Package 'mpt'

October 11, 2024

Version 1.0-0

Date 2024-10-11

Title Multinomial Processing Tree Models

Depends R (>= 4.3.0), stats

Imports graphics, utils

Suggests knitr, rmarkdown, lattice

VignetteBuilder knitr

Description Fitting and testing multinomial processing tree (MPT) models, a class of nonlinear models for categorical data. The parameters are the link probabilities of a tree-like graph and represent the latent cognitive processing steps executed to arrive at observable response categories (Batchelder & Riefer, 1999 <doi:10.3758/bf03210812>; Erdfelder et al., 2009 <doi:10.1027/0044-3409.217.3.108>; Riefer & Batchelder, 1988 <doi:10.1037/0033-295x.95.3.318>).

License GPL (>= 2)

URL https://www.mathpsy.uni-tuebingen.de/wickelmaier/

NeedsCompilation no

Author Florian Wickelmaier [aut, cre], Achim Zeileis [aut] (<https://orcid.org/0000-0003-0918-3766>)

Maintainer Florian Wickelmaier <wickelmaier@web.de>

Repository CRAN

Date/Publication 2024-10-11 10:00:03 UTC

Contents

`agememory` *Age Differences in Episodic Memory*

Description

Bayen (1990) presented 40 younger and 40 older adults with a list of 50 words to be learned. The list consisted of 20 semantic word pairs and 10 singleton words. In a later memory test, participants freely recalled the presented words. For pairs, responses were classified into four categories: both words in a pair are recalled adjacently (E1) or non-adjacently (E2), one word in a pair is recalled (E3), neither word in a pair is recalled (E4); for singletons, into two categories: word recalled (F1), word not recalled (F2).

The recall frequencies are available in Schmidt et al. (2023).

Usage

```
data(agememory)
```

Format

`agememory` A data frame containing 80 observations of 15 variables:

`group` factor. Younger versus older group of participants.

`id` participant ID within each group.

`age` participant age.

`sex` factor. Participant sex.

`IST70` intelligence score.

`lag0E1, lag0E2, lag0E3, lag0E4` recall frequencies for word pairs presented without lag.

`lag15E1, lag15E2, lag15E3, lag15E4` recall frequencies for word pairs presented with a lag of 15 items in between the two pair members.

`F1, F2` recall frequencies for singleton words.

Source

Schmidt, O., Erdfelder, E., & Heck, D. W. (2023). How to develop, test, and extend multinomial processing tree models: A tutorial. *Psychological Methods*. doi:10.1037/met0000561

References

Bayen, U.J. (1990). Zur Lokalisation von Altersdifferenzen im episodischen Gedaechtnis Erwachsener: Eine Querschnittsuntersuchung auf der Basis eines mathematischen Modells. *Berichte aus dem Psychologischen Institut der Universitaet Bonn*, **16**, 1–125.

See Also

`mpt`.

Examples

```
data(agememory)

aggregate(cbind(lag0E1, lag0E2, lag0E3, lag0E4,
                lag15E1, lag15E2, lag15E3, lag15E4, F1, F2) ~ group,
          data = agememory, sum)
xtabs(~ group + sex, agememory) |> addmargins()
```

---

citysize *City-Size Paired-Comparison Task*

Description

In a city-size paired-comparison task on each trial, participants judge which of two cities is more populous. After the paired comparisons, participants indicate for each city if they recognize its name. Hilbig, Erdfelder, and Pohl (2010) report a series of experiments to evaluate their model of recognition heuristic use at this task.

The `WorldCities` data are from a study designed to be similar to Hilbig et al.'s Experiment 6. The 17 cities were (in order of population; Wikipedia, 2016): Shanghai, Tianjin, Tokyo, Seoul, London, Bangkok, Chongqing, Wuhan, Santiago, Rangun, Ankara, Harbin, Kano, Busan, Durban, Ibadan, Montreal.

The `ItalianCities` data are from a study designed to be similar to Hilbig et al.'s Experiment 7. The 14 cities were: Milan, Naples, Turin, Palermo, Venice, Padua, Taranto, Prato, Reggio Emilia, Perugia, Cagliari, Foggia, Salerno, Ferrara.

Usage

```
data(citysize)
```

Format

`WorldCities` A data frame containing 37 observations of six variables:

`gender` factor. Participant gender.

`age` participant age.

`rt` median response time (in seconds) across paired comparisons.

`group` factor. The control group (`CG`) received standard instructions, the experimental group (`EG`) was instructed to choose the city they recognized whenever possible.

`country` number of cities whose country was correctly identified.

`y` a matrix of aggregate response frequencies per participant. The column names indicate each of eight response categories: correct/false responses when both cities were recognized (`KC`, `KF`), when both were unrecognized (`GC`, `GF`), when only one was recognized and the recognized city was chosen (`RC`, `RF`), and when only one was recognized and the unrecognized city was chosen (`UF`, `UC`).

`ItalianCities` A data frame containing 64 observations of six variables:

`gender`, `age`, `rt`, `y` see above.

`group` factor. The control group (`CG`) received standard instructions, the experimental group (`EG`) was asked to compare the cities with respect to their elevation above sea level.

`knowRH` factor. Does the participant have any knowledge about the recognition heuristic (RH)?

Source

The `WorldCities` data were collected at the Department of Psychology, University of Tuebingen, in June/July 2016. The `ItalianCities` data are from Rettich (2020). The original data are from Castela et al. (2014).

References

Hilbig, B.E., Erdfelder, E., & Pohl, R.F. (2010). One-reason decision-making unveiled: A measurement model of the recognition heuristic. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **36**(1), 123–134. doi:10.1037/a0017518

Castela, M., Kellen, D., Erdfelder, E., & Hilbig, B.E. (2014). The impact of subjective recognition experiences on recognition heuristic use: A multinomial processing tree approach. *Psychonomic Bulletin & Review*, **21**(5), 1131–1138. doi:10.3758/s1342301405874

Rettich, A. (2020). *Application of the recognition heuristic: An experimental validation of the r-model*. Bachelor thesis. University of Tuebingen, Germany. https://osf.io/mz47y/

Wikipedia. (2016). List of cities proper by population. Retrieved Jun 16 from https://en.wikipedia.org/wiki/List_of_cities_proper_by_population.

See Also

`mpt`.

Examples

```
data(citysize)

## Fit r-model separately for each instruction type
mpt(mptspec("rmodel"), unname(WorldCities[WorldCities$group == "CG",  "y"]))
mpt(mptspec("rmodel"), unname(WorldCities[WorldCities$group == "EG", "y"]))

## Test instruction effect on r parameter
city.agg <- aggregate(y ~ group, WorldCities, sum)
y <- as.vector(t(city.agg[, -1]))

m1 <- mpt(mptspec("rmodel", .replicates = 2), y)
m2 <- mpt(update(m1$spec, .restr = list(r2 = r1)), y)
anova(m2, m1)  # more use of RH with recognition instruction

## Fit r-model separately for each task type
mpt(mptspec("rmodel"),
    unname(ItalianCities[ItalianCities$group == "CG",  "y"]))
mpt(mptspec("rmodel"),
    unname(ItalianCities[ItalianCities$group == "EG", "y"]))

## Test task effect on r parameter
city.agg <- aggregate(y ~ group, ItalianCities, sum)
y <- as.vector(t(city.agg[, -1]))

m3 <- mpt(mptspec("rmodel", .replicates = 2), y)
m4 <- mpt(update(m1$spec, .restr = list(r2 = r1)), y)
anova(m4, m3)  # less use of RH with elevation task

## Plot parameter estimates
par(mfrow = 1:2)
dotchart(coef(m1)[c(4, 1:3)], xlim = 0:1, labels = c("a", "b", "g", "r"),
         xlab = "", main = "by instruction type")
points(coef(m1)[c(8, 5:7)], 1:4, pch = 16)
legend(0, 1, c("none", "recognition"), pch = c(1, 16),
       title = "Instruction", bty = "n")
dotchart(coef(m3)[c(4, 1:3)], xlim = 0:1, labels = c("a", "b", "g", "r"),
         xlab = "", main = "by task type")
points(coef(m3)[c(8, 5:7)], 1:4, pch = 16)
legend(0, 1, c("population", "elevation"), pch = c(1, 16),
       title = "Task", bty = "n")
title("Recognition heuristic use", outer = TRUE, line = -1)
mtext("Parameter estimate (r-model)", side = 1, outer = TRUE, line = -2)

## Compare with original results
Hilbig2010 <- rbind(
    WorldCities.CG = c(462, 204, 290, 272, 740, 205, 77, 62),
    WorldCities.EG = c(500, 307, 279, 264, 902, 235, 68, 29),
  ItalianCities.CG = c(232, 78, 135, 136, 465, 65, 56, 16),
  ItalianCities.EG = c(245, 176, 154, 150, 228, 160, 112, 140)
)
apply(Hilbig2010, 1, mpt, spec = mptspec("rmodel"))
```

logLik.mpt *Log-Likelihood of an mpt Object*

Description

Returns the log-likelihood value of the (joint) multinomial processing tree model represented by `object` evaluated at the estimated parameters.

Usage

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

Arguments

`object`	an object inheriting from class `mpt`, representing a fitted multinomial processing tree model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

The log-likelihood of the model represented by `object` evaluated at the estimated parameters.

See Also

`mpt`, `logLik.lm`, `AIC`, `deviance`, `nobs`.

Examples

```
m <- mpt(mptspec("SR2"), c(243, 64, 58, 55))  # from Riefer et al. (2002)
logLik(m)
deviance(m)
AIC(m)
AIC(m, k = log(sum(m$y)))  # BIC w/total number of data points
BIC(m)                     # BIC using nobs()
nobs(m)                    # number of non-redundant response categories
```

`moraldilemma` *Moral Dilemma Judgment*

Description

Hennig and Huetter (2020) proposed a multinomial model of moral dilemma judgment and evaluated the model in a series of experiments. Participants were presented with hypothetical scenarios that required a decision whether or not to break a moral norm.

Berentelg (2020) conducted a replication study that was designed to be similar to Experiment 2b in Hennig and Huetter (2020).

Usage

```
data(moraldilemma)
```

Format

`MDHennig2020` A data frame consisting of seven variables:

`selfrel` factor. Instructions about self-relevant consequences of the decision where either (`absent`) or (`present`).

`congrcy` factor. Endorsement of consequences and norm endorsement lead to different decisions (`incongruent`) or to the same decision (`congruent`).

`default` factor. The norm may be adhered to by continuing (`inaction` default state) or by changing (`action` default state) an ongoing behavior.

`breaknorm` factor. Decision to break the norm.

`exp1`, `exp2b` the aggregate response frequencies for Experiment 1 and 2b, respectively.

`treeid` an identifier for the single trees of the joint multinomial model.

`MDreplication` A data frame containing 751 observations of five variables:

`selfrel` factor. See above.

`gender` factor. Participant gender.

`age` participant age.

`rt` median response time (in seconds) across scenarios.

`y` a matrix of response frequencies per participant. Each column represents a combination of the factors `congrcy`, `default`, and `breaknorm`.

Source

Hennig, M., & Huetter, M. (2020). Revisiting the divide between deontology and utilitarianism in moral dilemma judgment: A multinomial modeling approach. *Journal of Personality and Social Psychology*, **118**(1), 22–56. doi:10.1037/pspa0000173

Berentelg, M. (2020). *Multinomial modeling of moral dilemma judgment: A replication study*. Bachelor thesis. University of Tuebingen, Germany. https://osf.io/mb32t/

See Also

mpt.

Examples

```
data(moraldilemma)

## Exp. 1: proCNI and process dissociation (PD) model
s <- mptspec("proCNI")
exp1 <- subset(MDHennig2020, selfrel == "absent")
mpt(update(s, .restr = list(J=I)),      data = exp1, freqvar = "exp1")
mpt(update(s, .restr = list(I=0, J=1)), data = exp1, freqvar = "exp1")

## Exp. 2b: self-relevant consequences and norm endorsement
s <- mptspec("proCNI", .replicates = 2, .restr = list(J1=I1, J2=I2))
m1 <- mpt(s, data = MDHennig2020, freqvar = "exp2b")
m2 <- mpt(update(m1$spec, .restr = list(N1=N2)), data = m1$y)
anova(m2, m1)

## Replication of Exp. 2b
md.agg <- aggregate(y ~ selfrel, MDreplication, sum)
y <- as.vector(t(md.agg[, -1]))

m3 <- mpt(s, data = y)
m4 <- mpt(update(s, .restr = list(N1=N2)), data = y)
anova(m4, m3)

coefs <- c(diff(coef(m3)[c("N2", "N1")]),
           diff(coef(m1)[c("N2", "N1")]))
names(coefs) <- c("Replication", "Hennig & Huetter\n(2020, Exp. 2b)")
ci <- coefs + rbind(
  qnorm(c(.025, .975))*sqrt(sum(diag(vcov(m3))[c("N2", "N1")])),
  qnorm(c(.025, .975))*sqrt(sum(diag(vcov(m1))[c("N2", "N1")]))
)
dotchart(coefs, pch = 16, xlim = c(-.2, 1),
  xlab = expression(N[absent] - N[present]~"(proCNI model, 95% CI)"),
  main = paste("Self-relevant consequences and norm endorsement",
               "in moral dilemma judgment", sep = "\n"))
abline(v = 0, col = "gray")
arrows(ci[, 1], 1:2, ci[, 2], 1:2, .05, 90, 3)
```

---

mpt *Multinomial Processing Tree (MPT) Models*

Description

Fits a (joint) multinomial processing tree (MPT) model specified by a symbolic description via mptspec.

Usage

```
mpt(spec, data, start = NULL, method = c("BFGS", "EM"), treeid = "treeid",
    freqvar = "freq", optimargs =
      if(method == "BFGS") list(control =
        list(reltol = .Machine$double.eps^(1/1.2), maxit = 1000))
      else list())

## S3 method for class 'mpt'
anova(object, ..., test = c("Chisq", "none"))

## S3 method for class 'mpt'
coef(object, logit = FALSE, ...)

## S3 method for class 'mpt'
confint(object, parm, level = 0.95, logit = TRUE, ...)

## S3 method for class 'mpt'
predict(object, newdata = NULL, type = c("freq", "prob"), ...)

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

Arguments

spec	an object of class mptspec: typically result of a call to mptspec. A symbolic description of the model to be fitted. (See Details and Examples.)
data	a data frame consisting at least of one variable that contains the absolute response frequencies. Alternatively, a (named) vector or matrix of frequencies.
start	a vector of starting values for the parameter estimates between zero and one.
method	optimization method. Implemented are optim(..., method = "BFGS") and the EM algorithm.
treeid	name of the variable that identifies the processing trees of a joint multinomial model. Alternatively, a factor that identifies each tree.
freqvar	if data is a data frame, name of the variable that holds the response frequencies; else ignored.
logit	logical. Parameter estimates on logit or probability scale.
optimargs	a list of arguments passed to the optimization function, either optim or mptEM.
object	an object of class mpt, typically the result of a call to mpt.
test	should the p-values of the chi-square distributions be reported?
parm, level	See confint.default.
newdata	a vector of response frequencies.
type	predicted frequencies or probabilities.
...	additional arguments passed to other methods.

Details

Multinomial processing tree models (Batchelder & Riefer, 1999; Erdfelder et al., 2009; Riefer & Batchelder, 1988) seek to represent the categorical responses of a group of subjects by a small number of latent (psychological) parameters. These models have a tree-like graph, the links being the parameters, the leaves being the response categories. The path from the root to one of the leaves represents the cognitive processing steps executed to arrive at a given response.

If `data` is a data frame, each row corresponds to one response category. If `data` is a vector or matrix, each element or column corresponds to one response category. The order of response categories and of model equations specified in `mptspec` should match.

Joint (or product) multinomial models consist of more than one processing tree. The `treeid` should uniquely identify each tree.

Per default, parameter estimation is carried out by `optim`'s BFGS method on the logit scale with analytical gradients; it can be switched to `mptEM` which implements the EM algorithm.

Value

An object of class `mpt` containing the following components:

`coefficients`	a vector of parameter estimates. For extraction, the `coef` function is preferred.
`loglik`	the log-likelihood of the fitted model.
`nobs`	the number of nonredundant response categories.
`fitted`	the fitted response frequencies.
`goodness.of.fit`	the goodness of fit statistic including the likelihood ratio fitted vs. saturated model (G2), the degrees of freedom, and the p-value of the corresponding chi-square distribution.
`ntrees`	the number of trees in a joint multinomial model.
`n`	the total number of observations per tree.
`y`	the vector of response frequencies.
`pcat`	the predicted probabilities for each response category.
`treeid`	a factor that identifies each tree.
`a`, `b`, `c`	structural constants passed to `mptEM`.
`spec`	the MPT model specification returned by `mptspec`.
`method`	the optimization method used.
`optim`	the return value of the optimization function.

References

Batchelder, W.H., & Riefer, D.M. (1999). Theoretical and empirical review of multinomial process tree modeling. *Psychonomic Bulletin & Review*, **6**(1), 57–86. doi:10.3758/bf03210812

Erdfelder, E., Auer, T., Hilbig, B.E., Assfalg, A., Moshagen, M., & Nadarevic, L. (2009). Multinomial processing tree models: A review of the literature. *Zeitschrift fuer Psychologie*, **217**(3), 108–124. doi:10.1027/00443409.217.3.108

Riefer, D.M., & Batchelder, W.H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, **95**(3), 318–339. doi:10.1037/0033295x.95.3.318

See Also

mptEM, mptspec, simulate.mpt, plot.mpt, residuals.mpt, logLik.mpt, vcov.mpt, optim.

Examples

```
## Storage-retrieval pair-clustering model (Riefer & Batchelder, 1988)
data(retroact)

spec <- mptspec(
  c*r,
  (1 - c)*u^2,
  2*(1 - c)*u*(1 - u),
  c*(1 - r) + (1 - c)*(1 - u)^2,
  u,
  1 - u
)
m <- mpt(spec, retroact[retroact$lists == 0, ])

summary(m)  # parameter estimates, goodness of fit
plot(m)     # residuals versus predicted values
confint(m)  # approximate confidence intervals

plot(coef(m), axes = FALSE, ylim = 0:1, pch = 16, xlab = "",
     ylab = "Parameter estimate (MPT model, 95% CI)")
axis(1, 1:3, names(coef(m))); axis(2)
arrows(1:3, plogis(confint(m))[, 1], 1:3, plogis(confint(m))[, 2],
       .05, 90, 3)

## See data(package = "mpt") for application examples.
```

---

mptEM *EM Algorithm for Multinomial Processing Tree Models*

Description

Applies the EM algorithm to fit a multinomial processing tree model.

Usage

```
mptEM(theta, data, a, b, c, maxit = 1000, tolerance = 1e-8,
      stepsize = 1, verbose = FALSE)
```

Arguments

theta	a vector of starting values for the parameter estimates.
data	a vector of absolute response frequencies.
a	a three-dimensional array representing the model structure.
b	a three-dimensional array representing the model structure.

`c`	a matrix of structural constants.
`maxit`	the maximum number of iterations.
`tolerance`	the convergence criterion; the iterations converge when $logLik - logLik.old < tolerance$.
`stepsize`	the step size defaulting to 1; slightly larger values may speed up convergence, but may also give errors; use with care.
`verbose`	logical indicating if output should be produced for each iteration.

Details

Usually, `mptEM` is automatically called by `mpt`.

A prerequisite for the application of the EM algorithm is that the probabilities of the i-th branch leading to the j-th category take the form

$$p_{ij}(\Theta) = c_{ij} \prod_{s=1}^{S} \vartheta_s^{a_{ijs}} (1 - \vartheta_s)^{b_{ijs}},$$

where $\Theta = (\vartheta_s)$ is the parameter vector, a_{ijs} and b_{ijs} count the occurrences of ϑ_s and $1 - \vartheta_s$ in a branch, respectively, and c_{kj} is a nonnegative real number. The branch probabilities sum up to the total probability of a given category, $p_j = p_{1j} + \cdots + p_{Ij}$. This is the structural restriction of the class of MPT models that can be represented by binary trees. Other model types have to be suitably reparameterized for the algorithm to apply.

See Hu and Batchelder (1994) and Hu (1999) for details on the algorithm.

Value

`theta`	the vector of parameter estimates.
`loglik`	the log-likelihood at termination of the algorithm.
`pcat`	a vector of predicted probabilities for each response category.
`pbranch`	a vector of predicted branch probabilities.
`iter`	the number of iterations of the algorithm.

References

Hu, X. (1999). Multinomial processing tree models: An implementation. *Behavior Research Methods, Instruments, & Computers*, **31**(4), 689–695. doi:10.3758/BF03200747

Hu, X., & Batchelder, W.H. (1994). The statistical analysis of general processing tree models with the EM algorithm. *Psychometrika*, **59**(1), 21–47. doi:10.1007/bf02294263

Riefer, D.M., Knapp, B.R., Batchelder, W.H., Bamber, D., & Manifold, V. (2002). Cognitive psychometrics: Assessing storage and retrieval deficits in special populations with multinomial processing tree models. *Psychological Assessment*, **14**(2), 184–201. doi:10.1037/10403590.14.2.184

See Also

`mpt`.

Examples

```
# Fit storage-retrieval pair-clustering model to data in Riefer et al.
# (2002) using EM algorithm
mpt(mptspec("SR2"), c(243, 64, 58, 55), method = "EM")
```

`mptspec` *Specify a Multinomial Processing Tree (MPT) Model*

Description

Returns the specification of an MPT model object for fitting with `mpt`.

Usage

```
mptspec(..., .replicates = NULL, .restr = NULL)

## S3 method for class 'mptspec'
update(object, .replicates = NULL, .restr = NULL, ...)
```

Arguments

`...`	(named) expressions or a character string specifying the model. See Details.
`.replicates`	the number of replicates of the model equations. See Details.
`.restr`	a named list of parameter restrictions. See Details.
`object`	an object of class `mptspec`.

Details

`...` is used to symbolically specify the MPT model equations by suitable expressions, for example, they could look like this

`r + (1 - r)*b, (1 - r)*(1 - b), b, 1 - b`

where each expression represents the probability of a response in the corresponding category (link probabilities are multiplied, branch probabilities are added). Thus, there usually are as many expressions as response categories.

Joint (or product) multinomial models consist of more than a single processing tree. To identify the trees in such a model, expressions may have optional names. Canonically, these names are of the form `x.y`, where `x` is the tree identifier (`treeid`) and `y` specifies the response category within a tree.

Alternatively, `...` may be a character string identifying one out of a list of pre-specified MPT models. Currently accessible are the following models (other models have to be specified by explicit expressions as described above):

`1HT`: the one-high-threshold model (Blackwell, 1963; Swets, 1961).

`2HT`: the two-high-threshold model (Snodgrass & Corwin, 1988; see also Broeder & Schuetz, 2009).

`PairAsso`: the paired-associate learning model (Riefer & Batchelder, 1988).

proCNI: the CNI model of moral dilemma judgment for proscriptive norms (Hennig & Huetter, 2020). The general formula includes the process dissociation (PD) model (Conway & Gawronski, 2013) as a special case.

prospec: the event-based prospective memory model (Smith & Bayen, 2004).

rmodel: the r-model of recognition heuristic use (Hilbig, Erdfelder, & Pohl, 2010).

SourceMon: the source-monitoring model (Batchelder & Riefer, 1990).

SR, SR2: the storage-retrieval pair-clustering model (Batchelder & Riefer, 1986). SR2 is the model without singleton items.

WST: the inference-guessing model with relaxed assumptions (Klauer, Stahl, & Erdfelder, 2007) for the Wason selection task.

The intended use of .replicates is to specify the number of replicates of the model equations, for example, when the same model is repeatedly applied in several experimental conditions. Accordingly, parameter names are augmented by numbers to make them unique.

Parameter restrictions included in .restr may be of the form b = r or b = 0.5 etc. Depending on the fitting algorithm employed in mpt (BFGS, but not EM), mathematical functions are permissible, for example, b = sqrt(r).

The update method is used to add parameter restrictions or replicates to an existing mptspec object.

Value

An object of class mptspec that serves as input to mpt which fits the model to data. It consists of the following components:

par2prob	a function that takes a vector of parameter values and computes the response probabilities.
par2deriv	a function that takes a vector of parameter values and computes first and second derivatives of the model equations.
prob	a list containing expressions of the model equations.
deriv	a list containing expressions of the first and second derivatives of the model equations.
par	a named vector of parameter values.
replicates	the number of replicates of the model equations.
restr	a list containing expressions of parameter restrictions.
treeid	a factor that identifies each tree.

References

Batchelder, W.H., & Riefer, D.M. (1986). The statistical analysis of a model for storage and retrieval processes in human memory. *British Journal of Mathematical and Statistical Psychology*, **39**(2), 129–149. doi:10.1111/j.20448317.1986.tb00852.x

Batchelder, W.H., & Riefer, D.M. (1990). Multinomial processing models of source monitoring. *Psychological Review*, **97**(4), 548–564. doi:10.1037/0033295x.97.4.548

Blackwell, H.R. (1963). Neural theories of simple visual discriminations. *Journal of the Optical Society of America*, **53**(1), 129–160. doi:10.1364/JOSA.53.000129

Broeder, A., & Schuetz, J. (2009). Recognition ROCs are curvilinear–or are they? On premature arguments against the two-high-threshold model of recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **35**(3), 587–606. doi:10.1037/a0015279

Conway, P., & Gawronski, B. (2013). Deontological and utilitarian inclinations in moral decision making: A process dissociation approach. *Journal of Personality and Social Psychology*, **104**(2), 216–235. doi:10.1037/a0031021

Hennig, M., & Huetter, M. (2020). Revisiting the divide between deontology and utilitarianism in moral dilemma judgment: A multinomial modeling approach. *Journal of Personality and Social Psychology* **118**(1), 22–56. doi:10.1037/pspa0000173

Hilbig, B.E., Erdfelder, E., & Pohl, R.F. (2010). One-reason decision-making unveiled: A measurement model of the recognition heuristic. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **36**(1), 123–134. doi:10.1037/a0017518

Klauer, K.C., Stahl, C., & Erdfelder, E. (2007). The abstract selection task: New data and an almost comprehensive model. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **33**(4), 680–703. doi:10.1037/02787393.33.4.680

Riefer, D.M., & Batchelder, W.H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, **95**(3), 318–339. doi:10.1037/0033295x.95.3.318

Smith, R.E., & Bayen, U.J. (2004). A multinomial model of event-based prospective memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **30**(4), 756–777. doi:10.1037/02787393.30.4.756

Snodgrass, J.G., & Corwin, J. (1988). Pragmatics of measuring recognition memory: Applications to dementia and amnesia. *Journal of Experimental Psychology: General*, **117**(1), 34–50. doi:10.1037/00963445.117.1.34

Swets, J. A. (1961). Is there a sensory threshold? *Science*, **134**(3473), 168–177. doi:10.1126/science.134.3473.168

See Also

`mpt`.

Examples

```
## Specify storage-retrieval pair-clustering model for pairs
spec1 <- mptspec(
  c*r,
  (1 - c)*u^2,
  2*(1 - c)*u*(1 - u),
  c*(1 - r) + (1 - c)*(1 - u)^2
)

## Specify model with parameter restrictions
spec2 <- mptspec(
  c*r,
  (1 - c)*u^2,
  2*(1 - c)*u*(1 - u),
  c*(1 - r) + (1 - c)*(1 - u)^2,
  .restr = list(c = r/2, u = 0.3)
)
```

```
## Optional names (dot notation) identifying trees in joint MPT model
spec3 <- mptspec(
      Target.Hit  = r + (1 - r)*b,
      Target.Miss = (1 - r)*(1 - b),
  Distractor.FA   = b,
  Distractor.CR   = 1 - b,
  .replicates = 3,
  .restr = list(r1 = r, r2 = r, r3 = r)
)

## Pre-specified one-high-threshold model
spec4 <- mptspec("1HT")

## Fit to data in Broeder and Schuetz (2009)
m <- mpt(spec4, c(55, 35, 45, 765))

## Working with the mptspec object
spec4$par2prob(c(0.5, 0.1))     # response probabilities
spec4$par2deriv(coef(m))$deriv  # Jacobian matrix at ML estimate

## See data(package = "mpt") for application examples.
```

`plot.mpt` *Diagnostic Plot for MPT Models*

Description

Plots MPT residuals against fitted values.

Usage

```
## S3 method for class 'mpt'
plot(x, showNames = TRUE,
     xlab = "Predicted response probabilities", ylab = "Deviance residuals",
     ...)

## S3 method for class 'mpt'
residuals(object, type = c("deviance", "pearson"), ...)
```

Arguments

`x`, `object`	an object of class `mpt`, typically the result of a call to `mpt`.
`showNames`	logical. Should the names of the residuals be plotted? Defaults to `TRUE`.
`xlab`, `ylab`	graphical parameters passed to plot.
`type`	the type of residuals which should be returned; the alternatives are: `"deviance"` (default) and `"pearson"`.
`...`	further arguments passed to or from other methods.

Details

The deviance residuals are plotted against the predicted response probabilities. If `showNames` is true, plotting symbols are the names of the residuals.

Value

For `residuals`, a named vector of residuals having as many elements as response categories.

See Also

`mpt`, `residuals.glm`.

Examples

```
## Compare two constrained MPT models
data(proact)

spec <- mptspec(
  p1*q1*r1,
  p1*q1*(1 - r1),
  p1*(1 - q1)*r1,
  (1 - p1) + p1*(1 - q1)*(1 - r1),

  p2*q2*r2,
  p2*q2*(1 - r2),
  p2*(1 - q2)*r2,
  (1 - p2) + p2*(1 - q2)*(1 - r2),

  p3*q3*r3,
  p3*q3*(1 - r3),
  p3*(1 - q3)*r3,
  (1 - p3) + p3*(1 - q3)*(1 - r3)
)
m1 <- mpt(update(spec, .restr = list(p2=p1, p3=p1)),
          proact[proact$test == 1, ])
m2 <- mpt(update(spec, .restr = list(q2=q1, q3=q1)), m1$y)

par(mfrow = c(1, 2))                # residuals versus fitted values
plot(m1, main = "p constrained", ylim = c(-3, 3.5))  # good fit
plot(m2, main = "q constrained", ylim = c(-3, 3.5))  # bad fit

sum( resid(m1)^2 )                  # likelihood ratio G2
sum( resid(m1, "pearson")^2 )       # Pearson X2
```

`proact` *Recall Frequencies for DaPolito's Experiment on Proactive Inhibition*

Description

In DaPolito's experiment (Greeno, James, DaPolito, & Polson, 1978), 60 subjects were presented with lists of stimulus-response associates to be learned, followed by a test in which only the stimuli were presented and the responses had to be recalled. Stimuli consisted of three-letter syllables, responses of the numbers from 1 to 30, so list items looked like, say, ESI-12, JOK-3, MAL-8, etc. Part of the items had two responses (A-B, A-C), the control items had only a single correct response. If the recall of C responses is poorer than that of control items, then proactive inhibition has occurred, that is interference with the recall by information that has been learned earlier.

Riefer and Batchelder (1988) analyzed only the A-B and A-C items. They investigated how repeated A-B presentation affects the B and C recall, respectively. The responses were classified into four categories and pooled across subjects.

Usage

```
data(proact)
```

Format

A data frame consisting of five variables:

`test` first or second test.

`abpres` the number of A-B presentations.

`resp` a factor giving the response category; `BC` both B and C responses are correctly recalled, `Bc` only B is recalled, `bC` only C is recalled, `bc` neither response is recalled.

`freq` the aggregate recall frequencies per condition.

`treeid` an identifier for the single trees of the joint multinomial model.

Source

Greeno, J.G., James, C.T., DaPolito, F, & Polson, P.G. (1978). *Associative learning: A cognitive analysis*. Englewood Cliffs, NJ: Prentice-Hall.

Riefer, D.M., & Batchelder, W.H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, **95**(3), 318–339. doi:10.1037/0033295x.95.3.318

See Also

`mpt`.

Examples

```
data(proact)

## Testing effects of repeated A-B presentations
spec <- mptspec(
  .BC = p*q*r,
  .Bc = p*q*(1 - r),
  .bC = p*(1 - q)*r,
  .bc = (1 - p) + p*(1 - q)*(1 - r),
```

```
  .replicates = 6
)
m1 <- mpt(spec, proact)
m2 <- mpt(update(spec, .restr = list(q2=q1, q3=q1, q5=q4, q6=q4)), proact)
m3 <- mpt(update(spec, .restr = list(r2=r1, r3=r1, r5=r4, r6=r4)), proact)

anova(m2, m1)  # q increases with number of A-B presentations
anova(m3, m1)  # r remains constant
```

`prospecMemory` *Prospective Memory and Task Importance*

Description

Smith and Bayen (2004) tested the performance of 64 participants in an event-based prospective memory task that was embedded in a color-matching task. On each trial, participants were presented with four colored rectangles followed by a colored word. Their task was to press a key to indicate whether the color of the word matched one of the rectangles. Interspersed among these nontarget words were six target words for which subjects had to remember to press a special key (prospective memory response) regardless of the color. Participants received two different instruction types either stressing the importance of the color-matching (CMI) or of the prospective-memory task (PMI).

In a replication study, the performance of 72 German-speaking participants was tested; this study was designed to be similar to Experiment 1 in Smith and Bayen (2004).

Usage

```
data(prospecMemory)
```

Format

`PMSmithBayen` A data frame consisting of five variables:

`instruction` instruction type, either color-matching importance (`cmi`) or prospective memory importance (`pmi`).

`item` a factor specifying one of four item types: either a target word that did or did not match the color of the rectangles, or a nontarget word that did or did not match.

`resp` a factor giving the response categories: `match`, `nonmatch`, or the prospective memory response (`prospec`).

`freq` the aggregate response frequencies per condition.

`treeid` an identifier for the single trees of the joint multinomial model.

`PMreplication` A data frame containing 72 observations of five variables:

`gender` factor. Participant gender.

`age` participant age.

`instr` factor. Instruction type.

`rtdiff` average response time difference (in milliseconds) between color-matching and prospective-memory task.

`y` a matrix of aggregate response frequencies per participant. The column names indicate each of twelve response categories: match, nonmatch, prospective memory response for targets in matching (`tmm`, `tmn`, `tmp`) or in nonmatching condition (`tnm`, `tnn`, `tnp`), and again for nontargets (`nmm`, `nmn`, `nmp` vs. `nnm`, `nnn`, `nnp`).

Source

Smith, R.E., & Bayen, U.J. (2004). A multinomial model of event-based prospective memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **30**(4), 756–777. doi:10.1037/02787393.30.4.756

For the replication study, data were collected at the Department of Psychology, University of Tuebingen, between December 2018 and January 2019.

See Also

`mpt`.

Examples

```
## Prospective memory model: identifiability
qr(mptspec("prospec",
   .restr = list(M1=M, M2=M))$par2deriv(runif(6))$deriv)$rank
qr(mptspec("prospec",
   .restr = list(M1=M, M2=M, g=.1, c=.5))$par2deriv(runif(4))$deriv)$rank

## Prospective memory model: goodness of fit
data(prospecMemory)
cmi <- PMSmithBayen[PMSmithBayen$instruction == "cmi", ]
m2 <- mpt(mptspec("prospec", .restr = list(M1=M, M2=M, g=.1, c=.5)), cmi)
m1 <- mpt(update(m2$spec, .restr = list(C2=C1)), cmi)
anova(m1, m2)

pmi <- PMSmithBayen[PMSmithBayen$instruction == "pmi", ]
anova(mpt(m1$spec, pmi), mpt(m2$spec, pmi))

## Testing P_cmi = P_pmi and M_cmi = M_pmi

## Smith and Bayen
m2 <- mpt(mptspec("prospec", .replicates = 2,
                  .restr = list(M11=M1, M21=M1, g1=.1, c1=.5,
                                M12=M2, M22=M2, g2=.1, c2=.5)),
          data = PMSmithBayen)
m1 <- mpt(update(m2$spec, .restr = list(P2=P1)), PMSmithBayen)
m0 <- mpt(update(m2$spec, .restr = list(M2=M1)), PMSmithBayen)
anova(m1, m2)
anova(m0, m2)

## Replication
pm.agg <- aggregate(y ~ instr, PMreplication, sum)
```

```
y <- as.vector(t(pm.agg[2:1, -1]))

m3 <- mpt(m2$spec, y)
m1 <- mpt(update(m3$spec, .restr = list(P2=P1)), y)
m0 <- mpt(update(m3$spec, .restr = list(M2=M1)), y)
anova(m1, m3)
anova(m0, m3)

par(mfrow = 1:2)
dotchart(coef(m2)[c("C12", "C22", "P2", "M2")], xlim=0:1, xlab="",
  labels=c("C1", "C2", "P", "M"), main="Smith and Bayen (2004, Exp. 1)")
points(coef(m2)[c("C11", "C21", "P1", "M1")], 1:4, pch=16)
legend("bottomleft", c("CMI", "PMI"), pch=c(1, 16), title="Instruction",
       title.adj=1, bty="n")

dotchart(coef(m3)[c("C12", "C22", "P2", "M2")], xlim=0:1, xlab="",
  labels=c("C1", "C2", "P", "M"), main="Replication study")
points(coef(m3)[c("C11", "C21", "P1", "M1")], 1:4, pch=16)
mtext("Parameter estimate (prospective memory model)", side=1,
      line=-2, outer=TRUE)
```

`pwrsim` *Simulation-Based Power Analysis for MPT Models*

Description

The `pwrsim1` data contain the results of a power analysis of the goodness-of-fit test of the storage-retrieval pair-clustering model. Specifically, the hypothesis $a = u$ is tested for various parameter differences and sample sizes.

The `pwrsim2` data contain the results of a power analysis of a test of age differences in retrieval. Specifically, the hypothesis $r_{young} = r_{old}$ is tested for various parameter differences and sample sizes.

Simulation-based power analysis involves four steps (e.g., Wickelmaier, 2022): specifying a model that includes the effect of interest, generating data from the model, testing the null hypothesis, repeating data generation and testing. The proportion of times the test turns out significant is an estimate of its power.

Usage

```
data(pwrsim)
```

Format

`pwrsim1` A data frame consisting of 36 rows and four columns:

`d` the difference between the a parameter and the u parameter.

`n` the sample size.

`pval` 500 p values, one for each replication of the experiment.

`pwr` the proportion of significant tests.

`pwrsim2` A data frame consisting of 15 rows and four columns:

`d` the difference between the r_{young} parameter and the r_{old} parameter.

`n` the sample size.

`pval` 500 p values, one for each replication of the experiment.

`pwr` the proportion of significant tests.

References

Wickelmaier, F. (2022). Simulating the power of statistical tests: A collection of R examples. *ArXiv*. doi:10.48550/arXiv.2110.09836

See Also

`mpt`.

Examples

```
data(pwrsim)

## Power simulation 1: goodness-of-fit test, H0: a = u ------------------

s <- mptspec(
  E.1 = c * r,
  E.2 = (1 - c) * u^2,
  E.3 = 2 * (1 - c) * u * (1 - u),
  E.4 = c * (1 - r) + (1 - c) * (1 - u)^2,
  F.1 = a,
  F.2 = 1 - a
)

## Before you use par2prob(), carefully check position of parameters!
s$par
s$par2prob(c(c = 0.5, r = 0.5, u = 0.4, a = 0.6))  # evaluate model eqns

dataGen <- function(nn, d) {
  structure(list(                                      # stub mpt object
    treeid = s$treeid,
         n = setNames((nn * c(2, 1)/3)[s$treeid], s$treeid),  # 2:1 ratio
      pcat = s$par2prob(c(c = 0.5, r = 0.5,
                          u = 0.5 - d/2, a = 0.5 + d/2))
  ), class = "mpt") |>
    simulate()
}

testFun <- function(nn, d) {
  y <- dataGen(nn, d)                          # generate data with effect
  m1 <- mpt(s, y)
  m2 <- mpt(update(s, .restr = list(a = u)), y)
  anova(m2, m1)$"Pr(>Chi)"[2]                    # test H0, return p value
```

```
}

pwrsim1 <- expand.grid(d = seq(0, 0.5, 0.1), n = 30 * 2^(0:5))

## Not run:
pwrsim1$pval <-
  mapply(function(nn, d) replicate(500, testFun(nn, d)),
         nn = pwrsim1$n, d = pwrsim1$d, SIMPLIFY = FALSE)
pwrsim1$pwr <- sapply(pwrsim1$pval, function(p) mean(p < .05))

## End(Not run)

## Power simulation 2: age differences in retrieval ---------------------

s <- mptspec("SR", .replicates = 2)

dataGen <- function(nn, d) {
  structure(list(
    treeid = s$treeid,
         n = setNames((nn/2 * c(2, 1, 2, 1)/3)[s$treeid], s$treeid),
      pcat = s$par2prob(c(c1 = 0.5, r1 = 0.4 + d/2, u1 = 0.3,   # young
                          c2 = 0.5, r2 = 0.4 - d/2, u2 = 0.3))  # old
  ), class = "mpt") |>
    simulate(m)
}

testFun <- function(nn, d) {
  y <- dataGen(nn, d)
  m1 <- mpt(s, y)
  m2 <- mpt(update(s, .restr = list(r1 = r2)), y)
  anova(m2, m1)$"Pr(>Chi)"[2]
}

pwrsim2 <- expand.grid(d = seq(0, 0.4, 0.1), n = 120 * 2^(0:2))

## Not run:
pwrsim2$pval <-
  mapply(function(nn, d) replicate(500, testFun(nn, d)),
         nn = pwrsim2$n, d = pwrsim2$d, SIMPLIFY = FALSE)
pwrsim2$pwr <- sapply(pwrsim2$pval, function(p) mean(p < .05))

## End(Not run)
```

---

recogROC *Recognition Receiver Operating Characteristics*

Description

In a series of experiments, Broeder and Schuetz (2009) tested the shape of recognition receiver operating characteristics. Participants studied a list of items. In a recognition test, old items intermixed

with new ones were presented, and participants had to classify them as old or new. The percentage of old items varied in order to manipulate the response bias.

Wellingerhof (2019) conducted a replication study that was designed to be similar to Experiment 3 in Broeder and Schuetz (2009).

Usage

```
data(recogROC)
```

Format

`ROCBroeder2009` A data frame consisting of seven variables:

`item` factor. Target (old) or distractor (new) item.

`resp` a factor giving the response category, `old` or `new`.

`treeid` an identifier for the single trees of the joint multinomial model.

`ptarget1`, `ptarget3` percentage of target (old) items in Experiment 1 and 3, respectively.

`exp1`, `exp3` the aggregate response frequencies.

`ROCreplication` A data frame containing 48 observations of five variables:

`gender` factor. Participant gender.

`age` participant age.

`arith` number of mental-arithmetic problems solved.

`lexical` number of correct trials in lexical selection task.

`y` a matrix of aggregate response frequencies per participant. The column names indicate each of 4 x 5 response categories: `hit`, `miss`, `false alarm`, and `correct rejection` in the five bias conditions.

Source

Broeder, A., & Schuetz, J. (2009). Recognition ROCs are curvilinear–or are they? On premature arguments against the two-high-threshold model of recognition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **35**(3), 587–606. doi:10.1037/a0015279

Wellingerhof, P. (2019). *Signal detection theory vs. 2-high-threshold model in recognition memory: A preregistered replication study*. Bachelor thesis. University of Tuebingen, Germany. https://osf.io/hvg4p/

See Also

`mpt`.

Examples

```
data(recogROC)

## Two-high-threshold model
s <- mptspec("2HT", .replicates = 5,
                    .restr = list(r1=r, r2=r, r3=r, r4=r, r5=r,
                                  d1=d, d2=d, d3=d, d4=d, d5=d))
m1 <- mpt(s, data = ROCBroeder2009, freqvar = "exp3")
m2 <- mpt(s, data = unname(ROCreplication$y))

## Table 4
rbind(Broeder2009 = c(deviance(m1), coef(m1)),
      Replication = c(deviance(m2), coef(m2)))

## Hit rate and false alarm rate
i.hit <- with(ROCBroeder2009, item == "target" & resp == "old")
i.fa  <- with(ROCBroeder2009, item == "distractor" & resp == "old")

hrfa <- data.frame(
   study = rep(c("Broeder2009", "Replication"), each=5),
   obshr = c((m1$y/m1$n)[i.hit], (m2$y/m2$n)[i.hit]),
   obsfa = c((m1$y/m1$n)[i.fa],  (m2$y/m2$n)[i.fa]),
  predhr = c(m1$pcat[i.hit],     m2$pcat[i.hit]),
  predfa = c(m1$pcat[i.fa],      m2$pcat[i.fa])
)

## ROC, Figure 7
plot(obshr ~ obsfa, hrfa[hrfa$study == "Broeder2009", ],
     xlim=0:1, ylim=0:1, pch=16,
     main="Linear recognition ROCs?",
     ylab="Hit rate", xlab="False alarm rate")
abline(0, 1, lty=2)
lines(predhr ~ predfa, hrfa[hrfa$study == "Broeder2009", ])
points(obshr ~ obsfa, hrfa[hrfa$study == "Replication", ], col = "blue")
lines(predhr ~ predfa, hrfa[hrfa$study == "Replication", ],
      col = "blue")
text(0.45, 0.93, "Replication", col = "blue")
text(0.59, 0.82, "Broeder and Schuetz\n(2009, Exp. 3)")
```

`retroact` *Recall Frequencies in Retroactive Inhibition*

Description

Riefer and Batchelder (1988) presented each of 75 participants with either one, two, three, four, or five successive lists of words (15 subjects per group). These words were shown in random order on a computer screen, one word at a time, at a rate of 5 s per word. Each list contained 25 words, consisting of 10 categories (with 2 associate words per category) and five singletons. Subjects were given 1.5 min to recall in writing the 25 words from each individual list. After all of the lists had

been presented, a final free-recall test was given in which subjects attempted to recall the words from all of the previous lists. Subjects were given up to 5 min for this final written recall.

The focus here is on the recall of the first-list words during the final recall task. The responses were classified into six categories and pooled across subjects.

Usage

```
data(retroact)
```

Format

A data frame consisting of four variables:

`lists` the number of interpolated lists.

`treeid` an identifier for the single trees of the joint multinomial model.

`resp` a factor giving the response category; `E1` pair is recalled adjacently, `E2` pair is recalled non-adjacently, `E3` one word in a pair is recalled, `E4` neither word in a pair is recalled, `F1` recall of a singleton, `F2` non-recall of a singleton.

`freq` the aggregate recall frequencies per condition.

Source

Riefer, D.M., & Batchelder, W.H. (1988). Multinomial modeling and the measurement of cognitive processes. *Psychological Review*, **95**(3), 318–339. doi:10.1037/0033295x.95.3.318

See Also

`mpt`.

Examples

```
data(retroact)

## Fitting separate storage-retrieval pair-clustering models per condition
spec <- mptspec(
  c*r,
  (1 - c)*u^2,
  2*(1 - c)*u*(1 - u),
  c*(1 - r) + (1 - c)*(1 - u)^2,
  u,
  1 - u
)
pars <- sapply(0:4,
  function(x) coef(mpt(spec, retroact[retroact$lists == x, ])))

## Figure 3 in Riefer & Batchelder (1988)
plot(pars["c", ] ~ I(0:4), pch = 16, type = "b", ylim = c(.3, 1),
     xlab = "Number of interpolated lists, j",
     ylab = "Parameter estimate (Storage-retrieval model)",
     main = "Riefer and Batchelder (1988)")
```

```
points(pars["r", ] ~ I(0:4), type = "b", lty = 2)
text(3, .89, expression("Storage of clusters," ~ hat(c)[j]))
text(3, .46, expression("Retrieval of clusters," ~ hat(r)[j]))

## Testing effects of interpolated lists: joint models
spec <- mptspec(
  c0*r0,
  (1 - c0)*u0^2,
  2*(1 - c0)*u0*(1 - u0),
  c0*(1 - r0) + (1 - c0)*(1 - u0)^2,
  u0,
  1 - u0,

  c1*r1,
  (1 - c1)*u1^2,
  2*(1 - c1)*u1*(1 - u1),
  c1*(1 - r1) + (1 - c1)*(1 - u1)^2,
  u1,
  1 - u1,

  c2*r2,
  (1 - c2)*u2^2,
  2*(1 - c2)*u2*(1 - u2),
  c2*(1 - r2) + (1 - c2)*(1 - u2)^2,
  u2,
  1 - u2,

  c3*r3,
  (1 - c3)*u3^2,
  2*(1 - c3)*u3*(1 - u3),
  c3*(1 - r3) + (1 - c3)*(1 - u3)^2,
  u3,
  1 - u3,

  c4*r4,
  (1 - c4)*u4^2,
  2*(1 - c4)*u4*(1 - u4),
  c4*(1 - r4) + (1 - c4)*(1 - u4)^2,
  u4,
  1 - u4
)
m1 <- mpt(spec, retroact)
m2 <- mpt(update(spec, .restr = list(r0=r, r1=r, r2=r, r3=r, r4=r)),
          retroact)
m3 <- mpt(update(spec, .restr = list(c0=c, c1=c, c2=c, c3=c, c4=c)),
          retroact)

anova(m2, m1)  # r decreases the more lists have been interpolated
anova(m3, m1)  # c remains constant
```

`selectiontask` *Wason Selection Task (WST) and Helpful Hints*

Description

In the Wason selection task, a participant is presented with four cards, each one having a letter side and a number side, e.g., A B 3 4. The task is to select the card(s) that have to be turned around in order to test the rule "If there is an A on the letter side then there is a 3 on the number side." Klauer, Stahl, and Erdfelder (2007) report a series of experiments to test their WST model using the aggregate frequencies of the 16 possible response patterns.

Bauder (2020) conducted a replication study that was designed to be similar to Experiment 1 in Klauer et al. (2007).

Usage

```
data(selectiontask)
```

Format

`WSTKlauer2007` A data frame consisting of four variables:

`group` factor. The control group (`CG`) received standard instructions, the experimental group (`EG`) got additional helpful hints.

`pattern` character. Response pattern indicating which card(s) were selected (`1`) or not selected (`0`).

`exp1`, `exp2` the aggregate response frequencies for Experiment 1 and 2, respectively.

`WSTreplication` A data frame containing 1118 observations of eight variables:

`status` factor. Was the participant excluded?

`group` factor. The experimental group.

`gender` factor. Participant gender.

`age` participant age.

`education` years of education.

`logic` factor. Familiarity with formal logic.

`time` seconds spent on the web page.

`y` a participant by response pattern indicator matrix.

Note

In the original analyses (Klauer et al., 2007), a constant of one was added to all frequencies.

Source

Klauer, K.C., Stahl, C., & Erdfelder, E. (2007). The abstract selection task: New data and an almost comprehensive model. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, **33**(4), 680–703. doi:10.1037/02787393.33.4.680

Bauder, D. (2020). *Die Modellierung der abstrakten Auswahlaufgabe von Wason - eine Replikationsstudie*. Bachelor thesis. University of Tuebingen, Germany. https://osf.io/3z7ux/

See Also

mpt.

Examples

```
data(selectiontask)

## Inference-guessing model with relaxed assumptions
s <- mptspec("WST", .replicates = 2)
m1 <- mpt(s, data = WSTKlauer2007$exp1, method = "EM")

## Inference-guessing model
m2 <- mpt(update(s, .restr = list(sf1=s1, sb1=s1, sfb1=s1,
                                  sf2=s2, sb2=s2, sfb2=s2)),
          data = m1$y, method = "EM")

## Effect of hint on i parameter (Exp. 1)
m3 <- mpt(update(m2$spec, .restr = list(i2=i1)), data = m1$y,
          method = "EM")

## Independence model
m4 <- mpt(update(m2$spec,
                 .restr = list(a1=0, c1=0, x1=0, d1=0, s1=0, i1=0,
                               a2=0, c2=0, x2=0, d2=0, s2=0, i2=0)),
          data = m1$y, method = "EM")

anova(m4, m3, m2, m1)
plogis(confint(m2))
AIC(m2)
BIC(m2)  # BIC w/number of non-redundant response categories
AIC(m2, k = log(sum(m2$y)))  # BIC w/total number of data points

## Effect of hint on c parameter (Exp. 2)
m5 <- mpt(m2$spec, data = WSTKlauer2007$exp2, method = "EM")
m6 <- mpt(update(m5$spec, .restr = list(c2=c1)), data = m5$y,
          method = "EM")
anova(m6, m5)

## Replication of Exp. 1
wst.agg <- aggregate(y ~ group, WSTreplication,
                     subset = status == "select", sum)
y <- as.vector(t(wst.agg[, -1]))

set.seed(1503)
m7 <- mpt(m2$spec, data = y, start = runif(20), method = "EM")

idx <- c("P", "p", "Q", "q", "a", "c", "x", "d", "s", "i")
par(mfrow = 1:2)
dotchart(coef(m2)[paste0(idx, 1)], xlim = 0:1, labels = idx,
         main = "Klauer et al. (2007, Exp. 1)", xlab = "")
points(coef(m2)[paste0(idx, 2)], 1:10, pch = 16)
legend(0, 11, c("standard", "hints"), pch = c(1, 16),
```

```
       title = "Instruction", bty = "n")
dotchart(coef(m7)[paste0(idx, 1)], xlim = 0:1, labels = idx,
         main = "Replication study", xlab = "")
points(coef(m7)[paste0(idx, 2)], 1:10, pch = 16)
mtext("Parameter estimate (inference-guessing model)", side = 1,
      outer = TRUE, line = -2)
```

`simulate.mpt` *Simulate Responses from MPT Models*

Description

Simulates responses from the distribution corresponding to a fitted `mpt` model object.

Usage

```
## S3 method for class 'mpt'
simulate(object, nsim, seed, pool = TRUE, ...)
```

Arguments

`object`	an object of class `mpt`, typically the result of a call to `mpt`.
`nsim`, `seed`	currently not used.
`pool`	logical, if `TRUE` (default), pooled responses (summed across respondents) are returned.
`...`	further arguments passed to or from other methods. None are used in this method.

Details

Responses are simulated by (repeatedly) applying `rmultinom` with sizes taken from the original sample and probabilities computed from the model object.

Value

A named vector of (pooled) responses. Names identify the tree from which responses were simulated.

See Also

`mpt`, `rmultinom`.

Examples

```
data(retroact)

m <- mpt(mptspec(
  c*r,
  (1 - c)*u^2,
  2*(1 - c)*u*(1 - u),
  c*(1 - r) + (1 - c)*(1 - u)^2,
  u,
  1 - u
), retroact[retroact$lists == 1, ])

simulate(m)

## Parametric bootstrap of goodness-of-fit test
LR.stat <- replicate(200, deviance(mpt(m$spec, simulate(m))))

hist(LR.stat, border="white", freq=FALSE, breaks=20,
     main="Parametric bootstrap")
curve(dchisq(x, df=1), add=TRUE)
abline(v=deviance(m), lty=2)
```

---

valence *World Valence and Source Memory for Vertical Position*

Description

Sixty-four participants studied words with positive, negative, or neutral valence displayed at the top or bottom part of a computer screen. Later, these words were presented intermixed with new words, and participants had to classify them as "top," "bottom," or "new." It was of interest if memory is improved in congruent trials, in which word valence and vertical position match (positive-top, negative-bottom), as opposed to incongruent trials.

Usage

```
data(valence)
```

Format

A data frame consisting of five components:

`id` factor. Participant ID.

`gender` factor. Participant gender.

`age` participant age.

`condition` factor. In `congruent` trials, positive words were presented at the top, negative words at the bottom, and vice versa for `incongruent` trials.

`y` a matrix of aggregate response frequencies per participant and condition. The column names indicate each of nine response categories, for example, `top.bottom` means that words were presented at the top, but participant responded "bottom."

Source

Data were collected at the Department of Psychology, University of Tuebingen, in 2010.

See Also

mpt.

Examples

```
data(valence)

## Fit source-monitoring model to subsets of data
spec <- mptspec("SourceMon", .restr=list(d1=d, d2=d))
names(spec$prob) <- colnames(valence$y)

mpt(spec, valence[valence$condition == "congruent" &
                  valence$gender == "female", "y"])
mpt(spec, valence[valence$condition == "incongruent" &
                  valence$gender == "female", "y"])

## Test the congruency effect
val.agg <- aggregate(y ~ gender + condition, valence, sum)
y <- as.vector(t(val.agg[, -(1:2)]))

spec <- mptspec("SourceMon", .replicates=4,
                .restr=list(d11=d1, d21=d1, d12=d2, d22=d2,
                            d13=d3, d23=d3, d14=d4, d24=d4))
m1 <- mpt(spec, y)
m2 <- mpt(update(spec, .restr=list(d1=d.f, d3=d.f, d2=d.m, d4=d.m)), y)
anova(m2, m1)  # better discrimination in congruent trials

## Plot parameter estimates
mat <- matrix(coef(m1), 5)
rownames(mat) <- c("D1", "d",  "g",  "b",  "D2")
mat <- mat[c("D1", "D2", "d", "b", "g"), ]
matplot(mat, type="b", axes=FALSE, ylab="MPT model parameter estimate",
        main="Word valence and source monitoring", ylim=0:1, pch=1:4)
axis(1, 1:5, rownames(mat)); axis(2)
legend("bottomleft", c("female, congruent", "male, congruent",
       "female, incongruent", "male, incongruent"), pch=1:4, bty="n")
```

---

vcov.mpt *Covariance and Information Matrix for MPT Models*

Description

Returns the covariance matrix or the Fisher information matrix of a fitted mpt model object.

Usage

```
## S3 method for class 'mpt'
vcov(object, logit = FALSE, what = c("vcov", "fisher"), ...)
```

Arguments

`object`	an object of class `mpt`, typically the result of a call to `mpt`.
`logit`	logical. Switch between logit and probability scale.
`what`	character. If `vcov` (default), the covariance matrix is returned; if `fisher`, the Fisher information matrix is returned.
`...`	further arguments passed to or from other methods. None are used in this method.

Details

If `logit` is false, the covariance matrix is based on the observed Fisher information matrix of the ML estimator on the probability scale. This is equivalent to the equations for the covariance matrix given in Hu and Batchelder (1994) and Hu (1999), although the implementation here is different.

If `logit` is true, the covariance matrix and the estimated information matrix (Elandt-Johnson, 1971) of the ML estimator on the logit scale are obtained by the multivariate delta method (Bishop, Fienberg, and Holland, 1975; Grizzle, Starmer, and Koch, 1969).

Value

A (named) square matrix.

References

Bishop, Y.M.M., Fienberg, S.E., & Holland, P.W. (1975). *Discrete multivariate analysis: Theory and practice*. Cambridge: MIT Press.

Elandt-Johnson, R. C. (1971). *Probability models and statistical methods in genetics*. New York: Wiley.

Grizzle, J.E., Starmer, C.F., & Koch, G. (1969). Analysis of categorical data by linear models. *Biometrics*, **25**(3), 489–504. doi:10.2307/2528901

Hu, X. (1999). Multinomial processing tree models: An implementation. *Behavior Research Methods, Instruments, & Computers*, **31**(4), 689–695. doi:10.3758/BF03200747

Hu, X., & Batchelder, W.H. (1994). The statistical analysis of general processing tree models with the EM algorithm. *Psychometrika*, **59**(1), 21–47. doi:10.1007/bf02294263

See Also

`mpt`.

Examples

```
data(retroact)
m <- mpt(mptspec("SR"), retroact[retroact$lists == 1, ])

vcov(m)                   # covariance matrix (probability scale)
vcov(m, logit = TRUE)     # covariance matrix (logit scale)
vcov(m, what = "fisher")  # Fisher information
```

Index