R/

December 2016

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!

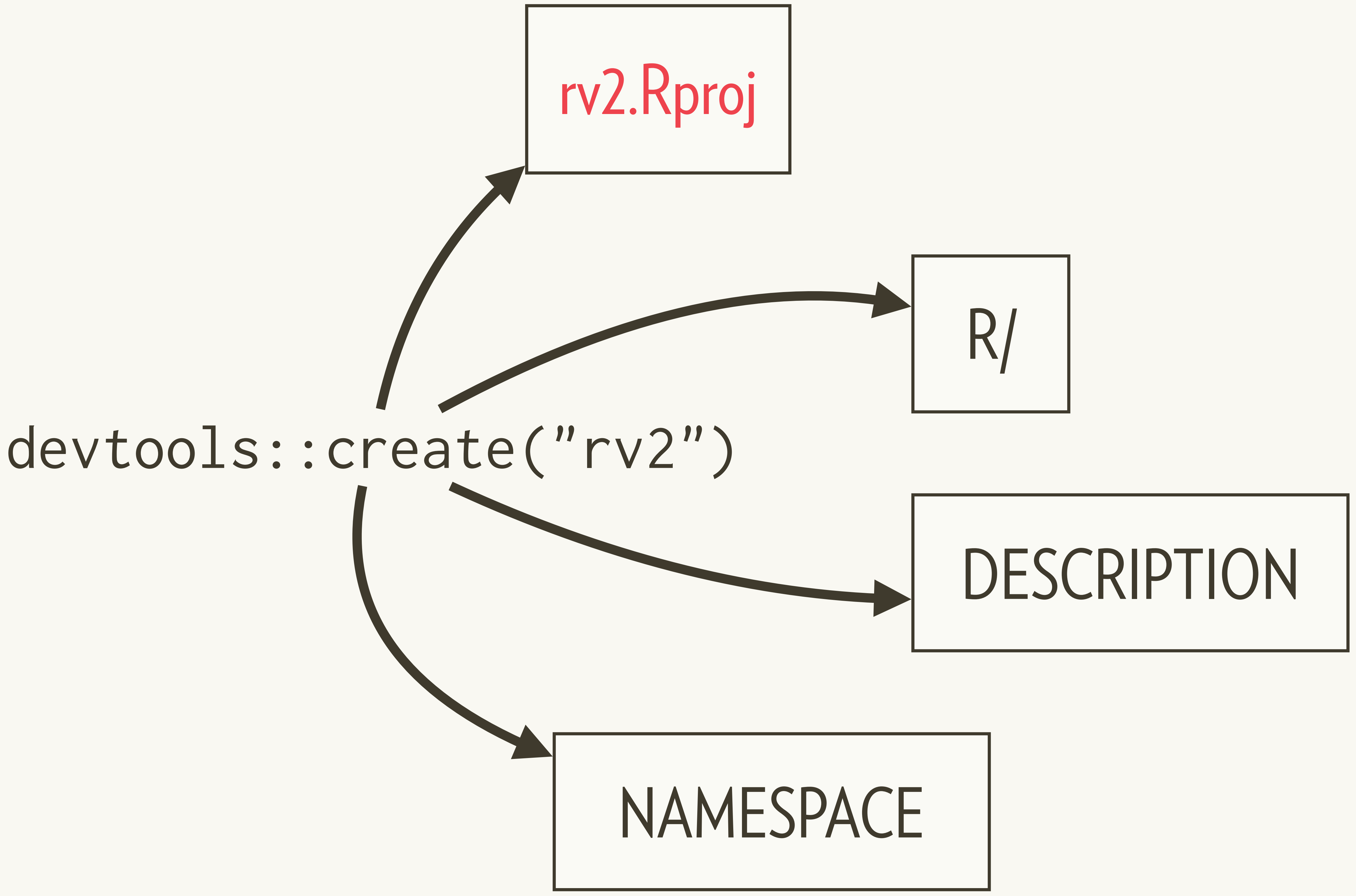
Once you have a name you can create the package

```
# 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)

Isolate code and results

Multiple projects open

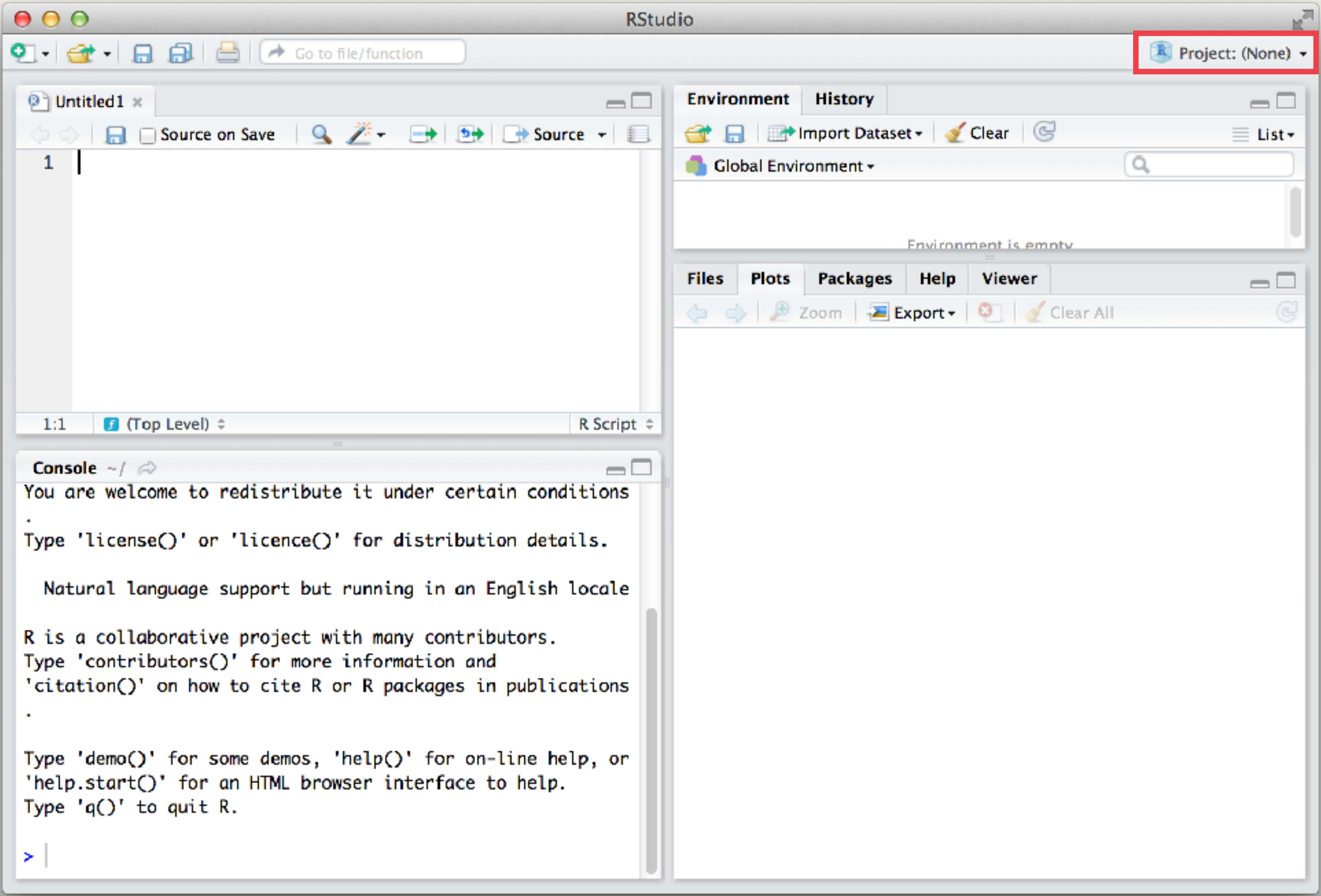
Start from where you left off

Easily manage working directories

Enhanced navigation

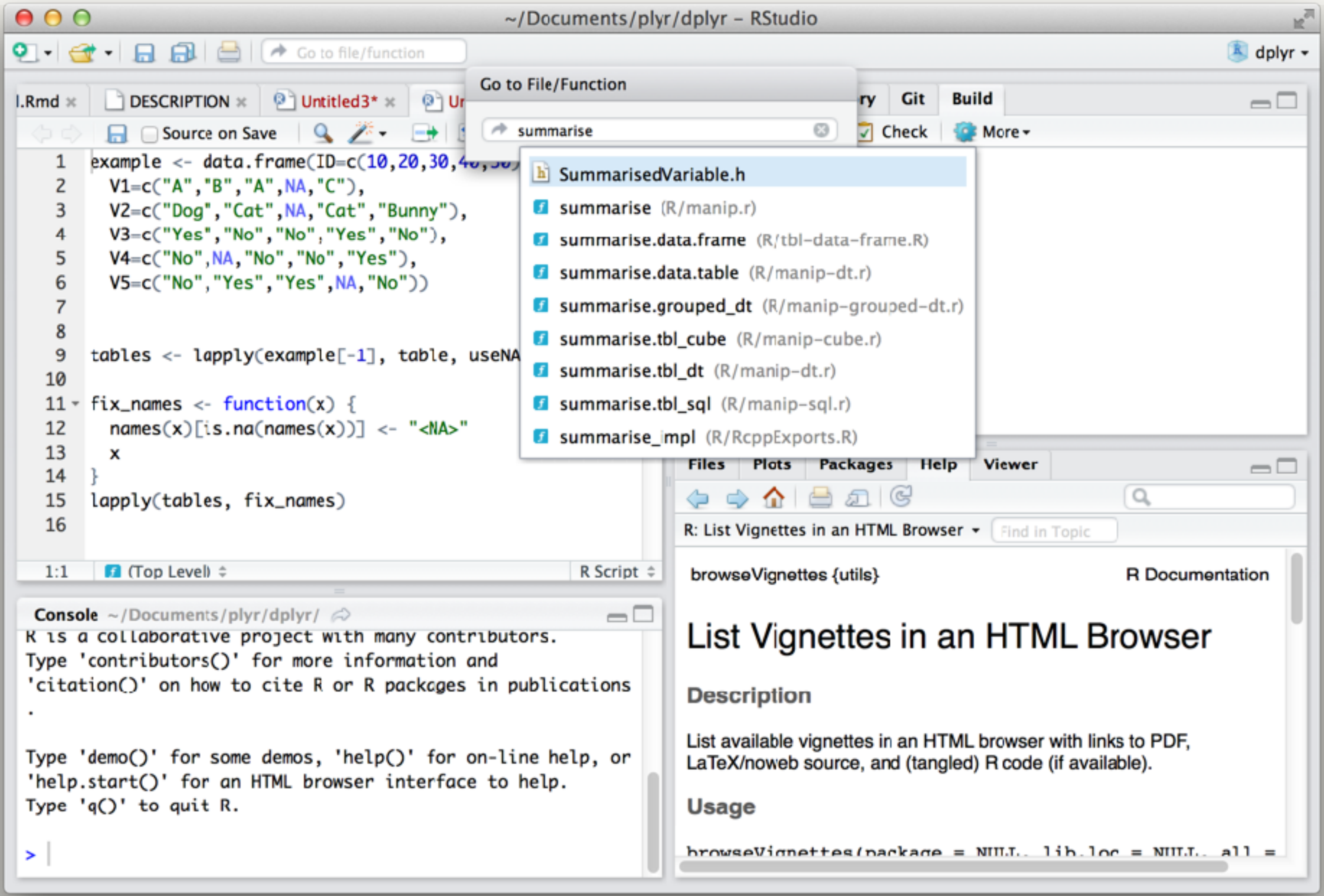
Not using RStudio?

Change working directories instead



```
You are welcome to redistribute it under certain conditions
.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

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.

>
```

Ctrl + . = find functions/files



Go to File/Function

summarise

- 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)

```r
example <- data.frame(ID=c(10,20,30,40,50),
  V1=c("A","B","A",NA,"C"),
  V2=c("Dog","Cat",NA,"Cat","Bunny"),
  V3=c("Yes","No","No","Yes","No"),
  V4=c("No",NA,"No","No","Yes"),
  V5=c("No","Yes","Yes",NA,"No"))


tables <- lapply(example[-1], table, useNA

fix_names <- function(x) {
  names(x)[is.na(names(x))] <- "<NA>"
  x
}
lapply(tables, fix_names)
```

```
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.

> 
```

browseVignettes {utils} R Documentation

List Vignettes in an HTML Browser

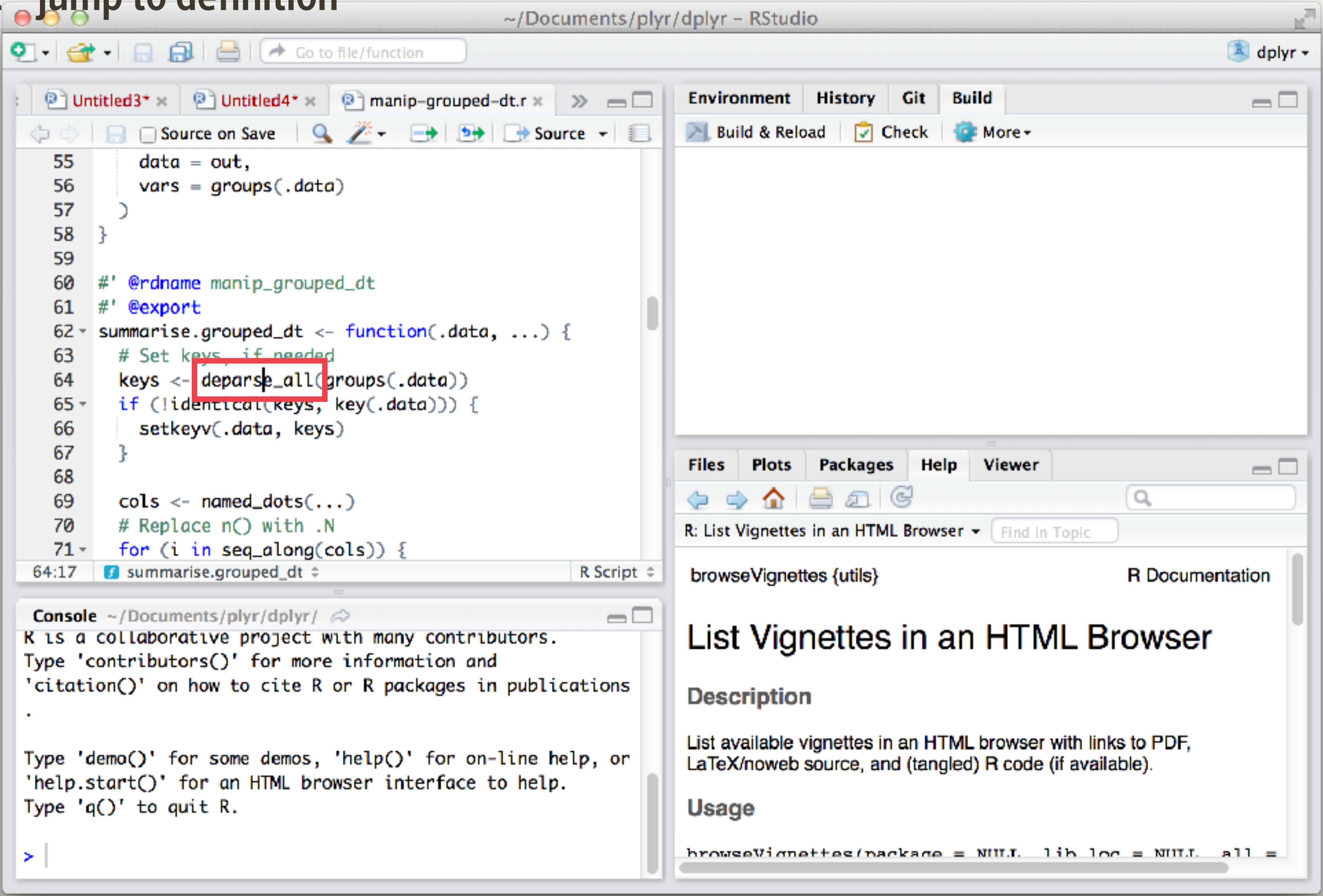
Description

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 =
```

F2 = jump to definition



```
    data = out,
    vars = groups(.data)
  )
}

#' @rdname manip_grouped_dt
#' @export
summarise.grouped_dt <- function(.data, ...) {
  # Set keys, if needed
  keys <- deparse_all(groups(.data))
  if (!identical(keys, key(.data))) {
    setkeyv(.data, keys)
  }

  cols <- named_dots(...)
  # Replace n() with .N
  for (i in seq_along(cols)) {
```

```
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.

>
```

browseVignettes {utils} R Documentation

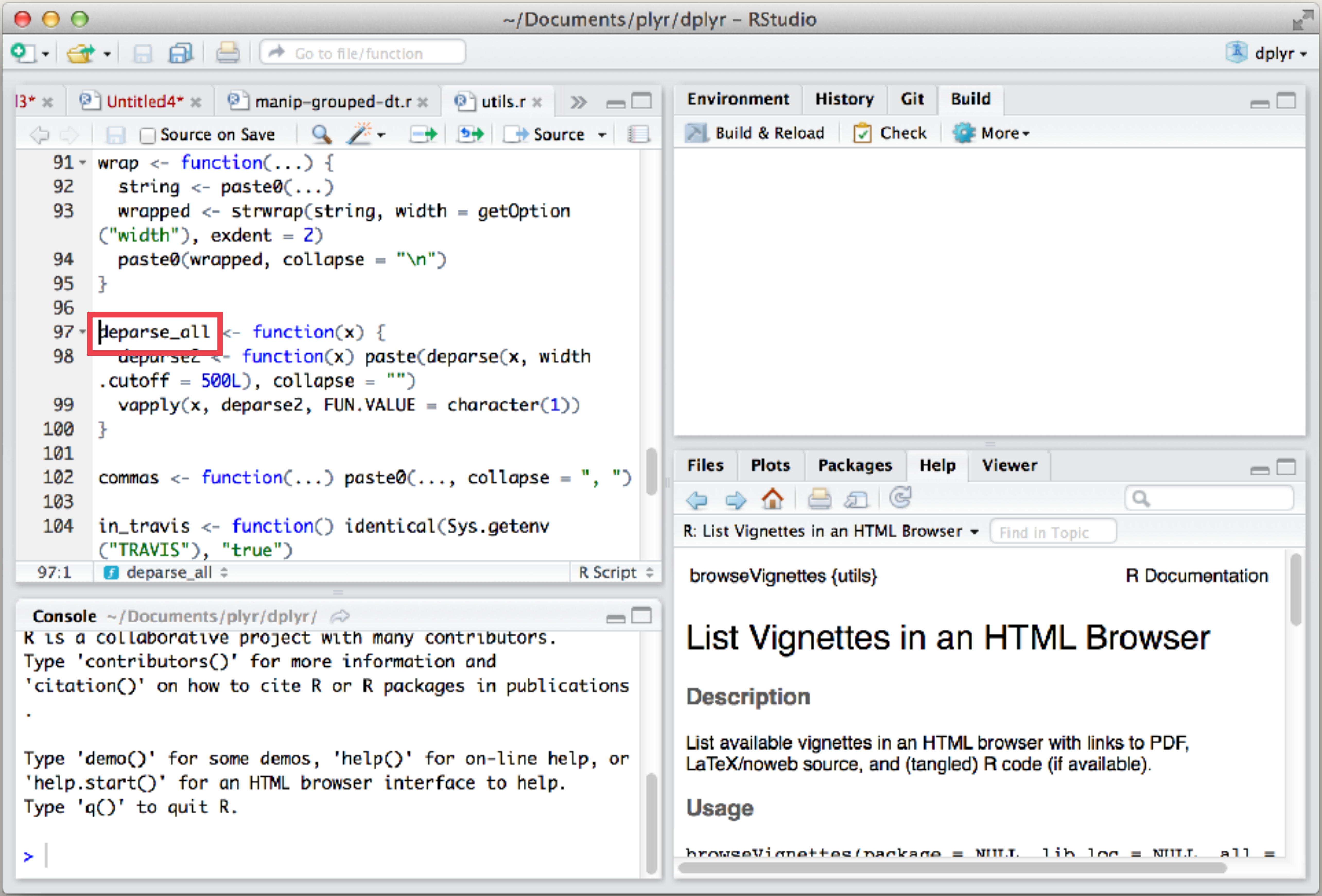
List Vignettes in an HTML Browser

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 =
```



```
wrap <- function(...) {
  string <- paste0(...)
  wrapped <- strwrap(string, width = getOption
("width"), exdent = 2)
  paste0(wrapped, collapse = "\n")
}

deparse_all <- function(x) {
  deparse2 <- function(x) paste(deparse(x, width
.cutoff = 500L), collapse = "")
  vapply(x, deparse2, FUN.VALUE = character(1))
}

commas <- function(...) paste0(..., collapse = ", ")

in_travis <- function() identical(Sys.getenv
("TRAVIS"), "true")
```

```
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.

>
```

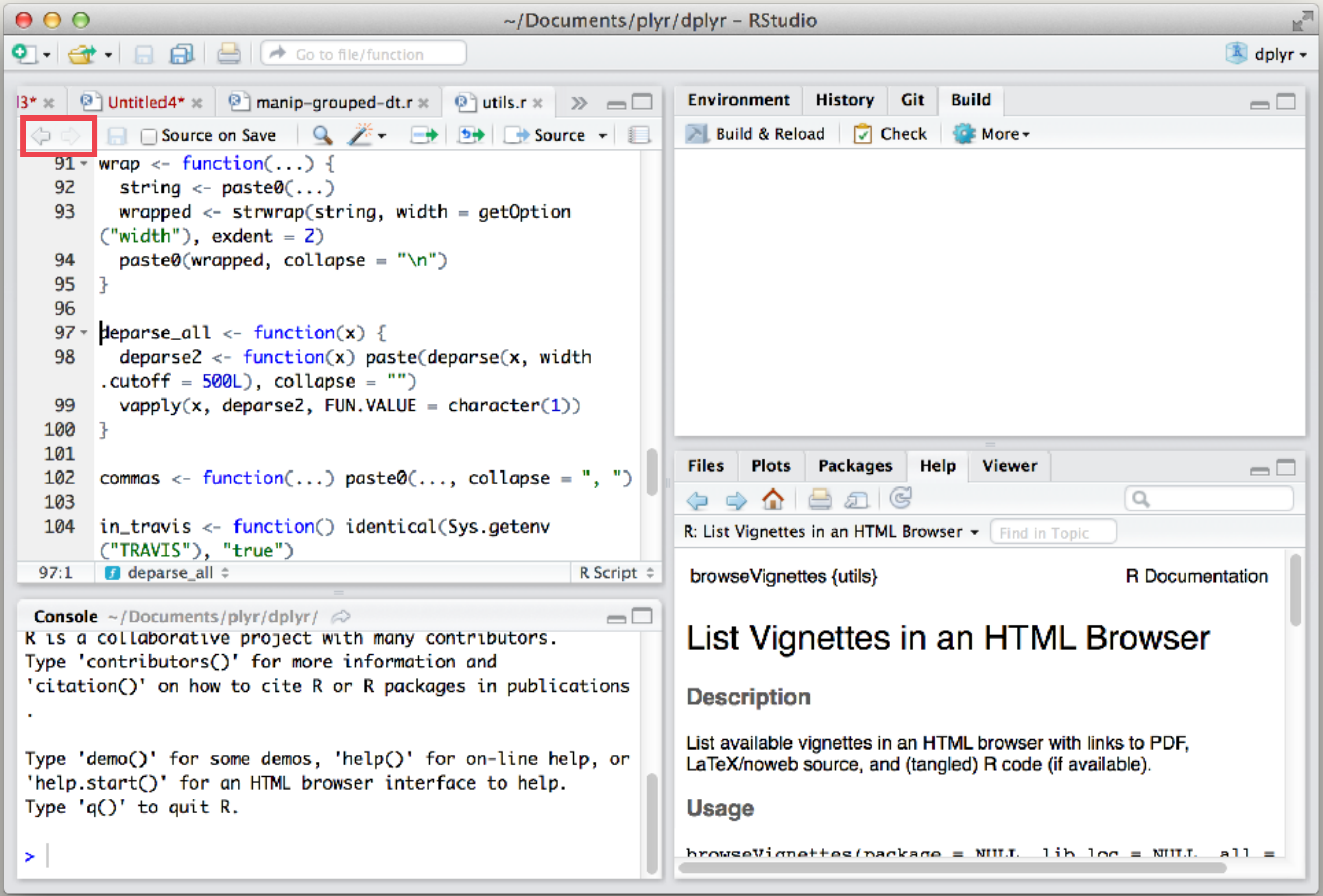
browseVignettes {utils} R Documentation

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



```r
wrap <- function(...) {
  string <- paste0(...)
  wrapped <- strwrap(string, width = getOption
("width"), exdent = 2)
  paste0(wrapped, collapse = "\n")
}

deparse_all <- function(x) {
  deparse2 <- function(x) paste(deparse(x, width
.cutoff = 500L), collapse = "")
  vapply(x, deparse2, FUN.VALUE = character(1))
}

commas <- function(...) paste0(..., collapse = ", ")

in_travis <- function() identical(Sys.getenv
("TRAVIS"), "true")
```

```
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.

>
```

browseVignettes {utils} R Documentation

List Vignettes in an HTML Browser

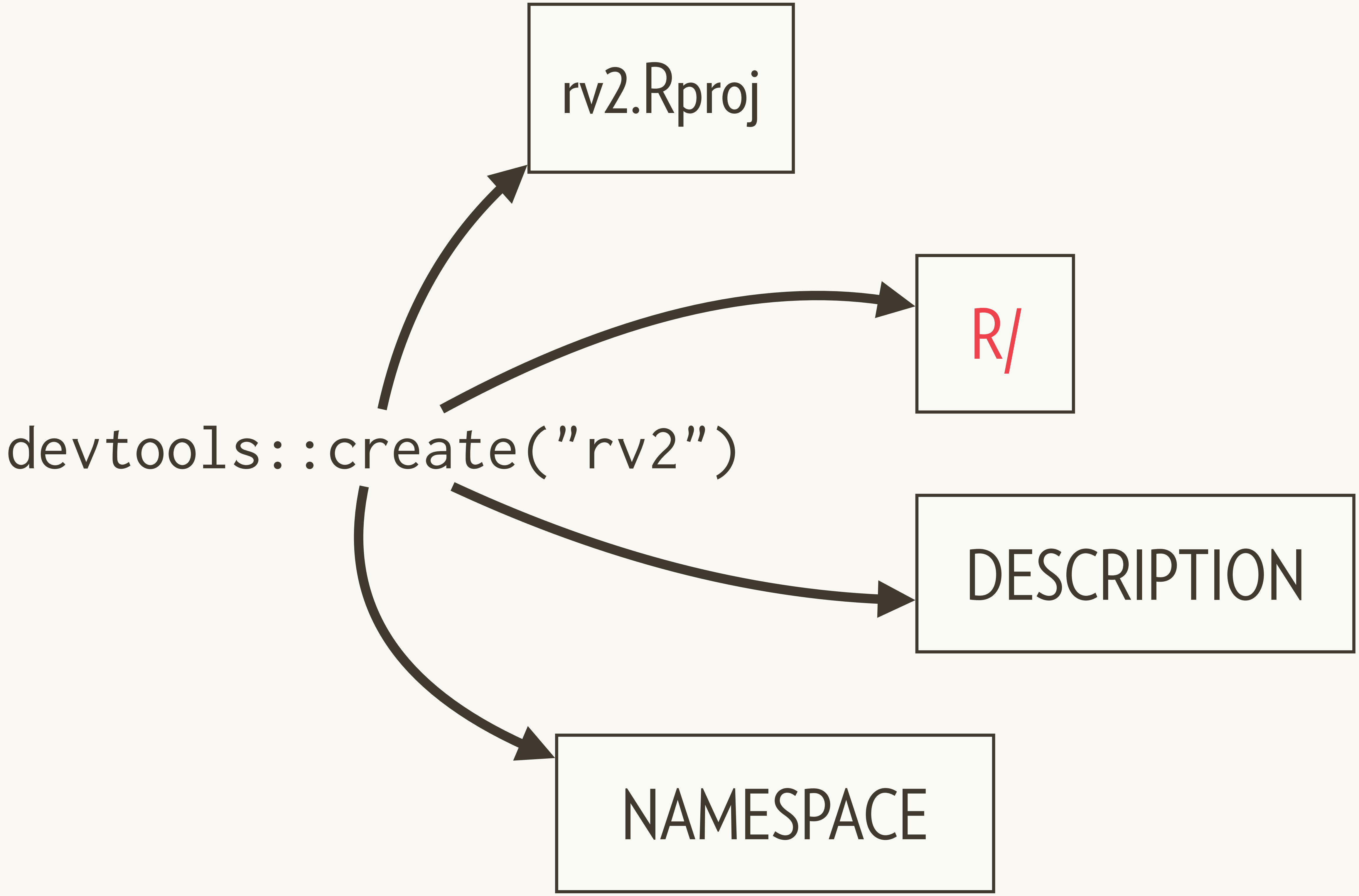
Description

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 =
```

What happens we run create?



R/

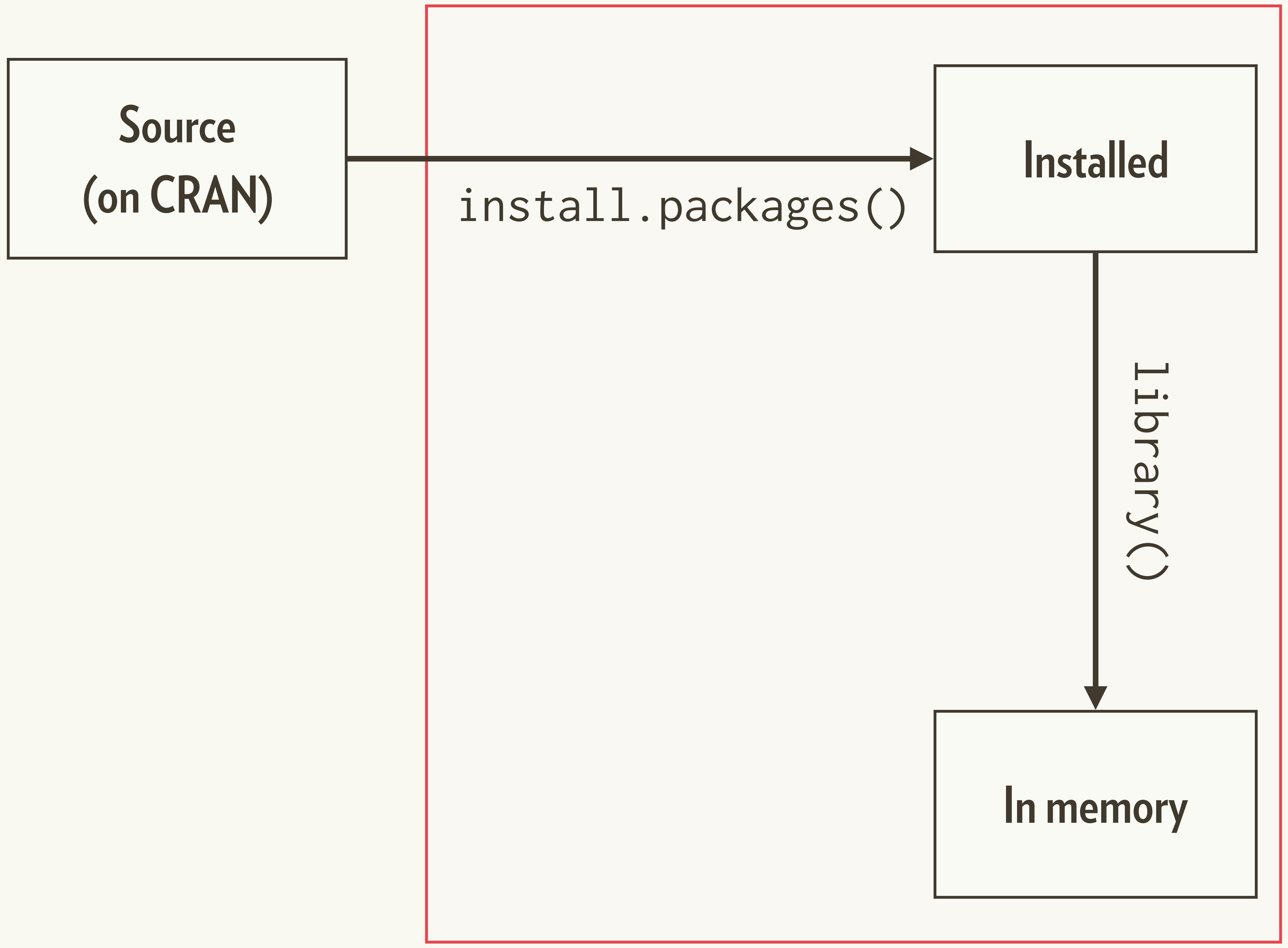
Where your code lives

A root directory (rv2/)

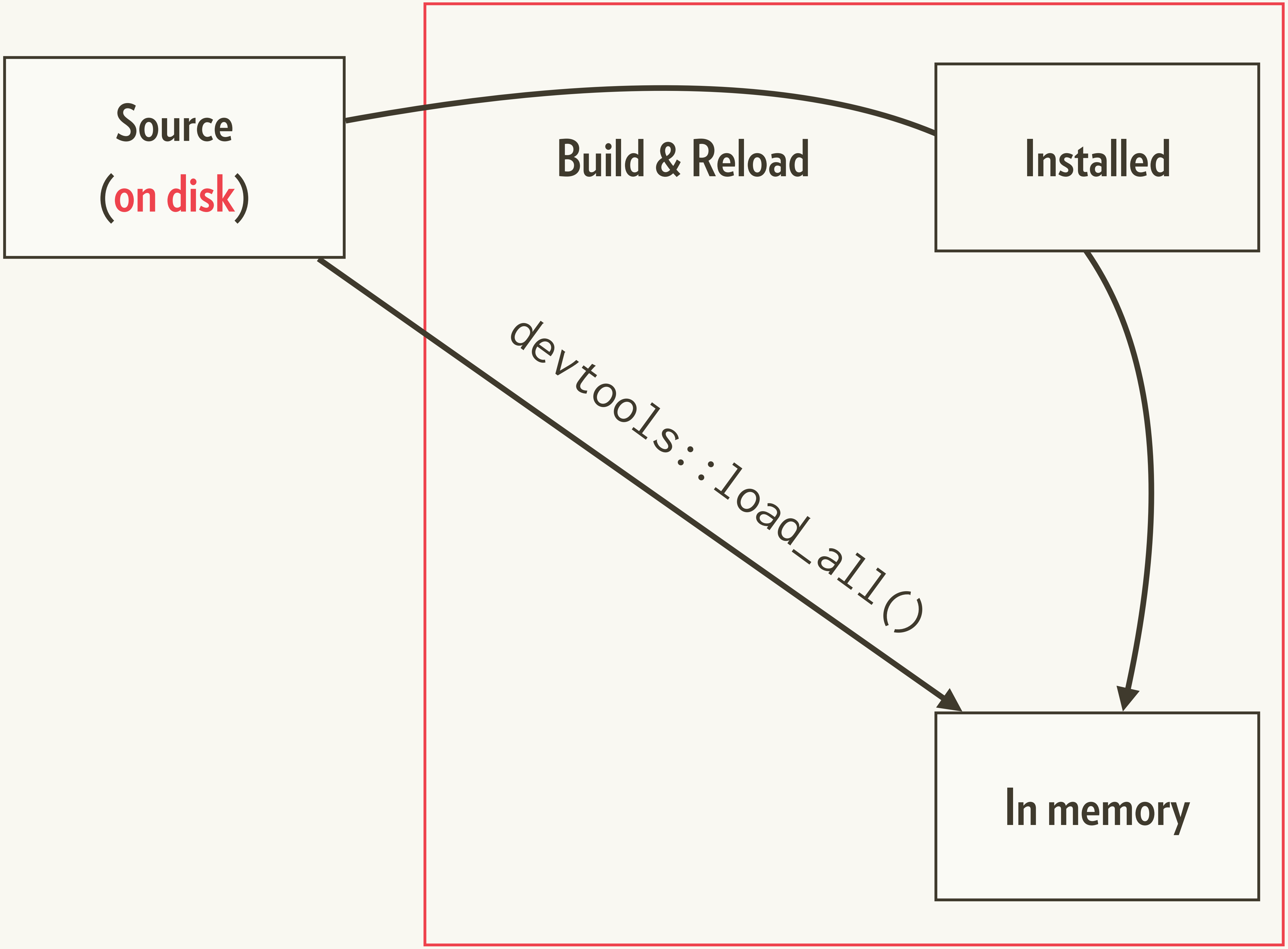
A directory of R code (rv2/R/)

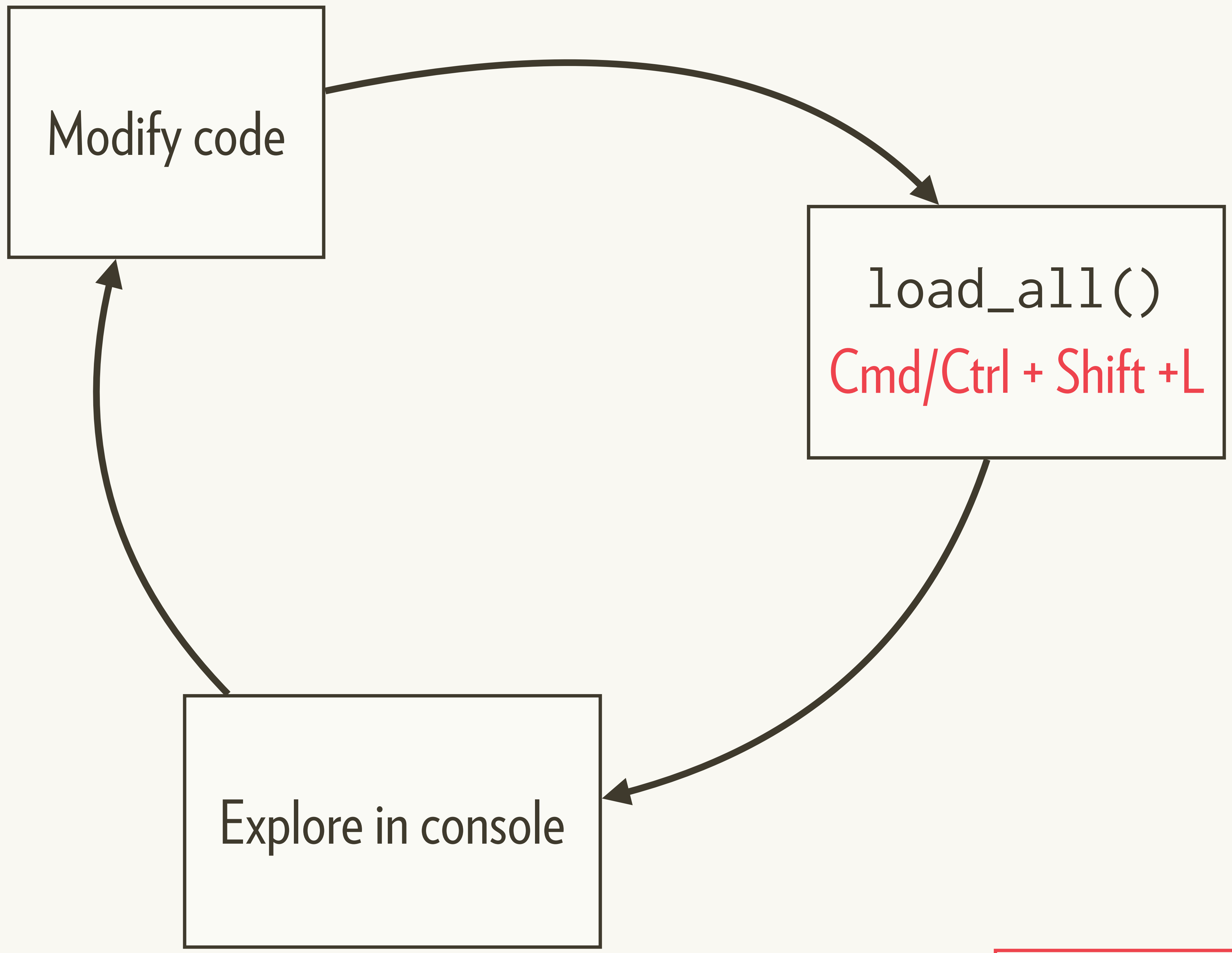
```
├── R
│   ├── file1.R
│   ├── file2.R
```

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
```

```r
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")
```

```r
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")
}
```

```r
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)
```

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()

# Isolate
source()
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()

```r
# Bad!
options(stringsAsFactors = FALSE)
read.csv(path)
```

Most setters invisibly return previous values

```r
# 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$sigma
  data$.cooksd <- 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

```r
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))
```

```r
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)
}
```

```r
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)
}
```

```r
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")
}
```