Package 'unnest'

September 22, 2020

Title Unnest Hierarchical Data Structures

Version 0.0.2

Description Fast flattening of hierarchical data structures (e.g. JSON and XML documents) into data.frames with a flexible spec language.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests data.table, dplyr, knitr, repurrrsive, rmarkdown, roxygen2, testthat, tibble, tidyr

VignetteBuilder knitr

URL https://github.com/vspinu/unnest

BugReports https://github.com/vspinu/unnest/issues

NeedsCompilation yes

Author Vitalie Spinu [aut, cre]

Maintainer Vitalie Spinu <spinuvit@gmail.com>

Repository CRAN

Date/Publication 2020-09-22 09:40:02 UTC

R topics documented:

s *Unnest spec is a nested list with the same structure as the nested json. It specifies concisely how the deeply nested components ought to be unnested.* s() *is a shorthand for* spec().

Description

Unnest spec is a nested list with the same structure as the nested json. It specifies concisely how the deeply nested components ought to be unnested. s() is a shorthand for spec().

Unnest nested lists

Usage

```
s(
  selector = NULL,
  ...,
  as = NULL,
  children = NULL,
  groups = NULL,
  include = NULL,
  exclude = NULL,
  stack = NULL,
  process = NULL
)

spec(
  selector = NULL,
  ...,
  as = NULL,
  children = NULL,
  groups = NULL,
  include = NULL,
  exclude = NULL,
  stack = NULL,
  process = NULL
)

unnest(x, spec = NULL, dedupe = FALSE, stack_atomic = FALSE, cross_join = TRUE)
```

Arguments

selector A shorthand syntax for an include selector. When a list each element of the list is expanded into the include element at the respective level. When selector is a string it is expanded into a list according to the following rules:

1. When selector is length 1 and contains "/" characters it is split with "/" separator. For instance s(c("a","b"),...), s("a/b",...) and s("a",s("b",...))

are all converted to a canonical `s(include = "a",s(include = "b",...))`. Components consisting entirely of digits are converted to integer. For example s("a/2/b" ...) is equivalent to `s("a",s(2,s("b",...)))`

2. Each element of the resulting from the previous step vector is split with ,. Thus `s("a/b,c/d")` is equivalent to `s("a",s(include = c("b","c"),s("d",...)))`

`as` name for this field in the extracted data.frame

`children, ...` Unnamed list of children spec. `...` is merged into `children`. `children` is part of the canonical spec.

`groups` Named list of specs to be processed in parallel. The return value is a named list of unnested data.frames. The results is the same as when each spec is `unnested` separately except that `dedupe` parameter of `unnest()` will work across groups and execution is faster because the nested list is traversed once regardless of the number of groups.

`include, exclude`
A list, a numeric vector or a character vector specifying components to include or exclude. A list can combine numeric indexes and character elements to extract.

`stack` Whether to stack this node (TRUE) or to spread it (FALSE). When `stack`is a string an index column is created with that name.

`process` Extra processing step for this element. Either NULL for no processing (the default), "asis" to return the entire element "as is" in a list column, or "paste" to paste elements together into a character column.

`x` a nested list to unnest

`spec` spec to use for unnesting. See `spec()`.

`dedupe` whether to dedupe repeated elements. If TRUE, if a node is visited for a second time and is not explicitly declared in the `spec` the node is skipped. This is particularly useful with `grouped` specs.

`stack_atomic` Whether atomic vectors should be stacked or not.

`cross_join` Specifies how the results from sibling nodes are joined (`cbind`) together. The shorter data.frames (in terms o number of rows) can be either recycled to the max number of rows across all components as with standard R's recycling (`cross_join = FALSE`). Or, with `cross_join = TRUE`, the results are cross joined (aka form all combinations of rows across joined components). `cross_join = TRUE` is the default because of no data loss and it is more conducive for earlier error detection with incorrect specs.

Value

A canonical spec; a list suitable for the C level unnest routine.

Examples

```
## `s()` returns a canonical spec list
s("a")
s("a//c2")
```

```
s("a/2/c2,cid")


x <- list(a = list(b = list(x = 1, y = 1:2, z = 10),
                   c = list(x = 2, y = 100:102)))
xxx <- list(x, x, x)

## spreading
unnest(x, s("a"))
unnest(x, s("a"), stack_atomic = TRUE)
unnest(x, s("a/b"), stack_atomic = TRUE)
unnest(x, s("a/c"), stack_atomic = TRUE)
unnest(x, s("a"), stack_atomic = TRUE, cross_join = TRUE)
unnest(x, s("a//x"))
unnest(x, s("a//x,z"))
unnest(x, s("a/2/x,y"))

## stacking
unnest(x, s("a/", stack = TRUE))
unnest(x, s("a/", stack = TRUE, as = "A"))
unnest(x, s("a/", stack = TRUE, as = "A"), stack_atomic = TRUE)
unnest(x, s("a/", stack = "id"), stack_atomic = TRUE)
unnest(x, s("a/", stack = "id", as = ""), stack_atomic = TRUE)

unnest(xxx, s(stack = "id"))
unnest(xxx, s(stack = "id"), stack_atomic = TRUE)
unnest(xxx, s(stack = "id", s("a/b/y/", stack = TRUE)))

## exclusion
unnest(x, s("a/b/", exclude = "x"))

## dedupe
unnest(x, s("a", s("b/y"), s("b")), stack_atomic = TRUE)
unnest(x, s("a", s("b/y"), s("b")), dedupe = TRUE, stack_atomic = TRUE)

## grouping
unnest(xxx, stack_atomic = TRUE,
       s(stack = TRUE,
         groups = list(first = s("a/b/x,y"),
                       second = s("a/b"))))

unnest(xxx, stack_atomic = TRUE, dedupe = TRUE,
       s(stack = TRUE,
         groups = list(first = s("a/b/x,y"),
                       second = s("a/b"))))

## processing asis
str(unnest(xxx, s(stack = "id",
                  s("a/b/y", process = "asis"),
                  s("a/c", process = "asis"))))
str(unnest(xxx, s(stack = "id", s("a/b/", process = "asis"))))
str(unnest(xxx, s(stack = "id", s("a/b", process = "asis"))))
```

```
## processing paste
str(unnest(x, s("a/b/y", process = "paste")))
str(unnest(xxx, s(stack = TRUE, s("a/b/", process = "paste"))))
str(unnest(xxx, s(stack = TRUE, s("a/b", process = "paste"))))
```

Index