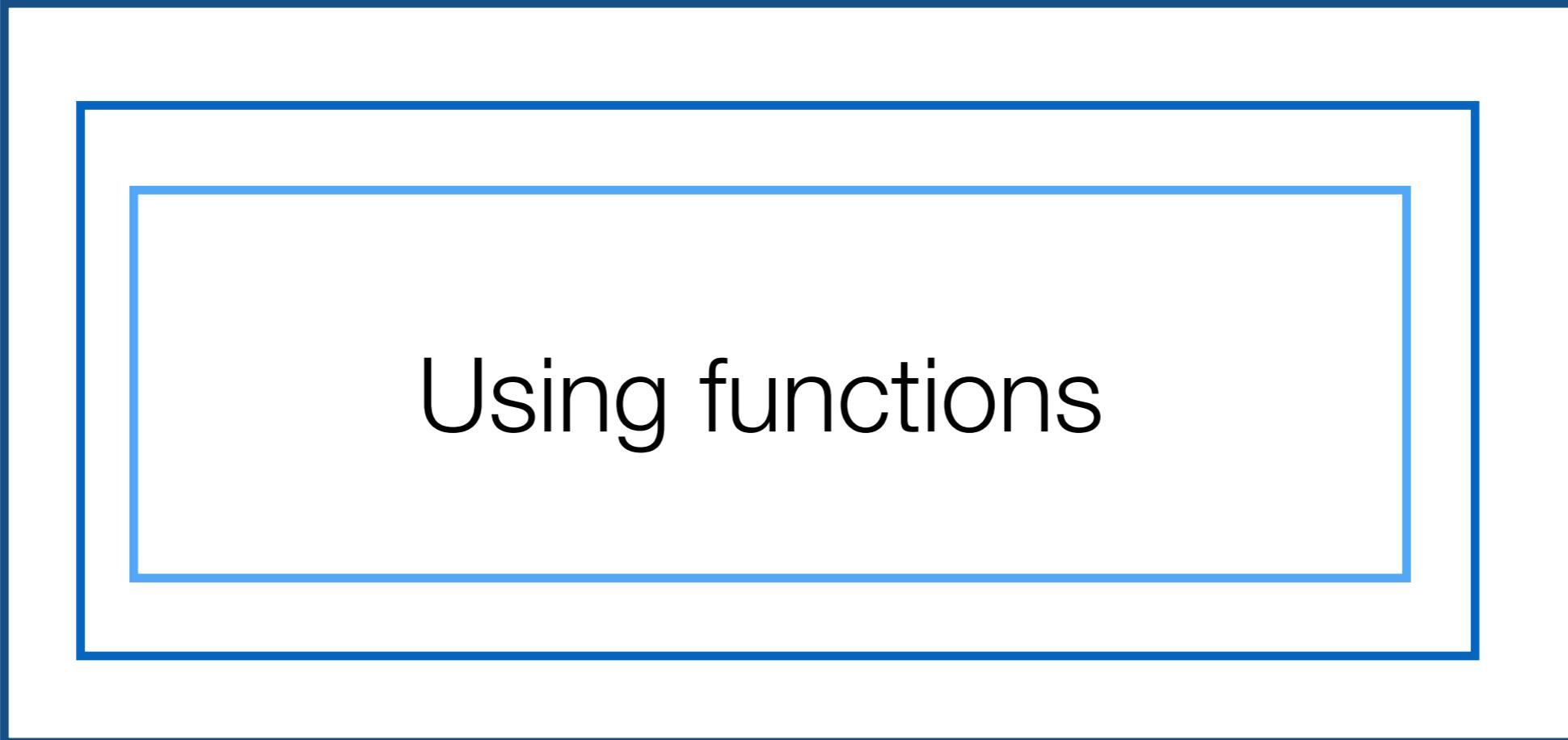


Introduction to R: Functions

Research Methods for Human Inquiry
Andrew Perfors



Using functions

Functions

There are not enough symbols on the keyboard to do everything you might want to do... so there are only a few operators

Most things are **functions**

Example: square root

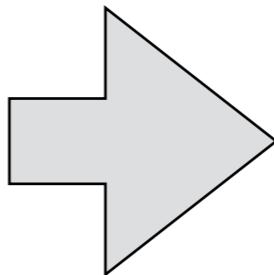
```
> sqrt(25)  
[1] 5
```

- The function is called `sqrt()`
- The 25 is the “**argument**” to the function.

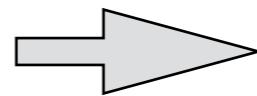
Arguments

Every function has arguments. You can think of functions like recipes, and arguments like the ingredients. You can't make the recipe without the right ingredients, and the recipe combines the ingredients in a specific way.

Ingredients
½ cup milk
½ cup fine fresh breadcrumbs
1 large egg
½ cup grated parmesan cheese
½ cup finely chopped flat leaf parsley
450g ground beef, pork, veal, chicken or turkey
1 clove minced garlic
100ml olive oil
200g tomato pasata
Salt flakes
Freshly ground pepper

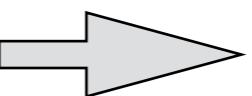


`meatball_recipe(milk,breadcrumbs,egg,etc..)`



`meatballs!`

`sqrt(25)`

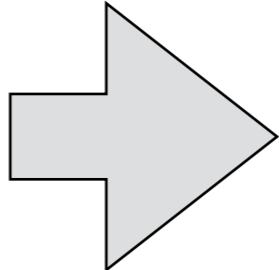


`5`

Arguments

Just like a recipe, it won't work if you don't give it the right ingredients (arguments)

Ingredients
½ cup milk
½ cup fine fresh breadcrumbs
1 large egg
½ cup grated parmesan cheese
½ cup finely chopped flat leaf parsley
450g ground beef, pork, veal, chicken or turkey
1 clove minced garlic
100ml olive oil
200g tomato pasata
Salt flakes
Freshly ground pepper



```
> sqrt()
```

```
Error in sqrt() : 0 arguments passed to 'sqrt' which requires 1
```

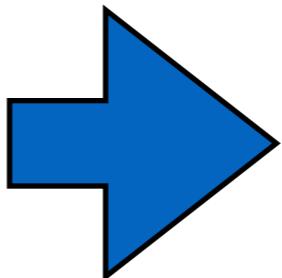
Is this a cooking show?
Are you going to teach us how
to cook?

Because I really like meatballs.
And food in general.



Some functions that you might see on a scientific calculator

<code>sqrt()</code>	- Square root
<code>round()</code>	- Round a number
<code>log()</code>	- Logarithm
<code>exp()</code>	- Exponentiation
<code>abs()</code>	- Absolute value



```
> sqrt(4)
[1] 2
> round(3.24)
[1] 3
> log(32)
[1] 3.465736
> exp(2)
[1] 7.389056
> abs(-3)
[1] 3
```

Some arguments have defaults

- A lot of arguments have “**default values**”.
- If you don’t tell R what value to use, it uses the default

```
> round( x = 3.1415 )  
[1] 3
```

The default number of digits to round to is zero, so that’s what R uses here

Functions with multiple arguments

- Many functions can “take” more than one argument;
- Separate the arguments with commas.

Argument #1: The number
that needs to be rounded

Argument #2: How many
digits to round it to?

```
> round( 3.1415, 2 )  
[1] 3.14
```

Arguments have names

- Most of the time, the arguments have “**names**”,
- You can use the names when typing commands

The number that needs
to be rounded is called **x**

The number of digits to
round to is called **digits**

```
> round( x = 3.1415, digits = 2 )  
[1] 3.14
```

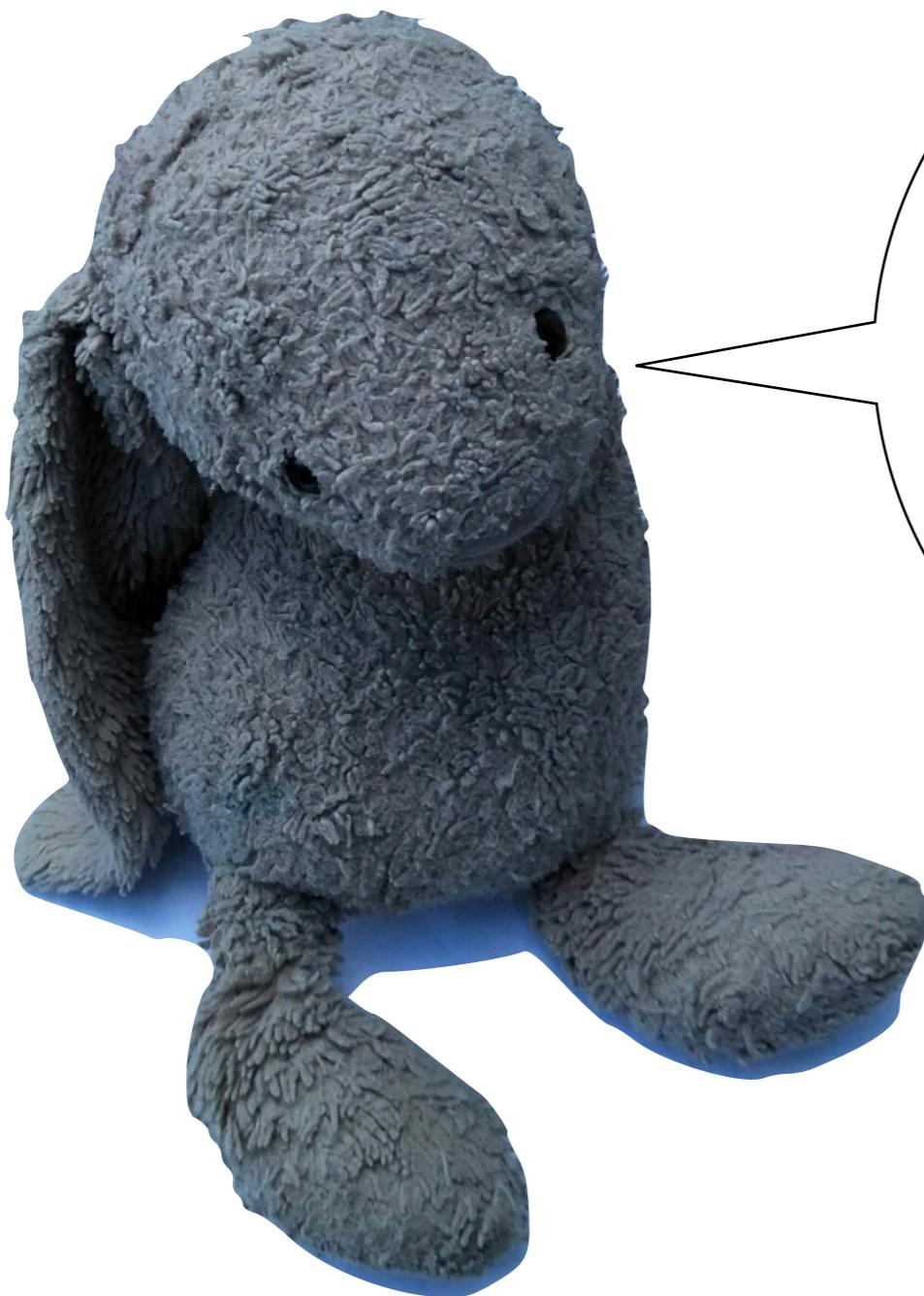
Arguments have names

If you specify the names, the order doesn't matter. But if you don't, it does!

```
> round(x=3.14159, digits=2)  
[1] 3.14  
> round(digits=2, x=3.14159)  
[1] 3.14
```

```
> round(3.14159, 2)  
[1] 3.14  
> round(2, 3.14159)  
[1] 2
```

This is known as a “silent fail.” What we input didn’t make sense, so it just went with the default, with no warning. Be careful of these!



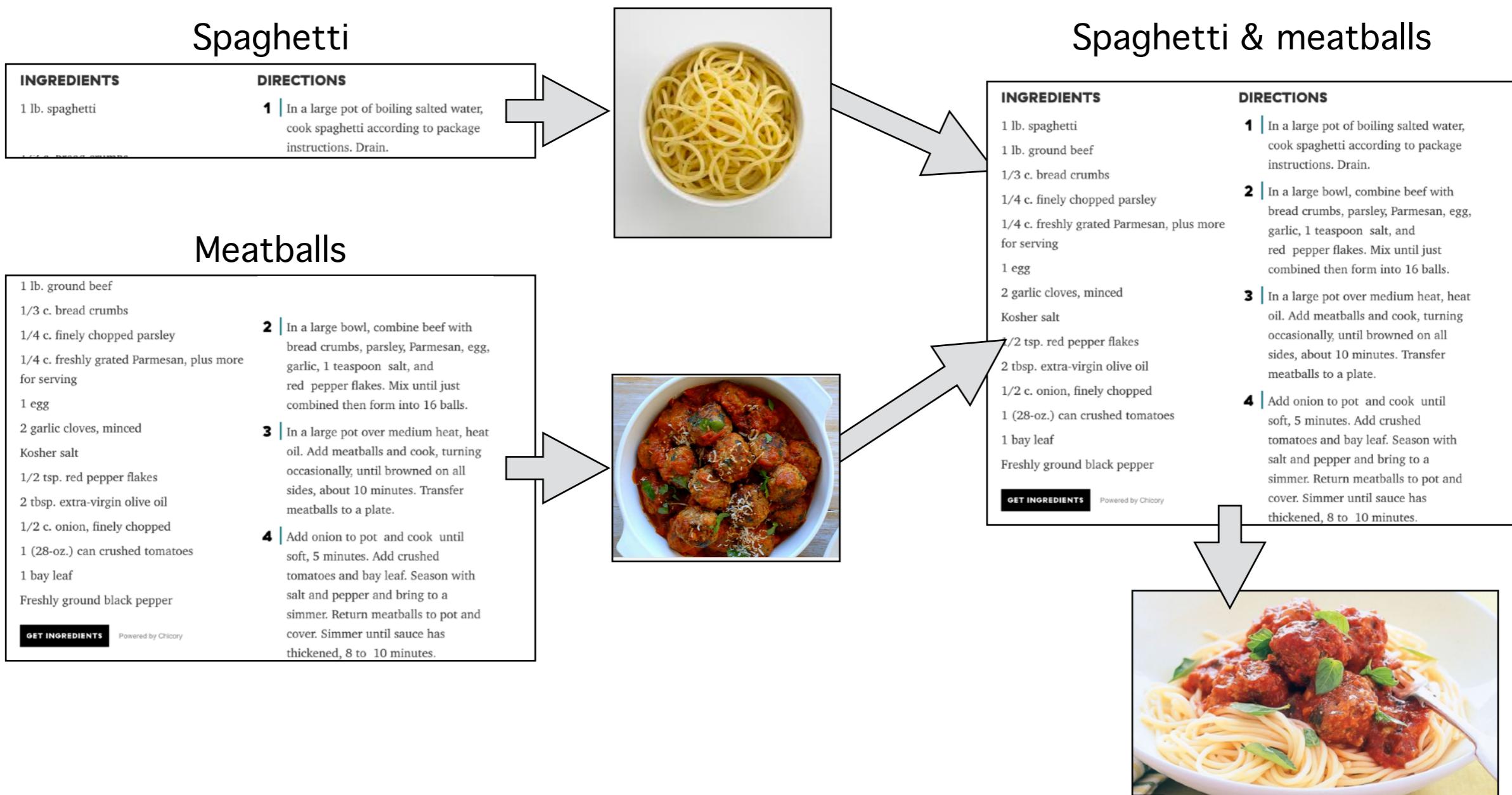
I get it! You could make a function for awesomeness which requires both Bunny and Gladly!

But Bunny has to come first of course.

`awesomeness(Bunny, Gladly)`

“Nesting” functions

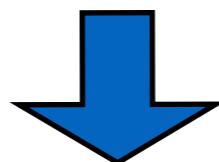
Just as a recipe can use the output of other recipes as ingredients, so too functions can use the output of other functions as arguments



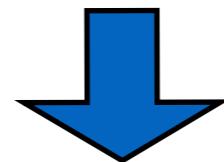
“Nesting” functions

Functions can take other functions as arguments!

```
sqrt ( round(4.45) )
```



```
sqrt ( 4 )
```



2

The inside functions “convert” to their answers.
So to evaluate it, you work from the inside out.

“Nesting” functions

Functions can take other functions as arguments!

```
sqrt ( round(4.45) )
```

Note that the parentheses are balanced. If they aren't, this can cause a problem because R won't know what goes inside what.

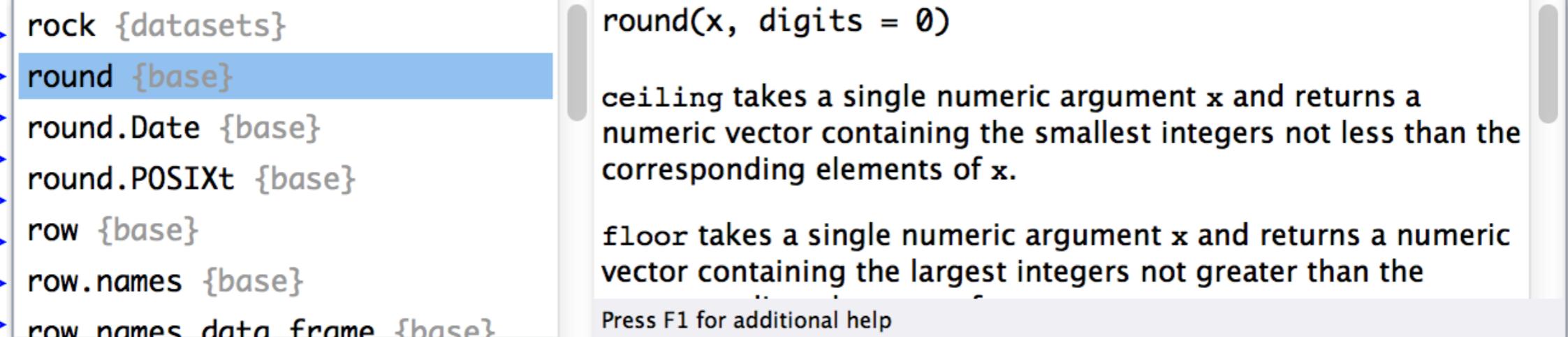
```
> sqrt(round4.33))  
Error: unexpected ')' in "sqrt(round4.33))"
```

```
> sqrt(round(4.33))  
+ |
```



Navigation hints

“Tab” autocomplete



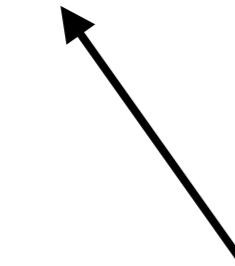
A screenshot of an RStudio session window. The command line shows:

```
> rock {datasets}  
> round {base} round  
> round.Date {base}  
> round.POSIXt {base}  
> row {base}  
> row.names {base}  
> row.names data frame {base}  
> ro |
```

A completion dropdown is open, listing:

- round(x, digits = 0)
- ceiling takes a single numeric argument **x** and returns a numeric vector containing the smallest integers not less than the corresponding elements of **x**.
- floor takes a single numeric argument **x** and returns a numeric vector containing the largest integers not greater than the

Press F1 for additional help



Type **ro** and then hit tab.

Brings up a window showing possible commands you might like to use

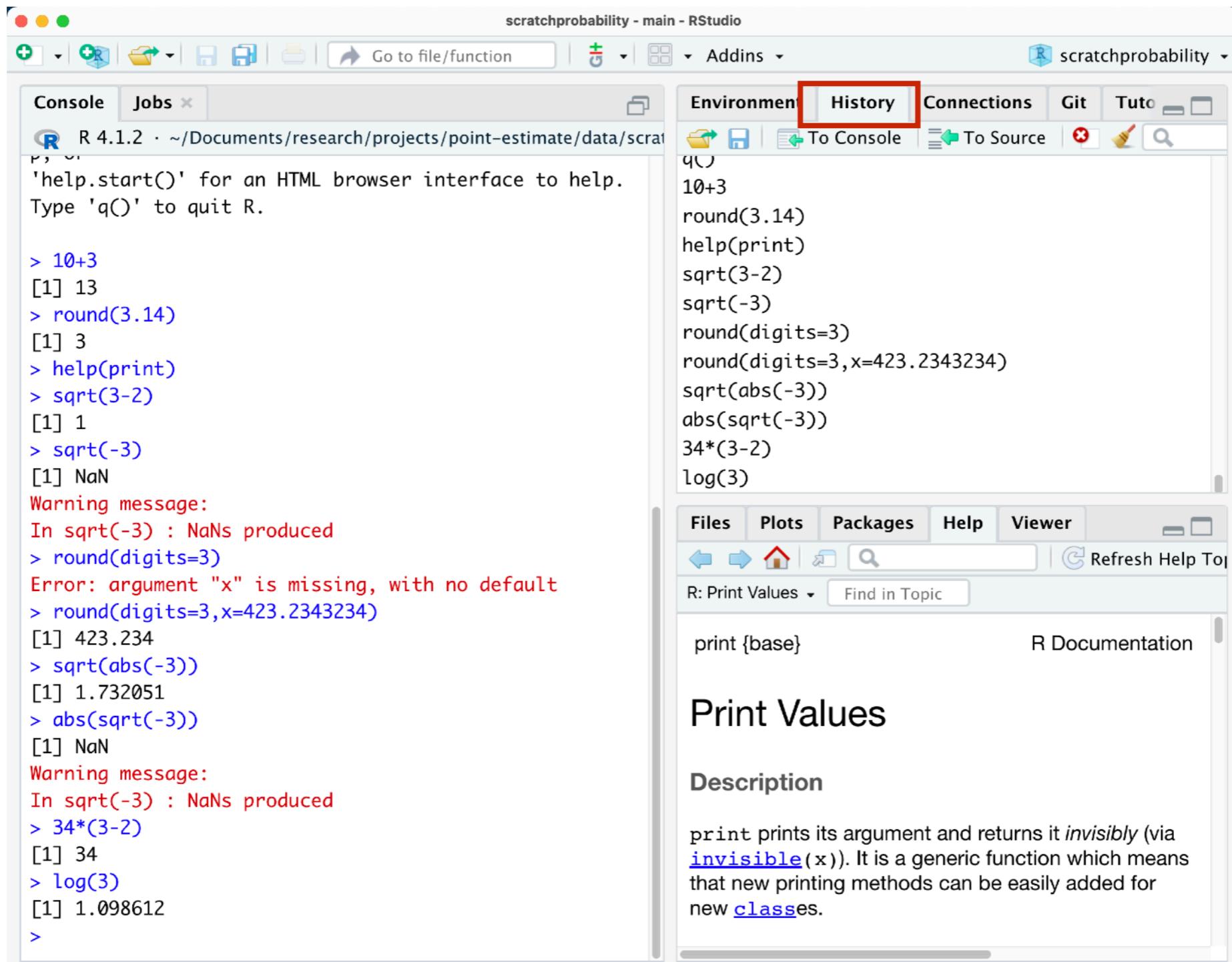
The up arrow

```
> 3+4+5  
[1] 12  
> 3*4*5  
[1] 60  
> sqrt(4)  
[1] 2
```

If you type the up arrow it will let you go through all your previous commands in reverse order.

(This would first show **sqrt(4)**, then **3*4*5**, then **3+4+5**)

The history panel



Shows you all your previous commands in reverse order

The `help()` function

Suppose you want to know more about a function...

```
print( "hello" )
```

Every function comes with documentation

`help(print)` or `?print`

R documentation

When you type `help()`, it shows up in the lower right panel

The screenshot shows the RStudio interface with the following components:

- Top Bar:** Includes icons for file operations (New, Open, Save, etc.), "Go to file/function", "Addins", and "Project: (None)".
- Left Panel:** Contains the script editor with the code:1 # this is my first script
2 # it's just for DRIP class
3 #
4 # author: Amy Perfors
5
6 # define some variables
7 age <- 19
8 box <- "cat"
9
10 # print something
11 print(box)
12 print(age)12:13 (Top Level) R Script
- Environment Tab:** Shows the Global Environment with variables `age` (19) and `box` ("cat").
- Console Tab:** Displays the output of running the script, including help information and the results of `print()` calls.
- Help Viewer:** A red box highlights the bottom-right panel, which contains the help documentation for `print`.
 - Header:** Files, Plots, Packages, Help, Viewer (Help is selected).
 - Title:** R: Print Values Find in Topic
 - Usage:** `print(x, ...)`
 - Description:** `print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new [classes](#).
 - Usage:** `print(x, ...)`
 - Note:** ## S3 method for class 'factor'

R documentation

The screenshot shows the RStudio interface with the 'Viewer' tab selected. The title bar says 'R: Print Values'. The main area displays the documentation for the 'print' function. A red box highlights the code 'print(base)'. An arrow points from this highlighted code to the 'Description' section below. Another arrow points from the 'Description' section to the explanatory text.

Print Values

Description

print prints its argument and returns it *invisibly* (via [invisible\(x\)](#)). It is a generic function which means that new printing methods can be easily added for new [classes](#).

Usage

```
print(x, ...)

## S3 method for class 'factor'
print(x, quote = FALSE, max.levels = NULL,
      width =getOption("width"), ...)

## S3 method for class 'table'
print(x, digits =getOption("digits"), quote = FALSE,
      na.print = "", zero.print = "0", justify = "none", ...)

## S3 method for class 'function'
print(x, useSource = TRUE, ...)
```

tells you what function the documentation is for

describes what the function does

R documentation

The screenshot shows the RStudio interface with the 'Viewer' tab selected. The title bar says 'R: Print Values'. The main content area displays the R documentation for the 'print' function. The 'Usage' section is highlighted with a red box and has two arrows pointing to it from the explanatory text on the right.

print {base}

R Documentation

Print Values

Description

print prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new [classes](#).

Usage

```
print(x, ...)
```

```
## S3 method for class 'factor'  
print(x, quote = FALSE, max.levels = NULL,  
      width = getOption("width"), ...)
```

```
## S3 method for class 'table'  
print(x, digits = getOption("digits"), quote = FALSE,  
      na.print = "", zero.print = "0", justify = "none", ...)
```

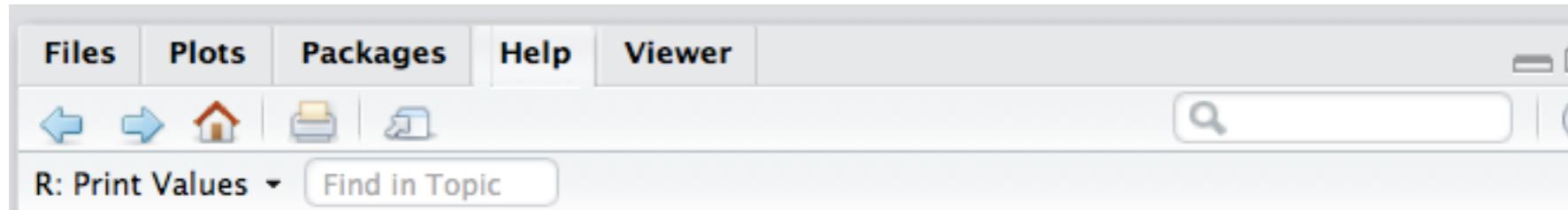
```
## S3 method for class 'function'  
print(x, useSource = TRUE, ...)
```

what you have
to type in order
to get the
function to run

which
arguments are
obligatory

indicates there
are optional
arguments

R documentation



The screenshot shows the RStudio interface with the 'Viewer' tab selected. The title bar says 'R: Print Values'. The main content area displays the documentation for the 'print' function, specifically the base version. It includes the source code for the S3 methods for 'factor', 'table', and 'function' classes, which are highlighted with a red box.

print {base}

R Documentation

Print Values

Description

print prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new [classes](#).

Usage

```
print(x, ...)

## S3 method for class 'factor'
print(x, quote = FALSE, max.levels = NULL,
      width = getOption("width"), ...)

## S3 method for class 'table'
print(x, digits = getOption("digits"), quote = FALSE,
      na.print = "", zero.print = "0", justify = "none", ...)

## S3 method for class 'function'
print(x, useSource = TRUE, ...)
```

don't worry
about this!

...scrolling down...

The screenshot shows the R Help Viewer window with the title bar "Files Plots Packages Help Viewer". Below the title bar, there are icons for back, forward, search, and help. The main content area has a header "R: Print Values" and a "Find in Topic" button. The text area begins with a comment line "#> S3 method for class 'function'" followed by the function definition "print(x, useSource = TRUE, ...)". The word "Arguments" is bolded. Below it, the arguments are listed:

- x an object used to select a method.
- ... further arguments passed to or from other methods.
- quote logical, indicating whether or not strings should be printed with surrounding quotes.
- max.levels integer, indicating how many levels should be printed for a factor; if 0, no extra "Levels" line will be printed. The default, NULL, entails choosing max.levels such that the levels print on one line of width width.
- width only used when max.levels is NULL, see above.
- digits minimal number of *significant* digits, see [print.default](#).
- na.print character string (or NULL) indicating NA values in printed output, see [print.default](#).
- zero.print character specifying how zeros (0) should be printed; for sparse tables, using " ." can produce more readable results, similar to printing sparse matrices in [Matrix](#).
- justify character indicating if strings should left- or right-justified or left alone, passed to [format](#).
- useSource logical indicating if internally stored source should be used for printing when present, e.g., if [options\(keep.source = TRUE\)](#) has been in use.

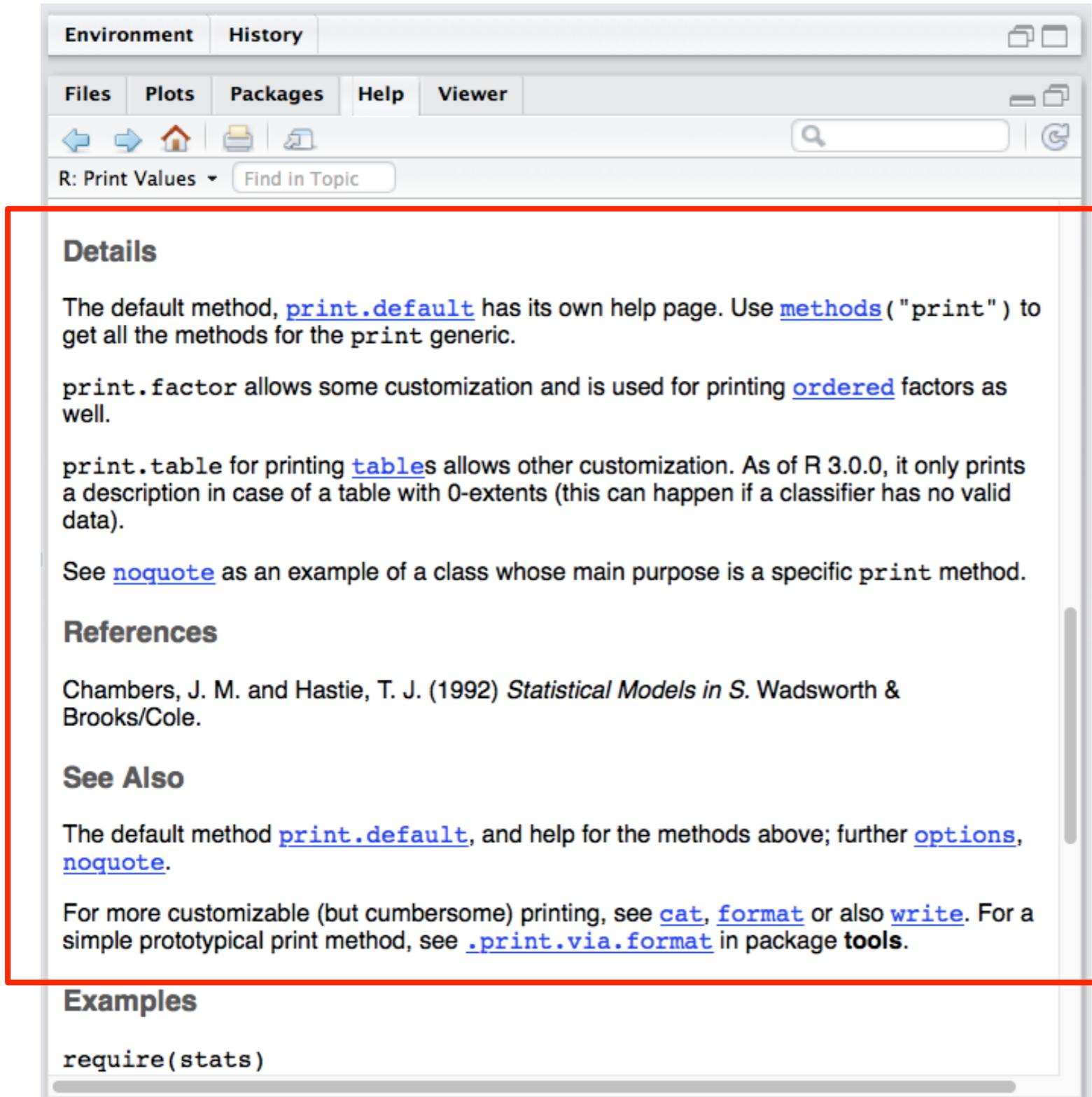
A red box highlights the entire list of arguments. A black arrow points from the word "Arguments" in the title bar to the start of the highlighted list. Another black arrow points from the word "Arguments" in the bolded header to the start of the highlighted list.

here it tells you what it needs to take as an argument

remember this was required

and these are optional

... scrolling even more...



The screenshot shows the R help viewer interface. The title bar says "R: Print Values". The menu bar includes "Environment", "History", "Files", "Plots", "Packages", "Help", and "Viewer". Below the menu is a toolbar with icons for back, forward, search, and help. The main content area has a red border and contains the following text:

Details

The default method, [print.default](#) has its own help page. Use [methods\("print"\)](#) to get all the methods for the `print` generic.

`print.factor` allows some customization and is used for printing [ordered](#) factors as well.

`print.table` for printing [tables](#) allows other customization. As of R 3.0.0, it only prints a description in case of a table with 0-extents (this can happen if a classifier has no valid data).

See [noquote](#) as an example of a class whose main purpose is a specific `print` method.

References

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S*. Wadsworth & Brooks/Cole.

See Also

The default method [print.default](#), and help for the methods above; further [options](#), [noquote](#).

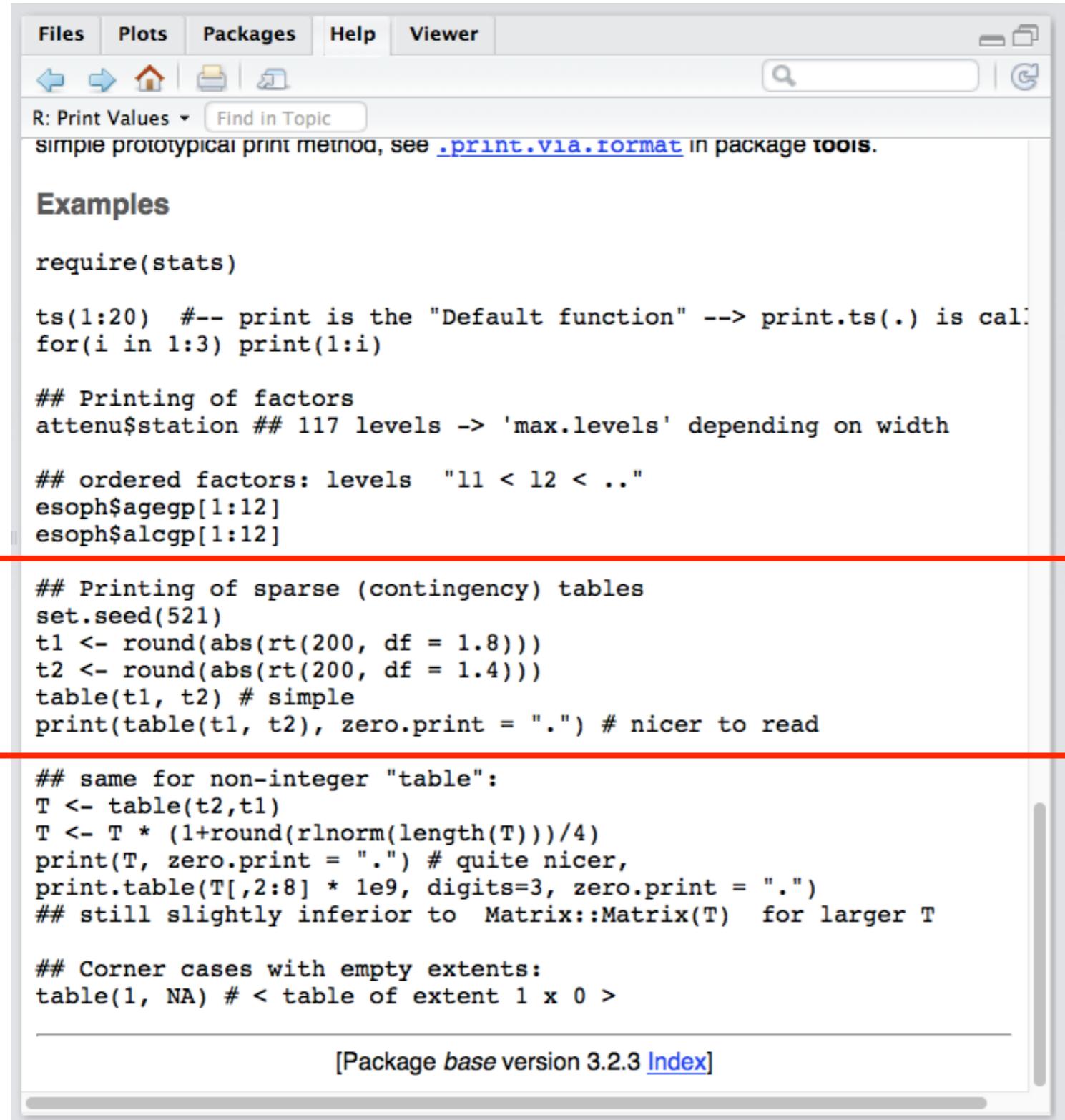
For more customizable (but cumbersome) printing, see [cat](#), [format](#) or also [write](#). For a simple prototypical print method, see [.print.via.format](#) in package [tools](#).

Examples

```
require(stats)
```

you can pretty much ignore all of this (it's far advanced of what you'll need usually)

the end of the scrolling...



The screenshot shows the R Help Viewer window with the title "R: Print Values". The main content area displays the "Examples" section of the "print" topic. A red box highlights a specific code block related to printing sparse contingency tables. The code is as follows:

```
## Printing of sparse (contingency) tables
set.seed(521)
t1 <- round(abs(rt(200, df = 1.8)))
t2 <- round(abs(rt(200, df = 1.4)))
table(t1, t2) # simple
print(table(t1, t2), zero.print = ".") # nicer to read
```

Below this highlighted block, there is more code related to non-integer tables and corner cases.

[Package base version 3.2.3 [Index](#)]

These can be useful to make sense of how to use some of the optional arguments.

But if they are confusing it's because it's almost certainly not something you need to understand!

Other debugging hints

** Remember that 80% of programming is debugging. Everyone has to debug *all of the time*. Having bugs means nothing about your skill (or lack thereof). Nothing at all. Good programmers = good debuggers.

1. Double-check that you have spelled everything right and have all of the right punctuation! Especially brackets. R is *super* picky.
2. READ AND THINK ABOUT THE ERROR MESSAGE. It is often really helpful. If you don't know what it means, google it exactly.
3. If you have a long command or multiple lines, separate it into parts and make sure each part is doing what you expect, one by one
4. Experiment around and try things! It's really really hard to break R. Do strange stuff, that's how you learn.

Exercises

1. Calculate the square root of 81.
2. What happens if you try to take the square root of -81?
Use the `abs()` function to make it non-negative first.
3. What is the exponent of the log of 2?
4. Round 4328.21874 to two digits after the decimal place.
How can you make it round it to 4000?
5. See if you can figure out how the `floor()` function is different from `round()`. Hint: just experiment around by giving it different values, or use the `help()` function.