Package 'textir'

February 11, 2018

Title Inverse Regression for Text Analysis

Version 2.0-5

Author Matt Taddy <mataddy@gmail.com>

Depends R (>= 2.15), distrom, gamlr, Matrix, stats, graphics

Suggests MASS

Description Multinomial (inverse) regression inference for text documents and associated attributes. For details see: Taddy (2013 JASA) Multinomial Inverse Regression for Text Analysis <arXiv:1012.2098> and Taddy (2015, AoAS), Distributed Multinomial Regression, <arXiv:1311.6139>. A minimalist partial least squares routine is also included. Note that the topic modeling capability of earlier 'textir' is now a separate package, 'maptpx'.

Maintainer Matt Taddy <mataddy@gmail.com>

License GPL-3

URL http://taddylab.com

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-11 20:55:45 UTC

R topics documented:

`congress109` *Ideology in Political Speeches*

Description

Phrase counts and ideology scores by speaker for members of the 109th US congress.

Details

This data originally appear in Gentzkow and Shapiro (GS; 2010) and considers text of the 2005 Congressional Record, containing all speeches in that year for members of the United States House and Senate. In particular, GS record the number times each of 529 legislators used terms in a list of 1000 phrases (i.e., each document is a year of transcripts for a single speaker). Associated sentiments are repshare – the two-party vote-share from each speaker's constituency (congressional district for representatives; state for senators) obtained by George W. Bush in the 2004 presidential election – and the speaker's first and second common-score values (from http://voteview.com). Full parsing and sentiment details are in Taddy (2013; Section 2.1).

Value

`congress109Counts`
: A `dgCMatrix` of phrase counts indexed by speaker-rows and phrase-columns.

`congress109Ideology`
: A `data.frame` containing the associated `repshare` and common scores `[cs1,cs2]`, as well as speaker characteristics: `party` ('R'epublican, 'D'emocrat, or 'I'ndependent), `state`, and `chamber` ('H'ouse or 'S'enate).

Author(s)

Matt Taddy, <mataddy@gmail.com>

References

Gentzkow, M. and J. Shapiro (2010), *What drives media slant? Evidence from U.S. daily newspapers*. Econometrica 78, 35-7. The full dataset is at http://dx.doi.org/10.3886/ICPSR26242.

Taddy (2013), *Multinomial Inverse Regression for Text Analysis*. http://arxiv.org/abs/1012.2098

See Also

srproj, pls, dmr, we8there

Examples

```
data(congress109)

## Bivariate sentiment factors (roll-call vote common scores)
covars <- data.frame(gop=congress109Ideology$party=="R",
cscore=congress109Ideology$cs1)
covars$cscore <- covars$cscore -
tapply(covars$cscore,covars$gop,mean)[covars$gop+1]
rownames(covars) <- rownames(congress109Ideology)

## cl=NULL implies a serial run.
## To use a parallel library fork cluster,
## uncomment the relevant lines below.
## Forking is unix only; use PSOCK for windows
cl <- NULL
# cl <- makeCluster(detectCores(), type="FORK")
## small nlambda for a fast example
fitCS <- dmr(cl, covars, congress109Counts, gamma=1, nlambda=10)
# stopCluster(cl)

## plot the fit
par(mfrow=c(1,2))
for(j in c("estate.tax","death.tax")){
plot(fitCS[[j]], col=c("red","green"))
mtext(j,line=2) }
legend("topright",bty="n",fill=c("red","green"),legend=names(covars))


## plot the IR sufficient reduction space
Z <- srproj(fitCS, congress109Counts)
par(mfrow=c(1,1))
plot(Z, pch=21, bg=c(4,3,2)[congress109Ideology$party], main="SR projections")
## two pols
Z[c(68,388),]
```

---

corr *2nd momments of sparse matrices*

Description

Correlation and deviation in sparse matrices.

Usage

```
corr(x, y)
sdev(x)
```

Arguments

x	A dgCMatrix or matrix of counts.
y	A matrix with nrow(y)=nrow(x).

Value

corr returns the ncol(x) by ncol(y) matrix of correlation between x and y, and sdev returns the column standard deviations.

Author(s)

Matt Taddy <taddy@chicagobooth.edu>

See Also

pls, congress109

Examples

```
# some congress examples
data(congress109)
r <- corr(congress109Counts, congress109Ideology$repshare)
## 20 terms for Democrats
sort(r[,1])[1:20]
## 20 terms for Republicans
sort(r[,1], decreasing=TRUE)[1:20]
## 20 high variance terms
colnames(congress109Counts)[
order(-sdev(congress109Counts))[1:20]]
```

---

pls *Partial Least Squares*

Description

A simple partial least squares procedure.

Usage

```
pls(x, y, K=1, scale=TRUE, verb=TRUE)

## S3 method for class 'pls'
predict( object, newdata, type="response", ... )

## S3 method for class 'pls'
summary( object, ... )
```

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

## S3 method for class 'pls'
plot(x, K=NULL, xlab="response", ylab=NULL, ...)
```

Arguments

x	The covariate matrix, in either dgCMatrix or matrix format. For plot and print: a pls output object.
y	The response vector.
K	The number of desired PLS directions. In plotting, this can be a vector of directions to draw, otherwise directions 1:fit$K are plotted.
scale	An indicator for whether to scale x; usually a good idea. If scale=TRUE, model is fit with x scaled to have variance-one columns.
verb	Whether or not to print a small progress script.
object	For predict and summary: a pls output object.
newdata	For predict, an ncol(x)-column matrix of new observations. Can be either a simple matrix or a simple_triplet_matrix.
type	For predict, a choice between output types: predictions scaled to the original response for "response", fitted partial least squares directions for "reduction".
xlab	For plot, the x-axis label.
ylab	For plot, the y-axis label. If null, will be set to 'pls(k) fitted values' for each k.
...	Additional arguments.

Details

pls fits the Partial Least Squares algorithm described in Taddy (2012; Appendix A.1). In particular, we obtain loadings loadings[,k] as the correlation between X and factors factors[,k], where factors[,1] is initialized at scale(as.numeric(y)) and subsequent factors are orthogonal to to the k'th pls direction, an ortho-normal transformation of x%*%loadings[,k].

predict.pls returns predictions from the object$fwdmod forward regression $\alpha + \beta * z$ for projections z = x*loadings -shift derived from new covariates, or if type="reduction" it just returns these projections. summary.pls prints dimension details and a quick summary of the corresponding forward regression. plot.pls draws response versus fitted values for least-squares fit onto the K pls directions.

Value

Output from pls is a list with the following entries

y	The response vector.
x	The unchanged covariate matrix.
directions	The pls directions: x%*%loadings - shift.

loadings	The pls loadings.
shift	Shift applied after projection to center the PLS directions.
fitted	K columns of fitted y values for each number of directions.
fwdmod	The lm object from forward regression lm(as.numeric(y)~directions).

predict.pls outputs either a vector of predicted resonse or an nrow(newcounts) by ncol(object$loadings) matrix of pls directions for each new observation. Summary and plot produce return nothing.

Author(s)

Matt Taddy <taddy@chicagobooth.edu>

References

Taddy (2013), *Multinomial Inverse Regression for Text Analysis*. Journal of the American Statistical Association 108.

Wold, H. (1975), *Soft modeling by latent variables: The nonlinear iterative partial least squares approach*. In Perspectives in Probability and Statistics, Papers in Honour of M.S. Bartlett.

See Also

normalize, sdev, corr, congress109

Examples

```
data(congress109)
x <- t( t(congress109Counts)/rowSums(congress109Counts) )
summary( fit <- pls(x, congress109Ideology$repshare, K=3) )
plot(fit, pch=21, bg=c(4,3,2)[congress109Ideology$party])
predict(fit, newdata=x[c(68,388),])
```

---

srproj *Multinomial Inverse Regression (MNIR)*

Description

Estimation of MNIR sufficient reduction projections. Note that mnlm is just a call to dmr from the distrom package.

Usage

```
srproj(obj, counts, dir=1:K, ...)
mnlm(cl, covars, counts, mu=NULL, bins=NULL, verb=0, ...)
```

Arguments

cl	A parallel library socket cluster. See the same argument in help(dmr) for details.
covars	A dense matrix or sparse Matrix of covariates. This should not include the intercept. See the same argument in help(dmr) for details.
counts	A dense matrix or sparse Matrix of response counts (e.g., token counts in text mining). See the same argument in help(dmr) for details. For srproj, this must have the same number of columns as the response dimensions (vocabulary size) in obj.
mu	Pre-specified fixed effects for each observation in the Poisson regression linear equation. See the same argument in help(dmr) for details.
bins	Number of bins into which we will attempt to collapse each column of covars. bins=NULL does no collapsing. See the same argument in help(dmr) for details.
verb	Whether to print some info. See the same argument in help(dmr) for details.
obj	Either a dmr object, as returned from mnlm, or the dmrcoef object obtained by calling coef on the output of mnlm or dmr. The latter will be faster, since coef.dmr is called inside srproj otherwise.
dir	The attribute (covar) dimensions onto which you want to project. The default is all dimensions: 1:K, where K is the number of columns in the covars argument to mnlm.
...	Additional arguments to gamlr from dmr (or mnlm), and to coef.dmr from srproj. See help(gamlr) and help(dmr) for details.

Details

These functions provide the first two steps of multinomial inverse regression (see MNIR paper).

mnlm fits multinomial logistic regression parameters under gamma lasso penalization on a factorized Poisson likelihood. The mnlm function, which remains in this package for backwards compatability only, is just call to the dmr function of the distrom library (see DMR paper). For simplicity, we recommend using dmr instead of mnlm. For model selection, coefficients, prediction, and plotting see the relevant functions in help(dmr).

srproj calculates the MNIR Sufficient Reduction projection from text counts on to the attribute dimensions of interest (covars in mnlm or dmr). In particular, for counts C, with row sums m, and mnlm/dmr coefficients ϕ_j corresponding to attribute j, $z_j = C'\phi_j/m$ is the SR projection in the direction of j. The MNIR paper explains how $V = [v_1 ... v_K]$, your original covariates/attributes, are independent of text counts C given SR projections $Z = [z_1 ... z_K]$.

The final step of MNIR is 'forward regression' for any element of V onto Z and the remaining elements of V. We do not provide a function for this because you are free to use whatever you want; see the MNIR and DMR papers for linear, logistic, and random forest forward regression examples.

Note that if you were previously using textir not for inverse regression, but rather just as fast code for multinomial logistic regression, you probably want to work directly with the gamlr (binary response) or dmr (multinomial response) packages.

Value

srproj returns a matrix with columns corresponding to directions dir, plus an additional column m holding the row totals of counts. mnlm returns a dmr s3 object. See help(dmr) for details.

Author(s)

Matt Taddy <mataddy@gmail.com>

References

Taddy (2013, JASA), *Multinomial Inverse Regression for Text Analysis* (MNIR).

Taddy (2015, AoAS), *Distributed Multinomial Regression* (DMR).

Taddy (2016, JCGS), *The Gamma Lasso* (GL).

See Also

congress109, we8there, dmr

Examples

```
### Ripley's Cushing Data; see help(Cushings) ###
library(MASS)
data(Cushings)
Cushings[,1:2] <- log(Cushings[,1:2])
train <- Cushings[Cushings$Type!="u",]
newdata <- as.matrix(Cushings[Cushings$Type == "u", 1:2])

## fit, coefficients, predict, and plot

# you could replace 'mnlm' with 'dmr' here.
fit <- mnlm(NULL,
  covars=train[,1:2],
  counts=factor(train$Type))

## dmr applies corrected AICc selection by default
round(coef(fit),1)
round(predict(fit, newdata, type="response"),1)
par(mfrow=c(1,3))
for(j in c("a","b","c")){
  plot(fit[[j]]); mtext(j,line=2) }

## see we8there and congress109 for MNIR and srproj examples
```

tfidf *tf-idf*

Description

term frequency, inverse document frequency

Usage

```
tfidf(x,normalize=TRUE)
```

Arguments

x	A dgCMatrix or matrix of counts.
normalize	Whether to normalize term frequency by document totals.

Value

A matrix of the same type as x, with values replaced by the tf-idf

$$f_{ij} * \log[n/(d_j + 1)],$$

where f_{ij} is x_{ij}/m_i or x_{ij}, depending on normalize, and d_j is the number of documents containing token j.

Author(s)

Matt Taddy <taddy@chicagobooth.edu>

See Also

pls, we8there

Examples

```
data(we8there)
## 20 high-variance tf-idf terms
colnames(we8thereCounts)[
order(-sdev(tfidf(we8thereCounts)))[1:20]]
```

we8there *On-Line Restaurant Reviews*

Description

Counts for 2804 bigrams in 6175 restaurant reviews from the site www.we8there.com.

Details

The short user-submitted reviews are accompanied by a five-star rating on four specific aspects of restaurant quality - food, service, value, and atmosphere - as well as the overall experience. The reviews originally appear in Maua and Cozman (2009), and the parsing details behind these specific counts are in Taddy (MNIR; 2013).

Value

`we8thereCounts` A `dgCMatrix` of phrase counts indexed by review-rows and bigram-columns.

`we8thereRatings`
A `matrix` containing the associated review ratings.

Author(s)

Matt Taddy, <mataddy@gmail.com>

References

Maua, D.D. and Cozman, F.G. (2009), *Representing and classifying user reviews*. In ENIA '09: VIII Enconro Nacional de Inteligencia Artificial, Brazil.

Taddy (2013, JASA), *Multinomial Inverse Regression for Text Analysis*.

Taddy (2013, AoAS), *Distributed Multinomial Regression*.

See Also

dmr, srproj

Examples

```
## some multinomial inverse regression
## we'll regress counts onto 5-star overall rating
data(we8there)

## cl=NULL implies a serial run.
## To use a parallel library fork cluster,
## uncomment the relevant lines below.
## Forking is unix only; use PSOCK for windows
cl <- NULL
# cl <- makeCluster(detectCores(), type="FORK")
## small nlambda for a fast example
```

```
fits <- dmr(cl, we8thereRatings[,'Overall',drop=FALSE],
we8thereCounts, bins=5, gamma=1, nlambda=10)
# stopCluster(cl)

## plot fits for a few individual terms
terms <- c("first date","chicken wing",
"ate here", "good food",
"food fabul","terribl servic")
par(mfrow=c(3,2))
for(j in terms)
{  plot(fits[[j]]); mtext(j,font=2,line=2) }

## extract coefficients
B <- coef(fits)
mean(B[2,]==0) # sparsity in loadings
## some big loadings in IR
B[2,order(B[2,])[1:10]]
B[2,order(-B[2,])[1:10]]

## do MNIR projection onto factors
z <- srproj(B,we8thereCounts)

## fit a fwd model to the factors
summary(fwd <- lm(we8thereRatings$Overall ~ z))

## truncate the fwd predictions to our known range
fwd$fitted[fwd$fitted<1] <- 1
fwd$fitted[fwd$fitted>5] <- 5
## plot the fitted rating by true rating
par(mfrow=c(1,1))
plot(fwd$fitted ~ factor(we8thereRatings$Overall),
varwidth=TRUE, col="lightslategrey")
```

Index