

# Package: csutil (via r-universe)

July 1, 2026

**Title** Common Base-R Problems Relating to Lists

**Version** 2023.4.25

**Description** Utility functions that help with common base-R problems relating to lists. Lists in base-R are very flexible. This package provides functions to quickly and easily characterize types of lists. That is, to identify if all elements in a list are null, data.frames, lists, or fully named lists. Other functionality is provided for the handling of lists, such as the easy splitting of lists into equally sized groups, and the unnesting of data.frames within fully named lists.

**Depends** R (>= 3.3.0)

**License** MIT + file LICENSE

**URL** <https://niphr.github.io/csutil/>, <https://github.com/niphr/csutil>

**BugReports** <https://github.com/niphr/csutil/issues>

**Encoding** UTF-8

**Imports** data.table, magrittr, ggplot2

**Suggests** testthat, knitr, rmarkdown, rstudioapi, glue

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**Config/Needs/website** niphr/cstemplate

**Repository** <https://niphr.r-universe.dev>

**Date/Publication** 2026-07-01 09:02:27 UTC

**RemoteUrl** <https://github.com/niphr/csutil>

**RemoteRef** HEAD

**RemoteSha** b5fc9e939919928dad6e4e301ddf3f63a3467bd3

## Contents

apply_fn_via_hash_table . . . . .	2
easy_split . . . . .	3
is_all_list_elements_null_or_df . . . . .	3
is_all_list_elements_null_or_fully_named_list . . . . .	4
is_all_list_elements_null_or_list . . . . .	5
is_fully_named_list . . . . .	5
unnest_dfs_within_list_of_fully_named_lists . . . . .	6
<b>Index</b>	<b>8</b>

---

apply\_fn\_via\_hash\_table

*Apply a function via hash table*

---

### Description

Extracts the unique values of `x`, applies `fn` to those unique values, and maps the results back to the original vector. This avoids redundant computation when `x` contains many repeated values.

### Usage

```
apply_fn_via_hash_table(x, fn, ...)
```

### Arguments

<code>x</code>	A vector whose values need a function applied.
<code>fn</code>	A function to apply to the unique values of <code>x</code> .
<code>...</code>	Additional arguments passed to <code>fn</code> .

### Details

This can dramatically speed up computation if there is a lot of data and a limited number of unique values: `fn` is called once per unique value rather than once per element.

### Value

A vector of the same length as `x`, containing the result of applying `fn` to each element (computed via unique-value lookup).

### Examples

```
x <- c("a", "b", "a", "c", "b", "a")
apply_fn_via_hash_table(x, toupper)

# passing extra arguments to fn
nums <- c(1.1, 2.2, 1.1, 3.3)
apply_fn_via_hash_table(nums, round, digits = 0)
```

---

easy_split	<i>Split a vector into a list of vectors</i>
------------	--

---

**Description**

Easily split a list into a list of equally sized vectors.

**Usage**

```
easy_split(x, size_of_each_group = NULL, number_of_groups = NULL)
```

**Arguments**

x	The vector to be split
size_of_each_group	If you want to split 'x' into a number of groups, each of 'size_of_each_group' size
number_of_groups	How many equally sized groups do you want?

**Details**

You can either specify the length of the list (via 'number\_of\_groups') or the length of the equally sized vectors within each list element (via 'size\_of\_each\_group'). The last element of the list can be shorter than the other elements.

**Value**

A list containing equally sized vectors.

**Examples**

```
easy_split(letters[1:20], size_of_each_group = 3)
easy_split(letters[1:20], number_of_groups = 3)
```

---

is_all_list_elements_null_or_df	<i>Are all elements in a list null or data.frames?</i>
---------------------------------	--

---

**Description**

Checks if A) 'x' is a list, B) All elements in 'x' are either null or data.frame.

**Usage**

```
is_all_list_elements_null_or_df(x)
```

**Arguments**

x                    An object

**Value**

Boolean.

**Examples**

```
is_all_list_elements_null_or_df(data.frame())
is_all_list_elements_null_or_df(list(data.frame()))
is_all_list_elements_null_or_df(list(1, NULL))
is_all_list_elements_null_or_df(list(data.frame(), NULL))
is_all_list_elements_null_or_df(list("a"=1, 2))
```

---

```
is_all_list_elements_null_or_fully_named_list
```

*Are all elements in a list null or fully named lists?*

---

**Description**

Checks if A) 'x' is a list, B) All elements in 'x' are either null or fully named lists.

**Usage**

```
is_all_list_elements_null_or_fully_named_list(x)
```

**Arguments**

x                    An object

**Details**

Fully named lists are lists with each element having a name.

**Value**

Boolean.

**Examples**

```
is_all_list_elements_null_or_fully_named_list(data.frame())
is_all_list_elements_null_or_fully_named_list(list(data.frame()))
is_all_list_elements_null_or_fully_named_list(list(1, NULL))
is_all_list_elements_null_or_fully_named_list(list(list(), NULL))
is_all_list_elements_null_or_fully_named_list(list(list("a" = 1), NULL))
is_all_list_elements_null_or_fully_named_list(list("a"=1, 2))
```

---

`is_all_list_elements_null_or_list`*Are all elements in a list null or lists?*

---

**Description**

Checks if A) 'x' is a list, B) All elements in 'x' are either null or list.

**Usage**

```
is_all_list_elements_null_or_list(x)
```

**Arguments**

x                    An object

**Value**

Boolean.

**Examples**

```
is_all_list_elements_null_or_list(data.frame())
is_all_list_elements_null_or_list(list(data.frame()))
is_all_list_elements_null_or_list(list(1, NULL))
is_all_list_elements_null_or_list(list(list(), NULL))
is_all_list_elements_null_or_list(list("a"=1, 2))
```

---

`is_fully_named_list`    *Is this a fully named list?*

---

**Description**

Checks if 'x' is a list with each element named.

**Usage**

```
is_fully_named_list(x)
```

**Arguments**

x                    An object

**Value**

Boolean.

**Examples**

```
is_fully_named_list(list())
is_fully_named_list(list(1))
is_fully_named_list(list("a"=1))
is_fully_named_list(list("a"=1, 2))
```

---

```
unnest_dfs_within_list_of_fully_named_lists
Unnest data.frames within fully named list
```

---

**Description**

Consider the situation where a function returns a list containing two data.frames. If this function is called repeatedly and the return values are stored in a list, we will have a list of fully named lists (each of which contains a data.frame). Typically, we want to extract the two data.frames from this nested list structure (and rbindlist them).

**Usage**

```
unnest_dfs_within_list_of_fully_named_lists(
  x,
  returned_name_when_dfs_are_not_nested = "data",
  ...
)
```

**Arguments**

`x` A list of fully named lists (which then contain data.frames)  
`returned_name_when_dfs_are_not_nested` When `x` is a single list of data.frames, what name should be returned?  
`...` parameters passed to `data.table::rbindlist`

**Value**

Fully named list, each element containing a data.table.

**Examples**

```
x <- list(
  list(
    "a" = data.frame("v1"=1),
    "b" = data.frame("v2"=3)
  ),
  list(
    "a" = data.frame("v1"=10),
    "b" = data.frame("v2"=30),
    "d" = data.frame("v3"=50)
  ),
)
```

```
list(  
  "a" = NULL  
)  
NULL  
)  
print(x)  
csutil::unnest_dfs_within_list_of_fully_named_lists(x)  
  
x <- list(  
  data.frame("v1"=1),  
  data.frame("v3"=50)  
)  
print(x)  
csutil::unnest_dfs_within_list_of_fully_named_lists(  
  x,  
  returned_name_when_dfs_are_not_nested = "NAME",  
  fill = TRUE  
)
```

# Index

`apply_fn_via_hash_table`, 2

`easy_split`, 3

`is_all_list_elements_null_or_df`, 3

`is_all_list_elements_null_or_fully_named_list`,  
4

`is_all_list_elements_null_or_list`, 5

`is_fully_named_list`, 5

`unnest_dfs_within_list_of_fully_named_lists`,  
6