

Function design

<https://design.tidyverse.org>

July 2022

Hadley Wickham

@hadleywickham

Chief Scientist, RStudio



Five reasonably complete chapters about defaults

1. Required arguments shouldn't have defaults

<https://design.tidyverse.org/def-required.html>

2. Enumerate possible options

<https://design.tidyverse.org/def-enum.html>

3. Avoid magical defaults

<https://design.tidyverse.org/def-magical.html>

4. Keep defaults short and sweet

<https://design.tidyverse.org/def-short.html>

5. Explain important defaults

<https://design.tidyverse.org/def-inform.html>

Enumerate possible options

```
rank ← function(x, ties.method =  
  c("average", "first", "last", "random", "max", "min")) {  
  
  ties.method ← match.arg(ties.method)  
}
```

Enumerate possible options

```
library(rlang)
```

```
rank ← function(x, ties.method =  
  c("average", "first", "last", "random", "max", "min")) {  
  
  ties.method ← arg_match(ties.method)  
}
```

Documentation

```
#' @param ties.method For each set of ties:  
# ' * ` "average" ` (default): replaces by mean.  
# ' * ` "first" `: break ties with index.  
# ' * ` "last" `: break ties with -index.  
# ' * ` "random" `: put in random order.  
# ' * ` "max" `: replace with maximum.  
# ' * ` "min" `: replace with minimum.  
# Typical sports ranking.
```

Keep defaults short and sweet

```
rank ← function(x, ties.method = "average") {  
  ties.method ← arg_match(ties.method, possible_ties)  
}  
possible_ties ← c(  
  "average", "first", "last", "random", "max", "min"  
)
```

```
rank ← function(x, ties.method = NULL) {  
  if (is.null(ties.method)) {  
    ties.method ← "average"  
  } else {  
    ties.method ← arg_match(ties.method, possible_ties)  
  }  
}  
  
possible_ties ← c(  
  "average", "first", "last", "random", "max", "min"  
)
```

```
rank ← function(x, ties.method = NULL) {  
  if (is.null(ties.method)) {  
    cli::cli_inform('Using `ties.method` = "average"')  
    ties.method ← "average"  
  } else {  
    ties.method ← arg_match(ties.method, possible_ties)  
  }  
}  
  
possible_ties ← c(  
  "average", "first", "last", "random", "max", "min"  
)
```



```
rank ← function(x, ties.method = NULL) {  
  ties.method ← arg_match(  
    ties.method %||% "average",  
    possible_ties  
  )  
}  
  
`%||%` ← function(x, y) if (is.null(x)) y else x  
possible_ties ← c(  
  "average", "first", "last", "random", "max", "min"  
)
```

?reshape

Instead use NULL

```
reshape ← function( ... , split = NULL)
```

```
  if (is.null(split)) {
```

```
    if (sep == "") {
```

```
      split ← list(regex = "[A-Za-z][0-9]", include = TRUE)
```

```
    } else {
```

```
      split ← list(regex = sep, include = FALSE, fixed = TRUE)
```

```
    }
```

```
  }
```

```
  ...
```

```
}
```

```
reshape ← function( ... , split_pattern = NULL, split_include = NULL,  
split_fixed = NULL)
```

```
  split_pattern ← split_pattern %||% if (sep == "") "[A-Za-z][0-9]"
```

```
  if (is.null(split)) {
```

```
    if (sep == "") {
```

```
      split ← list(regex = "[A-Za-z][0-9]", include = TRUE)
```

```
    } else {
```

```
      split ← list(regex = sep, include = FALSE, fixed = TRUE)
```

```
    }
```

```
  }
```

```
  ...
```

```
}
```

Or make a helper

```
reshape ← function( ... , split = reshape_split(sep))  
  ...  
}
```

```
reshape_split ← function(sep) {  
  if (sep == "") {  
    list(regex = "[A-Za-z][0-9]", include = TRUE)  
  } else {  
    list(regex = sep, include = FALSE, fixed = TRUE)  
  }  
}
```

```
# I'd probably go even further here since the split argument  
# has a very specific form
```

Or make a helper

```
reshape <- function(..., split = reshape_split_default(sep))
  ...
}

reshape_split_default <- function(sep) {
  if (is.null(split)) {
    if (sep == "") {
      reshape_split("[A-Za-z][0-9]")
    } else {
      reshape_split(sep, include = FALSE, fixed = TRUE)
    }
  }
}

reshape_split <- function(regexp, include = TRUE, fixed = FALSE) {
  list(regexp = regexp, include = include, fixed = fixed)
}

# I'd probably go even further here since the split argument
# has a very specific form
```