Package 'textmatching'

August 19, 2020

Type Package

Title Functions for Matching with Text-Based Confounding

Version 0.1.0

Description Text matching is an approach to adjustment in causal effect estimation when a document provides a proxy for unobserved confounding. The package includes functions to implement the approach to text matching developed in Roberts, Stewart and Nielsen (2020) <doi:10.1111/ajps.12526>.

License GPL-3

Depends R (>= 3.5.0)

Imports stm, Matrix, matrixStats, cem

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Richard Nielsen [aut],
Brandon Stewart [cre, aut],
Margaret Roberts [aut]

Maintainer Brandon Stewart <bms4@princeton.edu>

Repository CRAN

Date/Publication 2020-08-19 07:50:03 UTC

R topics documented:

cem_match *Match using CEM*

Description

A wrapper around CEM to match on refit topics and projection.

Usage

```
cem_match(
  theta,
  projection,
  treat,
  topic_breaks = c(0, 0.1, 1),
  projection_breaks = 5,
  returnX = FALSE,
  ...
)
```

Arguments

theta	an output of refit containing the refit topic proportions, theta. If given the argument NULL the function will omit matching on topics.
projection	an output of project containing the projection. If given the argument NULL the function will omit matching on the projection.
treat	the treatment variable.
topic_breaks	the cutpoints used for the topics. If this is a vector it defines the vector of cutpoints. If this is a single number, it determines the number of bins. By default it uses the cutpoints 0,.1,1 which corresponds to bins of less than 10 and more than 10% about a topic.
projection_breaks	the cutpoints used for the projection. If this is a vector it defines the vector of cutpoints. If this is a single number, it determines the number of bins. By default it uses 5 inductively learned bins. See **cem** for more details.
returnX	a logical which if true adds a copy of the data to the return object as a top level item in the list called X.
...	additional arguments passed to cem

Details

This function is a convenience wrapper around CEM which returns a **cem** object.

Value

a cem.match object

Examples

```
#See ?sim for a walkthrough
data(sim)
refitted <- refit(sim_topics, sim_documents, content_level="1")
projection <- project(sim_topics, sim_documents)
matched <- cem_match(refitted,projection=projection, sim_meta$treat,
                     projection_breaks=2)
```

`content_levels` *Content Levels*

Description

A helper function which returns the levels of the content covariate in an stm model. Returns `NULL` if the model is not a content covariate model.

Usage

```
content_levels(stm_model)
```

Arguments

`stm_model`	the stm content covariate model

Value

character vector containing levels of the content covariate.

References

Roberts, M., Stewart, B., Nielsen, R. (2020) "Adjusting for Confounding with Text Matching." In American Journal of Political Science

Examples

```
data(sim)
content_levels(sim_topics)
```

project *Create a projection for text matching*

Description

Calculates the linear information in the word counts apart from the topics about the treatment.

Usage

```
project(
  stm_model,
  documents,
  interactions = TRUE,
  type = c("theta", "phi", "refit"),
  verbose = TRUE
)
```

Arguments

stm_model	the stm content covariate model from which to develop the projection
documents	the documents that we want the projection for. Note that these must be aligned with the vocabulary. If they were not the original documents used to fit the model, see alignCorpus.
interactions	a logical which defaults to TRUE. Determines whether or not the topic-aspect interactions are included.
type	determines how the topic-covariate interactions are included (see details below). If interactions=FALSE this has no effect.
verbose	a logical indicating if progress should be printed to the screen

Details

The function returns one loading per document, per level of the factor. Thus in the standard case of two levels (treatment/control) the projection is actually two-dimensional (indicating words that are particularly indicative of treatment and words particularly indicative of control).

When `interactions=FALSE` only the content covariate parameters are used (and not the topic-covariate interactions). This may often be a decent approximation to the full calculation because the topic-covariate interactions are typically very sparse.

When `interactions=TRUE` information from the interaction of the topic and the content covariate is included in the projections. The software offers three ways to do this based on the options set for `type`. In each case the difference is how we reweight the topic-specific components of the interaction.

When `type="theta"` (the option used in the paper), we simply use the theta values estimated under the model. When `type="phi"`, we recompute the token-level topic loadings conditional on the document-topics theta. This allows individual words to have their own topic-specific loadings. When `type="refit"`, we recompute the token-level topic loadings but under each different level

of the content covariate. The option is called "refit" because it is essentially refitting the tokens under each different potential level of the content covariate when calculating the projection to that level.

Value

list

projection	the projection of word count information on the document
diagnostic	the sum of interaction projections for each word type normalized by the total number of words. This is helpful for assessing which elements of the vocabularly are contributing to the topic-specific elements of the projection. For non-topic specific parts, the relative contributions can be read directly off the kappa object in stm.

References

Roberts, M., Stewart, B., Nielsen, R. (2020) "Adjusting for Confounding with Text Matching." In American Journal of Political Science

Examples

```
data(sim)
projection <- project(sim_topics,sim_documents)
```

---

refit *Refit the Documents Under an Alternate Treatment Level*

Description

A function to refit all the document loadings in an STM under an alternative content covariate level.

Usage

```
refit(stm_model, documents, content_level, verbose = TRUE)
```

Arguments

stm_model	the stm content covariate model from which to develop the projection
documents	the documents that we want to refit. This currently only works if these are the original documents used to fit the model. However, this same refitting procedure could be done by using fitNewDocuments in the **stm**.
content_level	a string containing the level under which to refit the documents
verbose	a logical indicating if progress should be printed to the screen

Details

This function cycles through all the documents and refits the theta parameter (the document-topic loadings) as though they all had the same level of the content covariate as specified by `content_level`. For documents that already had that content_level the results should be very close to their originally learned value of theta if the model had converged. However, because **stm** is an iterative algorithm and the global parameters are updated last, they could be different.

Value

list

`content_level`	the level of the content covariate
`theta`	the document-topic matrix refit under the new content level

References

Roberts, M., Stewart, B., Nielsen, R. (2020) "Adjusting for Confounding with Text Matching." In American Journal of Political Science

Examples

```
data(sim)
refitted <- refit(sim_topics, sim_documents, content_level="1")
```

`sim` *Simulated Matching Data*

Description

A 270 document set along with a prefit topic model that is used to demonstrate the matching functionality.

Format

stm formatted object corresponding to simulated documents

`treat` a binary treatment variable

`confound` an unknown binary confounding variable

`simy` a simulated outcome

Details

This is a set of documents and a prefit topic model used to demonstrate functionality of the package. It is loosely based off of the gender citation example in Roberts, Stewart and Nielsen (2020). The data is simulated such that the true treatment effect is 1 for all units. There is separable 'unobserved' confounding provided by the binary variable `confound` variables which are themselves based on real data. The outcome `simy` is purely synthetic.

Note that due to data size limitations on CRAN we only included a subset of the documents and a prefit topic model. Because there are so many fewer documents than were used to fit the original topic model, any model fit with this data would likely look substantially different. The original model was fit on 3201 documents using the treatment as the content covariate with 15 topics. All other settings were at their default.

Because we had to select subsets of the data, we emphasize that the example here isn't reflective of a real problem, its just a way of getting a handle on the objects in the code.

Source

Roberts, M., Stewart, B., Nielsen, R. (2020) "Adjusting for Confounding with Text Matching." In American Journal of Political Science

Examples

```
#We start by assuming that you have run a topic model in stm using
#your treatment variable as a content covariate. This is step 1.

#We have done this already and the following command loads the
#topic model as well as the documents, vocab and meta data objects.
#See the stm package for more details about these model objects.
data(sim)

#Step 2 is to recalculate the topics as though they were observed
#as treated.  We do this using the refit() function.
refitted <- refit(sim_topics, sim_documents, content_level="1")

#to this we needed to specify the value of the treatment (here "1").
#If you have forgotten content_levels() will tell you the levels
#for a given stm model.
content_levels(sim_topics)

#Step 3 is to calculate the projection onto the treatment variable
projection <- project(sim_topics, sim_documents, interactions = FALSE)
#NB: here we have turned off interactions purely for speed during
#CRAN checks.  Consider including them if you believe topic-specific
#word choice is relevant.  See description above.

#Finally Step 4 is to match using CEM or other matching method of your
#choice
matched <- cem_match(refitted,projection=projection, sim_meta$treat,
                     projection_breaks=2)
#note here we use a much weaker match on the projections because the data
#have already been trimmed a lot.
```

```
#Now the matched data can be analyzed using standard tools from cem
cem::att(matched, simy ~ treat, data=sim_meta)
#the estimator overestimates a bit but contains the truth in the CI

#We can compare this to the unadjusted difference in means (overestimates)
summary(lm(simy ~ treat, data=sim_meta))
#and the oracle estimator (based on unobserved covariates)
summary(lm(simy ~ treat + confound1 + confound2 + confound3,data=sim_meta))

#Please, be sure to diagnose your matches!!! The key advantage of matching
#is being able to examine matched pairs.  It is always important to read
#the documents!
```

`textmatching-pkg` *Text Matching*

Description

Note: this package is in the very early stages. Our plan is to implement new functionality in future versions. As a result the API here might change substantially. Because functionality is early we haven't implemented diagnostics but please see the paper for ideas.

Details

This package implements functions designed to help use **stm** to perform adjustment for text-based confounders. The proposed method has four steps (see pg 5 of the Early Access publication).

Step 1: estimate a structural topic model including the treatment as a content covariate. This step can be done using **stm**.

Step 2: extract each document's topics calculated as though treated. This can be done using `refit`.

Step 3: extract each document's projection onto the treated variable. This can be done using `project`.

Step 4: match on results of steps 2 and 3. This can be done using **cem** or other matching package of your choice. We include the `cem_match` wrapper for convenience.

Pre-Fit Models and Data: `sim`. See the Examples in the help file of `sim` for a walkthrough of all functionality.

Please be sure to read your documents! This package currently only offers basic functionality so it is easy to overmatch or undermatch if you aren't carefully examining the matched pairs the algorithm returns.

Author(s)

Author: Margaret E. Roberts, Brandon M. Stewart and Richard Nielsen

Maintainer: Brandon Stewart <bms4@princeton.edu>

References

Roberts, M., Stewart, B., Nielsen, R. (2020) "Adjusting for Confounding with Text Matching." In American Journal of Political Science

Additional papers at: structuraltopicmodel.com

See Also

stm

Index