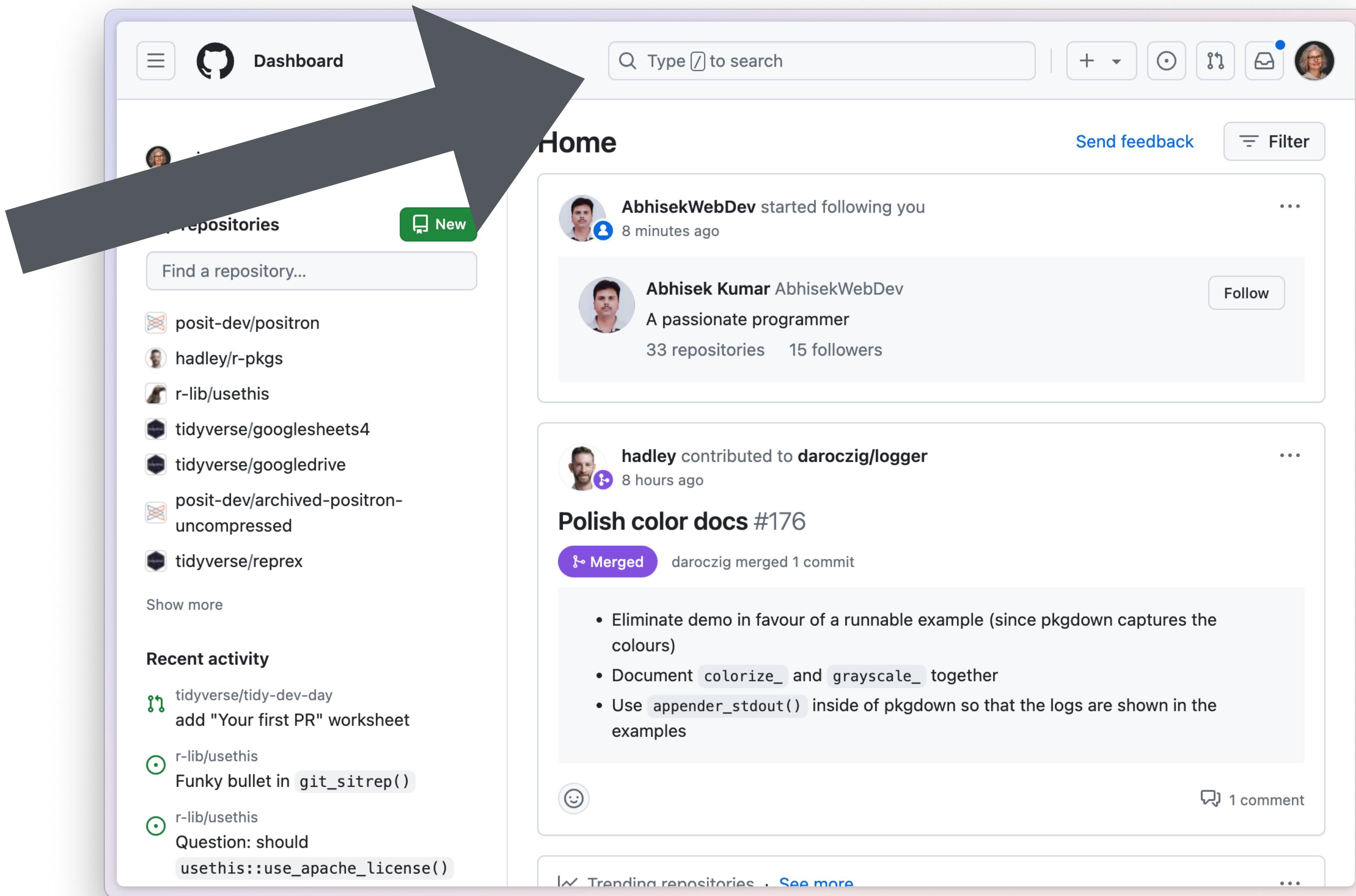
Let Me Google That For You...

Google Search

I'm Feeling Lucky

Type a question, click a button.

Let me GitHub search that for you



I merged a pull request (PR) to usethis that updated its GitHub Actions (GHA) workflows. I noticed that recent changes in r-lib/actions actually broke 2 tests in usethis. But why were they only failing on ubuntu-latest (release)?

The screenshot shows the GitHub Actions page for the repository `r-lib / usethis`. The navigation bar includes links for Code, Issues (65), Pull requests (5), Discussions, Actions (highlighted in orange), Security, and Insights. Below the navigation, a link points to `R-CMD-check.yaml`. A prominent red error message says **Use latest GHA workflows #2061**.

The main area displays a **Summary** card. It lists several jobs: macos-latest (release), windows-latest (release), windows-latest (oldrel-4), ubuntu-latest (devel), ubuntu-latest (release) (which is marked with a red X), ubuntu-latest (oldrel-1), ubuntu-latest (oldrel-2), ubuntu-latest (oldrel-3), and ubuntu-latest (oldrel-4). To the right of the summary, a detailed view of the `R-CMD-check.yaml` workflow is shown. It was triggered via a pull request 20 hours ago by `gaborcsardi` (PR #2033). The workflow is defined with the command `gaborcsardi:fix/gha-latest`. The workflow details show a matrix named `R-CMD-check` with 9 jobs completed, all of which have passed.

ubuntu-latest (release)

```
== Failed tests =====
— Failure ('test-github-actions.R:106:3'): use_github_action() accepts a name —
yml$name (`actual`) not identical to "R-CMD-check" (`expected`).

`actual`: "R-CMD-check.yaml"
`expected`: "R-CMD-check"

— Failure ('test-github-actions.R:132:3'): use_tidy_github_actions() configures the full check and pr commands —
yml$name (`actual`) not identical to "R-CMD-check" (`expected`).

`actual`: "R-CMD-check.yaml"
`expected`: "R-CMD-check"

[ FAIL 2 | WARN 0 | SKIP 0 | PASS 936 ]
```

macOS-latest (release)

```
> test_check("usethis")
Starting 2 test processes
[ FAIL 0 | WARN 0 | SKIP 31 | PASS 838 ]

= Skipped tests (31) =
• No Git user configured (31): 'test-github-actions.R:2:3',
  'test-github-actions.R:29:3', 'test-github-actions.R:45:3',
  'test-github-actions.R:60:3', 'test-github-actions.R:90:3',
  'test-github-actions.R:117:3', ... , 'test-utils-github.R:122:3'
```

Why were they only failing on ubuntu-latest (release)?

Because these tests were only **running** on ubuntu-latest (release).

But why is that?

```
test_that("use_github_action() accepts a name", {  
    skip_if_no_git_user()  
  
    ...  
})
```

Ah, it must be because no Git user is configured in those GHA jobs.

But I usually set that up! I must have goofed up and allowed those lines to get deleted.*

Hmmm, what's the usual way to configure the Git user in a GHA workflow?

* This was, in fact, not true. But I'm just taking you on my journey.

What's the usual way to configure the Git user in a GHA workflow?

- Google it?
- Ask ChatGPT or Co-Pilot?
- Read documentation for GitHub Actions?
- Read a git man page?
- Post on social media with #rstats hashtag?
- Ask in Slack?
- Ask my local expert?
- **Search GitHub to find working examples of exactly the thing I want to do.**

user.name owner:r-lib path:*.yaml

The screenshot shows a GitHub search interface with the query `user.name owner:r-lib path:*.yaml`. The results page displays 25 files found in the `r-lib` repository across three commits.

Filter by:

- Code**: 25
- Repositories**: 1
- Issues**: 26
- Pull requests**: 3
- Discussions**: 0
- Users**: 47
- More**

Repositories:

- `r-lib/actions`
- `r-lib/devtools`
- `r-lib/ellipsis`
- `r-lib/memoise`
- `r-lib/styler`
- + More repositories...**

Paths:

- `.github/workflows/`
- `examples/`
- + More directories...**

Results:

- `r-lib/actions · examples/document.yaml`**
YAML · v2-branch
38
39 - name: Commit and push changes
40 run: |
41 git config --local user.name "\$GITHUB_ACTOR"
42 git config --local user.email "\$GITHUB_ACTOR@users.noreply.github.com"
43 git add man/*
44 git commit -m "Update documentation" || echo "No changes to commit"
- `r-lib/actions · examples/style.yaml`**
YAML · v2-branch
67 if FILES_TO_COMMIT=\$(git diff-index --name-only \${{ github.sha }} \|
68 egrep --ignore-case '\.(R|qR)md|Rmarkdown|Rnw|Rprofile\$')
69 then
70 git config --local user.name "\$GITHUB_ACTOR"
71 git config --local user.email "\$GITHUB_ACTOR@users.noreply.github.com"
72 git commit \${FILES_TO_COMMIT[*]} -m "Style code (GHA)"
73 git pull --ff-only
- `r-lib/actions · examples/render-rmarkdown.yaml`**
YAML · v2-branch
31 run: |
32 RMD_PATH=\$(git diff --name-only \${{ github.event.before }} \${{ github.sha }} | grep
33 '[^.]Rmd\$'))
34 Rscript -e 'for (f in commandArgs(TRUE)) if (file.exists(f)) rmarkdown::render(f)'
35 \${RMD_PATH[*]}
36 git config --local user.name "\$GITHUB_ACTOR"
37 git config --local user.email "\$GITHUB_ACTOR@users.noreply.github.com"

user.name owner:r-lib path:*.yaml

I know that you do something like:
git config user.name

The screenshot shows a GitHub repository interface with a sidebar and three code snippets in the main area.

Sidebar:

- Users: 47
- More
- Repositories
 - r-lib/actions
 - r-lib/devtools
 - r-lib/ellipsis
 - r-lib/memoise
 - r-lib/styler
- More repositories...

Main Area:

- r-lib/actions · examples/document.yaml**
YAML · v2-branch
74 ms) in r-lib X

```
- name: Commit and push changes
  run: |
    git config --local user.name "$GITHUB_ACTOR"
    git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"
    git add man/* NAMESPACE DESCRIPTION
    git commit -m "Update documentation" || echo "No changes to commit"
```

- r-lib/actions · examples/style.yaml**
YAML · v2-branch
67 if FILES_TO_COMMIT=\$(git diff-index --name-only \${github.sha}) \
68 | egrep --ignore-case '\.(R|qR)md|Rmarkdown|Rnw|Rprofile\$')
69 then
70 git config --local user.name "\$GITHUB_ACTOR"
71 git config --local user.email "\$GITHUB_ACTOR@users.noreply.github.com"
72 git commit \${FILES_TO_COMMIT[*]} -m "Style code (GHA)"
73 git pull --ff-only
- r-lib/actions · examples/render-rmarkdown.yaml**
YAML · v2-branch
31 run: |
32 RMD_PATH=\$(git diff --name-only \${github.event.before} \${github.sha} | grep
33 '[.]Rmd\$')
34 Rscript -e 'for (f in commandArgs(TRUE)) if (file.exists(f)) rmarkdown::render(f)'
35 \${RMD_PATH[*]}
36 git config --local user.name "\$GITHUB_ACTOR"
37 git config --local user.email "\$GITHUB_ACTOR@users.noreply.github.com"

user.name owner:r-lib path:*.yaml

The screenshot shows a GitHub repository interface. On the left, there's a sidebar with navigation links like 'Code', 'Repositories', 'Issues', 'Pull requests', 'Discussions', 'Users' (47), and 'More'. Below that is a 'Repositories' section listing 'r-lib/actions', 'r-lib/devtools', 'r-lib/ellipsis', 'r-lib/memoise', and 'r-lib/styler'. Under 'Paths', it shows '.github/workflows/' and 'examples/'. The main area displays three YAML files:

- r-lib/actions · examples/style.yaml**

```
67      if FILES_TO_COMMIT=($(git diff-index --name-only ${github.sha} )\n68          | egrep --ignore-case '\.(R|qR)md|Rmarkdown|Rnw|Rprofile$'))\n69      then\n70          git config --local user.name "$GITHUB_ACTOR"\n71          git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"\n72          git commit ${FILES_TO_COMMIT[*]} -m "Style code (GHA)"\n73          git pull --ff-only
```
- r-lib/actions · examples/render-rmarkdown.yaml**

```
31      run: |\n32          RMD_PATH=($(git diff --name-only ${github.event.before} ${github.sha} ) | grep\n33              '[.]Rmd$'))\n34          Rscript -e 'for (f in commandArgs(TRUE)) if (file.exists(f)) rmarkdown::render(f)\n35              ${RMD_PATH[*]})\n36          git config --local user.name "$GITHUB_ACTOR"\n37          git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"
```

A large callout box highlights the text: "I know I do this in more than one package (and I trust my colleagues too)."

* owner seems to be an alias for the ?more correct? user

user.name owner:r-lib path:*.yaml

The screenshot shows a GitHub search interface with the query `user.name owner:r-lib path:*.yaml`. The results are filtered by code, showing 25 files. Three files are visible in the list:

- r-lib/actions · examples/document.yaml**
A snippet of the file shows configuration for committing changes:

```
38
39   - name: Commit and push changes
40     run: |
41       git config --local user.name "GITHUB_ACTOR"
42       git config --local user.email "GITHUB_ACTOR@users.noreply.github.com"
43       git add man/* NAMESPACE DESCRIPTION
44       git commit -m "Update documentation" || echo "No changes to commit"
```
- r-lib/actions · examples/style.yaml**
A snippet of the file shows configuration for styling code commits:

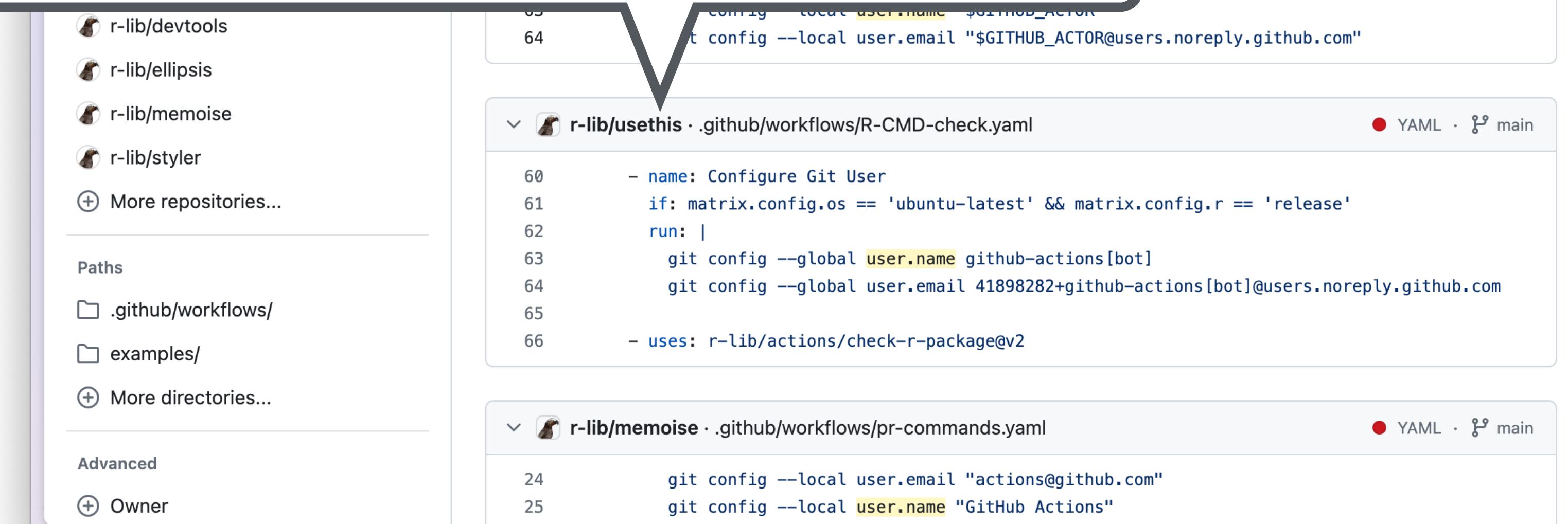
```
67       if FILES_TO_COMMIT=$(git diff-index --name-only ${github.sha} \
68         | egrep --ignore-case '\.(R|qR)md|Rmarkdown|Rnw|Rprofile$')
69       then
70         git config --local user.name "$GITHUB_ACTOR"
71         git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"
72         git commit ${FILES_TO_COMMIT[*]} -m "Style code (GHA)"
73         git pull --ff-only
```
- r-lib/actions · examples/render-rmarkdown.yaml**
A snippet of the file shows configuration for rendering R Markdown files:

```
31       run: |
32         RMD_PATH=$(git diff --name-only ${github.event.before} ${github.sha} | grep
33           '[.]Rmd$')
34         Rscript -e 'for (f in commandArgs(TRUE)) if (file.exists(f)) rmarkdown::render(f)'
35           ${RMD_PATH[*]}
36         git config --local user.name "$GITHUB_ACTOR"
37         git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"
```

A callout box highlights the first file's configuration for committing changes.

I (now) know I only want to see GHA config files. My first search without this had too many off-target hits.

One of the hits is in usethis, because I'm an idiot and the code is still there and only fires on ubuntu-lates (release), just like past Jenny set it up long ago. It's all coming back now!



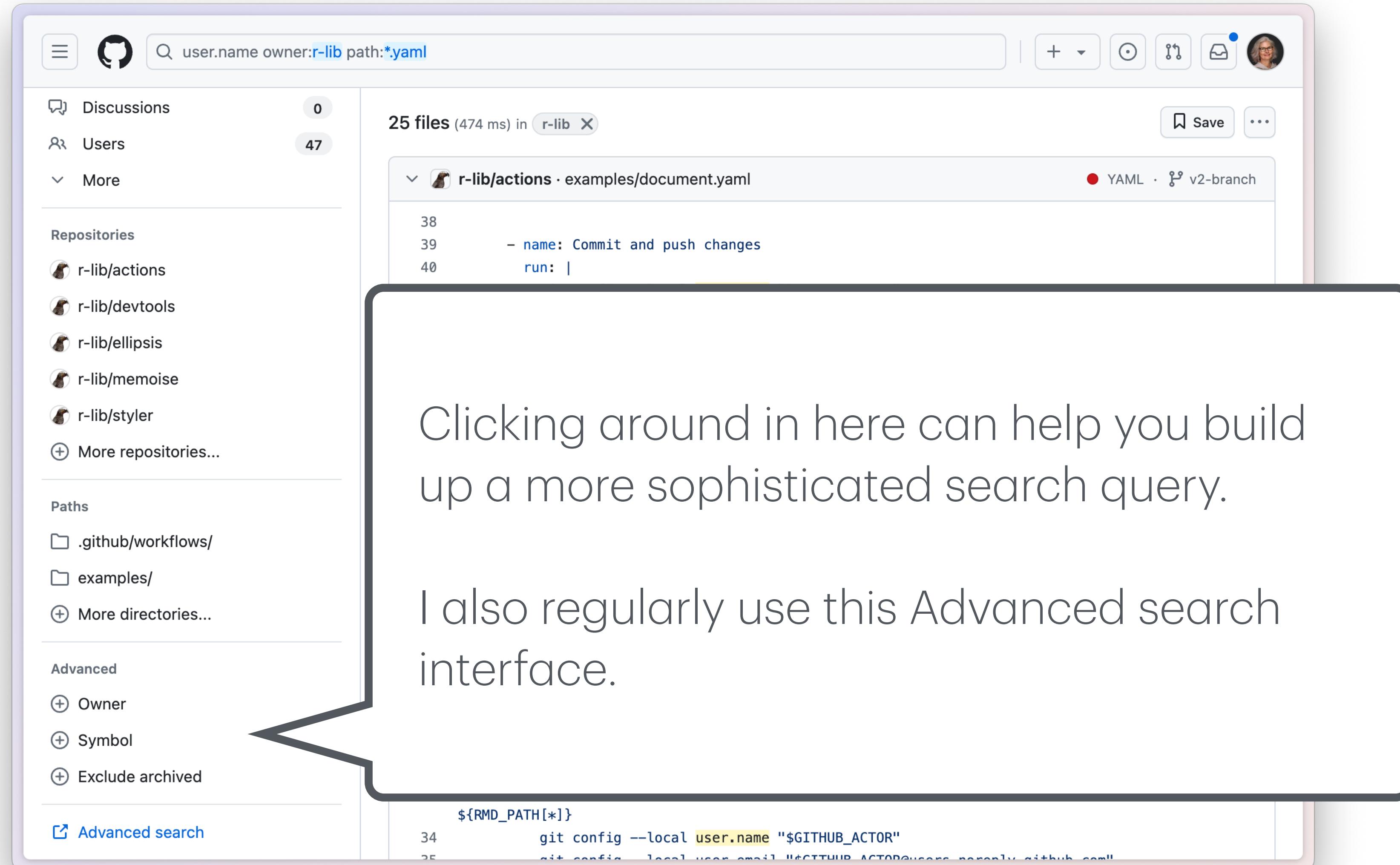
The screenshot shows a GitHub repository interface. On the left, a sidebar lists repositories: r-lib/devtools, r-lib/ellipsis, r-lib/memoise, r-lib/styler, More repositories..., Paths (.github/workflows/, examples/), More directories..., Advanced, and Owner. The main area displays three YAML configuration files:

- r-lib/usethis · .github/workflows/R-CMD-check.yaml**:
A snippet from line 64 shows:

```
git config --local user.name "$GITHUB_ACTOR"
git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"
```

A large black arrow points from this line to the corresponding line in the r-lib/memoise file.
- r-lib/memoise · .github/workflows/pr-commands.yaml**:
Lines 24 and 25 show:

```
git config --local user.email "actions@github.com"
git config --local user.name "GitHub Actions"
```



A screenshot of a GitHub search interface. The search bar at the top contains the query `user.name owner:r-lib path:*.yaml`. The results show 25 files from the repository `r-lib`. A specific file, `examples/document.yaml`, is selected, showing YAML code. A callout bubble points to the sidebar on the left, which includes sections for Discussions, Users, More, Repositories, Paths, and Advanced search.

25 files (474 ms) in `r-lib`

`r-lib/actions · examples/document.yaml`

YAML · v2-branch

```
38
39   - name: Commit and push changes
40     run: |
```

Clicking around in here can help you build up a more sophisticated search query.

I also regularly use this Advanced search interface.

Advanced search

```
34   git config --local user.name "$GITHUB_ACTOR"
35   git config --local user.email "$GITHUB_ACTOR@users.noreply.github.com"
```

GitHub code search pro tips

- `user:cran` is super powerful for searching across the source code of every package on CRAN
- I am biased, but I often use `user:tidyverse` or `user:r-lib` to focus on solutions by folks who generally approach R and package development the way I do.
- GitHub search is a tool where consulting the docs regularly really pays off:
 - Understanding GitHub Code Search syntax
- `repo:wch/r-source` narrows the search to a GitHub mirror of the R source
- Other qualifiers I use a lot:
 - `path`
 - `language`
- A qualifier that is newly relevant for R: `symbol`

<https://github.com/cran>

user:cran

The screenshot shows the GitHub repository page for the user 'cran'. The page title is 'cran' and it is described as an 'Unofficial read-only mirror of all CRAN R packages'. It has 404 followers and a link to its documentation at <https://docs.r-hub.io/#cran-source...>. A 'Follow' button is visible.

Popular repositories:

- glmnet** (Public)
! This is a read-only mirror of the CRAN R package repository.
glmnet — Lasso and Elastic-Net Regularized Generalized Linear Models. Homepage: <https://glmnet.stanford.edu>
C++ 71 85
- AppliedPredictiveModeling** (Public)
! This is a read-only mirror of the CRAN R package repository.
AppliedPredictiveModeling — Functions and Data Sets for 'Applied Predictive Modeling'. Homepage: <http://appliedpredictivemodeling.com/>
R 58 65
- bnlearn** (Public)
! This is a read-only mirror of the CRAN R package repository.
bnlearn — Bayesian Network Structure Learning, Parameter Learning and Inference. Homepage: <https://www.bnlearn.com/>
C 53 38
- WGCNA** (Public)
! This is a read-only mirror of the CRAN R package repository.
WGCNA — Weighted Correlation Network Analysis. Homepage: <http://horvath.genetics.ucla.edu/html/CoexpressionNetwork/Rpackages/WGCNA/>
R 49 57

People: META CRAN

Top languages: R C++ C HTML Fortran

Report abuse

<https://github.com/wch/r-source>

repo:wch/r-source

The screenshot shows the GitHub repository page for 'r-source' owned by 'wch'. The repository is public and has 307 branches and 0 tags. The 'Code' tab is selected. A list of recent commits is displayed, all made by 'ripley'. The commits are:

- report clang warning of undefined behaviour (49d6f07 · 5 hours ago)
- NEWS for c86950. (last week)
- Updated. (4 months ago)
- implement visibility control on macOS (last week)
- Remove note in po/README about the instr... (last year)
- don't use DESTDIR when building (2 weeks ago)
- report clang warning of undefined behaviour (5 hours ago)
- Updated. (5 days ago)
- LLVM only allows 16-bit fields in file version. (last year)
- use exact plain text form from www.fsf.org (...) (19 years ago)
- (source / installed R) (9 years ago)
- Use https for CRAN URLs. (8 years ago)

The right sidebar contains sections for 'About', 'Releases', and 'Packages'. The 'About' section describes it as a read-only mirror of R source code from <https://svn.r-project.org/R/>, updated hourly. It also links to the GitHub wiki and provides statistics: 90 watching, 311 forks, and 1.1k stars.

Your turn

You can find some GitHub code search challenges in the workshop repo:
<https://github.com/posit-conf-2024/pkg-dev>

Feel free to follow your heart and come up with your own, based on questions you have.

Big Idea	Featured implementation
Become fluid at package development and installation	devtools pak
Eliminate repetitive tasks	Curate your .Rprofile
Regular health checks	dev_sitrep() git_sitrep()
Make experimentation cheap	create_from_github() create_package()
Test drive your package often	load_all()
Check your package often	check()
Study how other people do things	GitHub code search