Package 'discursive'

June 11, 2023

Title Measuring Discursive Sophistication in Open-Ended Survey Responses

Version 0.1.1

Description A simple approach to measure political sophistication based on open-ended survey responses. Discursive sophistication captures the complexity of individual attitude expression by quantifying its relative size, range, and constraint. For more information on the measurement approach see: Kraft, Patrick W. 2023. ``Women Also Know Stuff: Challenging the Gender Gap in Political Sophistication." American Political Science Review (forthcoming).

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

Imports SnowballC, stm, stringr, tm, utils

NeedsCompilation no

Author Patrick Kraft [aut, cre, cph] (<https://orcid.org/0000-0003-0123-221X>)

Maintainer Patrick Kraft <kraft.pw@gmail.com>

Repository CRAN

Date/Publication 2023-06-11 11:50:05 UTC

R topics documented:

`cces` *Cooperative Congressional Election Study 2018*

Description

A subset of data from the UWM Team Content of the 2018 CCES wave. See Kraft (2023) for details.

Usage

```
cces
```

Format

`cces`**:**

A data frame with 1,000 rows and 15 columns:

age Age (in years)

female Gender (1 = female)

educ_cont Education level (1-6)

pid_cont Party identification (1-7)

educ_pid educ_cont * pid_cont

oe01-oe10 Open-ended responses

Source

https://cces.gov.harvard.edu/

`dict_sample` *Constraint Dictionary*

Description

A sample of terms that signal a higher level of constraint between different considerations (combining conjunctions and exclusive words). See Kraft (2023) for details.

Usage

```
dict_sample
```

Format

cces:
A data character vector with 4 elements:

conjunctions also, and

exclusive but, without

`discursive` *Compute discursive sophistication for a set of open-ended responses*

Description

This function takes a data frame (`data`) containing a set of open-ended responses (`openends`) to compute the three components of discursive sophistication (size, range, and constraint) and combines them in a single scale. See Kraft (2023) for details.

Usage

```
discursive(
  data,
  openends,
  meta,
  args_textProcessor = NULL,
  args_prepDocuments = NULL,
  args_stm = NULL,
  keep_stm = TRUE,
  dictionary,
  remove_duplicates = FALSE,
  type = c("scale", "average", "average_scale", "product"),
  progress = TRUE
)
```

Arguments

`data`	A data frame.
`openends`	A character vector containing variable names of open-ended responses in `data`.
`meta`	A character vector containing topic prevalence covariates included in `data`. See `stm::stm()` for details.
`args_textProcessor`	A named list containing additional arguments passed to `stm::textProcessor()`.
`args_prepDocuments`	A named list containing additional arguments passed to `stm::prepDocuments()`.
`args_stm`	A named list containing additional arguments passed to `stm::stm()`.
`keep_stm`	Logical. If TRUE function returns output of `stm::textProcessor()`, `stm::prepDocuments()`, and `stm::stm()`.

`dictionary` A character vector containing dictionary terms to flag conjunctions and exclusive words. May include regular expressions.

`remove_duplicates` Logical. If TRUE duplicates in `dictionary` are removed.

`type` The method of combining the three components, must be "scale", "average", "average_scale", or "product". The default is "scale", which creates an additive index that is re-scaled to mean 0 and standard deviation 1. Alternatively, "average" creates the same additive index without re-scaling; "average_scale" re-scales each individual component to mean 0 and standard deviation 1 before creating the additive index; "product" creates a multiplicative index.

`progress` Logical. Shows progress bar if TRUE.

Value

A list containing the measure of discursive sophistication and the underlying components in a data frame, as well as the output of `stm::textProcessor()`, `stm::prepDocuments()`, and `stm::stm()`.

Examples

```
discursive(data = cces,
           openends = c(paste0("oe0", 1:9), "oe10"),
           meta = c("age", "educ_cont", "pid_cont", "educ_pid", "female"),
           args_prepDocuments = list(lower.thresh = 10),
           args_stm = list(K = 25, seed = 12345),
           dictionary = dict_sample)
```

`discursive_combine` *Combine three components of discursive sophistication in a single scale*

Description

This function combines the `size`, `range`, and `constraint` of open-ended responses in a single scale. See Kraft (2023) for details.

Usage

```
discursive_combine(
  size,
  range,
  constraint,
  type = c("scale", "average", "average_scale", "product")
)
```

Arguments

size	A named list containing an element labeled size, which itself consists of a numeric vector containing the size component of discursive sophistication. Usually created via discursive_size().
range	A numeric vector containing the range component of discursive sophistication. Usually created via discursive_range().
constraint	A numeric vector containing the constraint component of discursive sophistication. Usually created via discursive_constraint().
type	The method of combining the three components, must be "scale", "average", "average_scale", or "product". The default is "scale", which creates an additive index that is re-scaled to mean 0 and standard deviation 1. Alternatively, "average" creates the same additive index without re-scaling; "average_scale" re-scales each individual component to mean 0 and standard deviation 1 before creating the additive index; "product" creates a multiplicative index.

Value

A numeric vector with the same length as the number of rows in data.

Examples

```
discursive_combine(size = list(size = runif(100)), range = runif(100), constraint = runif(100))
```

---

discursive_constraint *Compute the constraint component of discursive sophistication*

Description

This function takes a data frame (data) containing a set of open-ended responses (openends) and a dictionary to identify terms that signal a higher level of constraint between different considerations (usually conjunctions and exclusive words). It returns a numeric vector of dictionary counts re-scaled to range from 0 to 1. See Kraft (2023) for details.

Usage

```
discursive_constraint(data, openends, dictionary, remove_duplicates = FALSE)
```

Arguments

data	A data frame.
openends	A character vector containing variable names of open-ended responses in data.
dictionary	A character vector containing dictionary terms to flag conjunctions and exclusive words. May include regular expressions.
remove_duplicates	Logical. If TRUE duplicates in dictionary are removed.

Value

A numeric vector with the same length as the number of rows in data.

Examples

```
discursive_constraint(data = cces,
                      openends = c(paste0("oe0", 1:9), "oe10"),
                      dictionary = dict_sample)
```

---

discursive_range *Compute the range component of discursive sophistication*

Description

This function takes a data frame (data) containing a set of open-ended responses (openends) to compute the Shannon entropy in individual response lengths across items. The function returns a numeric vector of topic counts re-scaled to range from 0 to 1. See Kraft (2023) for details.

Usage

```
discursive_range(data, openends)
```

Arguments

data	A data frame.
openends	A character vector containing variable names of open-ended responses in data.

Value

A numeric vector with the same length as the number of rows in data.

Examples

```
discursive_range(data = cces,
                 openends = c(paste0("oe0", 1:9), "oe10"))
```

`discursive_size` *Compute the size component of discursive sophistication*

Description

This function takes a data frame (`data`) containing a set of open-ended responses (`openends`) and additional arguments passed to `stm::textProcessor()` and `stm::prepDocuments()` to estimate a structural topic model via `stm::stm()`. The results of the the structural topic model are used to compute the relative number of topics raised in each open-ended response. The function returns a numeric vector of topic counts re-scaled to range from 0 to 1. See Kraft (2023) for details.

Usage

```
discursive_size(
  data,
  openends,
  meta,
  args_textProcessor = NULL,
  args_prepDocuments = NULL,
  args_stm = NULL,
  keep_stm = TRUE,
  progress = TRUE
)
```

Arguments

`data`	A data frame.
`openends`	A character vector containing variable names of open-ended responses in `data`.
`meta`	A character vector containing topic prevalence covariates included in `data`. See `stm::stm()` for details.
`args_textProcessor`	A named list containing additional arguments passed to `stm::textProcessor()`.
`args_prepDocuments`	A named list containing additional arguments passed to `stm::prepDocuments()`.
`args_stm`	A named list containing additional arguments passed to `stm::stm()`.
`keep_stm`	Logical. If TRUE function returns output of `stm::textProcessor()`, `stm::prepDocuments()`, and `stm::stm()`.
`progress`	Logical. Shows progress bar if TRUE.

Value

A list containing the size component of discursive sophistication as well as the output of `stm::textProcessor()`, `stm::prepDocuments()`, and `stm::stm()`.

Examples

```
discursive_size(data = cces,
                openends = c(paste0("oe0", 1:9), "oe10"),
                meta = c("age", "educ_cont", "pid_cont", "educ_pid", "female"),
                args_prepDocuments = list(lower.thresh = 10),
                args_stm = list(K = 25, seed = 12345))
```

`ntopics` *Compute number of topics based on stm results*

Description

This function takes a structural topic model output estimated via `stm::stm()` as well as the underlying set of documents created via `stm::prepDocuments()` to compute the relative number of topics raised in each open-ended response. The function returns a numeric vector of topic counts re-scaled to range from 0 to 1. See Kraft (2023) for details.

Usage

```
ntopics(x, docs, progress = TRUE)
```

Arguments

`x`	A structural topic model estimated via `stm::stm()`.
`docs`	A set of documents used for the structural topic model; created via `stm::prepDocuments()`.
`progress`	Logical. Shows progress bar if TRUE.

Value

A numeric vector with the same length as the number of documents in `x` and `docs`.

Examples

```
meta <- c("age", "educ_cont", "pid_cont", "educ_pid", "female")
openends <- c(paste0("oe0", 1:9), "oe10")
cces$resp <- apply(cces[, openends], 1, paste, collapse = " ")
cces <- cces[!apply(cces[, meta], 1, anyNA), ]
processed <- stm::textProcessor(cces$resp, metadata = cces[, meta])
out <- stm::prepDocuments(processed$documents, processed$vocab, processed$meta, lower.thresh = 10)
stm_fit <- stm::stm(out$documents, out$vocab, prevalence = as.matrix(out$meta), K=25, seed=12345)
ntopics(stm_fit, out)
```

oe_shannon *Compute Shannon entropy*

Description

Internal function to compute Shannon entropy in relative word counts across a set of elements in a character vecotr. Entropy is re-scaled to range from 0 to 1. Function used in `discursive_range()`.

Usage

```
oe_shannon(x)
```

Arguments

x	Character vector containing open-ended responses.

Value

Numeric vector with the same length as x.

Index