# Unit testing

(With a dash of API design)

***May 2018***

Forwards Teaching Team

## Motivation

## Let’s add a column to a data frame

```
# Write a function that allows us to add a
# new column to a data frame at a specified
# position.

add_col(df, "name", value, where = 1)
# Start simple and try out as we go
```

## Your turn

```
# A useful building block is add_cols() -
# works like cbind() but can insert anywhere

add_cols <- function(x, y, where = 1) {
  if (where == 1) { # first col
    ...
  } else if (where > ncol(x)) { # last col
    ...
  } else {
    ...
  }
}
```

## A first attempt

```
add_cols <- function(x, y, where = 1) {
  if (where == 1) {
    cbind(x, y)
  } else if (where > ncol(x)) {
    cbind(y, x)
  } else {
    cbind(x[1:where], y, x[where:nrow(x)])
  }
}
```

## Actually correct

```
add_cols <- function(x, y, where = 1) {
  if (where == 1) {
    cbind(y, x)
  } else if (where > ncol(x)) {
    cbind(x, y)
  } else {
    lhs <- 1:(where - 1)
    cbind(x[lhs], y, x[-lhs])
  }
}
```

## A common workflow

```
# Create some simple inputs
df1 <- data.frame(a = 3, b = 4, c = 5)
df2 <- data.frame(X = 1, Y = 2)

# After tweaking the function, re-run these cases
add_cols(df1, df2, where = 1)
add_cols(df1, df2, where = 2)
add_cols(df1, df2, where = 3)
add_cols(df1, df2, where = 4)
```

## Two challenges

Duplication/unsaved changes

Looking at the outputs of each run is tedious

## We need a new workflow!

### Duplication/unsaved changes

Create a formal suite of unit tests

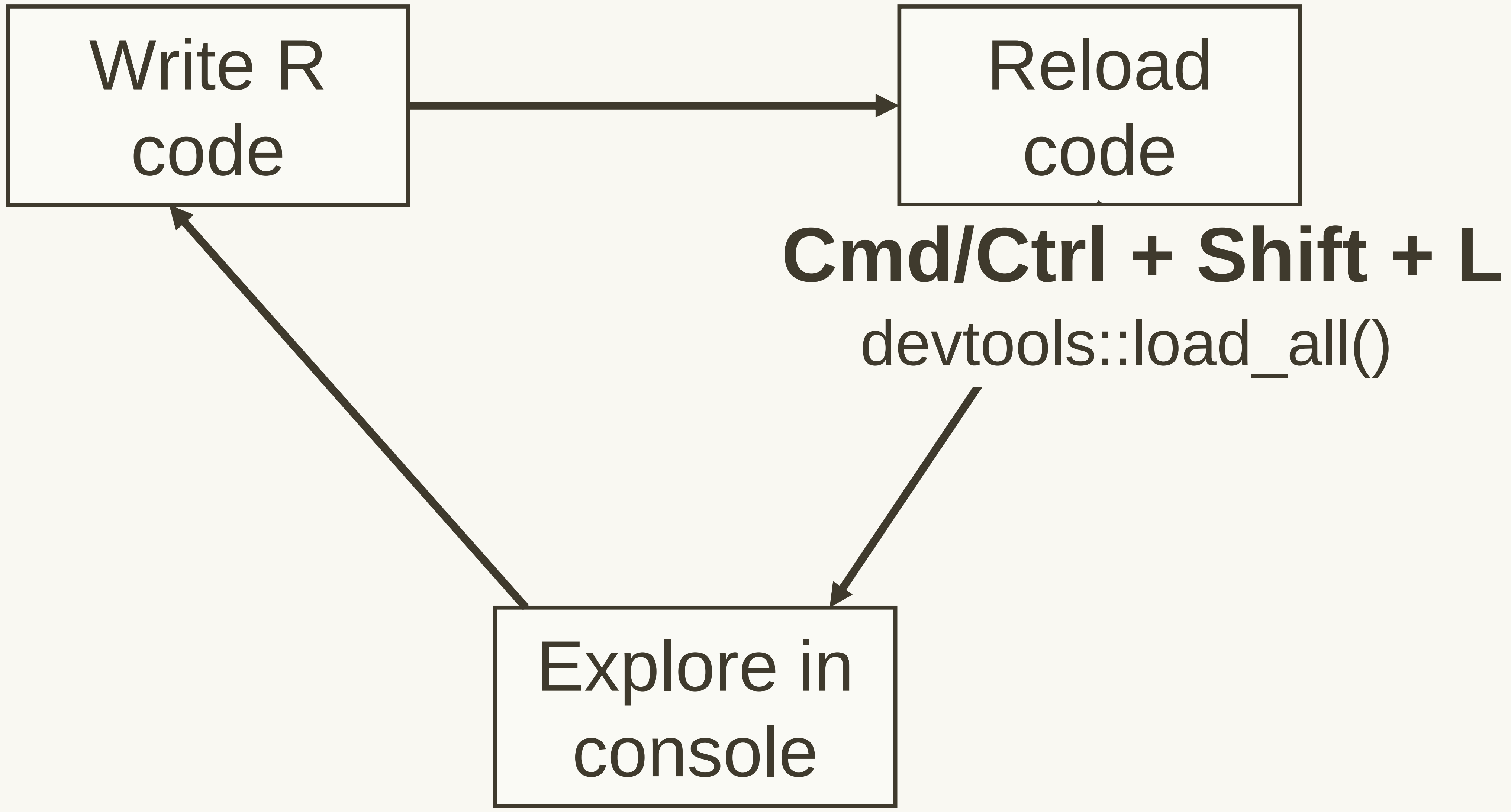
### Looking at the outputs of each run is tedious

Use devtools::**test()** to run & check

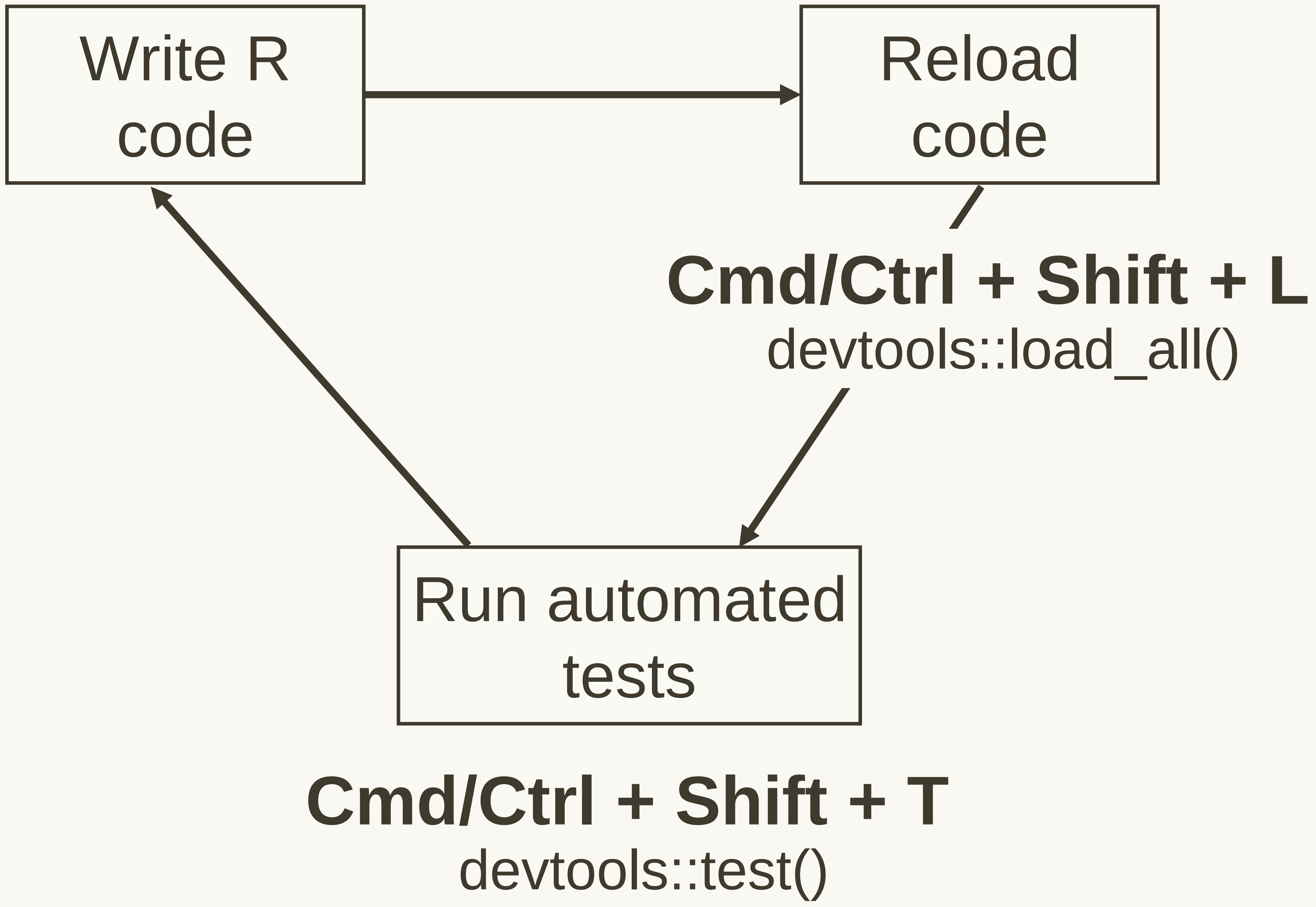
## Testing workflow

http://r-pkgs.had.co.nz/tests.html

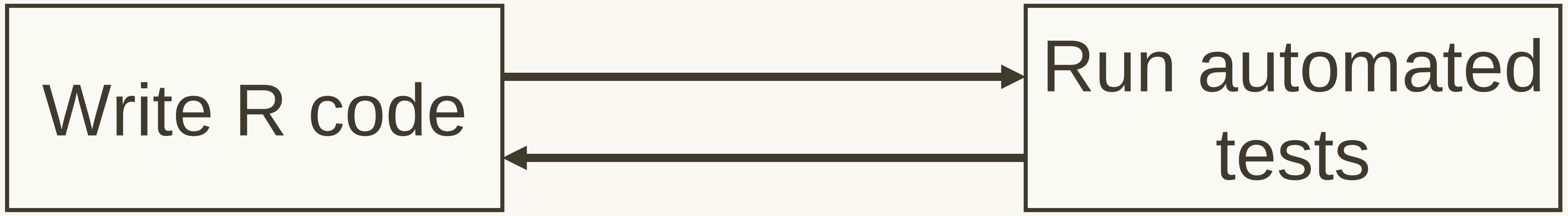
## So far we’ve done this:



# Testthat gives a new workflow



## But why load the code?



**Cmd/Ctrl + Shift + T**

devtools::test()

## We know how to create a package

```r
usethis::create_package("~/desktop/addcol")
usethis::use_r("add_col")

# add_cols <- function(x, y, where = 1) {
#   if (where == 1) {
#     cbind(y, x)
#   } else if (where > ncol(x)) {
#     cbind(x, y)
#   } else {
#     lhs <- 1:(where - 1)
#     cbind(x[lhs], y, x[-lhs])
#   }
# }
```

## Now add tests

Set up testthat inafrastructure

```
usethis::use_test()
✔ Adding 'testthat' to Suggests field
✔ Creating 'tests/testthat/'
✔ Writing 'tests/testthat.R'
✔ Writing 'tests/testthat/test-add_cols.R'
● Modify 'tests/testthat/test-add_cols.R'
```

Create test file matching script

```
devtools::test()
# Or Command + Shift + T
```

## Key idea of unit testing is to automate!

Helper function to reduce duplication

```
at_pos <- function(i) {
  add_cols(df1, df2, where = i)
}

expect_named(at_pos(1), c("X", "Y", "a", "b", "c"))
expect_named(at_pos(2), c("a", "X", "Y", "b", "c"))
expect_named(at_pos(3), c("a", "b", "X", "Y", "c"))
expect_named(at_pos(4), c("a", "b", "c", "X", "Y"))
```

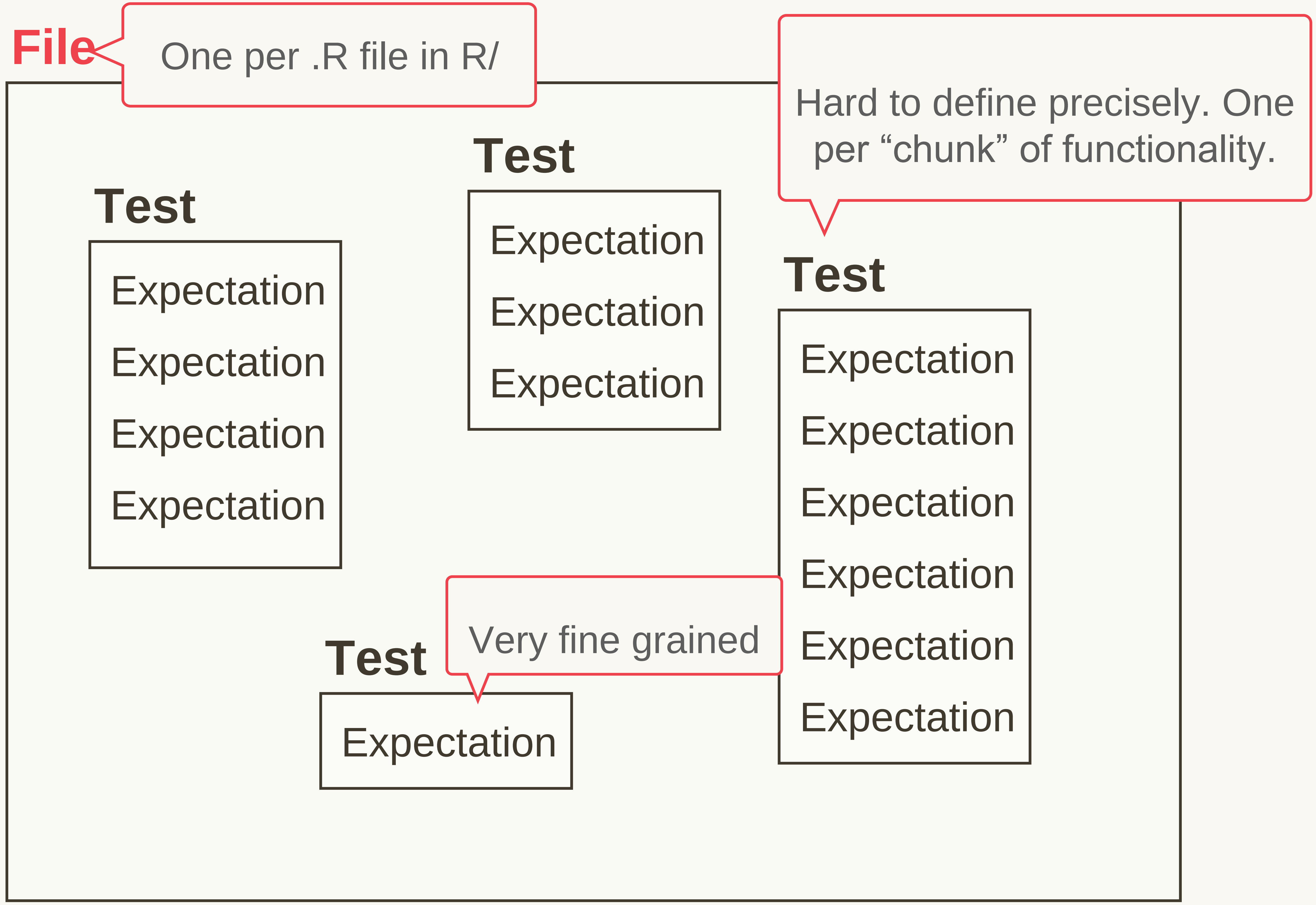
Describes an expected property of the output

## And this automation must follow conventions

Tests for R/add_cols.R

```
# In tests/testthat/test-add_cols.R

test_that("can add column at any position", {
  at_pos <- function(i) {
    add_cols(df1, df2, where = i)
  }

  expect_named(at_pos(1), c("X", "Y", "a", "b", "c"))
  expect_named(at_pos(2), c("a", "X", "Y", "b", "c"))
  expect_named(at_pos(3), c("a", "b", "X", "Y", "c"))
  expect_named(at_pos(4), c("a", "b", "c", "X", "Y"))
})
```

# Tests are organised in three layers



## Practice the workflow

Copy in your add_cols() function.

Create an add_cols() test file using use_test().

Put the previous expectations in a test case.

Verify that the tests pass with Cmd + Shift +T.

Add test using where = -1. Verify that it fails.

## Why test?

Writing tests improves your API

https://unsplash.com/photos/TL5Vy1IM-uA

Improve readability or performance without changing behaviour.

https://unsplash.com/photos/gYBMinOIAUE

When you stop work, leave a test failing

https://unsplash.com/photos/y02jEX_B0O0

https://unsplash.com/photos/AVRxWIsK9Jk

## Test-driven Development

# Next challenge is to implement add_col

```
df <- data.frame(x = 1)

add_col(df, "y", 2, where = 1)
add_col(df, "y", 2, where = 2)
add_col(df, "x", 2)
```

We’ll use **test-driven development** to write add_col such that it passes required tests.

# Four expectations cover 90% of cases

expect_equal(object, expected)

expect_error(code, regexp)

expect_warning(code, regexp)
expect_warning(code, NA)

expect_known_output(code)

## Make these tests pass

```
# use_test("add_col")
test_that("where controls position", {
  df <- data.frame(x = 1)

  expect_equal(
    add_col(df, "y", 2, where = 1),
    data.frame(y = 2, x = 1)
  )
  expect_equal(
    add_col(df, "y", 2, where = 2),
    data.frame(x = 1, y = 2)
  )
})
# Some hints on next slide
```

## Hints

```
# Start by establishing basic form of the
# function and setting up the test cases.
add_col <- function(x, name, value, where = 1) {

}

# Make sure that you can Cmd + Shift + T
# and get two test failures before you
# continue

# More hints on the next slide
```

## More hints

```
# Write the body of add_col(), using add_cols() to do most of the
# work

# add_cols() takes two data frames and
# you have a data frame and a vector

# setNames() lets you change the names of
# data frame
```

## A solution

```
add_col <- function(x, name, value, where) {
  df <- setNames(data.frame(value), name)
  add_cols(x, df, where = where)
}
```

# Make this test pass

```
test_that("can replace columns", {
  df <- data.frame(x = 1)

  expect_equal(
    add_col(df, "x", 2, where = 2),
    data.frame(x = 2)
  )
})
```

## A solution

```
add_col <- function(x, name, value, where) {
  if (name %in% names(x)) {
    x[[name]] <- value
    x
  } else {
    df <- setNames(data.frame(value), name)
    add_cols(x, df, where = where)
  }
}
```

## Make this test pass

```
test_that("default where is far right", {
  df <- data.frame(x = 1)

  expect_equal(
    add_col(df, "y", 2),
    data.frame(x = 1, y = 2)
  )
})
```

## A solution

```
add_col <- function(x, name, value,
                    where = ncol(x) + 1) {
  if (name %in% names(x)) {
    x[[name]] <- value
    x
  } else {
    df <- setNames(data.frame(value), name)
    add_cols(x, df, where = where)
  }
}
```

## Can we use add_col() to **remove** columns?

```
df <- data.frame(x = 1, y = 2)

expect_equal(
  add_col(df, "x", NULL)
  data.frame(y = 2)
)

# Should we?
# Would remove_col() be better?
```

## Can we use add_col() to **move** columns?

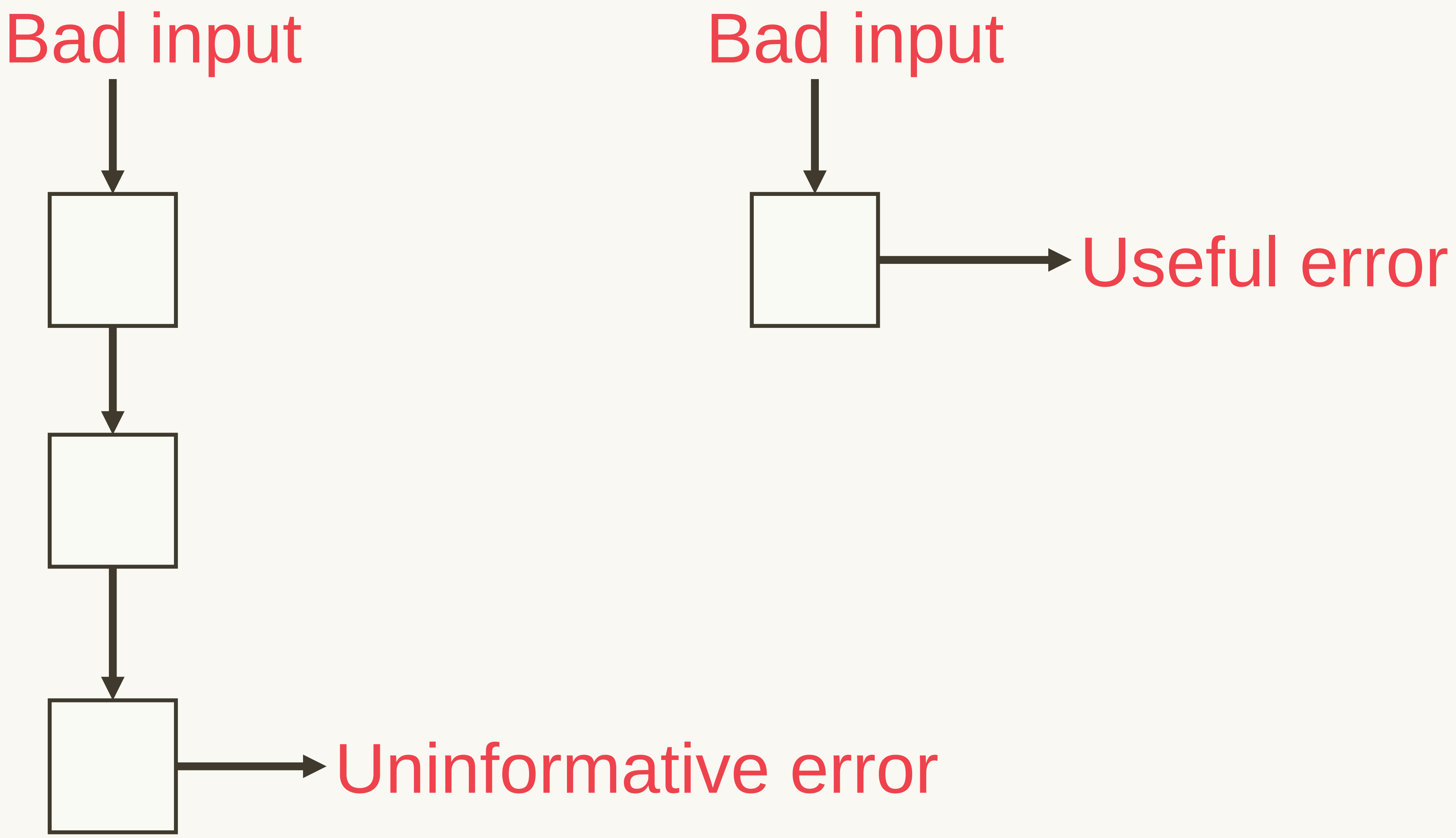
```
df <- data.frame(x = 1, y = 2)

expect_equal(
  add_col(df, "x", 1, where = 2)
  data.frame(y = 2, x = 2)
)

# Should we?
# Would move_col() be better?
```

## Fail fast

## What about bad inputs?

```
# We need to test for errors too
add_cols(df1, df2, where = 0)
add_cols(df1, df2, where = NA)
add_cols(df1, df2, where = 1:10)
add_cols(df1, df2, where = "a")
```

## For robust code, fail early



## We *could* add to add_cols directly

```
add_cols <- function(x, y, where = 1) {
  if (!is.numeric(where) || length(where) != 1) {
    stop("`where` is not a number", call. = FALSE)
  } else if (where == 0 || is.na(where)) {
    stop("`where` must not be 0 or NA", call. = FALSE)
  } else if (where == 1 || where <= -ncol(x)) {
   cbind(x, y)
  } else if (where >= ncol(x) || where == -1) {
    cbind(y, x)
  } else {
    if (where < 0) where <- nrow(x) + where
    cbind(x[1:where], y, x[where:nrow(x)])
  }
}
```

## But this confuses the intent of add_cols

```
# Better to have one function responsible
# for checking for valid inputs
check_where <- function(where, ncols) {
  ...
}

# This also makes it easier to test because
# it's independent of add_cols
```

## Example tests

```
# check_where() lives in same file as add_cols()
# so tests should live in test-add_cols()

test_that("where must be valid value", {
  expect_error(check_where("a"), "length one numeric vector")
  expect_error(check_where(1:10), "length one numeric vector")

  expect_error(check_where(0), "not be zero or missing")
  expect_error(check_where(NA_real_), "not be zero or missing")
})
```

## Test coverage

## Useful to know which lines have been tested

```
# Powered by the covr package
devtools::test_coverage()
```

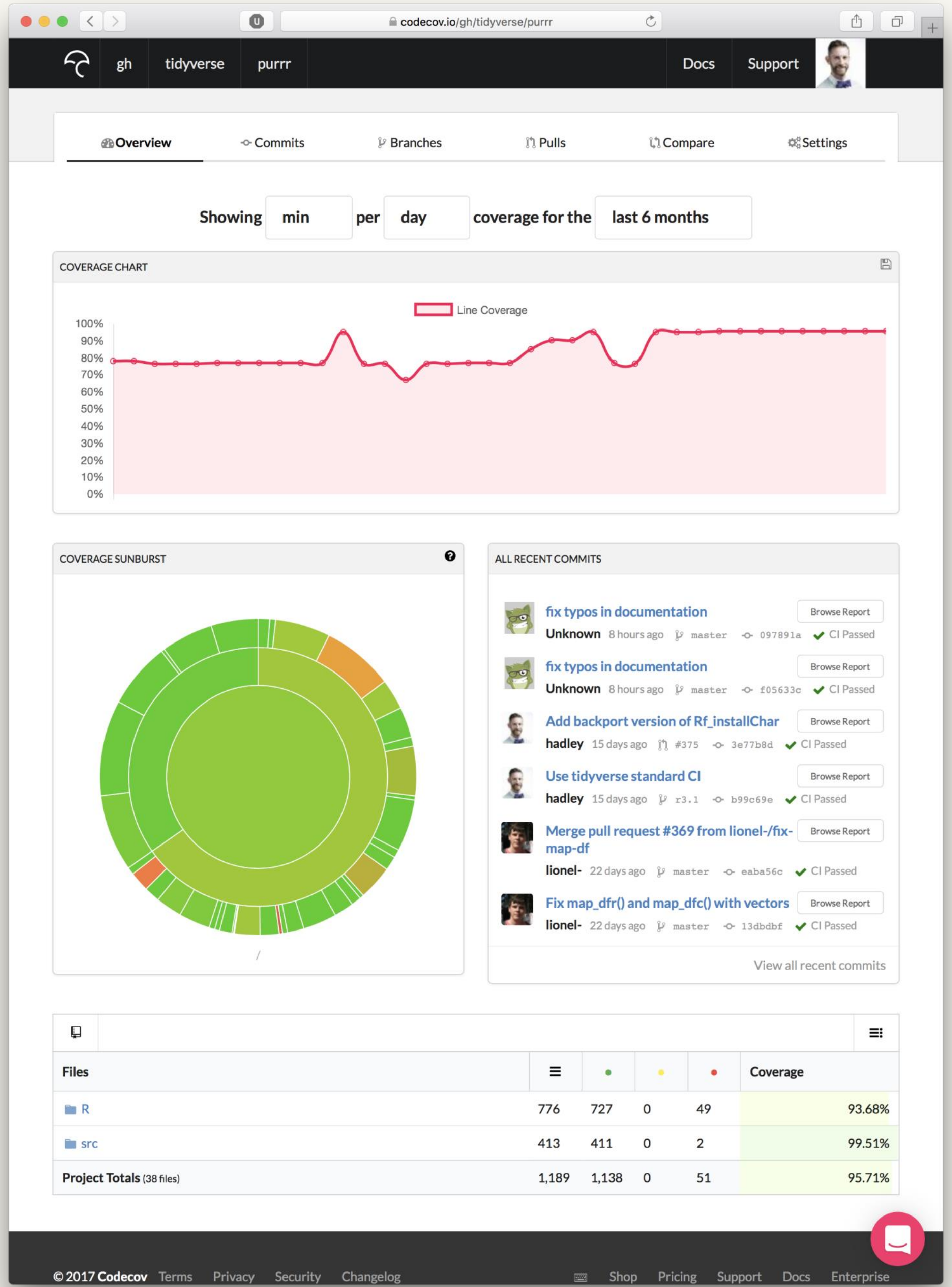
## You can also automate

Known as **continuous integration**

1. Publish your code on GitHub
2. Use Travis and/or Appveyor to build package and run tests every time you push changes.

   ```
   usethis::use_travis()
   usethis::appveyor()
   ```

3. Use Codecov to display test coverage



Showing min per day coverage for the last 6 months

COVERAGE CHART

COVERAGE SUNBURST

ALL RECENT COMMITS

fix typos in documentation — Browse Report
Unknown 8 hours ago master 097891a ✓ CI Passed

fix typos in documentation — Browse Report
Unknown 8 hours ago master f05633c ✓ CI Passed

Add backport version of Rf_installChar — Browse Report
hadley 15 days ago #375 3e77b8d ✓ CI Passed

Use tidyverse standard CI — Browse Report
hadley 15 days ago r3.1 b99c69e ✓ CI Passed

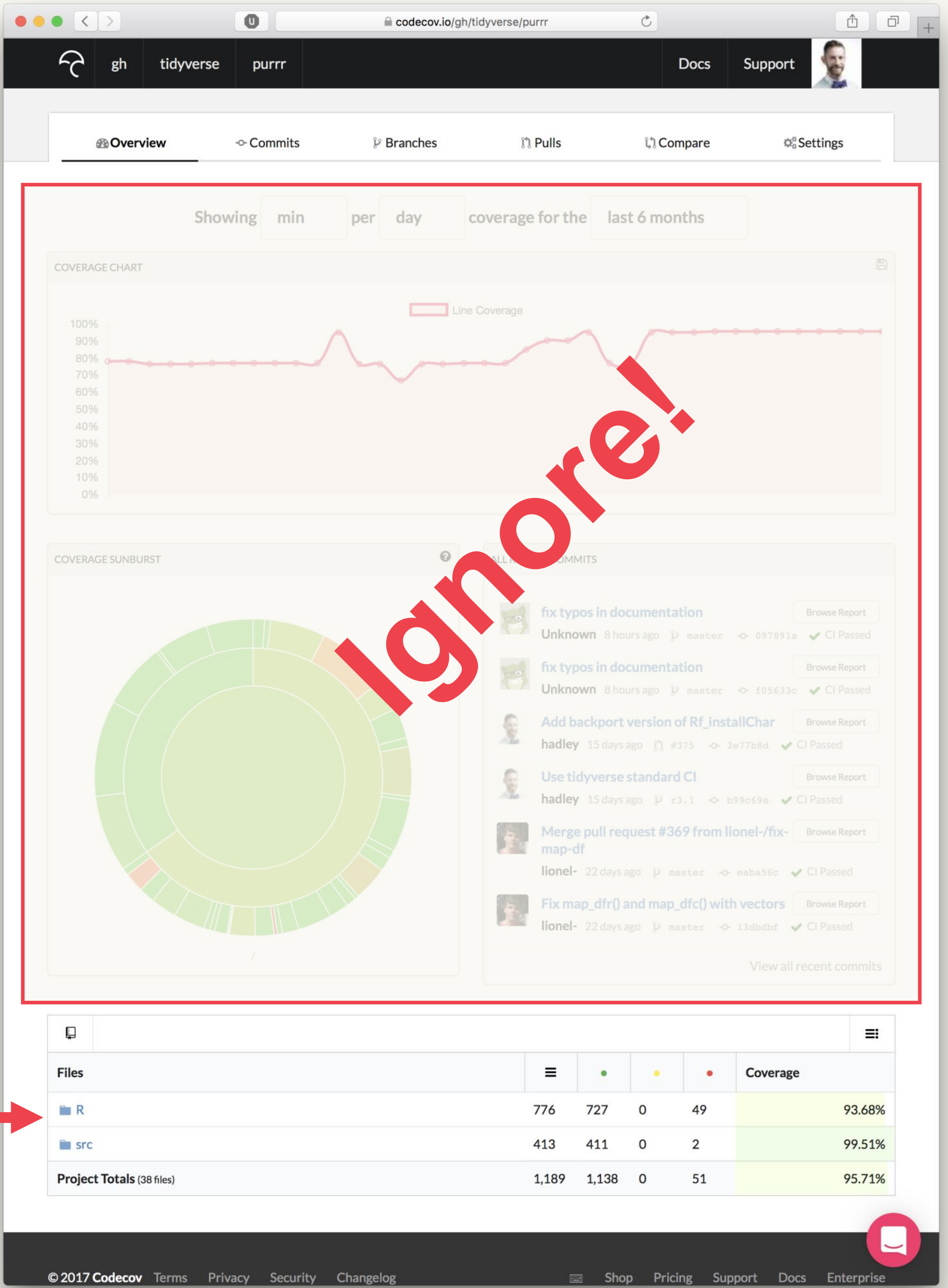
Merge pull request #369 from lionel-/fix-map-df — Browse Report
lionel- 22 days ago master eaba56c ✓ CI Passed

Fix map_dfr() and map_dfc() with vectors — Browse Report
lionel- 22 days ago master 13dbdbf ✓ CI Passed

View all recent commits

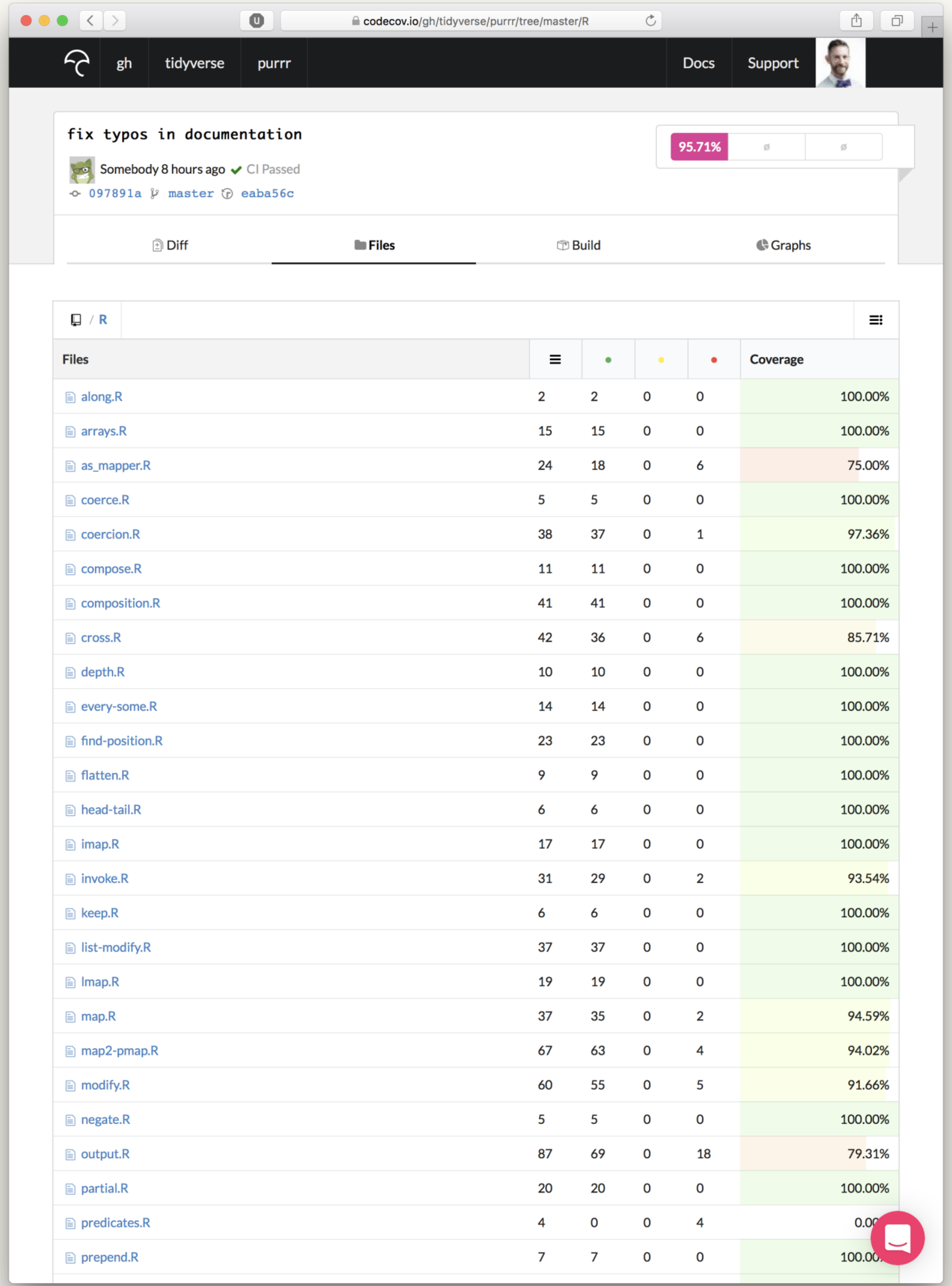
| Files | ≡ | ● | ● | ● | Coverage |
|---|---|---|---|---|---|
| R | 776 | 727 | 0 | 49 | 93.68% |
| src | 413 | 411 | 0 | 2 | 99.51% |
| Project Totals (38 files) | 1,189 | 1,138 | 0 | 51 | 95.71% |

© 2017 Codecov Terms Privacy Security Changelog Shop Pricing Support Docs Enterprise



codecov.io/gh/tidyverse/purrr
gh
tidyverse
purrr
Docs
Support
Overview
Commits
Branches
Pulls
Compare
Settings
Showing min per day coverage for the last 6 months
COVERAGE CHART
Line Coverage
COVERAGE SUNBURST
ALL RECENT COMMITS
fix typos in documentation
Unknown 8 hours ago master 097891a CI Passed
fix typos in documentation
Unknown 8 hours ago master f05633c CI Passed
Add backport version of Rf_installChar
hadley 15 days ago #375 3e77b8d CI Passed
Use tidyverse standard CI
hadley 15 days ago r3.1 b99c69e CI Passed
Merge pull request #369 from lionel-/fix-map-df
lionel- 22 days ago master eaba56c CI Passed
Fix map_dfr() and map_dfc() with vectors
lionel- 22 days ago master 13dbdbf CI Passed
Browse Report
View all recent commits
Ignore!
Click!
© 2017 Codecov Terms Privacy Security Changelog
Shop Pricing Support Docs Enterprise

| Files | ≡ | ● | ● | ● | Coverage |
|---|---|---|---|---|---|
| R | 776 | 727 | 0 | 49 | 93.68% |
| src | 413 | 411 | 0 | 2 | 99.51% |
| Project Totals (38 files) | 1,189 | 1,138 | 0 | 51 | 95.71% |



codecov.io/gh/tidyverse/purrr/tree/master/R

gh tidyverse purrr Docs Support

fix typos in documentation

Somebody 8 hours ago ✓ CI Passed

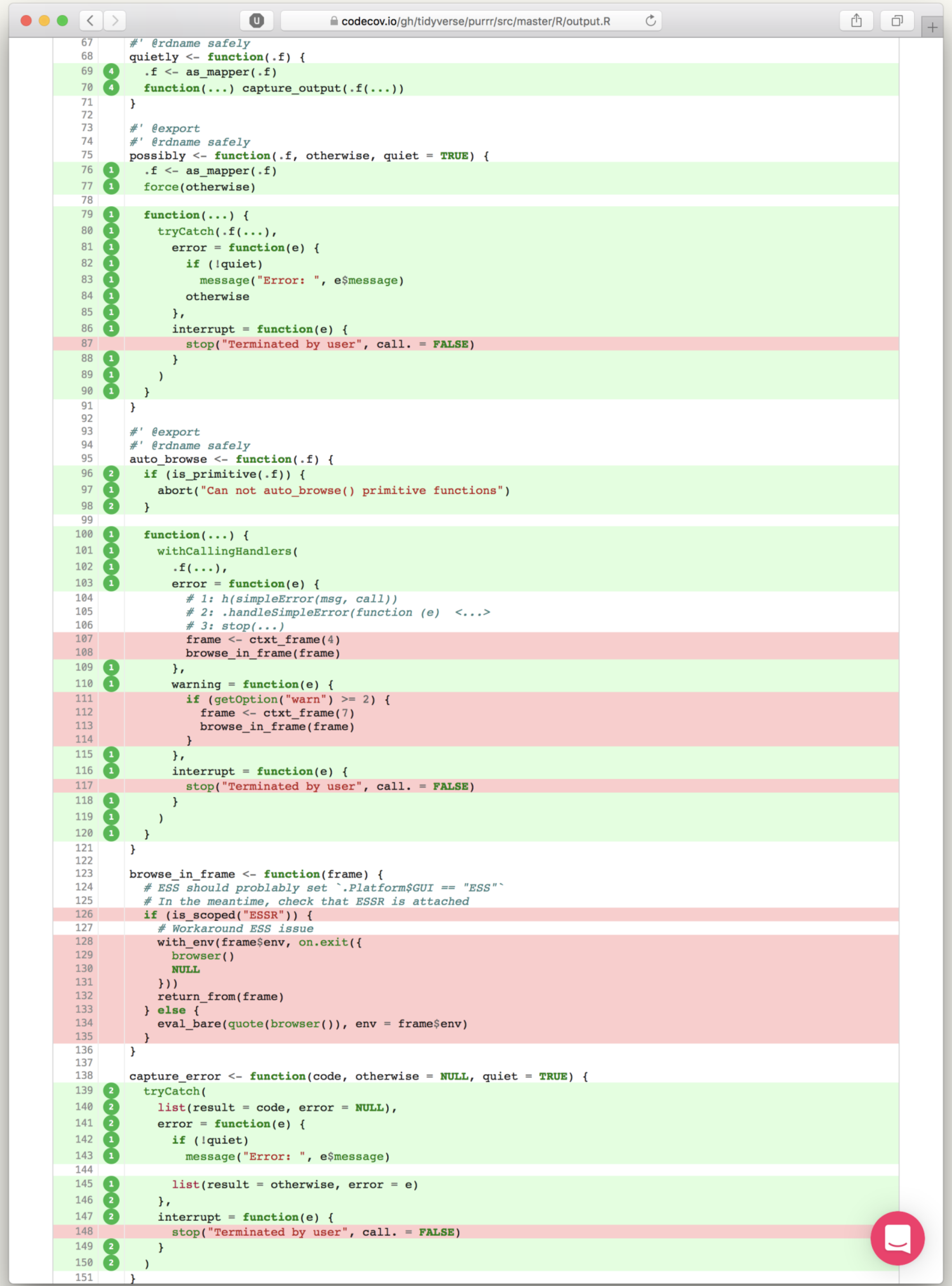
097891a master eaba56c

95.71% ø ø

Diff Files Build Graphs

/ R

| Files | ≡ | 🟢 | 🟡 | 🔴 | Coverage |
|---|---|---|---|---|---|
| along.R | 2 | 2 | 0 | 0 | 100.00% |
| arrays.R | 15 | 15 | 0 | 0 | 100.00% |
| as_mapper.R | 24 | 18 | 0 | 6 | 75.00% |
| coerce.R | 5 | 5 | 0 | 0 | 100.00% |
| coercion.R | 38 | 37 | 0 | 1 | 97.36% |
| compose.R | 11 | 11 | 0 | 0 | 100.00% |
| composition.R | 41 | 41 | 0 | 0 | 100.00% |
| cross.R | 42 | 36 | 0 | 6 | 85.71% |
| depth.R | 10 | 10 | 0 | 0 | 100.00% |
| every-some.R | 14 | 14 | 0 | 0 | 100.00% |
| find-position.R | 23 | 23 | 0 | 0 | 100.00% |
| flatten.R | 9 | 9 | 0 | 0 | 100.00% |
| head-tail.R | 6 | 6 | 0 | 0 | 100.00% |
| imap.R | 17 | 17 | 0 | 0 | 100.00% |
| invoke.R | 31 | 29 | 0 | 2 | 93.54% |
| keep.R | 6 | 6 | 0 | 0 | 100.00% |
| list-modify.R | 37 | 37 | 0 | 0 | 100.00% |
| lmap.R | 19 | 19 | 0 | 0 | 100.00% |
| map.R | 37 | 35 | 0 | 2 | 94.59% |
| map2-pmap.R | 67 | 63 | 0 | 4 | 94.02% |
| modify.R | 60 | 55 | 0 | 5 | 91.66% |
| negate.R | 5 | 5 | 0 | 0 | 100.00% |
| output.R | 87 | 69 | 0 | 18 | 79.31% |
| partial.R | 20 | 20 | 0 | 0 | 100.00% |
| predicates.R | 4 | 0 | 0 | 4 | 0.0 |
| prepend.R | 7 | 7 | 0 | 0 | 100.0 |



```r
#' @rdname safely
quietly <- function(.f) {
  .f <- as_mapper(.f)
  function(...) capture_output(.f(...))
}

#' @export
#' @rdname safely
possibly <- function(.f, otherwise, quiet = TRUE) {
  .f <- as_mapper(.f)
  force(otherwise)

  function(...) {
    tryCatch(.f(...),
      error = function(e) {
        if (!quiet)
          message("Error: ", e$message)
        otherwise
      },
      interrupt = function(e) {
        stop("Terminated by user", call. = FALSE)
      }
    )
  }
}

#' @export
#' @rdname safely
auto_browse <- function(.f) {
  if (is_primitive(.f)) {
    abort("Can not auto_browse() primitive functions")
  }

  function(...) {
    withCallingHandlers(
      .f(...),
      error = function(e) {
        # 1: h(simpleError(msg, call))
        # 2: .handleSimpleError(function (e)  <...>
        # 3: stop(...)
        frame <- ctxt_frame(4)
        browse_in_frame(frame)
      },
      warning = function(e) {
        if (getOption("warn") >= 2) {
          frame <- ctxt_frame(7)
          browse_in_frame(frame)
        }
      },
      interrupt = function(e) {
        stop("Terminated by user", call. = FALSE)
      }
    )
  }
}

browse_in_frame <- function(frame) {
  # ESS should problably set `.Platform$GUI == "ESS"`
  # In the meantime, check that ESSR is attached
  if (is_scoped("ESSR")) {
    # Workaround ESS issue
    with_env(frame$env, on.exit({
      browser()
      NULL
    }))
    return_from(frame)
  } else {
    eval_bare(quote(browser()), env = frame$env)
  }
}

capture_error <- function(code, otherwise = NULL, quiet = TRUE) {
  tryCatch(
    list(result = code, error = NULL),
    error = function(e) {
      if (!quiet)
        message("Error: ", e$message)

      list(result = otherwise, error = e)
    },
    interrupt = function(e) {
      stop("Terminated by user", call. = FALSE)
    }
  )
}
```