

# Package ‘sensR’

April 28, 2018

**Type** Package

**Title** Thurstonian Models for Sensory Discrimination

**Version** 1.5-1

**Date** 2018-04-25

**Imports** multcomp, MASS, numDeriv

**Suggests** ordinal, parallel, testthat (>= 0.8)

**ByteCompile** yes

**Description** Provides methods for sensory discrimination methods; duotrio, tetrad, triangle, 2-AFC, 3-AFC, A-not A, same-different, 2-AC and degree-of-difference. This enables the calculation of d-primes, standard errors of d-primes, sample size and power computations, and comparisons of different d-primes. Methods for profile likelihood confidence intervals and plotting are included.

**License** GPL-2 | GPL-3

**URL** <https://github.com/perbrock/sensR>

**BugReports** <https://github.com/perbrock/sensR/issues>

**NeedsCompilation** no

**Author** Rune Haubo Bojesen Christensen [aut],  
Per Bruun Brockhoff [aut, cre],  
Alexandra Kuznetsova [ctb],  
Sophie Birot [ctb],  
Karolina Amelia Stachlewska [ctb]

**Maintainer** Per Bruun Brockhoff <perbb@dtu.dk>

**Repository** CRAN

**Date/Publication** 2018-04-28 17:32:07 UTC

**R topics documented:**

|                  |    |
|------------------|----|
| AnotA            | 3  |
| AUC              | 5  |
| betabin          | 6  |
| clls-deprecated  | 10 |
| clm2twoAC        | 12 |
| confint.twoAC    | 14 |
| discrim          | 16 |
| discrimPwr       | 19 |
| discrimR         | 21 |
| discrimSim       | 23 |
| discrimSS        | 24 |
| dod              | 27 |
| dodControl       | 29 |
| dodPwr           | 30 |
| dodSim           | 33 |
| dod_fit          | 35 |
| dod_utils        | 36 |
| dprime_compare   | 38 |
| dprime_table     | 40 |
| dprime_test      | 41 |
| duotrio          | 43 |
| findcr           | 44 |
| hexad            | 46 |
| plot.discrim     | 47 |
| plot.samediff    | 48 |
| posthoc          | 49 |
| profile.discrim  | 51 |
| profile.samediff | 53 |
| rescale          | 54 |
| ROC              | 57 |
| samediff         | 58 |
| samediffPwr      | 59 |
| samediffSim      | 61 |
| SDT              | 62 |
| sensR-deprecated | 63 |
| summary.samediff | 63 |
| tetrad           | 65 |
| threeAFC         | 66 |
| triangle         | 68 |
| twoAC            | 69 |
| twoACPwr         | 71 |
| twoAFC           | 74 |
| twofive          | 75 |
| twofiveF         | 76 |

---

AnotA *Analysis of A-not-A tests*

---

### Description

Computation of  $d_{\text{prime}}$  and its uncertainty for the monadic A-not-A test together with the one-tailed P-value of the difference test (Fisher's Exact test).

### Usage

```
AnotA(x1, n1, x2, n2, ...)

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

## S3 method for class 'anota'
plot(x, main = TRUE, length = 1000, ...)
```

### Arguments

|        |  |
|--------|--|
| x1     | the number of (correct) A-answers on A-samples   |
| n1     | the total number of A-samples  |
| x2     | the number of A-answers on not-A-samples   |
| n2     | the number of not-A-samples  |
| object | an anota object  |
| parm   | currently not used   |
| level  | the desired confidence level   |
| x      | an anota object  |
| main   | should the plot have a main title?   |
| length | the discretization of the curves   |
| ...    | additional arguments passed to <code>glm</code> for <code>AnotA</code> ; not used for <code>confint</code> and <code>plot</code> |

### Details

The `AnotA` function uses the `glm` and `fisher.test` functions of the `stats` package. Note that all arguments have to be positive integers.

### Value

For `AnotA` an object of class `anota` (which has a `print` method). This is a list with elements

|                           |   |
|---------------------------|---|
| <code>coefficients</code> | named vector of coefficients ( $d_{\text{prime}}$ )   |
| <code>res.glm</code>      | the <code>glm</code> -object from the fitting process |

|         |   |
|---------|---|
| vcov    | variance-covariance matrix of the coefficients                                  |
| se      | named vector with standard error of the coefficients (standard error of d-prime |
| data    | a named vector with the data supplied to the function                           |
| p.value | one-sided p-value from Fisher's exact test ( <code>fisher.test</code> )         |
| test    | a string with the name of the test (A-Not A) for the print method               |
| call    | the matched call  |

For plot a figure of the distributions of sensory intensity is produced, and for confint a 2-by-2 matrix of confidence intervals is returned.

### Author(s)

Rune Haubo B Christensen and Per Bruun Brockhoff

### References

Brockhoff, P.B. and Christensen, R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

### See Also

[print.discrim](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [discrimSS](#), [findcr](#)

### Examples

```
# data: 10 of the A-samples were judged to be A
#       20 A-samples in total
#       3 of the not-A samples were judged to be A
#       20 not-A-samples in total

AnotA(10, 20, 3, 20)
(m1 <- AnotA(10, 20, 3, 20))

## plot distributions of sensory intensity:
plot(m1)

## likelihood based confidence intervals:
confint(m1)

## Extended example plotting the profile likelihood
xt <- cbind(c(3, 10), c(20 - 3, 20 - 10))
lev <- gl(2, 1)
summary(res <- glm(xt ~ lev,
                  family = binomial(link = probit)))

N <- 100
dev <- double(N)
level <- c(0.95, 0.99)
delta <- seq(1e-4, 5, length = N)
for(i in 1:N)
```

```

dev[i] <- glm(xt ~ 1 + offset(c(0, delta[i])),
             family = binomial(probit))$deviance
plot(delta, exp(-dev/2), type = "l",
      xlab = expression(delta),
      ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res)[2])^2 /
                (2 * vcov(res)[2,2])), lty = 2)
## Add confidence limits:
lim <- sapply(level, function(x)
             exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")

```

AUC

*AUC computation***Description**

This is the default AUC function for scalar d-primes, which will compute Area Under the ROC curve (ROC is an acronym for receiver operating characteristic) assuming a normal distribution for the underlying percepts.

**Usage**

```

## Default S3 method:
AUC(d, se.d, scale = 1, CI.alpha = 0.05, ...)

## S3 method for class 'anota'
AUC(d, CI.alpha = 0.05, ...)

```

**Arguments**

|          |   |
|----------|---|
| d        | a unit length vector with the value of d-prime for which AUC is to be computed or a <i>anota</i> object from the fitting of a A-not A test with <a href="#">AnotA</a> |
| scale    | a unit length vector giving the ratio of scale (ie. standard deviation) of the latent distribution for the no-class items relative to that of the yes-class items     |
| se.d     | standard error of d (d-prime). If provided, the function will compute confidence limits of value of AUC—cf. in section value.   |
| CI.alpha | the type I level of the confidence interval of AUC  |
| ...      | additional arguments passed integrate   |

**Details**

The AUC is computed using the standard normal distribution function `pnorm`.

Confidence limits are based on a normal approximation of  $d$  and not of AUC. The limits are computed, if an estimate of the standard error of  $d$  is provided. Note that the limits do not take the uncertainty in estimating the scale nor that of estimating the standard error of  $d$  into account.

A print method is implemented for objects of class AUC.

**Value**

A list with components. If `se.d` is supplied to the default method or if a `discrim` object is supplied, the object contains the latter three additional elements.

|                       |  |
|-----------------------|--|
| <code>value</code>    | the estimated value of AUC                         |
| <code>res.int</code>  | the result from the call to <code>integrate</code> |
| <code>lower</code>    | the lower confidence limit                         |
| <code>upper</code>    | the upper confidence limit                         |
| <code>CI.alpha</code> | echoes the provided <code>CI.alpha</code>          |

**Author(s)**

Rune Haubo B Christensen

**Examples**

```
## Compute AUC from d-prime and confidence interval for the AUC:
fm1 <- AnotA(8, 25, 1, 25)
AUC(d=fm1$coef, se.d=fm1$se)
## The AUC-method for AnotA-objects can be used for convenience:
AUC(fm1)
```

---

betabin

*Beta-binomial and chance-corrected beta-binomial models for over-dispersed binomial data*

---

**Description**

Fits the beta-binomial model and the chance-corrected beta-binomial model to (over-dispersed) binomial data.

**Usage**

```
betabin(data, start = c(.5,.5),
        method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                  "triangle", "hexad", "twofive", "twofiveF"),
        vcov = TRUE, corrected = TRUE, gradTol = 1e-4, ...)

## S3 method for class 'betabin'
summary(object, level = 0.95, ...)
```

**Arguments**

|           |  |
|-----------|--|
| data      | matrix or data.frame with two columns; first column contains the number of success and the second the total number of cases. The number of rows should correspond to the number of observations.   |
| start     | starting values to be used in the optimization   |
| vcov      | logical, should the variance-covariance matrix of the parameters be computed?  |
| method    | the sensory discrimination protocol for which d-prime and its standard error should be computed  |
| corrected | should the chance corrected or the standard beta binomial model be estimated?  |
| gradTol   | a warning is issued if $\max \text{gradient}  < \text{gradTol}$ , where 'gradient' is the gradient at the values at which the optimizer terminates. This is not used as a termination or convergence criterion during model fitting.   |
| object    | an object of class "betabin", i.e. the result of betabin().  |
| level     | the confidence level of the confidence intervals computed by the summary method  |
| ...       | betabin: The only recognized (hidden) argument is doFit (boolean) which by default is TRUE. When FALSE betabin returns an environment which facilitates examination of the likelihood surface via the (hidden) functions sensR:::getParBB and sensR:::setParBB. Not used in summary.betabin. |

**Details**

The beta-binomial models are parameterized in terms of  $\mu$  and  $\gamma$ , where  $\mu$  corresponds to a probability parameter and  $\gamma$  measures over-dispersion. Both parameters are restricted to the interval (0, 1). The parameters of the standard (i.e. `corrected = FALSE`) beta-binomial model refers to the mean (i.e. probability) and dispersion on the scale of the observations, i.e. on the scale where we talk of a probability of a correct answer ( $P_c$ ). The parameters of the chance corrected (i.e. `corrected = TRUE`) beta-binomial model refers to the mean and dispersion on the scale of the "probability of discrimination" ( $P_d$ ). The mean parameter ( $\mu$ ) is therefore restricted to the interval from zero to one in both models, but they have different interpretations.

The summary method use the estimate of  $\mu$  to infer the parameters of the sensory experiment;  $P_c$ ,  $P_d$  and d-prime. These are restricted to their allowed ranges, e.g.  $P_c$  is always at least as large as the guessing probability.

Confidens intervals are computed as Wald (normal-based) intervals on the  $\mu$ -scale and the confidence limits are subsequently transformed to the  $P_c$ ,  $P_d$  and d-prime scales. Confidence limits are

restricted to the allowed ranges of the parameters, for example no confidence limits will be less than zero.

Standard errors, and therefore also confidence intervals, are only available if the parameters are not at the boundary of their allowed range (parameter space). If parameters are close to the boundaries of their allowed range, standard errors, and also confidence intervals, may be misleading. The likelihood ratio tests are more accurate. More accurate confidence intervals such as profile likelihood intervals may be implemented in the future.

The summary method provides a likelihood ratio test of over-dispersion on one degree of freedom and a likelihood ratio test of association (i.e. where the null hypothesis is "no difference" and the alternative hypothesis is "any difference") on two degrees of freedom (chi-square tests). Since the gamma parameter is tested on the boundary of the parameter space, the correct degree of freedom for the first test is probably 1/2 rather than one, or somewhere in between, and the latter test is probably also on less than two degrees of freedom. Research is needed to determine the appropriate no. degrees of freedom to use in each case. The choices used here are believed to be conservative, so the stated p-values are probably a little too large.

The log-likelihood of the standard beta-binomial model is

$$\ell(\alpha, \beta; x, n) = \sum_{j=1}^N \left\{ \log \binom{n_j}{x_j} - \log \text{Beta}(\alpha, \beta) + \log \text{Beta}(\alpha + x_j, \beta - x_j + n_j) \right\}$$

and the log-likelihood of the chance corrected beta-binomial model is

$$\ell(\alpha, \beta; x, n) = \sum_{j=1}^N \left\{ C + \log \left[ \sum_{i=0}^{x_j} \binom{x_j}{i} (1 - p_g)^{n_j - x_j + i} p_g^{x_j - i} \text{Beta}(\alpha + i, n_j - x_j + \beta) \right] \right\}$$

where

$$C = \log \binom{n_j}{x_j} - \log \text{Beta}(\alpha, \beta)$$

and where  $\mu = \alpha/(\alpha + \beta)$ ,  $\gamma = 1/(\alpha + \beta + 1)$ , *Beta* is the Beta function, cf. [beta](#), *N* is the number of independent binomial observations, i.e.~the number of rows in data, and  $p_g$  is the guessing probability, `pGuess`.

The variance-covariance matrix (and standard errors) is based on the inverted Hessian at the optimum. The Hessian is obtained with the `hessian` function from the `numDeriv` package.

The gradient at the optimum is evaluated with `gradient` from the `numDeriv` package.

The bounded optimization is performed with the "L-BFGS-B" optimizer in [optim](#).

The following additional methods are implemented objects of class `betabin`: `print`, `vcov` and `logLik`.

## Value

An object of class `betabin` with elements

|                           |   |
|---------------------------|---|
| <code>coefficients</code> | named vector of coefficients  |
| <code>vcov</code>         | variance-covariance matrix of the parameter estimates if <code>vcov = TRUE</code> |
| <code>data</code>         | the data supplied to the function   |



|             |   |
|-------------|---|
| call        | the matched call  |
| logLik      | the value of the log-likelihood at the MLEs   |
| method      | the method used for the fit   |
| convergence | 0 indicates convergence. For other error messages, see <a href="#">optim</a> .            |
| message     | possible error message - see <a href="#">optim</a> for details                            |
| counts      | the number of iterations used in the optimization - see <a href="#">optim</a> for details |
| corrected   | is the chance corrected model estimated?  |
| logLikNull  | log-likelihood of the binomial model with prop = pGuess                                   |
| logLikMu    | log-likelihood of a binomial model with prop = sum(x)/sum(n)                              |

### Author(s)

Rune Haubo B Christensen

### References

Brockhoff, P.B. (2003). The statistical power of replications in difference tests. *Food Quality and Preference*, 14, pp. 405–417.

### See Also

[triangle](#), [twoAFC](#), [threeAFC](#), [duotrio](#), [tetrad](#) [twofive](#), [twofiveF](#), [hexad](#)

### Examples

```
## Create data:
x <- c(3,2,6,8,3,4,6,0,9,9,0,2,1,2,8,9,5,7)
n <- c(10,9,8,9,8,6,9,10,10,10,9,9,10,10,10,10,9,10)
dat <- data.frame(x, n)

## Chance corrected beta-binomial model:
(bb0 <- betabin(dat, method = "duotrio"))
summary(bb0)
## Un-corrected beta-binomial model:
(bb <- betabin(dat, corrected = FALSE, method = "duotrio"))
summary(bb)
vcov(bb)
logLik(bb)
AIC(bb)
coef(bb)
```

**Description**

**IMPORTANT:** This function and its methods are no longer supported. The user is advised to use `clm()` from package `ordinal` instead.

Fits a cumulative link location-scale model to an ordered response variable. When the scale part is left unspecified, the model reduces to a cumulative link model assuming a constant scale. With the default logistic link function, the model reduces to the famous *Proportional Odds Model*. With the probit link and a single two-level factor in both location and scale parts, the model is known as the *Binormal* model in the Signal Detection Theory and the Psychometric literature.

**Usage**

```
cpls(location, scale, data, weights, start, ..., subset,
      na.action, contrasts = NULL, Hess = FALSE, model = TRUE,
      method = c("logistic", "probit", "cloglog", "cauchit"))
```

**Arguments**

|                        |  |
|------------------------|--|
| <code>location</code>  | a formula expression as for regression models, of the form <code>response ~ predictors</code> . The response should be a factor (preferably an ordered factor), which will be interpreted as an ordinal response, with levels ordered as in the factor. The model must have an intercept: attempts to remove one will lead to a warning and be ignored. An offset may be used. See the documentation of <a href="#">formula</a> for other details. |
| <code>scale</code>     | a optional formula expression as for the location part, of the form <code>~ predictors</code> , ie. with an empty left hand side. If left unspecified, the model assumes a constant scale and reduces to the cumulative link model. An offset may be used. See the documentation of <a href="#">formula</a> for other details.   |
| <code>data</code>      | an optional data frame in which to interpret the variables occurring in <code>formula</code> .   |
| <code>weights</code>   | optional case weights in fitting. Default to 1.  |
| <code>start</code>     | initial values for the parameters. This is in the format <code>c(beta, theta, sigma)</code> : see the Values section.  |
| <code>...</code>       | additional arguments to be passed to <a href="#">optim</a> , most often a control argument.  |
| <code>subset</code>    | expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.  |
| <code>na.action</code> | a function to filter missing data.   |
| <code>contrasts</code> | a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.   |
| <code>Hess</code>      | logical for whether the Hessian (the observed information matrix) should be returned. Use this if you intend to call <code>summary</code> or <code>vcov</code> on the fit.   |
| <code>model</code>     | logical for whether the model matrix should be returned.   |

method            logistic or probit or complementary log-log or cauchit (corresponding to a Cauchy latent variable).

### Details

The implementation is highly inspired by `polr` in package MASS and should give compatible results, if `scale` is left unspecified.

Note that standard errors are appropriate for `tau = log sigma` and not for `sigma`, because the profile likelihood is usually more symmetric for `tau` than for `sigma`. Therefore `vcov` will give the variance-covariance matrix of the parameters with `tau` rather than `sigma` and `summary.c1ls` will report standard errors for `log sigma`. Notice also that a relevant test for `sigma` is  $H_0 : \sigma = 1$ , so the relevant test for `log sigma` is  $H_0 : \log(\sigma) = 0$ . This is reflected in the `z` value for `sigma` returned by `summary.c1ls`.

There are methods for the standard model-fitting functions, including `summary`, `vcov`, `anova`, and an `extractAIC` method.

### Value

A object of class "c1ls". This has components

|                             |   |
|-----------------------------|---|
| <code>coefficients</code>   | the coefficients of the location ( <code>beta</code> ), the intercepts ( <code>theta</code> ) and the scale ( <code>sigma</code> ). |
| <code>beta</code>           | the parameter estimates of the location part.   |
| <code>theta</code>          | the intercepts/thresholds for the class boundaries.   |
| <code>sigma</code>          | the parameter estimates of the scale part.  |
| <code>tau</code>            | parameter estimates of the scale part on the log scale; ie. <code>tau = log sigma</code> .  |
| <code>deviance</code>       | the residual deviance.  |
| <code>fitted.values</code>  | a matrix, with a column for each level of the response with the fitted probabilities.   |
| <code>fitted.case</code>    | a vector of same length as <code>response</code> , with the fitted probabilities on a case-by-case basis.                           |
| <code>lev</code>            | the names of the response levels.   |
| <code>terms.location</code> | a terms structure describing the location part.   |
| <code>terms.scale</code>    | a terms structure describing the scale part.  |
| <code>df.residual</code>    | the number of residual degrees of freedoms, calculated using the weights.   |
| <code>edf</code>            | the (effective) number of degrees of freedom used by the model  |
| <code>n, nobs</code>        | the (effective) number of observations, calculated using the weights.   |
| <code>call</code>           | the matched call.   |
| <code>method</code>         | the matched method used.  |
| <code>convergence</code>    | the convergence code returned by <code>optim</code> .   |
| <code>niter</code>          | the number of function and gradient evaluations used by <code>optim</code> .  |
| <code>Hessian</code>        | if <code>Hess</code> is true, the observed Fisher information matrix.   |
| <code>location</code>       | if <code>model</code> is true, the <code>model.frame</code> for the location part.  |
| <code>scale</code>          | if <code>model</code> is true, the <code>model.frame</code> for the scale part.   |

## References

- Agresti, A. (2002) *Categorical Data*. Second edition. Wiley.
- Christensen, R. H. B., Brockhoff, P. B. and Cleaver, G. (2008) Estimation and Inference in the A-Not A test with Sureness. Manuscript for *Food Quality and Preference*.
- Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth edition. Springer.

## See Also

[polr](#), [optim](#), [glm](#), [multinom](#).

## Examples

```
## Not run:
options(contrasts = c("contr.treatment", "contr.poly"))
## Extend example from polr in package MASS:
## Fit model from polr example:
data(housing, package = "MASS")
fm1 <- clls(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
fm1
summary(fm1)
## With probit link:
summary(update(fm1, method = "probit"))

## Allow scale to depend on Cont-variable
summary(fm2 <- update(fm1, scale =~ Cont))
anova(fm1, fm2)
## which seems to improve the fit

## End(Not run)
```

---

clm2twoAC

*Extract 2-AC coefficient table from a cumulative link model*

---

## Description

The Thurstonian model for the 2-AC protocol can be formulated as a cumulative link model (see the references). This function extracts the 2-AC model parameter estimates, standard errors, z-value and p-values from a cumulative link (mixed) model fitted with [clm](#) or [clmm](#) from package [ordinal](#).

## Usage

```
clm2twoAC(object, ...)
```

**Arguments**

object            a `clm` or `clmm` object  
 ...                not currently used.

**Value**

A data.frame with the coefficient table. The two first rows contain the estimates of tau and d.prime while the remaining rows contain optional regression variables for d.prime.

**Author(s)**

Rune Haubo B Christensen

**References**

Christensen R.H.B., Lee H-S and Brockhoff P.B. (2011). Estimation of the Thurstonian model for the 2-AC protocol. Submitted to Food Quality and Preference.

**See Also**

[twoAC](#), [twoACpwr](#)

**Examples**

```
## Example of a simple 2-AC model. First the conventional way:
twoAC(c(2, 2, 6))

## The using a cumulative link model (clm from package ordinal):
if(require(ordinal)) {
  response <- gl(3,1)
  fit.clm <- clm(response ~ 1, weights = c(2, 2, 6), link = "probit")
  clm2twoAC(fit.clm)
  ## Alternatively we could get estimates and standard errors "by hand":
  tab <- coef(summary(fit.clm))
  theta <- tab[,1]
  (tau <- (theta[2] - theta[1])/sqrt(2))
  (d.prime <- (-theta[2] - theta[1])/sqrt(2))
  VCOV <- vcov(fit.clm)
  (se.tau <- sqrt((VCOV[1,1] + VCOV[2,2] - 2*VCOV[2,1])/2))
  (se.d.prime <- sqrt((VCOV[1,1] + VCOV[2,2] + 2*VCOV[2,1])/2))

  ## Extended example with a regression model for d.prime
  ## (see the referenced paper for details):
  n.women <- c(2, 2, 6)*10
  n.men <- c(1, 2, 7)*10
  wt <- c(n.women, n.men)
  response <- gl(3,1, length = 6)
  gender <- gl(2, 3, labels = c("women", "men"))
  fm2 <- clm(response ~ gender, weights = wt, link = "probit")
  clm2twoAC(fm2)
```

}

---

|               |  |
|---------------|--|
| confint.twoAC | <i>Confidence intervals and profile likelihoods for parameters in 2AC models</i> |
|---------------|--|

---

### Description

Computes confidence intervals from the profiled likelihood and the Wald approximation in the 2AC model, or plots the profile likelihood function for d.prime.

### Usage

```
## S3 method for class 'twoAC'
confint(object, parm, level = 0.95,
        type = c("likelihood", "Wald"), ...)

## S3 method for class 'profile.twoAC'
confint(object, parm = "d.prime", level = 0.95, ...)

## S3 method for class 'twoAC'
profile(fitted, alpha = 1e-3, nSteps = 1e2, range, ...)

## S3 method for class 'profile.twoAC'
plot(x, level = c(0.95, 0.99), Log = FALSE,
     relative = TRUE, fig = TRUE, n = 1e3, ..., ylim = NULL)
```

### Arguments

|        |  |
|--------|--|
| object | a fitted <a href="#">twoAC</a> object or a profile.twoAC object.   |
| fitted | a fitted <a href="#">twoAC</a> object.   |
| x      | a profile.twoAC object.  |
| type   | the type of confidence interval required. "profile" is the most accurate.  |
| parm   | For confint.profile.twoAC: has to be "d.prime".<br>For confint.twoAC: for type = "Wald" a specification of which parameters the confidence interval is required for. Ignored for type = "profile". |
| level  | the confidence level required.   |
| alpha  | determines the range of profiling. By default the likelihood is profiled in the 99.9% Wald confidence interval region.   |
| range  | if supplied, d.prime will be profiled between min(range) and max(range). This over-rides the automatic range computation.  |
| nSteps | the number of profile steps.   |
| Log    | should the profile likelihood be plotted on the log-scale?   |

|                       |  |
|-----------------------|--|
| <code>relative</code> | should the relative or the absolute likelihood be plotted?                 |
| <code>fig</code>      | should the profile likelihood be plotted?                                  |
| <code>n</code>        | the no. points used in the spline interpolation of the profile likelihood. |
| <code>ylim</code>     | overrides default y-limits on the plot of the profile likelihood.          |
| <code>...</code>      | not currently used.  |

### Details

These `confint` methods call the appropriate profile method, then finds the confidence intervals by interpolation of the profile traces. If the profile object is already available, this should be used as the main argument rather than the fitted model object itself.

In `plot.profile.twoAC`: at least one of `Log` and `relative` arguments have to be `TRUE`.

### Value

`confint`: A matrix (or vector) with columns giving lower and upper confidence limits for each parameter. These will be labelled as  $(1-\text{level})/2$  and  $1 - (1-\text{level})/2$  in % (by default 2.5% and 97.5%). Profile likelihood confidence intervals are only available for `d.prime` and not `tau`.

`profile.twoAC`: a `data.frame` with the profile of `d.prime`.

`plot.profile.twoAC` invisibly returns the spline approximation to the profile.

### Author(s)

Rune Haubo B Christensen

### References

Christensen R.H.B., lee H-S and Brockhoff P.B. (2010). Estimation of the Thurstonian model for the 2-AC protocol. Manuscript for Food Quality and Preference.

### See Also

[profile](#) and [confint](#)

### Examples

```
(fm1 <- twoAC(c(2, 2, 6)))
confint(fm1)
confint(fm1, type = "Wald")

pr1 <- profile(fm1)
confint(pr1)

pr1 <- profile(fm1, alpha = 1e-5)
par(mfrow = c(2,2))
plot(pr1)
plot(pr1, Log = FALSE, relative = TRUE)
plot(pr1, Log = TRUE, relative = TRUE)
```

```
plot(pr1, Log = TRUE, relative = FALSE)
```

---

discrim

*Sensory discrimination analysis*


---

## Description

Computes the probability of a correct answer ( $P_c$ ), the probability of discrimination ( $P_d$ ) and  $d$ -prime, their standard errors, confidence intervals and a  $p$ -value of a difference or similarity test for one of the four common discrimination protocols.

## Usage

```
discrim(correct, total, d.prime0, pd0, conf.level = 0.95,
        method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                  "triangle", "hexad", "twofive", "twofiveF"),
        double = FALSE,
        statistic = c("exact", "likelihood", "score", "Wald"),
        test = c("difference", "similarity"), ...)
```

```
## S3 method for class 'discrim'
print(x, digits = max(3, getOption("digits")-3), ...)
```

## Arguments

|            |   |
|------------|---|
| correct    | the number of correct answers; non-negative scalar integer  |
| total      | the total number of answers (the sample size); positive scalar integer  |
| d.prime0   | The value of $d$ -prime under the null hypothesis; numerical non-zero scalar  |
| pd0        | the probability of discrimination under the null hypothesis; numerical scalar between zero and one  |
| conf.level | the confidence level for the confidence intervals   |
| method     | the discrimination protocol. Eight allowed values: "twoAFC", "threeAFC", "duotrio", "tetrad", "triangle", "twofive", "twofiveF", "hexad"              |
| double     | should the 'double' variant of the discrimination protocol be used? Logical scalar. Currently not implemented for "twofive", "twofiveF", and "hexad". |
| test       | the type of test  |
| statistic  | the statistic to be used for hypothesis testing and confidence intervals  |
| x          | an object of class "discrim"  |
| digits     | number of digits in resulting table of results  |
| ...        | not currently used  |



## Details

The degree of product difference/discrimination under the null hypothesis can be specified on *either* the d-prime scale or on the pd (proportion of discriminators) scale. This is done by using either the `d.prime0` or the `pd0` arguments. If unspecified, they default to zero and the conventional difference test of "no difference" is obtained.

For a similarity test either `d.prime0` or `pd0` have to be specified *and* a non-zero, positive value should to be given. Here, `d.prime0` or `pd0` define the limit of similarity or equivalence.

The probability under the null hypothesis is given by  $pd0 + pg * (1 - pd0)$  where `pg` is the guessing probability which is defined by the discrimination protocol given in the `method` argument.

All estimates are restricted to their allowed ranges, e.g. `Pc` is always as least as large as the guessing probability. Similarly confidence limits are also restricted to the allowed range of the parameters.

Standard errors are not defined when the parameter estimates are at the boundary of their allowed range, so these will be reported as NA in such cases.

If `double = "TRUE"`, the 'double' variants of the discrimination methods is used. For example in a double-triangle test each participant will perform two individual triangle tests and only obtain a correct answer in the double-triangle test if both of the answers to the individual triangle tests are correct. The guessing probability for the double methods are lower than in the conventional discrimination methods. If  $p_g$  is the guessing probability of the conventional discrimination method, then  $p_g^2$  is the guessing probability of the double variant of that discrimination method. All the double discrimination methods have their own psychometric functions.

The "Wald" statistic is *\*NOT\** recommended for practical use—it is included here for completeness and to allow comparisons.

For `statistic = "score"`, the confidence interval is computed from Wilson's score interval, and the p-value for the hypothesis test is based on Pearson's chi-square test, cf. [prop.test](#).

## Value

An object of class `discrim` with elements

|                           |   |
|---------------------------|---|
| <code>coefficients</code> | matrix of estimates, standard errors and confidence intervals                                       |
| <code>data</code>         | a named vector with the data supplied to the function   |
| <code>p.value</code>      | the p-value of the hypothesis test  |
| <code>call</code>         | the matched call  |
| <code>test</code>         | the type of test  |
| <code>method</code>       | the discrimination protocol   |
| <code>double</code>       | logical scalar; TRUE if a double discrimination method is used, otherwise FALSE                     |
| <code>statistic</code>    | the statistic used for confidence intervals and p-value   |
| <code>pd0</code>          | the probability of discrimination under the null hypothesis   |
| <code>alt.scale</code>    | the scale for the alternative hypothesis, e.g.~"d.prime" or "pd"                                    |
| <code>conf.level</code>   | the confidence level  |
| <code>stat.value</code>   | for <code>statistic != "exact"</code> the value of the test statistic used to calculate the p-value |

df for statistic == "score" the number of degrees of freedom used for the Pearson chi-square test to calculate the p-value

profile for statistic == "likelihood" the profile likelihood on the scale of Pc

### Author(s)

Rune Haubo B Christensen and Per Bruun Brockhoff

### References

Brockhoff, P.B. and Christensen, R.H.B (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

Bi, J. (2001) The double discrimination methods. *Food Quality and Preference*, 12, pp. 507-513.

### See Also

[discrimPwr](#), [discrimSim](#), [discrimSS](#), [samediff](#), [AnotA](#), [findcr](#), [profile](#), [plot.profile](#) [confint](#)

Link functions / discrimination protocols: [triangle](#), [twoAFC](#), [threeAFC](#), [duotrio](#), [tetrad](#), [twofive](#), [twofiveF](#), [hexad](#),

### Examples

```
## Running the simple discrimination (difference) tests:
discrim(10, 15, method = "twoAFC")
discrim(10, 15, method = "threeAFC", statistic = "likelihood")
discrim(10, 15, method = "tetrad", statistic = "likelihood")
discrim(10, 15, method = "duotrio", conf.level = 0.90)
discrim(10, 15, method = "triangle", statistic = "score")

# Example of double duotrio discrimination test from Bi (2001):
discrim(35, 100, method = "duotrio", double=TRUE, statistic = "exact")
# Critical value for a sample size of 100 and a guessing probability of 1/4:
findcr(100, p0=1/4) # 33

## plot the distributions of sensory intensity:
m1 <- discrim(10, 15, method = "twoAFC")
plot(m1)

## A similarity test where less than chance successes are obtained:
discrim(22, 75, method = "triangle", d.prime0 = 1, test = "similarity")
```

---

discrimPwr                      *Sensory discrimination power analysis*

---

### Description

Computes the power of a difference or similarity test for a sensory discrimination experiment using the binomial distribution. `d.primePwr` is a convenience function that calls `discrimPwr` but has arguments in terms of `d-prime` rather than `pd`, the probability of discrimination.

### Usage

```
discrimPwr(pdA, pd0 = 0, sample.size, alpha = 0.05, pGuess = 1/2,
           test = c("difference", "similarity"),
           statistic = c("exact", "normal", "cont.normal"))
```

```
d.primePwr(d.primeA, d.prime0 = 0, sample.size, alpha = 0.05,
           method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                    "triangle", "hexad", "twofive", "twofiveF"),
           double = FALSE,
           test = c("difference", "similarity"),
           statistic = c("exact", "normal", "cont.normal"))
```

### Arguments

|                          |   |
|--------------------------|---|
| <code>pdA</code>         | the probability of discrimination for the model under the alternative hypothesis; scalar between zero and one   |
| <code>d.primeA</code>    | <code>d-prime</code> for the model under the alternative hypothesis; non-negative numerical scalar  |
| <code>pd0</code>         | the probability of discrimination under the null hypothesis; scalar between zero and one  |
| <code>d.prime0</code>    | <code>d-prime</code> under the null hypothesis; non-negative numerical scalar   |
| <code>sample.size</code> | the sample size; a scalar positive integer  |
| <code>alpha</code>       | the type I level of the test; scalar between zero and one   |
| <code>method</code>      | the discrimination protocol for which the power should be computed  |
| <code>double</code>      | should the 'double' variant of the discrimination protocol be used? Logical scalar. Currently not implemented for "twofive", "twofiveF", and "hexad".   |
| <code>pGuess</code>      | the guessing probability for the discrimination protocol, e.g. 1/2 for duo-trio and 2-AFC, 1/3 for triangle, tetrad and 3-AFC, 1/10 for two-out-of-five and hexad and 2/5 for two-out-of-five with forgiveness; scalar between zero and one |
| <code>test</code>        | the type of one-sided binomial test (direction of the alternative hypothesis): "difference" corresponds "greater" and "similarity" corresponds to "less"  |
| <code>statistic</code>   | should power determination be based on the 'exact' binomial test, the normal approximation to this, or the normal approximation with continuity correction?   |

**Details**

The power of the standard one-tailed difference test where the null hypothesis is "no difference" is obtained with  $pd_0 = 0$ .

The probability under the null hypothesis is given by  $pd_0 + pg * (1 - pd_0)$  where  $pg$  is the guessing probability  $p_{Guess}$ . Similarly, the probability of the alternative hypothesis is given by  $pd_A + pg * (1 - pd_A)$

**Value**

The power; a numerical scalar.

**Author(s)**

Rune Haubo B Christensen and Per Bruun Brockhoff

**References**

Brockhoff, P.B. and Christensen, R.H.B (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

Bi, J. (2001) The double discrimination methods. *Food Quality and Preference*, 12, pp. 507-513.

**See Also**

[findcr](#), [discrim](#), [discrimSim](#), [AnotA](#), [discrimSS](#)

**Examples**

```
## Finding the power of a discrimination test with d-prime = 1,
## a sample of size 30 and a type I level of .05:
pd <- coef(rescale(d.prime = 1, method = "twoAFC"))$pd
discrimPwr(pd, sample.size = 30)
d.primePwr(1, sample.size = 30, method = "twoAFC")
## Obtaining the equivalent normal approximation with and without
## continuity correction:
discrimPwr(pd, sample.size = 30, statistic = "cont.normal")
discrimPwr(pd, sample.size = 30, statistic = "normal")

# Example from Bi (2001) with n=100 and 35 correct answers in a
# double duotrio test:
p1 <- 0.35
# Estimate of d-prime quoted by Bi(2001) was 1.06:
dp <- psyinv(p1, method="duotrio", double=TRUE)
# Power using normal approximation w/o continuity adjustment quoted by Bi(2001):
d.primePwr(dp, sample.size = 100, method="duotrio",
           double=TRUE, stat="normal") # 0.73
# d.primePwr(dp, sample.size = 100, method="duotrio", double=TRUE,
#           stat="cont.normal")

# Power of exact test:
d.primePwr(dp, sample.size = 100, method="duotrio",
```

```

double=TRUE, stat="exact") # 0.697

## A similarity example:
discrimPwr(pdA = 0.1, pd0 = 0.2, sample.size = 100, pGuess = 1/3,
          test = "similarity")

```

discrimR

*Replicated Thurstonian Model for discrimination analysis***Description**

The model is a synthesis of a mixture and a mixed effect model. The random effect distribution for the cluster term (often individuals) is a point mass for  $\delta = 0$  and a continuous distribution for  $\delta > 0$ .

The function fits the model and computes d-prime for an average subject, 2) the variance among subjects, 3) the "posterior" probability of a subject being a discriminator (with  $\delta > 0$ ), 4) the "posterior" expectation on the random effect (ie. the subject-specific  $\delta$ ) and 5) the probability that a randomly chosen individual is a discriminator (ie. the probability mass at  $\delta = 0$  in the random effects distribution)

Warning: This function is preliminary; see the details for further information.

**Usage**

```

discrimR(formula, data, weights, cluster, start, subset, na.action,
         contrasts = NULL, hess = FALSE, ranef = FALSE, zi = FALSE,
         method = c("duotrio", "probit", "threeAFC", "triangle",
                   "twoAFC"), ...)

```

**Arguments**

|           |  |
|-----------|--|
| formula   | A formula where the lhs is the binomial response. An indicator vector or a matrix with two column; successes and failures like in a call to <code>glm</code> with a binomial family. The rhs should be 1; no other predictors are currently allowed, but extending this is ongoing work. |
| data      | The data.frame in which to look for variables.   |
| weights   | Possible weights   |
| cluster   | The clustering variable; should be a factor.   |
| start     | Optional starting values; recommended in the current implementation  |
| subset    | ...  |
| na.action | ...  |
| contrasts | ...  |
| hess      | Should the hessian of the parameters be computed?  |

|        |  |
|--------|--|
| ranef  | Should the random effect estimates be computed?  |
| zi     | Should the posterior probabilities of a subject being a discriminator be computed?   |
| method | Should correspond to the actual test applied.  |
| ...    | Additional arguments to <code>optim</code> . <code>control=list(trace=TRUE, REPORT=1)</code> is recommended, so the reduction in deviance and convergence can be followed. |

### Details

This function is preliminary and improving it is ongoing work. The computational methods are expected to change completely. This will hopefully facilitate methods for more general rhs-formulae with additional predictors.

Currently no methods or extractor functions have been written, so the user will have to select the relevant elements from the fitted object (see below). Implementation of methods and extractor functions will occur in due course.

### Value

A list with the following elements:

|             |  |
|-------------|--|
| fpar        | The fixed effect parameter, ie. delta (for an average individual)  |
| rpar        | A vector with two elements: The first element is the variance component (standard deviation) on the log-scale, where optimization is performed. The second element is the variance component (standard deviation) on the original scale. |
| deviance    | Deviance for the model   |
| se          | standard errors for 1) the fixed effect parameter and 2) the variance component on the log-scale   |
| convergence | Convergence message from <code>optim</code>  |
| lli         | Log-likelihood contributions from each of the observations.  |
| ranef       | The random effect estimates for the levels of the clustering factor (often individual)   |
| zi          | posterior probabilities of a subject being a discriminator   |
| p           | The probability that a randomly chosen individual is a discriminator (ie. the probability mass for $\delta > 0$ in the random effects distribution)  |
| fitted      | Fitted values  |
| Y           | The scaled response vector on which optimization is performed.   |
| call        | the matched call   |

### Author(s)

Rune Haubo B Christensen

### See Also

[triangle](#), [twoAFC](#), [threeAFC](#), [duotrio](#), [discrimPwr](#), [discrimSim](#), [discrimSS](#), [samediff](#), [AnotA](#), [findcr](#)

**Examples**

```
## Not run:
freq <- c(10,8,10,9,8,9,9,1,10,10,8,2,6,7,6,7,6,4,5,5,3,3,9,9,5,5,8,8,9,9)
tmp <- data.frame(id = factor(1:30), n = rep(10, 30), freq = freq)
head(tmp)
str(tmp)

fm <- discrimR(cbind(freq, n - freq) ~ 1, tmp, cluster = id,
               start = c(.5, .5), method = "twoAFC",
               ranef = TRUE, zi = TRUE, hess = TRUE,
               control=list(trace=TRUE, REPORT=1))

names(fm)
fm[1:4]

## End(Not run)
```

discrimSim

*Simulates replicated difference tests***Description**

Simulates the outcome of sample . size replicated sensory difference tests (for any one of eight protocols: 2-AFC, 3-AFC, duotrio, tetrad, triangle, two-out-of-five, two-out-of-five with forgiveness and hexad tests) for a given d-prime value and a given overdispersion (default 0).

**Usage**

```
discrimSim(sample.size, replicates, d.prime, sd.indiv = 0,
            method = c("duotrio", "halfprobit", "probit", "tetrad",
                       "triangle", "twoAFC", "threeAFC", "hexad", "twofive", "twofiveF"),
            double = FALSE)
```

**Arguments**

|             |   |
|-------------|---|
| sample.size | the sample size - number of subjects  |
| replicates  | number of replications per subject  |
| d.prime     | the value of d-prime  |
| method      | the discrimination protocol   |
| sd.indiv    | the individual variability in d-prime values. A value of 0 (default) corresponds to complete independence   |
| double      | should the 'double' variant of the discrimination protocol be used? Logical scalar. Currently not implemented for "twofive", "twofiveF", and "hexad". |

**Details**

The d-prime for each subject is a random draw from a normal distribution with mean d.prime and standard deviation sd.indiv. All negative values are set to zero.

**Value**

A vector of length `sample.size` with the number of correct answers for each subject.

**Author(s)**

Rune Haubo B Christensen and Per Bruun Brockhoff

**References**

Brockhoff, P.B. and Christensen, R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

**See Also**

[triangle](#), [twoAFC](#), [threeAFC](#), [duotrio](#), [tetrad](#), [twofive](#), [twofiveF](#), [hexad](#), [discrimPwr](#), [discrim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Running simulations:
discrimSim(sample.size = 10, replicates = 3, d.prime = 2,
           method = "triangle", sd.indiv = 1)
```

---

discrimSS

*Sensory discrimination sample size calculation*

---

**Description**

Computes the sample size for a difference or similarity test for a sensory discrimination experiment using the binomial distribution. `d.primeSS` is a convenience function that calls `discrimSS` but has arguments in terms of `d-prime` rather than `pd`, the expected proportion of discriminators.

**Usage**

```
discrimSS(pdA, pd0 = 0, target.power = 0.90, alpha = 0.05,
          pGuess = 1/2, test = c("difference", "similarity"),
          statistic = c("exact", "stable.exact", "both.exact",
                       "normal", "cont.normal"))

d.primeSS(d.primeA, d.prime0 = 0, target.power = 0.90, alpha = 0.05,
          method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                    "triangle", "hexad", "twofive", "twofiveF"),
          double = FALSE,
          test = c("difference", "similarity"),
          statistic = c("exact", "stable.exact", "both.exact",
                       "normal", "cont.normal"))
```



**Arguments**

|                           |  |
|---------------------------|--|
| <code>pdA</code>          | the probability of discrimination for the model under the alternative hypothesis; scalar between zero and one  |
| <code>d.primeA</code>     | d-prime for the model under the alternative hypothesis; non-negative numerical scalar  |
| <code>pd0</code>          | the probability of discrimination under the null hypothesis; scalar between zero and one   |
| <code>d.prime0</code>     | d-prime under the null hypothesis; non-negative numerical scalar   |
| <code>target.power</code> | the desired power for the test   |
| <code>alpha</code>        | the type I level of the test; scalar between zero and one  |
| <code>method</code>       | the discrimination protocol for which the sample size should be computed   |
| <code>double</code>       | should the 'double' variant of the discrimination protocol be used? Logical scalar. Currently not implemented for "twofive", "twofiveF", and "hexad".  |
| <code>pGuess</code>       | the guessing probability for the discrimination protocol, e.g. 1/2 for duo-trio and 2-AFC, 1/3 for triangle, tetrad and 3-AFC, 1/10 for two-out-of-five and hexad and 2/5 for two-out-of-five with forgiveness;; scalar between zero and one |
| <code>test</code>         | the type of one-sided binomial test (direction of the alternative hypothesis): "difference" corresponds "greater" and "similarity" corresponds to "less"   |
| <code>statistic</code>    | options are explained in the Details section below   |

**Details**

For difference tests `pdA` or `d.primeA` (the sensory difference under the alternative hypothesis) has to be larger than `pd0` or `d.prime0` (the sensory difference under the null hypothesis). The sample size of the standard one-tailed difference test where the null hypothesis of "no difference" is obtained with  $pd0 = 0$  or  $d.prime0 = 0$ .

For similarity tests it is required that  $pd0 > pdA$  or equivalently that  $d.prime0 > d.primeA$ . Here, the interval  $[0, pdA]$  or  $[0, d.primeA]$  is the similarity region covering sensory differences for which we would say that the products are similar.

The probability of a correct answer under the null hypothesis is given by  $pd0 + pGuess * (1 - pd0)$ .

Similarly, the probability of a correct answer under the alternative hypothesis is given by  $pdA + pGuess * (1 - pdA)$ .

The `statistic` argument:

- "exact" is the conventional sample size for the exact binomial test: The smallest sample size that gives the desired power (`target.power`) at the given significance level. Usually slightly higher sample sizes will not have the desired power, however. This is due to the non-monotonic behavior of power as a function of sample size.
- "stable.exact" is so-called stable exact sample size proposed by Ennis and Jesionka (2011) which has the property that no larger sample sizes has a power less than the `target.power`.
- "both.exact" returns both exact and stable.exact sample sizes
- "normal" is the normal approximation to the exact binomial sample size without any continuity adjustment. This usually provides a sample size that is smaller than the sample size for the exact binomial test.

- "cont.normal" is the continuity adjusted normal approximation to the sample size for the exact binomial test. This sample size is usually closer to the exact sample size than the unadjusted approximation and usually higher than the unadjusted approximation.

If the sample size based on the continuity adjusted normal approximation is larger than 10,000, the function returns the normal approximation and issues a warning.

### Value

The sample size; an integer.

### Author(s)

Per Bruun Brockhoff and Rune Haubo B Christensen

### References

Brockhoff, P.B. and Christensen, R.H.B (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

Ennis, J.M. and V. Jesionka (2011). The power of sensory discrimination methods revisited. *Journal of Sensory Studies*, 26, pp. 371-382.

### See Also

[AnotA](#), [discrimPwr](#), [samediff](#), [findcr](#)

### Examples

```
## Finding the smallest necessary sample size:
discrimSS(pdA = 0.5, pd0 = 0, target.power = 0.80, alpha = 0.05,
  pGuess = 1/2, test = "difference", statistic = "exact")
## The stable-exact sample size is larger:
discrimSS(pdA = 0.5, pd0 = 0, target.power = 0.80, alpha = 0.05,
  pGuess = 1/2, test = "difference", statistic = "stable.exact")

## Give identical results:
pd <- coef(rescale(d.prime = 1, method = "twoAFC"))$pd
discrimSS(pdA = pd, pd0 = 0, target.power = 0.90, alpha = 0.05,
  pGuess = 1/2, test = "difference", statistic = "exact")
d.primeSS(1, target.power = 0.90, method = "twoAFC")

## A similarity example:
discrimSS(pdA = 0.1, pd0 = 0.2, target.power = 0.80, alpha = 0.05,
  pGuess = 1/2, test = "similarity", statistic = "exact")
```

---

dod *Thurstonian Degree-of-Difference (DOD) model*

---

### Description

Fits the Thurstonian Degree-of-Difference (DOD) model and performs hypothesis/significance tests of d-prime (Thurstonian delta). One-sided difference and similarity tests as well as two-sided tests of d-prime are available. The user may choose from a number of tests statistics.

### Usage

```
dod(same, diff, d.prime0 = 0, conf.level = 0.95,
    statistic = c("likelihood", "Pearson", "Wilcoxon", "Wald"),
    alternative = c("difference", "similarity", "two.sided",
                  "less", "greater"), control=dodControl(), ...)
```

```
## S3 method for class 'dod'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

### Arguments

|             |  |
|-------------|--|
| same        | the answers to same-pairs; either 1) a numeric vector of counts of length equal to the number of response categories ordered appropriately or 2) a factor where the levels indicate the response categories. |
| diff        | the answers to different-pairs in the same format as same.   |
| d.prime0    | the value of d.prime under the null hypothesis. In the standard no-difference test d.prime0 = 0, while it has to be positive for similarity tests and two-sided tests.                                       |
| conf.level  | the confidence level for the confidence intervals  |
| statistic   | the statistic to be used for hypothesis testing  |
| alternative | the nature of the alternative hypothesis in the hypothesis/significance test for d-prime. Note that "greater" is an alias for "difference" and "less" is an alias for "similarity"                           |
| control     | options to control the fitting process specified via a call to <a href="#">dodControl</a> .  |
| x           | an object of class "dod".  |
| digits      | number of digits in resulting table of results.  |
| ...         | not currently used.  |

### Details

dod will report the likelihood based confidence interval for d.prime unless `statistic = "Wald"` in which case the standard symmetric Wald type confidence interval is reported. This interval can be highly inaccurate and so is not recommended for practical use.

The p-value for the standard one-tailed difference test of "no difference" is obtained with `d.prime0 = 0` corresponding to the default setting.

The standard error of d-prime is not defined when the parameter estimate is zero (or numerically close) and it will be reported as NA in this case.

The "Wald" statistic is *\*NOT\** recommended for practical use—it is only included here for completeness and to allow comparisons with other software etc.

### Value

An object of class `dod`.

### Author(s)

Rune Haubo B Christensen

### References

Ennis, J.M. and R.H.B. Christensen (2014) A Thurstonian comparison of the tetrad and degree of difference tests. *Food Quality and Preference*, in press.

Christensen, R.H.B, J.M. Ennis, D.M. Ennis and P.B Brockhoff (2012) A Thurstonian model for the Degree of Difference test with extensions to unequal variance, sequence effects and replicated data. Talk at Sensometrics conference, Rennes, France, July 11th.

### See Also

[dodSim](#), [dodPwr](#), [dodControl](#), [dod\\_fit](#), [optimal\\_tau](#)

### Examples

```
## DOD example data:
same.pairs <- c(25, 22, 33, 20)
diff.pairs <- c(18, 22, 30, 30)

## Fit Thurstonian dod-model and perform difference test:
dod(same=same.pairs, diff=diff.pairs)

## Can choose another test statistic (e.g. Wilcoxon):
dod(same=same.pairs, diff=diff.pairs, statistic="Wilcox")

## A similarity test (with simulated data):
set.seed(121)
(Data2 <- dodSim(d.prime=0, ncat=4, sample.size=200, method.tau="equi.prob"))
dod(same=Data2[1, ], diff=Data2[2, ], d.prime0=1.2,
    alternative="similarity")

## Extract parameters from a dod fit:
fm <- dod(same=same.pairs, diff=diff.pairs)
coef(fm)
```

---

|            |  |
|------------|--|
| dodControl | <i>Control settings for the dod function</i> |
|------------|--|

---

### Description

Specify control setting when fitting the the Thurstonian Degree-of-Difference (DOD) model using [dod](#) and [dod\\_fit](#).

### Usage

```
dodControl(grad.tol = 1e-4,  
           integer.tol = 1e-8,  
           get.vcov = TRUE,  
           get.grad = TRUE,  
           test.args = TRUE,  
           do.warn=TRUE,  
           optCtrl=list())
```

### Arguments

|                          |   |
|--------------------------|---|
| <code>grad.tol</code>    | tolerance for the maximum absolute gradient of the parameters are convergence.                |
| <code>integer.tol</code> | tolerance for when to give a warning about non-integer counts in data.                        |
| <code>get.vcov</code>    | compute the variance-covariance matrix of the parameters (and the standard error of d-prime)? |
| <code>get.grad</code>    | compute the gradient of the parameters?   |
| <code>test.args</code>   | test admissibility of arguments to <a href="#">dod</a> and <a href="#">dod_fit</a> ?          |
| <code>do.warn</code>     | if FALSE warnings from the fitting process are suppressed.                                    |
| <code>optCtrl</code>     | control parameters passed on to the <a href="#">nlminb</a> optimizer.                         |

### Value

An list of class `dodControl` with the appropriate control settings for [dod](#) and [dod\\_fit](#).

### Author(s)

Rune Haubo B Christensen

### See Also

[dod](#), [dod\\_fit](#), [dodSim](#), [dodPwr](#), [optimal\\_tau](#),

## Examples

```
## DOD example data:
same.pairs <- c(25, 22, 33, 20)
diff.pairs <- c(18, 22, 30, 30)

## Fit Thurstonian dod-model and perform difference test:
dod(same=same.pairs, diff=diff.pairs)

## Display the fitting process using the trace argument to nlminb:
ctrl <- dodControl(optCtrl=list(trace=TRUE))
dod(same=same.pairs, diff=diff.pairs, control=ctrl)
```

---

dodPwr

*Power of the Degree-of-Difference (DOD) method*


---

## Description

Computes the power of the Degree-of-Difference (DOD) method by simulation

## Usage

```
dodPwr(d.primeA, d.prime0=0, ncat = 4, sample.size, nsim = 1e3,
       alpha = 0.05,
       method.tau=c("LR.max", "equi.prob", "se.min", "user.defined"),
       statistic=c("likelihood", "Wilcoxon", "Pearson", "Wald"),
       alternative = c("difference", "similarity", "two.sided",
                     "less", "greater"),
       tau=NULL, ...)
```

## Arguments

|             |  |
|-------------|--|
| d.primeA    | the value of d-prime under the alternative hypothesis; non-negative numerical scalar.  |
| d.prime0    | the value of d-prime under the null hypothesis.  |
| ncat        | the number of response categories in the DOD model   |
| sample.size | the sample size in each simulation for each of the same-pairs and different pairs. Can be a single scalar value or a 2-vector. |
| nsim        | the number of simulations.   |
| alpha       | the significance level.  |
| method.tau  | the method with which to choose the boundary parameters - see <a href="#">dodSim</a> for details on the methods.               |

|             |   |
|-------------|---|
| statistic   | the statistic to be used for hypothesis testing.  |
| alternative | the nature of the alternative hypothesis in the hypothesis/significance test for d-prime. Note that "greater" is an alias for "difference" and "less" is an alias for "similarity". |
| tau         | if method.tau = "user.defined" a vector of boundary parameters in the DOD model, otherwise not used.  |
| ...         | parsed on to <a href="#">wilcox.test</a> when appropriate.  |

### Value

The simulation based estimate of the power with the following attributes:

|           |  |
|-----------|--|
| se(power) | the estimated standard error of the estimated power. This is based on the formula $\sqrt{\text{pow} * (1 - \text{pow}) / n}$ , where pow is the estimated power and n is the number of simulations used to estimate the power. |
| n.used    | the number of simulations used to estimate the power. This is usually equal to nsim, but can sometimes be smaller than nsim due to non-convergences to which the Wald test is especially prone.                                |

### Author(s)

Rune Haubo B Christensen

### References

Ennis, J.M. and R.H.B. Christensen (2014) A Thurstonian comparison of the tetrad and degree of difference tests. *Food Quality and Preference*, in press.

### See Also

[dod](#), [dod\\_fit](#), [dodSim](#), [optimal\\_tau](#), [dodControl](#)

### Examples

```
## NOTE: The number of simulations (nsim) is set unrealistically low in
## the examples below to reduce the computation time for automatic
## package checks. nsim between 1e3 and 1e4 is usually sufficient and
## the latter often on the safe side. The standard error of the
## estimated power ('se(power)') reported by dodPwr() measures the
## accuracy of the estimated power and indicates if nsim needs to be
## increased.

## Estimate power of the conventional difference test (no-difference
## under the null hypothesis):
set.seed(125)
dodPwr(d.primeA=1, d.prime0=0, ncat=4, sample.size=100, nsim=50,
       alpha=.05, method.tau="LR.max", statistic="likelihood")
## [1] 0.62
## attr(,"se(power)")
```

```

## [1] 0.1825346
## attr(,"n.used")
## [1] 50
## Here the boundary parameters are chosen automatically so as to
## maximize the likelihood ratio test statistic, and so this setting
## amounts to a highest achievable power scenario given d-prime = 1.

## Using another (and faster) statistic:
dodPwr(d.primeA=1, d.prime0=0, ncat=4, sample.size=100, nsim=1e3,
       alpha=.05, method.tau="LR.max", statistic="Wilcox")

## Not run:
## Not automatically run to reduce computation time.

## Power of a similarity test:
set.seed(127)
dodPwr(d.primeA=0, d.prime0=1, ncat=4, sample.size=100, nsim=1e2,
       alpha=.05, method.tau="LR.max", statistic="Pearson",
       alternative="similarity")
## [1] 0.71
## attr(,"se(power)")
## [1] 0.1434922
## attr(,"n.used")
## [1] 100

## Same as above, but with a given set of boundary parameters:
dodPwr(d.primeA=0, d.prime0=1, sample.size=100, nsim=1e2,
       alpha=.05, method.tau="user.defined", statistic="Pearson",
       alternative="similarity", tau=1:3)

## Using parallel computing to speed up computations:
if(require(parallel)) {
  ## Use detectCores() to get an appropriate number of cores for
  ## practical use - for the example here we fix it at 2:
  ## cl <- makeCluster(detectCores())
  cl <- makeCluster(getOption("cl.cores", 2))
  dvec <- c(0, .2, .5, .7, 1, 1.2, 1.5, 1.75)
  system.time(
    res <- parLapply(cl, dvec, fun=function(dp) {
      library(sensR)
      x <- dodPwr(dp, 0, sample.size=100, nsim=1e4, stat="Wil")
      c("power"=x, "se"=attr(x, "se(power)"))
    })
  )
  stopCluster(cl)
  names(res) <- dvec
  mat <- do.call(cbind, res)
  round(mat[1:2, ], 3)
  ## Example output:
  ##           0  0.2  0.5  0.7  1  1.5  1.75  2
  ## power 0.051 0.058 0.123 0.238 0.578 0.983 1.000 1.000
  ## se     0.022 0.023 0.033 0.043 0.049 0.013 0.002 0.001
}

```



```
## End(Not run)
## Realistically one should use more simulations, e.g. nsim=1e4.
```

---

dodSim

*Simulate data from the Degree-of-Difference model*


---

### Description

Simulate data from the Degree-of-Difference model for a given value of d-prime. The boundary parameters can either be specified by the user, or be chosen automatically so as to 1) maximize the likelihood ratio statistic, 2) ensure responses in each category is equally probable across same-pairs and different-pairs or 3) minimize the standard error of d-prime.

### Usage

```
dodSim(d.prime, ncat=4, sample.size = c(100, 100),
       method.tau = c("equi.prob", "LR.max", "se.min", "user.defined"),
       tau = NULL, d.prime0 = 0, ...)
```

### Arguments

|             |  |
|-------------|--|
| d.prime     | the value of d-prime.  |
| ncat        | the number of response categories.   |
| sample.size | the sample size for same-pairs and different-pairs. The sample size can be a scalar number in which case the sample sizes for both same-pairs and different-pairs are taken to equal that number.                        |
| method.tau  | the method with which to choose the boundary parameters. If "user.defined", the user has to specify the tau argument, otherwise the set of boundary parameters are chosen automatically (see the Details section below). |
| tau         | if method.tau = "user.defined" the set of boundary parameters, otherwise not used.   |
| d.prime0    | if method.tau = "LR.max" the value of d-prime under the null hypothesis, otherwise not used.   |
| ...         | passed on to <a href="#">optimal_tau</a> .   |

### Details

In principle both d-prime and all boundary parameters have to be specified in order to be able to simulate from the DOD model. However, since it can be difficult to decide which boundary parameters to use for simulation, dodSim offers ways to choose these parameters automatically according to the following three criteria:

**equi.prob** the boundary parameters are chosen such that responses in each category are equally probable across same-pairs and different-pairs.

**LR.max** the boundary parameters are chosen such that the likelihood ratio statistic for the test of d-prime is maximized. This choice maximizes the power of the likelihood ratio test and is in a sense an optimal choice of boundary parameters.

**se.min** the boundary parameters are chosen such that the standard error of d-prime is minimized. This method also maximizes the power of the Wald test of d-prime when the null hypothesis is no-difference (d-prime = 0). This method can be numerical unstable for small and large d-prime values (approximately d.prime < 0.5 and d.prime > 5).

Experience shows that the asymptotic properties of the DOD model are not too sensitive to the choice of boundary parameters: power, standard error of d-prime and confidence intervals seem to be fairly constant irrespectively which of the above three criteria are used to choose the boundary parameters.

### Value

A 2-by-ncat matrix of counts with same-pairs in the first row and different-pairs in the second row. First/last column corresponds to "same"/"different" on the response scale.

### Author(s)

Rune Haubo B Christensen

### See Also

[dod](#), [dod\\_fit](#), [dodControl](#), [optimal\\_tau](#), [dodPwr](#)

### Examples

```
## Simulate data from the DOD model with the equi.prob method:
set.seed(125)
(Data <- dodSim(d.prime=1, sample.size=100, method.tau="equi.prob"))

## Simulate data that maximizes the likelihood ratio statistic:
set.seed(125)
dodSim(d.prime=1, sample.size=100, method.tau="LR.max")

## Simulate with user-defined values for the boundary parameters:
dodSim(d.prime=1.5, sample.size=c(100, 100),
       method.tau="user.defined", tau=1:4)

## Simulate using different sample sizes for same-pairs and
## different-pairs:
dodSim(d.prime=1, ncat=3, sample.size=c(75, 125),
       method.tau="se.min")
```

---

|         |  |
|---------|--|
| dod_fit | <i>Direct fitter of the Thurstonian Degree-of-Difference (DOD) model</i> |
|---------|--|

---

### Description

Fits the Thurstonian Degree-of-Difference (DOD) model. This function is for programming use. The ordinary user probably wants the [dod](#) function, which is for interactive use. `dod_fit` only estimates the DOD model and performs no hypothesis or significance tests.

### Usage

```
dod_fit(same, diff, tau=NULL, d.prime=NULL, control=dodControl(),
        ...)
```

### Arguments

|                      |  |
|----------------------|--|
| <code>same</code>    | the answers to same-pairs; either 1) a numeric vector of counts of length equal to the number of response categories ordered appropriately or 2) a factor where the levels indicate the response categories. |
| <code>diff</code>    | the answers to different-pairs in the same format as <code>same</code> .   |
| <code>tau</code>     | optional vector of boundary parameters. If specified, <code>dod_fit</code> will not optimize over the <code>tau</code> parameters.   |
| <code>d.prime</code> | optional d-prime value. If specified, <code>dod_fit</code> will not optimize over <code>d.prime</code> .   |
| <code>control</code> | options to control the fitting process specified via a call to <a href="#">dodControl</a> .  |
| <code>...</code>     | not currently used.  |

### Details

The standard error of d-prime is not defined when the parameter estimate is zero (or numerically close) and it will be reported as NA in this case.

### Value

An object of class `dod_fit`.

### Author(s)

Rune Haubo B Christensen

### See Also

[dod](#), [dodSim](#), [optimal\\_tau](#), [dodPwr](#), [dodControl](#)

**Examples**

```
## DOD example data:
same.pairs <- c(25, 22, 33, 20)
diff.pairs <- c(18, 22, 30, 30)

## Fit Thurstonian dod-model and perform difference test:
fm <- dod_fit(same=same.pairs, diff=diff.pairs)
names(fm)

## Estimate d-prime for given tau:
fm <- dod_fit(same=same.pairs, diff=diff.pairs, tau=1:3)

## Estimate tau for given d-prime:
fm <- dod_fit(same=same.pairs, diff=diff.pairs, d.prime=1)
```

---

dod\_utils

*Utility functions for the Degree-of-Difference model*


---

**Description**

Various utility functions supporting the Degree-of-Difference (DOD) model.

**Usage**

```
optimal_tau(d.prime, d.prime0 = 0, ncat=3,
            method=c("equi.prob", "LR.max", "se.min"),
            tau.start=NULL, equi.tol = 1e-4, grad.tol = 1e-2,
            do.warn=TRUE)

par2prob_dod(tau, d.prime)

dod_nll(tau, d.prime, same, diff, integer.tol=1e-8)

dod_null(same, diff, integer.tol=1e-8)

dod_null_tau(same, diff)
```

**Arguments**

|          |  |
|----------|--|
| d.prime  | the value of d-prime; non-negative numerical scalar.   |
| d.prime0 | d-prime under the null hypothesis; only used in <code>optimal_tau</code> when <code>method = "LR.max"</code> . |
| ncat     | the number of response categories in the DOD model.  |

|             |  |
|-------------|--|
| method      | the method with which to choose the boundary parameters — see <a href="#">dodSim</a> for details on the methods.   |
| tau.start   | optional vector of starting values.  |
| equi.tol    | convergence tolerance for the "equi.prob" method.  |
| grad.tol    | gradient convergence tolerance.  |
| do.warn     | issue warning if estimation of optimal tau does not converge?  |
| same        | The answers to same-pairs; either 1) a numeric vector of counts of length equal to the number of response categories ordered appropriately or 2) a factor where the levels indicate the response categories. |
| diff        | the answers to different-pairs in the same format as same.   |
| tau         | vector of boundary parameters in the DOD model.  |
| integer.tol | tolerance for when same or diff arguments are considered non-integer counts: a warning is issued if non-integer counts are encountered.  |

### Value

|              |  |
|--------------|--|
| optimal_tau  | computes optimal boundary parameters (tau) using various criteria.                   |
| par2prob_dod | computes the multinomial probability vectors from DOD model parameters.              |
| dod_nll      | implements the negative log-likelihood function for the DOD model.                   |
| dod_null     | implements the negative log-likelihood function for the DOD model where d-prime = 0. |
| dod_null_tau | Estimates tau for the DOD model where d-prime = 0.                                   |

### Author(s)

Rune Haubo B Christensen

### See Also

[dod](#), [dod\\_fit](#), [dodSim](#), [dodPwr](#), [dodControl](#)

### Examples

```
## Compute optimal boundary parameters using the LR.max criterion for
## d.prime=1 with 4 categories:
dp <- 1
(Tau <- optimal_tau(d.prime=dp, d.prime0 = 0, ncat=4,
                    method="LR.max")$tau)
## [1] 1.244642 2.109140 3.098985
## This set of boundary parameters optimize the power of the DOD test
## with d.prime = 1 under the alternative hypothesis.

## Compute the probability that an observation will fall in each of
## the (here 2*4=8) response categories given values of tau and d.prime:
par2prob_dod(tau=Tau, d.prime=dp)
##           [,1]      [,2]      [,3]      [,4]
```

```

## p.same 0.6211921 0.2429480 0.1074307 0.02842911
## p.diff 0.5124361 0.2571691 0.1596425 0.07075227

## Compute the negative log-likelihood given data and parameters:
Same <- c(10, 20, 30, 20)
Diff <- c(10, 10, 20, 40)
dod_nll(tau=Tau, d.prime=dp, same=Same,
        diff=Diff)
## [1] 334.0986

## Compute the negative log-likelihood under the null hypothesis
## (where d.prime = 0):
dod_null(same=Same, diff=Diff)
## [1] 208.8154
## ## The boundary parameters for this:
(Tau0 <- dod_null_tau(same=Same, diff=Diff))
## [1] 0.2224709 0.5688675 1.2546147

## Some equalities:
stopifnot(
  dod_nll(tau=Tau0, d.prime=0, same=Same, diff=Diff) ==
  dod_null(same=Same, diff=Diff))
stopifnot(
  dod_null(same=Same, diff=Diff) ==
  -dod_fit(same=Same, diff=Diff, d.prime=0)$logLik
)
stopifnot(
  dod_nll(same=Same, diff=Diff, tau=Tau, d.prime=dp) ==
  -dod_fit(same=Same, diff=Diff, tau=Tau, d.prime=dp)$logLik
)
stopifnot(all(
  dod_null_tau(same=Same, diff=Diff) ==
  dod_fit(Same, Diff, d.prime=0)$tau))

```

---

dprime\_compare

*Test the 'any-differences' hypothesis and estimate common d-prime*


---

## Description

This function will test the 'any-differences' hypothesis (conceptually a one-way ANOVA test for d-primes) with one of the Wald, Pearson or likelihood ratio chi-square test statistics. The common d-prime is estimated with ML or weighted average.

## Usage

```

dprime_compare(correct, total, protocol, conf.level = 0.95,
  statistic = c("likelihood", "Pearson", "Wald.p", "Wald.d"),
  estim = c("ML", "weighted.avg"))

```

**Arguments**

|            |  |
|------------|--|
| correct    | a numeric vector of the number of correct answers; one element for each test.  |
| total      | a numeric vector of the total number of trials; one element for each test.   |
| protocol   | a character vector or factor naming the protocol used; one element for each test.<br>Currently the following protocols are supported: "triangle", "duotrio", "threeAFC", "twoAFC", " |
| conf.level | the confidence level for the estimated common d-prime.   |
| statistic  | the test statistic for testing the 'any-differences' hypothesis.   |
| estim      | The estimation method for the common d-prime.  |

**Details**

The vectors correct, total and protocol have to be of the same length.

The function has a print method.

**Value**

an object of class "dprime\_compare" with the following elements

|              |  |
|--------------|--|
| stat.value   | the value of the (chi-square) test statistic for the 'any-differences' hypothesis.   |
| df           | the degrees of freedom for the stat.value test statistic.  |
| p.value      | the p-value for the 'any-differences' test.  |
| statistic    | the name of the test statistic for the 'any-differences' test.   |
| data         | the data table produced by <a href="#">dprime_table</a> .  |
| coefficients | 'table' with estimated common d-prime, standard error and confidence limits stored as a one-row <a href="#">data.frame</a> . |
| conf.level   | confidence level for the common d-prime.   |
| conf.int     | the confidence interval for the common d-prime.  |
| estim        | the estimation method for the common d-prime.  |
| conf.method  | the statistical method/test statistic used to compute the confidence interval for the common d-prime.                        |

**Author(s)**

Rune Haubo B Christensen

**See Also**

[dprime\\_test](#), [dprime\\_table](#), [posthoc.dprime\\_compare](#).

## Examples

```
## Make some fake data:
n <- rep(40, 4)
x <- c(25, 25, 30, 35)
protocol <- c("triangle", "duotrio", "threeAFC", "twoAFC")
## Look at the data table with d-primes etc.:
dprime_table(x, n, protocol)

## 'any differences' test:
## ML estimation and test with likelihood statistic:
(dpc <- dprime_compare(x, n, protocol))
## Other estimation/statistic options:
dprime_compare(x, n, protocol, estim="weighted.avg")
dprime_compare(x, n, protocol, statistic="Pearson")
dprime_compare(x, n, protocol, statistic="Wald.p")
dprime_compare(x, n, protocol, statistic="Wald.d")
```

---

|              |  |
|--------------|--|
| dprime_table | <i>Summary table of several discrimination experiments using the simple-binomial protocols (Duo-Trio, Triangle, Tetrad, 2-AFC and 3-AFC)</i> |
|--------------|--|

---

## Description

This function provides a summary table with the following quantities: no. correct trials (correct), total number of trials (total), the protocol (protocol), probability of a correct answer (pHat), standard error of pHat (se.pHat), d-prime (dprime), and standard error of d-prime (se.dprime).

## Usage

```
dprime_table(correct, total, protocol, restrict.above.guess = TRUE)
```

## Arguments

|                      |   |
|----------------------|---|
| correct              | a numeric vector of the number of correct answers; one element for each test.   |
| total                | a numeric vector of the total number of trials; one element for each test.  |
| protocol             | a character vector or factor naming the protocol used; one element for each test. Currently the following protocols are supported: "triangle", "duotrio", "threeAFC", "twoAFC", "   |
| restrict.above.guess | controls if pHat should be restricted at or above the guessing probability for the given protocol. This also affects se.pHat. Note that dprime is zero and se.dprime is NA when pHat is at or below the guessing probability of the given protocol. |

## Details

The vectors correct, total and protocol have to be of the same length.



**Value**

a data.frame with columns:

|           |   |
|-----------|---|
| correct   | numeric vector of no. correct.                  |
| total     | numeric vector of no. trials.                   |
| protocol  | character vector naming the protocols used.     |
| pHat      | Estimate of the probability of correct answers. |
| se.pHat   | standard error of pHat.                         |
| dprime    | estimate of d-prime.                            |
| se.dprime | standard error of dprime.                       |

**Author(s)**

Rune Haubo B Christensen

**See Also**

[dprime\\_compare](#), [dprime\\_test](#), [posthoc.dprime\\_compare](#).

**Examples**

```
n <- rep(40, 4)
x <- c(25, 25, 30, 35)
protocol <- c("triangle", "duotrio", "threeAFC", "twoAFC")
dprime_table(x, n, protocol)
```

---

dprime\_test

*Test of simple hypothesis with the common d-prime*

---

**Description**

This function tests the hypothesis that the common d-prime is equal to or greater/less than a certain value, e.g. zero in a Wald or likelihood root test.

**Usage**

```
dprime_test(correct, total, protocol, conf.level = 0.95, dprime0 = 0,
  statistic = c("likelihood", "Wald"),
  alternative = c("difference", "similarity", "two.sided", "less", "greater"),
  estim = c("ML", "weighted.avg"))
```

**Arguments**

|             |   |
|-------------|---|
| correct     | a numeric vector of the number of correct answers; one element for each test.   |
| total       | a numeric vector of the total number of trials; one element for each test.  |
| protocol    | a character vector or factor naming the protocol used; one element for each test. Currently the following protocols are supported: "triangle", "duotrio", "threeAFC", "twoAFC", " |
| conf.level  | the confidence level for the confidence interval of the estimated common d-prime.   |
| dprime0     | Value of d-prime under the Null hypothesis. Non-negative numeric scalar.  |
| statistic   | the test statistic for computing the confidence interval as well as p-value.  |
| alternative | the direction of the hypothesis test. "difference" and "similarity" are just alternative ways of specifying "greater" and "less" respectively.                                    |
| estim       | The estimation method for the common d-prime.   |

**Details**

The vectors correct, total and protocol have to be of the same length.

The function has a print method.

**Value**

an object of class "dprime\_test" with the following elements

|             |   |
|-------------|---|
| p.value     | the p-value for the 'any-differences' test.   |
| alternative | character naming the direction of the hypothesis test.  |
| statistic   | the name of the test statistic.   |
| data        | the data table produced by <a href="#">dprime_table</a> .   |
| conf.level  | confidence level for the common d-prime.  |
| conf.int    | the confidence interval for the common d-prime.   |
| estim       | the estimation method for the common d-prime.   |
| conf.method | the statistical method/test statistic used to compute the confidence interval for the common d-prime. |

**Author(s)**

Rune Haubo B Christensen

**See Also**

[dprime\\_compare](#), [dprime\\_table](#), [posthoc.dprime\\_compare](#).

## Examples

```
n <- rep(40, 4)
x <- c(25, 25, 30, 35)
protocol <- c("triangle", "duotrio", "threeAFC", "twoAFC")
## Look at the data table with d-primes etc.:
dprime_table(x, n, protocol)

## Test of common d':
dprime_test(x, n, protocol)

## Another setting:
dprime_test(x, n, protocol, dprime0=2, statistic="Wald",
            alternative="less", estim="weighted.avg")
```

---

|         |                                       |
|---------|---------------------------------------|
| duotrio | <i>Create duotrio binomial family</i> |
|---------|---------------------------------------|

---

## Description

Creates a copy of the binomial family with the inverse link function changed to equal the duotrio psychometric function and correspondingly changed link function and derivative of the inverse link function.

## Usage

```
duotrio()
```

## Value

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct `dprime` computation) as `linkfun`.

## Note

Several functions in this package make use of the function, but it may also be used on its own—see the example below.

## Author(s)

Per Bruun Brockhoff

## References

Brockhoff, P.B. and Christensen, R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

**See Also**

[triangle](#), [twoAFC](#), [threeAFC](#), [tetrad](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [Anota](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Estimating d-prime using glm for a Duotrio test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = duotrio)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="duotrio")

## Extended example plotting the profile likelihood
## data: 10 correct answers, 5 incorrect
xt <- matrix(c(10, 5), ncol = 2)
summary(res <- glm(xt ~ 1, family = duotrio))
N <- 100
dev <- double(N)
delta <- seq(1e-4, 5, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = duotrio)$deviance
plot(delta, exp(-dev/2), type = "l",
     xlab = expression(delta),
     ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
level <- c(0.95, 0.99)
lim <- sapply(level, function(x)
              exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")
points(confint(res), rep(lim[1], 2), pch = 4)
```

---

findcr

*Find the critical value of a one-tailed binomial test*

---

**Description**

Finds the critical value in a one-tailed binomial test

**Usage**

```
findcr(sample.size, alpha = .05, p0 = .5, pd0 = 0,
       test = c("difference", "similarity"))
```

**Arguments**

|             |   |
|-------------|---|
| sample.size | the sample size of the binomial test (must be a positive integer)   |
| alpha       | the test I error-level of the test (must be between zero and one)   |
| p0          | the guessing probability under the null-hypothesis (must be between zero and one); 1/2 for the duotrio and twoAFC tests and 1/3 for the triangle, tetrad and threeAFC tests |
| pd0         | the proportion of discriminators in the population of interest  |
| test        | the type of test  |

**Details**

The critical value of the standard one-tailed difference test of "no difference" is obtained with  $pd0 = 0$ .

The probability of a correct answer under the null hypothesis is given by  $pd0 + p0 * (1 - pd0)$ .

**Value**

The critical value in a one-tailed binomial test, that is, the smallest integer such that the null hypothesis binomial probability of being larger (smaller for similarity hypotheses) than or equal to this number is smaller than or equal to the type I error-level of the test.

**Author(s)**

Rune Haubo B Christensen and Per Bruun Brockhoff

**See Also**

[triangle](#), [twoAFC](#), [threeAFC](#), [duotrio](#), [tetrad](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#) [discrimSS](#), [samediff](#)

**Examples**

```
## Find the critical value for a triangle test for the level 0.05 test
## with 25 subjects:
findcr(sample.size = 25, , p0 = 1/3)

## Similarity example:
findcr(sample.size = 25, p0 = 1/3, pd0 = .2, test = "simil")
```

---

|       |                                     |
|-------|-------------------------------------|
| hexad | <i>Create hexad binomial family</i> |
|-------|-------------------------------------|

---

**Description**

Creates a binomial family object with the inverse link function equal to the psychometric function for the unspecified Hexad test.

**Usage**

```
hexad()
```

**Value**

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct d-prime computation) as `linkfun`.

**Note**

Several functions in this package makes use of functions in the hexad family object, but it may also be used on its own—see the example below.

**Author(s)**

Karolina Stachlewska

**References**

Eberhardt, K., Aubry, V., & Robinson, K. (2008). A thurstonian model for the unspecified hexad test. In 2008 Sensometrics Meeting 'Discover a New World of Data' (E-5). Kraft Foods.

**See Also**

[duotrio](#), [triangle](#), [twoAFC](#), [threeAFC](#), [tetrad](#), [twofive](#), [twofiveF](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Estimating d-prime using glm for an unspecified Hexad test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = hexad)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="hexad")
```

```
## Extended example plotting the profile likelihood
## data: 10 correct answers, 9 incorrect
xt <- matrix(c(10, 9), ncol = 2)
summary(res <- glm(xt ~ 1, family = hexad))
N <- 100
dev <- double(N)
delta <- seq(1e-4, 3, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = hexad)$deviance
plot(delta, exp(-dev/2), type = "l",
      xlab = expression(delta),
      ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
level <- c(0.95, 0.99)
lim <- sapply(level, function(x) exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")
```

---

plot.discrim

*Plot function for discrim objects*


---

## Description

This function plots the latent distributions of sensory intensity corresponding to the items or products tested in the discrimination test.

## Usage

```
## S3 method for class 'discrim'
plot(x, main = TRUE, length = 1000, ...)
```

## Arguments

|        |   |
|--------|---|
| x      | The discrim object whose latent distributions are to be plotted                   |
| main   | include an automatically generated title on the plot? Default is TRUE             |
| length | the length of the vectors to be plotted. Longer vectors gives more smooth curves. |
| ...    | additional arguments to plot and lines  |

## Author(s)

Rune Haubo B Christensen

**Examples**

```
## Generate discrim objects to be plotted:
fm1 <- discrim(10, 15, method = "threeAFC")
fm2 <- discrim(10, 15, method = "triangle")
par(mfrow=c(2,1)) ## Split plotting window in two
## Plot the distributions of sensory intensity for the two objects
## and increase the line width
plot(fm1, lwd=2)
plot(fm2, lwd=2)
```

---

plot.samediff

*Plot function for samediff objects*


---

**Description**

This function plots the latent distributions of sensory intensity corresponding to the items or products tested in the discrimination test.

**Usage**

```
## S3 method for class 'samediff'
plot(x, main = TRUE, length = 1000,
      limits, fig = TRUE, ...)
```

**Arguments**

|        |   |
|--------|---|
| x      | The samediff object whose latent distributions are to be plotted  |
| main   | include an automatically generated title on the plot? Default is TRUE   |
| length | the length of the vectors to be plotted. Longer vectors gives more smooth curves, but can take a little time. |
| limits | optional limits on the x-axis; vector of length two.  |
| fig    | logical: Should the function create the plot? Defaults to TRUE.   |
| ...    | additional arguments to plot and lines  |

**Value**

If fig = TRUE, the function will produce the plot. The function invisibly returns a data.frame with elements

|            |  |
|------------|--|
| z          | values for the x-axis of length length.  |
| base.dist  | y-values for the base distribution of same-samples, ie. a standard normal distribution                             |
| delta.dist | y-values for the distribution of different-samples, ie. a normal distribution centred at delta with unit variance. |

This facilitates later plotting and changing the appearance of the plot.



**Author(s)**

Rune Haubo B Christensen

**Examples**

```
## Make same-diff object:
sadi <- samediff(8, 5, 4, 9)
## Plot distributions of sensory intensity:
plot(sadi)
```

posthoc

*Post-hoc estimates and tests for multiple discrimination experiments.***Description**

This function provides estimates and p-values for post-hoc tests such as pairwise comparisons. p-values are (by default) adjusted for multiplicity.

**Usage**

```
posthoc(x, ...)

## S3 method for class 'dprime_compare'
posthoc(x, alpha = 0.05,
  test = c("pairwise", "common", "base", "zero"), base = 1,
  alternative = c("two.sided", "less", "greater"),
  statistic = c("likelihood", "Wald"),
  padj.method = c("holm", "bonferroni", "none"), ...)

## S3 method for class 'dprime_test'
posthoc(x, alpha = 0.05,
  test = c("pairwise", "common", "base", "zero"), base = 1,
  alternative = c("two.sided", "less", "greater"),
  statistic = c("likelihood", "Wald"),
  padj.method = c("holm", "bonferroni", "none"), ...)
```

**Arguments**

|             |  |
|-------------|--|
| x           | an object of class <code>dprime_compare</code> or <code>dprime_test</code> .       |
| alpha       | the significance level for tests and confidence intervals.                         |
| test        | the type of post-hoc tests performed. See the details section for further details. |
| base        | when test = "base", the experiment against which to provide pairwise comparisons.  |
| alternative | direction of the hypothesis test.  |

|             |  |
|-------------|--|
| statistic   | The test statistic used - currently there is only partial support for statistic = "likelihood".  |
| padj.method | controls the method by which p-values are adjusted for multiplicity. Any one of the values in p.adjust.methods (currently "holm" "hochberg" "hommel" "bonferroni" "BH" "BY" "fdr" "none") may be specified, cf. <a href="#">p.adjust</a> . |
| ...         | currently not used.  |

## Details

The `test` argument specifies the type of test performed. "pairwise" performs all pairwise comparisons and produces a compact letter display indicating groups of experiments that different/not-different. "common" tests, for each experiment in turn, if the by-experiment d-prime is different from a common d-prime computed from the remaining experiments. "base" provides pairwise comparisons to a single experiment indicated by the separate argument `base`. If `test = "zero"` all d-primes are tested versus zero. As a final option a numeric value can be supplied, e.g. `test = 1` in which case all d-primes are tested versus one. Note that `test = 0` gives the same test as `test = "zero"`.

When `test = "pairwise"` a compact letter display is provided and it is determined from the p-values *after* adjustment of these for multiplicity.

The `dprime_compare` and `dprime_test` methods have (common) print method.

## Value

an object of class `c(paste0("posthoc.", class(x)), class(x))` with the following elements from the original object, `x` and :

|              |   |
|--------------|---|
| posthoc      | coefficient table for the post-hoc tests.   |
| test         | the value of the test argument.   |
| alternative  | the value of the alternative argument.  |
| padj.method  | the method used to adjust p-values with.  |
| base         | the value of the base argument.   |
| posthoc.stat | name of the statistic for the post-hoc tests.   |
| Letters      | if <code>test = "pairwise"</code> the compact letter display, otherwise NULL.                     |
| dprime0      | unless <code>test = "pairwise"</code> or "common" the value of d-prime under the null hypothesis. |

## Author(s)

Rune Haubo B Christensen

## See Also

[dprime\\_test](#), [dprime\\_table](#), [dprime\\_compare](#).

**Examples**

```
## Make some fake data:
n <- rep(40, 4)
x <- c(25, 25, 30, 35)
protocol <- c("triangle", "duotrio", "threeAFC", "twoAFC")
## Look at the data table with d-primes etc.:
dprime_table(x, n, protocol)

## 'any differences' test:
## ML estimation and test with likelihood statistic:
(dpc <- dprime_compare(x, n, protocol))

posthoc(dpc, alpha=.1) ## test="pairwise"

## Test if each d' is different from the common d' estimated from the
## remaining experiments:
posthoc(dpc, test="common")

## Test if d' from experiment 2 is different from the others (with
## adjustment for multiplicity):
posthoc(dpc, test="base", base=2)

## Test if each d' is different from 2 (with Bonferroni adjustment for
## multiplicity) using the Wald statistic:
posthoc(dpc, test=2, stat="Wald", padj.method="bonferroni")
```

---

profile.discrim

*Profile likelihood and confidence interval methods for discrim objects*


---

**Description**

Computes the (normalized or relative) profile likelihood for the parameters of a discrimination test, plots the normalized profile likelihood.

**Usage**

```
## S3 method for class 'discrim'
profile(fitted, ...)

## S3 method for class 'profile.discrim'
plot(x, level = c(0.99, 0.95), fig = TRUE,
      method = "natural", n = 1e3, ...)

## S3 method for class 'discrim'
confint(object, parm, level = 0.95, ...)
```

**Arguments**

|        |  |
|--------|--|
| fitted | a discrim object   |
| x      | a profile.discrim object   |
| object | a discrim object   |
| parm   | currently not used   |
| method | the type of spline to be used in approximating the profile likelihood curve (trace)—see <a href="#">spline</a> for details   |
| n      | the number of spline interpolations to use in plotting the profile likelihood curve (trace)  |
| level  | for plot: At which levels to include horizontal lines to indicate confidence levels in plots of the normalized profile likelihoods. For confint: at which level to compute the confidence interval |
| fig    | logical: should the normalized profile likelihoods be plotted?   |
| ...    | For plot: additional arguments to plot. Otherwise not used.  |

**Details**

confint returns the confidence interval computed in [discrim](#) possibly at another level. The statistic used to compute the confidence interval is therefore determined in the [discrim](#) call and may not be the likelihood root.

The likelihood profile is extracted from the [discrim](#) object fitted with `statistic = "likelihood"`.

**Value**

For profile: An object of class "profile.discrim", "data.frame"—a data.frame with two columns giving the value of the parameter and the corresponding value of the profile likelihood.

For plot: The profile object is returned invisibly.

For confint:

A 3x2 matrix with columns named "lower", "upper" giving the lower and upper (100 \* level)% confidence interval for the parameters named in the rows.

**Author(s)**

Rune Haubo B Christensen and Per Bruun Brockhoff

**References**

Brockhoff, P.B. and Christensen R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

**See Also**

[discrim](#)

**Examples**

```
## 7 success out of 10 samples in a duo-trio experiment:
(dd <- discrim(7, 10, method = "duotrio", statistic = "likelihood"))
confint(dd)
plot(profile(dd))
points(confint(dd)[3,], rep(.1465, 2), pch = 3, cex = 2, lwd=2)
```

---

profile.samediff      *Profile likelihood methods for samediff objects.*

---

**Description**

Computes the (normalized or relative) profile likelihood for the parameters of a same-different test, plots the normalized profile likelihood and computes profile likelihood confidence intervals.

**Usage**

```
## S3 method for class 'samediff'
profile(fitted, which = 1:2, max = 2, numpts = 100,
        max.delta = 10, max.tau = 10, ...)
## S3 method for class 'profile.samediff'
plot(x, which = 1:nc, level = c(0.99, 0.95),
      fig = TRUE, ...)
## S3 method for class 'samediff'
confint(object, parm = c("tau", "delta"), level = 0.95, max = c(10, 10)
        , ...)
```

**Arguments**

|           |   |
|-----------|---|
| fitted    | a samediff object   |
| x         | a profile.samediff object   |
| object    | a samediff object   |
| which     | numeric: which parameters to profile or plot; either "1" or "2" or "1:2" to mean "tau", "delta" or both respectively.   |
| parm      | the parameter(s) to compute the confidence interval for   |
| max       | for profile: control parameter to specify how many units beyond the MLE, the profiling should proceed. For confint: control parameter, that can control the convergence for especially very large delta |
| numpts    | control parameter: At how many points should the profile likelihood be evaluated?   |
| max.delta | control parameter: The maximum point at which to evaluate the profile likelihood for delta  |
| max.tau   | same as max.delta for "tau".  |

|       |   |
|-------|---|
| level | for plot: At which levels to include horizontal lines to indicate confidence levels in plots of the normalized profile likelihoods. For confint: at which level to compute the confidence interval. |
| fig   | logical: Should the normalized profile likelihoods be plotted?  |
| ...   | not currently used.   |

### Value

For profile: An object of class "profile.samediff", "data.frame"—a data.frame with two columns for each parameter profiled giving the value of the parameter and the corresponding value of the profile likelihood.

For plot: An object of class "nProfile.samediff", "data.frame"—the data.frame from the profile-object with extra columns corresponding to the which parameter containing the normalized profile likelihood.

For confint: A 2x2 matrix with columns named "lower", "upper" giving the lower and upper (1 - alpha)% confidence interval for the parameters named in the rows.

### Author(s)

Rune Haubo B Christensen

### See Also

[summary.samediff](#)

### Examples

```
# data: 8 of the same samples were judged to be same
#       5 of the same samples were judged to be different
#       4 of the different samples were judged to be same
#       9 of the different samples were judged to be different

sadi <- samediff(8, 5, 4, 9)
confint(sadi)
plot(profile(sadi))
```

## Description

Transforms or rescales estimates and optionally standard errors between the three levels at which a sensory difference is measured: pc (proportion of correct answers), pd (proportion of discriminators) and d-prime. `rescale` is the main function and only one of pc, pd or d-prime should be given as argument — values for the remaining two scales will be computed.

A number of auxiliary functions are also provided:

`psyfun` implements the psychometric functions and maps from d-prime to pc

`psyinv` implements the inverse psychometric functions and maps from pc to d-prime

`psyderiv` implements the derivative of the psychometric functions

`pc2pd` maps from pc to pd

`pd2pc` maps from pd to pc

## Usage

```
rescale(pc, pd, d.prime, std.err,
        method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                  "triangle", "hexad", "twofive", "twofiveF"),
        double = FALSE)

psyfun(d.prime, method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                          "triangle", "hexad", "twofive", "twofiveF"),
       double = FALSE)

psyinv(pc, method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                    "triangle", "hexad", "twofive", "twofiveF"),
       double = FALSE)

psyderiv(d.prime, method = c("duotrio", "tetrad", "threeAFC", "twoAFC",
                             "triangle", "hexad", "twofive", "twofiveF"),
        double = FALSE)

pc2pd(pc, Pguess)

pd2pc(pd, Pguess)
```

## Arguments

|                      |   |
|----------------------|---|
| <code>pc</code>      | the proportion of correct answers; a numerical vector between 0 and 1   |
| <code>pd</code>      | the proportion of discriminators; a numerical vector between 0 and 1  |
| <code>d.prime</code> | the sensory difference on the d-prime scale; a non-negative numerical vector.   |
| <code>std.err</code> | optional numerical vector of standard errors of the same length as the either of pc, pd or d.prime. Negative values are not allowed, but values may be NA |
| <code>method</code>  | the sensory discrimination protocol for which the results should apply  |

|        |   |
|--------|---|
| double | should the 'double' variant of the discrimination protocol be used? Logical scalar. |
| Pguess | the guessing probability implied by the protocol; a numeric scalar between 0 and 1  |

### Details

The rescale function is based on the fact that once the protocol and one of pc, pd and d-prime is known, the other two can be computed. The same applies to the standard errors of these parameters.

Standard errors are optional, but if they are supplied, the length of the std.err argument has to match the length of pc, pd or d.prime whichever is given.

A print method is implemented for rescale objects.

### Value

For rescale an object of class rescale with elements

|              |  |
|--------------|--|
| coefficients | a data.frame with values of pc, pd and d.prime corresponding to the input  |
| std.err      | if standard errors are given through the std.err argument a data.frame of the same size and shape as coefficients with standard errors. Otherwise missing. |
| method       | the sensory discrimination protocol for which the results apply  |

For psyfun, psyinv, psyderiv, pc2pd and pd2pc a numerical vector of the same length as the first argument with appropriate contents.

### Author(s)

Rune Haubo B Christensen

### Examples

```
## suppose 15 out of 20 are observed in a duo-trio experiment, then
## the estimated probability of correct a answer is
(pc <- 15/20)
## The standard error of this estimate is
(se.pc <- sqrt(pc * (1 - pc) / 20))
## The corresponding estimate of proportion of discriminators (pd) and
## d-prime with associated standard errors are:
rescale(pc = pc, std.err = se.pc, method = "duotrio")

## Can also do
rescale(pd = c(.6,.7), std.err = c(.2, NA))
psyfun(2, method = "triangle")
psyinv(0.8, method = "twoAFC")
psyderiv(2, method = "duotrio")
pc2pd(0.7, 1/2)
pd2pc(0.3, 1/3)
```



**Description**

The function computes and plots the empirical ROC (receiver operating characteristic) curve.

**Usage**

```
ROC(object, ...)

## Default S3 method:
ROC(object, se.d, scale = 1, length = 1000,
     fig = TRUE, se.type = c("CI", "SE"), CI.alpha = 0.05, ...)

## S3 method for class 'anota'
ROC(object, length = 1000, fig = TRUE,
     se.type = c("CI", "SE"), CI.alpha = 0.05, ...)
```

**Arguments**

|          |  |
|----------|--|
| object   | the class of the object defines, which of the methods is invoked. If object is a single element numeric vector it is taken as a d-prime value and the default method is invoked. If the object is of class anota, the method for anota objects is invoked. |
| se.d     | a unit length vector with the standard error of d-prime. If supplied confidence intervals or standard errors are plotted   |
| scale    | a unit length vector giving the ratio of scale (ie. standard deviation) of the latent distribution for the no-class items relative to that of the yes-class items  |
| length   | the length of the vectors to be plotted. Longer vectors gives more smooth curves.  |
| fig      | Should a plot be produced?   |
| se.type  | The type of band for the ROC curve, "CI" for confidence interval and "SE" for standard error.  |
| CI.alpha | the type I level of the confidence interval of AUC   |
| ...      | additional arguments to plot and lines   |

**Details**

The function currently ignores the variance of the scale in the computation of the uncertainty of the ROC curve.

**Value**

The function makes a plot of the ROC curve, and if `se.d` is supplied, standard errors or confidence intervals for the curve are added to the plot.

The function also (invisibly) returns a list with the following components

ROCx            x-coordinates to the ROC curve  
 ROCy            y-coordinates to the ROC curve

If `se.d` is supplied, the object also contains

lower            y-coordinates to the lower limit  
 upper            y-coordinates to the upper limit

**Author(s)**

Rune Haubo B Christensen

**Examples**

```
## ROC.default:
(mat <- matrix(c(8, 17, 1, 24), 2, byrow = TRUE))
(d.prime <- SDT(mat, "probit")[3])
ROC(d.prime)
## ROC.anota:
fm1 <- Anota(8, 25, 1, 25)
ROC(fm1)
```

---

samediff

*Computation of tau and dprime for same different test*

---

**Description**

Computation of tau and dprime and their uncertainties for the same different test using maximum likelihood.

**Usage**

```
samediff(nsamesame, ndiffsame, nsamediff, ndiffdiff, VCOV = TRUE)
```

**Arguments**

nsamesame        The number of same-answers on same-samples  
 ndiffsame        The number of different-answers on same-samples  
 nsamediff        The number of same-answers on different-samples  
 ndiffdiff        The number of different-answers on different-samples  
 VCOV            Should the variance-covariance matrix of the parameters be computed. Defaults to TRUE.

**Details**

The function computes the maximum likelihood estimates of tau and delta.

**Value**

An object of class `samediff` with elements

|                          |   |
|--------------------------|---|
| <code>coef</code>        | named vector of coefficients (d-prime and tau)  |
| <code>vcov</code>        | variance-covariance matrix of the coefficients  |
| <code>se</code>          | named vector with standard error of the coefficients (standard error of d-prime)            |
| <code>data</code>        | a named vector with the data supplied to the function                                       |
| <code>test</code>        | a string with the name of the test (same-different)   |
| <code>call</code>        | the matched call  |
| <code>convergence</code> | convergence indicator. 0 indicates convergence. For error codes see <a href="#">optim</a> . |
| <code>logLik</code>      | Value of the log-likelihood at the MLE of the parameters.                                   |
| <code>case</code>        | A case indicator for internal use   |

**Author(s)**

Rune Haubo B Christensen

**References**

Christensen, R.H.B., Brockhoff, P.B. (2009). Estimation and inference in the same-different test. *Food, Quality and Preference*, 20 pp. 514–520

**Examples**

```
# data: 8 of the same samples were judged to be same
#       5 of the same samples were judged to be different
#       4 of the different samples were judged to be same
#       9 of the different samples were judged to be different

samediff(8, 5, 4, 9)
```

---

samediffPwr

*Power Analysis for Same-different Experiments*

---

**Description**

Computes the power for at same-different discrimination experiment with a no-difference null hypothesis via simulation.

**Usage**

```
samediffPwr(n = 1000, tau, delta, Ns, Nd, alpha = 0.05)
```

### Arguments

|       |   |
|-------|---|
| n     | the number of samples to use in the simulation. More samples means higher precision, but takes longer to compute. |
| tau   | the value of tau  |
| delta | the underlying sensory difference under the <i>alternative</i> hypothesis (non-negative)                          |
| Ns    | the number of same-samples (a positive integer)   |
| Nd    | the number of different-samples (a positive integer)  |
| alpha | the type I level of the test (must be between zero and one)   |

### Details

The power is computed using simulations.  $n$  datasets is simulated from the Same Different model with specified parameters. The power is the fraction of times the p-value is lower than alpha.

Under some parameter combinations, there is a non-significant probability that data will fall, so that the MLE of delta is not defined and the p-value is not defined. All such undefined p-values are silently ignored.

The estimated power may change between runs and especially if the power is either very large or very small (ie. close to 0 or 1). Using more simulations will provide higher accuracy.

It is often a good idea to run the power simulation a couple of times to ensure that the variation in the result is acceptable.

### Value

A single numeric value giving the power of the specified test.

### Author(s)

Rune Haubo B Christensen

### References

Christensen, R.H.B., Brockhoff, P.B. (2009). Estimation and inference in the same-different test. *Food, Quality and Preference*, 20 pp. 514–520

### See Also

[samediff](#), [samediffSim](#)

### Examples

```
## Finding the power of a discrimination test with a sensory delta of 2
## (alternative hypothesis) versus a null hypothesis of delta = 0 with
## a sample of size 2 x 10 and a type I level of .05. n should be higher
## for a reasonable precision:
```

```
samediffPwr(n = 100, tau = 1, delta = 2, Ns = 10, Nd = 10)
```

---

|             |  |
|-------------|--|
| samediffSim | <i>Simulates data from a samediff test</i> |
|-------------|--|

---

**Description**

Simulates the outcome of n same-different experiments.

**Usage**

```
samediffSim(n, tau, delta, Ns, Nd)
```

**Arguments**

|       |  |
|-------|--|
| n     | the number of experiments to simulate. |
| tau   | the value of "tau".                    |
| delta | the value of delta (d-prime).          |
| Ns    | number of same-samples                 |
| Nd    | number of different-samples            |

**Details**

The function makes two calls to [rbinom](#).

**Value**

A matrix of with n rows and four columns named ss, ds, sd, dd with the number of same-answers to same-samples, different-answers to same-samples, same-answers to different-samples and different-answers to different-samples respectively.

**Author(s)**

Rune Haubo B Christensen

**References**

Christensen, R.H.B., Brockhoff, P.B. (2009). Estimation and inference in the same-different test. Food, Quality and Preference, 20 pp. 514–520

**See Also**

[discrimSim](#)

**Examples**

```
## Running simulations:  
samediffSim(n = 10, tau = 1, delta = 1, Ns = 10, Nd = 10)
```

**Description**

The function computes d-prime for any 2 x J table where  $J \geq 2$  for the "yes-no" or "A-Not A" experiment using the Signal Detection Theory (SDT) algorithm to compute J-1 d-prime's. The algorithm is also called the "empirical probit transform". The function also provides the "logit" counterpart.

**Usage**

```
SDT(tab, method = c("probit", "logit"))
```

**Arguments**

|        |  |
|--------|--|
| tab    | A 2 x J table with true class relation in rows (only two true classes) and the J-class response in columns |
| method | should the empirical probit or logit transform be computed?  |

**Value**

A (J-1) x 3 matrix. The first two columns contains the z-transform of the Hit rate and the False Alarm rate respectively—ready to plot along with the empirical ROC curve. The third column contains the estimated d-primes.

**Author(s)**

Rune Haubo B Christensen

**References**

MacMillan, A. N. and Creelman, C. D (2005) Detection Theory A User's Guide. Lawrence Erlbaum Associates, Inc. 2nd edition.

**Examples**

```
### Design table:
## 8 "yes"-responses to yes-samples
## 1 "yes"-responses to no-samples
## 17 "no"-response to yes-samples
## 24 "no"-responses to no-samples
## Note that response-class is columnwise and true-class is rowwise:
(mat <- rbind(c(8, 17),
              c(1, 24)))
SDT(mat, "logit")
SDT(mat, "probit")
```

```

## compare to AnotA():
m1 <- AnotA(8, 25, 1, 25)
m1

## Compute d-prime 'by hand':
## Hit rate and False alarm rates:
H <- 8/(8+17)
FA <- 1/(1+24)
zH <- qnorm(H)
zFA <- qnorm(FA)
## d-prime:
zH - zFA # d'

## Multi-response-class example (odor example from MacMillan and
## Creelman, 2005)
(odor <- matrix(c(112, 112, 72, 53, 22, 4, 7, 38, 50, 117, 101, 62), 2,
                byrow = TRUE))
obj <- SDT(odor)
ROC(obj[3,3])

```

---

sensR-deprecated

*Deprecated Functions in sensR Package*


---

## Description

These functions are provided for compatibility with older versions of the **sensR** package only, and may be removed eventually.

## Details

The `c11s` function is deprecated and users encouraged to use `clm` instead from the **ordinal** package.

---

summary.samediff

*Summary method for samediff objects.*


---

## Description

Makes a summary of a `samediff` object with option to use profile likelihood for confidence intervals and p-values or the asymptotic variance-covariance matrix.

## Usage

```

## S3 method for class 'samediff'
summary(object, profile = TRUE, ...)

```

**Arguments**

|         |   |
|---------|---|
| object  | a samediff object   |
| profile | logical: Should the profile likelihood be used for confidence intervals and p-values for the parameters? Defaults to TRUE. If FALSE the asymptotic variance-covariance matrix derived from the observed Fisher information matrix will be used. See Details for more information. |
| ...     | can be level, eg 0.95 to specify the confidence level of the intervals.   |

**Details**

Note that the variance-covariance matrix does not always exist in contrast to the profile likelihood. `profile = FALSE` may therefore cause confidence intervals etc. to be NA.

**Value**

An object of class `summary.samediff` inheriting elements from the `samediff` object and with the following additional elements

|       |  |
|-------|--|
| table | matrix with parameter estimates, standard errors, confidence intervals and p-values. |
| AIC   | the AIC of the object.   |

**Author(s)**

Rune Haubo B Christensen

**See Also**

[confint.samediff](#), [profile.samediff](#)

**Examples**

```
# data: 8 of the same samples were judged to be same
#       5 of the same samples were judged to be different
#       4 of the different samples were judged to be same
#       9 of the different samples were judged to be different

sadi <- samediff(8, 5, 4, 9)
summary(sadi)
summary(sadi, FALSE)
```



---

|        |                                      |
|--------|--------------------------------------|
| tetrad | <i>Create tetrad binomial family</i> |
|--------|--------------------------------------|

---

**Description**

Creates a binomial family object with the inverse link function equal to the psychometric function for the unspecified method of tetrads.

**Usage**

```
tetrad()
```

**Value**

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct d-prime computation) as `linkfun`.

**Note**

Several functions in this package makes use of functions in the tetrad family object, but it may also be used on its own—see the example below.

**Author(s)**

Rune Haubo B Christensen

**References**

Ennis, J. M., Ennis, D. M., Yip, D., & O'Mahony, M. (1998). Thurstonian models for variants of the method of tetrads. *British Journal of Mathematical and Statistical Psychology*, 51, pp. 205-215.

Ennis, J. M., & Jesionka, V. (2011). The power of sensory discrimination methods revisited. *Journal of Sensory Studies*, 26, pp. 371-382.

**See Also**

[duotrio](#), [twoAFC](#), [threeAFC](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Estimating d-prime using glm for a Tetrad test:  
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect  
res <- glm(xt ~ 1, family = tetrad)  
summary(res)  
## Equivalent to (Estimate and Std. Error):
```

```

discrim(10, 15, method="tetrad")

## Extended example plotting the profile likelihood
## data: 10 correct answers, 9 incorrect
xt <- matrix(c(10, 9), ncol = 2)
summary(res <- glm(xt ~ 1, family = tetrad))
N <- 100
dev <- double(N)
delta <- seq(1e-4, 3, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = tetrad)$deviance
plot(delta, exp(-dev/2), type = "l",
      xlab = expression(delta),
      ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
level <- c(0.95, 0.99)
lim <- sapply(level, function(x) exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")

```

---

threeAFC

*Create 3-AFC binomial family*


---

## Description

Creates a copy of the binomial family with the inverse link function changed to equal the 3-AFC psychometric function and correspondingly changed link function and derivative of the inverse link function.

## Usage

```
threeAFC()
```

## Value

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct `dprime` computation) as `linkfun`.

## Note

Several functions in this package makes use of the function, but it may also be used on its own—see the example below.

**Author(s)**

Rune Haubo B Christensen and Per Bruun Brockhoff

**References**

Brockhoff, P.B. and Christensen, R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

**See Also**

[triangle](#), [twoAFC](#), [tetrad](#), [duotrio](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Estimating d-prime using glm for a 3-AFC test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = threeAFC)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="threeAFC")

## Extended example plotting the profile likelihood
## data: 10 correct answers, 5 incorrect
xt <- matrix(c(10, 2), ncol = 2)
summary(res <- glm(xt ~ 1, family = threeAFC))#, etastart = etastart))
N <- 100
dev <- double(N)
level <- c(0.95, 0.99)
delta <- seq(1e-4, 5, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = threeAFC)$deviance
plot(delta, exp(-dev/2), type = "l",
     xlab = expression(delta),
     ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
lim <- sapply(level, function(x)
              exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")
```

---

|          |  |
|----------|--|
| triangle | <i>Create triangle binomial family</i> |
|----------|--|

---

### Description

Creates a copy of the binomial family with the inverse link function changed to equal the triangle psychometric function and correspondingly changed link function and derivative of the inverse link function.

### Usage

```
triangle()
```

### Value

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct `dprime` computation) as `linkfun`.

### Note

Several functions in this package makes use of the function, but it may also be used on its own—see the example below.

### Author(s)

Rune Haubo B Christensen and Per Bruun Brockhoff

### References

Brockhoff, P.B. and Christensen, R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

### See Also

[duotrio](#), [twoAFC](#), [tetrad](#), [threeAFC](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [Anota](#), [discrimSS](#), [samediff](#), [findcr](#)

### Examples

```
## Estimating d-prime using glm for a Triangle test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = triangle)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="triangle")

## Extended example plotting the profile likelihood
```

```

## data: 10 correct answers, 9 incorrect
xt <- matrix(c(10, 9), ncol = 2)
summary(res <- glm(xt ~ 1, family = triangle))
N <- 100
dev <- double(N)
delta <- seq(1e-4, 3, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = triangle)$deviance
plot(delta, exp(-dev/2), type = "l",
      xlab = expression(delta),
      ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
level <- c(0.95, 0.99)
lim <- sapply(level, function(x) exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")

```

twoAC

*2-AC Discrimination and Preference Protocol***Description**

Computes estimates and standard errors of d-prime and tau for the two alternative (2-AC) protocol. A confidence interval and significance test for d-prime is also provided. The 2-AC protocol is equivalent to a 2-AFC protocol with a "no-difference" option, and equivalent to a paired preference test with an "no-preference" option.

**Usage**

```

twoAC(data, d.prime0 = 0, conf.level = 0.95,
      statistic = c("likelihood", "Wald"),
      alternative = c("two.sided", "less", "greater"), ...)

```

**Arguments**

|            |   |
|------------|---|
| data       | a non-negative numeric vector of length 3 with the number of observations in the three response categories in the form ("prefer A", "no-preference", "prefer B"). If the third element is larger than the first element, the estimate of d-prime is positive. |
| d.prime0   | the value of d-prime under the null hypothesis for the significance test.   |
| conf.level | the confidence level.   |
| statistic  | the statistic to use for confidence level and significance test.  |

alternative the type of alternative hypothesis.  
 ... not currently used.

### Details

[confint](#), [profile](#), `logLik`, `vcov`, and `print` methods are implemented for `twoAC` objects.  
 Power computations for the 2-AC protocol is implemented in [twoACpwr](#).

### Value

An object of class `twoAC` with elements

`coefficients` 2 by 2 coefficient matrix with estimates and standard errors of d-prime and tau. If the variance-covariance matrix of the parameters is not defined, the standard errors are NA.

`vcov` variance-covariance matrix of the parameter estimates. Only present if defined for the supplied data.

`data` the data supplied to the function.

`call` the matched call.

`logLik` the value of the log-likelihood at the maximum likelihood estimates.

`alternative` the name of the alternative hypothesis for the significance test.

`statistic` the name of the test statistic used for the significance test.

`conf.level` the confidence level for the confidence interval for d-prime.

`d.prime0` the value of d-prime under the null hypothesis in the significance test.

`p.value` p-value of the significance test.

`confint` two-sided confidence interval for d-prime. This is only available if the standard errors are defined, which may happen in boundary cases. Use `profile` and `confint` methods to get confidence intervals instead; see the examples.

### Author(s)

Rune Haubo B Christensen

### References

Christensen R.H.B., Lee H-S and Brockhoff P.B. (2011). Estimation of the Thurstonian model for the 2-AC protocol. Submitted to Food Quality and Preference.

### See Also

[c1m2twoAC](#), [twoACpwr](#)

**Examples**

```

## Simple:
fit <- twoAC(c(2,2,6))
fit

## Typical discrimination-difference test:
(fit <- twoAC(data = c(2, 5, 8), d.prime0 = 0, alternative = "greater"))

## Typical discrimination-similarity test:
(fit <- twoAC(data = c(15, 15, 20), d.prime0 = .5, alternative = "less"))

## Typical preference-difference test:
(fit <- twoAC(data = c(3, 5, 12), d.prime0 = 0,
              alternative = "two.sided"))

## Typical preference (non-)inferiority test:
(fit <- twoAC(data = c(3, 5, 12), d.prime0 = 0,
              alternative = "greater"))

## For preference equivalence tests (two-sided) use CI with alpha/2:
## declare equivalence at the 5% level if 90% CI does not contain,
## e.g, -1 or 1:
(fit <- twoAC(data = c(15, 10, 10), d.prime0 = 0, conf.level = .90))

## The var-cov matrix and standard errors of the parameters are not
## defined in all situations. If standard errors are not
## defined, then confidence intervals are not provided directly:
(fit <- twoAC(c(5, 0, 15)))
## We may use profile and confint methods to get confidence intervals
## never the less:
pr <- profile(fit, range = c(-1, 3))
confint(pr)
plot(pr)

```

---

twoACpwr

*Exact Power Computation for the 2-AC Discrimination Protocol*


---

**Description**

Computes the exact power for the 2-AC protocol using the (signed) likelihood root statistic. Power is computed for a significance test of d-prime. The tol argument specifies the precision with which power should be computed.

**Usage**

```
twoACpwr(tau, d.prime, size, d.prime0 = 0, alpha = 0.05, tol = 1e-5,
```

```
return.dist = FALSE, statistic = "likelihood",
alternative = c("two.sided", "less", "greater"))
```

### Arguments

|             |  |
|-------------|--|
| tau         | the value of tau under the alternative hypothesis  |
| d.prime     | the value of d.prime under the alternative hypothesis  |
| size        | the sample size  |
| d.prime0    | the value of d-prime under the null hypothesis in the significance test for which power should be computed   |
| alpha       | the size of the test   |
| tol         | specifies the precision with which power should be computed, e.g., 1e-4 cause power to be computed correctly to three significant digits. Lower values of tau gives higher precision, but also longer computation times. |
| return.dist | should the p-value distribution be returned rather than the power be computed?   |
| statistic   | the statistic used in the significance test for which the power should be computed. Currently only the (signed) likelihood root statistic is available—see the details for more information.                             |
| alternative | the type of alternative hypothesis in the significance test for which the power should be computed   |

### Details

The main idea in this function is to compute all possible data outcomes and then compute the p-value for the chosen significance test for each of these outcomes. This gives the exact distribution of p-values from which the exact power can be computed. This is basically what happens if  $tol = 0$ .

There is, however, a problem with this approach if *size* is large, since the the number of possible outcomes increases very fast with the *size*; the order is  $O(n^2)$ . The solution to this problem is to ignore those outcomes which will occur with very small probability. Often, a large proportion of the outcomes, say 90% will occur so rarely that they account for, say 1e-4 percent of the probability mass; it is therefore safe to ignore those outcomes without compromising the accuracy of the computed power by any relevant amount. For more information see the referenced paper and the package vignette *Statistical Methodology*.

The Wald statistic is not available here. The reason is that the Wald statistic is not always defined and the problem is therefore what to do with those cases where it is not defined? On the other hand the likelihood root statistic is defined in all cases, so there is no problem here, and since the likelihood root statistic is more accurate than the Wald statistic, there is not much reason to use the Wald statistic after all.

For the record; the Wald statistic is not defined, when the standard error of d-prime is not defined. This happens when the variance-covariance matrix of tau and d-prime is not defined, which occurs in a number of boundary cases, i.e., when one or more cells contain zero frequencies. Since these outcomes occur with positive probability, the algorithm used by *twoACpwr* will always encounter those cases and have to deal with them. This would be cumbersome to implement.



**Value**

A data.frame with one line and the following entries

|              |   |
|--------------|---|
| power        | the computed power  |
| actual.alpha | the actual size of the test (different from the nominal alpha given as argument due to the discreteness of the observations). |
| samples      | the number of possible outcomes for this size   |
| discarded    | the number of outcomes for which the p-value is not computed. This number is zero if tol = 0                                  |
| kept         | the number of outcomes for which the p-value is computed in. This number equals samples if tol = 0                            |
| p            | the probability vector of the multinomial distribution implied by the values of tau and d.prime.                              |

**Author(s)**

Rune Haubo B Christensen

**References**

Christensen R.H.B., Lee H-S and Brockhoff P.B. (2011). Estimation of the Thurstonian model for the 2-AC protocol. Submitted to Food Quality and Preference.

**See Also**

[c1m2twoAC](#), [twoACpwr](#)

**Examples**

```
## Exact power:
twoACpwr(tau = .5, d.prime = .7, size = 50, tol = 0)

## Power exact to a reasonable number of digits
twoACpwr(tau = .5, d.prime = .7, size = 50, tol = 1e-5)

## Power for a similarity test in a discrimination setting where the
## true parameter values are expected to be tau = 0.4 and true d.prime
## = .5, while we want to show that d.prime < 1, i.e., under the null
## hypothesis d.prime = 1:
twoACpwr(tau = .4, d.prime = .5, size = 100, d.prime0 = 1, tol = 1e-5,
         alternative = "less")

## Power for a difference test in a preference setting where the true
## parameter values are expected to be tau = 0.4 and d.prime = -0.5,
## while we want to show that d.prime is different from zero:
twoACpwr(tau = 0.4, d.prime = -0.5, size = 100, d.prime0 = 0, tol = 1e-5,
         alternative = "two.sided")
```

---

`twoAFC`*Create 2-AFC binomial family*

---

**Description**

Creates a copy of the binomial family with the inverse link function changed to equal the 2-AFC psychometric function and correspondingly changed link function and derivative of the inverse link function.

**Usage**

```
twoAFC()
```

**Value**

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct `dprime` computation) as `linkfun`.

**Note**

Several functions in this package makes use of the function, but it may also be used on its own—see the example below.

**Author(s)**

Rune Haubo B Christensen and Per Bruun Brockhoff

**References**

Brockhoff, P.B. and Christensen, R.H.B. (2010). Thurstonian models for sensory discrimination tests as generalized linear models. *Food Quality and Preference*, 21, pp. 330-338.

**See Also**

[triangle](#), [threeAFC](#), [tetrad](#), [duotrio](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Estimating d-prime using glm for a 2-AFC test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = twoAFC)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="twoAFC")
```

```
## Extended example plotting the profile likelihood
## data: 10 correct and 8 incorrect:
```

```

xt <- matrix(c(10, 8), ncol = 2)
summary(res <- glm(xt ~ 1, family = twoAFC))
N <- 100
dev <- double(N)
level <- c(0.95, 0.99)
delta <- seq(1e-4, 3, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = twoAFC)$deviance
plot(delta, exp(-dev/2), type = "l",
      xlab = expression(delta),
      ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
lim <- sapply(level, function(x)
              exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")

```

---

twofive

---

*Create twofive binomial family*


---

## Description

Creates a binomial family object with the inverse link function equal to the psychometric function for the Two-Out-of-Five test.

## Usage

```
twofive()
```

## Value

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct  $d'$ -prime computation) as `linkfun`.

## Note

Several functions in this package make use of functions in the `twofive` family object, but it may also be used on its own—see the example below.

## Author(s)

Karolina Stachlewska

## References

Ennis, J. M. (2013). A thurstonian analysis of the Two-Out-of-Five test. *Journal of Sensory Studies*, 28(4), pp. 297-310.

## See Also

[duotrio](#), [triangle](#), [twoAFC](#), [threeAFC](#), [tetrad](#), [twofiveF](#), [hexad](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

## Examples

```
## Estimating d-prime using glm for a Two-Out-of-Five test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = twofive)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="twofive")

## Extended example plotting the profile likelihood
## data: 10 correct answers, 9 incorrect
xt <- matrix(c(10, 9), ncol = 2)
summary(res <- glm(xt ~ 1, family = twofive))
N <- 100
dev <- double(N)
delta <- seq(1e-4, 3, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = twofive)$deviance
plot(delta, exp(-dev/2), type = "l",
     xlab = expression(delta),
     ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
level <- c(0.95, 0.99)
lim <- sapply(level, function(x) exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")
```

---

twofiveF

*Create twofiveF binomial family*

---

## Description

Creates a binomial family object with the inverse link function equal to the psychometric function for the Two-Out-of-Five with forgiveness test.

**Usage**

```
twofiveF()
```

**Value**

A binomial family object for models. Among other things it includes the psychometric function as `linkinv` and the inverse psychometric function (for direct d-prime computation) as `linkfun`.

**Note**

Several functions in this package makes use of functions in the `twofiveF` family object, but it may also be used on its own—see the example below.

**Author(s)**

Karolina Stachlewska

**References**

Ennis, J. M. (2013). A thurstonian analysis of the Two-Out-of-Five test. *Journal of Sensory Studies*, 28(4), pp. 297-310.

**See Also**

[duotrio](#), [triangle](#), [twoAFC](#), [threeAFC](#), [tetrad](#), [twofive](#), [hexad](#), [discrim](#), [discrimPwr](#), [discrimSim](#), [AnotA](#), [discrimSS](#), [samediff](#), [findcr](#)

**Examples**

```
## Estimating d-prime using glm for a Two-Out-of-Five with forgiveness test:
xt <- matrix(c(10, 5), ncol = 2) ## data: 10 correct answers, 5 incorrect
res <- glm(xt ~ 1, family = twofiveF)
summary(res)
## Equivalent to (Estimate and Std. Error):
discrim(10, 15, method="twofiveF")
```

```
## Extended example plotting the profile likelihood
## data: 10 correct answers, 9 incorrect
xt <- matrix(c(10, 9), ncol = 2)
summary(res <- glm(xt ~ 1, family = twofiveF))
N <- 100
dev <- double(N)
delta <- seq(1e-4, 3, length = N)
for(i in 1:N)
  dev[i] <- glm(xt ~ -1 + offset(delta[i]),
               family = twofiveF)$deviance
plot(delta, exp(-dev/2), type = "l",
```

```
      xlab = expression(delta),
      ylab = "Normalized Profile Likelihood")
## Add Normal approximation:
lines(delta, exp(-(delta - coef(res))^2 /
                (2 * vcov(res))), lty = 2)
## Add confidence limits:
level <- c(0.95, 0.99)
lim <- sapply(level, function(x) exp(-qchisq(x, df=1)/2) )
abline(h = lim, col = "grey")
```

# Index

- \*Topic **hplot**
  - plot.discrim, 47
  - plot.samediff, 48
  - ROC, 57
- \*Topic **htest**
  - AUC, 5
- \*Topic **models**
  - AnotA, 3
  - betabin, 6
  - c1ls-deprecated, 10
  - clm2twoAC, 12
  - confint.twoAC, 14
  - discrim, 16
  - discrimPwr, 19
  - discrimR, 21
  - discrimSim, 23
  - discrimSS, 24
  - dod, 27
  - dod\_fit, 35
  - dod\_utils, 36
  - dodControl, 29
  - dodPwr, 30
  - dodSim, 33
  - dprime\_compare, 38
  - dprime\_table, 40
  - dprime\_test, 41
  - duotrio, 43
  - findcr, 44
  - hexad, 46
  - posthoc, 49
  - profile.discrim, 51
  - profile.samediff, 53
  - rescale, 54
  - samediff, 58
  - samediffPwr, 59
  - samediffSim, 61
  - SDT, 62
  - summary.samediff, 63
  - tetrad, 65
  - threeAFC, 66
  - triangle, 68
  - twoAC, 69
  - twoACpwr, 71
  - twoAFC, 74
  - twofive, 75
  - twofiveF, 76
- AnotA, 3, 5, 18, 20, 22, 24, 26, 44–46, 65, 67, 68, 74, 76, 77
- anova, 11
- AUC, 5
- beta, 8
- betabin, 6
- c1ls (c1ls-deprecated), 10
- c1ls-deprecated, 10
- clm, 12, 13, 63
- clm2twoAC, 12, 70, 73
- clmm, 12, 13
- confint, 15, 18, 70
- confint.anota (AnotA), 3
- confint.discrim (profile.discrim), 51
- confint.profile.twoAC (confint.twoAC), 14
- confint.samediff, 64
- confint.samediff (profile.samediff), 53
- confint.twoAC, 14
- d.primePwr (discrimPwr), 19
- d.primeSS (discrimSS), 24
- data.frame, 39
- discrim, 4, 16, 20, 24, 44–46, 52, 65, 67, 68, 74, 76, 77
- discrimPwr, 4, 18, 19, 22, 24, 26, 44–46, 65, 67, 68, 74, 76, 77
- discrimR, 21
- discrimSim, 4, 18, 20, 22, 23, 44–46, 61, 65, 67, 68, 74, 76, 77

- discrimSS, [4](#), [18](#), [20](#), [22](#), [24](#), [24](#), [44–46](#), [65](#),  
[67](#), [68](#), [74](#), [76](#), [77](#)  
 dod, [27](#), [29](#), [31](#), [34](#), [35](#), [37](#)  
 dod\_fit, [28](#), [29](#), [31](#), [34](#), [35](#), [37](#)  
 dod\_nll (dod\_utils), [36](#)  
 dod\_null (dod\_utils), [36](#)  
 dod\_null\_tau (dod\_utils), [36](#)  
 dod\_utils, [36](#)  
 dodControl, [27](#), [28](#), [29](#), [31](#), [34](#), [35](#), [37](#)  
 dodPwr, [28](#), [29](#), [30](#), [34](#), [35](#), [37](#)  
 dodSim, [28–31](#), [33](#), [35](#), [37](#)  
 dprime\_compare, [38](#), [41](#), [42](#), [49](#), [50](#)  
 dprime\_table, [39](#), [40](#), [42](#), [50](#)  
 dprime\_test, [39](#), [41](#), [41](#), [49](#), [50](#)  
 duotrio, [9](#), [18](#), [22](#), [24](#), [43](#), [45](#), [46](#), [65](#), [67](#), [68](#),  
[74](#), [76](#), [77](#)  
  
 findcr, [4](#), [18](#), [20](#), [22](#), [24](#), [26](#), [44](#), [44](#), [46](#), [65](#),  
[67](#), [68](#), [74](#), [76](#), [77](#)  
 formula, [10](#)  
  
 glm, [12](#), [21](#)  
  
 hexad, [9](#), [18](#), [24](#), [46](#), [76](#), [77](#)  
  
 multinom, [12](#)  
  
 nlminb, [29](#)  
  
 optim, [8–10](#), [12](#), [22](#), [59](#)  
 optimal\_tau, [28](#), [29](#), [31](#), [33–35](#)  
 optimal\_tau (dod\_utils), [36](#)  
  
 p.adjust, [50](#)  
 par2prob\_dod (dod\_utils), [36](#)  
 pc2pd (rescale), [54](#)  
 pd2pc (rescale), [54](#)  
 plot.anota (AnotA), [3](#)  
 plot.discrim, [47](#)  
 plot.profile, [18](#)  
 plot.profile.discrim (profile.discrim),  
[51](#)  
 plot.profile.samediff  
     (profile.samediff), [53](#)  
 plot.profile.twoAC (confint.twoAC), [14](#)  
 plot.samediff, [48](#)  
 pnorm, [6](#)  
 polr, [11](#), [12](#)  
 posthoc, [49](#)  
 posthoc.dprime\_compare, [39](#), [41](#), [42](#)  
  
 print.AUC (AUC), [5](#)  
 print.discrim, [4](#)  
 print.discrim(discrim), [16](#)  
 print.dod (dod), [27](#)  
 print.twoAC (twoAC), [69](#)  
 profile, [15](#), [18](#), [70](#)  
 profile.discrim, [51](#)  
 profile.samediff, [53](#), [64](#)  
 profile.twoAC (confint.twoAC), [14](#)  
 prop.test, [17](#)  
 psyderiv (rescale), [54](#)  
 psyfun (rescale), [54](#)  
 psyinv (rescale), [54](#)  
  
 rbinom, [61](#)  
 rescale, [54](#)  
 ROC, [57](#)  
  
 samediff, [18](#), [22](#), [24](#), [26](#), [44–46](#), [58](#), [60](#), [65](#),  
[67](#), [68](#), [74](#), [76](#), [77](#)  
 samediffPwr, [59](#)  
 samediffSim, [60](#), [61](#)  
 SDT, [62](#)  
 sensR-deprecated, [63](#)  
 spline, [52](#)  
 summary, [11](#)  
 summary.betabin (betabin), [6](#)  
 summary.samediff, [54](#), [63](#)  
  
 tetrad, [9](#), [18](#), [24](#), [44–46](#), [65](#), [67](#), [68](#), [74](#), [76](#), [77](#)  
 threeAFC, [9](#), [18](#), [22](#), [24](#), [44–46](#), [65](#), [66](#), [68](#), [74](#),  
[76](#), [77](#)  
 triangle, [9](#), [18](#), [22](#), [24](#), [44–46](#), [67](#), [68](#), [74](#), [76](#),  
[77](#)  
 twoAC, [13](#), [14](#), [69](#)  
 twoACpwr, [13](#), [70](#), [71](#), [73](#)  
 twoAFC, [9](#), [18](#), [22](#), [24](#), [44–46](#), [65](#), [67](#), [68](#), [74](#),  
[76](#), [77](#)  
 twofive, [9](#), [18](#), [24](#), [46](#), [75](#), [77](#)  
 twofiveF, [9](#), [18](#), [24](#), [46](#), [76](#), [76](#)  
  
 vcov, [11](#)  
  
 wilcox.test, [31](#)