

Package: structenforcement (via r-universe)

May 7, 2026

Title Struct-Like Data Type Checking and Enforcement

Version 0.2.0

Description Enforcement of field types in lists. A drop-in tool to allow for dynamic input data that might be questionably parsed or cast to be coerced into the specific desired format in a reasonably performant manner.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr, lubridate, rlang

NeedsCompilation yes

Author Samuel Sapire [aut, cre, cph], Sean Barrett [ctb]

Maintainer Samuel Sapire <sapires@protonmail.com>

Repository <https://cran.r-universe.dev>

Date/Publication 2026-04-07 21:07:00 UTC

RemoteUrl <https://github.com/cran/structenforcement>

RemoteRef HEAD

RemoteSha 9ab5bcd25d05bfb32bb27a43dfe0290793c12d57

Contents

bind_as_struct	2
type_check	2

Index	4
--------------	----------

bind_as_struct	<i>Bind as Struct</i>
----------------	-----------------------

Description

Given a set of lists/dataframes, attempt to join them as a dataframe with field types matching the specified template. The default and fastest approach simply relies on `dplyr::bind_rows` to use all fields present in the lists to be joined, while strict mode ensures that the template fields and only the template fields are present.

Usage

```
bind_as_struct(template, ..., .list = NULL, strict = FALSE)
```

Arguments

template	A named list to use as a template.
...	The lists to join
.list	A pre-built list of frames to join, as an alternative to ... Use exactly one of ... or .list.
strict	Use all and only the fields in the template. Default: FALSE

Value

A dataframe containing the combined inputs.

Examples

```
bind_as_struct(list("a" = character(0)), list("a" = 1), list("a" = "a"))
```

type_check	<i>List Type Checking</i>
------------	---------------------------

Description

Given two named objects, go through both and make the types of the second match the types of the first.

Usage

```
type_check(
  template,
  target,
  with_cast = FALSE,
  log_items = c("casts", "missing", "excess", "debug")[c(1, 3)]
)
```

Arguments

- | | |
|------------------------|---|
| <code>template</code> | • A named list to use as a template. |
| <code>target</code> | • A named list to use as the output. |
| <code>with_cast</code> | • If true, edits the target instead of just checking types. |
| <code>log_items</code> | • Which debug info to print. Takes a character vector. By default, logs casts and excess fields (target fields not in template). We expect some missing for the moment. |

Value

The target object, with its types appropriately cast.

Examples

```
type_check(  
  list("a" = character(0), "b" = integer(0)),  
  data.frame("a" = c(1,2), "b" = c(3,4)),  
  TRUE, NULL  
)
```

Index

`bind_as_struct`, [2](#)

`type_check`, [2](#)