Some more R, Git, and R Markdown and Quarto tips

MRC IEU Code Clinic
07/05/2025
Tom Palmer

Overview

- Some more R Markdown and Quarto tips
  - Seven tips for making Quarto revealjs presentations
  - Creating R Markdown and Quarto tutorial documents
  - Creating multilanguage Quarto documents (i.e., R and Stata executed code in same doc)
- Some more R tips
  - Creating a reproducible environment in R without using renv - CRAN snapshot repositories for the win
- Some more Git, GitHub Desktop (and Lazygit), and GitHub tips
  - Making suggestions on a GitHub PR
  - Amending a previous commit message in GitHub Desktop
  - Amending a previous commit in GitHub Desktop
  - Amending previous commits and commits messages in Lazygit - interactive rebase made "easy"

Some more R Markdown and Quarto tips

Seven tips for making Quarto presentations

- Quarto revealjs docs https://quarto.org/docs/presentations/revealjs/
- https://remlapmot.github.io/post/2025/quarto-revealjs-tips/

Creating R Markdown and Quarto tutorial documents

- R Markdown:
- Quarto: https://remlapmot.github.io/post/2025/quarto-conditional-content/

Creating multilanguage Quarto documents

- Technically Quarto documents may only have 1 engine
- Workaround: use {{< embed >}} shortcode to include chunks and output from documents using other engines
- https://remlapmot.github.io/post/2025/multi-engine-quarto/
- Setting up the nbstata Jupyter kernel in a uv venv
    - uv docs (including installation instructions) https://docs.astral.sh/uv/

Some more R tips

Reproducible R environments using CRAN snapshot repos without using renv/pak/pacman

- (Without using renv or similar)
- Simply note the date at the top of your script in a comment
- Then run update.packages()
- (Assume using current version of R)
- Then if you need to recreate this environment use a snapshot from https://packagemanager.posit.co/client/#/repos/cran/setup
- Nb. RSPM = PPPM = P3M

packagemanager.posit.co/client/#/repos/cran/setup?distribution=windows&snapshot=2025-03-20

posit Package Manager | Repository: cran | Packages in the cran repository

Set up your environment to install R packages from cran

Select from the following options, then follow the customized instructions below to complete your setup.

Operating System: what system are you using the packages on?

- ○ macOS
- ◉ Windows
- ○ Linux

Windows

Package Manager
compatible with W
optionally selecte

Snapshots: do you want to freeze package versions to enhance reproducibility?

- ○ No, install the most recent packages available
- ◉ Yes, always install packages from the date I choose

025

TUE	WED	THU	FRI	SAT
	1	2	3	4
7	8	9	10	11
14	15	16	17	18
21	22	23	24	25
28	29	30	31	

February 2025

SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

March 2025

SUN	MON	TUE	WED	THU	FRI	SAT
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Frozen to March

Package Manager
as of March 20, 2

Freezing to a spec
when re-installing
always match, or
from a previous p

The URL below encodes the following snapshot references:

- cran [Last update: Mar 20, 2025 12:00 AM UTC]

Environment: where are you using R?

- ◉ RStudio IDE
- ○ Posit Workbench
- ○ Posit Connect
- ○ Some other R outside of RStudio IDE

RStudio IDE

Get more info on

Repository URL:

```
https://packagemanager.posit.co/cran/2025-03-20
```

Copy

Repository URL

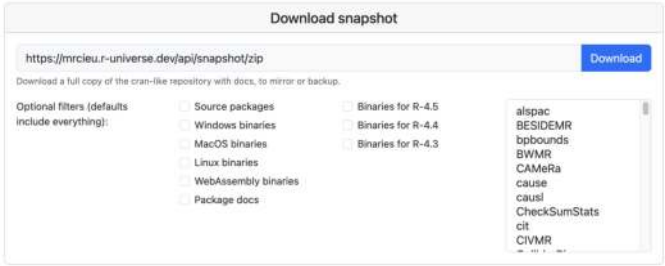
Use this URL to in
Manager directly.

- install.packages(c("package1", "package2"), repos = "https://packagemanager.posit.co/cran/2025-03-20")
- Snapshots created at midnight - so you might need tomorrow's date
- The Public Posit Package Manager now only CRAN snapshotting service (used to be MRAN as well)
- Many companies advocated this approach: RevolutionR, Cynkra (e.g., https://github.com/cynkra/cynkrathis)
- Can be used within renv

cynkra RSPM snapshots

cynkra makes use of certain RStudio Package Manager (RSPM) snapshots across projects. Usually each R Version is tied to a snapshot near it's release date. If a snapshot is considered unstable due to certain R package version clashes, additional snapshots for specific R versions can be listed.

- For a GitHub only package you can record the most recent commit and then include its installation as: remotes::install_github('MRCIEU/TwoSampleMR@6268c2c')
- Or use date to go through commit history later to find SHA
  - https://github.com/MRCIEU/TwoSampleMR/commits/
- Or you could save a copy of the repo or fork the repo on the day you installed it
- Only works for simple examples - as you need to know SHA of any hard (and soft) GitHub only dependency packages you used
- Or you can download the package from an r-universe - see apis page
  - https://mrcieu.r-universe.dev/apis



Some more Git, GitHub Desktop (and Lazygit), and GitHub tips

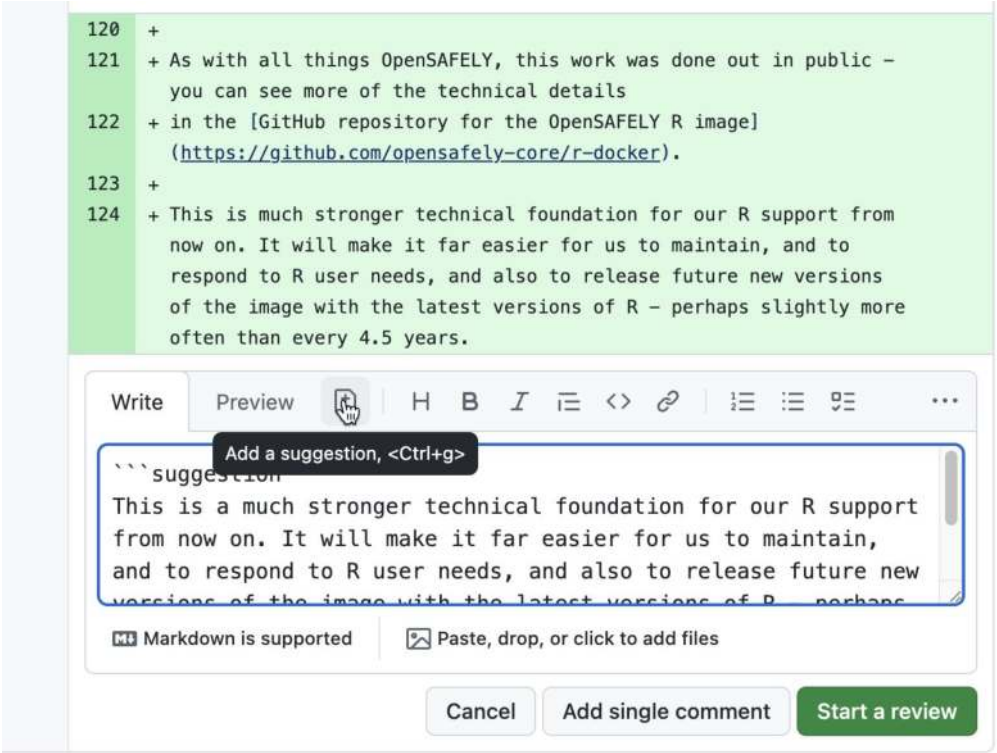
Making suggestions on a GitHub PR

Click + on LHS for a line (or Shift click for several lines)

Then button third right in popup is Add a suggestion; and has Markdown syntax

```suggestion

```



Amending previous commit message in GitHub Desktop

- https://remlapmot.github.io/post/2025/amend-commit-messages/

Interactive rebase made "easy" in Lazygit

- My Git levels of attainment
  - Beginner: branches, push and pull between local and GitHub, make PR
  - Medium: squash, reorder, resolve merge conflicts, stash, worktrees
  - Advanced: interactive rebase
- Interactive rebase at the command line is very very hard (see next slide)
  https://opensource.com/article/20/3/lazygit
- Lazygit installation instructions:
  https://github.com/jesseduffield/lazygit?tab=readme-ov-file#installation
- Launch by running lazygit in Terminal

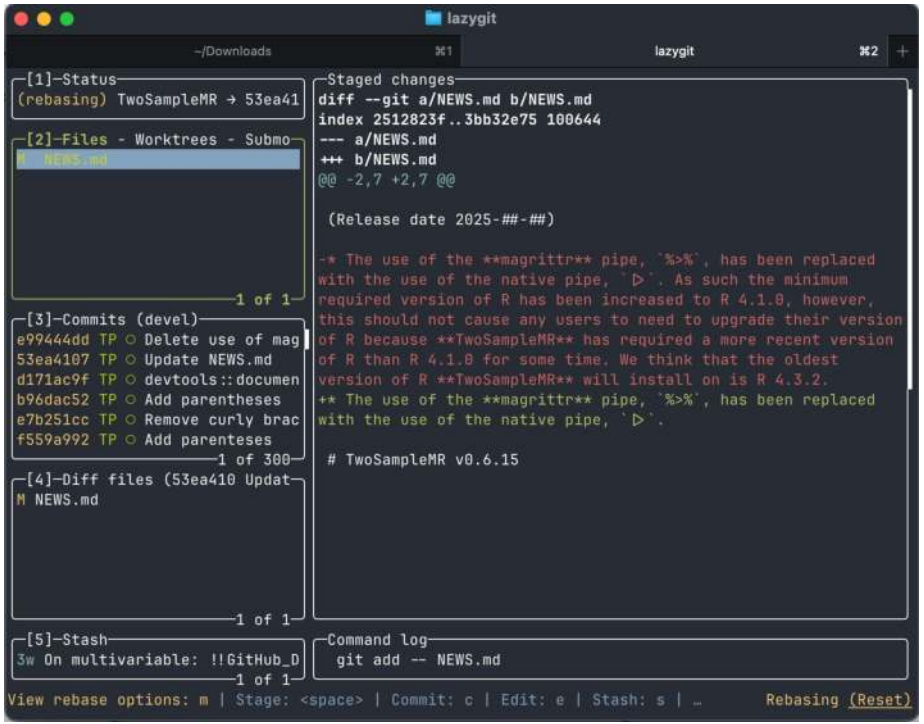
Interactive rebase at the command line is hard (don't do it)

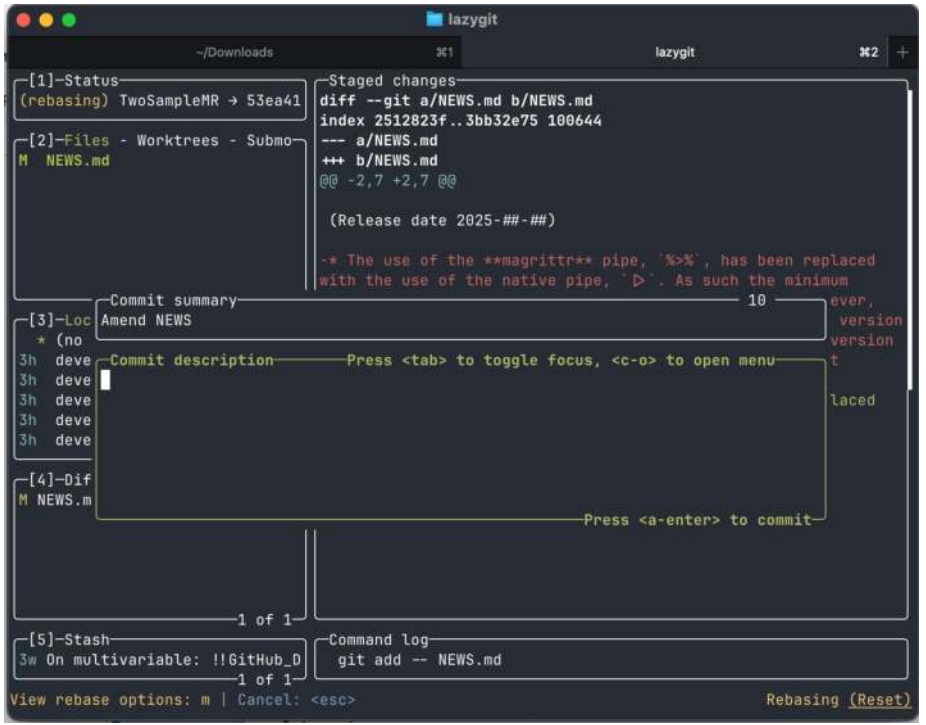
The user experience of an interactive rebase on the Git command line is a horror story that belongs in a Stephen King novel. To do something as simple as amending an old commit requires following these steps (do not let your kids read this without parental guidance):

1. Stash the changes you want to apply with **git stash**.
2. Copy the SHA of the commit you want to amend.
3. Begin the rebase with **git rebase --interactive <commit-SHA>^**.
4. (This is where the screams start...) A TODO file opens in Vim, where you'll need to find your commit and replace **pick** on the line with **edit**.
5. Save the file.
6. Unstash your changes with **git stash pop**.
7. Amend the commit with **git commit --amend**.
8. Continue the rebase with **git rebase --continue**.

Just thinking back to the days of CLI terror gives me heart palpitations.

Lazygit in action





Rewording and editing commits in Lazygit

- To edit previous commit messages in Lazygit
  - Navigate to Reflog
  - r - Reword - to edit previous commit message
  - Move to message box (<Tab>), alt + <Enter> to commit
- To edit previous commits in Lazygit
  - Move to Reflog [4]
  - Move to commit want to edit
  - e - Edit - to edit previous commit
  - Press <Enter> on commit want to edit
  - e - Edit
  - Pops you into vim - make edits - then save and quit :wq
  - Move back to top left Files pane
  - Space - stage changed file/s
  - Now continue rebase - m - rebase options - <Enter> or c - continue
  - Move to Commit pane - Escape - and should see edited commit
  - Now probably want to edit commit message of edited commit - Move to Reflog - r - Reword commit message of squashed commit

Summary

- Slightly random collection of R, Quarto, and Git tips
- I hope something was interesting/useful