How to use the NFreq package

CSQ Siew

2017-10-06

Vignette Info

The `NFreq` package was built to help the user calculate the neighborhood density/degree and neighborhood frequency of a given list of words. The definition of 'neighbors' is based on the Levenshtein edit distance of 1. Therefore words that differ from each other by the substitution, deletion, or addition of 1 character (phoneme/letter) are considered to be neighbors.

This could be useful to any researcher who is working with linguistic data and wants to calculate the neighborhood density or frequency of words (or nonwords) based on character similarity (i.e., this would work for phonological transcriptions or orthographic representations), and based on a given corpus of data (which contains a set of phonological/orthographic transcriptions and the corresponding frequencies for each word).

This vignette will demonstrate how to use the functions in this package.

First things first

This package is not hosted on CRAN, so the easiest way to download it is by installing the `devtools` package and downloading the `NFreq` package from my github page. Here's how to do it:

```
install.packages('devtools')
library(devtools)
devtools::install_github("csqsiew/NFreq")
library(NFreq)
```

Viola!

Set up

You will need to upload two sets of data: A list of words that you want to calculate these measures for, and a reference data set (i.e., some linguistic corpus).

Note that it is **very important** that `words` and `data` are set up correctly in order for the functions to work (an error message will be returned otherwise). `words` should be a vector of character type. `data` should be a dataframe that minimally contains two columns, `Phono` (e.g., phonological transcriptions) and `Frequency` (e.g., your favorite log frequency measures). You can double check using the following:

```
# there is some sample data pre-loaded in the package for demonstration
purposes
# take a look
words
```

```
##  [1] "tapsel"     "IntXmIks"   "@l|bY"      "@nsxlEri"   "ikwxnaks"
##  [6] "brYd"       "pOzN"       "s@krxmEntL" "askyUleS|n" "sYnxSUr"
## [11] "mInt"       "dxmcrxlYz"  "wen"        "@fxbIl|ti"  "pErIS"
## [16] "s@pi"       "fYl"        "InumXxbL"   "pUsi"       "tEl|skapIk"
```

```
class(words)        # character
```

```
## [1] "character"
```

```
is.vector(words)    # should be TRUE
```

```
## [1] TRUE
```

```
head(data)
```

```
##      Ortho   Phono Frequency
## 1        a       x    5.3662
## 2 aardvark ardvark    1.0000
## 3    aback    xb@k    1.3010
## 4   abacus  @bxkxs    1.0000
## 5    abaft   xb@ft    1.0000
## 6  abalone @bxloni    1.0000
```

```
is.data.frame(data) # should be TRUE
```

```
## [1] TRUE
```

It should also be noted that `words` could consist of nonwords and `Phono` could be orthographic representations instead (e.g., 'house' instead of /hWs/)--since it is all based on character similarity anyway :)

The get_degree function

The `get_degree` function calculates the degree or neighborhood density of a word based on Levenshtein edit distance of 1. It returns a dataframe with the words and their corresponding degree.

```
words.degree <- NFreq:::get_degree(stimuli = words, database = data)
words.degree
```

```
##       Stimuli Degree
## 1      tapsel      0
## 2    IntXmIks      0
## 3       @l|bY      0
## 4    @nsxlEri      0
## 5    ikwxnaks      0
## 6        brYd     15
```

```
## 7        pOzN      2
## 8  s@krxmEntL      0
## 9  askyUleS|n      0
## 10    sYnxSUr      0
## 11       mInt     12
## 12  dxmcrxlYz      0
## 13        wen     29
## 14  @fxbIl|ti      0
## 15      pErIS      3
## 16       s@pi      4
## 17        fYl     27
## 18   InumXxbL      1
## 19       pUsi      2
## 20 tEl|skapIk      0

# and then you can output the data if you wish
# write.csv(words.degree, file='word.degree.csv')
```

The get_neighbors function

The `get_neighbors` function outputs a list of 1-edit distance neighbors of each word. It returns a dataframe with the words and their neighbors. I doubt this function would be used very often, but it might be useful to examine the internal contents of a word's neighborhood.

```
words.neighbors <- NFreq:::get_neighbors(stimuli = words, database = data)
# words.neighbors - did not output data as it is messy
# and then you can output the data if you wish
# write.csv(words.neighbors, file='word.neighbors.csv')
```

The get_nfreq function

The `get_nfreq` function calculates the neighborhood frequency of a word, which the average frequency of its neighbors. It returns a dataframe with the words and their corresponding neighborhood frequencies. For this function, your `data` *must* contain a `Frequency` column with nummeric values.

```
words.nfreq <- NFreq:::get_nfreq(stimuli = words, database = data)
words.nfreq

##       Stimuli NeighborFreq
## 1      tapsel          NaN
## 2    IntXmIks          NaN
## 3       @l|bY          NaN
## 4    @nsxlEri          NaN
## 5    ikwxnaks          NaN
## 6        brYd     1.912833
## 7        pOzN     1.389100
## 8  s@krxmEntL          NaN
## 9  askyUleS|n          NaN
```

```
## 10    sYnxSUr          NaN
## 11       mInt     1.740258
## 12  dxmcrxlYz          NaN
## 13        wen     2.142910
## 14  @fxbIl|ti          NaN
## 15      pErIS     1.580133
## 16       s@pi     1.776275
## 17        fYl     2.114430
## 18   InumXxbL     1.000000
## 19       pUsi     2.041400
## 20 tEl|skapIk          NaN

# and then you can output the data if you wish
# write.csv(words.nfreq, file='word.nfreq.csv')
```

Note that if a word does not have any neighbors, it will have an undefined `NeighborFreq` value of `NaN`.

Comments, suggestions, bugs?

Email me at cynsiewsq at gmail dot com - I would love to hear from you! :)