Package 'rnames'

May 9, 2026

Title Recursive Display of Items in Nested Lists

Version 1.0.1

Maintainer Diego Ciccia <cicciadiego99@gmail.com>

Description
Recursive display of names and paths of all the items nested within sublists of a list object.

License MIT + file LICENSE

Author Diego Ciccia [aut, cre]

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-03-14 08:30:02 UTC

Contents

`print.rnames` *A print method for rnames*

Description

A customized printed display for rnames output

Usage

```
## S3 method for class 'rnames'
print(x, ...)
```

Arguments

x	A rnames object
...	Undocumented

Value

No return, custom print method for rnames objects.

---

rnames *Recursive function to get names in nested lists*

Description

Recursive display of names and paths of all the items nested within sublists of a list object.

Usage

```
rnames(obj, ignore, ...)
```

Arguments

obj	A list to be traversed.
ignore	A list of sublists to exclude from binary tree traversal. The program will report the ignored sublists as end-points. This option is normally suggested for very deep sublists that may cause recursion errors.
...	Undocumented

Value

A list with entries corresponding to the end-points of the traversed list. The name of each element is the name of the end-point, while the item is a vector of the sublists that lead to the end-point.

Description

The `rnames()` function recursively runs `names()` on a list object and returns a list with the names and paths of all the end items. The paths are arrays cointaining all the sublists that need to be accessed in order to retrieve the corresponding item. The built-in `names()` function only returns the names of the objects on the top-most layer of the list. Therefore, all the other subobjects can only be browsed by reiterating `names()` on their parent object. Instead, `rnames()` returns the name and paths of all the end-points of any list. This allows the user to browse all the non-list elements of a nested list without having to manually inspect each sublist.

The program may halt if the recursion goes too deep. With the `ignore` option, the user can halt the execution of the traversal algorithm beyond certain specified nodes. In this way, the program is prevented from exceeding recursion limits. Nodes can be referenced by their object name (e.g., the output of `names()` on their parent object). Notice that the nodes specified in the `ignore` argument

will be included in the output. The function stops from traversing the nodes nested inside those specified in the ignore option.

By definition, a list object can be very complex to visualize due to the presence of sublists. The key idea is that a list object and its subobjects can be represented as the root and leaves of a tree graph, respectively. Sublists form subtrees which can be inspected for subleaves. An object having no subobjects is the end-point of a list, since it is not a list itself. At the same time, data and other objects are stored in end-points. Thus, if these objects are stored in nested lists, it is surely convenient to traverse the list and show all the subobjects at once.

Examples

```
deep_list <- list(
    A = list(B = 2, C = 3, D = list(L = "A", M = "B", N = "C")),
    B = list(V1 = 2, V2 = 3),
    C = list(V1 = 2, V2 = 3),
    D = 4)
print(rnames(deep_list, ignore = c("D")))
```

---

rnames.list *General rnames method for lisis*

Description

General rnames method for generic lists.

Usage

```
## S3 method for class 'list'
rnames(obj = obj, ignore = c(), ...)
```

Arguments

obj	A list
ignore	A set of sublists to be ignored
...	Undocumented

Value

A list with rname class.

Index