CALLING FUNCTIONS - FUNCTION ARGUMENTS

DSC 205 - Advanced introduction to data science

Marcus Birkenkrahe

January 30, 2025

README

You will learn:

- ☐ The meaning of function argument matching: `f(args)`
- ☐ Difference between exact, partial, positional, mixed arguments
- ☐ Use of ellipses in functions

Argument matching

- Argument matching conditions allow you to provide arguments to functions either with abbreviated names or without names at all.
- **Exact** argument matching means that the argument tag is written out in full.
- **Partial** argument matching means that the argument tags are abbreviated, e.g. `nr` instead of `nrow`.
- **Positional** argument matching means that the arguments are inferred by their position only (no tags, e.g. `3` instead of `nrow=3`).
- **Mixed** argument matching means that different matching styles are mixed in one function argument list.
- **Elliptic** arguments stand for any type and number of argument.

Exact argument matching

Def: All argument tags are spelled out in full.

PRO:

- Safe - less prone to wrong argument specifications
- Order of arguments is irrelevant
- Useful when many arguments are possible

CON:

- Can be cumbersome for simple operations
- Need to know the full, case-sensitive tags

Example:

```
foo <- matrix(
  data=1:9,
  nrow=3,
  dimnames=list(LETTERS[1:3], # 'dimnames' must be a list
LETTERS[4:6]))
foo
```

```
  D E F
A 1 4 7
B 2 5 8
C 3 6 9
```

Switching around the arguments:

```
bar <- matrix(
  nrow=3,
  dimnames=list(LETTERS[1:3],
LETTERS[4:6]),
  data=1:9)
bar
```

```
  D E F
A 1 4 7
B 2 5 8
C 3 6 9
```

- What is `foo == bar`?

```
foo==bar
```

```
     D    E    F
A TRUE TRUE TRUE
B TRUE TRUE TRUE
C TRUE TRUE TRUE
```

Partial argument matching

Def: Argument tags are abbreviated, e.g. `nr` instead of `nrow`.

PRO:

- Requires less code than exact matching
- Argument tags are still visible
- Order of arguments does not matter

CON:

- Must be aware of other potentially matching arguments
- Each tag must have a unique identification

Example:

```
baz <- matrix(
  da=1:9,
  nr=3,
  di=list(LETTERS[1:3],
  LETTERS[4:6]))
baz
```

```
  D E F
A 1 4 7
B 2 5 8
C 3 6 9
```

- Change `da` to `d` - what happens and why?[1]

[1] The argument tag `d` could belong to `dimnames` or `data` - R cannot resolve this ambiguity on it own and returns an error.

```
baz <- matrix(
  d=1:9,
  nr=3,
  di=list(LETTERS[1:3],
  LETTERS[4:6]))
baz

Error in matrix(d = 1:9, nr = 3, di = list(LETTERS[1:3], LETTERS[4:6])) :
  argument 1 matches multiple formal arguments
  D E F
A 1 4 7
B 2 5 8
C 3 6 9
```

Positional argument matching

Def: Arguments are inferred by their position.

PRO:

- Shorter, cleaner code
- Faster for routine tasks and simple code
- No need to remember specific argument tags

CON:

- Must look up and be aware of the exact defined order of arguments
- Reading code written by others might be more difficult
- Unfamiliar functions written by you or others will slow you down
- The argument order information can be found in the `Usage` section of the function's `help` file
- The argument order can be shown with `args` or by printing the function name without arguments:

```
args(matrix)

function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
NULL
```

Example:

```
bar <-
  matrix(
    1:9,  # data
    3,    # nrow
    3,    # ncol
    F,    # byrow
    list(LETTERS[1:3],LETTERS[4:6])) #dimnames
bar
```

```
  D E F
A 1 4 7
B 2 5 8
C 3 6 9
```

- What happens if you leave out the `byrow` argument value?

```
bar2 <-
  matrix(
    1:9,  # data
    3,    # nrow
    3,    # ncol
    list(LETTERS[1:3],LETTERS[4:6])) #dimnames
bar2
```

```
Error in matrix(1:9, 3, 3, list(LETTERS[1:3], LETTERS[4:6])) :
  invalid 'byrow' argument
Error: object 'bar2' not found
```

Mixed argument matching

Def: Mixing exact, partial, and positional argument matching styles

Example:

```
bar <-
  matrix(1:9,3,3,
 dim = list(c("A","B","C"),c("C","D","E")))
bar
```

```
  C D E
A 1 4 7
B 2 5 8
C 3 6 9
```

Use of ellipses in arguments: *variadic* vs. *elliptic*

- Many functions exhibit *variadic* behavior, i.e. they accept a variable number of arguments, or no arguments at all
- E.g. when you call `data.frame`, you can specify any number of members as arguments:

```
data.frame(...,
           row.names = NULL,
           check.rows = FALSE,
           check.names = TRUE,
           fix.empty.names = TRUE,
           stringsAsFactors = FALSE)
```

- What happens when you specify NO arguments for `data.frame`?

```
df <- data.frame()
df
```

```
data frame with 0 columns and 0 rows
```

- The *ellipsis* in the `Usage` section of the `help` signifies this.
- `args` will also tell you:

```
args(data.frame)
```

```
function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE,
    fix.empty.names = TRUE, stringsAsFactors = FALSE)
NULL
```

- R functions fall into two groups:
  1. ellipsis is the main ingredient (like `c` or `data.frame`)
  2. ellipsis is a supplement (like `plot`)
- `plot` is not variadic but accepts ellipsis arguments:

```
args(plot)
```

```
function (x, y, ...)
NULL
```

- What type of ellipsis arguments does `plot` accept? See the `help`:

  ... : other graphical parameters (see `par` and section `'Details'` below).

Exercises

1. Is `matrix` elliptic?

   ```
   args(matrix) # no
   ```

   ```
   function (data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)
   NULL
   ```

2. Use positional matching with `seq` to create a sequence of values between -4 and 4 that progresses in steps of 0.2.

   ```
   seq(-4,4,0.2)
   ```

   ```
    [1] -4.0 -3.8 -3.6 -3.4 -3.2 -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1
   [17] -0.8 -0.6 -0.4 -0.2  0.0  0.2  0.4  0.6  0.8  1.0  1.2  1.4  1.6  1.8  2.0  2
   [33]  2.4  2.6  2.8  3.0  3.2  3.4  3.6  3.8  4.0
   ```

3. Identify, which style of argument matching is being used: exact, partial, positional, or mixed. If mixed, which arguments are specified?

   (a) `array`

   ```
   array(8:1,dim=c(2,2,2)) # mixed, data is positional
   ```

   ```
   , , 1

        [,1] [,2]
   [1,]    8    6
   [2,]    7    5

   , , 2
   ```

```
     [,1] [,2]
[1,]    4    2
[2,]    3    1
```

(b) rep

```
rep(1:2,3)  # positional
```

```
[1] 1 2 1 2 1 2
```

(c) seq

```
seq(from=10,to=8,length=5)  # exact
```

```
[1] 10.0  9.5  9.0  8.5  8.0
```

(d) sort

```
sort(decreasing=T,x=c(2,1,1,2,0.3,3,1.3)) #exact
```

```
[1] 3.0 2.0 2.0 1.3 1.0 1.0 0.3
```

(e) which

```
matrix(c(T,F,T,T),2,2)
which(matrix(c(T,F,T,T),2,2))  # positional
```

```
      [,1] [,2]
[1,]  TRUE TRUE
[2,] FALSE TRUE
[1] 1 3 4
```

1. which

```
which(matrix(c(T,F,T,T),2,2),a=T) # mixed, arr.ind as a
args(which)
```

```
     row col
[1,]   1   1
[2,]   1   2
[3,]   2   2
function (x, arr.ind = FALSE, useNames = TRUE)
NULL
```

```
args(which)

function (x, arr.ind = FALSE, useNames = TRUE)
NULL
```

Glossary

TERM	MEANING
Exact arguments	Full argument tag
Partial argument	Argument tags abbreviated
Positional argument	Arguments inferred by position alone
Mixed arguments	Different matching styles are mixed
Ellipsis	Variable number of arguments is accepted
`args`	Return exact argument tags with defaults
`...`	Ellipsis in the `args` or `Usage` section of the `help`

References

- Davies, T.D. (2016). The Book of R. NoStarch Press.