Package 'rcttext'

August 31, 2023

Title Tools for impact analysis in randomized trials with text outcomes

Version 0.0.0.9000

Description A flexible and user-friendly toolkit for performing impact analysis in randomized trials with outcomes generated through human, machine, and/or hybrid scoring of text data. Provides functionality for feature extraction and aggregation, applying supervised and unsupervised machine learning models for semi-automated text scoring, estimating model-assisted treatment impacts with respect to text outcomes under various randomized designs, visually representing found impacts on text outcomes, and additional functionality for performing text analysis using existing frameworks, especially quanteda.

Encoding UTF-8

Imports quanteda,
textreg,
sampling,
tm,
dplyr,
caret,
quanteda.textstats,
plotrix,
randomizr,
stringi

RdMacros Rdpack

RoxygenNote 7.2.3

Depends R (>= 2.10)

LazyData true

Suggests knitr,
rmarkdown,
testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

R topics documented:

`apply_hunspell` *Force spell-correct*

Description

Try to spell correct unrecognized words as best as able using the hunspell package. This will take the first suggestion of hunspell and just replace the text with the suggestion, word by word.

Usage

```
apply_hunspell(
  text,
  additional_words = NULL,
  skip_prefix = NULL,
  threshold = 1,
  to_lower = FALSE,
  verbose = FALSE
)
```

Arguments

`text`	The text to spell check (list of character vectors)
`additional_words`	List of words that should be considered as spelled correctly.
`skip_prefix`	List of prefixes that if a word has we should skip over
`threshold`	Don't try to correct anything that only occurs threshold or less times in corpus.
`to_lower`	If TRUE will keep everything lowercase, regardless of spelling suggestions.

Details

All words of 2 or 1 character are skipped.

Value

Vector of text, spell-corrected we hope.

`clean_features` *Clean and simplify feature set*

Description

Given set of generated features, simplify set of features using carat package.

Usage

```
clean_features(
  meta,
  ignore = NULL,
  remove.lc = TRUE,
  uniqueCut = 1,
  freqCut = 99,
  cor = 0.95,
  verbose = FALSE
)
```

Arguments

`meta`	The set of features
`ignore`	List of column names to ignore when simplifying (e.g., ID column and other columns that should be preserved).
`remove.lc`	TRUE means remove colinear combinations of features.
`uniqueCut`	Param for carat's nearZeroVar
`freqCut`	Param for carat's nearZeroVar
`cor`	Cutoff of how correlated features should be before dropping one.
`verbose`	Print out progress to console.

Value

Updated meta with fewer columns of the preserved features.

clean_text *Generic text cleaning function*

Description

Quick and easy text cleaning. Take given copus and remove punctuation, remove whitespace, convert everything to lowercase. This function is used for pre-processing text within the generate_features function. The main use for clean_text is to check which elements of your text are converted to empty strings. Empty strings may result in clean_features dropping desired columns.

Usage

```
clean_text(x, split_hyphens = TRUE)
```

Arguments

x	Character vector or corpus object.
split_hyphens	A logical indicating whether hypenated words should be treated as two tokens (split at the hypen).

estimate_impacts *Estimate treatment impacts for hybrid-scored text outcomes*

Description

Given text from a randomized trial with a binary treatment, where a subset of the documents have been human-scored, this function computes model-assisted estimates for the average treatment effect with respect to the human-coded outcome.

Usage

```
estimate_impacts(
  y.obs,
  yhat,
  Z,
  wts = NULL,
  design = c("crd", "multi", "cluster", "rcbd"),
  siteID = NULL,
  clusterID = NULL,
  data,
  adjust = NULL
)
```

Arguments

y.obs	A vector of human-coded scores (with NAs for unscored documents).
yhat	A vector of predicted scores estimated via predict_scores.
Z	Indicator for treatment assignment.
wts	Sampling weights for which documents were human scored. Assumed uniform if null.
design	Type of design used for random assignment (complete randomization, multisite randomized, cluster randomized, and blocked and cluster randomized).
siteID	Vector of IDs for site, for multi-site randomized experiments.
clusterID	Vector of IDs for cluster, for cluster-randomized experiments.
data	A data.frame of subject-level identifiers, demographic variables, group membership, and/or other pre-treatment covariates.
adjust	(optional) character vector or named list of variables in the data matrix to adjust for when estimating treatment impacts.

Value

A model object for estimating treatment impact across an array of features.

---

extract_liwc *Functions for processing and appending output from Linguistic Inquiry Word Count (LIWC) software*

Description

Functions for processing and appending output from Linguistic Inquiry Word Count (LIWC) software

Usage

```
extract_liwc(file, meta = NULL, ID.liwc = 1, ID.meta = NULL, clean = TRUE)
```

Arguments

file	character path to LIWC output file
meta	optional data frame to attach LIWC output to
ID.liwc	ID column (either name or column number) of document IDs in the liwc file.
ID.meta	If meta is specified, character vector of the document ID column to use for merging, corresponding with IDs in ID.liwc.
clean	should LIWC output be cleaned prior to appending (e.g., remove linear combinations, repeat variables, etc.)

`extract_taaco` *Import and merge text features generated using TAACO*

Description

This method helps support feature extraction using TAACO. You will have to use the external TACCO program to generate these features; these methods just help move back and forth from R to TACCO.

Usage

```
extract_taaco(
  file,
  meta = NULL,
  ID.meta = NULL,
  drop_para = FALSE,
  drop_sent = TRUE
)
```

Arguments

`file`	Filename where TAACO results are stored
`meta`	Optional `data.frame` with additional document-level variables to include in output.
`ID.meta`	If `meta` is specified, character vector with name of variables used for merging.
`drop_para`	Drop paragraph level measures of cohesion from features (TRUE/FALSE).
`drop_sent`	Drop adjacent overlap between sentences (TRUE/FALSE).

Details

See `prep_external()` for generating text files that would be ready for TAACO analysis. Call this on the corpus to make files that can be read in and processed easily.

Once external processing is complete, `extract_taaco()` reads output and log files produced by the TAACO program and returns a `data.frame` that can be merged with other feature sets.

The "Filename" column in the read file should be the IDs (with a '.txt' suffix that will be dropped). The results can then be merged with 'meta', if passed,

Value

Returns a `data.frame` of text features.

`extract_w2v` *Compute document-level feature vectors from a pre-trained embedding model.*

Description

This function generates a vector embedding for each word in a string using a set of pre-trained word vectors such as GloVe (Pennington et al. 2014) and returns the mean vector projection across all words in a document.

Usage

```
extract_w2v(x, meta = NULL, model = NULL)
```

Arguments

`x`	A corpus object or character vector of text documents.
`meta`	Dataframe corresponding to the corpus x. If passed, and non-NULL, all features will be added to this dataframe.
`model`	User-specified model object pointing to a custom pre-trained embedding model, represented as a matrix or data frame where the first column is the word/token and the following columns are numeric vectors. If NULL, use default "mini_glove" embeddings on 1000 common words (not recommended).

Value

A list of data frames containing the Word2Vec projections of the corpus

References

Mikolov T, Chen K, Corrado G, Dean J (2013). "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781*. Pennington J, Socher R, Manning C (2014). "Glove: Global vectors for word representation." In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

`generate_features` *Generate an array of text features*

Description

Generates a rich feature representation for documents provided as a character vector or quanteda::corpus() object by applying an array of linguistic and syntactic indices, available text analysis dictionaries, and pre-trained embedding models to all documents.

Usage

```
generate_features(
  x,
  meta = NULL,
  lex = TRUE,
  sent = TRUE,
  ld = "all",
  clean_features = TRUE,
  read = c("ARI", "Coleman", "DRP", "ELF", "Flesch", "Flesch.Kincaid",
    "meanWordSyllables"),
  terms = NULL,
  preProc = list(uniqueCut = 1, freqCut = 99, cor = 0.95, remove.lc = TRUE),
  verbose = FALSE,
  ignore = NULL
)
```

Arguments

x	A corpus object or character vector of text documents.
meta	Dataframe corresponding to the corpus x. If passed, and non-NULL, all features will be added to this dataframe.
lex	Logical, indicating whether to compute lexical indices including measures of lexical diversity, readability, and entropy
sent	Logical, indicating whether to compute sentiment analysis features from available dictionaries
ld	character vector defining lexical diversity measures to compute; see quanteda.textstats::textstat_lexdiv
clean_features	TRUE means implement cleaning step where we drop features with no variation and colinear features. (This happens before any term generation features are added.)
read	character vector defining readability measures to compute; see quanteda.textstats::textstat_readability
terms	character vector of terms to evaluate as standalone features based on document-level frequency (case-insensitive). Not cleaned by clean_features.
preProc	Named list of arguments passed to `caret::preProcess()` for applying pre-processing transformations across the set of text features (e.g., removing collinear features).
ignore	List of column names to ignore when simplifying (e.g., ID column and other columns that should be preserved).
...	(optional) additional arguments passed to quanteda::tokens() for text pre-processing.

Value

A data.frame of available text features, one row per document, one column per feature.

References

Pennington J, Socher R, Manning C (2014). "Glove: Global vectors for word representation." In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

mini_glove *Mini glove dataset*

Description

glove embeddings (50 dimensional) for 1000 common words beyond those words listed in several stopword lists provided by quanteda.

Usage

```
mini_glove
```

Format

A matrix with 1000 rows and 50 columns

Details

Original glove embeddings downloaded from Stanford CITE.

1000 by 50 matrix, each row is an embedding. Rownames are words.

plot_ccs *Plot the results from a CCS run*

Description

This function provides a visualization of the set of words and phrases found to differ systematically between treatment and control groups

Usage

```
plot_ccs(out, xlim = NULL, xadj = c(-0.025, 0.025), ...)
```

Arguments

out	a textreg.result() object
xlim	limits for x-axis
xadj	adjustments to the lower and upper limits on the x-axis of the plot
...	additional arguments passed to plot

plot_textfx *Plot the results from an impact analysis with text outcomes*

Description

This function provides a visualization of the set of textual features found to differ systematically between treatment and control groups.

Usage

```
plot_textfx(out, alpha = 0.05, cols = F, group = T, xlim = NULL, ...)
```

Arguments

out	a model object output from estimate_impacts()
alpha	the threshold for determining statistical significance
cols	should effects be colored by direction (red for negative impacts, blue for positive impacts)
group	(optional) should effects be grouped by category (e.g., higher-level summary measures, linguistic features, etc.)
xlim	(optional) bounds for x-axis
...	additional arguments passed to plot

predict_scores *Extract predictions from a fitted text scoring model.*

Description

This function computes the predicted scores for a collection of documents based on the results of a trained ensemble learner.

Usage

```
predict_scores(fit, newdata, na.action = na.omit, ...)
```

Arguments

fit	a model or list of models to use for prediction
newdata	an optional data frame or matrix of predictors
na.action	the method for handling missing data
...	additional arguments to pass to predict.train

Value

A vector of predictions

prep_external *Prepare text documents for analysis using external programs*

Description

Text pre-processing and corpus management functions to provide compatibility with external text analysis programs and standalone software packages such as Linguistic Inquiry Word Count (LIWC), the Tool for Automated Analysis of Cohesion (TAACO) and the Sentiment Analysis and Social Cognition Engine (SEANCE).

Usage

```
prep_external(x, dir, docnames = NULL, preProc = NULL)
```

Arguments

x	A corpus object or character vector of text documents.
dir	Name of directory where the generated intermediate text files should be stored.
docnames	Optional character string specifying file names for each document in x.
preProc	Optional text pre-processing function(s) (e.g., stemming) to apply prior to writing text files for analysis in external programs.

References

Pennebaker JW, Booth RJ, Boyd RL, Francis ME (2015). "Linguistic Inquiry and Word Count: LIWC 2015." www.liwc.net. Crossley SA, Kyle K, McNamara DS (2016). "The tool for the automatic analysis of text cohesion (TAACO): Automatic assessment of local, global, and text cohesion." *Behavior research methods*, **48**(4), 1227–1237. Crossley SA, Kyle K, McNamara DS (2017). "Sentiment Analysis and Social Cognition Engine (SEANCE): An automatic tool for sentiment, social cognition, and social-order analysis." *Behavior research methods*, **49**(3), 803–821.

repair_spelling *Replace all words in dictionary with alternates*

Description

Given text as a list of character strings, and a dictionary as a two-column dataframe with the first column being misspelled words and the second being corect spelling, swap all misspelled words with the correct spellings.

Usage

```
repair_spelling(text, dictionary, to_words = NULL)
```

Arguments

text	Character vector
dictionary	Dataframe with two columns of text, or a list of words.
to_words	If dictionary is a list of words, this is list of corresponding words.
Character	vector, revised version of text.

results.tab *Make results table for grid CCS run*

Description

Make results table for grid CCS run

Usage

```
results.tab(result, corp, Z)
```

Arguments

result	a textreg.result() object
corp	a corpus or character vector to calculate term frequencies across
Z	an indicator for treatment assignment
clusterID	optional vector of cluster ID's
...	additional arguments passed to textreg().

Value

a textreg.result() object.

run_ccs *Perform Concise Comparative Summarization across a grid of tuning parameters*

Description

Wrapper for textreg::textreg().

Determine the penalty C that will zero out the textreg model for a series of randomly permuted labelings with random assignment dictated by a blocked and cluster-randomized experiment.

Usage

```
run_ccs(x, Z, clusterID = NULL)

## S3 method for class 'threshold.C'
cluster(
  x,
  z,
  design = c("crd", "multi", "cluster", "rcbd"),
  clusterID = NULL,
  siteID = NULL,
  R,
  ...
)
```

Arguments

x	a corpus, character vector of text documents, or set of text features.
Z	an indicator for treatment assignment
clusterID	vector of cluster ID's
design	Type of design used for random assignment (complete randomization, multisite randomized, cluster randomized, and blocked and cluster randomized).
siteID	vector of block ID's
R	Number of times to scramble treatment assignment labels
...	additional arguments passed to textreg().

Details

Method repeatedly generates +1/-1 vectors within the given blocking structure with blocks of +1/-1 within the clustering vector, and then finds a threshold C for each permutation.

Value

a textreg.result() object.

List of numbers. First is the threshold C for the passed labeling. Remainder are the reference distribution based on the permutations.

---

textfx *Given text from a randomized trial with a binary treatment, this function computes estimates for the average treatment effect with respect to an array of text-based outcomes*

Description

Given text from a randomized trial with a binary treatment, this function computes estimates for the average treatment effect with respect to an array of text-based outcomes

Usage

```
textfx(
  x,
  Z,
  adj = NULL,
  data,
  wts = NULL,
  design = list(siteID = NULL, clusterID = NULL)
)
```

Arguments

x	A character vector of text documents or a feature matrix returned by tada
Z	Indicator for treatment assignment.
adj	(optional) character vector or named list of variables in the data matrix to adjust for when estimating treatment impacts.

data	A data.frame of subject-level identifiers, demographic variables, group membership, and/or other pre-treatment covariates.
wts	Sampling weights for documents. Assumed uniform if null.
design	For multi-site and cluster randomized experiments, a named list of vectors containing site IDs and/or cluster IDs.
mcp	character string specifying the correction method to be applied to adjust for multiple comparisons. Defaults to no adjustments. See p.adjust for available adjustment methods.

Value

A model object for estimating treatment impact across an array of features.

`textfx_terms` *Estimates the average treatment effect on the frequency and prevalence of a list of specified terms and phrases*

Description

Estimates the average treatment effect on the frequency and prevalence of a list of specified terms and phrases

Usage

```
textfx_terms(x, Z, terms, ...)
```

Arguments

x	A character vector of text documents or a feature matrix returned by tada
Z	Indicator for treatment assignment.
terms	Terms and phrases to evaluate
...	optional parameters passed to quanteda::tokens()

Value

A vector showing the frequency and prevalence of the specified terms within each treatment group and results of the hypothesis test comparing prevalence across groups.

textML *Model-assisted impact analysis through hybrid human/machine text scoring*

Description

A wrapper function for the multiple steps of generating features, training a scoring model on the human-coded data, predicting scores, and comparing human v. machine estimates.

Usage

```
textML(
  x,
  y,
  z = NULL,
  wts = NULL,
  design = c("crd", "multi", "cluster", "rcbd"),
  siteID = NULL,
  clusterID = NULL,
  max.features = NULL,
  ...
)
```

Arguments

x	a corpus or character vector of text documents.
y	a vector of human-coded scores. Set elements to 'NA' for documents not previously scored.
z	optional indicator for treatment assignment. If specified, separate ensembles will be trained for each treatment group;
wts	Sampling weights for which documents were human scored. Assumed uniform if null.
design	Type of design used for random assignment (complete randomization, multisite randomized, cluster randomized, and blocked and cluster randomized).
siteID	Vector of IDs for site, for multi-site randomized experiments.
clusterID	Vector of IDs for cluster, for cluster-randomized experiments.
max.features	maximum number of text features to use for model training. Defaults to 'NULL' (no strict limit)
...	additional arguments passed to train.

Details

This function takes in a corpus of text documents (or a set of computed text features) along with a sample of human-coded outcome values, and trains an ensemble of machine learning models to predict the outcome as a function of the machine measures of text.

Value

a textML model object

`textsamp` *Select a random sample of documents*

Description

Functions to select random samples of documents using different sampling schemes and/or along different design criteria.

Usage

```
textsamp(
  x,
  size = length(x),
  prob = NULL,
  wt.fn = NULL,
  scheme = NULL,
  method = c("srswr", "srswor", "systematic", "poisson")
)

textsamp_strata(x, by = NULL, ...)

textsamp_cluster(x, by = NULL, ...)
```

Arguments

`x`	A corpus object or character vector of text documents.
`size`	a non-negative integer giving the number of documents to sample.
`prob`	a vector of probability weights for each document.
`wt.fn`	a function for generating probability weights; ignored when `prob` is used. See Details.
`scheme`	optional sampling scheme to implement
`method`	the following methods are implemented: simple random sampling without replacement ('srswor'), simple random sampling with replacement ('srswr'), Poisson sampling ('poisson'), systematic sampling ('systematic'); if `method` is missing, the default method is `srswor`.
`by`	a `data.frame` with document-level grouping variable(s) or character vector with names of variables in 'docvars(x)'
`...`	additional arguments passed on to 'textsamp'. Cannot include 'scheme'.

Value

Returns a `data.frame` containing identifiers for the selected documents.

toy_reads *Dataset with 20 essays from READS pilot data*

Description

Used for testing and illustation of tada functions.

Usage

```
toy_reads
```

Format

A data.frame with 5 columns and 20 rows

Details

5 column data.frame, ID is the id of subject, Q1, Q2, more are meta information on scoring, and text contains character string of the text of the essay.

train_ensemble *Train an ensemble learner for semi-supervised text scoring*

Description

This function takes in a corpus of text documents or a set of computed text features, along with a sample of human-coded outcome values and trains an ensemble of machine learning models to predict the outcome as a function of machine measures of text.

Usage

```
train_ensemble(
  x,
  y,
  z = NULL,
  n.tune = 3,
  cvf = 5,
  bounds = NULL,
  ...,
  return.all = TRUE
)
```

Arguments

x	a data.frame or matrix of numeric text features.
y	a vector of human-coded scores for the outcome of interest.
z	optional indicator for treatment assignment. If specified, separate ensembles will be trained for each treatment group;

n.tune	an integer denoting the amount of granularity in the tuning parameter grid. By default, this argument is the number of levels for each tuning parameters that should be generated by train.
cvf	number of folds for cross validation
bounds	a vector (y1, y2) specifying the lower and upper limits for prediction
...	additional arguments passed to trainControl.
return.all	should all component models be returned? If 'FALSE', returns only the fitted ensemble(s).

Value

a fitted model object

Index