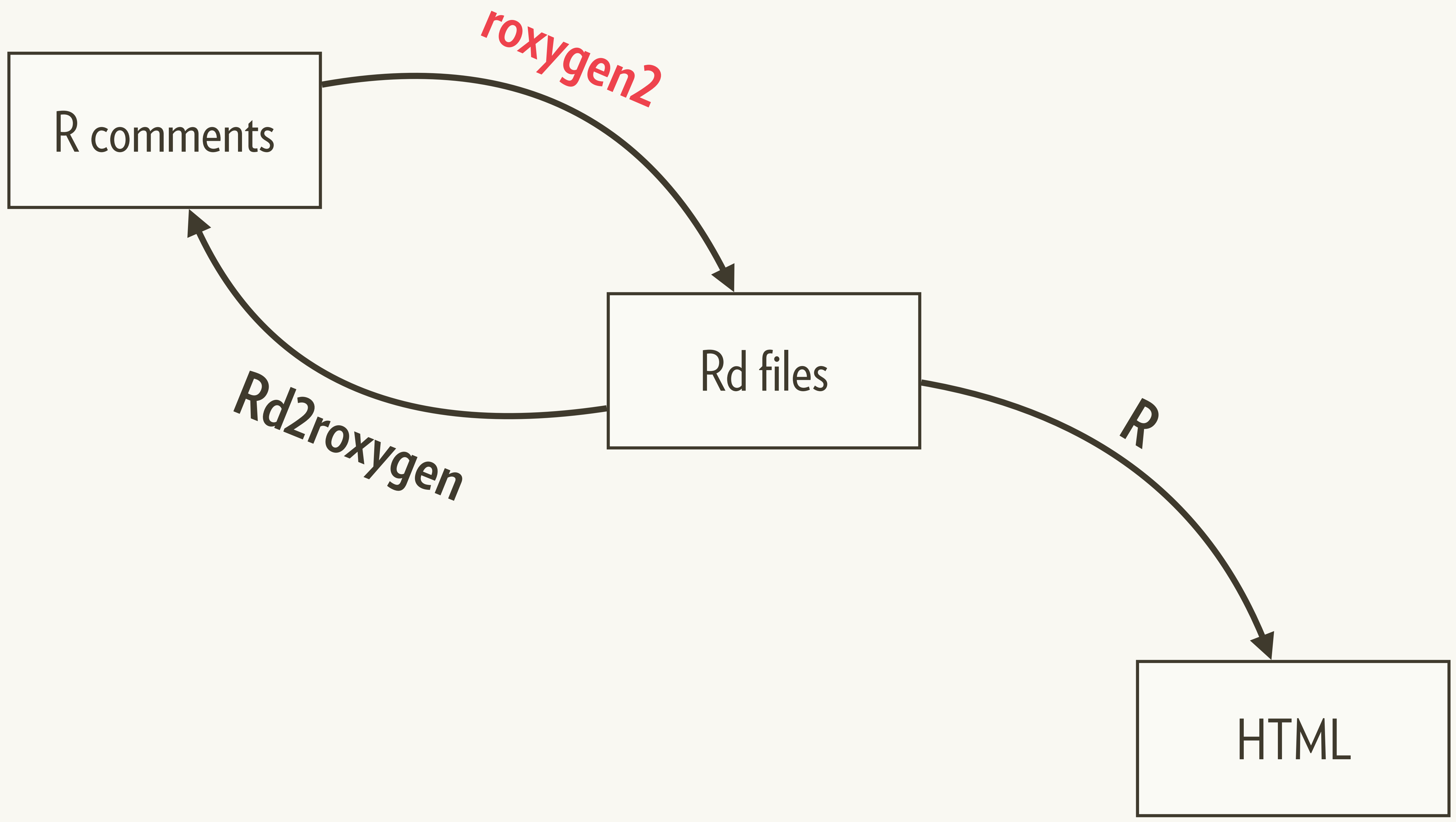
man/

December 2016

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

Roxygen2



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

Raw R comments

```
#' Make a discrete random vaiable.
#'
#' @param x a numberic vector giving the values of the random vaiable.
#' @param probs optional, a numeric vector giving the proabilities
#'   corresponding to each x value. If not specific, assumes all outcomes
#'   are equally likely
#' @export
#' @return An S3 objct of class rv.
#' @examples
#' dice <- rv(1:6)
#' P(dice > 3)
#' E(dice)
#' P(dice > dice + 1
rv <- function(x, probs = NULL) {
   ...
}
```

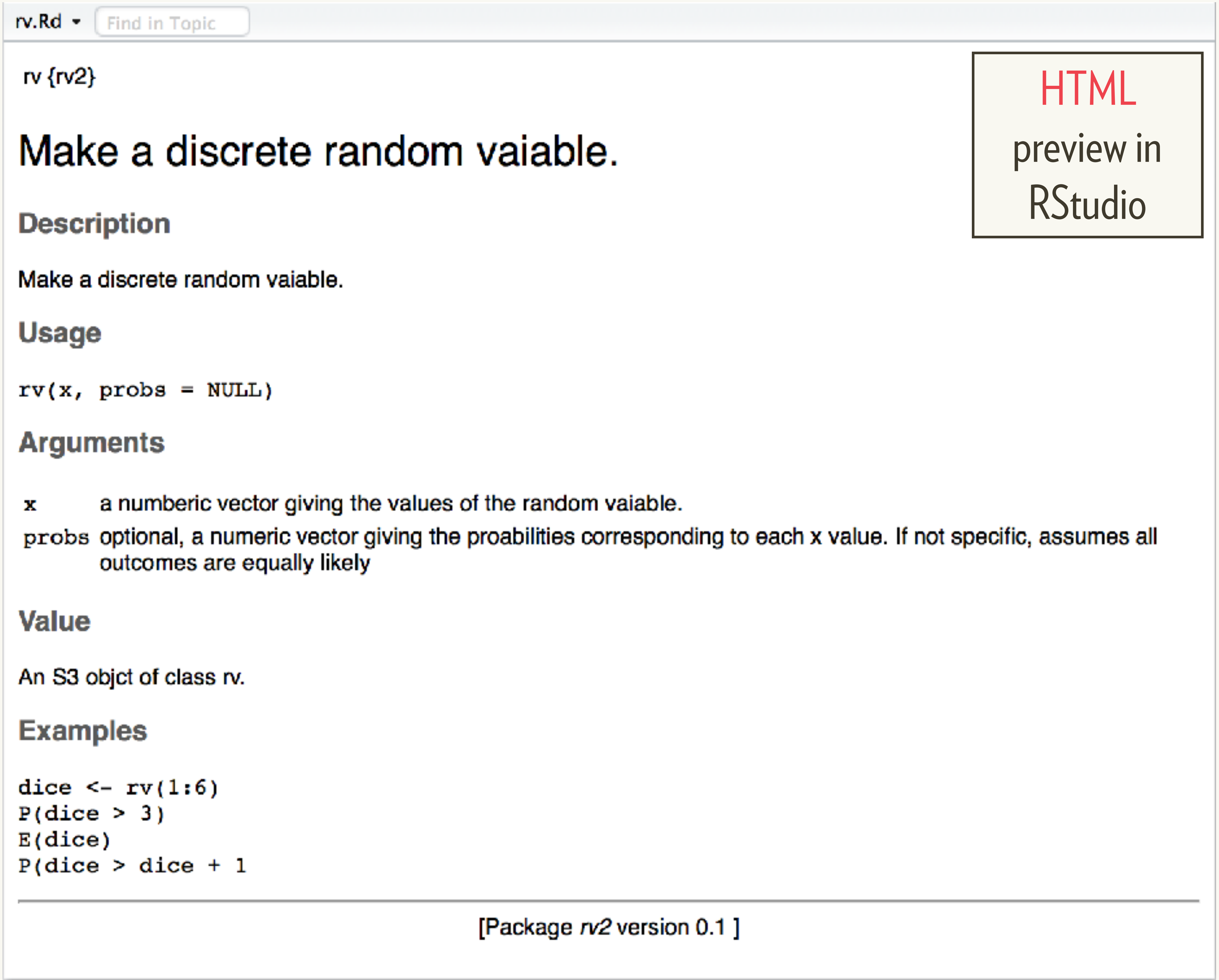
Raw R comments

```r
#' Make a discrete random vaiable.
#'
#' @param x a numberic vector giving the values of the random vaiable.
#' @param probs optional, a numeric vector giving the proabilities
#'   corresponding to each x value. If not specific, assumes all outcomes
#'   are equally likely
#' @export
#' @return An S3 objct of class rv.
#' @examples
#' dice <- rv(1:6)
#' P(dice > 3)
#' E(dice)
#' P(dice > dice + 1
rv <- function(x, probs = NULL) {
    ...
}
```

Generated Rd file

```
\name{rv}
\alias{rv}
\title{Make a discrete random vaiable.}
\usage{
rv(x, probs = NULL)
}
\arguments{
  \item{x}{a numberic vector giving the values of the
  random vaiable.}

  \item{probs}{optional, a numeric vector giving the
  proabilities corresponding to each x value. If not
  specific, assumes all outcomes are equally likely}
}
\value{
An S3 objct of class rv.
}
\description{
Make a discrete random vaiable.
}
\examples{
dice <- rv(1:6)
P(dice > 3)
E(dice)
P(dice > dice + 1
}
```



rv.Rd ▾ Find in Topic

rv {rv2}

Make a discrete random vaiable.

Description

Make a discrete random vaiable.

Usage

```
rv(x, probs = NULL)
```

Arguments

`x`	a numberic vector giving the values of the random vaiable.
`probs`	optional, a numeric vector giving the proabilities corresponding to each x value. If not specific, assumes all outcomes are equally likely

Value

An S3 objct of class rv.

Examples

```
dice <- rv(1:6)
P(dice > 3)
E(dice)
P(dice > dice + 1
```

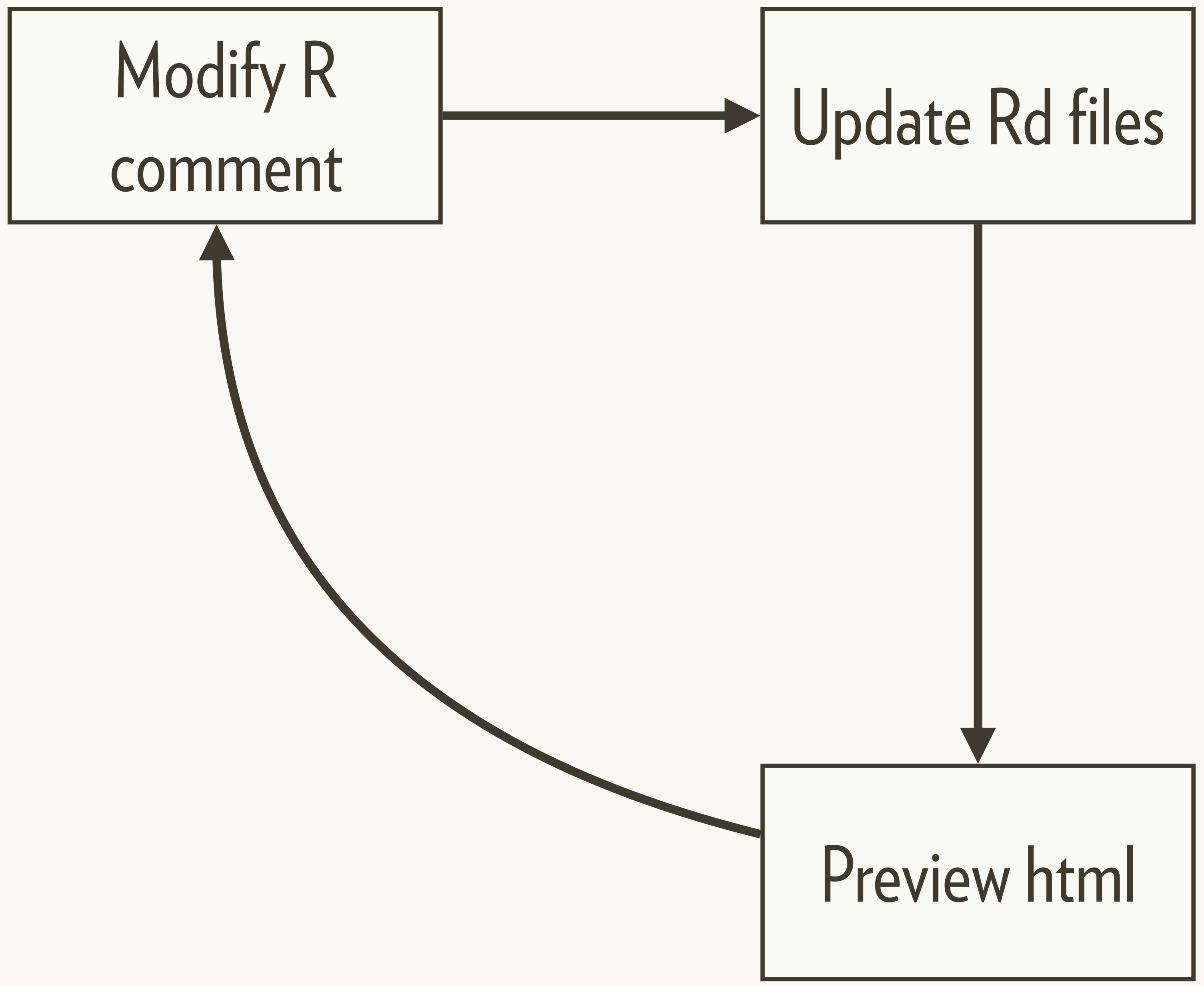
[Package *rv2* version 0.1]

HTML preview in RStudio

Change working directory/project to:

[document-me]

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

Modify the comments

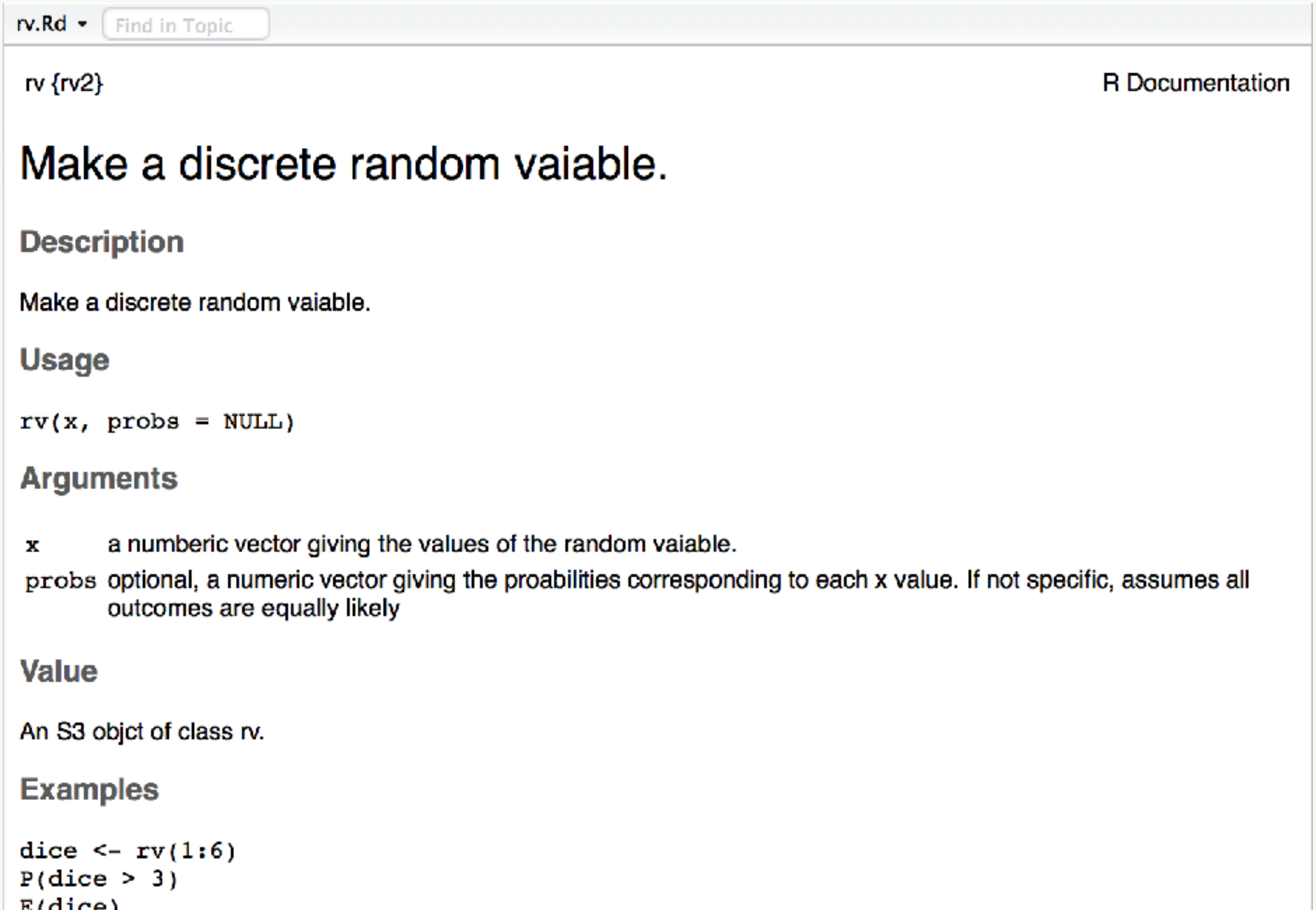
```r
#' Make a discrete random vaiable.
#'
#' @param x a numberic vector giving the values of the random vaiable.
#' @param probs optional, a numeric vector giving the proabilities
#'   corresponding to each x value. If not specific, assumes all outcomes
#'   are equally likely
#' @export
#' @return An S3 objct of class rv.
#' @examples
#' dice <- rv(1:6)
#' P(dice > 3)
#' E(dice)
#' P(dice > dice + 1
rv <- function(x, probs = NULL) {
   ...
}
```

Cmd/Ctrl + Shift + D

```
\name{rv}
\alias{rv}
\title{Make a discrete random vaiable.}
\usage{
rv(x, probs = NULL)
}
\arguments{
  \item{x}{a numberic vector giving the values of the
  random vaiable.}

  \item{probs}{optional, a numeric vector giving the
  proabilities corresponding to each x value. If not
  specific, assumes all outcomes are equally likely}
}
\value{
An S3 objct of class rv.
}
\description{
Make a discrete random vaiable.
}
\examples{
dice <- rv(1:6)
P(dice > 3)
E(dice)
P(dice > dice + 1
}
```

?rv



rv {rv2}   R Documentation

Make a discrete random vaiable.

Description

Make a discrete random vaiable.

Usage

```
rv(x, probs = NULL)
```

Arguments

`x`	a numberic vector giving the values of the random vaiable.
`probs`	optional, a numeric vector giving the proabilities corresponding to each x value. If not specific, assumes all outcomes are equally likely

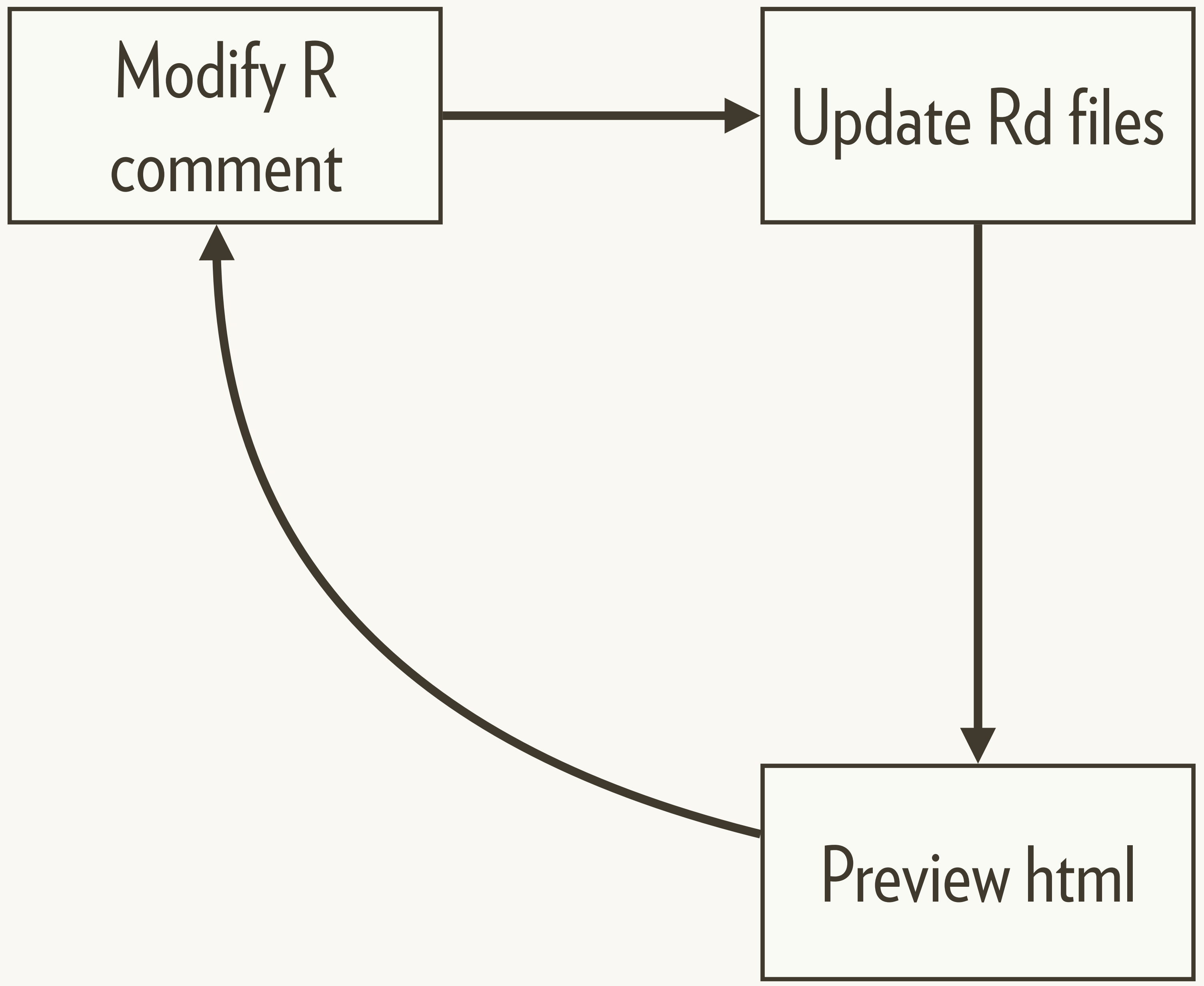
Value

An S3 objct of class rv.

Examples

```
dice <- rv(1:6)
P(dice > 3)
E(dice)
```

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

Your turn

Fix the typos in the documentation for rv.

Run the documentation workflow to check your work

Roxygen tags

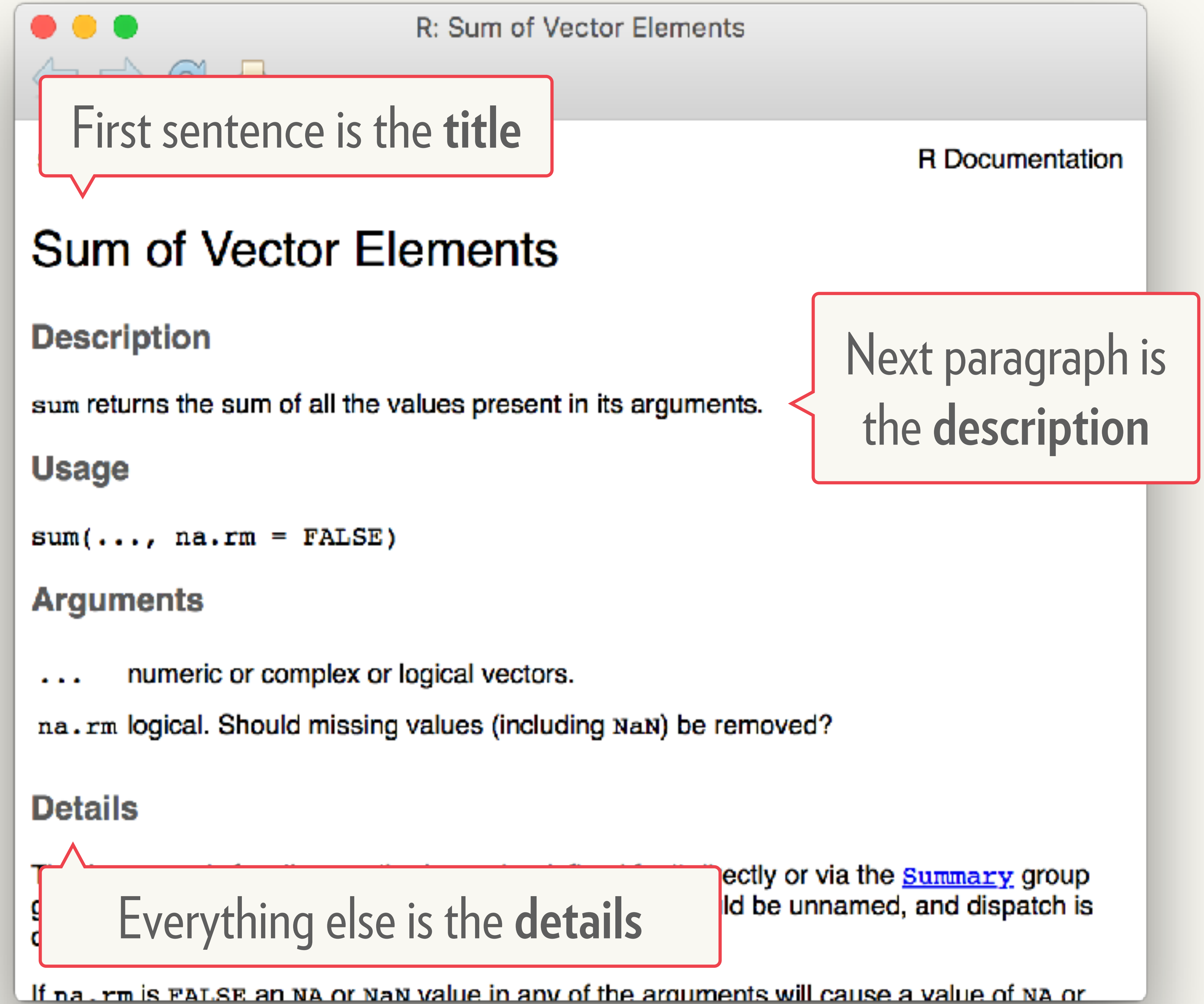
The description block

First sentence is the **title**

Next paragraph is the **description**

```
#' 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.
```

Everything else is the **details**


R: Sum of Vector Elements

R Documentation

Sum of Vector Elements

Description

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

Usage

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

Arguments

`...`	numeric or complex or logical vectors.
`na.rm`	logical. Should missing values (including `NaN`) be removed?

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

First sentence is the **title**
Next paragraph is the **description**
Everything else is the **details**

There are five tags you'll for most functions

Tag	Purpose
@param arg	Describe inputs
@examples	Show how the function works
@seealso	Pointers to related functions
@return	Describe outputs (value)
@export	We'll learn about this later

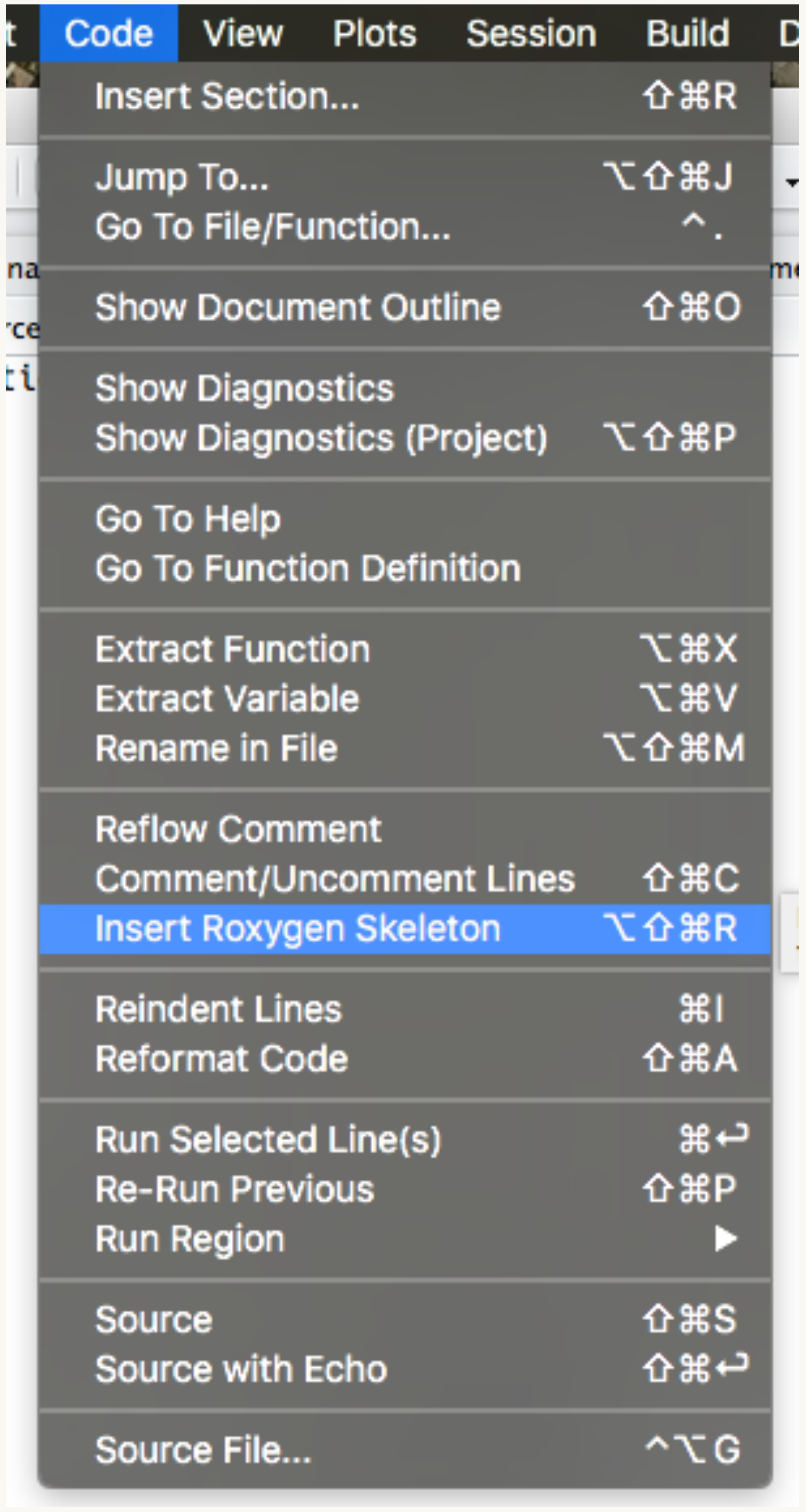
Your turn

Document P( ).

Here’s my attempt

```r
#' Compute the probability that an event occurs.
#'
#' @param x an event. An event is a special type of discrete random variable
#'   that only has two outcomes: \code{TRUE} or \code{FALSE}. It is usually
#'   created by applying a comparison operator to a random variable.
#' @return a probability (numeric vector of length 1) between 0 and 1.
#' @export
#' @examples
#' wheel <- rv(1:20)
#' P(wheel > 10)
#' P(wheel %% 2 == 0)
P <- function(x) {
  stopifnot(is.logical(x), is.rv(x))
  sum(probs(x)[x])
}
```

RStudio helps you remember



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

}
```

Text formatting

Rd uses a special language for text formatting

Tag	Purpose
\code{}	Inline R code
\eqn{}	Inline equation (standard latex)
\emph{}	Italic text
\strong{}	Bold text

```
#' A bulleted list:
#' \itemize{
#'   \item First item
#'   \item Second item
#' }

#' An ordered list:
#' \enumerate{
#'   \item First item
#'   \item Second item
#' }
```

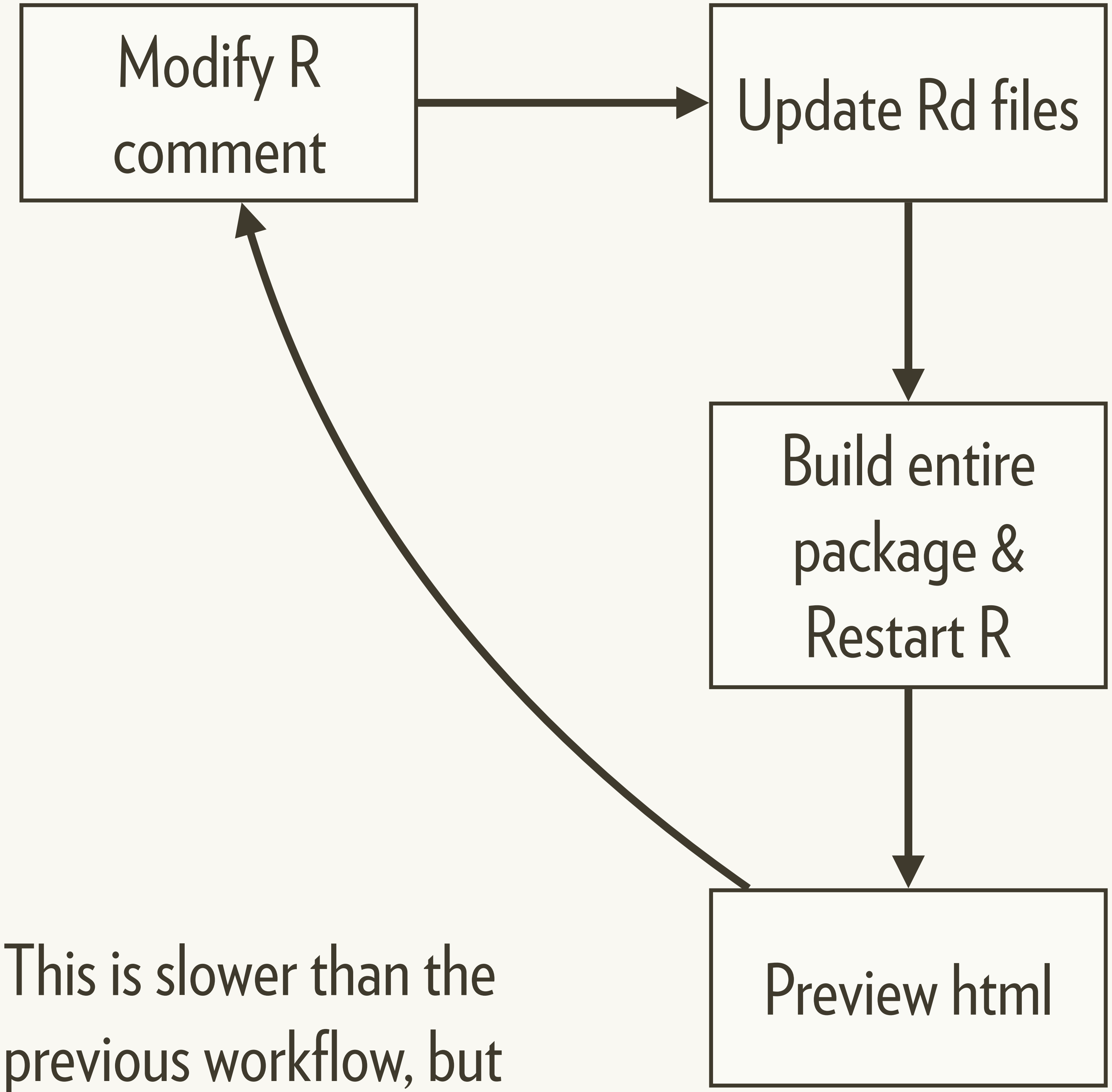
Your turn

Make a bulleted list with bold, italic, and code items.

Links need commands and a new workflow

Tag	Purpose
\link{foo}	Link to foo in current package
\link[bar]{foo}	Link to foo in package bar
\url{http://rstudio.com}	Link to website
\href{http://rstudio.com}{Rstudio}	Link to website with custom text
\email{hadley@rstudio.com}	Email address

Documentation workflow 2



Cmd/Ctrl + Shift + D

`devtools::document()`

Cmd/Ctrl + Shift + B

Build & Reload

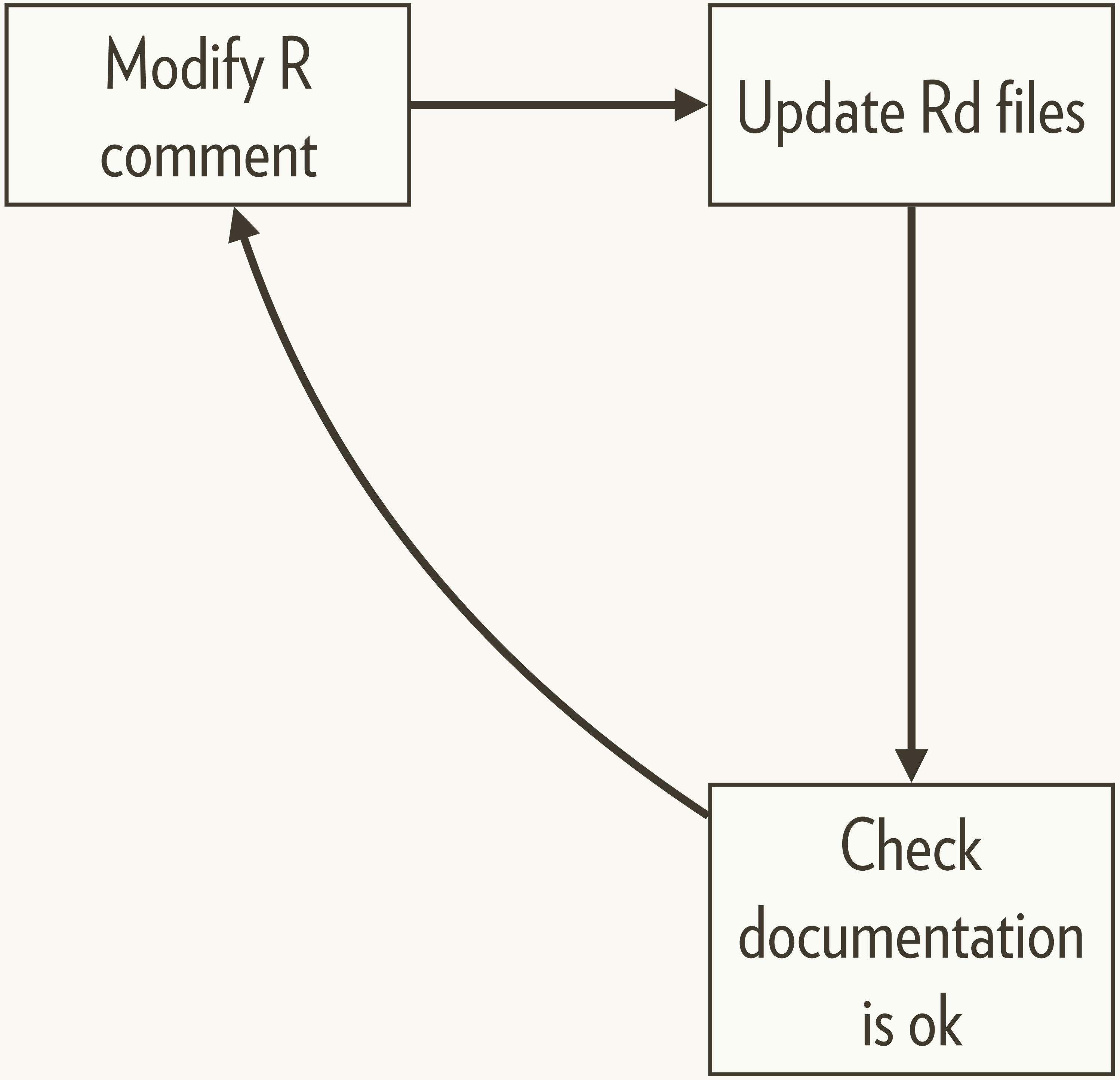
?topicname

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

Your turn

Add a see also section (@seealso) to the documentation for `rv()` that points to the most important functions.

Documentation workflow 3



Cmd/Ctrl + Shift + D

`devtools::document()`

`devtools::check_man()`

Your turn

Run `devtools::check_man()` and `document()` and iterate until all problems are fixed. (`check_man()`returns nothing when OK)

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