Package 'rpm'

July 23, 2025

Type Package

Title Modeling of Revealed Preferences Matchings

Version 0.7-3

Date 2024-04-17

Description Statistical estimation of revealed preference models from data collected on bipartite matchings. The models are for matchings within a bipartite population where individuals have utility for people based on known and unknown characteristics. People can form a partnership or remain unpartnered. The model represents both the availability of potential partners of different types and preferences of individuals for such people. The software estimates preference parameters based on sample survey data on partnerships and population composition. The simulation of matchings and goodness-of-fit are considered. See Goyal, Handcock, Jackson, Rendall and Yeung (2022) <doi:10.1093/jrsssa/qnad031>.

License GPL-3 + file LICENSE

License_is_FOSS yes

License_restricts_use no

URL https://github.com/handcock/rpm

BugReports https://github.com/handcock/rpm/issues

LinkingTo Rcpp, RcppArmadillo

Depends R (>= 4.0.0), abind, future, doRNG, methods

Suggests testthat

Imports Rcpp, nloptr, matrixStats, MASS, dplyr, ggplot2, coda, doFuture, foreach

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation yes

Author Mark S. Handcock [aut, cre] (ORCID: <https://orcid.org/0000-0002-9985-2785>), Ryan M. Admiraal [ctb], Fiona C. Yeung [ctb], Heide M. Jackson [ctb],

Michael S. Rendall [ctb],
Shuchi Goyal [ctb]

Maintainer Mark S. Handcock <handcock@stat.ucla.edu>

Repository CRAN

Date/Publication 2024-04-18 08:20:03 UTC

Contents

`rpm-package` *Modeling of Revealed Preferences Matchings*

Description

An integrated set of tools to fit a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

Details

For a complete list of the functions, use `library(help="rpm")` or read the rest of the manual.

When publishing results obtained using this package the original authors are to be cited as:

Mark S. Handcock, Ryan M. Admiraal, Fiona C. Yeung, Heide M. Jackson, Michael S. Rendall and Shuchi Goyal (2022) **rpm**: Modeling of Revealed Preferences Matchings R package, Los Angeles, CA. Version 0.70, https://github.com/handcock/rpm.

All programs derived from this package must cite it. For complete citation information, use `citation(package="rpm")`.

For details on how to construct data for input to `rpm()` see the documentation:

`help(fauxmatching)`

For information on the current terms that can be used in formulas for `rpm()` see the documentation:

`help("rpm-terms")`

Value

No return value, called for side effects.

Author(s)

Mark S. Handcock <handcock@stat.ucla.edu>

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets*, *International Economic Review*, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems*, Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

Useful links:

- https://github.com/handcock/rpm
- Report bugs at https://github.com/handcock/rpm/issues

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
```

```
          sampled="sampled",sampling_design="stock-flow")
summary(fit)

# For details on how to construct data for input:
help(fauxmatching)
# For information on the current terms that can be used in formulas:
help("rpm-terms")
```

`anova.rpm` *ANOVA for rpm Fits*

Description

Compute an analysis of variance table for one or more rpm fits.

Usage

```
## S3 method for class 'rpm'
anova(object, ...)

anova_rpmlist(object, ...)
```

Arguments

`object, ...`	objects of class `rpm`, usually, a result of a call to `rpm`.

Details

Specifying a single object gives a sequential analysis of variance table for that fit. That is, the reductions in the residual sum of squares as each term of the formula is added in turn are given in the rows of a table, plus the residual sum of squares.

The table will contain F statistics (and P values) comparing the mean square for the row to the residual mean square.

If more than one object is specified, the table has a row for the residual degrees of freedom and sum of squares for each model. For all but the first model, the change in degrees of freedom and sum of squares is also given. (This only make statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

Optionally the table can include test statistics. Normally the F statistic is most appropriate, which compares the mean square for a row to the residual sum of squares for the largest model considered. If `scale` is specified chi-squared tests can be used. Mallows' C_p statistic is the residual sum of squares plus twice the estimate of σ^2 times the residual degrees of freedom.

If any of the objects do not have estimated log-likelihoods, produces an error, unless `eval.loglik=TRUE`.

Value

An object of class `"anova"` inheriting from class `"data.frame"`.

Warning

The comparison between two or more models will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values.

See Also

The model fitting function `rpm`, `anova`, `logLik.rpm` for adding the log-likelihood to an existing `rpm` object.

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled",sampling_design="stock-flow")
anova(fit)
```

---

control.rpm *Auxiliary for Controlling rpm*

Description

Auxiliary function as user interface for fine-tuning RPM model fitting algorithm, which computes the MLPLE of the Revealed Preferences Model via optimization.

Usage

```
control.rpm(
  init_theta = NULL,
  algorithm = "NLOPT_LD_SLSQP",
  print_level = 0,
  xtol_rel = 1e-08,
  ftol_rel = 1e-08,
  ftol_abs = 1e-06,
  lower.bound = -10,
  upper.bound = 10,
  maxeval = 2000,
  bs.maxeval = 2000,
  bs.xtol_rel = 1e-08,
  bs.save.data = FALSE,
  check_derivatives = FALSE,
  bootstrap = TRUE,
  hessian = FALSE,
```

```
  seed = NULL,
  parallel.type = "PSOCK",
  parallel.ncores = 1,
  ncores = 1,
  constraints = c("none", "M_single"),
  logodds_single = FALSE,
  save.data = TRUE,
  robust.cov = FALSE,
 local_opts = list(algorithm = "NLOPT_LD_SLSQP", xtol_rel = 1e-07, maxeval = maxeval),
  nbootstrap = 50,
  nbootstrap.SD = 20,
  large.population.bootstrap = 5000,
  alpha = 0.05
)
```

Arguments

`init_theta`	vector; numeric vector of starting parameter values. This value and other possible starting values are applied to find a good optimizer. This can either have length the number of parameters corresponding to the terms in the formula or in addition the equilibrium constraints.
`algorithm`	string; The optimization algorithm to use. See `nloptr::nloptr.print.options()` and the `NLopt` website for a description of the algorithms.
`print_level`	integer; possible values: 0, 1, 2, or 3. This controls how much output is shown during the optimization process. Possible values: 0 (default): no output; 1: show iteration number and value of objective function; 2: 1 + show value of equalities/constraints; 3: 2 + show value of controls.
`xtol_rel`	scalar; Stop when an optimization step (or an estimate of the optimum) changes every parameter by less than xtol_rel multiplied by the absolute value of the parameter. If there is any chance that an optimal parameter is close to zero, you might want to set an absolute tolerance with xtol_abs as well. Criterion is disabled if xtol_rel is non-positive. Possible values: xtol_rel > 0. Default value: 1.0e-08.
`ftol_rel`	scalar; Stop when an optimization step (or an estimate of the optimum) changes the log-likelihood by less than ftol_rel multiplied by the absolute value of the log-likelihood.
`ftol_abs`	scalar; Stop when an optimization step (or an estimate of the optimum) changes the log-likelihood by less than ftol_abs. tolerance with xtol_abs as well. Criterion is disabled if ftol_abs is non-positive. Possible values: ftol_abs > 0. Default value: 1.0e-06.
`lower.bound`	numeric; lower bounds on the parameter estimates (that is, the beta and gamma parameters in the model). Can be a vector of the same size as the coefficient vector or a single number which is used for all bounds.
`upper.bound`	numeric; upper bounds on the parameter estimates (that is, the beta and gamma parameters in the model). Can be a vector of the same size as the coefficient vector or a single number which is used for all bounds.

`maxeval` integer; Stop when the number of function evaluations exceeds maxeval. This is not a strict maximum: the number of function evaluations may exceed maxeval slightly, depending upon the algorithm. Criterion is disabled if maxeval is non-positive. Default value: 1000.

`bs.maxeval` integer; Stop the bootstrap optimization when the number of function evaluations exceeds bs.maxeval. This is not a strict maximum: the number of function evaluations may exceed bs.maxeval slightly, depending upon the algorithm. Criterion is disabled if bs.maxeval is non-positive. Default value:50

`bs.xtol_rel` scalar; Stop the bootstrap optimization when an optimization step (or an estimate of the optimum) changes every parameter by less than bs.xtol_rel multiplied by the absolute value of the parameter. See the parameter xtol_rel for details.

`bs.save.data` logical; Should the bootstrapped data be saved in the bootstrap return list (as components `Xdata` and `Zdata`).

`check_derivatives`
logical; Compare the user-supplied analytic gradients with the finite difference approximations.

`bootstrap` logical; If 'TRUE' use a bootstrap to compute the standard errors and associated covariance matrices. If 'FALSE' base the standard errors and associated covariance matrices on the Hessian of the (constrained) log-likelihood. In all cases the extended covariance matrix is returned in `ext.covar.hessian`. This is the matrix of parameters, log-odds of being single and the Lagrange multipliers.

`hessian` logical; Depreciated. The negation of the 'bootstrap' argument.

`seed` Seed value (integer) for the random number generator. See `set.seed`

`parallel.type` The type of cluster to run. The typical choices are "MPI" and "PSOCK", where you must have "MPI" installed to use the former. The default is "PSOCK".

`parallel.ncores`
count; Depreciated. The renamed 'ncores' argument.

`ncores` Number of processors to use in the bootstrap computations. The default is 1, that is no parallel processing.

`constraints` string; Additional constraints to force the proportions of singles to match the (weighted) population estimates? This should not be required, but does stabilize the estimates in cases where there is much uncertainty. The possible values are "none" and "M_single" (the numbers of male singles of each type are reproduced). Note that adding constraints leads to over-constrained optimization which may fail.

`logodds_single` logical; Should the log-odds ratio of being single relative to a randomly chosen person of the same sex from the the population be returned. If FALSE the log-odds of being single relative is returned. This is a pure preference parameter.

`save.data` logical; Should the data be saved in the return list (as components `Xdata` and `Zdata`).

`robust.cov` logical; Should the covariance matrix of the estimates be computed using a robust method (MASS::cov.mcd)? Only use if the bootstrap is unstable.

`local_opts` list; list of options for `nloptr` sub-algorithm. See the `nloptr` package, but these are rarely changed.

`nbootstrap` integer; Number of bootstrap resamples to take in the estimation of the covariance matrix of the parameter estimates.

`nbootstrap.SD` integer; Number of bootstrap resamples to take in the estimation of the variances used in the studentized bootstrap. This is run for each nbootstrap sample and so is expensive.

`large.population.bootstrap` integer; If the population size exceeds `large.population.bootstrap` then the large population approximation is used to simulate the matchings in the bootstrap. Otherwise the small population simulation is used (including the Gale-Shapley algorithm). The small population method is more accurate in smaller populations, with the default cutoff being 5000 people.

`alpha` proportion; Type I error rate for the confidence intervals produced by the bootstrap.

Details

This function is only used within a call to the `rpm` function.

Some of the arguments are not yet fully implemented. It will evolve slower to incorporate more arguments as the package develops.

Value

A list with arguments as components.

See Also

`rpm`

`fauxmatching` *Faux Data on Heterosexual Matching*

Description

This data set represents a simulation of a bipartite matching. The data set is named fauxmatching. Its primary use is to illustrate the fitting of a Revealed Preference Matchings Model (rpm). The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. This provides such data for a matching between men and women of certain characteristics (or shared characteristics) of people of the opposite sex.

Usage

```
data(fauxmatching)
```

Format

`fauxmatching` is a list containing a pair of `data.frame` objects: `Xdata` and `Zdata`.

`Xdata` is for women. Each row is a woman, each column is a variable on that women or her partnerships. The women's ID variable s called `pid` and the variable with the ID of the women's partner is called `pair_id`. If the women is single the men's ID is NA. `Zdata` is for men. Each row is a man, each column is a variable on that men The men's ID variable is called `pid`.

pair_id The ID of the person's partner. This is in both `Xdata` and `Zdata`.

sampled The indicator that the person was sampled directly (as distinct from being included as the match of a directly sampled person. All single people are directly sampled. This is in both `Xdata` and `Zdata`.

Details

The pairings are determined by the `pair_id` variable in `Xdata`. If that variable is NA then the women is assumed to be single. If men are listed in `Zdata` and are not partnered then they are assumed single. Weights are specified by three optional variables in `Xdata`.

X_w The weight variable for women. The sum of the weights of the sampled women is the number of women in the population.

Z_w The weight variable for men. The sum of the weights of the sampled men is the number of men in the population.

pair_w The weight variable for pairs.

Value

No return value, called for side effects.

Source

The data set is simulation based upon an rpm model fit to data from the 2008 SIPP.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
```

```
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled")
summary(fit)
```

`Gale_Shapley` *This is the version of Gale-Shapley stable matching algorithm (translated from the Matlab code in Menzel (2015)).*

Description

This code allows the self-matched option

Usage

```
Gale_Shapley(U, V, return.data.frame = FALSE, cpp = TRUE, nmax = 10 * nrow(U))
```

Arguments

`U`	The utility matrix for the women's side. Each row is a woman, each column is a man. The matrix entry (i,j) is the utility that woman `i` gains from pairing with man `j`. In other words, the utility is computed from woman `i`'s perspective.
`V`	The utility matrix for the men's side. Each column is a man, each row is a woman. The matrix entry (i,j) is the utility that man `j` gains from pairing with woman `i`. In other words, the utility is computed from man `j`'s perspective.
`return.data.frame`	logical Should a `data.frame` of the matching be returned instead of the paring matrix mu?
`cpp`	logical Should the `Rcpp` version of the code be used. This is much faster and uses a lot less memory.
`nmax`	count The maximum number of iterations of the inner loop within the Gale-Shapley algorithm. This can be reduced to speed up the algorithm at the potential cost of many partnerships being non-equilibruim.

Value

The function return depends on the `return.data.frame` value. If TRUE, it returns

`data.frame`	a two-column `data.frame` with the first column a women's index and the second column the men's index of their partner. It has as many rows as there are partnerships.

If FALSE, it returns the following matrix:

mu — If cpp=TRUE, a vector of length the number of women (nrow(U)) with the index of the matching man (i.e., the index is the row in V of the man). If there is no matching man, the index is 0. This can be used to reconstruct the matching matrix. If cpp=FALSE, the matching matrix, where 1 represents a pairing, 0 otherwise. Each row is a woman, each column is a man. The order of the rows is the same as the rows in U. The order of the columns is the same as the columns in V.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

rpm

---

gof — *Calculate goodness-of-fit statistics for Revealed Preference Matchings Model based on observed data*

Description

gof.rpm ... It is typically based on the estimate from a rpm() call.

Usage

```
gof(object, ...)

## S3 method for class 'rpm'
gof(
  object,
  ...,
  empirical_p = TRUE,
  compare_sim = "sim-est",
  control = object$control,
  reboot = FALSE,
  verbose = FALSE
)

## S3 method for class 'gofrpm'
plot(x, ..., cex.axis = 0.7, main = "Goodness-of-fit diagnostics")
```

Arguments

object	list; an object of classrpm that is typically the result of a call to rpm().
...	Additional arguments, to be passed to lower-level functions.
empirical_p	logical; (Optional) If TRUE the function returns the empirical p-value of the sample statistic based on nsim simulations
compare_sim	string; describes which two objects are compared to compute simulated goodness-of-fit statistics; valid values are "sim-est": compares the marginal distribution of pairings in a simulated sample to the rpm model estimate of the marginal distribution based on that same simulated sample; mod-est: compares the marginal distribution of pairings in a simulated sample to the rpm model estimate used to generate the sample
control	A list of control parameters for algorithm tuning. Constructed using control.rpm, which should be consulted for specifics.
reboot	logical; if this is TRUE, the program will rerun the bootstrap at the coefficient values, rather than expect the object to contain a bs.results component with the bootstrap results run at the solution values. The latter is the default for rpm fits.
verbose	logical; if this is TRUE, the program will print out additional information, including data summary statistics.
x	a list, usually an object of class gofrpm
cex.axis	the magnification of the text used in axis notation;
main	Title for the goodness-of-fit plots.

Details

The function rpm is used to fit a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

The model represents the dyadic utility functions as deterministic linear utility functions of dyadic variables. These utility functions are functions of observed characteristics of the women and men. These functions are entered as terms in the function call to rpm. This function simulates from such a model.

Value

gof.rpm returns a list consisting of the following elements:

observed_pmf	numeric matrix giving observed probability mass distribution over different household types
model_pmf	numeric matrix giving expected probability mass distribution from rpm model
obs_chi_sq	the count-based observed chi-square statistic comparing marginal distributions of the population the data and the model estimate

`obs_chi_sq_cell`
: the contribution to the observed chi-squared statistic by household type

`obs_kl`
: the Kullback-Leibler (KL) divergence computed by comparing the observed marginal distributions to the expected marginal distribution based on the `rpm` model estimate

`obs_kl_cell`
: the contribution to the observed KL divergence by household type

`empirical_p_chi_sq`
: the proportion of simulated chi-square statistics that are greater than or equal to the observed chi-square statistic

`empirical_p_kl`
: the proportion of simulated KL divergences that are greater than or equal to the observed KL divergence

`chi_sq_simulated`
: vector of size `nsim` storing all simulated chi-square statistics

`kl_simulated`
: vector of size `nsim` storing all simulated KL divergences

`chi_sq_cell_mean`
: Mean contributions of each household type to the simulated chi_sq statistic

`chi_sq_cell_sd`
: Standard deviation of the contributions of each household type to the simulated chi_sq statistics

`chi_sq_cell_median`
: Median contributions of each household type to the simulated chi_sq statistic

`chi_sq_cell_iqr`
: Interquartile range of the contributions of each household type to the simulated chi_sq statistics

`kl_cell_mean`
: Mean contributions of each household type to the simulated KL divergences

`kl_cell_sd`
: Standard deviation of the contributions of each household type to the simulated KL divergencesc

`kl_cell_median`
: Median contributions of each household type to the simulated KL divergences

`kl_cell_iqr`
: Interquartile range of the contributions of each household type to the simulated KL divergences

Methods (by class)

- `gof(rpm)`: Calculate goodness-of-fit statistics for Revealed Preference Matchings Model based on observed data

Functions

- `plot(gofrpm)`: `plot.gofrpm` plots diagnostics such empirical p-value based on chi-square statistics and KL divergences. See `rpm` for more information on these models.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

Examples

```
library(rpm)

data(fauxmatching)
fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled")
a <- gof(fit)
```

`list_rhs.formula` *Returns a list containing the terms in a given formula*

Description

Returns a list containing the terms in a given formula

Usage

```
list_rhs.formula(object)
```

Arguments

`object`	formula A formula having a right-hand-side that can be interpretated as a rpm specification. returns a list containing terms in a given formula, handling `+` and `-` operators and parentheses, and keeping track of whether a term has a plus or a minus sign.

Value

`list_rhs.formula` returns a list of formula terms, with an additional numerical vector attribute `"sign"` with of the same length, giving the corresponding term's sign as `+1` or `-1`.

logLik.rpm *A* logLik *method for ['rpm'] fits.*

Description

A function to return the log-likelihood associated with an rpm fit

Usage

```
## S3 method for class 'rpm'
logLik(object, ...)
```

Arguments

object	An rpm fit, returned by rpm.
...	Other arguments to the likelihood functions.

Value

a logLik object.

See Also

logLik, logLikNull

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled",sampling_design="stock-flow")
logLik(fit)
```

logLikNull *Calculate the null model likelihood*

Description

Calculate the null model likelihood

Usage

```
logLikNull(object, ...)

## S3 method for class 'rpm'
logLikNull(object, ...)
```

Arguments

object	a fitted model.
...	further arguments to lower-level functions.
	logLikNull computes, when possible the log-probability of the data under the null model (reference distribution).

Value

logLikNull returns an object of type logLik if it is able to compute the null model probability, and NA otherwise.

Methods (by class)

- logLikNull(rpm): A method for [‘rpm‘] fits to compute the null likelihood (that is, relative to the constant only model).

message_print *[‘print‘] objects to the [‘message‘] output.*

Description

A thin wrapper around [‘print‘] that captures its output and prints it as a [‘message‘], usually to STDERR. Tis is part of [‘statnet.common‘].

Usage

```
message_print(..., messageArgs = NULL)
```

Arguments

...	arguments to [‘print‘].
messageArgs	a list of arguments to be passed directly to [‘message‘].

Value

No return value, called for side effects.

Examples

```
cat(1:5)

print(1:5)
message_print(1:5) # Looks the same (though may be in a different color on some frontends).

suppressMessages(print(1:5)) # Still prints
suppressMessages(message_print(1:5)) # Silenced
```

---

microsimulate *Micro simulate a population from a Revealed Preference Matchings Model*

Description

microsimulate simulates a population of the pairs and singles from a Revealed Preference Matchings Model. It is typically based on the estimate from a rpm() call.

Usage

```
microsimulate(
  object,
  nsim = 1,
  seed = NULL,
  pmfW_N = NULL,
  pmfM_N = NULL,
  large.population = TRUE,
  bootstrap = FALSE,
  control = control.rpm(),
  counts.only = FALSE,
  verbose = FALSE
)
```

Arguments

object	list; an object of classrpm that is typically the result of a call to rpm().
nsim	Number of matchings to be randomly drawn from the given model on the set of all matchings / singles.

seed	integer; (Optional) random number seed.
pmfW_N	vector; The population count of the number of women of each type. This should be compatible with the type in the object.
pmfM_N	vector; The population count of the number of men of each type. This should be compatible with the type in the object.
large.population	logical; If TRUE a large population approximation is used to generate the matchings (rather than the individual level generation of utilities). This is much faster and uses a lot less memory. It is TRUE by default. If used, a sample is drawn rather than the population being returned. The sample size is controlled by pmfW_N and pmfM_N.
bootstrap	logical; If TRUE the original population is sampled from. If FALSE the population underlying the fitted model is sampled from.
control	A list of control parameters for algorithm tuning. Constructed using control.rpm, which should be consulted for specifics.
counts.only	logical; If TRUE only the matrices of counts and the PMF of the population of households is returned. If FALSE It is FALSE by default.
verbose	logical; Should verbose messages be printed out.

Details

The function requites the numbers of women of each type and the number of men of each type to be specified.

The function rpm is used to fit a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

The model represents the dyadic utility functions as deterministic linear utility functions of dyadic variables. These utility functions are functions of observed characteristics of the women and men. These functions are entered as terms in the function call to rpm. This function simulates a population from such a model.

Value

A list of lists, each a simulation from the population. Each of the simulation lists contains components population being a list with components Xdata and Zdata (for use with rpm()).

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

Examples

```
library(rpm)

data(fauxmatching)
fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled")
num_women = fit$N*exp(fit$gw)
num_men   = fit$N*exp(fit$gm)
pmfW_N <- round(fit$pmfW * num_women)
pmfM_N <- round(fit$pmfM * num_men)
a <- microsimulate(fit, pmfW_N=pmfW_N, pmfM_N=pmfM_N)
```

---

rpm *Fit a Revealed Preference Matchings Model*

Description

rpm estimates the parameters of a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

Usage

```
rpm(
  formula,
  Xdata,
  Zdata,
  Xid = NULL,
  Zid = NULL,
  pair_id = NULL,
  X_w = NULL,
  Z_w = NULL,
  pair_w = NULL,
  sampled = NULL,
  sampling_design = "stock-flow",
  fixed.margins = NULL,
  control = control.rpm(),
  verbose = FALSE
)
```

Arguments

formula	formula; an formula object, of the form ~ <model terms>. For the details on the possible <model terms>, see rpm-terms.
Xdata	data.frame for women. Each row is a woman, each column is a variable on that women or her partnerships. It must contain the women's ID variable (see Xid) and a variable with the ID of the women's partner. If the women is single the men's ID should be NA.
Zdata	data.frame for men. Each row is a man, each column is a variable on that men It must contain the men's ID variable (see Zid).
Xid	string The name of the variable in Xdata containing the IDs of the women.
Zid	string The name of the variable in Zdata containing the IDs of the men.
pair_id	string The name of the variable in Xdata containing the ID of the men paired with the women in Xid. If the women is not paired it must be NA.
X_w	string The name of the variable in Xdata containing the individual weight of the women. If this is NULL then it is assumed the sample is unweighted from a population with 2000 women in it.
Z_w	string The name of the variable in Zdata containing the individual weight of the man If this is NULL then it is assumed the sample is unweighted from a population with 2000 men in it.
pair_w	string The name of the variable in Xdata containing the pair weight of that women. If the women is not paired it should be NA. If this is NULL then it is computed from the individual weights using the sampling_design. Note that the pair weights currently do not play a role in the estimation. They do in the quasi-likelihood version of the code. If this is NULL then it is assumed the sample is unweighted from a population with 2000 men in it.
sampled	string The name of the logical variable in Xdata and Zdata containing the indicator that the person was sampled directly (as distinct from being included as the match of a directly sampled person. All single people are directly sampled.
sampling_design	string; The name of the sampling protocol used to select the survey data. Valid values are "stock-flow" (default) (individuals are sampled, data contains both singles and couples); "stock-stock" (households are sampled, each household can be a single or a couple); "census" (the sample is a census of the population of people).
fixed.margins	list If not NULL the numbers of men and women (i.e, in pmfW and pmfM) are assumed determined by outside information and are hence fixed. In this case fixed.margins should be a list with two elements. The first is a vector of women's margins for each type and the second is the men's margins for each type. The default, NULL, means these are estimated from sample data.
control	A list of control parameters for algorithm tuning. Constructed using control.rpm, which should be consulted for specifics.
verbose	logical; if this is TRUE, the program will print out additional information, including data summary statistics.

Details

The pairings are determined by the `pair_id` variable in `Xdata`. If that variable is NA then the women is assumed to be single. If men are listed in `Zdata` and are not partnered then they are assumed single. Weights are specified by three optional variables in `Xdata`.

X_w : This is character string of the name of the weight variable for women. The sum of the weights should be the number of women in the population.

Z_w : This is character string of the name of the weight variable for men. The sum of the weights should be the number of men in the population.

pair_w : This is character string of the name of the weight variable for pairs.

Value

`rpm` returns an object of class `rpm.object` that is a list consisting of the following elements:

`coef`	The maximum psuedo-likelihood estimate of θ, the vector of coefficients for the model parameters. This includes the model β and the model Γ.
`coefficients`	The bias-corrected bootstrap estimate of θ, the vector of coefficients for the model parameters. This includes the model β and the model Γ.
`loglik`	The value of the maximized log-likelihood.
`exitflag`	integer value with the status of the optimization (4 is success as `xtol_rel` or `xtol_abs` was reached). Other codes are 1 = generic success; 2 = optimization stopped because `ftol_rel` or `ftol_abs` was reached; 3 = optimization stopped because `stopval` was reached; 4 = optimization stopped because `xtol_rel` or `xtol_abs` was reached; 5 = optimization stopped because `maxeval` was reached; 6 = optimization stopped because `maxtime` was reached.
`call`	the call that was made to `nloptr`.
`x0`	vector with starting values for the optimization.
`message`	more informative message with the status of the optimization.
`iterations`	number of iterations that were executed.
`objective`	value if the objective function in the solution.
`solution`	optimal value of the controls.
`version`	version of NLopt that was used.
`covar`	Approximate covariance matrix of the estimates.
`eq`	Values from the equality constraints. Larger values indicate non-convergence.
`sample`	A matrix with the number of rows the MCMC sample size and the number of rows the number of parameters.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

control.rpm, summary.rpm, print.rpm

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled",sampling_design="stock-flow")
summary(fit)
```

---

rpm-terms *Terms used in a Revealed Preference Matchings Model*

Description

The function rpm is used to fit a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

Details

The model represents the dyadic utility functions as deterministic linear utility functions of dyadic variables. These utility functions are functions of observed characteristics of the women and men. These functions are entered as terms in the function call to rpm. This page describes the possible terms (and hence linear utility functions) included in rpm package.

Value

No return value, called for side effects.

Specifying models

Terms to rpm are specified by a formula to represent the pairings and covariates This is done via a formula, that is, an formula object, of the form ~ <term 1> + <term 2> ..., where <term 1>, <term 2>, etc, are each terms chosen from the list given below.

`absdiff(attr)` **(quantitative attribute),** `absdiff(attr)` **(quantitative attribute)** *Absolute difference:* The `attr` argument specifies a quantitative attribute This term adds one statistic to the model equaling `abs(attr[i]-attr[j])` for all women-man dyad (i,j).

`W_greaterthan(attr)` *Women's value greater than the men's value* Adds one statistic indicating if the women's value exceeds the men's value.

`M_greaterthan(attr)` *Men's value greater than the women's value* Adds one statistic indicating if the men's value exceeds the women's value.

`W_atleast(attr,threshold=0)` *Values greater than or equal to a threshold* Adds one statistic indicating if the women's value of the attribute equals or exceeds `threshold`.

`W_atmost(threshold=0)` *Values less than or equal to a threshold* Adds one statistic indicating if the women's value equals or is exceeded by `threshold`.

`W_cov(attr)` **(quantitative attribute),** `W_cov(attr)` **(quantitative attribute)** *Main effect of a covariate for women:* The `attr` argument specifies a quantitative attribute This term adds a single statistic equaling the value of `attr(i)` for women i in the dyad. For categorical attributes, see `W_factor`.

`diff(attr)` **(quantitative attribute),** `diff(attr)` **(quantitative attribute)** *Woman's Gap:* The `attr` argument specifies a quantitative attribute This term adds one statistic to the model being `attr[i]-attr[j]` for women i and man j. Specifically, it is the excess of the woman's value over the man's value.

`WtoM_diff(attr, diff)` **(ordinal categorical attribute),** `WtoM_diff(attr)` **(ordinal categorical discrete attribute)** *Woman's Gap:* The `attr` argument specifies a ordinal categorical attribute This term adds one statistic to the model being an indicator that `attr[i]=attr[j]+diff` for women i and man j. Specifically, it indicates if the woman's value is `diff` higher than the man's value.

`MtoW_diff(attr, diff)` **(ordinal categorical attribute),** `MtoW_diff(attr)` **(ordinal categorical discrete attribute)** *Man's Gap:* The `attr` argument specifies a ordinal categorical attribute This term adds one statistic to the model being an indicator that `attr[j]=attr[i]+diff` for women i and man j. Specifically, it indicates if the man's value is `diff` higher than the woman's value.

`MtoW_diff(attr)` **(quantitative attribute),** `MtoW_diff(attr)` **(quantitative attribute)** *Difference:* The `attr` argument specifies a quantitative attribute This term adds one statistic to the model `attr[j]-attr[i]` for women i and man j.

`W_factor(attr, base=1, levels=-1)` **(categorical attribute),** `W_factor(attr, base=1, levels=-1)` **(categorical attri** *Factor attribute effect for women:* The `attr` argument specifies a categorical attribute This term adds multiple statistics to the model, one for each of (a subset of) the unique values of the `attr` attribute. Each of these statistics indicates if the women's has that attribute.

`homophily(attr)` *Uniform homophily effect:* The `attr` argument specifies a categorical attribute This term adds one statistic to the model indicating that the dyad matches on that attribute.

`match(attr, diff=FALSE, collapse=NULL)` *Attribute-based homophily effect:* The `attr` argument specifies a categorical attribute This term adds one statistic to the model for each categorical level, unless `diff` is set to `TRUE`, in which case the term adds multiple statistics to the model, one for each of (a subset of) the unique values of the `attr` attribute. If `diff` is set to `TRUE`, the optional argument `collapse` control what dyads are collapsed (or pooled). Specifically, it is a list of indices of attribute values which are to be collapsed into a single term. For example, `collapse=list(c(1,4))` will collapse the `(1,1)` and the `(4,4)` dyads into a single term (and group). Multiple lists can be included with arbitrary numbers of dyads in a group.

`mix(attr, base=NULL, collapse=NULL)` *Attribute mixing:* The `attr` argument specifies a categorical attributes By default, this term adds one statistic to the model for each possible pairing of attribute values. The statistic indicates if the dyad has that pairing of values. In other words, this term produces one statistic for every entry in the mixing matrix for the attribute(s). The ordering of the attribute values is lexicographic: alphabetical (for nominal categories) or numerical (for ordered categories). The optional argument `base` control what statistics are included in the model, specifically it lists the index of the omitted terms (in order). For example, `base=2` omits the second term. The optional argument `collapse` control what dyads are collapsed (or pooled). Specifically, it is a list of lists. Each element of the list is a list of dyads which are to be collapsed into a single term. For example, `collapse=list(list(c(1,4),c(2,4)))` will collapse the `(1,4)` and the `(2,4)` dyads into a single term (and group). Multiple lists can be included with arbitrary numbers of dyads in a group.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

`rpm` package, `rpm`

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled")
summary(fit)
```

`rpm.model.functions` *Creates a model function list for the continuous terms in a Revealed Preference Matchings Model*

Description

`rpm.model.matrix` assumes a bipartite network (i.e. two-sided matching market) It creates a model matrix according to the formula passed in. See `rpm-terms` for a description of the possible terms.

Usage

```
rpm.model.functions(model.terms, control)
```

Arguments

model.terms	For the details on the possible countinuous <model terms>, see rpm-terms. This includes the covariates used to construct the model matrix. They are used in conjunction with the model terms.
control	A list of control parameters for algorithm tuning. Constructed using control.rpm, which should be consulted for specifics.

Value

A list of model terms as bivariate functions.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

rpm

Examples

```
# nothing yet
```

`rpm.model.matrix` *Creates a model matrix to estimate the parameters of a Revealed Preference Matchings Model*

Description

rpm.model.matrix assumes a bipartite network (i.e. two-sided matching market) It creates a model matrix according to the formula passed in. See rpm-terms for a description of the possible terms.

Usage

```
rpm.model.matrix(model.terms, Xall, Zall, intercept = TRUE)
```

Arguments

model.terms	For the details on the possible <model terms>, see rpm-terms. This includes the covariates used to construct the model matrix. They are used in conjunction with the model terms.
Xall	the unique types of women
Zall	the unique types of men
intercept	logical; If TRUE, the default, an intercept term is prepended.

Value

A list consists of the following elements:

X	the model matrix for women.
Z	the model matrix for men.
Xnames	the names of the covariates for women.
Znames	the names of the covariates for men.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

rpm

Examples

```
# nothing yet
```

---

rpmpopulationpmf *Compute the population distribution of pairs and singles from a Revealed Preference Matchings Model*

Description

rpmpopulationpmf computes the probability mass function for a population of the pairs and singles from a Revealed Preference Matchings Model based on arbitary availability distribution and preferences. It is typically based on the estimate from a rpm() call.

Usage

```
rpmpopulationpmf(
  object,
  N = 2000,
  num_women = NULL,
  num_men = NULL,
  pmfW = NULL,
  pmfM = NULL,
  verbose = FALSE
)
```

Arguments

object	list; an object of classrpm that is typically the result of a call to rpm().
N	integer; The total population size. This must be set. The number of women and men are derived from the (weighted) data.
num_women	integer; (Optional) The number of women in the population.
num_men	integer; (Optional) The number of men in the population.
pmfW	vector; (Optional) The population proportions of the numbers of women of each type. This should be compatible with the type in the object.
pmfM	vector; (Optional) The population proportions of the numbers of men of each type. This should be compatible with the type in the object.
verbose	logical; Should verbose messages be printed out.

Details

The function rpm is used to fit a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

The model represents the dyadic utility functions as deterministic linear utility functions of dyadic variables. These utility functions are functions of observed characteristics of the women and men. These functions are entered as terms in the function call to rpm. This function simulates from such a model.

Value

A list of data.frame, each a simulation from the population.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled")
a <- rpmpopulationpmf(fit)
```

---

rpm_MLPLE *Fit a Revealed Preference Matchings Model*

Description

rpm_MLPLE estimates the parameters of a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

Usage

```
rpm_MLPLE(
  formula,
  Xdata,
  Zdata,
  Xid = NULL,
  Zid = NULL,
  pair_id = NULL,
  X_w = NULL,
  Z_w = NULL,
  pair_w = NULL,
  sampled = NULL,
  sampling_design = "stock-flow",
  fixed.margins = fixed.margins,
  control = control.rpm(),
  verbose = FALSE
)
```

Arguments

formula	formula; an formula object, of the form ~ <model terms>. For the details on the possible <model terms>, see rpm-terms.
Xdata	data.frame for women. Each row is a woman, each column is a variable on that women or her partnerships. It must contain the women's ID variable (see Xid) and a variable with the ID of the women's partner. If the women is single the men's ID should be NA.
Zdata	data.frame for men. Each row is a man, each column is a variable on that men It must contain the men's ID variable (see Zid).
Xid	string The name of the variable in Xdata containing the IDs of the women.
Zid	string The name of the variable in Zdata containing the IDs of the men.
pair_id	string The name of the variable in Xdata containing the ID of the men paired with the women in Xid. If the women is not paired it must be NA.
X_w	string The name of the variable in Xdata containing the individual weight of the women. If this is NULL then it is assumed the sample is unweighted from a population with 2000 women in it.
Z_w	string The name of the variable in Zdata containing the individual weight of the man If this is NULL then it is assumed the sample is unweighted from a population with 2000 men in it.
pair_w	string The name of the variable in Xdata containing the pair weight of that women. If the women is not paired it should be NA. If this is NULL then it is computed from the individual weights using the sampling_design. Note that the pair weights currently do not play a role in the estimation. They do in the quasi-likelihood version of the code. If this is NULL then it is assumed the sample is unweighted from a population with 2000 men in it.
sampled	string The name of the logical variable in Xdata and Zdata containing the indicator that the person was sampled directly (as distinct from being included as the match of a directly sampled person. All single people are directly sampled.
sampling_design	string; The name of the sampling protocol used to select the survey data. Valid values are "stock-flow" (default) (individuals are sampled, data contains both singles and couples); "stock-stock" (households are sampled, each household can be a single or a couple); "census" (the sample is a census of the population of people).
fixed.margins	list If not NULL the numbers of men and women (i.e, in pmfW and pmfM) are assumed determined by outside information and are hence fixed. In this case fixed.margins should be a list with two elements. The first is a vector of women's margins for each type and the second is the men's margins for each type. The default, NULL, means these are estimated from sample data.
control	A list of control parameters for algorithm tuning. Constructed using control.rpm, which should be consulted for specifics.
verbose	logical; if this is TRUE, the program will print out additional information, including data summary statistics.

Details

It is usually called via the rpm function.

The pairings are determined by the pair_id variable in Xdata. If that variable is NA then the women is assumed to be single. If men are listed in Zdata and are not partnered then they are assumed single. Weights are specified by three optional variables in Xdata.

X_w : This is character string of the name of the weight variable for women. The sum of the weights should be the number of women in the population.

Z_w : This is character string of the name of the weight variable for men. The sum of the weights should be the number of men in the population.

pair_w : This is character string of the name of the weight variable for pairs.

Value

rpm returns an object of class rpm.object that is a list consisting of the following elements:

coefficients	The bias-corrected bootstrap estimate of θ, the vector of coefficients for the model parameters. This includes the model β and the model Γ.
loglik	The value of the maximized log-likelihood.
exitflag	integer value with the status of the optimization (4 is success as xtol_rel or xtol_abs was reached). Other codes are 1 = generic success; 2 = optimization stopped because ftol_rel or ftol_abs was reached; 3 = optimization stopped because stopval was reached; 4 = optimization stopped because xtol_rel or xtol_abs was reached; 5 = optimization stopped because maxeval was reached; 6 = optimization stopped because maxtime was reached.
call	the call that was made to nloptr.
x0	vector with starting values for the optimization.
message	more informative message with the status of the optimization.
iterations	number of iterations that were executed.
objective	value if the objective function in the solution.
solution	optimal value of the controls.
version	version of NLopt that was used.
covar	Approximate covariance matrix of the estimates.
eq	Values from the equality constraints. Larger values indicate non-convergence.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

control.rpm, summary.rpm, print.rpm

Examples

```
library(rpm)

data(fauxmatching)
fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled",sampling_design="stock-flow")
summary(fit)
```

`simulate.rpm` *Simulate a sample of pairs and singles from a Revealed Preference Matchings Model*

Description

`simulate.rpm` simulates a population of the pairs and singles from a Revealed Preference Matchings Model. It is typically based on the estimate from a `rpm()` call.

Usage

```
## S3 method for class 'rpm'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  ...,
  N = NULL,
  num_women = NULL,
  num_men = NULL,
  pmfW = NULL,
  pmfM = NULL,
  large.population = TRUE,
  num_sampled = NULL,
  bootstrap = FALSE,
  sampling_design = NULL,
  control = control.rpm(),
  verbose = FALSE
)
```

Arguments

object	list; an object of classrpm that is typically the result of a call to rpm().
nsim	Number of matchings to be randomly drawn from the given model on the set of all matchings / singles.
seed	integer; (Optional) random number seed.
...	Additional arguments, to be passed to lower-level functions.
N	integer; The total population size. This must be set. The number of women and men are derived from the (weighted) data.
num_women	integer; (Optional) The number of women in the population.
num_men	integer; (Optional) The number of men in the population.
pmfW	vector; (Optional) The population proportions of the numbers of women of each type. This should be compatible with the type in the object.
pmfM	vector; (Optional) The population proportions of the numbers of men of each type. This should be compatible with the type in the object.
large.population	logical; If TRUE a large population approximation is used to generate the matchings (rather than the individual level generation of utilities). This is much faster and uses a lot less memory. It is TRUE by default. If used, a sample is drawn rather than the population being returned. The sample size is controlled by num_sampled.
num_sampled	integer; The size of the sample to be drawn. For "stock-stock" sampling this is the number of sampled households. For "stock-flow" it is the number of sampled people. For "census" it is the total population size, N. If NULL the size is the same as the passed fitted object (that is, the original data), although this is only a guess and it should be explicitly set.
bootstrap	logical; If TRUE the original population is sampled from. If FALSE the population underlying the fitted model is sampled from.
sampling_design	string; The name of the sampling protocol used to select the survey data. Valid values are "stock-flow" (individuals are sampled, data contains both singles and couples); "stock-stock" (households are sampled, each household can be a single or a couple); "census" (the sample is a census of the population of people). The final option, the default, is NULL whereby the design is taken from the passed object.
control	A list of control parameters for algorithm tuning. Constructed using control.rpm, which should be consulted for specifics.
verbose	logical; Should verbose messages be printed out.

Details

The function rpm is used to fit a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities. It does this using an large-population likelihood based on ideas from Dagsvik (2000), Menzel (2015) and Goyal et al (2023).

The model represents the dyadic utility functions as deterministic linear utility functions of dyadic variables. These utility functions are functions of observed characteristics of the women and men. These functions are entered as terms in the function call to rpm. This function simulates from such a model.

Value

A list of data.frame, each a simulation from the population.

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

Examples

```
library(rpm)

data(fauxmatching)
fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled")
a <- simulate(fit)
```

`summary.rpm` *Summarizing rpm Model Fits*

Description

[base::summary()] method for [rpm()] fits.

Usage

```
## S3 method for class 'rpm'
summary(
  object,
  ...,
  digits = max(3, getOption("digits") - 3),
  correlation = FALSE,
  covariance = FALSE,
```

```
  include.single = TRUE
)

## S3 method for class 'summary.rpm'
print(
  x,
  digits = max(3, getOption("digits") - 3),
  correlation = FALSE,
  covariance = FALSE,
  signif.stars = getOption("show.signif.stars"),
  eps.Pvalue = 1e-04,
  print.header = TRUE,
  print.formula = FALSE,
  print.fitinfo = TRUE,
  print.coefmat = TRUE,
  print.message = TRUE,
  print.deviances = TRUE,
  print.drop = TRUE,
  ...
)
```

Arguments

`object`	an object of class `rpm`, usually, a result of a call to [rpm()].
`...`	For [summary.rpm()] additional arguments are passed to [logLik.rpm()]. For [print.summary.rpm()], to [stats::printCoefmat()].
`digits`	significant digits for coefficients. The default is max(3, getOption("digits")-3).
`correlation`	logical whether the correlation matrix of the estimated parameters should be printed (T or F); default=FALSE
`covariance`	logical whether the covariance matrix of the estimated parameters should be printed (T or F); default=FALSE
`include.single`	logical; if ‘TRUE‘, include in the summary table the coefficients of the log-odds of being single for each category of women and men.
`x`	object of class ‘summary.rpm‘ returned by [summary.rpm()].
`signif.stars`	whether to print dots and stars to signify statistical significance. See [print.summary.lm()].
`eps.Pvalue`	p-values below this level will be printed as "<‘eps.Pvalue‘".
`print.formula`, `print.fitinfo`, `print.coefmat`, `print.message`, `print.deviances`, `print.drop`, `print.header`	which components of the fit summary to print.

Details

[summary.rpm()] tries to be smart about formatting the coefficients, standard errors, etc.

The default printout of the summary object contains the call, number of iterations used, null and residual deviances, and the values of AIC and BIC. The coefficient table contains the following columns:

- ‘Estimate‘, ‘Std. Error‘ - parameter estimates and their standard errors - ‘z value‘, ‘Pr(>|z|)‘ - z-test and p-values

Value

The function [summary.rpm()] computes and returns a list of summary statistics of the fitted [rpm()] model given in ‘object‘. Note that for backwards compatibility, it returns the coefficient table.

The returned object is a list of class "summary.rpm" with the following elements:

`formula`	ERGM model formula
`digits`	the ’digits’ inputted to <summary.rpm> or the default value (despite the fact the digits will be 5)
`correlation`, `covariance`	whether to print correlation/covariance matrices of the estimated parameters
`iterations`	object$iterations
`control`	the [control.rpm()] object used
`samplesize`	MCMC sample size
`message`	optional message on the validity of the standard error estimates
`aic.null`, `bic.null`	values of AIC and BIC for the null model
`aic`, `bic`	values of AIC and BIC
`coefficients`	data frames with model parameters and associated statistics
`asycov`	asymptotic covariance matrix
`asyse`	asymptotic standard error matrix
`offset`, `drop`, `estimate`, `iterations`, `mle.lik`, `null.lik`	see documentation of the object returned by [rpm()]

See Also

The model fitting function [rpm()], [print.rpm()], and [base::summary()]. Function [stats::coef()] will extract the data frame of coefficients with standard errors, t-statistics and p-values.

Examples

```
library(rpm)
data(fauxmatching)

fit <- rpm(~match("edu") + WtoM_diff("edu",3),
          Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
          X_w="X_w", Z_w="Z_w",
          pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
          sampled="sampled",sampling_design="stock-flow")
summary(fit)
```

summary_rpm *Summarize Revealed Preference Matchings data via a Model Specification*

Description

summary_rpm produces tabular summaries of data revealed preference matchings based on a formula specifying a revealed preference model for men and women of certain characteristics (or shared characteristics) of people of the opposite sex. The model assumes a one-to-one stable matching using an observed set of matchings and a set of (possibly dyadic) covariates to estimate the parameters for linear equations of utilities.

Usage

```
summary_rpm(
  formula,
  Xdata,
  Zdata,
  Xid = NULL,
  Zid = NULL,
  pair_id = NULL,
  X_w = NULL,
  Z_w = NULL,
  pair_w = NULL,
  sampled = NULL,
  sampling_design = "stock-flow",
  control = control.rpm(),
  verbose = FALSE
)
```

Arguments

formula	formula; an formula object, of the form ~ <model terms>. For the details on the possible <model terms>, see rpm-terms.
Xdata	data.frame for women. Each row is a woman, each column is a variable on that women or her partnerships. It must contain the women's ID variable (see Xid) and a variable with the ID of the women's partner. If the women is single the men's ID should be NA.
Zdata	data.frame for men. Each row is a man, each column is a variable on that men It must contain the men's ID variable (see Zid).
Xid	string The name of the variable in Xdata containing the IDs of the women.
Zid	string The name of the variable in Zdata containing the IDs of the men.
pair_id	string The name of the variable in Xdata containing the ID of the men paired with the women in Xid. If the women is not paired it must be NA.

`X_w` string The name of the variable in `Xdata` containing the individual weight of the women. If this is NULL then it is assumed the sample is unweighted from a population with 2000 women in it.

`Z_w` string The name of the variable in `Zdata` containing the individual weight of the man If this is NULL then it is assumed the sample is unweighted from a population with 2000 men in it.

`pair_w` string The name of the variable in `Xdata` containing the pair weight of that women. If the women is not paired it should be NA. If this is NULL then it is computed from the individual weights using the `sampling_design`. Note that the pair weights currently do not play a role in the estimation. They do in the quasi-likelihood version of the code. If this is NULL then it is assumed the sample is unweighted from a population with 2000 men in it.

`sampled` string The name of the logical variable in `Xdata` and `Zdata` containing the indicator that the person was sampled directly (as distinct from being included as the match of a directly sampled person. All single people are directly sampled.

`sampling_design`
string; The name of the sampling protocol used to select the survey data. Valid values are `"stock-flow"` (default) (individuals are sampled, data contains both singles and couples); `"stock-stock"` (households are sampled, each household can be a single or a couple); `"census"` (the sample is a census of the population of people).

`control` A list of control parameters for algorithm tuning. Constructed using `control.rpm`, which should be consulted for specifics.

`verbose` logical; if this is `TRUE`, the program will print out additional information, including data summary statistics.

Details

The pairings are determined by the `pair_id` variable in `Xdata`. If that variable is NA then the women is assumed to be single. If men are listed in `Zdata` and are not partnered then they are assumed single. Weights are specified by three optional variables in `Xdata`.

X_w : This is character string of the name of the weight variable for women. The sum of the weights should be the number of women in the population.

Z_w : This is character string of the name of the weight variable for men. The sum of the weights should be the number of men in the population.

pair_w : This is character string of the name of the weight variable for pairs.

Value

`summary` returns a list with many components, like `rpm` object without the model estimates. In particular it includes `stats` and `popstats`. `stats` is the named vector of sample statistics from the model. while `popstats` is the named vector of population statistics from the model. It alos includes `counts` and `pmf`. Each of these is a contingency table in array representation of S3 class `c("xtabs", "table")`, with a `"call"`

References

Goyal, Shuchi; Handcock, Mark S.; Jackson, Heide M.; Rendall, Michael S. and Yeung, Fiona C. (2023). *A Practical Revealed Preference Model for Separating Preferences and Availability Effects in Marriage Formation*, *Journal of the Royal Statistical Society*, A. doi:10.1093/jrsssa/qnad031

Dagsvik, John K. (2000) *Aggregation in Matching Markets International Economic Review*,, Vol. 41, 27-57. JSTOR: https://www.jstor.org/stable/2648822, doi:10.1111/14682354.00054

Menzel, Konrad (2015). *Large Matching Markets as Two-Sided Demand Systems* Econometrica, Vol. 83, No. 3 (May, 2015), 897-941. doi:10.3982/ECTA12299

See Also

control.rpm, summary.rpm, rpm

Examples

```
library(rpm)
data(fauxmatching)
summary_rpm(~match("edu") + WtoM_diff("edu",3),
        Xdata=fauxmatching$Xdata, Zdata=fauxmatching$Zdata,
        X_w="X_w", Z_w="Z_w",
        pair_w="pair_w", pair_id="pair_id", Xid="pid", Zid="pid",
        sampled="sampled",sampling_design="stock-flow")
```

`ult<-` *Extract or replace the *ult*imate (last) element of a vector or a list, or an element counting from the end.*

Description

Extract or replace the *ult*imate (last) element of a vector or a list, or an element counting from the end.

Usage

```
ult(x, i = 1L) <- value

ult(x, i = 1L)
```

Arguments

`x`	a vector or a list.
`i`	index from the end of the list to extract or replace (where 1 is the last element, 2 is the penultimate element, etc.).
`value`	Replacement value for the 'i'th element from the end.

Value

An element of ‘x‘.

Note

Due to the way in which assigning to a function is implemented in R, ‘ult(x) <- e‘ may be less efficient than ‘x[[length(x)]] <- e‘.

Examples

```
(x <- c(1:5))
(ult(x) <- 6)
(ult(x, 2) <- 7) # 2nd last.
x



x <- 1:5
(last <- ult(x))
(penultimate <- ult(x, 2)) # 2nd last.
```

Index