

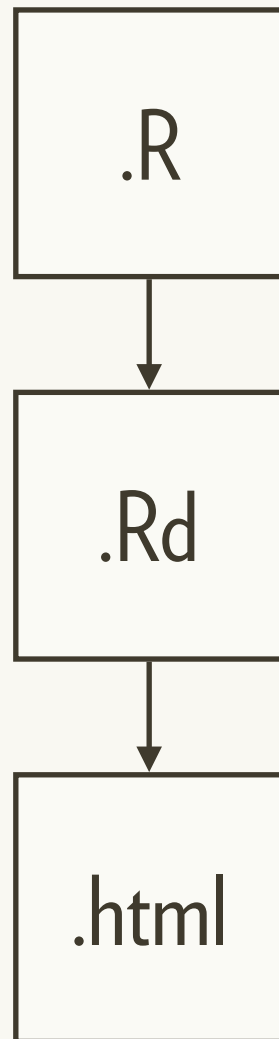
Back at 3:35pm

# Documentation

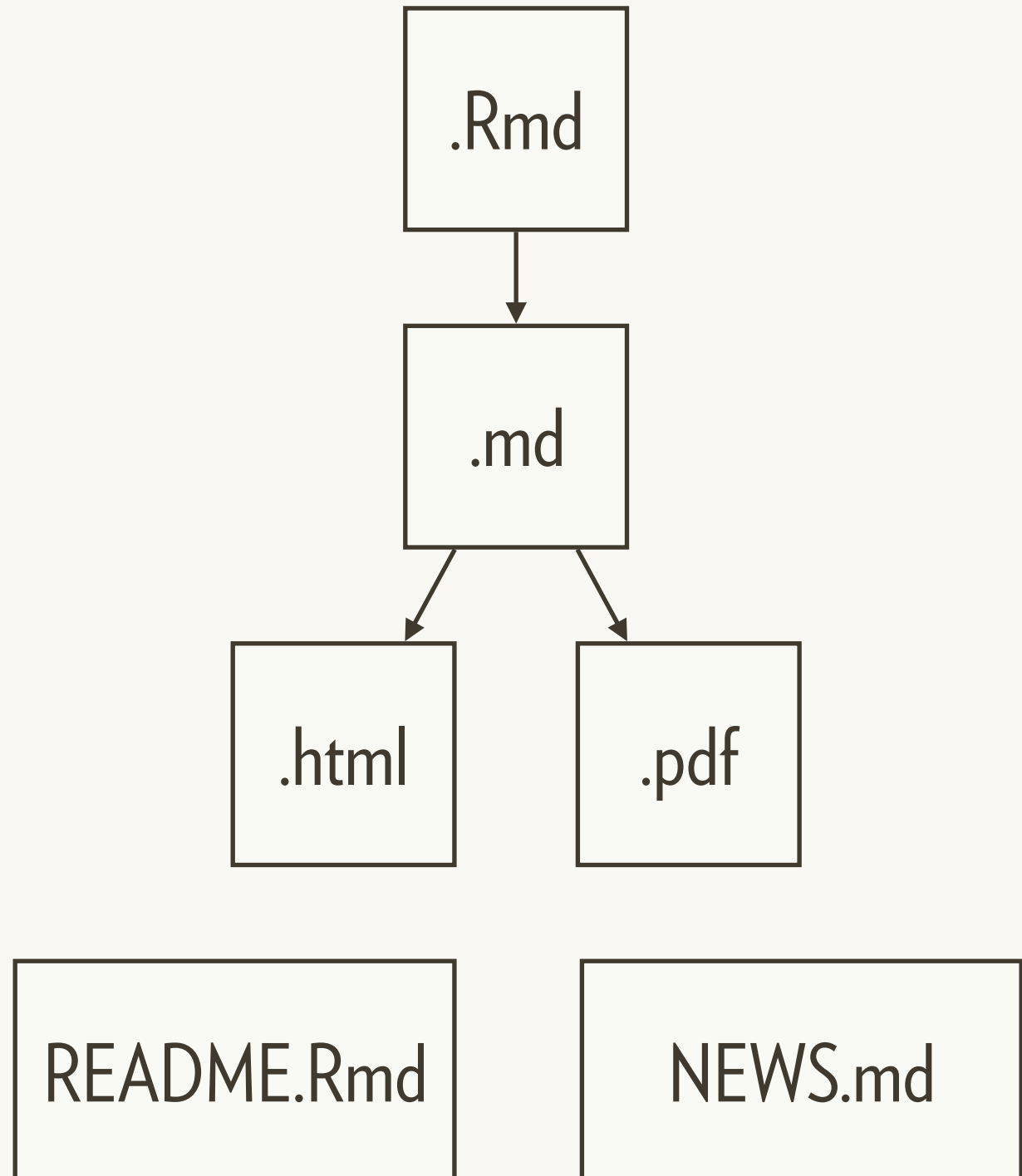
*May 2018*

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

## Function-level with roxygen2



## Package-level with rmarkdown



# Markdown

I assume you are already familiar with it

# Basic markdown formatting

# This is a top level heading

This is some text. Make text *italic* with single underscores (or stars). Make it **bold** with double stars (or underscores). Here is a [link to a markdown guide](http://bit.ly/19fAexE).

- \* This is a list

- \* This is another item

```
```R
```

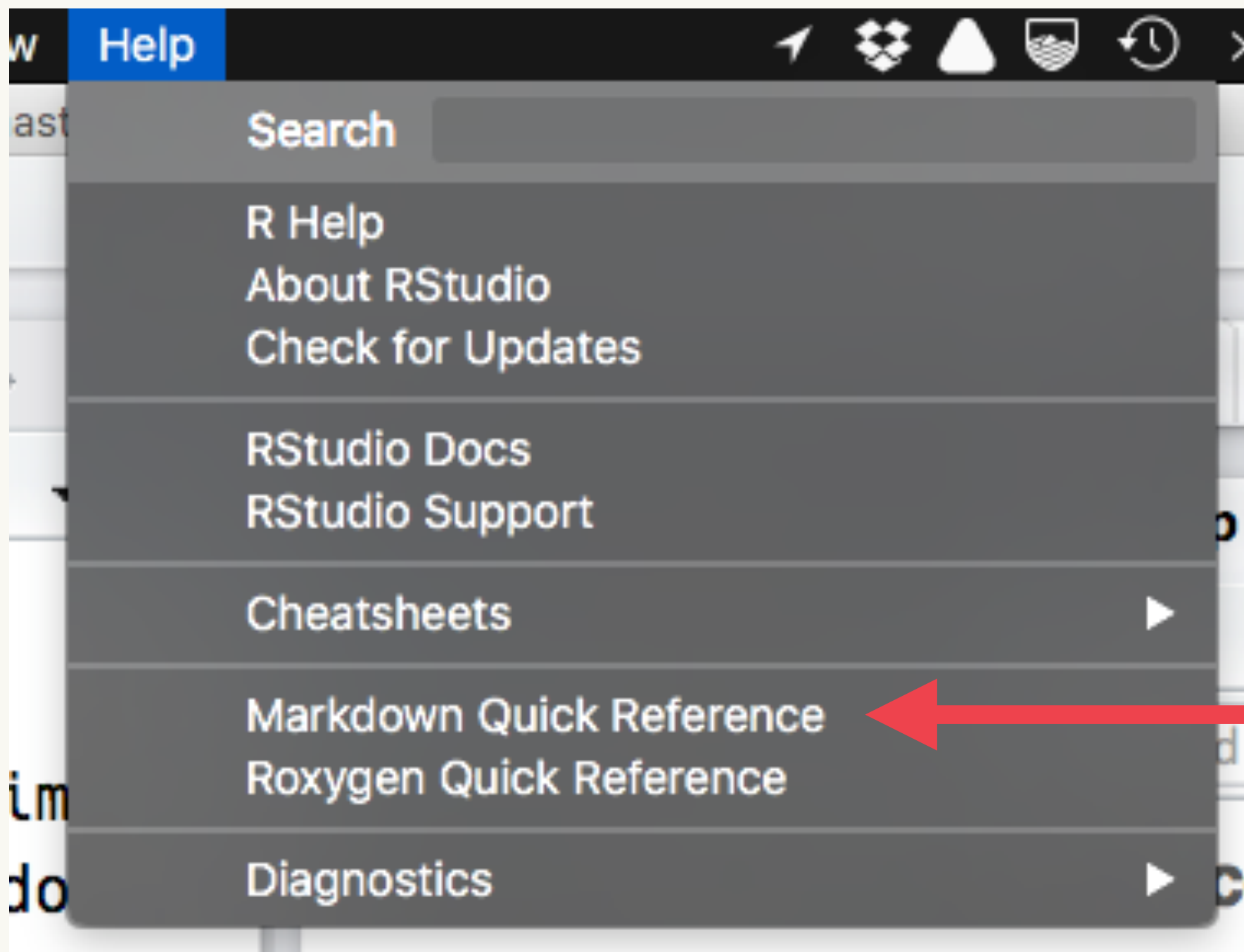
```
# Some R code
```

```
f <- function() x + 1
```

```
```
```

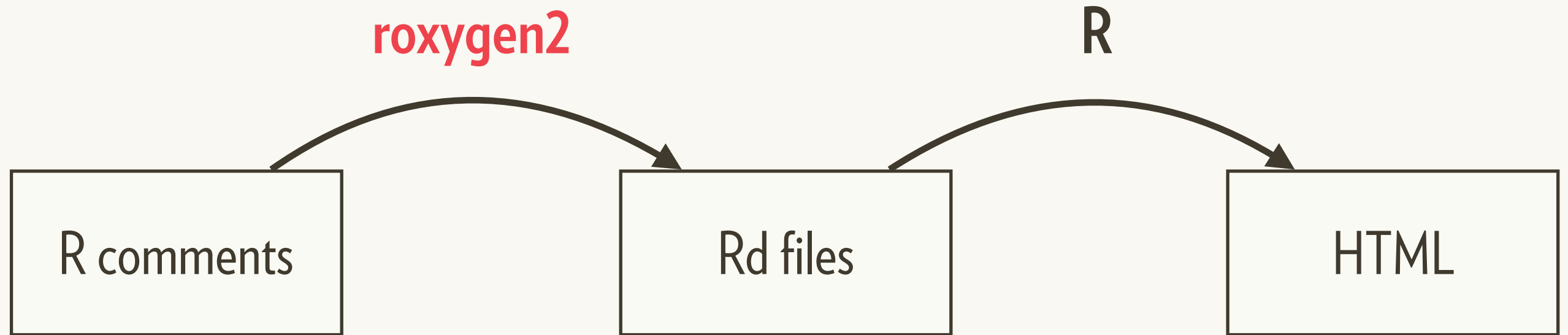
## This is a secondary heading

You can also do ``inline code``, numbered lists and quotes and more.



# Function documentation with `roxygen2`

# Roxxygen2



<http://r-pkgs.had.co.nz/man.html>

# You write specially formatted comments in .R

```
#' Add a column to a data frame
#
#' Allows you to specify the position. Will replace existing variable
#' with the same name if present.
#
#' @param x A data frame
#' @param name Name of variable to create. If a variable of that name
#' already exists it will be replaced
#' @param value Values to insert.
#' @param where Position to insert. Use 1 to insert on LHS, or -1 to insert on
#' RHS.
#' @examples
#' df <- data.frame(x = 1:5)
#' add_col(df, "y", runif(5))
#' add_col(df, "y", runif(5), where = 1)
#
#' add_col(df, "x", 5:1)
```





# Roxygen translates to **.Rd**

*In almost all cases you  
can ignore these files*

```
% Generated by roxygen2: do not edit by hand
% Please edit documentation in R/add_col.R
```

```
\name{add_col}
```

```
\alias{add_col}
```

```
\title{Add a column to a data frame}
```

```
\usage{
```

```
add_col(x, name, value, where = -1)
```

```
}
```

```
\arguments{
```

```
\item{x}{A data frame}
```

```
\item{name}{Name of variable to create. If a variable of that name  
already exists it will be replaced}
```

```
\item{value}{Values to insert.}
```

```
\item{where}{Position to insert. Use 1 to insert on LHS, or -1 to insert on  
RHS.}
```

```
}
```

```
\description{
```

```
Allows you to specify the position. Will replace existing variable  
with the same name if present.
```

```
}
```

```
add_col {hadcol}
```

R translates to  
[.html](#) for viewing

# Add a column to a data frame

## Description

Allows you to specify the position. Will replace existing variable with the same name if present.

## Usage

```
add_col(x, name, value, where = -1)
```

## Arguments

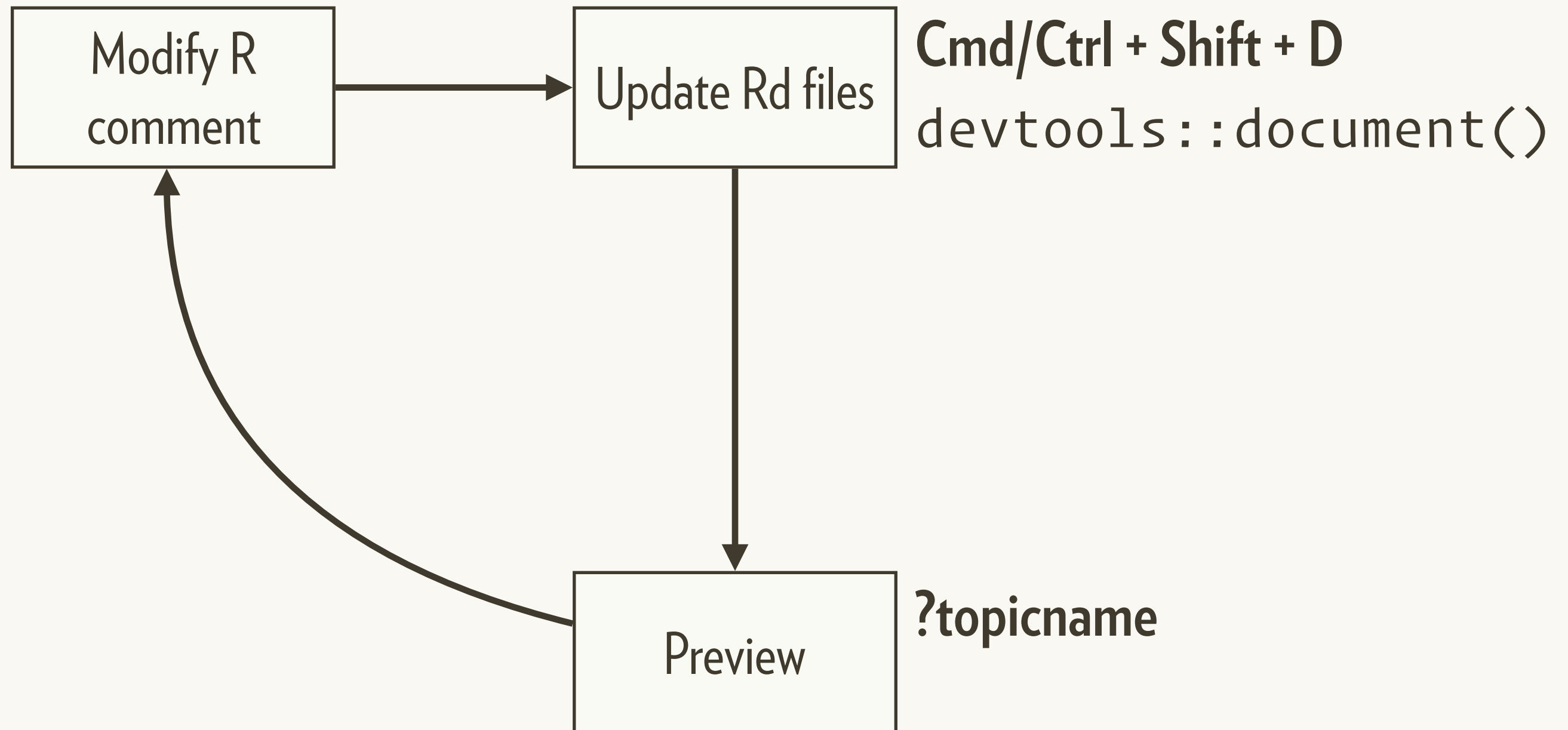
- x** A data frame
- name** Name of variable to create. If a variable of that name already exists it will be replaced
- value** Values to insert.
- where** Position to insert. Use 1 to insert on LHS, or -1 to insert on RHS.

## Examples

```
df <- data.frame(x = 1:5)
add_col(df, "y", runif(5))
add_col(df, "y", runif(5), where = 1)

add_col(df, "x", 5:1)
```

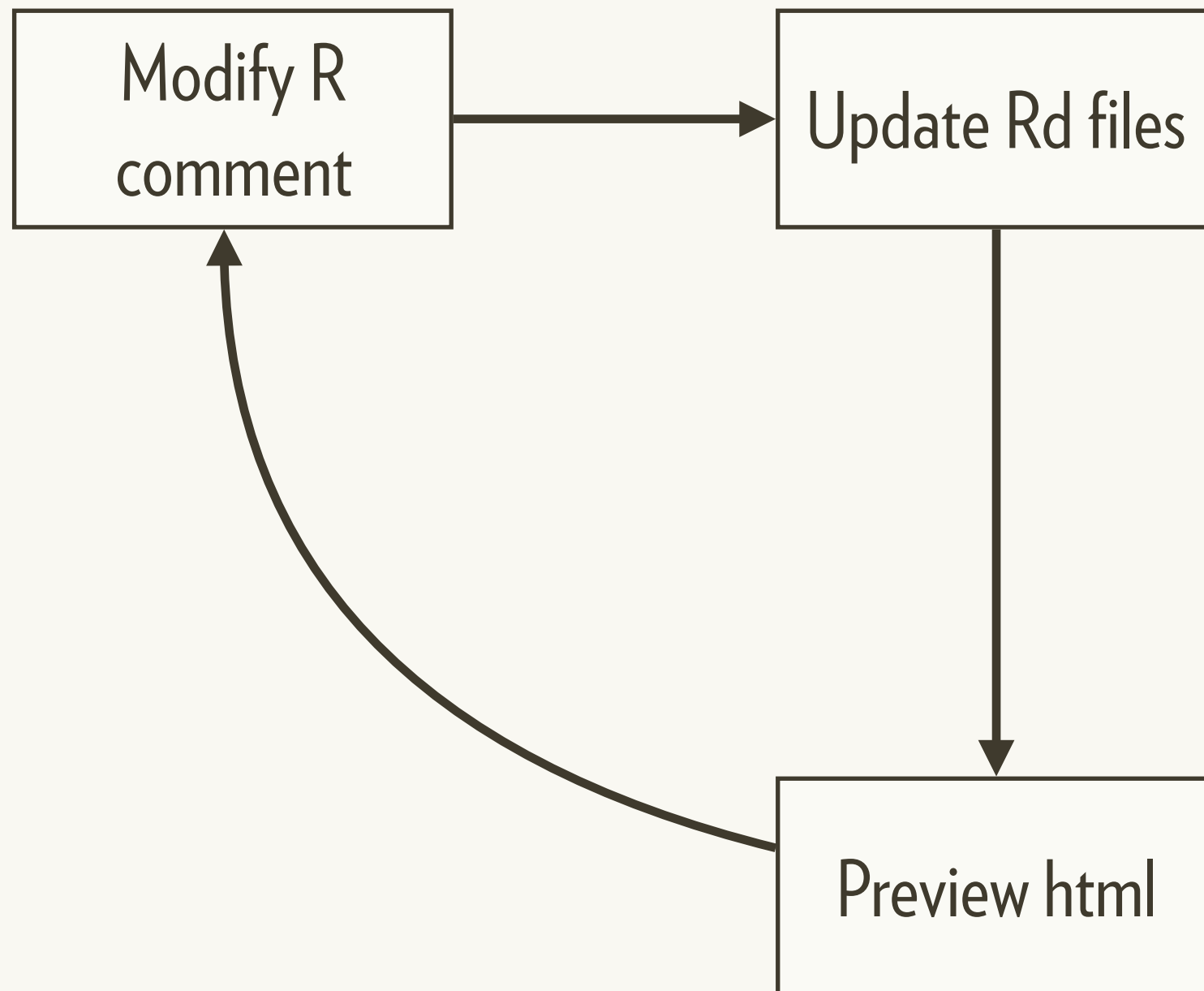
# Documentation workflow



# Two caveats

1. You must have loaded the package with `load_all()` at least once.  
Check for message "Using development documentation..."
2. This technique only builds individual files so links do not work.

# Documentation workflow



**Cmd/Ctrl + Shift + D**

`devtools::document()`

**NB:** You must have loaded the package with `load_all()` at least once

**?topicname**

Only shows single file,  
so links do not work

Change working directory/project to:

[hadcol]

# Your turn

Fix the typos in the documentation for `add_col`.

Run the documentation workflow to check your work



First sentence is the **title**

# Sum of Vector Elements

## Description

`sum` returns the sum of all the values present in its arguments.

Next paragraph is the **description**

## Usage

```
sum(..., na.rm = FALSE)
```

## Arguments

`...` numeric or complex or logical vectors.

`na.rm` logical. Should missing values (including `NaN`) be removed?

## Details

Everything else is the **details**

ectly or via the [Summary](#) group  
ld be unnamed, and dispatch is

If `na.rm` is `FALSE` an `NA` or `NaN` value in any of the arguments will cause a value of `NA` or

# The description block

First sentence is the **title**

```
#' Sum of vector elements
```

```
#'
```

```
#' \code{sum} returns the sum of all the values present in its arguments.
```

```
#'
```

```
#' This is a generic function: methods can be defined for it directly or via the
```

```
#' \code{\link{Summary}} group generic. For this to work properly, the arguments
```

```
#' \code{...} should be unnamed, and dispatch is on the first argument.
```

Next paragraph is the **description**

Everything else is the **details**

There are five **tags** you'll use for most functions

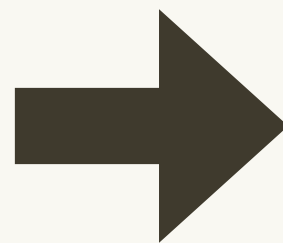
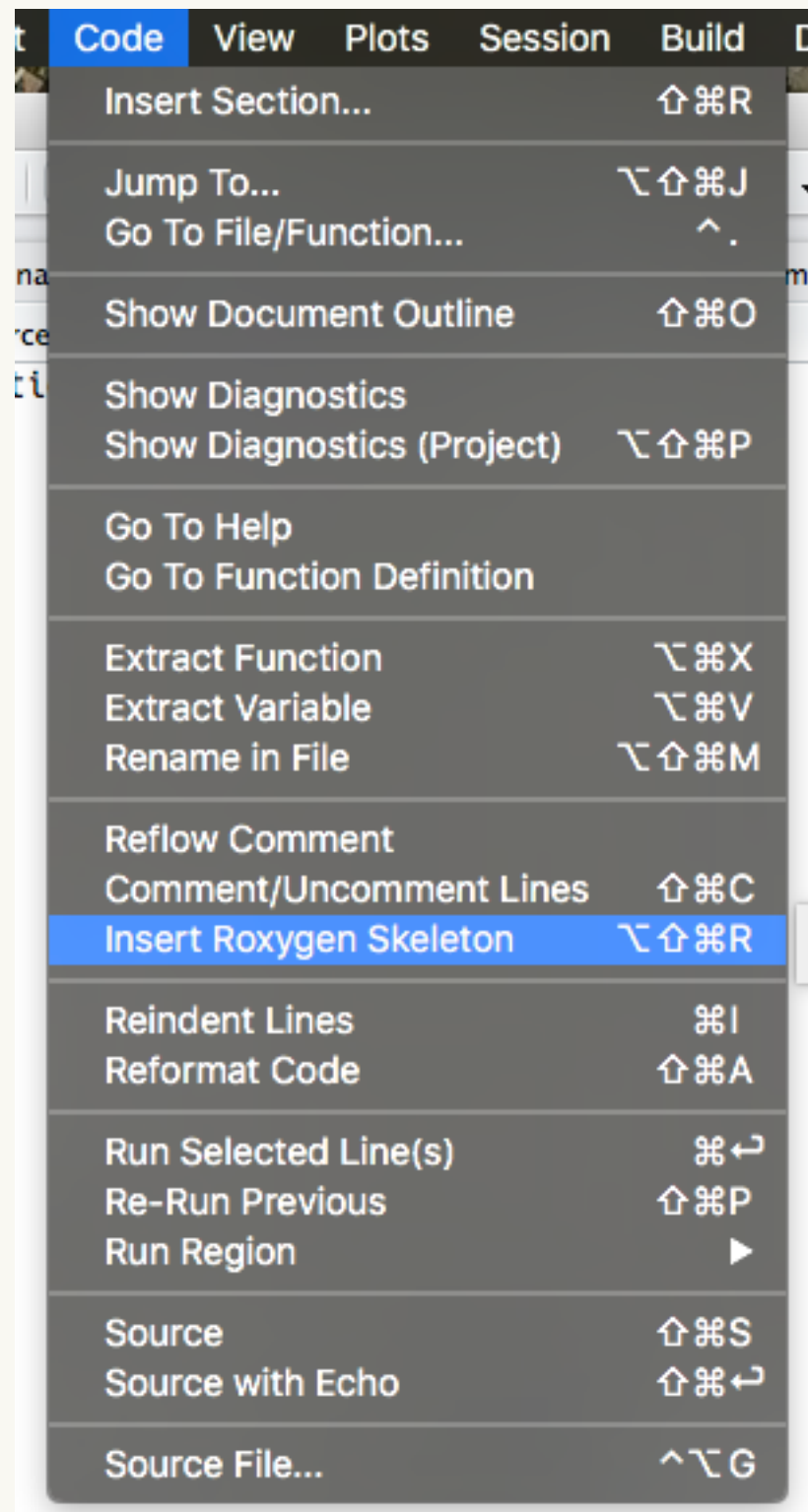
| Tag        | Purpose  |
|------------|--|
| @param arg | Describe inputs  |
| @examples  | Show how the function works.<br>(Usual RStudio shortcuts work) |
| @seealso   | Pointers to related functions                                  |
| @return    | Describe outputs (value)                                       |
| @export    | Is this a user-visible function?                               |

# Your turn

Document `add_cols()`.

(See next slide for hint)

# RStudio helps you remember



```
#' Title
#'
#' @param x
#' @param y
#' @param z
#'
#' @return
#' @export
#'
#' @examples
fun <- function(x, y, z) {
}
```

# Use markdown for formatting

```
# In new package, activate by running  
# usethis::use_roxygen_md()
```

```
**bold**, _italic_, `code`
```

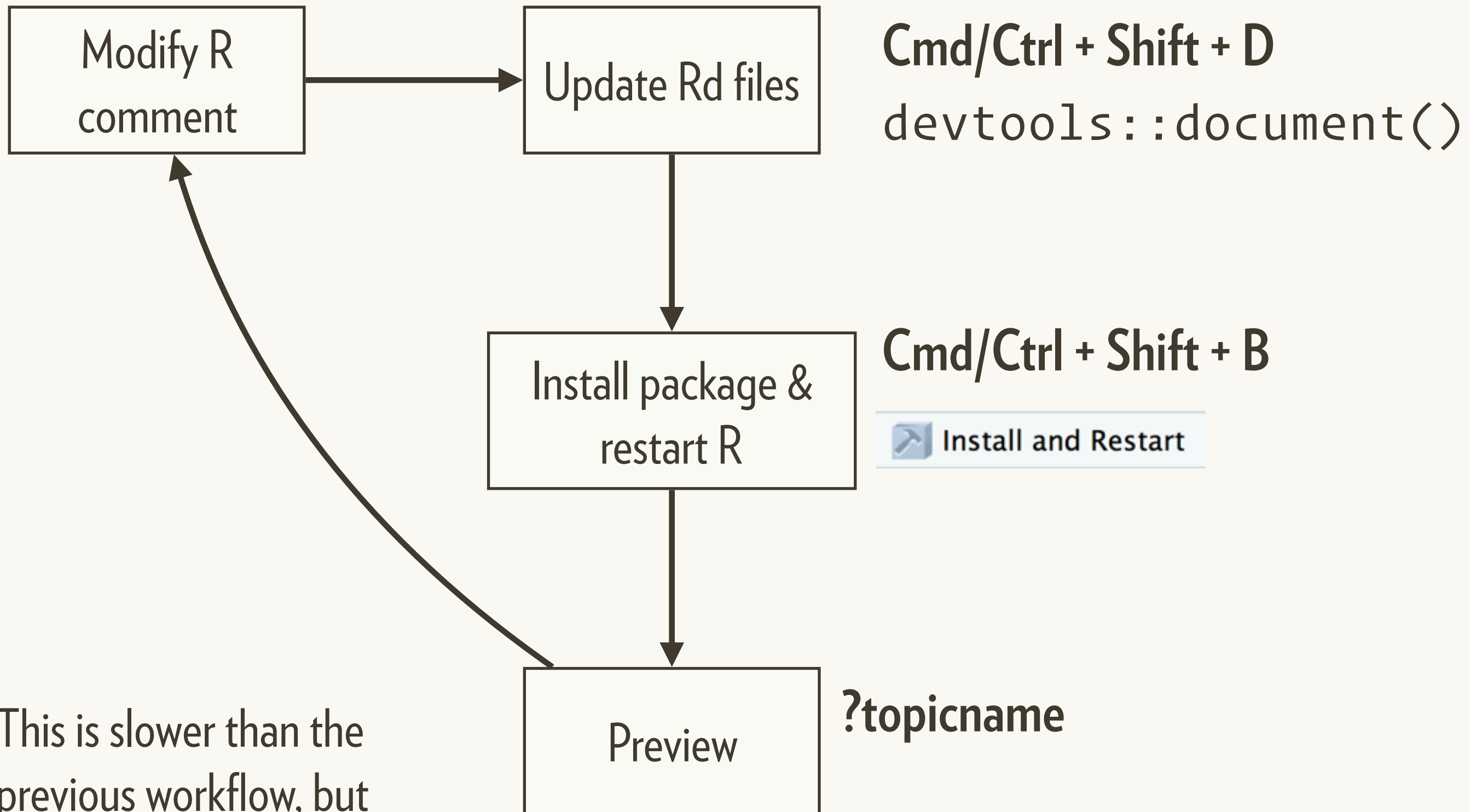
```
* [func()]
```

```
* [pkg::func()]
```

```
* [link text][func()]
```

```
* [link text][pkg::func()]
```

# Documentation workflow 2



This is slower than the previous workflow, but there are fewer caveats

# Your turn

Make real link to `cbind()`

Add a see also section (`@seealso`) to `add_col()` and `add_cols()` that links them together.

What happens if you add `@family xyz` to both?



# roxygen2 comes with other tools to reduce duplication

```
# Document multiple functions in the same file
```

```
#' @rdname add_col
```

```
# Inherit the parameter descriptions from
```

```
# another function
```

```
#' @inheritParams add_col
```

```
# Inherit everything from another topic
```

```
#' @inherit add_col
```

```
# Inherit selectively
```

```
#' @inherit add_col parameters return references
```

```
#'   title description details
```

```
#'   sections seealso
```

Read online about how to document other objects

# Data

<http://r-pkgs.had.co.nz/data.html#documenting-data>

# Classes & methods

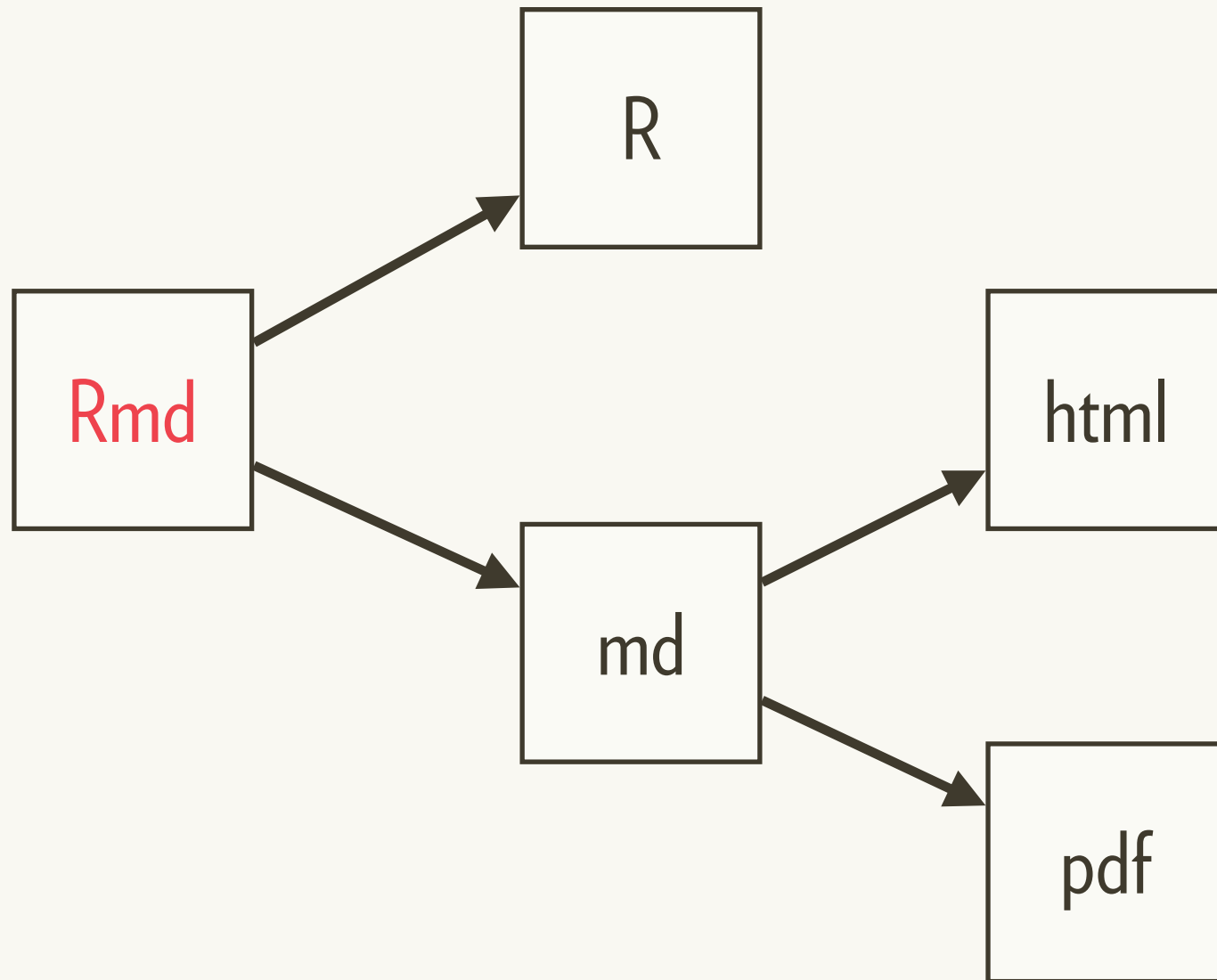
<http://r-pkgs.had.co.nz/man.html#man-classes>

# Packages

<http://r-pkgs.had.co.nz/man.html#man-packages>

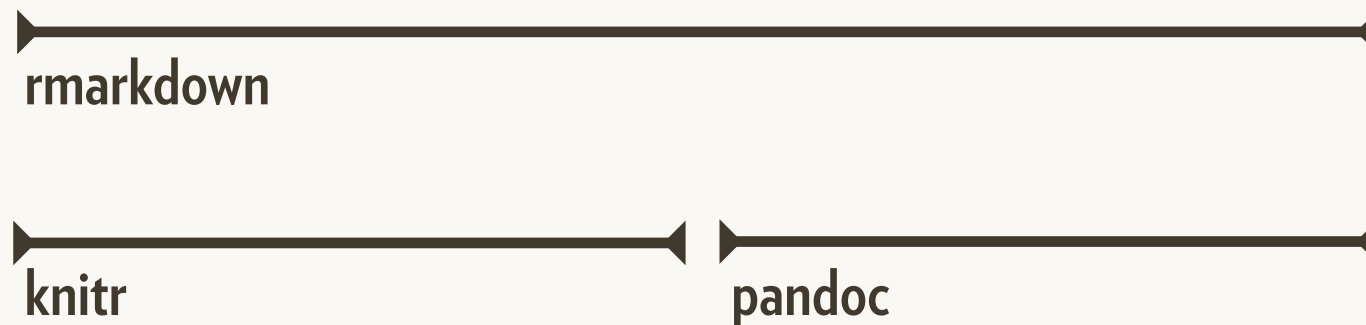
# Package documentation with `rmarkdown`

# Vignettes



Lets you combine prose  
and code to explain your  
how you package works.

The hard part is the  
writing, not the  
technology!



<http://r-pkgs.had.co.nz/vignettes.html>

Easiest way to get started is with `use_vignette()`

```
usethis::use_vignette("name")
```

```
# Adds to DESCRIPTION
```

```
Suggests: knitr
```

```
VignetteBuilder: knitr
```

```
# Creates vignettes/
```

```
# Drafts vignettes/name.Rmd
```

# Vignette = Rmarkdown + special metadata

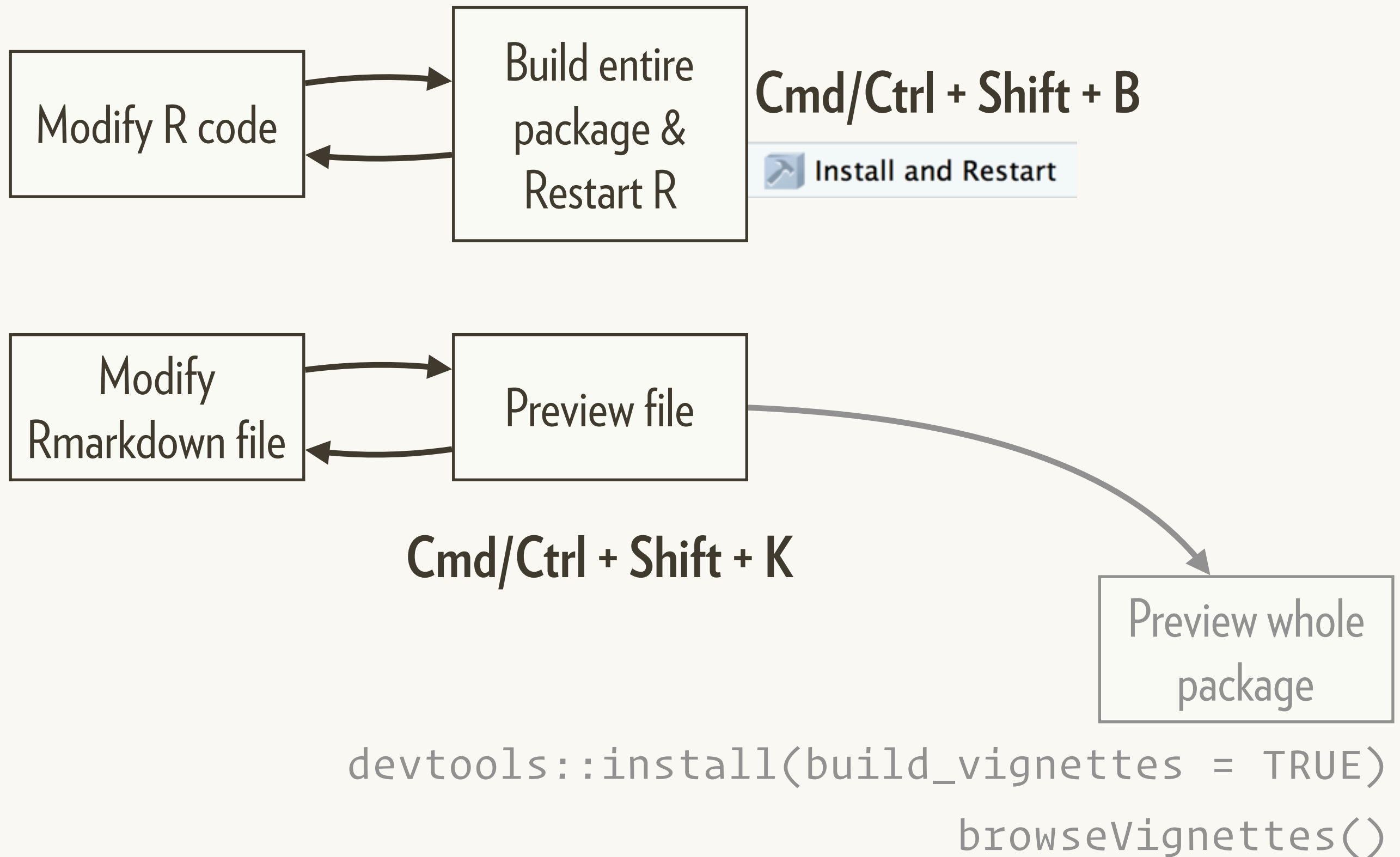
```
---  
title: "Vignette Title"  
author: "Vignette author"  
date: "`r Sys.Date()`"  
output: rmarkdown::html_vignette  
vignette: >  
  %\VignetteIndexEntry{Vignette Title}  
  %\VignetteEngine{knitr::rmarkdown}  
  %\VignetteEncoding{UTF-8}  
---
```



Vignettes are long form documentation commonly included in packages. Because they are part of the distribution of the package, they need to be as compact as possible. The ``html_vignette`` output type provides a custom style sheet (and tweaks some options) to ensure that the resulting html is as small as possible. The ``html_vignette`` format:

...

# Vignette workflow



# If sharing with others, include a README

```
# Your choice: but often useful to include
```

```
# results of running code
```

```
usethis::use_readme_md()
```

```
usethis::use_readme_rmd()
```

```
# For public projects this should include a
```

```
# brief overview, instructions on how to
```

```
# install, and a few examples. For private
```

```
# projects, this is a great place to jot down
```

```
# notes
```



If evolving over time, note changes to API

```
usethis::use_news_md()
```

```
# For public projects, this should note  
# important changes from perspective of  
# the user. Most important to describe API  
# changes. Less important for private projects
```



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/>