Package ‘wfindr’

October 12, 2022

Title  Crossword, Scrabble and Anagram Solver
Version  0.1.0
Date  2016-07-02
Description  Provides a large English words list and tools to find words by patterns. In particular, anagram finder and scrabble word finder.

URL  https://github.com/idmn/wfindr
BugReports  https://github.com/idmn/wfindr/issues
License  GPL-2
LazyData  true
Depends  R (>= 3.1.2)
Imports  dplyr, magrittr
RoxygenNote  5.0.1
NeedsCompilation  no
Author  Iaroslav Domin [aut, cre]
Maintainer  Iaroslav Domin <ya.domin@gmail.com>
Repository  CRAN
Date/Publication  2016-07-03 17:58:53

R topics documented:

char_count .................................................. 2
find_word .................................................. 2
model_to_regex ........................................... 4
scrabble ..................................................... 5
words.eng .................................................. 6
Index .................................................... 7
**char_count**  
*Characters count*

**Description**
Calculates character frequencies in a vector.

**Usage**
```
char_count(x)
```

**Arguments**

- **x** character vector, or a list that can be unlisted to a character vector.

**Value**

data.frame with two columns: `char` - character and `count` - number of its occurrences.

**Examples**
```
char_count("character")
char_count(words.eng)
```

---

**find_word**  
*Find words that fit the chosen parameters.*

**Description**

Uses regex constructed by `model_to_regex` to search words. By default the search is done among `words.eng`.

`find_word` returns a vector of found words, `find_word_l` returns a logical vector that can be used for subsetting.

**Usage**
```
find_word(model = "*", allow = letters, ban = character(0),
          type = "usual", words = wfindr::words.eng)
```
```
find_word_l(model = "*", allow = letters, ban = character(0),
           type = "usual", words = wfindr::words.eng)
```
find_word

Arguments

model  pattern that a word should match. Consists of letters and unknown characters specifications. Dot . stands for unknown character. It may be followed by {...} repetition quantifier (i.e. .\{n\}, .\{n,\}, .\{n,m\}). Asterisk * stands for unknown number of unknown characters. See examples.

By default model is set to "*".

allow  characters allowed to fill gaps in a word. Can be listed in a single string or in a vector. By default is set to letters.

ban  characters not allowed to fill gaps in a word.

type  can be "usual", "scrabble", or "anagram". Abbreviated input is allowed: e.g. "u", "s", or "a".

type defines how often allowed characters can be used to fill the gaps. Say, character appears n times in allow and m times in ban. If d = n - m is less or equal to zero, whatever the type is, this character won’t be used to fill the gaps. For the case when d > 0:

• If type is "usual" then the character is allowed to fill the gaps unlimited number of times.

• If type is "scrabble" then the character is allowed to fill the gaps no more than d times.

• If type is "anagram" then the character should be used exactly d times.

words  vector of words to search within. By default is set to words.eng.

See Also

scrabble, anagram

Examples

## Search 4-letter words starting with "c".
find_word("c.\{3\}")

## Disallow "a" and "b".
find_word("c.\{3\}", ban = "ab")

## Allow only "a" and "b" to fill the gap.
find_word("c.\{3\}", allow = "ab")

## Allow "a", "b", and "c", but then ban "c"
find_word("c.\{3\}", allow = "abc", ban = "c")

## result is the same as in the previous example
find_word("c.\{3\}", allow = "abc", ban = "c")

## Find no more than 4-letter words that have "th" bigram
library(magrittr)
find_word(".\{0,4\}") %>% find_word("*th*x", words = .)

## count words that start with "th"
sum(find_word_l("*th*x", words = .))

## length(find_word("th*x"))

## Find words that can be constructed of the "thing" word's letters.
## Build a regular expression to fit chosen parameters

**Description**

Build a regular expression to fit chosen parameters.

**Usage**

```r
model_to_regex(model = "*", allow = letters, ban = character(0),
               type = "usual")
```

**Arguments**

- **model**
  - pattern that a word should match. Consists of letters and unknown characters specifications. Dot . stands for unknown character. It may be followed by {...} repetition quantifier (i.e. .{n}, .{n,}, .{n,m}). Asterisk * stands for unknown number of unknown characters. See examples.
  - By default model is set to "*".
- **allow**
  - characters allowed to fill gaps in a word. Can be listed in a single string or in a vector. By default is set to letters.
- **ban**
  - characters not allowed to fill gaps in a word.
- **type**
  - can be "usual", "scrabble", or "anagram". Abbreviated input is allowed: e.g. "u", "s", or "a".
  - type defines how often allowed characters can be used to fill the gaps. Say, character appears n times in allow and m times in ban. If d = n - m is less or equal to zero, whatever the type is, this character won’t be used to fill the gaps.
  - For the case when d > 0:
    - If type is "usual" then the character is allowed to fill the gaps unlimited number of times.
    - If type is "scrabble" then the character is allowed to fill the gaps no more than d times.
    - If type is "anagram" then the character should be used exactly d times.

**Warning**

If type = "scrabble" or "anagram", output regex will contain perl-like syntax. So, to use it in grep or gsub for example, set perl parameter to TRUE.
See Also

find_word, scrabble, anagram

Examples

## Regular expression to match all the 5-letter words starting with "c".
model_to_regex("c.{4}")
## Disallow "a" and "b".
model_to_regex("c.(4)", ban = "ab")
## Allow only "a" and "b" to fill the gap.
model_to_regex("c.(4)", allow = "ab")
## Allow "a", "b", and "c", but then ban "c" (result is the same as the previous example)
model_to_regex("c.(4)", allow = "abc", ban = "c")

## Regex to match all words that start with "p" and end with "zed".
model_to_regex("p*zed")

## Regex to match all the words that can be constructed of the word "thing".
model_to_regex(allow = "thing", type = "scrabble")
## Get at least 4-letter words.
model_to_regex(".*(4,)", allow = "thing", type = "scrabble")

## Regex to match anagrams of the word "thing"
model_to_regex(allow = "thing", type = "anagram")

<table>
<thead>
<tr>
<th>scrabble</th>
<th>Find words that can be constructed from the specified letters</th>
</tr>
</thead>
</table>

**Description**

scrabble finds words that can be constructed from the specified set of letters.
anagram finds words that are permutations of the specified set of letters. Usually this set of letters is a word itself.

**Usage**

```
scrabble(allow, model = "+", ban = character(0),
         words = wfindr::words.eng)
```

```
anagram(allow, model = "+", ban = character(0), words = wfindr::words.eng)
```

**Arguments**

- **allow**: characters allowed to use to construct words.
- **model**: pattern that a word should match. Consists of letters and unknown characters specifications. Dot . stands for unknown character. It may be followed by {...} repetition quantifier (i.e. .{n}, .{n,}, .{n,m}). Asterisk * stands for
unknown number of unknown characters. See examples.
By default model is set to "*".

ban characters not allowed to fill gaps in a word.

words vector of words to search within. By default is set to \texttt{words.eng}.

\section*{Details}

\texttt{scrabble} and \texttt{anagram} are functions built on top of the \texttt{find_word} function with parameter \texttt{type} set to "\texttt{scrabble}" or "\texttt{anagram}" respectively and \texttt{allow} parameter moved to the first place to simplify usage (see the first example).

\section*{See Also}
\texttt{find_word}

\section*{Examples}

\begin{verbatim}
## Find all words that can be constructed of the "thing" word's letters
scrabble("thing")
## same as
find_word(allow = "thing", type = "s")
## take at least 4-letter words
scrabble("thing", ".(4,)")
## same as
find_word(".(4,)", "thing", type = "s")

## Pick 8 random letters and find words that can be constructed of them.
library(magrittr)
sample(letters, 8, TRUE) %>% list(letters = ., words = scrabble(.))

## Find anagrams of the word "thing"
anagram("thing")
\end{verbatim}

\section*{words.eng \textit{English words list}}

\section*{Description}

263,533 english words list took from \url{http://norvig.com/ngrams/} (See word.list file).

\section*{Format}

An object of class \texttt{character} of length 263533.
Index

* datasets
  words.eng.6

  anagram, 3, 5
  anagram (scrabble), 5

  char_count, 2

  find_word, 2, 5, 6
  find_word_1 (find_word), 2

  model_to_regex, 2, 4

  scrabble, 3, 5, 5

  words.eng, 2, 3, 6, 6