Unit testing

(With a dash of API design)

December 2017

Hadley Wickham,
Jenny Bryan,
Di Cook

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)
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 {
    ...
  }
}
```

My 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])
  }
}
```

How did I write that code?

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

# Then each time I tweaked it, I re-ran
# 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

Cmd + Enter is error prone

Looking at the outputs
each run is tedious

We need a new workflow!

Cmd + Enter is error prone

Put code in R/ and use devtools::**load_all()**

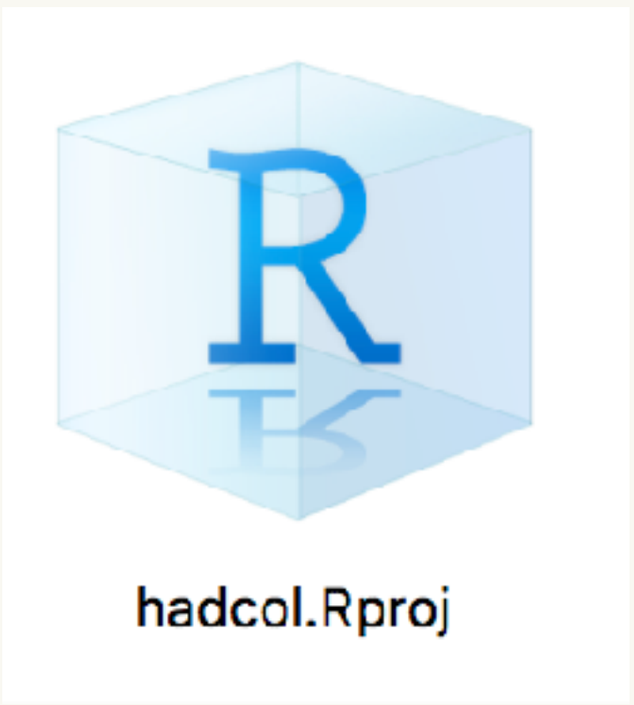
Looking at the outputs each run is tedious

Write unit tests and use devtools::**test()**

We know how to create a package

```
usethis::create_package("~/desktop/hadcol")
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])
#   }
# }
```

Or just open hadcol



Testing workflow

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

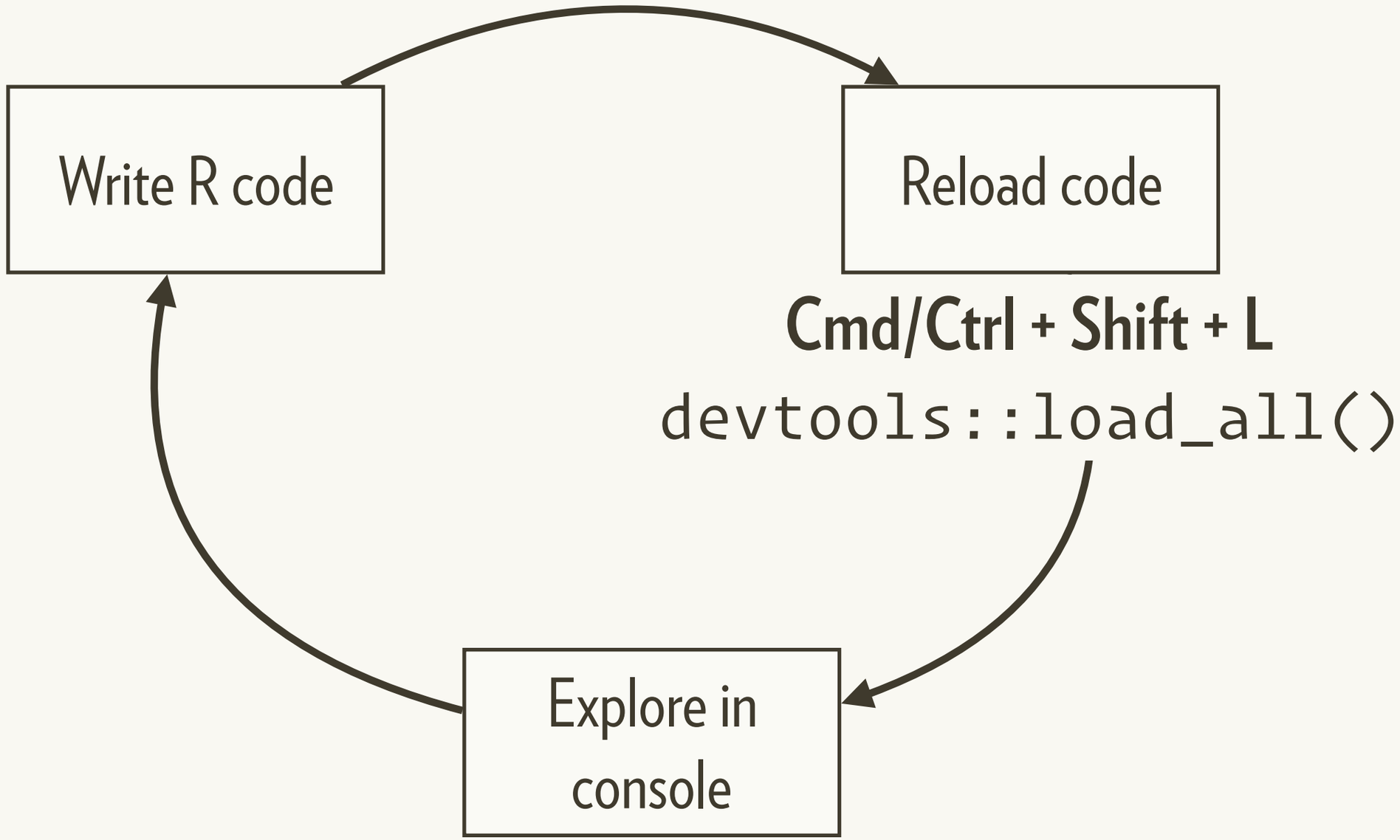
Even more convenient with some conventions

```
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'

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

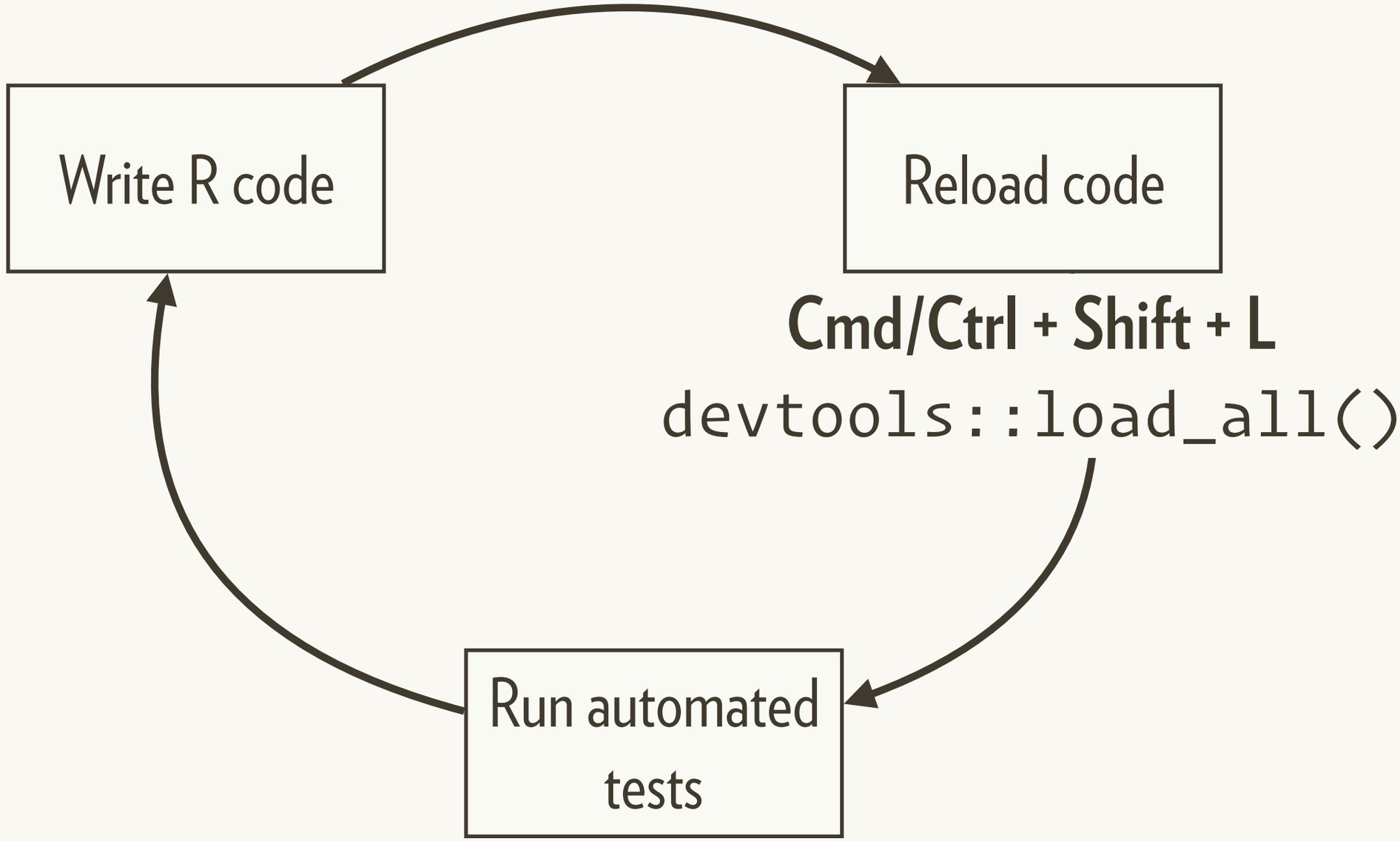
Set up testthat infrastructure

Create test file matching script

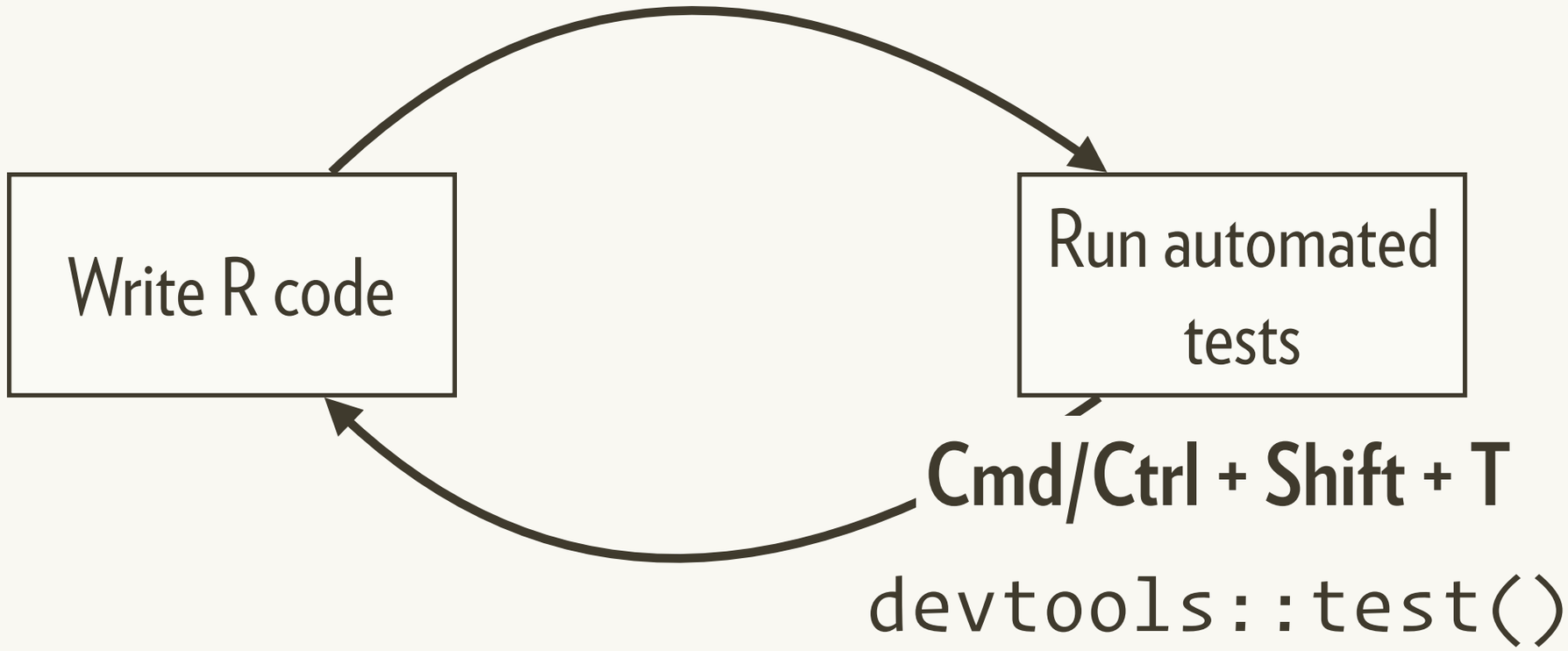
So far we've done this:



Testthat gives a new workflow



But why load the code?



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"))
```

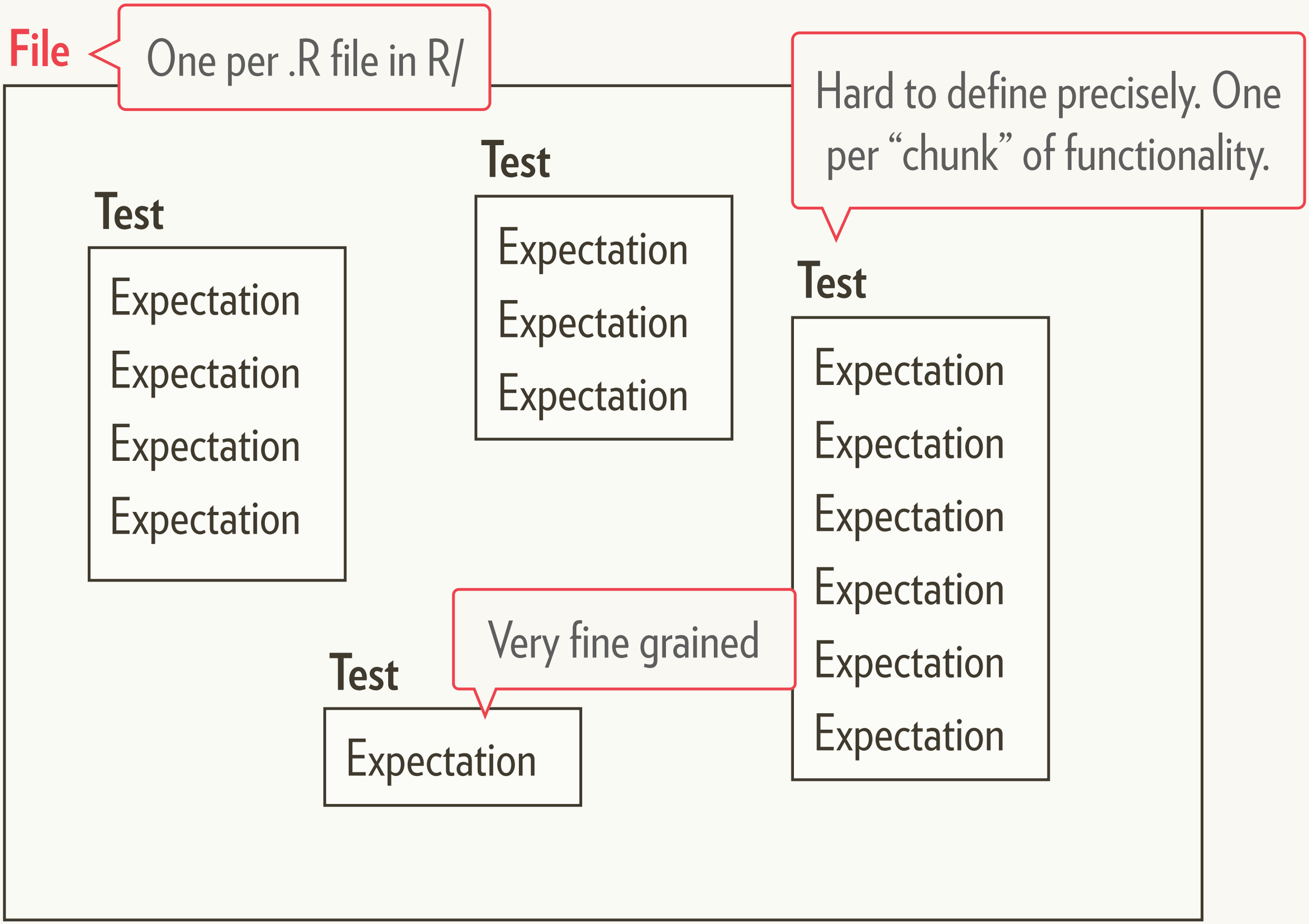
Describes an expected property of the output

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 and 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?

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

If you're bored in this class, write tests!

https://unsplash.com/photos/AVRxWIsK9Jk

add_col

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)
```

Four expectations cover 90% of cases

```
expect_equal(obj, exp)

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

```
# You'll need to use add_cols

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

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

My 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)
  )
})
```

My solution

```
add_col <- function(x, name, value) {
  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)
  )
})
```

My 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?

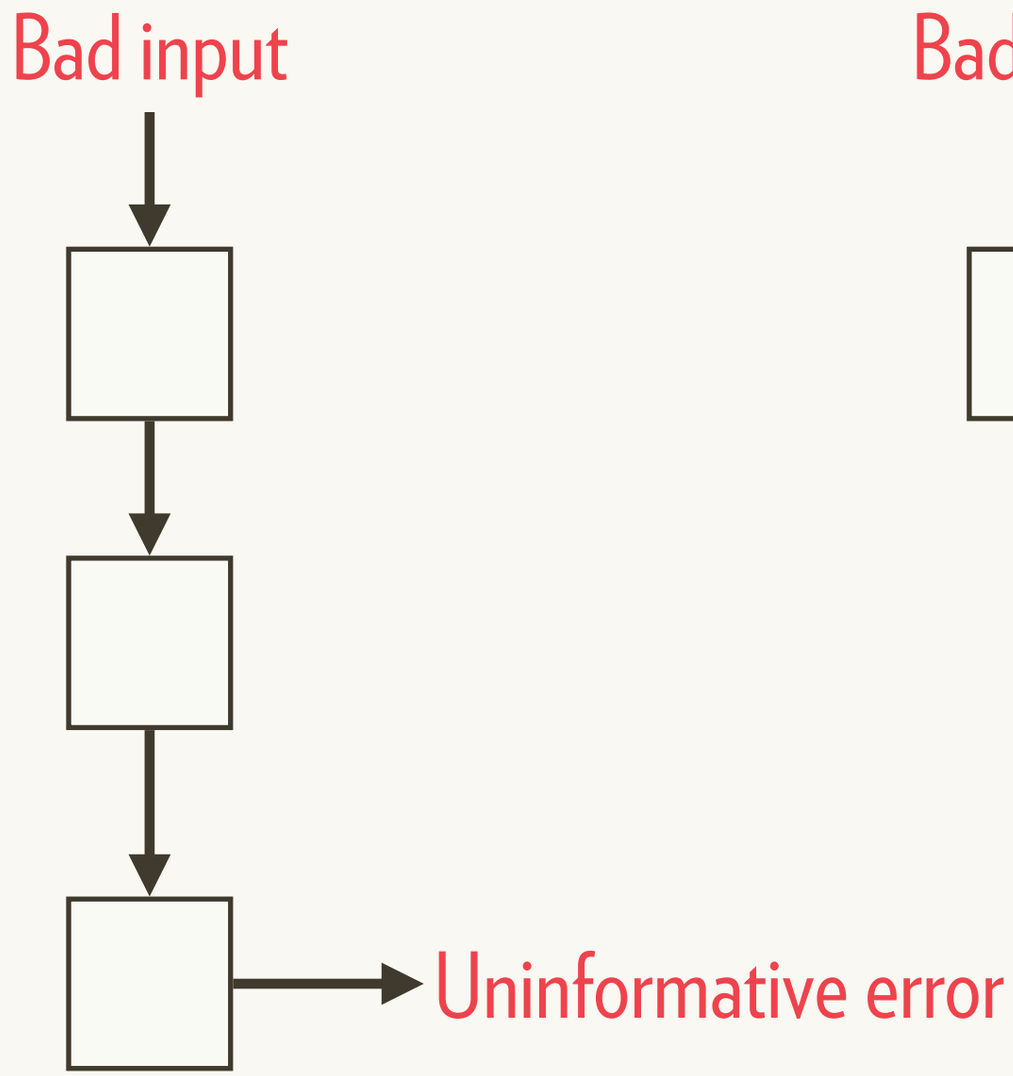
```
df <- data.frame(x = 1, y = 2)

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

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

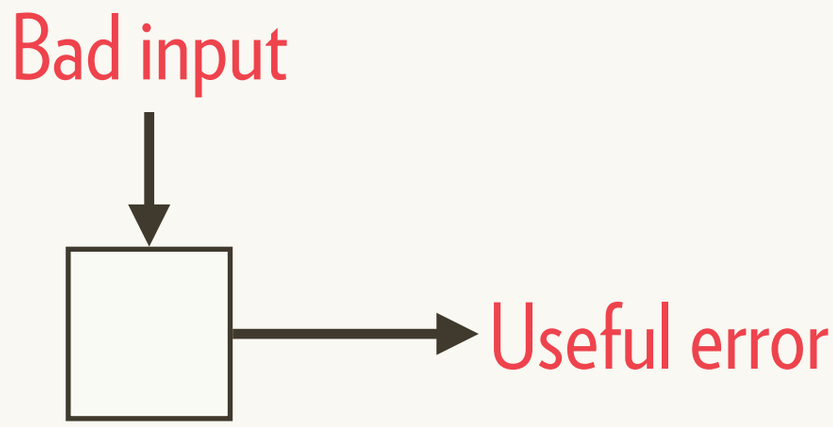
Fail fast

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 (and handling
  # -ve values)
  check_where <- function(where, ncols) {
    ...
}

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

Your turn

Write check_where(). It should return an integer or throw an error. I suggest you put in the same file as add_cols().

It will need to take where and number of columns.

My answer

```
  check_where <- function(x, ncol) {
    if (length(x) != 1 || !is.numeric(x)) {
      stop("`where` must be a length one numeric vector.", call. = FALSE)
    }
    x <- as.integer(x)

    if (x == 0 || is.na(x)) {
      stop("`where` must not be zero or missing", call. = FALSE)
    } else {
      x
    }
  }
```

A few conventions for stop()

- Always use `call. = FALSE`
- Surround variable names in `` `...` ``, and strings in '...'
- Message should say what is needed, compared to what was provided

Use expect_error() to test for errors

```
expect_error(
  check_where("a")
)
```

```
expect_error(
  check_where("a"),
  "not a number"
)
```

A regular expression

Your turn

Write tests to ensure that check_where() only allows valid inputs. (Where should the tests live?)

My 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")
})
```

What about negative values?

1	2	3	4
-4	**-3**	**-2**	**-1**

x	y	z
1	a	4.5
2	b	1.2
4	c	6.7

Your turn

Write some tests that establish what check_where() should return for negative positions. Implement the code to make the tests pass.

Tests

Updated check_where()

Other side effects

```
# Check for warning and something else
expect_warning(out <- foo(), "blah")
expect_equal(out, 10)
```

```
# Checking for absence
expect_warning(code, NA)
expect_message(code, NA)
expect_output(code, NA)
```

Test coverage

Useful to know which lines have been tested

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

Your turn

Run covr and verify that every line has been tested.

Have we missed anything? Can you add a test that checks it?

You can also automate

- GitHub = publish your code online
- Travis = run code every time your code changes
- Codecov = display which functions are tested

codecov.io/gh/tidyverse/purrr

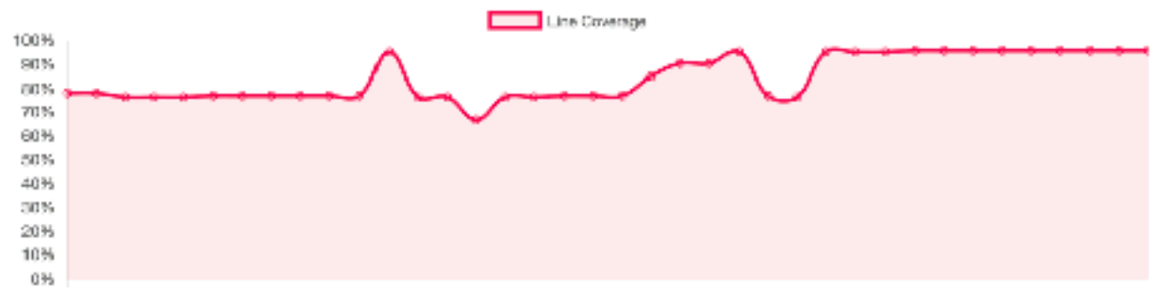
gh / tidyverse / purrr

Docs Support

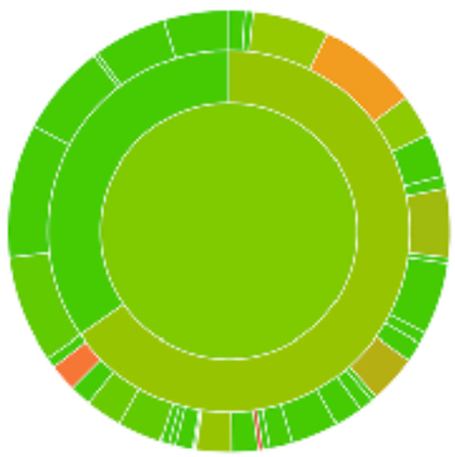
Overview | Commits | Branches | Pulls | Compare | Settings

Showing min per day coverage for the last 6 months

COVERAGE CHART



COVERAGE SUNBURST



ALL RECENT COMMITS

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

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

Add backport version of Rf_installChar — Browse Report
hadley 15 days ago — CI Passed

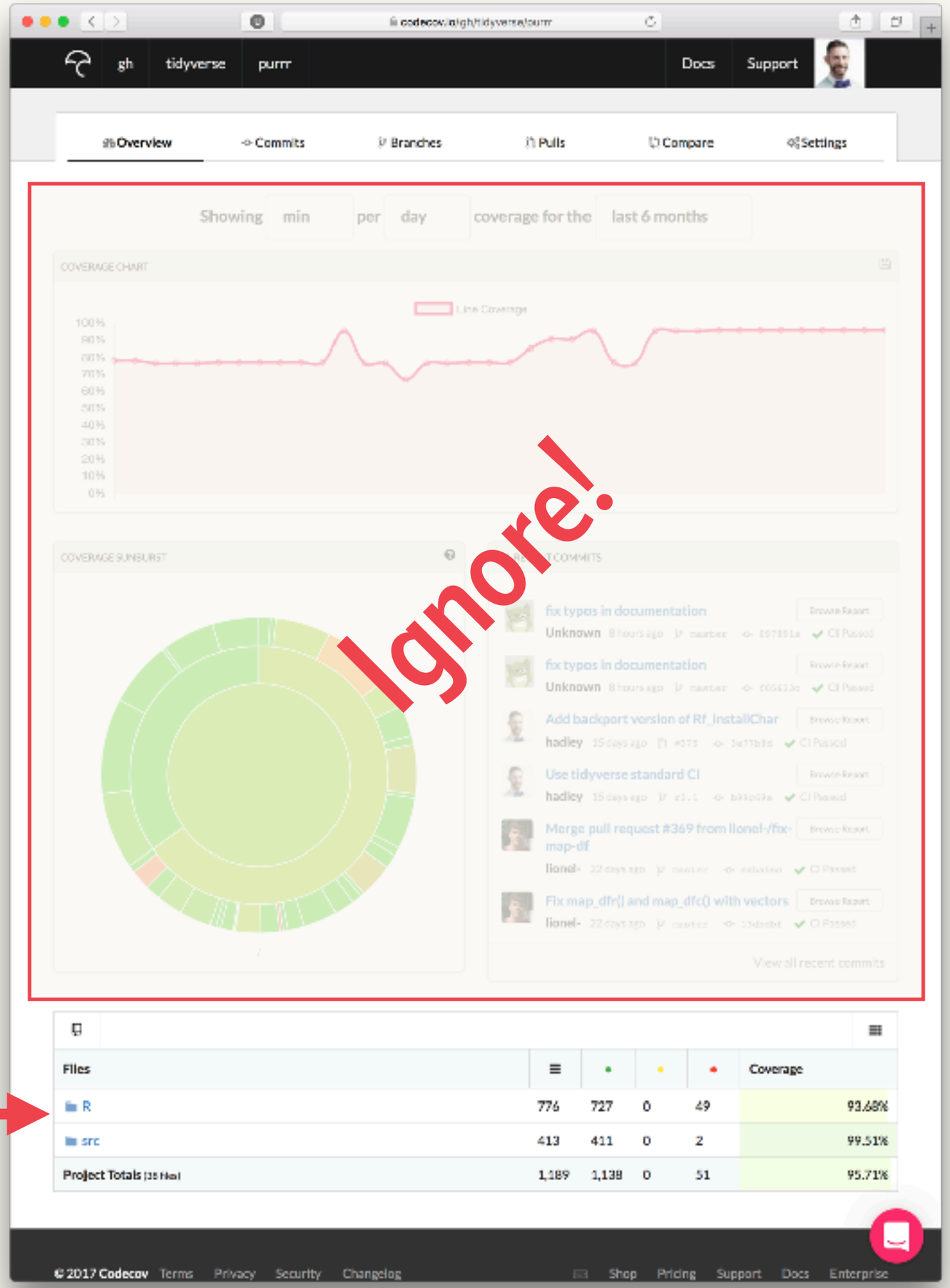
Use tidyverse standard CI — Browse Report
hadley 15 days ago — CI Passed

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

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

View all recent commits

Files				Coverage
R	776	727	0	49	93.68%
src	413	411	0	2	99.51%
Project Totals (35 files)	1,189	1,138	0	51	95.71%


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
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
COVERAGE SUNBURST
fix typos in documentation
Unknown
fix typos in documentation
Unknown
Add backport version of Rf_installChar
hadley
Use tidyverse standard CI
hadley
Merge pull request #369 from lionel-/fix-map-df
lionel-
Fix map_dfr() and map_dfc() with vectors
lionel-
View all recent commits
Ignore!
Files
Coverage
R
776
727
0
49
93.68%
src
413
411
0
2
99.51%
Project Totals
1,189
1,139
0
51
95.71%
© 2017 Codecov
Terms
Privacy
Security
Changelog
Shop
Pricing
Support
Docs
Enterprise

Click!

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%

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	0.00%
prepend.R	7	7	0	0	100.00%


codecov.io/gh/tidyverse/purrr/src/master/R/output.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$message)
otherwise
},
interrupt = function(e) {
stop("Terminated by user", call. = FALSE)
}
)
}
}
#' @export
#' @rdname safely
auto_browse <- function(.f) {
if (is_primitive(.f)) {
abort("Can not auto_browse() primitive functions")
}
function(...) {
withCallingHandlers(
.f(...),
error = function(e) {
# 1: h(simpleError(msg, call))
# 2: .handleSimpleError(function (e) <...>
# 3: stop(...)
frame <- ctxt_frame(4)
browse_in_frame(frame)
},
warning = function(e) {
if (getOption("warn") >= 2) {
frame <- ctxt_frame(7)
browse_in_frame(frame)
}
},
interrupt = function(e) {
stop("Terminated by user", call. = FALSE)
}
)
}
}
browse_in_frame <- function(frame) {
# RGG should probably use `.Platform$GUI == "RGG"`
# In the meantime, check that RGGR is attached
if (is_scoped("RGGR")) {
# Workaround RGG 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)
}
)
}