Package ‘dummy’

January 5, 2013

Type Package

Title Tools for being a dummy

Version 0.1.0

Date 2012-11-26

Author Tyler Rinker

Maintainer Tyler Rinker <`tyler.rinker@gmail.com`>

Description This is where I test things and don’t worry about it all going bad.

Depends R (>= 2.15)

License GPL-2

Collate ’blah.R’ ’guy.R’ ’dummy-
package.R’ ’unnamed.R’ ’pos.R’’bracketX.R’ ’termco.a.R’ ’gantt.R’ ’gantt_plot.R’’gantt_rep.R’ ’gantt_wrap.R’ ’kullb

R topics documented:

automated_readability_index 2
bracketX 3
cm_r2l 5
common 5
fun1 6
gantt 6
gantt_plot 8
gantt_rep 9
gantt_wrap 10
kullback.leibler 12
metaDAT 13
pos 13
print.POS 15
print.POSby 16
print.termco_d 16
qdap 16
syllable.sum 17
termco.a 18
wfm 21
word.count 23

Index 25

---

automated_readability_index *Readabilitiy Measures*

Description

`automated_readability_index` - Apply Automated Readability Index to transcript(s) by zero or more grouping variable(s).

`coleman_liau` - Apply Coleman Liau Index to transcript(s) by zero or more grouping variable(s).

`SMOG` - Apply SMOG Readability to transcript(s) by zero or more grouping variable(s).

`flesch_kincaid` - Flesch-Kincaid Readability to transcript(s) by zero or more grouping variable(s).

`fry` - Apply Fry Readability to transcript(s) by zero or more grouping variable(s).

`linsear_write` - Apply Linsear Write Readability to transcript(s) by zero or more grouping variable(s).

Usage

```
automated_readability_index(text.var,
  grouping.var = NULL, rm.incomplete = FALSE, ...)

coleman_liau(text.var, grouping.var = NULL,
  rm.incomplete = FALSE, ...)

SMOG(text.var, grouping.var = NULL, output = "valid",
  rm.incomplete = FALSE, ...)

flesch_kincaid(text.var, grouping.var = NULL,
  rm.incomplete = FALSE, ...)

fry(text.var, grouping.var = NULL, labels = "automatic",
  rm.incomplete = FALSE, ...)

linsear_write(text.var, grouping.var = NULL,
  rm.incomplete = FALSE, ...)
```

Arguments

`text.var`	The text variable
`grouping.var`	The grouping variables. Default NULL generates one word list for all text. Also takes a single grouping variable or a list of 1 or more grouping variables.
`rm.incomplete`	logical. If TRUE removes incomplete sentences from the analysis.
`...`	Other arguments passed to `endf`.
`output`	A character vector character string indicating output type. One of "valid" (default and congruent with McLaughlin's intent) or "all".
`labels`	A character vector character string indicating output type. One of "automatic" (default; adds labels automatically) or "click" (interactive).

Value

Generates a dataframe with selected readability statistic by grouping variable(s). The `frey` function returns a graphic representation of the readability.

Note

Many of the indeices (e.g. Automated Readability Index) are derived from word difficulty (letters per word) and sentence difficulty (words per sentence). If you have not run the sentSplit function on your data the results may not be accurate.

References

Coleman, M. & Liau, T. L. (1975). A computer readability formula designed for machine scoring. Journal of Applied Psychology, Vol. 60, pp. 283-284.

Flesch R. (1948). A new readability yardstick. Journal of Applied Psychology. Vol. 32(3), pp. 221-233. doi: 10.1037/h0057532.

Gunning, T. G. (2003). Building Literacy in the Content Areas. Boston: Allyn & Bacon.

McLaughlin, G. H. (1969). SMOG Grading: A New Readability Formula. Journal of Reading, Vol. 12(8), pp. 639-646.

Senter, R. J., & Smith, E. A.. (1967) Automated readability index. Technical Report AMRLTR-66-220, University of Cincinnati, Cincinnati, Ohio.

Examples

```
## Not run:
with(rajSPLIT, automated_readability_index(dialogue, list(person, act)))
with(rajSPLIT, automated_readability_index(dialogue, list(sex, fam.aff)))

with(rajSPLIT, coleman_liau(dialogue, list(person, act)))
with(rajSPLIT, coleman_liau(dialogue, list(sex, fam.aff)))

with(rajSPLIT, SMOG(dialogue, list(person, act)))
with(rajSPLIT, SMOG(dialogue, list(sex, fam.aff)))

with(rajSPLIT, flesch_kincaid(dialogue, list(person, act)))
with(rajSPLIT, flesch_kincaid(dialogue, list(sex, fam.aff)))

(x <- with(rajSPLIT, fry(dialogue, list(sex, fam.aff))))
with(rajSPLIT, fry(dialogue, list(sex, fam.aff), labels = "click"))

with(rajSPLIT, linsear_write(dialogue, list(person, act)))
with(rajSPLIT, linsear_write(dialogue, list(sex, fam.aff)))

## End(Not run)
```

`bracketX` *Bracket Parsing*

Description

`bracketX` - Apply bracket removal to character vectors.

`bracketXtract` - Apply bracket extraction to character vectors.

Usage

```
  bracketX(text.var, bracket = "all", missing = NULL,
    names = FALSE)

  bracketXtract(text.var, bracket = "all", with = FALSE)
```

Arguments

`text.var`	The text variable
`bracket`	The type of bracket (and encased text) to remove. This is one of the strings `"curly"`, `"square"`, `"round"`, `"angle"` and `"all"`. These strings correspond to: {, [, (, < or all four types.
`missing`	Value to assign to empty cells.
`names`	logical. If TRUE the sentences are given as the names of the counts.
`with`	logical. If TRUE returns the brackets and the bracketted text.

Value

bracketX returns a vector of text with brackets removed.

bracketXtract returns a list of vectors of bracketed text.

Author(s)

Martin Morgan and Tyler Rinker <tyler.rinker@gmail.com>.

References

http://stackoverflow.com/questions/8621066/remove-text-inside-brackets-parens-and-or-braces

Examples

```
examp2 <- examp2 <- structure(list(person = structure(c(1L, 2L, 1L, 3L),
    .Label = c("bob", "greg", "sue"), class = "factor"), text =
    c("I love chicken [unintelligible]!",
    "Me too! (laughter) It's so good.[interupting]",
    "Yep it's awesome {reading}.", "Agreed. {is so much fun}")), .Names =
    c("person", "text"), row.names = c(NA, -4L), class = "data.frame")

examp1
bracketX(examp2$text, 'square')
bracketX(examp2$text, 'curly')
bracketX(examp2$text)

examp2
bracketXtract(examp2$text, 'square')
bracketXtract(examp2$text, 'curly')
bracketXtract(examp2$text)
bracketXtract(examp2$text, with = TRUE)

paste2(bracketXtract(examp2$text, 'curly'), " ")
```

`cm_r2l` *Transform Codes to Start-End Durations*

Description

A helper function for cm_range2long that transforms the range coding structure from cm_range.temp (in list format) into a data frame of start and end times in long format.

Usage

```
cm_r2l(range.list, v.name = "variable", list.var = TRUE)
```

Arguments

`range.list`	A complete list object in the form generated by `cm_range.temp`.
`v.name`	sn optional name for the column created for the list.var argument.
`list.var`	logical. If TRUE creates a column for the data frame created by each range.list passed to `cm_r2l`.

Value

Generates a data frame of start and end times for each code.

References

Miles, M. B. & Huberman, A. M. (1994). An expanded sourcebook: Qualitative data analysis. 2nd ed. Thousand Oaks, CA: SAGE Publications.

See Also

`cm2long` `cm_range.temp` `cm_r2l`

`common` *Prints an common.list*

Description

Prints an common.list.

Usage

```
## S3 method for class 'common.list'
print(x, ...)
```

Arguments

`x`	The common.list object
`...`	ignored

`fun1` *Readabiltiy stat*

Description

fun1 - blah 2

fun2 - blah2

fun3 - blah 3

Usage

```
fun1(X, ...)

fun2(X, simplify = FALSE, ...)

fun3(X, noargs = TRUE, ...)
```

Arguments

`X`	Something called X
`...`	those crazy dots
`simplify`	shut yo face
`noargs`	no arguments

`gantt` *Generate Unit Spans*

Description

Generates start and end times of supplied text selections (i.e. text selections are determined by any number of grouping variables).

Usage

```
gantt(text.var, grouping.var, plot = TRUE,
  units = "words", sums = FALSE, plot.colors = NULL,
  box.color = NULL, col.sep = "_")
```

Arguments

`text.var`	The text variable
`grouping.var`	The grouping variables. Also takes a single grouping variable or a list of 1 or more grouping variables.
`plot`	logical. If TRUE plots the start-end times as a gantt plot.
`units`	The unit of measurement to analyze. One of the strings `"character"`, `"syllable"`, `"word"`, or `"sentence"`.

`sums`	logical. If TRUE reports and optionally plots the total units used by grouping variable(s).
`plot.colors`	The colors of the Gannt plot bars. Either a single color or a length equal to the number of grouping variable(s).
`box.color`	A single color of the box around the Gantt plot bars.

Value

Returns a data frame of start and end times by grouping variable(s) or optionally returns a list of two: (1) A data frame of the total units used by grouping variable(s) and (2) a data frame of of start and end times by grouping variable(s). Optionally plots a gantt plot of the returned data.

Note

For repeated measures data output use `gantt_rep`; for a convientent wrapper that takes text and generates plots use `gantt_plot`; and for a flexible gantt plot that words with code matrix functions (cm) use `gantt_wrap`.

Author(s)

DigEmAll (`stackoverflow.com`) and Tyler Rinker <tyler.rinker@gmail.com>.

References

Wallace Clark and Henry Gantt (1922) The Gantt chart, a working tool of management. New York, Ronald Press

See Also

`gantt_rep`, `gantt_wrap`, `gantt_plot`

Examples

```
## Not run:
gantt(DATA$state, DATA$person)
gantt(DATA$state, DATA$person, sums = TRUE)
gantt(DATA$state, list(DATA$sex, DATA$adult))
gantt(mraja1$dialogue, mraja1$person) #hard to see without box color
gantt(mraja1$dialogue, mraja1$sex)
gantt(mraja1$dialogue, mraja1$person, box.col = "black")
gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex), plot.colors = NULL)
gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex), plot.colors = "black")
gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex), plot = FALSE)
gantt(mraja1$dialogue, mraja1$person, units = "characters", box.color = "black")
gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex), units = "characters")
with(mraja1, gantt(dialogue, list(fam.aff, sex, died),
   units = "characters", sums = TRUE))
gantt(mraja1$dialogue, mraja1$person, units = "syllables", box.color = "black", sums = TRUE)
gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex), units = "syllables")

(dat <- gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex), units = "sentences",
     plot.colors = 'black', sums = TRUE, col.sep = "_")$gantt.df)
gantt_wrap(dat, fam.aff_sex, title = "Gantt Plot")

## End(Not run)
```

`gantt_plot` *Gantt Plot*

Description

A convenience that wraps gantt, gantt_rm and gantt_wrap into a single plotting function.

Usage

```
gantt_plot(text.var, grouping.var, rm.var = NULL,
  fill.var = NULL, xlab = "duration (in words)",
  units = "words", col.sep = "_", ...)
```

Arguments

`text.var`	The text variable
`grouping.var`	The grouping variables. Also takes a single grouping variable or a list of 1 or more grouping variables.
`rm.var`	an optional single vector or list of 1 or 2 of repeated measures to facet by
`fill.var`	an optional variable to fill the code stips by.
`units`	The unit of measurement.
`col.sep`	The column separator.
`\ldots`	Other arguments passed to gantt_wrap

Value

Returns a Gantt style visualization.

Note

For non repeated measures data/plotting use `gantt`; for repeated measures data output use `gantt_rep`; and for a flexible gantt plot that words with code matrix functions (cm) use `gantt_wrap`.

References

Wallace Clark and Henry Gantt (1922) The Gantt chart, a working tool of management. New York, Ronald Press

See Also

`gantt` `gantt_rep`, `gantt_wrap`,

Examples

```
## Not run:
with(rajSPLIT, gantt_plot(text.var = dialogue, grouping.var = person, size=4))
with(rajSPLIT, gantt_plot(text.var = dialogue, grouping.var =
    list(fam.aff, sex), rm.var  = act,
    title = "Romeo and Juliet's dialogue"))
with(rajSPLIT, gantt_plot(dialogue, list(fam.aff, sex), act, transform=T))
rajSPLIT2 <- rajSPLIT
```

```
rajSPLIT2$newb <- as.factor(sample(LETTERS[1:2], nrow(rajSPLIT2),
    replace=TRUE))
z <- with(rajSPLIT2, gantt_plot(dialogue, list(fam.aff, sex),
    list(act, newb), size = 4))
z + theme(panel.margin = unit(1, "lines")) + scale_colour_grey()
z + scale_colour_brewer(palette="Dark2")

## End(Not run)
```

gantt_rep *Generate Unit Spans for Repeated Measures*

Description

Produces start and end times for occurances for each repeated measure condition.

Usage

```
gantt_rep(rm.var, text.var, grouping.var,
  units = "words", col.sep = "_")
```

Arguments

`rm.var`	an optional single vector or list of 1 or 2 of repeated measures to facet by
`text.var`	The text variable
`grouping.var`	The grouping variables. Also takes a single grouping variable or a list of 1 or more grouping variables.
`units`	The unit of measurement to analyze. One of the strings `"character"`, `"syllable"`, `"word"`, or `"sentence"`.

Value

Returns a data frame of start and end times by repeated measure and grouping variable(s)

Note

For non repeated measures data/plotting use `gantt`; for a convientent wrapper that takes text and generates plots use `gantt_plot`; and for a flexible gantt plot that words with code matrix functions (cm) use `gantt_wrap`.

References

Wallace Clark and Henry Gantt (1922) The Gantt chart, a working tool of management. New York, Ronald Press

See Also

`gantt`, `gantt_wrap`, `gantt_plot`

Examples

```
## Not run:
(dat3 <- with(rajSPLIT, gantt_rep(act, dialogue, list(fam.aff, sex), units = "words",
   col.sep = "_")))
gantt_wrap(dat3, fam.aff_sex, facet.vars = "act", title = "Repeated MeasuresGantt Plot",
   minor.line.freq = 25, major.line.freq = 100)

## End(Not run)
```

`gantt_wrap` *Gantt Plot*

Description

A ggplot2 wrapper that produces a Gantt plot

Usage

```
gantt_wrap(dataframe, plot.var, facet.vars = NULL,
  fill.var = NULL, title = NULL,
  ylab = as.character(plot.var),
  xlab = "duration.default", rev.factor = TRUE,
  transform = FALSE, ncol = NULL, minor.line.freq = NULL,
  major.line.freq = NULL, sig.dig.line.freq = 1,
  hms.scale = NULL, scale = NULL, space = NULL, size = 3,
  rm.horiz.lines = FALSE, x.ticks = TRUE, y.ticks = TRUE,
  legend.position = NULL, bar.color = NULL,
  border.color = NULL, border.size = 2,
  border.width = 0.1, constrain = TRUE)
```

Arguments

`dataframe`	a data frame with ploting variable(s) and a column of start and end times.
`plot.var`	a factor plotting variable (y axis)
`facet.vars`	an optional single vector or list of 1 or 2 to facet by
`fill.var`	an optional variable to fill the code stips by.
`title`	an optional title for the plot.
`ylab`	an optional y label.
`xlab`	an optional x label.
`rev.factor`	logical. if TRUE reverse the current plotting order so the first element in the plotting variable's levels is plotted on top.
`ncol`	if an integer value is passed to this gantt_wrap uses facet_wrap rather than facet_grid
`transform`	logical. if TRUE the repeated facets will be transformed from stacked to side by side.
`minor.line.freq`	a numeric value for frequency of minor grid lines.

`major.line.freq`	a numeric value for frequency of major grid lines.
`sig.dig.line.freq`	An internal rounding factor for minor and major line freq. Generally, default value of 1 suffices for larger range of x scale may need to be set to -2..
`hms.scale`	logical. If TURE converts scale to h:m:s format. Default NULL attempts to detect if object is a cm_time2long object
`scale`	should scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y")
`space`	if "fixed", the default, all panels have the same size. If "free_y" their height will be proportional to the length of the y scale; if "free_x" their width will be proportional to the length of the x scale; or if "free" both height and width will vary. This setting has no effect unless the appropriate scales also vary.
`size`	the width of the plot bars.
`rm.horiz.lines`	logical. if TRUE the horzontal lines will be removed
`x.ticks`	logical. if TRUE the x tixks will be displayed
`y.ticks`	logical. if TRUE the y tixks will be displayed
`legend.position`	the position of legends. ("left", "right", "bottom", "top", or two-element numeric vector)
`bar.color`	optional color to constrain all bars
`border.color`	the color to plot border around Gantt bars (default is NULL)
`border.size`	an integer value for the size to plot borders around Gantt bars. Controls length (width also controlled if not specified).
`border.width`	controls broder width around Gantt bars. Use a numeric value in addition to border size if plot borders appear disproportional.
`constrain`	logical. If TRUE the Gantt bars touch the edge of the graph.

Value

Returns a Gantt style visualization.

Note

For non repeated measures data/plotting use `gantt`; for repeated measures data output use `gantt_rep`; and for a convientent wrapper that takes text and generates plots use `gantt_plot`.

Author(s)

Andrie de Vries and and Tyler Rinker <tyler.rinker@gmail.com>.

References

Wallace Clark and Henry Gantt (1922) The Gantt chart, a working tool of management. New York, Ronald Press

See Also

`gantt`, `gantt_plot`, `gantt_rep`

Examples

```
## Not run:
(dat <- gantt(mraja1$dialogue, list(mraja1$fam.aff, mraja1$sex),
    units = "sentences", plot.colors = 'black', sums = TRUE,
    col.sep = "_")$gantt.df)
gantt_wrap(dat, fam.aff_sex, title = "Gantt Plot")
dat$codes <- sample(LETTERS[1:3], nrow(dat), TRUE)
gantt_wrap(dat, fam.aff_sex, fill.var = "codes", legend.position = "bottom")

(dat3 <- with(rajSPLIT, gantt_rep(act, dialogue, list(fam.aff, sex),
    units = "words", col.sep = "_")))
x <- gantt_wrap(dat3, fam.aff_sex, facet.vars = "act",
    title = "Repeated MeasuresGantt Plot")
x + scale_color_manual(values=rep("black", length(levels(dat3$fam.aff_sex))))

## End(Not run)
```

`kullback.leibler` *Kullback Leibler Statistic*

Description

A proximatey measure between two probability distributions applied to speech.

Usage

```
kullback.leibler(x, y = NULL, digits = 3)
```

Arguments

`x`	A numeric vector, matrix or data frame.
`y`	A second numeric vector if x is also a vector. Default is NULL.
`digits`	Number of dicimal places to round.

Details

Uses Kullback & Leibler's (1951) formula:

$$D_{KL}(P||Q) = \sum_i ln\left(\frac{P_i}{Q_i}\right)P_i$$

Value

Returns a matrix of the Kullback Leibler measure between each vector of probabiltiies.

References

Kullback, S., & Leibler, R.A. (1951). On Information and sufficiency. Annals of Mathematical Statistics 22 (1): 79-86. doi:10.1214/aoms/1177729694

Examples

```
## Not run:
p.df <- word.freq.df(DATA$state, DATA$person)
p.mat <- wfm(text.var = DATA$state, grouping.var = DATA$person)

kullback.leibler(p.mat)
kullback.leibler(p.df)
kullback.leibler(p.df$greg, p.df$sam)

p.df2 <- word.freq.df(raj$dialogue, raj$person)
kullback.leibler(p.df2)

## End(Not run)
```

`metaDAT` *metaDAT: Tools for Data Management in Meta-Analysis*

Description

This package automates many of the tasks associated with quantitative discourse analysis oftranscripts containing discourse including frequency counts of sentence types, words, sentence, turns of talk, syllable counts and other assorted analyysis tasks. The package provides parsing tools for preparing transcript data. Many functions enable the user to aggregate data by any number of grouping variables providing analysis and seamless integration with other R packages that undertake higher level analysis and visualization of text. This provides the user with a more efficient and targeted analysis.

`pos` *Transcript Apply Parts of Speech Tagging*

Description

`pos` - Apply part of speech tagger to transcript(s).

`pos.by` - Apply part of speech tagger to transcript(s) by zero or more grouping variable(s).

`pos.tags` - Useful for interpreting the parts of speech tags created by pos and pos.by.

Usage

```
  pos(text.var, parallel = FALSE, na.omit = FALSE,
    digits = 2, progress.bar = TRUE, gc.rate = 10)

  pos.by(text.var, grouping.var = NULL, digits = 2, ...)

  pos.tags(type = "pretty")
```

Arguments

`text.var`	The text variable
`parallel`	logical. If TRUE attempts to run the function on multiple cores. Note that this may not mean a spead boost if you have one core or if the data set is smaller as the cluster takes time to create.
`na.omit`	logical. If TRUE missing values (]codeNA) will be omitted.
`digits`	integer indicating the number of decimal places (round) or significant digits (signif) to be used. Negative values are allowed
`progress.bar`	logical. If TRUE attempts to provide a OS appropriate progress bar. If parallel is TRUE this argument is ignored. Nite that setting this argument to TRUE may slow down the function.
`gc.rate`	An integer value. This is a necessary argument because of a problem with the garbage collection in the openNLP function that `pos` wraps. Consider adjusting this argument upward if the error `java.lang.OutOfMemoryError` occurs.
`grouping.var`	The grouping variables. Default NULL generates one word list for all text. Also takes a single grouping variable or a list of 1 or more grouping variables.
`...`	Other argument supplied to `pos`.
`type`	An optional character string giving the output of the pos tags. This must be one of the strings `"pretty"` (a left justified version of the output optimized for viewing but not good for export), `"matrix"` (a matrix version of the output), `"dataframe"\ "df"` (a dataframe varseion of the output), `"all"` (a list of all three of the previous output types).

Value

pos returns a list of 4:

`text`	The original text
`POStagged`	The original words replaced with parts of speech in context.
`POSprop`	Dataframe of the proportion of parts of speech by row.
`POSfreq`	Dataframe of the frequency of parts of speech by row.

pos.by returns a list of 6:

`text`	The original text
`POStagged`	The original words replaced with parts of speech in context.
`POSprop`	Dataframe of the proportion of parts of speech by row.
`POSfreq`	Dataframe of the frequency of parts of speech by row.
`pos.by.prop`	Dataframe of the proportion of parts of speech by grouping variable.
`pos.by.freq`	Dataframe of the frequency of parts of speech by grouping variable.

References

openNLP `http:/opennlp.apache.org`

See Also

`tagPOS`

Examples

```
## Not run:
posdat <- pos(DATA$state)
str(posdat)
names(posdat)
posdat$text       #original text
posdat$POStagged  #words replaced with parts of speech
posdat$POSprop    #proportion of parts of speech by row
posdat$POSfreq    #frequency of parts of speech by row

pos(DATA$state, parallel = TRUE) # not always useful

## End(Not run)

#use pos.tags to interpret part of speech tags used by pos & pos.by
pos.tags()
pos.tags("matrix")
pos.tags("dataframe")
pos.tags("df")
pos.tags("all")

## Not run:
posbydat <- with(DATA, pos.by(state, sex))
names(posbydat)
posbydat
posbydat$pos.by.prop
with(DATA, pos.by(state, list(adult, sex)))

## End(Not run)
```

`print.POS` *Prints a POS object*

Description

Prints a POS object.

Usage

```
 ## S3 method for class 'POS'
print(x, ...)
```

Arguments

`x`	The POS object
`...`	ignored

print.POSby *Prints a POSby object*

Description

Prints a POSby object.

Usage

```
 ## S3 method for class 'POSby'
print(x, ...)
```

Arguments

x	The POSby object
...	ignored

print.termco_d *Prints an termco_d object.*

Description

Prints an termco_d object.

Usage

```
 ## S3 method for class 'termco_d'
print(x, ...)
```

Arguments

x	The termco_d object
...	ignored

qdap *qdap: Quantitative Discourse Analysis Package*

Description

This package automates many of the tasks associated with quantitative discourse analysis oftranscripts containing discourse including frequency counts of sentence types, words, sentence, turns of talk, syllable counts and other assorted analyysis tasks. The package provides parsing tools for preparing transcript data. Many functions enable the user to aggregate data by any number of grouping variables providing analysis and seamless integration with other R packages that undertake higher level analysis and visualization of text. This provides the user with a more efficient and targeted analysis.

syllable.sum *Syllabication*

Description

`syllable.sum` - Count the number of syllables per row of text.

`syllable.count` - Count the number of syllables in a single text string.

`polysyllable.sum` - Count the number of polysyllables per row of text.

`combo_syllable.sum` - Count the number of both syllables and polysyllables per row of text.

Usage

```
syllable.sum(text.var, parallel = FALSE)

syllable.count(text, remove.bracketed = TRUE,
  algorithm.report = FALSE)

polysyllable.sum(text.var, parallel = FALSE)

combo_syllable.sum(text.var, parallel = FALSE)
```

Arguments

`text.var`	The text variable
`parallel`	logical. If TRUE attempts to run the function on multiple cores. Note that this may not mean a spead boost if you have one core or if the data set is smaller as the cluster takes time to create.
`text`	A single character vector of text.
`remove.bracketed`	logical. If TRUE brackets are removed from the analysis.
`algorithm.report`	logical. If TRUE generates a report of words not found in the dictionary (i.e. syllables were calculated with an algorithm).

Value

syllable.sum returns a vector of syllable counts per row.

syllable.count returns a dataframe of syllable counts and algorithm/dictionary uses and, optionally, a report of words not found in the dictionary.

polysyllable.sum returns a vector of polysyllable counts per row.

combo_syllable.sum returns a dataframe of syllable and polysyllable counts per row.

Note

The worker of all the syllable functions is `syllable.count` though it is not intendeded for direct use on a transcript. This function relies on a a dual dictionary lookup (based on the Nettalk Corpus (Sejnowski & Rosenberg, 1987)) and backup algorithm method.

References

Sejnowski, T.J., and Rosenberg, C.R. (1987). "Parallel networks that learn to pronounce English text" in Complex Systems, 1, 145-168.

Examples

```
## Not run:
syllable.count("Robots like Dason lie.")
syllable.count("Robots like Dason lie.", algorithm.report = TRUE)
syllable.sum(DATA$state)
polysyllable.sum(DATA$state)
combo_syllable.sum(DATA$state)

## End(Not run)
```

---

termco.a *Search For and Count Terms*

Description

termco.a - Search a transcript by any number of grouping variables for categories (themes) of grouped root terms. While there are other termco functions in the termco family (i.e. termco.d) termco.a is a wrapper for general use.

termco.d - Search a transcript by any number of grouping variables for root terms.

term.match - Search a transcript for words that exactly match term(s).

termco2mat - Convert a termco dataframe to a matrix for use with visualization functions (e.g. heatmap2 of the gplots package).

Usage

```
termco.a(text.var, grouping.var = NULL, match.list,
  short.term = TRUE, ignore.case = TRUE, elim.old = TRUE,
  output = "percent", digits = 2,
  apostrophe.remove = FALSE, char.keep = NULL,
  digit.remove = NULL, ...)

termco.d(text.var, grouping.var = NULL, match.string,
  short.term = FALSE, ignore.case = TRUE,
  zero.replace = 0, output = "percent", digits = 2,
  apostrophe.remove = FALSE, char.keep = NULL,
  digit.remove = TRUE, ...)

term.match(text.var, terms, return.list = TRUE,
  apostrophe.remove = FALSE)

termco2mat(dataframe, drop.wc = TRUE, short.terms = TRUE,
  rm.zerocol = FALSE, no.quote = TRUE, transform = TRUE,
  trim.terms = TRUE)
```

Arguments

`text.var`	The text variable.
`grouping.var`	The grouping variables. Default NULL generates one word list for all text. Also takes a single grouping variable or a list of 1 or more grouping variables.
`match.list`	a list of named character vectors
`short.term`	logical. If TRUE column names are trimmed versions of the match list, other wise the terms are wrapped with 'term(phrase)'
`ignore.case`	logical. If TRUE case is ignored.
`elim.old`	logical. If TRUE eliminates the columns that are combined together by the named match.list.
`output`	Type of proportion output; either "proportion" (decimal format) or "percent". Default is percent.
`digits`	integer indicating the number of decimal places (round) or significant digits (signif) to be used. Negative values are allowed.
`apostrophe.remove`	logical. If TRUE removes apostrophes from the text before examining.
`char.keep`	A character vector of symbol character (i.e. punctioation) that strip should keep. The default is to strip everything except apostophes.
`digit.remove`	logical. If TRUE strips digits from the text.
`...`	Other argument supplied to strip.
`match.string`	A vector of terms to search for. When using inside of `term.match` the term(s) must be words or partial words but do not have to be when using `termco.d` (i.e. they can be phrases, symbols etc.).
`zero.replace`	Value to replace 0 values with.
`return.list`	logical. If TRUE returns the output for multiple terms as a list by term rather than a vector.
`dataframe`	A termco.a (or termco.d) dataframe or object.
`drop.wc`	logical. If TRUE the word count column will be dropped.
`short.colnames`	logical. If TRUE the "term()" portion of column names will be dropped.
`no.quote`	logical. If TRUE the matrix will be printed without quotes if it's character.
`transform`	logical. If TRUE the matrix will be transformed.

Value

both `termco.a` and `termco.d` return a list, of class "termco.d", of data frames and information regarding word counts:

`raw`	raw word counts by grouping variable
`prop`	proportional word counts by grouping variable; proportional to each individual's word use
`rnp`	a character combination data frame of raw and proportional
`zero_replace`	value to replace zeros with; mostly internal use
`output`	character value for outpur type (either" "proportion" or "percent"; mostly internal use
`digits`	integer value od number of digits to display; mostly internal use

term.match returns a list or vector of possible words that match a term.

termco2mat returns a matrix of term counts.

Note

The match.list/match.string is (optionally) case and character sensitive. Spacing is an important way to grab specific words and requires careful thought. Using "read"will find the words "bread", "read" "reading", and "ready". If you want to search fo just the word "read" you'd supply a vector of c(" read ", " reads", " reading", " reader"). To search for non character arguments (i.e. numbers and symbols) additional arguments from strip must be passed.

See Also

See Also as `termco.d` See Also as `term.match` See Also as `termco2matrix`

Examples

```
## Not run:
#termco.a examples:

# General form for match.list
#
# ml <- list(
#     cat1 = c(),
#     cat2 = c(),
#     catn = c()
# )

ml <- list(
    cat1 = c(" the ", " a ", " an "),
    cat2 = c(" I'" ),
    "good",
    the = c("the", " the ", " the", "the")
)

(dat <- with(raj.act.1,  termco.a(dialogue, person, ml)))
names(dat)
dat$rnp  #useful for presenting in tables
dat$raw  #prop and raw are useful for performing calculations
dat$prop
dat <- with(raj.act.1,  termco.a(dialogue, person, ml,
    short.term = FALSE, elim.old=FALSE))

dat2 <- data.frame(dialogue=c("@bryan is bryan good @br",
    "indeed", "@ brian"), person=qcv(A, B, A))

ml <- list(wrds=c("bryan", "indeed"), bryan=c("bryan", "@ br", "@br"))

with(dat2, termco.a(dialogue, person, match.list=ml, char.keep="@"))

with(dat2, termco.a(dialogue, person, match.list=ml,
    char.keep="@", output="proportion"))

DATA$state[1] <- "12 4 rgfr  r0ffrg0"
termco.a(DATA$state, DATA$person, '0', digit.remove=FALSE)

#Using with term.match and exclude
exclude(term.match(DATA$state, qcv(th), FALSE), "truth")
termco.a(DATA$state, DATA$person, exclude(term.match(DATA$state, qcv(th),
FALSE), "truth"))
```

```
MTCH.LST <- exclude(term.match(DATA$state, qcv(th, i)), qcv(truth, stinks))
termco.a(DATA$state, DATA$person, MTCH.LST)

#termco.d examples:
term.match(DATA$state, qcv(i, the))
termco.d(DATA$state, DATA$person, c(" the", " i'"))
termco.d(DATA$state, DATA$person, c(" the", " i'"), ignore.case=FALSE)
termco.d(DATA$state, DATA$person, c(" the ", " i'"))

# termco2mat example:
MTCH.LST <- exclude(term.match(DATA$state, qcv(a, i)), qcv(is, it, am, shall))
termco_obj <- termco.a(DATA$state, DATA$person, MTCH.LST)
termco2mat(termco_obj)

# as a visual
dat <- termco2mat(termco_obj)
library(gplots)
heatmap.2(dat, trace="none")

## End(Not run)
```

wfm *Word Frequencty Matrix*

Description

`wfm` - Generate a word frequency matrix by grouping variable(s).

`wfdf` - Generate a word frequency data frame by grouping variable.

`wfm.expanded` - Expand a word frequency matrix to have multiple rows for each word.

`wf.combine` - Combines words (rows) of a word frequency data frame (`wfdf`) together.

Usage

```
wfm(text.var = NULL, grouping.var = NULL, wfdf = NULL,
  output = "raw", stopwords = NULL, digits = 2)

wfdf(text.var, grouping.var = NULL, stopwords = NULL,
  margins = FALSE, output = "raw", digits = 2)

wfm.expanded(text.var, grouping.var = NULL, ...)

wf.combine(wf.obj, word.lists, matrix = FALSE)
```

Arguments

`text.var`	The text variable
`grouping.var`	The grouping variables. Default NULL generates one word list for all text. Also takes a single grouping variable or a list of 1 or more grouping variables.
`wfdf`	A word frequency data frame given instead of raw text.var and optional grouping.var. Basically converts a word frequency dataframe (wfdf) to a word frequency matrix (`wfm`). Default is NULL.

`output`	Output type (either `"proportion"`, `"proportion"` or `"percent"`).
`stopwords`	A vector of stop words to remove.
`digits`	An integer indicating the number of decimal places (round) or significant digits (signif) to be used. Negative values are allowed
`margins`	logical. If TRUE provides grouping.var and word variable totals.
`...`	Other arguments supplied to `wfm`.
`wf.obj`	A `wfm` or `wfdf` object.
`word.lists`	A list of character vectors of words to pass to `wf.combine`
`matrix`	logical. If TRUE returns the output as a `wfm` rather than a `wfdf` object

Value

wfm returns a word frequency of the class matrix.

wfdf returns a word frequency of the class data.frame with a words column and optional margin sums.

wfm.expanded returns a matrix similar to a word frequency matrix (`wfm`)but the rows are expanded to represent the maximum usages of the word and cells are dummy coded to indicate that numer of uses.

`wf.combine` returns a word frequency matrix (`wfm`) or dataframe (`wfdf`) with counts for the combined word.lists merged and remaining terms(else).

Examples

```
## Not run:
#word frequency matrix (wfm) example:
with(DATA, wfm(state, list(sex, adult)))
dat <- with(DATA, wfm(state, person))

#word frequency dataframe (wfdf) example:
with(DATA, wfdf(state, list(sex, adult)))
with(DATA, wfdf(state, person))

#wfm.expanded example:
z <- wfm(DATA$state, DATA$person)
wfm.expanded(z)
wfm.expanded(DATA$state, DATA$person)
wfm.expanded(DATA$state, list(DATA$sex, DATA$adult))

#wf.combine example:
#raw no margins (will work)
x <- wfm(DATA$state, DATA$person)

#raw with margin (will work)
y <- wfdf(DATA$state, DATA$person, margins = TRUE)

#porportion (will not work)
z <- wfdf(DATA$state, DATA$person, output = "proportion")

WL1 <- c(y[, 1])
WL2 <- list(c("read", "the", "a"), c("you", "your", "your're"))
WL3 <- list(bob = c("read", "the", "a"), yous = c("you", "your", "your're"))
WL4 <- list(bob = c("read", "the", "a"), yous = c("a", "you", "your", "your're"))
```

```
WL5 <- list(yous = c("you", "your", "your're"))
WL6 <- list(c("you", "your", "your're"))  #no name so will be called words 1
WL7 <- c("you", "your", "your're")

wf.combine(z, WL2) #Won't work not a raw frequency matrix
wf.combine(x, WL2) #Works (raw and no margins)
wf.combine(y, WL2) #Works (raw with margins)
wf.combine(y, c("you", "your", "your're"))
wf.combine(y, WL1)
wf.combine(y, WL3)
wf.combine(y, WL4) #Error b/c there's overlapping words in the word lists
wf.combine(y, WL5)
wf.combine(y, WL6)
wf.combine(y, WL7)

worlis <- c("you", "it", "it's", "no", "not", "we")
y <- wfdf(DATA$state, list(DATA$sex, DATA$adult), margins = TRUE)
z <- wfdf.combine(y, worlis, matrix = TRUE)

chisq.test(z)
chisq.test(wfm(wfdf = y))

## End(Not run)
```

word.count *Word Counts*

Description

word.count - Transcript Apply Word Counts

character.count - Transcript Apply Character Counts

character.table - Computes a table of character counts by grouping variable(s).

Usage

```
word.count(text.var, byrow = TRUE, missing = NA,
  digit.remove = TRUE, names = FALSE)

wc(text.var, byrow = TRUE, missing = NA,
  digit.remove = TRUE, names = FALSE)

character.count(text.var, byrow = TRUE, missing = NA,
  apostrophe.remove = TRUE, digit.remove = TRUE,
  count.space = FALSE)

character.table(text.var, grouping.var)

char.table(text.var, grouping.var)
```

Arguments

`text.var`	The text variable
`byrow`	logical. If TRUE counts by row, if FALSE counts all words.
`missing`	Value to insert for missing values (empty cells).
`digit.remove`	logical. If TRUE removes digits before counting words.
`names`	logical. If TRUE the sentences are given as the names of the counts.
`apostrophe.remove`	= TRUE logical. If TRUE apostrophes will be counted in the character count.
`count.space`	logical. If TRUE spaces are counted as characters.

Value

word.count returns a word count by row or total.

character.count returns a character count by row or total.

character.table returns a dataframe of character counts by grouping variable.

Note

wc is a convienent short hand for word.count.

See Also

`syllable.count`

Examples

```
# WORD COUNT
word.count(DATA$state)
wc(DATA$state)
word.count(DATA$state, names = TRUE)
word.count(DATA$state, byrow=FALSE, names = TRUE)
sum(word.count(DATA$state))

# CHARACTER COUNTS
character.count(DATA$state)
character.count(DATA$state, byrow=FALSE)
sum(character.count(DATA$state))

# CHARACTER TABLE
character.table(DATA$state, DATA$person)
char.table(DATA$state, DATA$person)
character.table(DATA$state, list(DATA$sex, DATA$adult))
colsplit2df(character.table(DATA$state, list(DATA$sex, DATA$adult)))
```

Index

∗Topic)
cm_r2l, 5
∗Topic **100:120,**
cm_r2l, 5
∗Topic **150'),**
cm_r2l, 5
∗Topic **<-**
cm_r2l, 5
∗Topic **=**
cm_r2l, 5
∗Topic **AA**
cm_r2l, 5
∗Topic **Automated**
automated_readability_index, 2
∗Topic **BB**
cm_r2l, 5
∗Topic **CC**
cm_r2l, 5
∗Topic **Coleman**
automated_readability_index, 2
∗Topic **DD**
cm_r2l, 5
∗Topic **Flesch-Kincaid,**
automated_readability_index, 2
∗Topic **Fry,**
automated_readability_index, 2
∗Topic **Gantt**
gantt, 6
gantt_plot, 8
gantt_rep, 9
gantt_wrap, 10
∗Topic **Index,**
automated_readability_index, 2
∗Topic **Kullback-Leibler**
kullback.leibler, 12
∗Topic **Liau,**
automated_readability_index, 2
∗Topic **Linsear**
automated_readability_index, 2
∗Topic **Readability**
automated_readability_index, 2
∗Topic **SMOG,**
automated_readability_index, 2
∗Topic **Write**
automated_readability_index, 2
∗Topic **bracket,**
bracketX, 3
∗Topic **bracket-remove,**
bracketX, 3
∗Topic **character-count**
word.count, 23
∗Topic **coding,**
cm_r2l, 5
∗Topic **curly-braces**
bracketX, 3
∗Topic **foo**
cm_r2l, 5
∗Topic **list(**
cm_r2l, 5
∗Topic **parenthesis,**
bracketX, 3
∗Topic **parts-of-speech**
pos, 13
∗Topic **polysyllable**
syllable.sum, 17
∗Topic **qcv(terms='')**
cm_r2l, 5
∗Topic **qcv(terms='40'),**
cm_r2l, 5
∗Topic **qcv(terms='50:90'),**
cm_r2l, 5
∗Topic **qcv(terms='60:90,**
cm_r2l, 5
∗Topic **readability,**
automated_readability_index, 2
∗Topic **search**
termco.a, 18
∗Topic **span**
cm_r2l, 5
∗Topic **syllabication,**
syllable.sum, 17
∗Topic **syllable,**
syllable.sum, 17
∗Topic **time**
cm_r2l, 5
∗Topic **word-count,**

word.count, 23
∗Topic **word-frequency-matrix**
wfm, 21
∗Topic **word**
termco.a, 18

automated_readability_index, 2

bracketX, 3
bracketXtract *(*bracketX*)*, 3

char.table *(*word.count*)*, 23
character.count *(*word.count*)*, 23
character.table *(*word.count*)*, 23
cm2long, *5*
cm_r2l, *5*, 5
cm_range.temp, *5*
coleman_liau *(*automated_readability_index*)*, 2
combo_syllable.sum *(*syllable.sum*)*, 17
common, 5

flesch_kincaid *(*automated_readability_index*)*, 2
fry *(*automated_readability_index*)*, 2
fun1, 6
fun2 *(*fun1*)*, 6
fun3 *(*fun1*)*, 6

gantt, 6, *8*, *9*, *11*
gantt_plot, *7*, 8, *9*, *11*
gantt_rep, *7*, *8*, 9, *11*
gantt_wrap, *7–9*, 10

kullback.leibler, 12

linsear_write *(*automated_readability_index*)*, 2

metaDAT, 13
metaDAT-package *(*metaDAT*)*, 13

package-metaDAT *(*metaDAT*)*, 13
package-qdap *(*qdap*)*, 16
polysyllable.sum *(*syllable.sum*)*, 17
pos, 13
print.POS, 15
print.POSby, 16
print.termco_d, 16

qdap, 16
qdap-package *(*qdap*)*, 16

SMOG *(*automated_readability_index*)*, 2
syllable.count, *24*
syllable.count *(*syllable.sum*)*, 17
syllable.sum, 17

tagPOS, *14*
term.match, *20*
term.match *(*termco.a*)*, 18
termco.a, 18
termco.d, *20*
termco.d *(*termco.a*)*, 18
termco2mat *(*termco.a*)*, 18
termco2matrix, *20*

wc *(*word.count*)*, 23
wf.combine *(*wfm*)*, 21
wfdf *(*wfm*)*, 21
wfm, 21
word.count, 23