Package: lettern (via r-universe)

September 5, 2024

Title Makes Nonsense Words Based on English Letter Frequency Data

Version 0.0.1

Description This constructs ``words" based on weighted sampling from letter and ngram frequency data in English, as summarised by Peter Norvig.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports dplyr, magrittr, purrr, rlang, rvest, stringr, tidyr, utils

Depends R (>= 2.10)

LazyData true

URL https://github.com/francisbarton/lettern

BugReports https://github.com/francisbarton/lettern/issues

Repository https://francisbarton.r-universe.dev

RemoteUrl https://github.com/francisbarton/lettern

RemoteRef HEAD

RemoteSha 0061131bfb8e94463d53a8c3ddaa8ddd8d3dbfef

Contents

bigram_frequencies 2
build_a_sentence 2
choose_a_letter 3
letter_frequencies_by_position 3
letter_frequencies_overall 4
sample_letters 5
word_length_frequencies 5
write_a_poem 6

Index **7**

bigram_frequencies *Bigram frequencies*

Description

The count and proportion of all 676 bigrams (pairs of consecutive letters) within Norvig's corpus.

Usage

```
bigram_frequencies
```

Format

A data frame with 676 rows and 4 variables:

`bigram_1` character. The first letter of the bigram.

`bigram_2` character. The second letter of the bigram.

`count` double. Overall count of this bigram within corpus.

`percentage` double. Overall percentage of this bigram within corpus.

Source

https://www.norvig.com/mayzner.html

build_a_sentence *Build a full sentence of n nonsense "words"*

Description

Build a full sentence of n nonsense "words"

Usage

```
build_a_sentence(n, end = ".", ...)
```

Arguments

`n`	The number of words to include in the sentence.
`end`	The string to end the sentence with. Defaults to a full stop.
`...`	A place to pass on parameters such as `cutoff`, which affects the available frequencies of the bigrams used to build words.

Value

A single string of n words with an ending character such as a full stop.

Examples

```
build_a_sentence(6, cutoff = 0.005)
```

`choose_a_letter` *Choose a letter based on frequency (or frequency at position n within a word)*

Description

Choose a letter based on frequency (or frequency at position n within a word)

Usage

```
choose_a_letter(n)
```

Arguments

`n`	If n is a non-zero integer between -7 and 7 (position in word, from 1 (first) to 7 (seventh) or from -1 (last) to -7 (seventh from last)), it returns a single letter based on letter frequencies at that position only.

Value

A lower-case letter (character string of `nchar` 1) between a and z.

Examples

```
choose_a_letter(3)
```

`letter_frequencies_by_position`
Letter counts/frequencies by position

Description

From Norvig: "Now we show the letter frequencies by position within word. That is, the frequencies for just the first letter in each word, just the second letter, and so on. We also show frequencies for positions relative to the end of the word: "-1" means the last letter, "-2" means the second to last, and so on."

Usage

```
letter_frequencies_by_position
```

Format

A data frame with 364 rows and 4 variables:

`position` integer. Letter's position within a word.

`letter` character. The 26 letters of the English alphabet (lower case).

`count` double. Overall count of this letter within corpus, at this position within a word.

`percentage` double. Overall percentage of this letter within corpus, at this position within a word.

Source

https://www.norvig.com/mayzner.html

---

letter_frequencies_overall

Letter counts/frequencies overall

Description

DATASET_DESCRIPTION

Usage

```
letter_frequencies_overall
```

Format

A data frame with 26 rows and 3 variables:

`letter` character. The 26 letters of the English alphabet (lower case).

`count` double. Overall count of this letter within corpus.

`percentage` double. Overall percentage of this letter within corpus.

Source

https://www.norvig.com/mayzner.html

sample_letters *Pull a weighted sample of n letters from the alphabet*

Description

The `replace` parameter is fixed to TRUE, as this is what makes sense given the frequency-dependent nature of this particular sampling approach. This function returns a single letter based on a weighted sampling from all 26 letters, based on their overall frequency in Norvig's corpus.

Usage

```
sample_letters(n)
```

Arguments

`n`	The number of letters to return.

Value

A vector of lower-case letters.

Examples

```
sample_letters(3)
```

word_length_frequencies *Word length frequencies in English, from Peter Norvig's analysis*

Description

From Norvig: "Here is the breakdown of mentions (in millions) by word length (looking like a Poisson distribution).

Usage

```
word_length_frequencies
```

Format

A data frame with 23 rows and 3 variables:

`word_length` double. Word length.

`count_millions` double. Count of words of this length within corpus (in millions).

`percentage` double. Percentage of words of this length within corpus.

Source

https://www.norvig.com/mayzner.html

`write_a_poem` *Writes a poem of pure gibberish*

Description

Writes a poem of pure gibberish

Usage

```
write_a_poem(lines, mean_line_length = 7, cat = TRUE, ...)
```

Arguments

`lines`	The number of lines per stanza. A single integer returns a single stanza of this many lines. A vector of multiple integers, of length n, will return a poem of n stanzas, with lengths as given in the vector.
`mean_line_length`	Line lengths will be generated at random from a normal distribution around this mean, with SD equal to 1 by default.
`cat`	Boolean. Whether to spew the poem straight to stdout via `cat()` (TRUE, default), or return it invisibly (you'll want to pipe it to an object or some other function, presumably).
`...`	A place to pass on parameters such as `cutoff`, which affects the available frequencies of the bigrams used to build words.

Value

A beautiful poem (character strings concatenated with line breaks)

Examples

```
write_a_poem(c(4, 4), cutoff = 0.01)
```

Index

∗ **datasets**
    bigram_frequencies, 2
    letter_frequencies_by_position, 3
    letter_frequencies_overall, 4
    word_length_frequencies, 5

bigram_frequencies, 2
build_a_sentence, 2

choose_a_letter, 3

letter_frequencies_by_position, 3
letter_frequencies_overall, 4

sample_letters, 5

word_length_frequencies, 5
write_a_poem, 6