Package ‘stminsights’

June 21, 2024

Type Package

Title A 'Shiny' Application for Inspecting Structural Topic Models

Version 0.4.3

Date 2024-06-21

URL https://github.com/cschwem2er/stminsights

BugReports https://github.com/cschwem2er/stminsights/issues

Description
This app enables interactive validation, interpretation and visualization of structural topic models from the ‘stm’ package by Roberts and others (2014) <doi:10.1111/ajps.12103>. It also includes helper functions for model diagnostics and extracting data from effect estimates.

Imports stm (>= 1.3.7), tidygraph (>= 1.3.1), ggraph (>= 2.2.1), igraph (>= 2.0.3), ggrepel (>= 0.9.5), shiny (>= 1.8.1), shinyBS (>= 0.6.0), shinydashboard (>= 0.7.2), shinyjs (>= 2.1.0), ggplot2 (>= 3.5.1), purrr (>= 1.0.2), stringr (>= 1.5.1), dplyr (>= 1.1.4), tibble (>= 3.2.1), DT (>= 0.33.0), readr (>= 2.1.5), huge (>= 1.3.5), stats, scales

Suggests quanteda (>= 4.0.2), knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation no

Author Carsten Schwemmer [aut, cre] (<https://orcid.org/0000-0001-9084-946X>), Jonne Guyt [ctb]

Maintainer Carsten Schwemmer <c.schwem2er@gmail.com>

Repository CRAN

Date/Publication 2024-06-21 12:20:02 UTC
Contents

get_diag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
get_effects . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
get_network . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
run_stminsights . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

Index 9

get_diag computes stm model diagnostics

Description

get_diag() is a helper function to compute average and median semanticCoherence and exclusivity for a number of stm models. The function does not work for models with content covariates.

Usage

get_diag(models, outobj)

Arguments

models A list of stm models.
outobj The out object containing documents for all stm models.

Value

Returns model diagnostics in a data frame.

Examples

library(stm)
library(dplyr)
library(ggplot2)
library(quanteda)

# prepare data
data <- corpus(gadarian, text_field = 'open.ended.response')
docvars(data)$text <- as.character(data)
data <- tokens(data, remove_punct = TRUE) |> tokens_wordstem() |> tokens_remove(stopwords('english')) |> dfm() |> dfm_trim(min_termfreq = 2)
out <- convert(data, to = 'stm')

# fit models
gadarian_3 <- stm(documents = out$documents,
get_effects

```
vocab = out$vocab,
data = out$meta,
prevalence = ~ treatment + s(pid_rep),
K = 3,
max.em.its = 1, # reduce computation time for example
verbose = FALSE)
gadarian_5 <- stm(documents = out$documents,
vocab = out$vocab,
data = out$meta,
prevalence = ~ treatment + s(pid_rep),
K = 5,
max.em.its = 1, # reduce computation time for example
verbose = FALSE)

# get diagnostics
diag <- get_diag(models = list(
  model_3 = gadarian_3,
  model_5 = gadarian_5),
  outobj = out)
  ## Not run:
  # plot diagnostics
  diag |> ggplot(aes(x = coherence, y = exclusivity, color = statistic)) +
  geom_text(aes(label = name), nudge_x = 5) + geom_point() +
  labs(x = 'Semantic Coherence', y = 'Exclusivity') + theme_light()
  ## End(Not run)
```

get_effects

```
get_effects(estimates, variable, type, ci = 0.95, moderator = NULL, modval = NULL, cov_val1 = NULL, cov_val2 = NULL)
```

Description

get_effects() is a helper function to store effect estimates from stm in a data frame.

Usage

```
get_effects

Arguments

- **estimates**: The object containing estimates calculated with `estimateEffect`.
- **variable**: The variable for which estimates should be extracted.
- **type**: The estimate type. Must be either 'pointestimate', 'continuous', or 'difference'.
- **ci**: The confidence interval for uncertainty estimates. Defaults to 0.95.
- **moderator**: The moderator variable in case you want to include an interaction effect.
- **modval**: The value of the moderator variable for an interaction effect. See examples for combining data for multiple values.
- **cov_val1**: The first value of a covariate for type 'difference'.
- **cov_val2**: The second value of a covariate for type 'difference'. The topic proportion of 'cov_val2' will be subtracted from the proportion of 'cov_val1'.

Value

Returns effect estimates in a tidy data frame.

Examples

```r
library(stm)
library(dplyr)
library(ggplot2)

# store effects
prep <- estimateEffect(1:3 ~ treatment + pid_rep, gadarianFit, gadarian)
effects <- get_effects(estimates = prep,
                       variable = 'treatment',
                       type = 'pointestimate')

# plot effects
effects |> filter(topic == 3) |> 
ggplot(aes(x = value, y = proportion)) +
  geom_errorbar(aes(ymin = lower, ymax = upper), width = 0.1, size = 1) +
  geom_point(size = 3) +
  coord_flip() + theme_light() + labs(x = 'Treatment', y = 'Topic Proportion')

# combine estimates for interaction effects
prep_int <- estimateEffect(1:3 ~ treatment * s(pid_rep),
gadarianFit, gadarian)
effects_int <- get_effects(estimates = prep_int,
                         variable = 'pid_rep',
                         type = 'continuous',
                         moderator = 'treatment',
                         modval = 1) |
bind_rows(
  get_effects(estimates = prep_int,
              variable = 'pid_rep',
              type = 'continuous',
              moderator = 'treatment',
              modval = 1))
```
get_network

```r
variable = 'pid_rep',
type = 'continuous',
m moderator = 'treatment',
modval = 0)
)

# plot interaction effects
effects_int |> filter(topic == 2) |> 
  mutate(moderator = as.factor(moderator)) |> 
  ggplot(aes(x = value, y = proportion, color = moderator,
            group = moderator, fill = moderator)) + 
  geom_line() + 
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) + 
  theme_light() + labs(x = 'PID Rep.', y = 'Topic Proportion',
                      color = 'Treatment', group = 'Treatment', fill = 'Treatment')
```

get_network

---

**extract topic correlation network**

**Description**

`get_network()` is a helper function to extract topic correlation networks as tidygraph objects and add labels and topic proportions.

**Arguments**

- `model`: The stm model for computing the correlation network.
- `method`: The method for determining edges. Can be either 'simple' or 'huge'.
- `cutoff`: The correlation cutoff criterion for `method = 'cutoff'`. Defaults to 0.05.
- `labels`: An optional vector of topic labels. Must include a label for each topic of the model.
- `cutiso`: Remove isolated notes without any edges from the network. Defaults to FALSE.

**Value**

Returns tidygraph network of topic correlations.

**Examples**

```r
library(stm)
library(ggraph)
library(quanteda)

# prepare data
data <- corpus(gadarian, text_field = 'open.ended.response')
```
run_stminsights

```r
docvars(data)$text <- as.character(data)

data <- tokens(data, remove_punct = TRUE) |> tokens_wordstem() |> tokens_remove(stopwords('english')) |> dfm() |> dfm_trim(min_termfreq = 2)

out <- convert(data, to = 'stm')

# fit model
gadarian_10 <- stm(documents = out$documents, vocab = out$vocab, data = out$meta, prevalence = ~ treatment + s(pid_rep), K = 10, max.em.its = 1, # reduce computation time for example verbose = FALSE)

## Not run:
# extract network
stm_corrs <- get_network(model = gadarian_10, method = 'simple', labels = paste('Topic', 1:10), cutoff = 0.001, cutiso = TRUE)

## End(Not run)
```

run_stminsights

run_stminsights

launch the stminsights shiny app
run_stminsights

Description
run_stminsights launches the app to analyze Structural Topic models. It requires a .RData file with stm objects as illustrated in the example below.

Usage
run_stminsights(use_browser = TRUE)

Arguments
use_browser
Choose whether you want to launch the shiny app in your browser. Defaults to TRUE.

Examples
## Not run:
library(stm)
library(quanteda)

# prepare data
data <- corpus(gadarian, text_field = 'open.ended.response')
docvars(data)$text <- as.character(data)
data <- tokens(data, remove_punct = TRUE) |> tokens_wordstem() |> tokens_remove(stopwords('english')) |> dfm() |> dfm_trim(min_termfreq = 2)
out <- convert(data, to = 'stm')

# fit models and effect estimates
gadarian_3 <- stm(documents = out$documents,
                  vocab = out$vocab,
data = out$meta,
prevalence = ~ treatment + s(pid_rep),
K = 3,
max.em.its = 1, # reduce computation time for example
verbose = FALSE)
prep_3 <- estimateEffect(1:3 ~ treatment + s(pid_rep), gadarian_3,
                         meta = out$meta)

gadarian_5 <- stm(documents = out$documents,
                  vocab = out$vocab,
data = out$meta,
prevalence = ~ treatment + s(pid_rep),
K = 5,
max.em.its = 1, # reduce computation time for example
verbose = FALSE)
prep_5 <- estimateEffect(1:5 ~ treatment + s(pid_rep), gadarian_5,
meta = out$meta)

# save objects in .RData file
save.image(paste0(tempdir(), '/stm_gadarian.RData'))

# launch the app
if(interactive()){
  run_stminsights()
}

## End(Not run)
Index

estimateEffect, 4
exclusivity, 2

get_diag, 2
get_effects, 3
get_network, 5

run_stminsights, 6

semanticCoherence, 2
stm, 2