Package ‘searchable’

August 29, 2016

Type Package

Title Tools for Custom Searches / Subsets / Slices of Named R Objects

Version 0.3.3.1

Date 2015-04-06

Author `DecisionPatterns [aut, cre]`

Maintainer Christopher Brown <chris.brown@decisionpatterns.com>

Description Provides functionality for searching / subsetting and slicing named objects using `stringr/i`-style modifiers by case (in)sensitivity, regular expressions or fixed expressions; searches uses the standard `[` operator and allows specification of default search behavior to either the search target (named object) and/or the search pattern.

Depends R (>= 3.1.0)

Imports methods, magrittr(>= 1.5), stringi(>= 0.4.1)

Suggests testthat

License GPL-2

URL https://github.com/decisionpatterns/searchable

BugReports https://github.com/decisionpatterns/searchable/issues

Collate 'Class-Pattern.R' 'Class-PatternOrCharacter.R'
  'Class-Searchables.R' 'Class-Searchable.R'
  'Class-SearchableOrPattern.R' 'boundary.R' 'case.R' 'coll.R'
  'detect.R' 'extract.R' 'fixed.R' 'invert.R' 'is.string.R'
  'regex.R' 'reverse.lookup.R' 'searchable-package.R' 'std.R'
  'stri_detect_std.R' 'zzz.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2015-04-08 00:37:36
**R topics documented:**

searchable-package ................................................. 2
boundary ................................................................. 3
coll ................................................................. 4
detect ................................................................. 5
fixed ................................................................. 6
ignore.case .......................................................... 7
invert ................................................................. 8
is.string .............................................................. 9
Pattern-class ......................................................... 9
regex ................................................................. 11
reverse.lookup ..................................................... 11
Searchable-class ..................................................... 12
std ................................................................. 15
stri_detect_std ....................................................... 16
[,Searchable,PatternOrCharacter,missing-method ................. 17

---

**Description**

Tools For Custom Searches / Subsets / Slices of Named R Objects

**Details**

The 'searchable' package provides flexible methods for subsetting named object by matching the names using case (in)sensitivity, regular or fixed expressions. searches uses the standard '[' operator and allows specification of default search behavior to either the search target (named object) and/or the search pattern.

It was designed to make flexible, high performance dictionary and thesaurus structures.

**References**

http://stackoverflow.com/questions/5671719/case-insensitive-search-of-a-list-in-r

**See Also**

searchable
http://cran.r-project.org/web/packages/qdap
Examples

# ATOMIC VECTORS:
v <- c( a=1, b=2, c=3, c=4, c=5 )
sv <- searchable(v)

# FLEXIBLY FIND ELEMENTS BY NAME
sv[ regex('c') ]
sv[ fixed('c') ]
sv[ ignore.case('b') ]

# FLEXIBLY REPLACEMENT ELEMENTS BY NAME
sv[ regex('c.?') ] <- "3rd"
sv

# SET DEFAULT SEARCH FOR TARGET/OBJECT
sv <- searchable(v, case_insensitive = TRUE )
sv['b']
sv['B']
sv <- regex(sv)
sv['c']
sv <- ignore.case(sv)
sv['b']
sv['c'] # st

# USE ON (RECURSIVE) LISTS:
l <- list( a=1, b=2, c=3 )
sl <- searchable(l)
sl['b']
sl[ ignore.case("B") ]

# USE WITH MAGRITTR
## Not run:
sl[ "B" %>% ignore.case ]
"b" %>% sl[.]
"B" %>% ignore.case %>% sl[.]

## End(Not run)
Description

Sets boundary type matching

Usage

boundary(object, type = c("partial", "full", "word", "sentence", "line",
"starts_with", "ends_with"))

full(object)

partial(object)

word(object)

sentence(object)

startswith(object)

endswith(object)

Arguments

  object target or pattern for search
  type character; one of partial (default), full, word, sentence, line, starts_with, ends_with

Sets the options for matching at specified boundaries. Boundaries may also be supplied to by pattern types; regex, fixed, coll, ... When declared explicitly, these take precedent. Except when the boundad

See Also

  # -tk

Examples

  # -tk

Description

Creates or modifies the search type to use coll matching
Usage

coll(object, ...)

## Default S3 method:
coll(object, ...)

## S3 method for class 'character'
coll(object, ...)

## S3 method for class 'SearchableOrPattern'
coll(object, ...)

Arguments

object to make specification
...
additional arguments passes to pattern
coll

See Also

pattern

Examples

pat <- coll("a")
detect( c('alpha','beta'), pat )

detect  detect
detect  detect

description

which elements matching the search pattern

Usage

detect(str, pattern)

Arguments

str searcheable target
pattern method for searching

Value

logical; vector indicating which elements match
Note

- may not export this function

See Also

.matches

Description

Creates or modifies the search type to use fixed matching

Usage

fixed(object, ...)

## Default S3 method:
fixed(object, ...)

## S3 method for class 'character'
fixed(object, ...)

## S3 method for class 'SearchableOrPattern'
fixed(object, ...)

Arguments

object to make specification
... additional arguments passed to pattern

See Also

pattern

Examples

pat <- fixed("a")
  detect( c('alpha','beta'), pat )
ignore.case  

**Turn on/off case sensitivity for Searchable and Pattern objects**

Description

Functions for affecting the case sensitivity of matching/

Usage

```r
ignore.case(object)

## S3 method for class 'SearchableOrPattern'
ignore.case(object, ...)

## S3 method for class 'character'
ignore.case(object)

## Default S3 method:
ignore.case(object)

case_insensitive(object)

use.case(object)

## S3 method for class 'SearchableOrPattern'
use.case(object)

## S3 method for class 'character'
use.case(object)

## Default S3 method:
use.case(object)

case_sensitive(object)
```

Arguments

- `object` search pattern or target
- `...` additional arguments

ignore.case/case_insensitive and use.case/case_sensitive control the case sensitivity of the matching.
The default is to perform case-sensitive matching.

See Also

`stri_detect_*` from the stringi package
Examples

use.case("pattern")  # case-sensitive (Default)
ignore.case("pattern")  # case-insensitive

invert

## invert

Invert a structure by swapping keys and values

Description

Invert a structure by swapping keys and values

Usage

invert(x)

## S4 method for signature 'vector'
invert(x)

## S4 method for signature 'Searchable'
invert(x)

## S4 method for signature 'list'
invert(x)

Arguments

x  
object to invert

Details

Inverts named vectors

Value

A character vector in which the names are the former values.

Note

- currently applies to atomic vectors only - apply to list (recursive structures), data.frames, matrices and arrays - invert might be an ambiguous name ... call it swap_kv? kvswap? swapKV? swapNV?

Examples

v <- 1:26
names(v) <- letters

invert(v)

l <- as.list(v)
is.string  Test if an object is a string

Description
A string is a one element character vector

Usage
is.string(x)

Arguments

x  
Equivalent to:
   is.character(x) && length(x) == 1

Pattern-class  Defines or extract a search pattern

Description
Patterns defines how searches are conducted against a searchable target

Usage

pattern(object, type, ...)

## Default S3 method:
pattern(object = NULL, type = "std", ...)

## S3 method for class 'character'
pattern(object, type = "std", ...)

## S3 method for class 'Pattern'
pattern(object, type = object@type, ...)

## S3 method for class 'Searchable'
pattern(object, type = object@type, ...)

## S4 method for signature 'Pattern'
show(object)
Pattern-class

Arguments

object character or pattern;
type character; the type of match: std (default), regex, coll, fixed.
... additional arguments to be passed to stri_opts_* functions. See details.

The pattern class defines how the search is conducted.
The function pattern is the constructor for the class. It takes a 'string' can be used to define a pattern that controls matching against a searchable target. Most often the user will want to use the type specific functions: regex, coll, fixed or basic. Each is described below.

These are closely related to the

Slots

.Data character object representing a pattern.
type character; type of search performed; one of "std" (default), "regex", "fixed", "coll", or "char-class". See details.
options list; name = value pairs for search options used.

std

The default is std matching which performs matching as base R would. This is equivalent to fixed and case_insensitive = FALSE. Though the internal matching is sed.

regex

regex matching takes a regular expression for matching using stri_*_regex functions.

coll

...

fixed

...

Examples

pattern('hello')
pattern('hello', type="regex", boundary="starts_with" )
Description

Creates or modifies the search type to use regular expression matching

Usage

regex(object, ...)

## Default S3 method:
regex(object, ...)

## S3 method for class 'character'
regex(object, ...)

## S3 method for class 'SearchableOrPattern'
regex(object, ...)

Arguments

object to make specification
...
additional arguments passes to `pattern`
regex

See Also

`pattern`

Examples

pat <- regex("a.+")
detect(c('alpha','beta'), pat)

---

reverse.lookup

Perform a reverse lookup on searchables

Description

This function causes the pattern search to be performed against an object’s values instead of its names

Usage

reverse.lookup(string)
Arguments

- **String**
  - Pattern for which to match against an object's values.

- **reverse.lookup**
  - Sets and toggles the logical attribute with name `reverse.lookup`.
  - Actual implementation of the reverse lookup is performed in the extract methods.
  - In order to perform a reverse lookup, values must be converted to character names.

Value

- A string object with the `reverse.lookup` attribute set.

reverse.lookup

- When performing a reverse lookup, values (not names) are searched. The corresponding names are returned. NOTE: this is highly experimental and only works for atomic vectors. It is uncertain how this might be applied to recursive structures like lists.

Note

- What happens if there are two `reverse.lookup`?

See Also

- The `invert` function in the `hash` package.

Examples

- `reverse.lookup("string")`

---

**Searchable-class**

**Searchable**

Description

- `searchable` makes a named object a `Searchable` target, optionally specifying the default search options.

Usage

- `searchable(object, type = "std", ...)`

  ```r
  ## S4 method for signature 'Searchable'
  show(object)
  ```
Arguments

object  searchable object or object to be made searchable

type    character; the type of search to perform

... additional arguments defining the search pattern. See ?pattern for details.

Details

The searchable class allows 'stringr/i'-like searches using \[ and \[<\] operators. The following search types are supported:

- std standard R matching, the default
- regex for regular expression matching,
- fixed for fixed string matching,
- coll for collation matching,

Class Searchable objects allow customizations of how R's \[ operator match objects' names.

Value

By default, extraction from a searchable objects does not produce a subset that is also searchable. It is assumed that in most cases, the developer will not want another searchable object and only wish to have the subclass.

Differences from stringr

\[stringr and stringi\] are general purpose string manipulations library allowing flexible search and pattern matching against character strings. The searchable package applies this type of matching to objects' names using the standard \[ accessor. Thus,

\texttt{searchable}\texttt{(sv)[ regex('b') ]}

returns objects the subset of whose names contain 'b'.

Unlike \texttt{stringr/i}, searchable allows search specification to applied to either the search pattern or search target. When applied to the target, a default search method is configured. All subsequent searches of the searchable target will use this default pattern.

The search method can be specified with the type argument of the searchable function or any of match-modifying functions, e.g. fixed, regex, coll, ignore.case, etc. See examples.

When modifiers are applied to both target and pattern, \texttt{modifiers applied to the pattern take precedence} and the target's modifiers are disabled.

Differences from base R

\texttt{searchable} is designed to be minimally invasive. When no search types or options are specified, matching defaults to R's normal behavior.

Here are the other difference from standard R operations:

- $ and
• \[\ldots\] are unaltered by the package. It is unclear, how these operators might accommodate the indeterminate number of matches.
• Searches using multiple patterns recycle the patterns, but rather return elements that match any of the patterns.
• In base R, there is output value every element of input argument, i.e. Input elements that do not match a named element of x return NA. Because of the indeterminate number of matches given a pattern search against a searchable object, there is no guarantee that a search pattern have a match. If no matches are found, a zero-length object is returned. (This may change to NA to be more consistent.)
• Results do not yield a Searchable object, but the superclass that the searchable class wraps. See Value below.

replacement

searchable can be used to replace objects as well. See ?extract for additional examples.

multiple dimension objects

Multiple dimension objects such as data.frames, data.tables, matrices and arrays are not supported at this time.

Note

- Environments cannot be (easily) be made "searchable" due to the way the they are implemented.
- The extraction methods for searchable objects are (at present) limited to only one pattern. This may change in the future.

See Also

extract
stri_detect_regex
reverse.lookup

Examples

# ATOMIC VECTORS:
  v <- c(a=1, b=2, B=3, c=4, c2=5)
  sv <- searchable(v)

# FLEXIBLY FIND ELEMENTS BY NAME
  sv[ regex("c") ]
  sv[ fixed("c") ]
  sv[ ignore.case('b') ]

# FLEXIBLY REPLACEMENT ELEMENTS BY NAME
  sv[ regex("c.?") ] <- "3rd"
# SET DEFAULT SEARCH FOR TARGET/OBJECT
sv <- searchable(v, case_insensitive = TRUE)
sv['b']
sv['B']

sv <- regex(sv)
sv['c']

sv <- ignore.case(sv)
sv['b']
sv['c']  # st

# USE ON (RECURSIVE) LISTS:
l <- list(a=1, b=2, c=3)
sl <- searchable(l)
sl['b']
sl[ ignore.case("B") ]

# USE WITH MAGRITTR
## Not run:
sl[ "B" %>% ignore.case ]
"b" %>% sl[.]
"B" %>% ignore.case %>% sl[.]

## End(Not run)

---

std  

Use/revert to standard matching

Description

Creates or modifies the search type to use default R matching

Usage

std(object, ...)

## Default S3 method:
std(object, ...)

## S3 method for class 'character'
std(object, ...)

## S3 method for class 'SearchableOrPattern'
std(object, ...)
Arguments

- object to make specification
- ... additional arguments passes to pattern

See Also

pattern

Examples

```r
pat <- std("a")
detect( c('alpha','beta'), pat )
```

Description

Functions for matching using standard, default matching

Usage

```r
stri_detect_std(str, pattern, ..., opts_std = NULL)
stri_opts_std(case_insensitive = FALSE, ...)
```

Arguments

- str search target
- pattern pattern to attempt
- ... supplementary arguments passed to the underlying functions, including additional settings for stri_opts_std
- opts_std list; optional arguments used by stri_*_std functions
- case_insensitive logical; enable simple case insensitive matching

Value

logical indicating the matching elements in str
See Also

`stri_detect`

Examples

```r
stri_detect_std( letters[1:5], letters[1:2] )  # TRUE TRUE ...
stri_detect_std( letters[1:5], LETTERS[1:2] )  # ALL FALSE
stri_detect_std( letters[1:5], LETTERS[1:2], opts_std = list(case_insensitive = TRUE ) )
```

Description

Defines `[`, `[`, and `$` for Searchable objects

Usage

```r
## S4 method for signature 'Searchable,PatternOrCharacter,missing'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'Searchable,character,missing'
x[i] <- value
```

Arguments

- `x` Searchable object
- `i` character; pattern with potential match modifiers applied,
- `j` missing; never specified
- `...` additional arguments. See `Extract`
- `drop` For matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See `drop` for further details.
- `value` replacement value for replacement functions

Value

The values after the extracting methods have been applied:

- `\[` returns a subset of `x`, but which is not Searchable.
- `\[\[\[` and `$` return a single element of `x`
[,<- 

[ and [<- are used for subsetting and replacing zero or more elements of x. Used with searchable objects, these operators differ from normal R operations in the following respects:

- The search returns elements of the target that matches ANY of the search patterns.
- Unlike the its normal behavior, \[ does not guarantee the output to have as many elements as elements to pattern.
- [ does not return a Searchable object. It is thought that the return valuable will not be subsequently searched. It is easy to turn the results into a Searchable object using searchable however.
- Unlike for environments and hashes, no constraints exist for ensuring uniqueness for names in vectors and lists. These structures may contain multiple elements with the same name. Normal attempts to extract by name yield only the first element that matches the name. Using a Searchable patterns match yields all matching elements.

See Also

Searchable
Extract
Match mofiers: fixed, regex, coll and ignore.case reverse.lookup

Examples

# ATOMIC VECTORS:
  v <- c( a=1, b=2, B=3, c=4, c2=5 )
  sv <- searchable(v)

# FLEXIBLY FIND ELEMENTS BY NAME
  sv[ regex('c') ]
  sv[ fixed('c') ]

  sv[ ignore.case('b') ]

# FLEXIBLY REPLACEMENT ELEMENTS BY NAME
  sv[ regex('c.') ] <- "3rd"

# SET DEFAULT SEARCH FOR TARGET/OBJECT
  sv <- searchable(v, caseInsensitive = TRUE )
  sv['b']
  sv['B']

  sv <- regex(sv)
  sv['c']

  sv <- ignore.case(sv)
  sv['b']
sv['c']            # st

# USE ON (RECURSIVE) LISTS:
1 <- list(a=1, b=2, c=3)
s1 <- searchable(l)
s1['b']
s1[ ignore.case("B") ]

# USE WITH MAGRITTR
## Not run:
s1[ "B" %>% ignore.case ]
"b" %>% s1[.]
"B" %>% ignore.case %>% s1[.]

## End(Not run)
Index

[,Searchable,PatternOrCharacter,missing-method,regex, 11, 18
   17 reverse.lookup, 11, 14, 18
[<-,Searchable,character,missing-method
   ([,Searchable,PatternOrCharacter,missing-method),
   17 Searchable (Searchable-class), 12
Searchable (Searchable-class), 12
Searchable-class, 12
Searchable-class, 12
searchable-package, 2
sentence (boundary), 3
show,Pattern-method (Pattern-class), 9
show,Searchable-method
   (Searchable-class), 12
startswith (boundary), 3
std, 15
stri_detect, 17
stri_detect_regex, 14
stri_detect_std, 16
stri_opts_std (stri_detect_std), 16
use.case (ignore.case), 7
word (boundary), 3

boundary, 3
case_insensitive (ignore.case), 7
case_sensitive (ignore.case), 7
coll, 4, 18
detect, 5
drop, 17
drop
endswith (boundary), 3
extract, 14
extract
   ([,Searchable,PatternOrCharacter,missing-method),
   17
fixed, 6, 18
full (boundary), 3
ignore.case, 7, 18
invert, 8
invert,list-method (invert), 8
invert,Searchable-method (invert), 8
invert,vector-method (invert), 8
is.string, 9

partial (boundary), 3
Pattern (Pattern-class), 9
pattern, 5, 6, 11, 16
pattern (Pattern-class), 9
Pattern-class, 9
pattern.character (Pattern-class), 9
pattern.default (Pattern-class), 9
pattern.Pattern (Pattern-class), 9
pattern.Searchable (Pattern-class), 9

20