Package 'inverseRegex'

May 8, 2026

Title Reverse Engineers Regular Expression Patterns for R Objects

Version 0.2.0

Description Reverse engineer a regular expression pattern for the characters contained in an R object. Individual characters can be categorised into digits, letters, punctuation or spaces and encoded into run-lengths. This can be used to summarise the structure of a dataset or identify non-standard entries. Many non-character inputs such as numeric vectors and data frames are supported.

Depends R (>= 3.5.0)

Suggests tibble, testthat, knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

BugReports https://github.com/rntq472/inverseRegex/issues

RoxygenNote 7.3.3

VignetteBuilder knitr

Language en-GB

NeedsCompilation no

Author Jasper Watson [aut, cre]

Maintainer Jasper Watson <jasper.g.watson@gmail.com>

Repository CRAN

Date/Publication 2025-10-28 12:00:02 UTC

Contents

inverseRegex 2
occurrencesLessThan 4

Index **6**

`inverseRegex` *Reverse Engineers a Regular Expression Pattern to Represent the Input Object.*

Description

Deconstructs the input into collections of letters, digits, punctuation, and spaces that represent a regex pattern consistent with that input.

Usage

```
inverseRegex(
  x,
  numbersToKeep = c(2, 3, 4, 5, 10),
  combineCases = FALSE,
  combineAlphanumeric = FALSE,
  combinePunctuation = FALSE,
  combineSpace = FALSE,
  sep = "",
  escapePunctuation = FALSE,
  enclose = FALSE,
  priority = NULL
)
```

Arguments

`x`	Object to derive a regex pattern for.
`numbersToKeep`	Set of numbers giving the length for which elements repeated that many times should be counted explicitly (e.g. "[[:digit:]]{5}"). Repeat sequences not included in numbersToKeep will be coded with a "+" (e.g. "[[:digit:]]+"). Defaults to c(2, 3, 4, 5, 10). Set to NULL to have all runs coded as "+" and set to `2:maxCharacters` to have the length specified for all repeated values. If one is included then all unique patterns with be counted as "{1}"; if it is not then the "{1}" is left off.
`combineCases`	Logical indicating whether to treat upper and lower case letters as a single entity. Defaults to FALSE.
`combineAlphanumeric`	Logical indicating whether to treat alphabetic characters and digits as a single entity. Defaults to FALSE.
`combinePunctuation`	Logical indicating whether to treat all punctuation characters as a single entity. Defaults to FALSE.
`combineSpace`	Logical indicating whether to treat all space characters as a single entity. Defaults to FALSE.
`sep`	Value used to separate the regex patterns. Defaults to an empty string.

`escapePunctuation`
: Logical indicating whether to escape any punctuation characters. Defaults to FALSE. Set to TRUE if you want to use the returned value as an argument to grep.

`enclose`
: Logical indicating whether to surround each returned value with `'^'` and `'$'`. Defaults to FALSE.

`priority`
: Character vector allowing users to specify characters that will take precedence over regex grouping patterns. Defaults to NULL, meaning it is ignored. See @details for more information.

Details

The fundamental use of inverseRegex applies only to strings. Inputs of a class other than character are treated as follows:

- Integer: Converted using `as.character()`.
- Factor: Converted using `as.character()`.
- Logical: Left as is. Converting to character would not provide any simplification.
- Numeric: Converted to character by applying `format(..., nsmall = 1)` element by element. NA values will be returned as NA_character_, whilst `NaN`, `Inf`, and `-Inf` will be returned as literal strings: `"NaN"`, `"Inf"`, and `"-Inf"`.
- Date: Converted using `as.character()`.
- POSIXct: Converted using `as.character()`.
- Data frame: Each column is assessed individually and the results combined together so that the output is a data frame of regex patterns with the same dimensions as the input. The columns of class other than character will each be converted as described previously, with one exception: Unlike above where numerics are passed to `format(..., nsmall = 1)` element by element, here the entire column is passed to `trimws(format(...))`. This will lead to a common number of digits to the right of the decimal point and a variable number of digits with no padding on the left side.
- Matrix: Creates a matrix of regex patterns with the same dimensions as the input. If the matrix has a mode of numeric then it will first be passed to `trimws(format(...))`.
- Tibble: Same as a data frame except the returned object is a tibble. Requires the tibble package to be installed.
- Anything else: Not supported; an error will be thrown.

If these conversion methods are not appropriate then you can do the conversion yourself so that the input is dispatched directly to inverseRegex.character.

The regex patterns are identified using the constructs "[:digit:]", "[:upper:]", "[:lower:]", "[:alpha:]", "[:alnum:]", "[:punct:]", and "[:space:]" as described in `?regex`. This will allow for non-ASCII characters to be identified, to the extent supported by `grep`. Any characters not identified by these search patterns will be left as is. Note that for characters from unicameral alphabets the combineCases argument will need to be set to TRUE otherwise they will not be detected by "lower" and "upper".

The priority argument allows users to exclude certain characters from being grouped into the regex patterns described above. For example when using the string "ABC" if using priority = c("A",

"[[:upper:]]", "B") then the result would be "A[[:upper:]]{2}" since the "A" is a higher priority than the "[[:upper:]]" grouping. If you leave the regex groups out then they are assumed to take last priority (meaning in that example using priority = "A" would give the same result). One can also use '.' to make certain characters be collapsed together.

NA values in the input will remain as NA values in the output.

Value

A set of regex patterns that match the input data. These patterns will either be character vectors or the same class as the input object if it was a matrix, data frame, or tibble.

Author(s)

Jasper Watson

See Also

occurrencesLessThan, regex

Examples

```
inverseRegex('Hello World!')

table(inverseRegex(c(rep('HELLO', 10), 'HELL0')))

unique(inverseRegex(iris, numbersToKeep = 1:10))

inverseRegex(c(1, NA, 3.45, NaN, Inf, -Inf))

inverseRegex('abc123?!', priority = c('a', '1', '!', '[[:lower:]]', '[[:digit:]]', '.'))
```

---

occurrencesLessThan *Identifies Infrequent inverseRegex Patterns in an R Object.*

Description

Calls `inverseRegex` on the input object and identifies values that occur infrequently.

Usage

```
occurrencesLessThan(x, fraction = 0.05, n = NULL, ...)
```

Arguments

x	Object to analyse for infrequent regex patterns.
fraction	Fraction of the R object size; regex patterns that occur less (or equal) often than this will be identified. For a vector this fraction will be multiplied by the length of the object; for a matrix it will be multiplied by the total number of entries; and for a data frame or tibble it will be multiplied by the number of rows. Defaults to 0.05.
n	Alternative to the fraction argument which allows a literal number of occurrences to be searched for. Defaults to NULL, in which case fraction will be used.
...	Other arguments to be passed to inverseRegex.

Details

This function is essentially a wrapper around calling table() on the return value of inverseRegex. It can be used to identify the indices of values that consist of a regex pattern different to others in the R object.

Value

A collection of logical values with TRUE indicating entries with an infrequent regex pattern. The class of the return value will depend on the input object; matrices, data frames, and tibbles will be returned in kind; all others are returned as vectors.

Note

NA values are not considered and will need to be identified separately.

Author(s)

Jasper Watson

See Also

inverseRegex, regex

Examples

```
occurrencesLessThan(c(LETTERS, 1))

x <- iris
x$Species <- as.character(x$Species)
x[27, 'Species'] <- 'set0sa'
apply(occurrencesLessThan(x), 2, which)
```

Index

inverseRegex, 2

occurrencesLessThan, 4