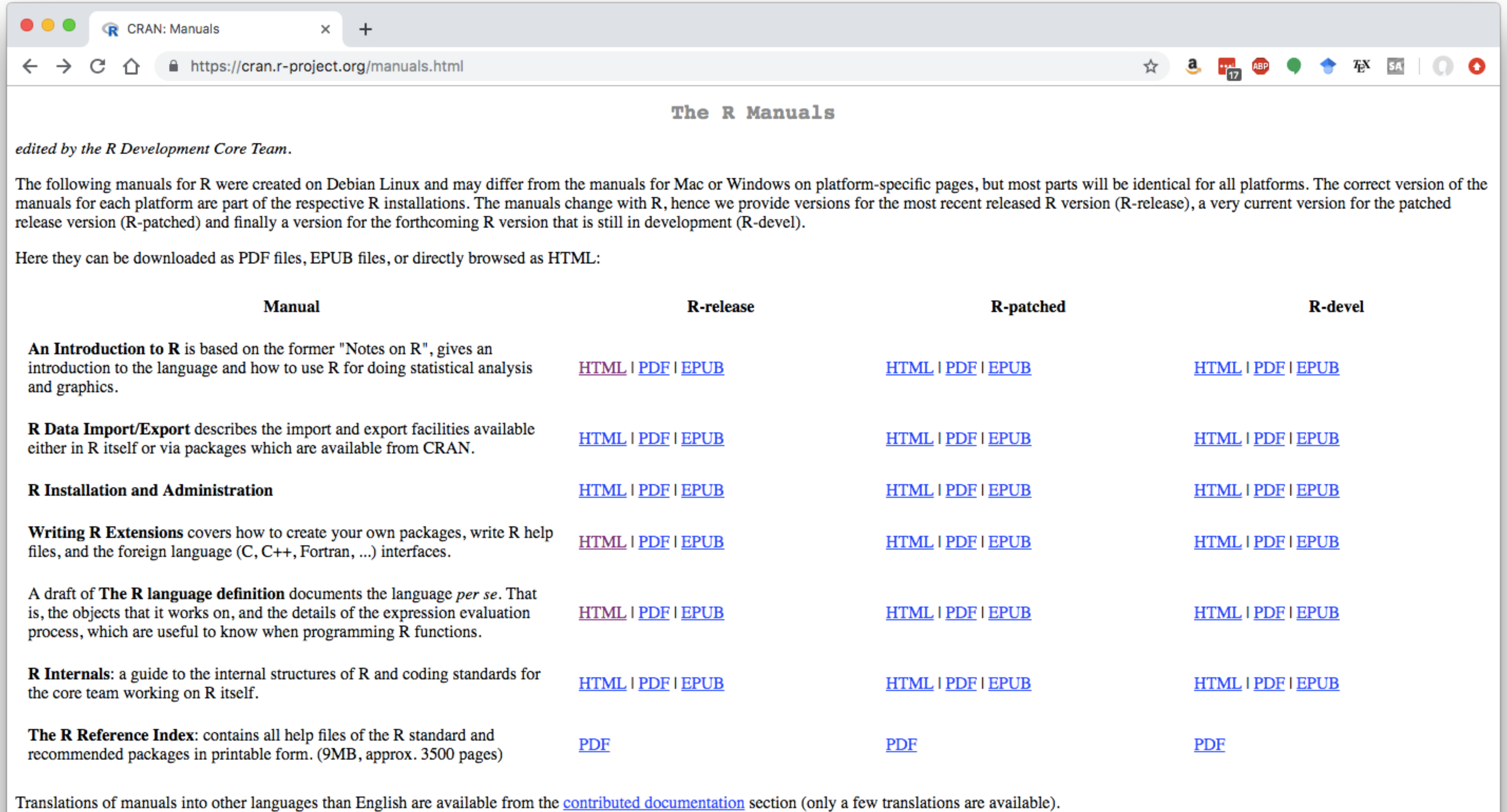
Non-standard evaluation (NSE)

An aside— how we used to learn R



The R Manuals

edited by the R Development Core Team.

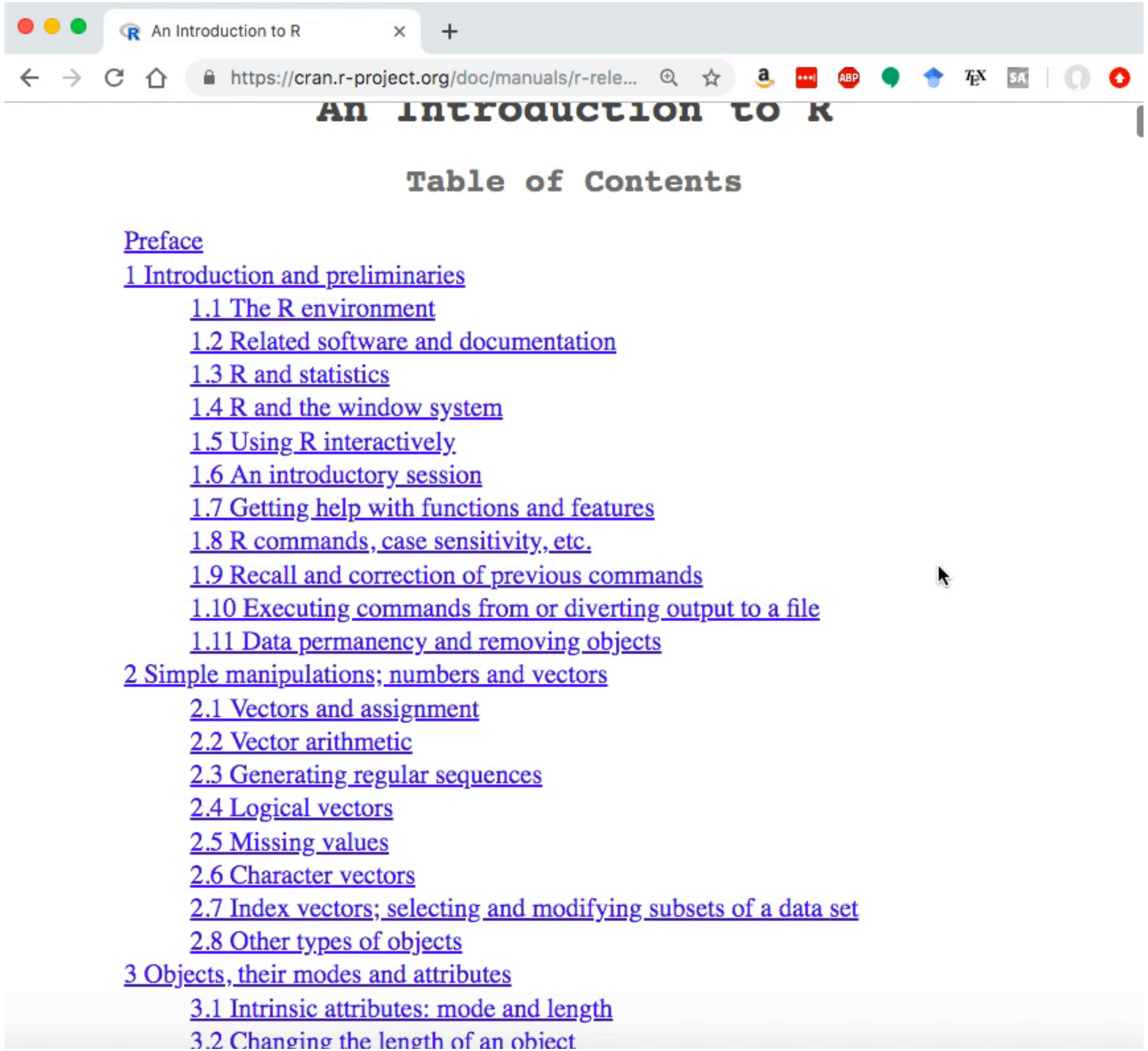
The following manuals for R were created on Debian Linux and may differ from the manuals for Mac or Windows on platform-specific pages, but most parts will be identical for all platforms. The correct version of the manuals for each platform are part of the respective R installations. The manuals change with R, hence we provide versions for the most recent released R version (R-release), a very current version for the patched release version (R-patched) and finally a version for the forthcoming R version that is still in development (R-devel).

Here they can be downloaded as PDF files, EPUB files, or directly browsed as HTML:

Manual	R-release	R-patched	R-devel
An Introduction to R is based on the former "Notes on R", gives an introduction to the language and how to use R for doing statistical analysis and graphics.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Data Import/Export describes the import and export facilities available either in R itself or via packages which are available from CRAN.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Installation and Administration	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
Writing R Extensions covers how to create your own packages, write R help files, and the foreign language (C, C++, Fortran, ...) interfaces.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
A draft of **The R language definition** documents the language *per se*. That is, the objects that it works on, and the details of the expression evaluation process, which are useful to know when programming R functions.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Internals: a guide to the internal structures of R and coding standards for the core team working on R itself.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
The R Reference Index: contains all help files of the R standard and recommended packages in printable form. (9MB, approx. 3500 pages)	PDF	PDF	PDF

Translations of manuals into other languages than English are available from the contributed documentation section (only a few translations are available).

An aside— how we used to learn R



An Introduction to R

Table of Contents

"There are three kinds of language objects that are available for modification, calls, expressions, and functions."

Some useful functions for computing on the language using base R:

- `quote()`
- `enquote()`
- `substitute()`
- `deparse()`
- `eval()`

Call objects

"sometimes referred to as “unevaluated expressions”, although this terminology is somewhat confusing" (thanks, R!)

We can get a call object using the function `quote()` (not to be confused with a quoted string)

```
> quote(2+2)
2 + 2
> "2+2" # just a character string
[1] "2+2"
```

Call objects

If you wanted to then evaluate a `quote()`ed call object, you could use `eval()`

```
> eval(quote(2+2))
[1] 4
> eval("2+2") # there's nothing to evaluate here
[1] "2+2"
```

Remember, R is lazy

Sometimes, quote() doesn't give you exactly what you were expecting, because R is lazy.

```
> a <- 1
> b <- 2
> quote(a + b)
a + b
```

Remember, R is lazy

This is where `substitute()` comes in. `substitute()` will substitute in the values it knows about in a particular environment.

```
> substitute(a + b, env = .GlobalEnv)
a + b
> ?substitute
```

"If it is an ordinary variable, its value is substituted, unless env is .GlobalEnv in which case the symbol is left unchanged."

Remember, R is lazy

Okay… but environments are just lists! So we can make our own.

```
> substitute(a + b, list(a = 1, b = 2))
1 + 2
```

Of course, everything needs to be defined in that environment

```
> substitute(a + b, list(a = 1, b = x))
Error: object 'x' not found
> substitute(a + b, list(a = 1, b = quote(x)))
1 + x
```

Tidyverse NSE

`quo()` is like `quote()`

`enquo()` is like `substitute()`

`!!` is like `eval()` ?

Where we're going…

I want you to create a new version of your bootstrap function, which works in the tidyverse. In other words, instead of calling

```
bootstrap(mtcars$mpg, samples = 500)
```

I want to be able to call

```
mtcars %>% bootstrap(mpg, samples = 500)
```