Package 'defm'

September 7, 2023

Type Package

Title Estimation and Simulation of Multi-Binary Response Models

Version 0.1-1

Description Multi-binary response models are a class of models that allow for the estimation of multiple binary outcomes simultaneously. This package provides functions to estimate and simulate these models using the Discrete Exponential-Family Models [DEFM] framework. In it, we implement the models described in Vega Yon, Valente, and Pugh (2023) <doi:10.48550/arXiv.2211.00627>. DEFMs include Exponential-Family Random Graph Models [ERGMs], which characterize graphs using sufficient statistics, which is also the core of DEFMs. Using sufficient statistics, we can describe the data through meaningful motifs, for example, transitions between different states, joint distribution of the outcomes, etc.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

LinkingTo Rcpp

Imports Rcpp, stats

Depends R (>= 2.10), stats4

Suggests texreg

NeedsCompilation yes

Author George Vega Yon [aut, cre] (<https://orcid.org/0000-0002-3171-0844>), Department of Veterans Affairs - Rehabilitation, Research, and Development Service [fnd] (Award/W81XWH-18-PH/TBIRP-LIMBIC under Award No. I01 RX003443)

Maintainer George Vega Yon <g.vegayon@gmail.com>

Repository CRAN

Date/Publication 2023-09-07 08:30:07 UTC

R topics documented:

DEFM *Discrete Exponential Family Model (DEFM)*

Description

Discrete Exponential Family Models (DEFMs) are models from the exponential family that deal with discrete data. Here, we deal with binary arrays which can be used to represent, among other things, networks and multinomial binary Markov processes.

Discrete Exponential Family Models (DEFMs) are models from the exponential family that deal with discrete data. Here, we deal with binary arrays which can be used to represent, among other things, networks and multinomial binary Markov processes.

Usage

```
init_defm(m)

print_stats(m, i = 0L)

nterms_defm(m)

nrow_defm(m)

ncol_defm_y(m)

ncol_defm_x(m)

nobs_defm(m)

morder_defm(m)

new_defm(id, Y, X, order = 1)
```

Arguments

m	An object of class DEFM.
i	An integer scalar indicating which set of statistics to print (see details.)
id	Integer vector of length n. Observation ids, for example, person id.
Y	0/1 matrix of responses of n_y columns and n rows.
X	Numeric matrix of covariates of size n_x by n.
order	Integer. Order of the markov process, by default, 1.

Details

The print_stats function prints the supportset of the ith type of array in the model.

Value

An external pointer of class DEFM.

- nterms_defm returns the number of terms in the model.
- nrow_defm returns the number of rows in the model.
- ncol_defm_y returns the number of output variables in the model.
- ncol_defm_x returns the number of covariates in the model.
- nobs_defm returns the number of observations (events) in the model.
- morder_defm returns the order of the Markov process.

An external pointer of class DEFM.

References

Vega Yon, G. G., Pugh, M. J., & Valente, T. W. (2022). Discrete Exponential-Family Models for Multivariate Binary Outcomes (arXiv:2211.00627). arXiv. https://arxiv.org/abs/2211.00627

See Also

defm_mle() for maximum likelihood estimation and loglike_defm() for the log-likelihood function.

Examples

```
# Loading Valente's SNS data
data(valentesnsList)

mymodel <- new_defm(
  id = valentesnsList$id,
  Y = valentesnsList$Y,
  X = valentesnsList$X,
  order = 1
```

```
)

# Adding the intercept terms and a motif from tobacco to mj
term_defm_logit_intercept(mymodel)
term_defm_transition_formula(mymodel, "{y1, 0y2} > {y1, y2}")

# Initialize the model
init_defm(mymodel)

# Fitting the MLE
```

---

defm-names *Access to the names of a model's datasets*

Description

Retrieve the column names of the dependent variable (y) and independent variable (x) of an object of class DEFM.

Usage

```
get_Y_names(m)

get_X_names(m)
```

Arguments

m	An object of class DEFM.

Value

A character vector.

A character vector with the names of the dependent or independent variables.

Examples

```
#' Using Valente's SNS data
data(valentesnsList)

# Creating the DEFM object
mymodel <- new_defm(
  id = valentesnsList$id,
  X = valentesnsList$X,
  Y = valentesnsList$Y,
  order = 0
)

# Getting the names
get_X_names(mymodel)
get_Y_names(mymodel)
```

defm_terms *Model specification for DEFM*

Description

Model specification for DEFM

Usage

```
term_defm_ones(m, idx = "", vname = "")

term_defm_fe(m, idx = "", k = 1, vname = "")

term_defm_transition(m, mat, idx = "", vname = "")

term_defm_transition_formula(m, formula, idx = "", vname = "")

term_defm_logit_intercept(m, coords = as.integer(c()), idx = "", vname = "")

rule_not_one_to_zero(m, idx)

## S3 method for class 'DEFM'
e1 + e2
```

Arguments

m	An object of class DEFM.
idx	Character scalar. Name of the variable to include in the term.
vname	Character scalar. Name to be assigned for the new term.
k	Numeric scalar. Exponent used in the term.
mat	Integer matrix. The matrix specifies the type of motif to capture (see details.)
formula	Character scalar (see details).
coords	Integer vector with the coordinates to include in the term.
e1, e2	e1 An object of class DEFM (e1) and a character (e2).

Details

In term_defm_transition, users can specify a particular motif to model. Motifs are represented by cells with values equal to 1, for example, the matrix:

```
t0:   1 NA NA
t1:   1  1 NA
```

represents a transition `y0 -> (y1, y2)`. If 0 is a motif of interest, then the matrix should include 0 to mark zero values.

The function `term_defm_transition_formula`, will take the formula and generate the corresponding input for defm::counter_transition(). Formulas can be specified in the following ways:

- Intercept effect: ... No transition, only including the current state.
- Transition effect: ... > ... Includes current and previous states.

The general notation is `[0]y[column id]_[row id]`. A preceeding zero means that the value of the cell is considered to be zero. The column id goes between 0 and the number of columns in the array - 1 (so it is indexed from 0,) and the row id goes from 0 to m_order.

Intercept effects:

Intercept effects only involve a single set of curly brackets. Using the 'greater-than' symbol (i.e., '<') is only for transition effects. When specifying intercept effects, users can skip the `row_id`, e.g., `y0_0` is equivalent to `y0`. If the passed `row id` is different from the Markov order, i.e., `row_id != m_order`, then the function returns with an error.

Examples:

- `"{y0, 0y1}"` is equivalent to set a motif with the first element equal to one and the second to zero.

Transition effects:

Transition effects can be specified using two sets of curly brackets and an greater-than symbol, i.e., `{...} > {...}`. The first set of brackets, which we call LHS, can only hold `row id` that are less than `m_order`.

The term `term_defm_logit_intercept` will add what is equivalent to an intercept in a logistic regression. When `coords` is specified, then the function will add one intercept per outcome. These can be weighted by a covariate.

The function `rule_not_one_to_zero` will avoid the transition one to zero in a Markov process.

The + method is a shortcut for term_formula

Value

Invisible 0.

Examples

```
# Loading Valtente's SNS data
data(valentesnsList)

mymodel <- new_defm(
  id    = valentesnsList$id,
  Y     = valentesnsList$Y,
  X     = valentesnsList$X,
  order = 1
)

# Adding the intercept terms and a motif from tobacco to mj
```

```
term_defm_logit_intercept(mymodel)
term_defm_transition_formula(mymodel, "{y1, 0y2} > {y1, y2}")

# Inspecting the model
mymodel
```

`get_stats` *Get sufficient statistics counts*

Description

This function computes the individual counts of the sufficient statistics included in the model.

Usage

```
get_stats(m)
```

Arguments

`m`	An object of class DEFM.

Value

A matrix with the counts of the sufficient statistics.

Examples

```
data(valentesnsList)

mymodel <- new_defm(
  id = valentesnsList$id,
  Y = valentesnsList$Y,
  X = valentesnsList$X,
  order = 1
)

# Adding the intercept terms and a motif from tobacco to mj
term_defm_logit_intercept(mymodel)
term_defm_transition_formula(mymodel, "{y1, 0y2} > {y1, y2}")

# Initialize the model
init_defm(mymodel)

# Get the counts
head(get_stats(mymodel))
```

loglike_defm *Log-Likelihood of DEFM*

Description

Log-Likelihood of DEFM

Usage

```
loglike_defm(m, par, as_log = TRUE)
```

Arguments

m	An object of class DEFM
par	A vector of parameters of length nterms_defm(m).
as_log	Logical scalar. When TRUE (default) returns the log-likelihood, otherwise it returns the likelihood.

Value

Numeric, the computed likelihood or log-likelihood of the model.

Examples

```
# Loading Valtente's SNS data
data(valentesnsList)

mymodel <- new_defm(
  id    = valentesnsList$id,
  Y     = valentesnsList$Y,
  X     = valentesnsList$X,
  order = 1
)

# Adding the intercept terms and a motif from tobacco to mj
term_defm_logit_intercept(mymodel)
term_defm_transition_formula(mymodel, "{y1, 0y2} > {y1, y2}")

# Computing the log-likelihood
loglike_defm(mymodel, par = c(-1, -1, -1, 2), as_log = TRUE)
```

logodds *Maximum Likelihood Estimation of DEFM*

Description

Fits a Discrete Exponential-Family Model using Maximum Likelihood.

Usage

```
logodds(m, par, i, j)

defm_mle(object, start, lower, upper, ...)

summary_table(object, as_texreg = FALSE, ...)
```

Arguments

`m`	An object of class DEFM.
`par`	The parameters of the model.
`i, j`	The row and column of the array to turn on for the log odds.
`object`	An object of class DEFM.
`start`	Double vector. Starting point for the MLE.
`lower, upper`	Lower and upper limits for the optimization (passed to stats4::mle.)
`...`	Further arguments passed to stats4::mle.
`as_texreg`	When `TRUE`, wraps the result in a texreg object

Details

The likelihood function of the DEFM is closely-related to the Exponential-Family Random Graph Model [ERGM]. Furthermore, the DEFM can be treated as a generalization of the ERGM. The model implemented here can be viewed as an ERGM for a bipartite network, where the actors are individuals and the events are the binary outputs.

If the model features no markov terms, i.e., terms that depend on more than one output, then the model is equivalent to a logistic regression. The example below shows this equivalence.

The function `summary_table` computes pvalues and returns a table with the estimates, se, and pvalues. If `as_texreg = TRUE`, then it will return a texreg object.

Value

- `logodds` returns a numeric vector with the log-odds for each observation in the data.

An object of class stats4::mle.

References

Vega Yon, G. G., Pugh, M. J., & Valente, T. W. (2022). Discrete Exponential-Family Models for Multivariate Binary Outcomes (arXiv:2211.00627). arXiv. https://arxiv.org/abs/2211.00627

See Also

DEFM for objects of class DEFM and `loglike_defm()` for the log-likelihood function of DEFMs.

Examples

```
#' Using Valente's SNS data
data(valentesnsList)

# Creating the DEFM object
logit_0 <- new_defm(
  id = valentesnsList$id,
  X = valentesnsList$X,
  Y = valentesnsList$Y[,1,drop=FALSE],
  order = 0
)

# Building the model
term_defm_logit_intercept(logit_0)
term_defm_logit_intercept(logit_0, idx = "Hispanic")
term_defm_logit_intercept(
 logit_0, idx = "exposure_smoke",
 vname = "Smoke Exp"
)
term_defm_logit_intercept(logit_0, idx = "Grades")
init_defm(logit_0) # Needs to be initialized

# Fitting the model
res_0 <- defm_mle(logit_0)

# Refitting the model using GLM
res_glm <- with(
  valentesnsList,
  glm(Y[,1] ~ X[,1] + X[,3] + X[,7], family = binomial())
  )

# Comparing results
summary_table(res_0)
summary(res_glm)

# Comparing the logodds
head(logodds(logit_0, par = coef(res_0), i = 0, j = 0))
```

`motif_census` *Motif census*

Description

Calculates the total motif counts for a given model, in terms of the number of times each motif appears in the data.

Usage

```
motif_census(m, locs)
```

Arguments

m	An object of class DEFM.
locs	Idx (starting from zero) with the variables that will be included in the census.

Value

A matrix of class defm_motif_census with the motif counts.

References

Vega Yon, G. G., Pugh, M. J., & Valente, T. W. (2022). Discrete Exponential-Family Models for Multivariate Binary Outcomes (arXiv:2211.00627). arXiv. https://arxiv.org/abs/2211.00627

Examples

```
# Loading Valente's SNS data
data(valentesnsList)

mymodel <- new_defm(
  id = valentesnsList$id,
  Y = valentesnsList$Y,
  X = valentesnsList$X,
  order = 1
)

# Adding the intercept terms and a motif from tobacco to mj
term_defm_logit_intercept(mymodel)
term_defm_transition_formula(mymodel, "{y1, 0y2} > {y1, y2}")

# Initialize the model
init_defm(mymodel)

# Motif counts featuring only the first two variables
motif_census(mymodel, locs = 0:1)
```

---

sim_defm *Simulate data using a DEFM*

Description

Simulate data using a DEFM

Usage

```
sim_defm(m, par, fill_t0 = TRUE)
```

Arguments

m	An object of class DEFM. The baseline model.
par	Numeric vector of model parameters.
fill_t0	Logical scalar. When TRUE (default) will fill-in the baseline value of each observation (i.e., the starting condition) (see details.)

Details

Each observation in the simulation must have initial condition. In practice, this means we start the markov process with a matrix of size `morder_defm(m) x ncol_defm_y(m)`, i.e., order of the Markov process times the number of output variables. when `fill_t0 = TRUE`, the function return the rows corresponding to baseline states with the original value, otherwise it replaces them with -1. This option is mostly for testing purposes.

Value

An integer vector of size `nrows_defm(m) x ncol_defm_y(m)`.

---

valentesns *Valente's SNS data*

Description

This dataset contains the data used in Valente et al. (2013) to study the influence of peers on adolescent smoking, drinking, and marijuana use. The `valentesnsList` is a transformed version of the data ready to be used to create defm objects.

Usage

```
valentesns
```

Format

The `valentesns` dataset has 1,722 records for 568 individuals, featuring the following 18 columns:

- `id`: Id of the individual.
- `year`: Wave number.
- `Hispanic`: Indicator variable equal to 1 if the individual is Hispanic.
- `Female`: Indicator variable equal to 1 if the individual is female.
- `Grades`: Academic grades ranging from 1 (mostly F) to 5 (mostly As).
- `tobacco`: Indicator variable if the individual ever smoked tobacco.
- `alcohol`: Indicator variable if the individual ever drink alcohol.
- `mj`: : Indicator variable if the individual ever smoked marijuana.
- `sibsmoke` : Indicator variable if the individual's sibling smokes.

- `sibdrink`: Indicator variable if the individual's sibling drinks alcohol.
- `adultdrink`: Indicator variable equal to one if there's an adult who drinks in the household.
- `year_value`: Year of the survey.
- `present`: Indicator variable equal to 1 if the individual was present.
- `school`: School id.
- `has_sib`: Indicator variable equal to 1 if the individual has siblings.
- `exposure_smoke`: Proportion of friends who have smoked tobacco in the past.
- `exposure_drink`: Proportion of friends who have drink alcohol in the past.
- `exposure_mj`: Proportion of friends who have smoked marijuana in the past.

Exposure variables are marked with -1 for each individuals' first wave.

Source

Valente, T. W., Fujimoto, K., Unger, J. B., Soto, D. W., & Meeker, D. (2013). Variations in network boundary and type: A study of adolescent peer influences. Social Networks, 35(3), 309–316. doi:10.1016/j.socnet.2013.02.008.

Index