

R/

January 2017

Hadley Wickham
[@hadleywickham](#)
Chief Scientist, RStudio

What is a source
package?

A package is a set of
conventions that
(with the right tools)
makes your life easier

Start by picking a name

Naming recommendations

**Only lowercase letters
& numbers**

Add r

tidyr

stringr

Find related word and modify

plyr

lubridate

httr

Be googleable!

ggplot2

Be memorable

Your turn

Brainstorm a better name than rv2!

Once you have a name you can create the package

Get started with:

```
devtools::create("path/to/package")
```

```
devtools::setup("existing/path/")
```

You can also create new project using RStudio

but it has some slightly differences that will

cause hassles today (but not in general)

They both create the minimal valid package.

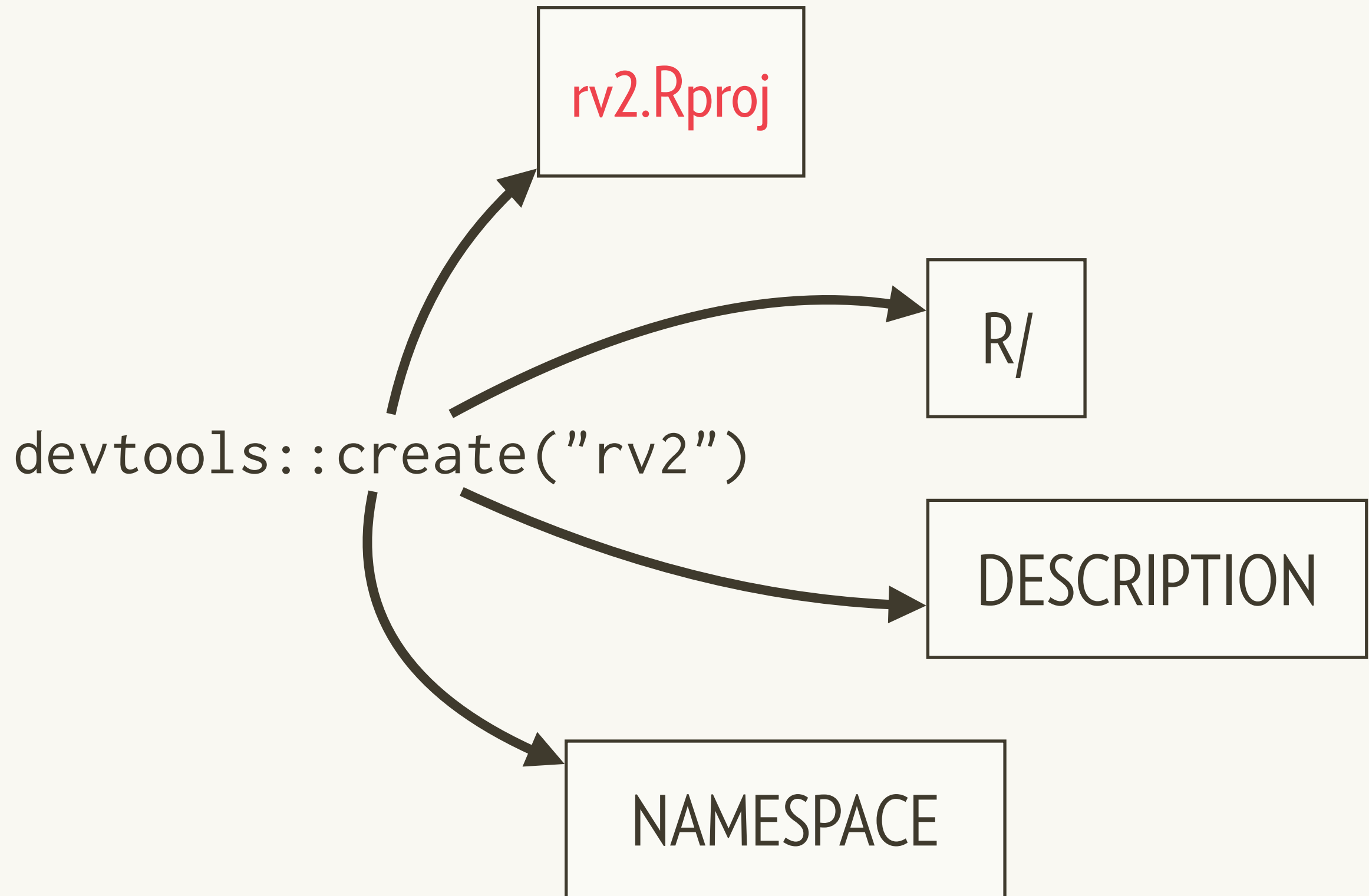
You'll learn about all the pieces today



```
package.skeleton()
```

Never use this!

What happens we run create?



.Rproj

RStudio projects

Projects make your life easier

(not just for packages — use for all data analyses)



gadgets.Rproj

Isolate code and results

Multiple projects open

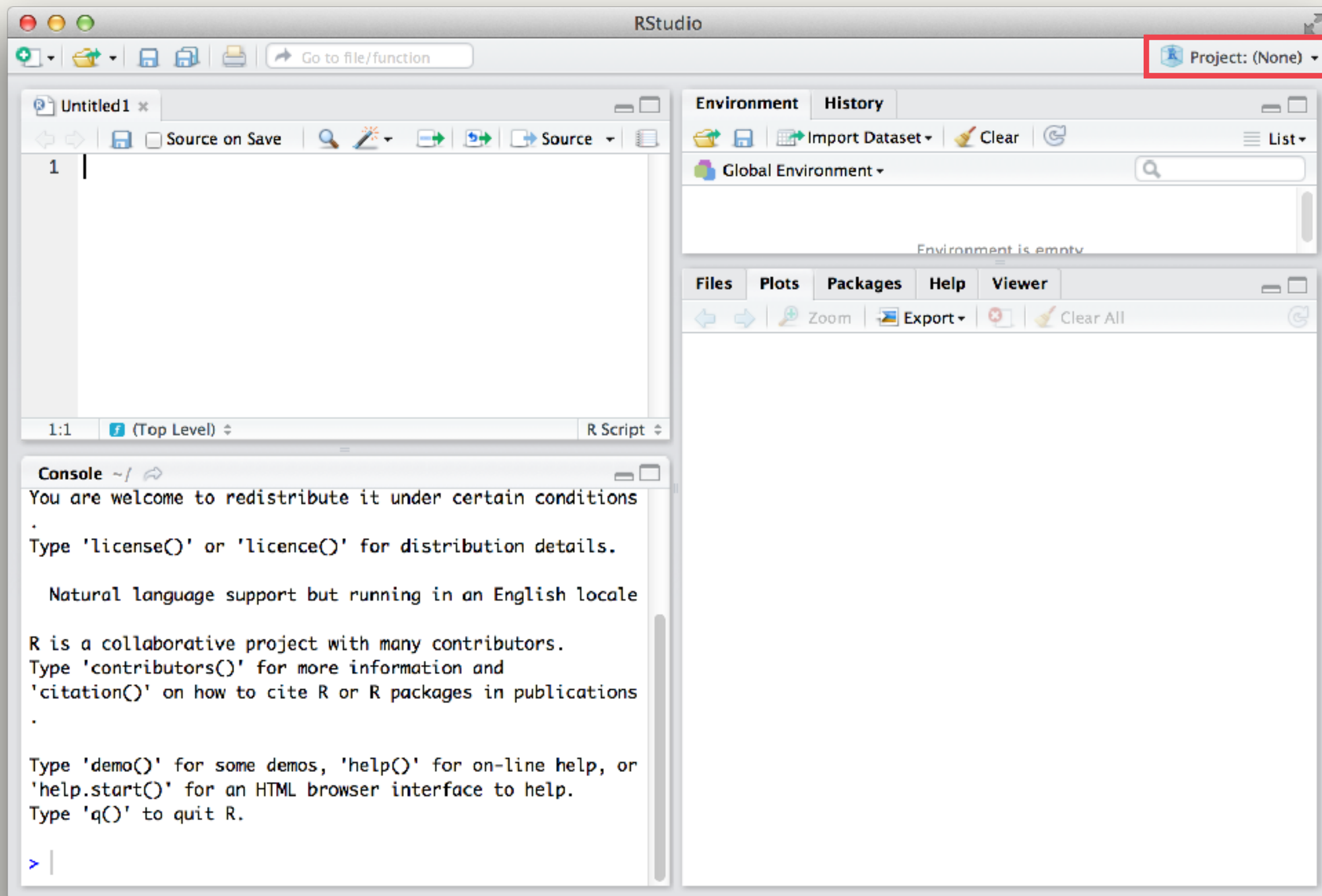
Start from where you left off

Easily manage working directories

Enhanced navigation

Not using RStudio?

Change working
directories instead



Ctrl + . = find functions/files

The screenshot shows the RStudio interface with the 'Go to File/Function' search results for the term 'summarise'. The search results list several files and functions:

- SummarisedVariable.h
- summarise (R/manip.r)
- summarise.data.frame (R/tbl-data-frame.R)
- summarise.data.table (R/manip-dt.r)
- summarise.grouped_dt (R/manip-grouped-dt.r)
- summarise.tbl_cube (R/manip-cube.r)
- summarise.tbl_dt (R/manip-dt.r)
- summarise.tbl_sql (R/manip-sql.r)
- summarise_impl (R/RcppExports.R)

The main editor shows the following R code:

```
1 example <- data.frame(ID=c(10,20,30,40,50),
2   V1=c("A","B","A",NA,"C"),
3   V2=c("Dog","Cat",NA,"Cat","Bunny"),
4   V3=c("Yes","No","No","Yes","No"),
5   V4=c("No",NA,"No","No","Yes"),
6   V5=c("No","Yes","Yes",NA,"No"))
7
8
9 tables <- lapply(example[-1], table, useNA="ifany")
10
11 fix_names <- function(x) {
12   names(x)[is.na(names(x))] <- "<NA>"
13   x
14 }
15 lapply(tables, fix_names)
16
```

The console shows the following output:

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications
.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

The viewer pane shows the R documentation for 'List Vignettes in an HTML Browser' from the 'utils' package. The title is 'List Vignettes in an HTML Browser'. The description states: 'List available vignettes in an HTML browser with links to PDF, LaTeX/noweb source, and (tangled) R code (if available)'. The usage section shows the function signature: `browseVignettes(package = NULL, lib.loc = NULL, all = FALSE)`.

F2 = jump to definition

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Displays the source code for a function named `summarise.grouped_dt`. The function is defined as follows:

```
55 data = out,  
56 vars = groups(.data)  
57 )  
58 }  
59  
60 #' @rdname manip_grouped_dt  
61 #' @export  
62 summarise.grouped_dt <- function(.data, ...) {  
63   # Set keys, if needed  
64   keys <- deparse_all(groups(.data))  
65   if (!identical(keys, key(.data))) {  
66     setkeyv(.data, keys)  
67   }  
68  
69   cols <- named_dots(...)  
70   # Replace n() with .N  
71   for (i in seq_along(cols)) {
```

The function `deparse_all` is highlighted with a red box.
The status bar at the bottom of the script editor shows the cursor is at line 64, column 17, within the `summarise.grouped_dt` function.- Environment Pane:** Located at the top right, it shows the current environment with tabs for Environment, History, Git, and Build. The Environment tab is active, showing a list of objects and their attributes.- Viewer Pane:** Located at the bottom right, it displays the help page for the `browseVignettes` function. The title is "List Vignettes in an HTML Browser". The description states: "List available vignettes in an HTML browser with links to PDF, LaTeX/noweb source, and (tangled) R code (if available).". The usage section shows the function signature: `browseVignettes(package = NULL, lib.loc = NULL, all =`.


```

13* x  Untitled4* x  manip-grouped-dt.r x  utils.r x
Source on Save  Source
91 wrap <- function(...) {
92   string <- paste0(...)
93   wrapped <- strwrap(string, width = getOption
94     ("width"), exdent = 2)
95   paste0(wrapped, collapse = "\n")
96 }
97 deparse_all <- function(x) {
98   deparse2 <- function(x) paste(deparse(x, width
99     .cutoff = 500L), collapse = "")
100   vapply(x, deparse2, FUN.VALUE = character(1))
101 }
102 commas <- function(...) paste0(..., collapse = ", ")
103
104 in_travis <- function() identical(Sys.getenv
105   ("TRAVIS"), "true")
97:1  deparse_all  R Script

```

Console ~/Documents/plyr/dplyr/

R is a collaborative project with many contributors.
 Type 'contributors()' for more information and
 'citation()' on how to cite R or R packages in publications
 .

Type 'demo()' for some demos, 'help()' for on-line help, or
 'help.start()' for an HTML browser interface to help.
 Type 'q()' to quit R.

> |

Environment History Git Build

Build & Reload Check More

Files Plots Packages Help Viewer

R: List Vignettes in an HTML Browser Find in Topic

browseVignettes {utils}

R Documentation

List Vignettes in an HTML Browser

Description

List available vignettes in an HTML browser with links to PDF, LaTeX/noweb source, and (tangled) R code (if available).

Usage

```
browseVignettes(package = NULL, lib.loc = NULL, all =
```

~/Documents/plyr/dplyr - RStudio

Go to file/function

13* x Untitled4* x manip-grouped-dt.r x utils.r x

Source on Save Source

```
91 wrap <- function(...) {
92   string <- paste0(...)
93   wrapped <- strwrap(string, width = getOption
94     ("width"), exdent = 2)
95   paste0(wrapped, collapse = "\n")
96 }
97 deparse_all <- function(x) {
98   deparse2 <- function(x) paste(deparse(x, width
99     .cutoff = 500L), collapse = "")
100   vapply(x, deparse2, FUN.VALUE = character(1))
101 }
102 commas <- function(...) paste0(..., collapse = ", ")
103
104 in_travis <- function() identical(Sys.getenv
105   ("TRAVIS"), "true")
```

97:1 deparse_all R Script

Environment History Git Build

Build & Reload Check More

Files Plots Packages Help Viewer

R: List Vignettes in an HTML Browser Find in Topic

browseVignettes {utils} R Documentation

List Vignettes in an HTML Browser

Description

List available vignettes in an HTML browser with links to PDF, LaTeX/noweb source, and (tangled) R code (if available).

Usage

```
browseVignettes(package = NULL, lib.loc = NULL, all =
```

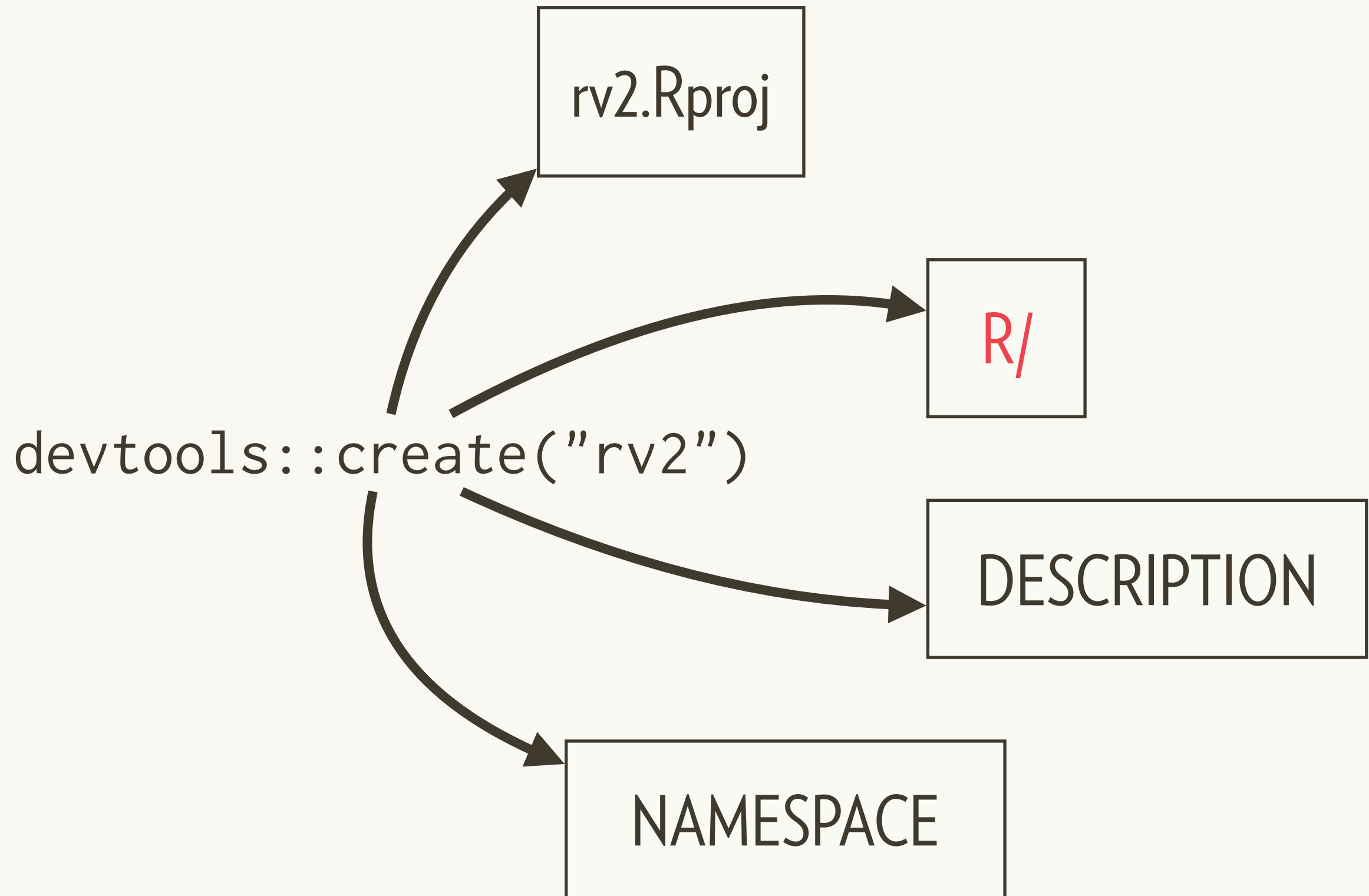
Console ~/Documents/plyr/dplyr/

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications
.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |

What happens we run create?

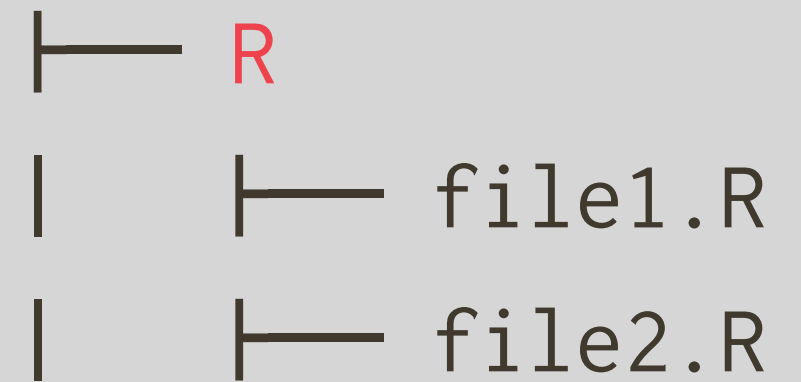




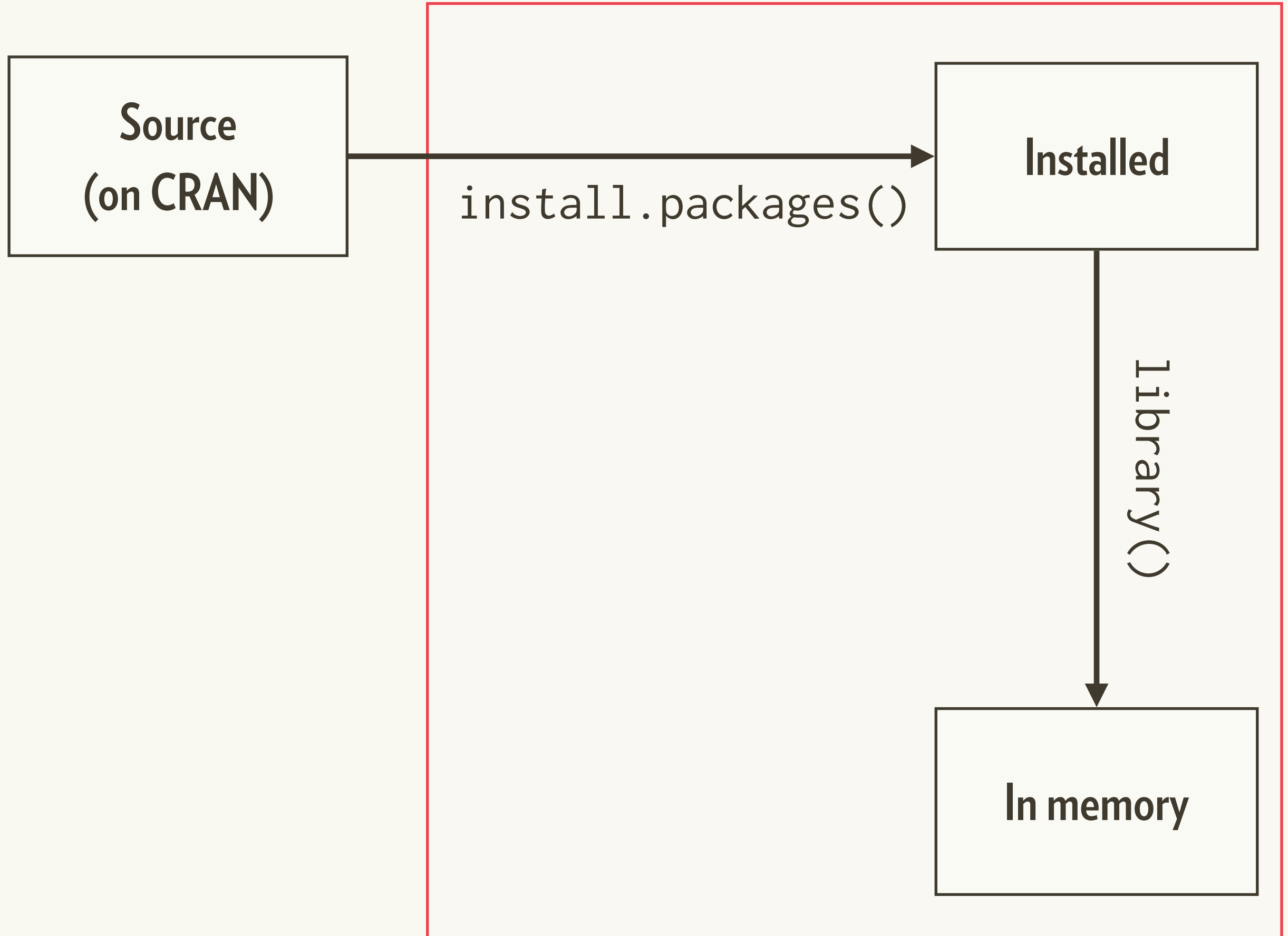
Where your code lives

A root directory (rv2/)

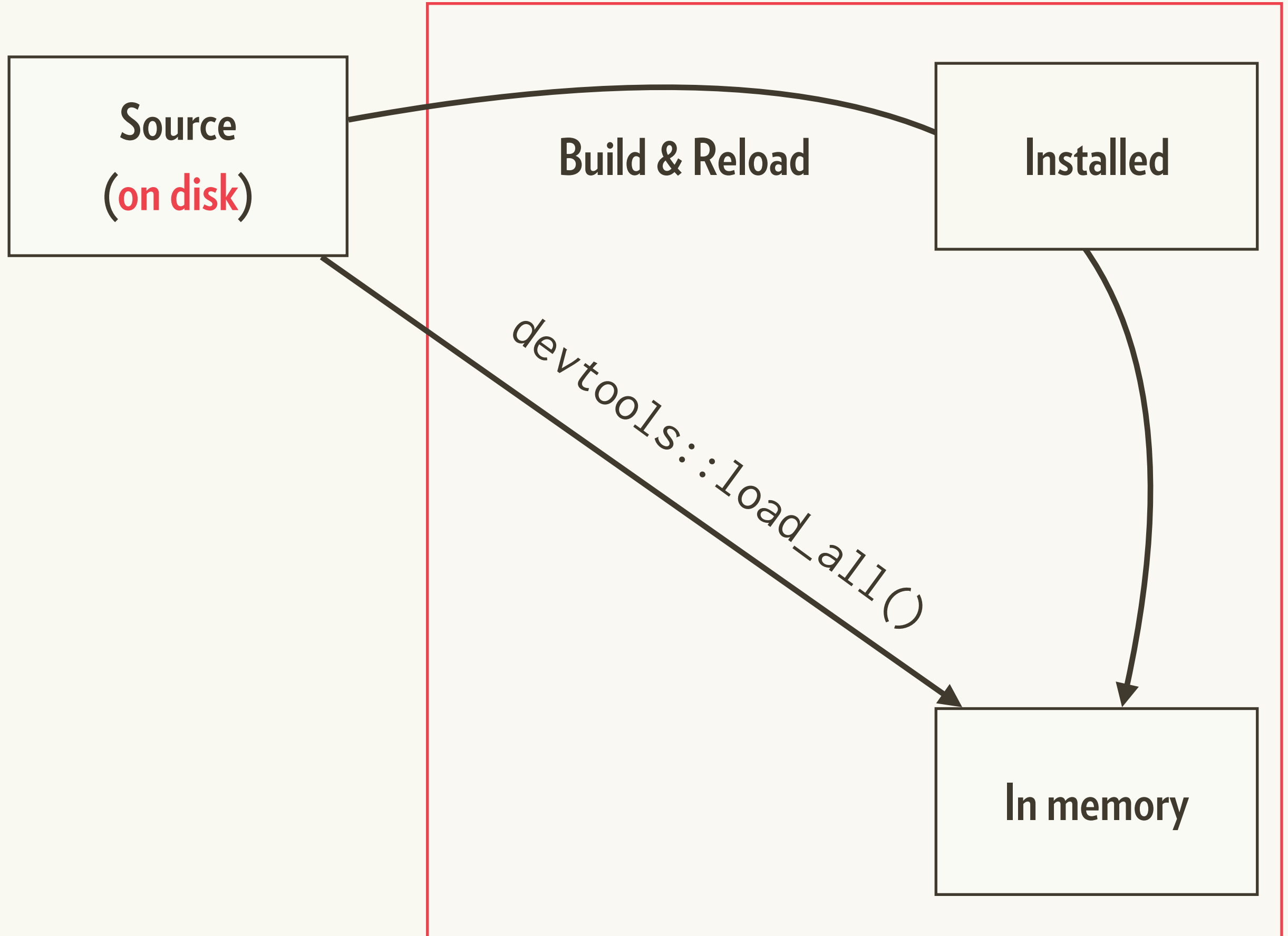
A directory of R code (rv2/R/)

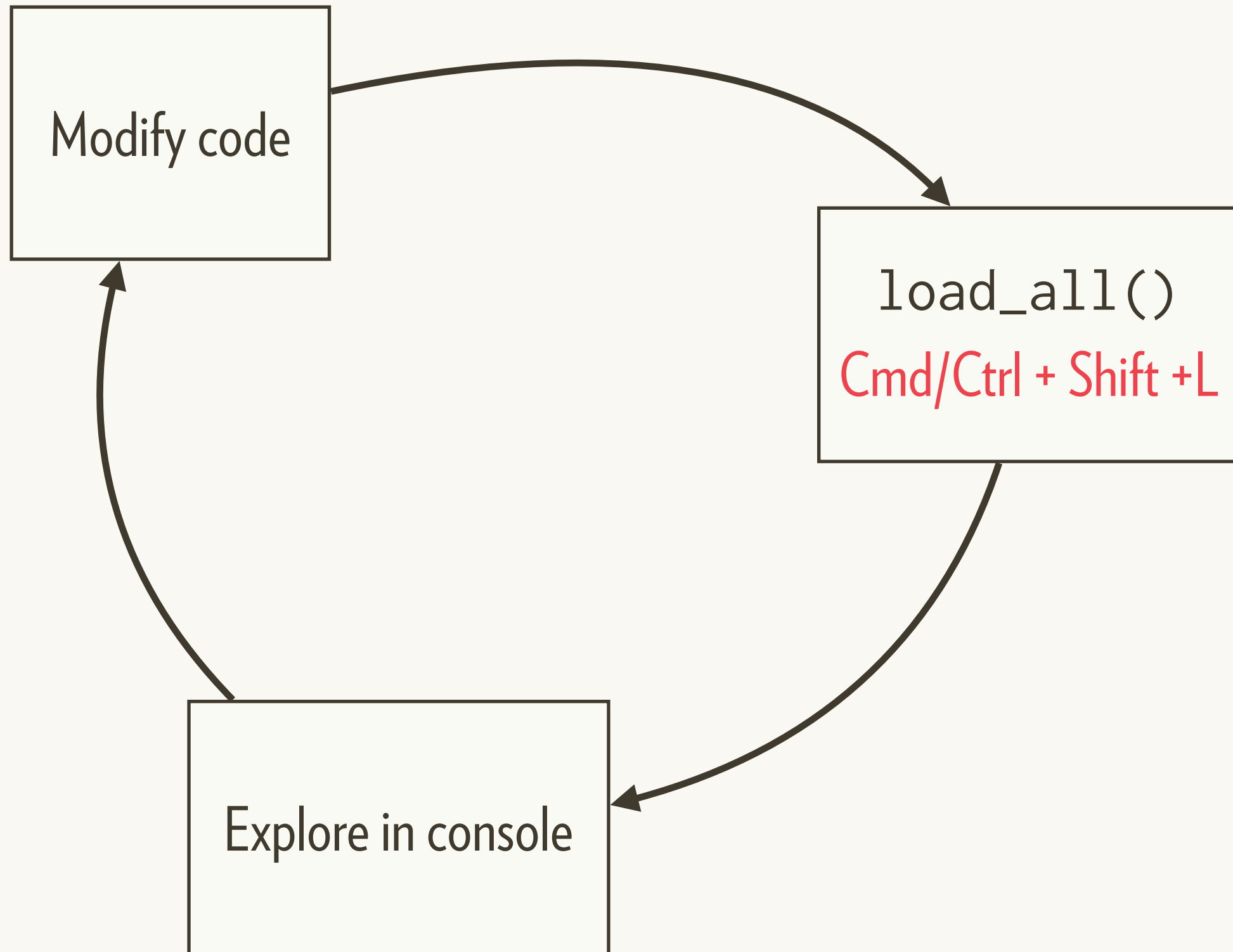


To load code into memory you've previously used



But this package isn't on CRAN





You don't even need
to save your code!

Your turn

Create a new package

```
devtools::create("path/to/rv2/")
```

Open rv2.Rproj

Add code on following slides into the package.

Load the code with Cmd/Ctrl + Shift + L, or

```
devtools::load_all()
```

Make sure the demo code works

```
rv <- function(x, probs = NULL) {  
  if (is.null(probs)) {  
    probs <- rep(1, length(x)) / length(x)  
  }  
  
  structure(x, probs = probs, class = "rv")  
}
```

```
probs <- function(x) attr(x, "probs")
```



```
print.rv <- function(x, ...) {  
  X <- format(x, digits = 3)  
  P <- format(probs(x), digits = 3)  
  out <- cbind(X = X, "P(X)" = P)  
  rownames(out) <- rep("", nrow(out))  
  print(out, quote = FALSE)  
}
```

```
plot.rv <- function(x, ...) {  
  name <- deparse(substitute(x))  
  ylim <- range(0, probs(x))  
  
  plot(as.numeric(x), probs(x), type = "h", ylim = ylim,  
       xlab = name, ylab = paste0("P(", name, ")"), ...)  
  points(as.numeric(x), probs(x), pch = 20)  
  abline(h = 0, col = "gray")  
}
```

```
E <- function(x) {  
  sum(as.numeric(x) * probs(x))  
}
```

```
VAR <- function(x) E((x - E(x)) ^ 2)
```

```
SD <- function(x) sqrt(VAR(x))
```

```
Z <- function(x) (x - E(x)) / SD(x)
```

```
dice <- rv(1:6)  
dice  
plot(dice)
```

```
E(dice)  
VAR(dice)
```

Grab the slides, code and data from <http://bit.ly/advr16-2a>

Common problems

Symptoms: `load_all()` doesn't update existing function

Cause: Accidentally pressed source

Diagnosis: Check your environment pane

Cure: Restart R

Symptoms: `load_all()` doesn't load new function

Cause: Forgot to save file

Diagnosis: Check for "Untitled1"

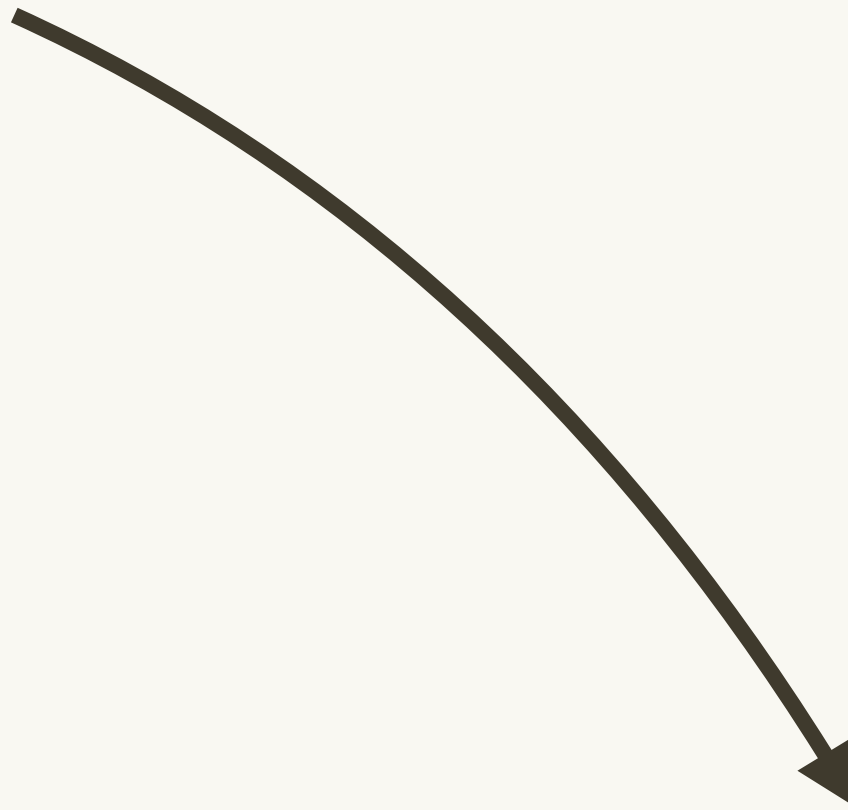
Cure: Press Cmd/Ctrl + S

Packages vs.
scripts

Script

One off data analysis

Primarily side-effects



Package

Defines reusable components

No side-effects

A function is **pure** if:

- (a) Its **output** only depends on its **inputs**
- (b) It makes **no changes** to the state of the world

1 minute: what common R functions are impure?

Functions with side-effects

Output

print()

plot()

write.csv()

Input

source()

read.csv()

Sys.time()

Other

options()

library()

install.packages()

Functions with side-effects

Avoid

library()

install.packages()

print()

source()

Isolate

read.csv()

Sys.time()

options()

plot()

write.csv()

Things to avoid

```
# Instead of print(), use message().
# Should also give some way to opt-out:
if (!quiet) {
  message("Processing file ", path)
}
if (is.null(by)) {
  by <- ...
  message("Joining by ", paste(by, collapse = ", "))
}

# Never use source() - just put the files in R/
# Instead of library/require()/install.packages()
# use DESCRIPTION - which we'll discuss next.
```

Isolate options with on.exit()

Bad!

```
options(stringsAsFactors = FALSE)
```

```
read.csv(path)
```

Most setters invisibly
return previous values

```
# Better  
old <- options(stringsAsFactors = FALSE)
```

```
on.exit(option(old), add = TRUE)
```

```
read.csv(path)
```

Best: use stringsAsFactors arguments to specific

functions (not always possible)

```
read.csv(path, stringsAsFactors = FALSE)
```

on.exit() runs regardless of how function exits

Default is to replace
previous on.exit()

```
f <- function(x) {  
  on.exit(message("Hi!"), add = TRUE)  
  
  if (x < 0) {  
    stop("!")  
  } else {  
    10  
  }  
}  
f(-10)  
f(10)
```

Good practice to clean up after yourself

```
f <- function(x) {  
  tmp <- tempfile()  
  on.exit(unlink(tmp), add = TRUE)  
  
  old_par <- par(bg = "red")  
  on.exit(par(old_par), add = TRUE)  
  plot(1:10)  
}
```

Don't mix side-effects and computation

```
fortify.lm <- function(model, data = model$model, ...) {  
  infl <- influence(model, do.coef = FALSE)  
  data$.hat <- infl$hat  
  data$.sigma <- infl$sigma  
  data$.cooks <- cooks.distance(model, infl)  
  
  data$.fitted <- predict(model)  
  data$.resid <- resid(model)  
  data$.stdresid <- rstandard(model, infl)  
  
  data  
}
```

See also <https://github.com/dgrtwo/broom>

Improve this function

Back at 10:50am

```
plot_function <- function(f, xlim = c(0, 1), n = 100) {  
  x <- seq(xlim[1], xlim[2], length.out = n)  
  y <- f(x)  
  
  print(paste0("Using ", n, " points"))  
  par(bg = "grey90")  
  plot(x, y, xlab = "x", ylab = "f(x)", type = "l")  
}
```

```
plot_function(sin)  
plot(runif(5))
```

```
grid_function <- function(f, xlim = NULL, n = NULL) {  
  if (is.null(xlim)) {  
    xlim <- c(0, 1)  
    message("Using xlim: ", xlim[1], "-", xlim[2])  
  } else {  
    if (!is.numeric(xlim) || length(xlim) != 2) {  
      stop("`xlim` must be a numeric vector of length 2")  
    }  
  }  
  
  if (is.null(n)) {  
    n <- 100  
    message("Using ", n, " points")  
  }  
  
  x <- seq(xlim[1], xlim[2], length.out = n)  
  y <- f(x)  
  
  data.frame(x, y)  
}
```



```
grid_function <- function(f, xlim = c(0, 1), n = 100, quiet = FALSE) {  
  if (!quiet) {  
    message("Using xlim: ", xlim[1], "-", xlim[2])  
  }  
  
  if (!quiet) {  
    message("Using ", n, " points")  
  }  
  
  x <- seq(xlim[1], xlim[2], length.out = n)  
  y <- f(x)  
  
  data.frame(x, y)  
}
```

```
plot_function <- function(f, xlim = c(0, 1), n = NULL) {  
  fun <- grid_function(f, xlim, n)  
  
  old <- par(bg = "grey90")  
  on.exit(par(old), add = TRUE)  
  
  plot(fun$x, fun$y, xlab = "x", ylab = "f(x)",  
        type = "l")  
}
```


This work is licensed under the
Creative Commons Attribution-Noncommercial 3.0
United States License.

To view a copy of this license, visit
<http://creativecommons.org/licenses/by-nc/3.0/us/>