# Package 'nestr'

May 9, 2026

**Title** Build Nesting or Hierarchical Structures

**Version** 0.1.2

**Description** Facilitates building a nesting or hierarchical structure as a list or data frame by using a human friendly syntax.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.1.1

**Imports** magrittr, rlang, vctrs, tidyselect

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Emi Tanaka [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-1455-259X>)

**Maintainer** Emi Tanaka <dr.emi.tanaka@gmail.com>

**Repository** CRAN

**Date/Publication** 2022-02-01 21:20:02 UTC

## Contents

amplify *Amplify the data frame with a given structure*

## Description

The dplyr::mutate function modifies, deletes or creates a new column for a data frame without altering the number of rows. The amplify function can create new columns which generally increase (or amplify) the size of the row dimension. The observations in other columns are duplicated.

## Usage

```
amplify(.data, ...)

## S3 method for class 'data.frame'
amplify(
  .data,
  ...,
  .keep = c("all", "used", "unused", "none"),
  .before = NULL,
  .after = NULL
)
```

## Arguments

| | |
|---|---|
| .data | An object with the data. |
| ... | Name-value pairs. |
| .keep, .before, .after | Use to control which columns are retained and how it is ordered in the output. See documentation of dplyr::mutate for more information. |

## Details

If you are familiar with gene replication process then you can recall these functions in genetic terms; an amplified gene is a duplication of the original while a mutated gene modifies the original state.

## Value

Returns a data frame.

## Examples

```
df <- data.frame(x = 1:3, y = c("a", "b", "b"))
amplify(df, z = nest_in(y, "a" ~ 5,
                           "b" ~ 3))
```

`nest_in` *Create a nested structure*

## Description

This function results in a two column data frame with nested structure. Currently only one parent is supported and child is only specified by giving the number of levels. (This will change shortly).

## Usage

```
nest_in(
  x,
  ...,
  prefix = "",
  suffix = "",
  distinct = FALSE,
  leading0 = FALSE,
  compact = TRUE,
  keyname = NULL
)
```

## Arguments

| | |
|---|---|
| `x` | A vector where each entry is the level of a parent. It may be a factor or character. If character, levels are ordered alphanumerically. |
| `...` | A single integer, character vector or sequence of two-sided formula. If a single integer or character vector then each parent will have children specified by the given value. If it is sequence of two-sided formula, then the left hand side (LHS) specifies the level as an integer or character. E.g. `1` means the first level of the parent vector. If it is a character then it is assumed that it corresponds to the label of the parental level. Vector is supported for LHS. The right hand side (RHS) can only be an integer or a character vector. |
| `prefix` | The prefix for the child labels. |
| `suffix` | The suffix for the child labels. |
| `distinct` | A logical value to indicate whether the child labels across parents should be distinct. The labels are only distinct if the RHS of the formula is numeric. |
| `leading0` | By default it is `FALSE`. If `TRUE`, this is the same as setting `0` or `1`. If a positive integer is specified then it corresponds to the minimum number of digits for the child labels and there will be leading zeros augmented so that the minimum number is met. |
| `compact` | A logical value to indicate whether the returned list should be a compact representation or not. Ignored if distinct is `TRUE` since it's not possible to make compact representation if unit labels are all distinct. |
| `keyname` | The name of the parent variable. It's usually the key that connects the output to another table. |

## Value

A named list where the entry corresponding to the child levels and the names correspond to parental levels.

## Examples

```
# Each element in the supplied the vector has 4 child.
nest_in(1:3, 4)

# prefix and suffix can be added to child labels
# along with other aesthesitics like leading zeroes
# with minimum number of digits.
nest_in(1:3, 10, prefix = "id-", suffix = "xy", leading0 = 4)

# you can specify unbalanced nested structures
nest_in(2:4,
         1 ~ 3,
         2 ~ 4,
         3 ~ 2)

# A `.` may be used to specify "otherwise".
nest_in(c("A", "B", "C", "D"),
             2:3 ~ 10,
               . ~ 3)

# The parental level can be referred by its name or vectorised.
nest_in(c("A", "B", "C"),
         c("A", "B") ~ 10,
                "C" ~ 3)
```

# Index