

# Package: saperlipopette (via r-universe)

August 15, 2026

**Title** Create Example Git Messes

**Version** 1.0.0

**Description** Holds functions creating Git messes, that users would then solve, to follow <<https://ohshitgit.com/>>.

**License** MIT + file LICENSE

**URL** <https://docs.ropensci.org/saperlipopette/>,  
<https://github.com/ropensci-training/saperlipopette>

**BugReports** <https://github.com/ropensci-training/saperlipopette/issues>

**Depends** R (>= 4.1.0)

**Imports** brio, cli, fs, gert, parsedate, purrr, rlang, roxygen2,  
tibble, usethis, withr

**Suggests** knitr, quarto, testthat (>= 3.0.0)

**VignetteBuilder** quarto

**Config/Needs/website** rmarkdown, quarto

**Config/potools/style** explicit

**Config/ropensci/maintainer** staff

**Config/testthat/edition** 3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Config/pak/sysreqs** cmake git make libgit2-dev libuv1-dev libxml2-dev libssl-dev libx11-dev

**Repository** <https://r-multiverse-staging.r-universe.dev>

**Date/Publication** 2026-05-11 07:06:40 UTC

**RemoteUrl** <https://github.com/ropensci-training/saperlipopette>

**RemoteRef** eb09bef3fcb017b5fcd6215acf199d9f528cacfc

**RemoteSha** eb09bef3fcb017b5fcd6215acf199d9f528cacfc

Contents

|                                  |           |
|----------------------------------|-----------|
| create_all_exercises . . . . .   | 2         |
| exo_bisect . . . . .             | 3         |
| exo_blame . . . . .              | 4         |
| exo_check_editor . . . . .       | 5         |
| exo_clean_dir . . . . .          | 6         |
| exo_committed_to_main . . . . .  | 7         |
| exo_committed_to_wrong . . . . . | 8         |
| exo_conflict . . . . .           | 9         |
| exo_latest_message . . . . .     | 10        |
| exo_log_deleted_file . . . . .   | 11        |
| exo_log_deleted_line . . . . .   | 12        |
| exo_one_small_change . . . . .   | 13        |
| exo_rebase_i . . . . .           | 14        |
| exo_reset . . . . .              | 15        |
| exo_revert_file . . . . .        | 16        |
| exo_revparse . . . . .           | 17        |
| exo_split_changes . . . . .      | 18        |
| exo_time_machine . . . . .       | 19        |
| exo_undo_commit . . . . .        | 20        |
| exo_worktree . . . . .           | 21        |
| <b>Index</b>                     | <b>22</b> |

---

|                      |                                     |
|----------------------|-------------------------------------|
| create_all_exercises | Create all exercises folder at once |
|----------------------|-------------------------------------|

---

Description

But do not open them! Having all the exercises as folders within parent\_path makes it possible to resolve them one by one, if one wants to practice all that is available with the package. Run this function only if there are no exercise folder already created in parent\_path: you can use a brand-new folder for instance.

Usage

create\_all\_exercises(parent\_path)

Arguments

parent\_path      Path where to create the exercise repo

Value

The parent path

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Examples

```
parent_path <- withr::local_tempdir()
path <- create_all_exercises(parent_path = parent_path)
```

---

`exo_bisect`*"Hey I'd like to find which commit introduced a bug!"*

---

## Description

I notice a bug in my codebase. I can see the bug was not there a bunch of commits ago. Beside doing regular debugging, I can find out which commit introduced the bug by using `git bisect`. See <https://git-scm.com/docs/git-bisect> and <https://www.jimhester.com/post/2019-04-24-git-bisect/>.

## Usage

```
exo_bisect(parent_path)
```

## Arguments

`parent_path`      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

[git bisect](#).

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_bisect(parent_path = parent_path)
```

---

exo\_blame

*"I want to find who added a specific line!"*

---

## Description

I want to find who added the line `x <- x + 1` to the script and when. The tool for that is `git blame path/to/file`.

## Usage

```
exo_blame(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

`git blame`.

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_blame(parent_path = parent_path)
```

---

|                  |                                                        |
|------------------|--------------------------------------------------------|
| exo_check_editor | <i>"Hey, I don't want to get into Vim by mistake!"</i> |
|------------------|--------------------------------------------------------|

---

## Description

An exercise to check your Git's core.editor is set correctly. <https://docs.github.com/es/get-started/git-basics/associating-text-editors-with-git>

## Usage

```
exo_check_editor(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

`git config, git commit -allow-empty.`

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_check_editor(parent_path = parent_path)
```

---

`exo_clean_dir`*"Hey I'd like to remove these untracked files I created to test stuff!"*

---

### Description

If debugging for instance created now useless untracked files and directories, there's no need to remove them "manually". The tool for that is `git clean`:

- `git clean -n` for a dry run;
- `git clean -f` to run it; Add `-d` to also remove directories. See <https://git-scm.com/docs/git-clean>.

### Usage

```
exo_clean_dir(parent_path)
```

### Arguments

`parent_path`      Path where to create the exercise repo

### Value

The path to the new project

### Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

### Related Git documentation

`git clean`.

### Examples

```
parent_path <- withr::local_tempdir()
path <- exo_clean_dir(parent_path = parent_path)
```

---

```
exo_committed_to_main  "Oh shit, I accidentally committed something to main that should have  
                        been on a brand new branch!"
```

---

## Description

To go with <https://ohshitgit.com/#accidental-commit-master>

## Usage

```
exo_committed_to_main(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

[git reset --hard](#), [git branch](#), [git switch](#), [git checkout](#).

## Examples

```
parent_path <- withr::local_tempdir()  
path <- exo_committed_to_main(parent_path = parent_path)
```

---

`exo_committed_to_wrong`*"Oh shit, I accidentally committed to the wrong branch!"*

---

## Description

To go with <https://ohshitgit.com/#accidental-commit-wrong-branch>

## Usage

```
exo_committed_to_wrong(parent_path)
```

## Arguments

`parent_path`      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

[git cherry-pick](#), [git reset](#), [git switch](#), [git checkout](#).

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_committed_to_wrong(parent_path = parent_path)
```



---

`exo_conflict`*"Hey I'd like to see what merge conflicts look like!"*

---

### Description

I made some work in a feature branch and want to merge it. Meanwhile, the main branch advanced. Unfortunately someone touched the same file as I did. Now I need to fix a merge conflict!

See also <https://happygitwithr.com/git-branches.html#dealing-with-conflicts>.

### Usage

```
exo_conflict(parent_path)
```

### Arguments

`parent_path`      Path where to create the exercise repo

### Value

The path to the new project

### Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

### Related Git documentation

`git merge`.

### Examples

```
parent_path <- withr::local_tempdir()
path <- exo_conflict(parent_path = parent_path)
```

---

|                    |                                                                   |
|--------------------|-------------------------------------------------------------------|
| exo_latest_message | <i>"Oh shit, I need to change the message on my last commit!"</i> |
|--------------------|-------------------------------------------------------------------|

---

## Description

To go with <https://ohshitgit.com/#change-last-commit-message>

## Usage

```
exo_latest_message(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

`git commit -amend.`

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_latest_message(parent_path = parent_path)
```

---

```
exo_log_deleted_file  "I want to find which commit deleted a file!"
```

---

## Description

I want to find which commit deleted script.R. The tool for that is `git log --oneline -- path`.

## Usage

```
exo_log_deleted_file(parent_path)
```

## Arguments

`parent_path`      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

```
git log - <path>.
```

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_log_deleted_file(parent_path = parent_path)
```

---

```
exo_log_deleted_line  "I want to find which commit deleted a line!"
```

---

## Description

I want to find which commit deleted the line "iris" from the text file. The tool for that is `git log -S<string> path/to/file` or `git log -G<regex> path/to/file`.

## Usage

```
exo_log_deleted_line(parent_path)
```

## Arguments

`parent_path`      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

`git log -S<string>`, `git log -G<regex>`.

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_log_deleted_line(parent_path = parent_path)
```

---

|                      |                                                                                         |
|----------------------|-----------------------------------------------------------------------------------------|
| exo_one_small_change | <i>"Oh shit, I committed and immediately realized I need to make one small change!"</i> |
|----------------------|-----------------------------------------------------------------------------------------|

---

## Description

To go with <https://ohshitgit.com/#change-last-commit>

## Usage

```
exo_one_small_change(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

```
git commit -amend -no-edit.
```

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_one_small_change(parent_path = parent_path)
# Now add "thing 3" to the "bla" file
# And amend the latest commit
```

---

|              |                                                                                  |
|--------------|----------------------------------------------------------------------------------|
| exo_rebase_i | <i>"Hey I'd like to make my commits in a branch look informative and smart!"</i> |
|--------------|----------------------------------------------------------------------------------|

---

## Description

I am working in a feature branch that's all my own. I made many small commits as I was figuring things out. Now I want the commits to tell a story for the PR reviewers, and not a story of how many stupid mistakes I made! The tool for that is `git base --interactive` also available as `git rebase -i`. Useful links:

- <https://jvns.ca/blog/2023/11/06/rebasing-what-can-go-wrong-/>
- <https://wizardzines.com/comics/rules-for-rebasing/>
- <https://github.com/MikeMcQuaid/GitInPractice/blob/main/06-RewritingHistoryAndDisasterRecovery.adoc#rebase-commits-interactively-git-rebase-interactive>
- <https://github.blog/developer-skills/github/write-better-commits-build-better-projects/>

## Usage

```
exo_rebase_i(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

`git rebase -i`.

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_rebase_i(parent_path = parent_path)
```

---

`exo_reset`*"Hey I'd like to restart from scratch"*

---

## Description

I am working in a feature branch that's all my own. I made many small commits as I was figuring things out. Now I want the commits to tell a story for the PR reviewers, and not a story of how many stupid mistakes I made! Instead of `git base --interactive` also available as `git rebase -i`, I can also use `git reset --mixed` and then build the commits. Useful links:

- <https://github.blog/developer-skills/github/write-better-commits-build-better-projects/>
- <https://masalmon.eu/2024/06/11/rewrite-git-history/>

## Usage

```
exo_reset(parent_path)
```

## Arguments

`parent_path`      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

`git reset --mixed`.

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_reset(parent_path = parent_path)
```

---

|                 |                                                        |
|-----------------|--------------------------------------------------------|
| exo_revert_file | <i>"Oh shit, I need to undo my changes to a file!"</i> |
|-----------------|--------------------------------------------------------|

---

## Description

To go with <https://ohshitgit.com/#undo-a-file>

## Usage

```
exo_revert_file(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

[git log](#), [git restore](#), [git checkout](#), [git commit](#).

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_revert_file(parent_path = parent_path)
```



---

exo\_revparse*"I want to understand ancestry references!"*

---

## Description

I want to find the commit ID and message for HEAD~5 and HEAD^^. The tool for that is `git rev-parse` combined with `git log` or `git show`. More on ancestry references: [https://git-scm.com/book/be/v2/Git-Tools-Revision-Selection.html#\\_ancestry\\_references](https://git-scm.com/book/be/v2/Git-Tools-Revision-Selection.html#_ancestry_references).

## Usage

```
exo_revparse(parent_path)
```

## Arguments

`parent_path`      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

`git rev-parse`, `git log`, `git show`.

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_revparse(parent_path = parent_path)
```

---

|                   |                                                                                     |
|-------------------|-------------------------------------------------------------------------------------|
| exo_split_changes | <i>"Hey I'd like to split these changes to the same file into several commits!"</i> |
|-------------------|-------------------------------------------------------------------------------------|

---

## Description

I made many edits to a file, in different places. This is too much for a commit, since small commits are best practice. I need to add the changes to Git bit by bit. The tool for that is `git add --patch`, also available as `git add -p`. If all your changes are presented to you as one chunk by `git add --patch`, choose the "s" option for splitting. See [https://git-scm.com/book/en/v2/Git-Tools-Interactive-Staging#\\_staging\\_patches](https://git-scm.com/book/en/v2/Git-Tools-Interactive-Staging#_staging_patches).

Note that `patch` is also an option for `git commit`, if you prefer so.

## Usage

```
exo_split_changes(parent_path)
```

## Arguments

|             |                                        |
|-------------|----------------------------------------|
| parent_path | Path where to create the exercise repo |
|-------------|----------------------------------------|

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

## Related Git documentation

`git add -patch`.

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_split_changes(parent_path = parent_path)
```

---

|                  |                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------|
| exo_time_machine | <i>"Oh shit, I did something terribly wrong, please tell me git has a magic time machine!?!"</i> |
|------------------|--------------------------------------------------------------------------------------------------|

---

## Description

To go with <https://ohshitgit.com/#magic-time-machine>

## Usage

```
exo_time_machine(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

[git reset](#), [git reflog](#).

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_time_machine(parent_path = parent_path)
```

---

|                 |                                                                    |
|-----------------|--------------------------------------------------------------------|
| exo_undo_commit | <i>"Oh shit, I need to undo a commit from like 5 commits ago!"</i> |
|-----------------|--------------------------------------------------------------------|

---

## Description

To go with <https://ohshitgit.com/#undo-a-commit>

## Usage

```
exo_undo_commit(parent_path)
```

## Arguments

parent\_path      Path where to create the exercise repo

## Value

The path to the new project

## Workflow

Running the function will create the exercise as a new folder in parent\_path. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the tip() function from that new R session will display additional guidance.

## Related Git documentation

[git log](#), [git revert](#).

## Examples

```
parent_path <- withr::local_tempdir()
path <- exo_undo_commit(parent_path = parent_path)
```

---

`exo_worktree`*"I need to see what the project looked like at a certain version!"*

---

**Description**

I want to create a folder containing the project as it was at tag v2, to read the text file. The tool for that is `git worktree`. <https://masalmon.eu/2024/01/23/git-worktree/>

**Usage**

```
exo_worktree(parent_path)
```

**Arguments**

`parent_path`      Path where to create the exercise repo

**Value**

The path to the new project

**Workflow**

Running the function will create the exercise as a new folder in `parent_path`. If called from RStudio or Positron, the function will open a new R session in that IDE. If not, the user will need to navigate to the path returned by the function, and launch an R session from there. The new R session will display messages about what challenge to solve, and running the `tip()` function from that new R session will display additional guidance.

**Related Git documentation**

[git log](#), [git worktree](#), [git tag](#).

**Examples**

```
parent_path <- withr::local_tempdir()
path <- exo_worktree(parent_path = parent_path)
```

# Index

`create_all_exercises`, [2](#)

`exo_bisect`, [3](#)

`exo_blame`, [4](#)

`exo_check_editor`, [5](#)

`exo_clean_dir`, [6](#)

`exo_committed_to_main`, [7](#)

`exo_committed_to_wrong`, [8](#)

`exo_conflict`, [9](#)

`exo_latest_message`, [10](#)

`exo_log_deleted_file`, [11](#)

`exo_log_deleted_line`, [12](#)

`exo_one_small_change`, [13](#)

`exo_rebase_i`, [14](#)

`exo_reset`, [15](#)

`exo_revert_file`, [16](#)

`exo_revparse`, [17](#)

`exo_split_changes`, [18](#)

`exo_time_machine`, [19](#)

`exo_undo_commit`, [20](#)

`exo_worktree`, [21](#)