Package 'predint'

July 22, 2025

Type Package

Title Prediction Intervals

Version 2.3.0

Description An implementation of prediction intervals for overdispersed count data, for overdispersed binomial data and for linear random effects models.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Imports stats, graphics, methods

RoxygenNote 7.3.2

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5.0), ggplot2, lme4, MASS

URL https://github.com/MaxMenssen/predint

BugReports https://github.com/MaxMenssen/predint/issues

NeedsCompilation no

Author Max Menssen [aut, cre]

Maintainer Max Menssen <menssen@cell.uni-hannover.de>

Repository CRAN

Date/Publication 2025-07-22 11:01:31 UTC

Contents

`ames_HCD` *Historical numbers of revertant colonies in the Ames test (OECD 471)*

Description

This data set contains artificial historical control data that was sampled in order to mimic the number of revertant colonies based on two or three petri dishes.

Usage

```
ames_HCD
```

Format

A data.frame with 2 rows and 10 columns:

rev_col no. of revertant colonies

no_dish no. of petri dishes in the control group

---

as.data.frame.predint *Store prediction intervals or limits as a* data.frame

---

Description

Get the prediction intervals or limits of an object of class predint and save them as a data.frame.

Usage

```
## S3 method for class 'predint'
as.data.frame(x, ...)
```

Arguments

x	object of class predint
...	additional arguments to be passed to base::as.data.frame()

Value

This function returns the prediction intervals or limits stored in an object of class "predint" as a data.frame

Examples

```
### PI for quasi-Poisson data
pred_int <- quasi_pois_pi(histdat=ames_HCD,
                          newoffset=3,
                          nboot=100,
                          traceplot = FALSE)

# Return the prediction intervals as a data.frame
as.data.frame(pred_int)

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

bb_dat1 *Beta-binomial data (example 1)*

Description

This data set contains sampled beta-binomial data from 10 clusters each of size 50. The data set was sampled with `rbbinom(n=10, size=50, prob=0.1, rho=0.06)`.

Usage

```
bb_dat1
```

Format

A `data.frame` with 10 rows and 2 columns:

succ number of successes

fail number of failures

bb_dat2 *Beta-binomial data (example 2)*

Description

This data set contains sampled beta-binomial data from 3 clusters each of different size. The data set was sampled with `rbbinom(n=3, size=c(40, 50, 60), prob=0.1, rho=0.06)`.

Usage

```
bb_dat2
```

Format

A `data.frame` with 3 rows and 2 columns:

succ number of successes

fail number of failures

bb_pi *Simple uncalibrated prediction intervals for beta-binomial data*

Description

`bb_pi()` is a helper function that is internally called by `beta_bin_pi()`. It calculates simple uncalibrated prediction intervals for binary data with overdispersion changing between the clusters (beta-binomial).

Usage

```
bb_pi(
  newsize,
  histsize,
  pi,
  rho,
  q = qnorm(1 - 0.05/2),
  alternative = "both",
  newdat = NULL,
  histdat = NULL,
  algorithm = NULL
)
```

Arguments

`newsize`	number of experimental units in the historical clusters
`histsize`	number of experimental units in the future clusters
`pi`	binomial proportion
`rho`	intra class correlation
`q`	quantile used for interval calculation
`alternative`	either "both", "upper" or "lower" `alternative` specifies, if a prediction interval or an upper or a lower prediction limit should be computed
`newdat`	additional argument to specify the current data set
`histdat`	additional argument to specify the historical data set
`algorithm`	used to define the algorithm for calibration if called via `beta_bin_pi()`. This argument is not of interest for the calculation of simple uncalibrated intervals

Details

This function returns a simple uncalibrated prediction interval

$$[l,u]_m = n_m^*\hat{\pi} \pm q\sqrt{n_m^*\hat{\pi}(1-\hat{\pi})[1+(n_m^*-1)\hat{\rho}] + [\frac{n_m^{*2}\hat{\pi}(1-\hat{\pi})}{\sum_h n_h} + \frac{\sum_h n_h - 1}{\sum_h n_h} n_m^{*2}\hat{\pi}(1-\hat{\pi})\hat{\rho}]}$$

with n_m^* as the number of experimental units in the $m = 1, 2, ..., M$ future clusters, $\hat{\pi}$ as the estimate for the binomial proportion obtained from the historical data, $\hat{\rho}$ as the estimate for the intra class correlation and n_h as the number of experimental units per historical cluster.

The direct application of this uncalibrated prediction interval to real life data is not recommended. Please use `beta_bin_pi()` for real life applications.

Value

`bb_pi()` returns an object of class `c("predint", "betaBinomialPI")` with prediction intervals or limits in the first entry (`$prediction`).

Examples

```
# Pointwise uncalibrated PI
bb_pred <- bb_pi(newsize=c(50), pi=0.3, rho=0.05, histsize=rep(50, 20), q=qnorm(1-0.05/2))
summary(bb_pred)
```

`beta_bin_pi` *Prediction intervals for beta-binomial data*

Description

`beta_bin_pi()` calculates bootstrap calibrated prediction intervals for beta-binomial data

Usage

```
beta_bin_pi(
  histdat,
  newdat = NULL,
  newsize = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22mod"
)
```

Arguments

histdat	a data.frame with two columns (number of successes and number of failures) containing the historical data
newdat	a data.frame with two columns (number of successes and number of failures) containing the future data
newsize	a vector containing the future cluster sizes
alternative	either "both", "upper" or "lower". alternative specifies if a prediction interval or an upper or a lower prediction limit should be computed
alpha	defines the level of confidence (1-alpha)
nboot	number of bootstraps
delta_min	lower start value for bisection
delta_max	upper start value for bisection
tolerance	tolerance for the coverage probability in the bisection
traceplot	if TRUE: Plot for visualization of the bisection process
n_bisec	maximal number of bisection steps
algorithm	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If algorithm is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u]_m = n_m^* \hat{\pi} \pm q^{calib} \hat{se}(Y_m - y_m^*)$$

where

$$\hat{se}(Y_m - y_m^*) = \sqrt{n_m^* \hat{\pi}(1-\hat{\pi})[1 + (n_m^* - 1)\hat{\rho}] + [\frac{n_m^{*2}\hat{\pi}(1-\hat{\pi})}{\sum_h n_h} + \frac{\sum_h n_h - 1}{\sum_h n_h} n_m^{*2}\hat{\pi}(1-\hat{\pi})\hat{\rho}]}$$

with n_m^* as the number of experimental units in the future clusters, $\hat{\pi}$ as the estimate for the binomial proportion obtained from the historical data, q^{calib} as the bootstrap-calibrated coefficient, $\hat{\rho}$ as the estimate for the intra class correlation (Lui et al. 2000) and n_h as the number of experimental units per historical cluster.

If algorithm is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u]_m = \left[n_m^* \hat{\pi} - q_l^{calib} \hat{se}(Y_m - y_m^*), \quad n_m^* \hat{\pi} + q_u^{calib} \hat{se}(Y_m - y_m^*)\right]$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

Value

`beta_bin_pi` returns an object of class `c("predint", "betaBinomialPI")` with prediction intervals or limits in the first entry (`$prediction`).

References

Lui et al. (2000): Confidence intervals for the risk ratio under cluster sampling based on the beta-binomial model. Statistics in Medicine.
doi:10.1002/10970258(20001115)19:21<2933::AIDSIM591>3.0.CO;2Q

Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica. doi:10.1111/stan.12260

Examples

```
# Prediction interval
pred_int <- beta_bin_pi(histdat=mortality_HCD, newsize=40, nboot=100)
summary(pred_int)

# Upper prediction bound
pred_u <- beta_bin_pi(histdat=mortality_HCD, newsize=40, alternative="upper", nboot=100)
summary(pred_u)

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

---

bisection *Bisection algorithm for bootstrap calibration of prediction intervals*

Description

This helper function returns a bootstrap calibrated coefficient for the calculation of prediction intervals (and limits).

Usage

```
bisection(
  y_star_hat,
  pred_se,
  y_star,
  alternative,
  quant_min,
  quant_max,
  n_bisec,
  tol,
  alpha,
  traceplot = TRUE
)
```

Arguments

`y_star_hat`	a list of length B that contains the expected future observations. Each entry in this list has to be a numeric vector of length M.
`pred_se`	a list of length B that contains the standard errors of the prediction. Each entry in this list has to be a numeric vector of length M.
`y_star`	a list of length B that contains the future observations. Each entry in this list has to be a numeric vector of length M.
`alternative`	either "both", "upper" or "lower". `alternative` specifies if a prediction interval or an upper or a lower prediction limit should be computed
`quant_min`	lower start value for bisection
`quant_max`	upper start value for bisection
`n_bisec`	maximal number of bisection steps
`tol`	tolerance for the coverage probability in the bisection
`alpha`	defines the level of confidence $(1 - \alpha)$
`traceplot`	if `TRUE`: Plot for visualization of the bisection process

Details

This function is an implementation of the bisection algorithm of Menssen and Schaarschmidt 2022. It returns a calibrated coefficient q^{calib} for the calculation of pointwise and simultaneous prediction intervals

$$[l, u] = \hat{y}^*_m \pm q^{calib} \hat{se}(Y_m - y^*_m),$$

lower prediction limits

$$l = \hat{y}^*_m - q^{calib} \hat{se}(Y_m - y^*_m)$$

or upper prediction limits

$$u = \hat{y}^*_m + q^{calib} \hat{se}(Y_m - y^*_m)$$

that cover all of $m = 1, ..., M$ future observations.

In this notation, $\hat{y}^*_m$ are the expected future observations for each of the m future clusters, q^{calib} is the calibrated coefficient and $\hat{se}(Y_m - y^*_m)$ are the standard errors of the prediction.

Value

This function returns q^{calib} in the equation above.

References

Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica.
doi:10.1111/stan.12260

boot_predint *Bootstrap new data from uncalibrated prediction intervals*

Description

`boot_predint()` is a helper function to bootstrap new data from the simple uncalibrated prediction intervals implemented in predint.

Usage

```
boot_predint(pred_int, nboot, adjust = "within")
```

Arguments

`pred_int`	object of class `c("quasiPoissonPI", "betaBinomialPI","quasiBinomialPI", negativeBinomialPI)`
`nboot`	number of bootstraps
`adjust`	specifies if simultaneous prediction should be done for several control groups of different studies (`between`), or for the outcome of the current control and some treatment groups `within` the same trial

Details

This function only works for binomial and Poisson type data. For the sampling of new data from random effects models see `lmer_bs`.

Value

`boot_predint` returns an object of class `c("predint", "bootstrap")` which is a list with two entries: One for bootstrapped historical observations and one for bootstrapped future observations.

Examples

```
# Simple quasi-Poisson PI
test_pi <- qp_pi(histoffset=c(3,3,3,4,5), newoffset=3, lambda=10, phi=3, q=1.96)

# Draw 5 bootstrap samples
test_boot <- boot_predint(pred_int = test_pi, nboot=50)
str(test_boot)
summary(test_boot)

# Please note that the low number of bootstrap samples was chosen in order to
# decrease computing time. For valid analysis draw at least 10000 bootstrap samples.
```

c2_dat1 *Cross-classified data (example 1)*

Description

c2_dat1 contains data that is sampled from a balanced cross-classified design. This data set is used in order to demonstrate the functionality of the `lmer_pi_...()` functions.

Usage

```
c2_dat1
```

Format

A `data.frame` with 27 rows and 3 columns:

y_ijk observations

a treatment a

b treatment b

c2_dat2 *Cross-classified data (example 2)*

Description

c2_dat2 contains data that was sampled from an unbalanced cross-classified design. This data set is used in order to demonstrate the functionality of the `lmer_pi_...()` functions.

Usage

```
c2_dat2
```

Format

A `data.frame` with 21 rows and 3 columns:

y_ijk observations

a treatment a

b treatment b

c2_dat3 *Cross-classified data (example 3)*

Description

c2_dat3 contains data that was sampled from a balanced cross-classified design. This data set is used in order to demonstrate the functionality of the `lmer_pi_...()` functions.

Usage

```
c2_dat3
```

Format

A `data.frame` with 8 rows and 3 columns:

y_ijk observations

a treatment a

b treatment b

c2_dat4 *Cross-classified data (example 4)*

Description

c2_dat4 contains data that was sampled from an unbalanced cross-classified design. This data set is used in order to demonstrate the functionality of the `lmer_pi_...()` functions.

Usage

```
c2_dat4
```

Format

A `data.frame` with 6 rows and 3 columns:

y_ijk observations

a treatment a

b treatment b

lmer_bs *Sampling of bootstrap data from a given random effects model*

Description

`lmer_bs()` draws bootstrap samples based on the estimates for the mean and the variance components drawn from a random effects model fit with `lme4::lmer()`. Contrary to `lme4::bootMer()`, the number of observations for each random factor can vary between the original data set and the bootstrapped data. Random effects in `model` have to be specified as `(1|random effect)`.

Usage

```
lmer_bs(model, newdat = NULL, futmat_list = NULL, nboot)
```

Arguments

`model`	a random effects model of class lmerMod
`newdat`	a `data.frame` with the same column names as the historical data on which `model` depends
`futmat_list`	a list that contains design matrices for each random factor
`nboot`	number of bootstrap samples

Details

The data sampling is based on a list of design matrices (one for each random factor) that can be obtained if `newdat` and the model formula are provided to `lme4::lFormula()`. Hence, each random factor that is part of the initial model must have at least two replicates in `newdat`.
If a random factor in the future data set does not have any replicate, a list that contains design matrices (one for each random factor) can be provided via `futmat_list`.

Value

A list of length `nboot` containing the bootstrapped observations.

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat1

fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

#------------------------------------------------------------------------

### Using c2_dat2 as newdat
```

```
c2_dat2

lmer_bs(model=fit, newdat=c2_dat2, nboot=100)

#------------------------------------------------------------------------------

### Using futmat_list

# c2_dat4 has no replication for b. Hence the list of design matrices can not be
# generated by lme4::lFormula() and have to be provided by hand via futmat_list.

c2_dat4

# Build a list containing the design matrices

fml <- vector(length=4, "list")

names(fml) <- c("a:b", "b", "a", "Residual")

fml[["a:b"]] <- matrix(nrow=6, ncol=2, data=c(1,1,0,0,0,0, 0,0,1,1,1,1))

fml[["b"]] <- matrix(nrow=6, ncol=1, data=c(1,1,1,1,1,1))

fml[["a"]] <- matrix(nrow=6, ncol=2, data=c(1,1,0,0,0,0, 0,0,1,1,1,1))

fml[["Residual"]] <- diag(6)

fml

lmer_bs(model=fit, futmat_list=fml, nboot=100)
```

`lmer_pi` *Prediction intervals for future observations based on linear random effects models (DEPRECATED)*

Description

This function is deprecated. Please use `lmer_pi_unstruc()`, `lmer_pi_futvec()` or `lmer_pi_futmat()`.

Usage

```
lmer_pi(
  model,
  newdat = NULL,
  m = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
```

```
  lambda_min = 0.01,
  lambda_max = 10,
  traceplot = TRUE,
  n_bisec = 30
)
```

Arguments

model	a random effects model of class "lmerMod"
newdat	a data.frame with the same column names as the historical data on which the model depends
m	number of future observations
alternative	either "both", "upper" or "lower". alternative specifies if a prediction interval or an upper or a lower prediction limit should be computed
alpha	defines the level of confidence (1-alpha)
nboot	number of bootstraps
lambda_min	lower start value for bisection
lambda_max	upper start value for bisection
traceplot	if TRUE: plot for visualization of the bisection process
n_bisec	maximal number of bisection steps

Details

This function returns a bootstrap calibrated prediction interval

$$[l, u] = \hat{y} \pm q\sqrt{v\hat{a}r(\hat{y} - y)}$$

with $\hat{y}$ as the predicted future observation, y as the observed future observations, $\sqrt{v\hat{a}r(\hat{y} - y)}$ as the prediction standard error and q as the bootstrap calibrated coefficient that approximates a quantile of the multivariate t-distribution.
Please note that this function relies on linear random effects models that are fitted with lmer() from the lme4 package. Random effects have to be specified as (1|random_effect).

Value

If newdat is specified: A data.frame that contains the future data, the historical mean (hist_mean), the calibrated coefficient (quant_calib), the prediction standard error (pred_se), the prediction interval (lower and upper) and a statement if the prediction interval covers the future observation (cover).

If m is specified: A data.frame that contains the number of future observations (m) the historical mean (hist_mean), the calibrated coefficient (quant_calib), the prediction standard error (pred_se) and the prediction interval (lower and upper).

If alternative is set to "lower": Lower prediction limits are computed instead of a prediction interval.

If alternative is set to "upper": Upper prediction limits are computed instead of a prediction interval.

If traceplot=TRUE, a graphical overview about the bisection process is given.

Examples

```
# This function is deprecated.
# Please use lmer_pi_unstruc() if you want exactly the same functionality.
# Please use lmer_pi_futmat() or lmer_pi_futvec() if you want to take care
# of the future experimental design
```

`lmer_pi_futmat` *Prediction intervals for future observations based on linear random effects models*

Description

`lmer_pi_futmat()` calculates a bootstrap calibrated prediction interval for one or more future observation(s) based on linear random effects models. With this approach, the experimental design of the future data is taken into account (see below).

Usage

```
lmer_pi_futmat(
  model,
  newdat = NULL,
  futmat_list = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22"
)
```

Arguments

`model`	a random effects model of class `"lmerMod"`
`newdat`	either 1 or a `data.frame` with the same column names as the historical data on which `model` depends
`futmat_list`	a list that contains design matrices for each random factor
`alternative`	either "both", "upper" or "lower". `alternative` specifies if a prediction interval or an upper or a lower prediction limit should be computed
`alpha`	defines the level of confidence (1-`alpha`)
`nboot`	number of bootstraps
`delta_min`	lower start value for bisection

delta_max	upper start value for bisection
tolerance	tolerance for the coverage probability in the bisection
traceplot	if TRUE: Plot for visualization of the bisection process
n_bisec	maximal number of bisection steps
algorithm	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If algorithm is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u] = \hat{\mu} \pm q^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}$$

with $\hat{\mu}$ as the expected future observation (historical mean) and $\hat{\sigma}_c^2$ as the $c = 1, 2, ..., C$ variance components and $\hat{\sigma}_{C+1}^2$ as the residual variance obtained from the random effects model fitted with lme4::lmer() and q^{calib} as the as the bootstrap-calibrated coefficient used for interval calculation.

If algorithm is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u] = \left[\hat{\mu} - q_l^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}, \quad \hat{\mu} + q_u^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}\right].$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

If newdat is defined, the bootstrapped future observations used for the calibration process mimic the structure of the data set provided via newdat. The data sampling is based on a list of design matrices (one for each random factor) that can be obtained if newdat and the model formula are provided to lme4::lFormula(). Hence, each random factor that is part of the initial model must have at least two replicates in newdat.
If a random factor in the future data set does not have any replicate, a list that contains design matrices (one for each random factor) can be provided via futmat_list.

This function is an implementation of the PI given in Menssen and Schaarschmidt 2022 section 3.2.4, except, that the bootstrap calibration values are drawn from bootstrap samples that mimic the future data as described above.

Value

lmer_pi_futmat() returns an object of class c("predint", "normalPI") with prediction intervals or limits in the first entry ($prediction).

References

Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica, doi:10.1111/stan.12260

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

#------------------------------------------------------------------------
### Using newdat

# Prediction interval using c2_dat2 as future data
pred_int <- lmer_pi_futmat(model=fit, newdat=c2_dat2, alternative="both", nboot=100)
summary(pred_int)

# Upper prediction limit for m=1 future observations
pred_u <- lmer_pi_futmat(model=fit, newdat=1, alternative="upper", nboot=100)
summary(pred_u)

#------------------------------------------------------------------------

### Using futmat_list

# c2_dat4 has no replication for b. Hence the list of design matrices can not be
# generated by lme4::lFormula() and have to be provided by hand via futmat_list.

c2_dat4

# Build a list containing the design matrices

fml <- vector(length=4, "list")

names(fml) <- c("a:b", "b", "a", "Residual")

fml[["a:b"]] <- matrix(nrow=6, ncol=2, data=c(1,1,0,0,0,0, 0,0,1,1,1,1))

fml[["b"]] <- matrix(nrow=6, ncol=1, data=c(1,1,1,1,1,1))

fml[["a"]] <- matrix(nrow=6, ncol=2, data=c(1,1,0,0,0,0, 0,0,1,1,1,1))

fml[["Residual"]] <- diag(6)

fml

# Please note, that the design matrix for the interaction term a:b is also
# provided even there is no replication for b, since it is assumed that
# both, the historical and the future data descent from the same data generating
```

```
# process.

# Calculate the PI
pred_fml <- lmer_pi_futmat(model=fit, futmat_list=fml, alternative="both", nboot=100)
summary(pred_fml)

#------------------------------------------------------------------------------

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

`lmer_pi_futvec` *Prediction intervals for future observations based on linear random effects models*

Description

`lmer_pi_futvec()` calculates a bootstrap calibrated prediction interval for one or more future observation(s) based on linear random effects models. With this approach, the experimental design of the future data is taken into account (see below).

Usage

```
lmer_pi_futvec(
  model,
  futvec,
  newdat = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22"
)
```

Arguments

`model`	a random effects model of class lmerMod
`futvec`	an integer vector that defines the structure of the future data based on the row numbers of the historical data. If `length(futvec)` is one, a PI for one future observation is computed
`newdat`	a `data.frame` with the same column names as the historical data on which `model` depends

alternative	either "both", "upper" or "lower". alternative specifies if a prediction interval or an upper or a lower prediction limit should be computed
alpha	defines the level of confidence (1-alpha)
nboot	number of bootstraps
delta_min	lower start value for bisection
delta_max	upper start value for bisection
tolerance	tolerance for the coverage probability in the bisection
traceplot	if TRUE: Plot for visualization of the bisection process
n_bisec	maximal number of bisection steps
algorithm	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If algorithm is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u] = \hat{\mu} \pm q^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}$$

with $\hat{\mu}$ as the expected future observation (historical mean) and $\hat{\sigma}_c^2$ as the $c = 1, 2, ..., C$ variance components and $\hat{\sigma}_{C+1}^2$ as the residual variance obtained from the random effects model fitted with lme4::lmer() and q^{calib} as the as the bootstrap-calibrated coefficient used for interval calculation.

If algorithm is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u] = \left[\hat{\mu} - q_l^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}, \quad \hat{\mu} + q_u^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}\right].$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

Be aware that the sampling structure of the historical data must contain the structure of the future data. This means that the observations per random factor must be less or equal in the future data compared to the historical data.

This function is an implementation of the PI given in Menssen and Schaarschmidt 2022 section 3.2.4 except that the bootstrap calibration values are drawn from bootstrap samples that mimic the future data.

Value

`lmer_pi_futvec()` returns an object of class `c("predint", "normalPI")` with prediction intervals or limits in the first entry (`$prediction`).

References

Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica, doi:10.1111/stan.12260

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

#-------------------------------------------------------------------------

### Prediction interval using c2_dat3 as future data
# without printing c2_dat3 in the output

# Row numbers of the historical data c2_dat1 that define the structure of
# the future data c2_dat3
futvec <- c(1, 2, 4, 5, 10, 11, 13, 14)

# Calculating the PI
pred_int <- lmer_pi_futvec(model=fit, futvec=futvec, nboot=100)
summary(pred_int)

#-------------------------------------------------------------------------

### Calculating the PI with c2_dat3 printed in the output
pred_int_new <- lmer_pi_futvec(model=fit, futvec=futvec, newdat=c2_dat3, nboot=100)
summary(pred_int_new)

#-------------------------------------------------------------------------

### Upper prediction limit for m=1 future observation
pred_u <- lmer_pi_futvec(model=fit, futvec=1, alternative="upper", nboot=100)
summary(pred_u)

#-------------------------------------------------------------------------

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

`lmer_pi_unstruc` *Prediction intervals for future observations based on linear random effects models*

Description

`lmer_pi_unstruc()` calculates a bootstrap calibrated prediction interval for one or more future observation(s) based on linear random effects models as described in section 3.2.4. of Menssen and Schaarschmidt (2022). Please note, that the bootstrap calibration used here does not consider the sampling structure of the future data, since the calibration values are drawn randomly from bootstrap data sets that have the same structure as the historical data.

Usage

```
lmer_pi_unstruc(
  model,
  newdat = NULL,
  m = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22"
)
```

Arguments

`model`	a random effects model of class lmerMod
`newdat`	a `data.frame` with the same column names as the historical data on which the model depends
`m`	number of future observations
`alternative`	either "both", "upper" or "lower". `alternative` specifies if a prediction interval or an upper or a lower prediction limit should be computed
`alpha`	defines the level of confidence (1-`alpha`)
`nboot`	number of bootstraps
`delta_min`	lower start value for bisection
`delta_max`	upper start value for bisection
`tolerance`	tolerance for the coverage probability in the bisection
`traceplot`	if `TRUE`: Plot for visualization of the bisection process
`n_bisec`	maximal number of bisection steps
`algorithm`	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If `algorithm` is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u] = \hat{\mu} \pm q^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}$$

with $\hat{\mu}$ as the expected future observation (historical mean) and $\hat{\sigma}_c^2$ as the $c = 1, 2, ..., C$ variance components and $\hat{\sigma}_{C+1}^2$ as the residual variance obtained from the random effects model fitted with `lme4::lmer()` and q^{calib} as the as the bootstrap-calibrated coefficient used for interval calculation.

If `algorithm` is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u] = \left[\hat{\mu} - q_l^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}, \quad \hat{\mu} + q_u^{calib} \sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}\right].$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

This function is an direct implementation of the PI given in Menssen and Schaarschmidt 2022 section 3.2.4.

Value

`lmer_pi_futvec()` returns an object of class `c("predint", "normalPI")` with prediction intervals or limits in the first entry (`$prediction`).

References

Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica, doi:10.1111/stan.12260

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

# Prediction interval using c2_dat2 as future data
```

```
pred_int <- lmer_pi_unstruc(model=fit, newdat=c2_dat2, alternative="both", nboot=100)
summary(pred_int)

# Upper prediction limit for m=3 future observations
pred_u <- lmer_pi_unstruc(model=fit, m=3, alternative="upper", nboot=100)
summary(pred_u)

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

`mortality_HCD` *Historical mortality of male B6C3F1-mice*

Description

This data set contains historical control data about the mortality of male B6C3F1-mice obtained in long term carcinogenicity studies at the National Toxicology Program presented in NTP Historical Control Reports from 2013 to 2016. It was used in Menssen and Schaarschmidt 2019 as a real life example.

Usage

```
mortality_HCD
```

Format

A `data.frame` with 2 rows and 10 columns:

dead no. of dead mice

alive no. of living mice

References

Menssen and Schaarschmidt (2019): Prediction intervals for overdispersed binomial data with application to historical controls. Statistics in Medicine. doi:10.1002/sim.8124
NTP Historical Control Reports: https://ntp.niehs.nih.gov/data/controls

nb_pi *Simple uncalibrated prediction intervals for negative-binomial data*

Description

`nb_pi()` is a helper function that is internally called by `neg_bin_pi()`. It calculates simple uncalibrated prediction intervals for negative-binomial data with offsets.

Usage

```
nb_pi(
  newoffset,
  histoffset,
  lambda,
  kappa,
  q = qnorm(1 - 0.05/2),
  alternative = "both",
  newdat = NULL,
  histdat = NULL,
  algorithm = NULL
)
```

Arguments

`newoffset`	number of experimental units in the future clusters
`histoffset`	number of experimental units in the historical clusters
`lambda`	overall Poisson mean
`kappa`	dispersion parameter
`q`	quantile used for interval calculation
`alternative`	either "both", "upper" or "lower". `alternative` specifies, if a prediction interval or an upper or a lower prediction limit should be computed
`newdat`	additional argument to specify the current data set
`histdat`	additional argument to specify the historical data set
`algorithm`	used to define the algorithm for calibration if called via `quasi_pois_pi()`. This argument is not of interest for the calculation of simple uncalibrated intervals

Details

This function returns a simple uncalibrated prediction interval

$$[l,u]_m = n_m^*\hat{\lambda} \pm q\sqrt{n_m^*\frac{\hat{\lambda}+\hat{\kappa}\bar{n}\hat{\lambda}}{\bar{n}H} + (n_m^*\hat{\lambda} + \hat{\kappa}n_m^{*2}\hat{\lambda}^2)}$$

with n_m^* as the number of experimental units in $m = 1, 2, ..., M$ future clusters, $\hat{\lambda}$ as the estimate for the Poisson mean obtained from the historical data, $\hat{\kappa}$ as the estimate for the dispersion parameter, n_h as the number of experimental units per historical cluster and $\bar{n} = \sum_h^{n_h} n_h/H$.

The direct application of this uncalibrated prediction interval to real life data is not recommended. Please use the `neg_bin_pi()` function for real life applications.

Value

`np_pi` returns an object of class `c("predint", "negativeBinomialPI")`.

Examples

```
# Prediction interval
nb_pred <- nb_pi(newoffset=3, lambda=3, kappa=0.04, histoffset=1:9, q=qnorm(1-0.05/2))
summary(nb_pred)
```

`neg_bin_pi` *Prediction intervals for negative-binomial data*

Description

`neg_bin_pi()` calculates bootstrap calibrated prediction intervals for negative-binomial data.

Usage

```
neg_bin_pi(
  histdat,
  newdat = NULL,
  newoffset = NULL,
  alternative = "both",
  adjust = "within",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22mod"
)
```

Arguments

histdat	a data.frame with two columns. The first has to contain the historical observations. The second has to contain the number of experimental units per study (offsets).
newdat	data.frame with two columns. The first has to contain the future observations. The second has to contain the number of experimental units per study (offsets).
newoffset	vector with future number of experimental units per historical study.
alternative	either "both", "upper" or "lower". alternative specifies if a prediction interval or an upper or a lower prediction limit should be computed
adjust	specifies if simultaneous prediction should be done for several control groups of different studies (between), or for the outcome of the current control and some treatment groups within the same trial
alpha	defines the level of confidence $(1 - \alpha)$
nboot	number of bootstraps
delta_min	lower start value for bisection
delta_max	upper start value for bisection
tolerance	tolerance for the coverage probability in the bisection
traceplot	if TRUE: Plot for visualization of the bisection process
n_bisec	maximal number of bisection steps
algorithm	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If algorithm is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u]_m = n_m^* \hat{\lambda} \pm q \sqrt{n_m^* \frac{\hat{\lambda} + \hat{\kappa}\bar{n}\hat{\lambda}}{\bar{n}H} + (n_m^* \hat{\lambda} + \hat{\kappa} n_m^{*2} \hat{\lambda}^2)}$$

with n_m^* as the number of experimental units in the future clusters, $\hat{\lambda}$ as the estimate for the Poisson mean obtained from the historical data, $\hat{\kappa}$ as the estimate for the dispersion parameter, n_h as the number of experimental units per historical cluster and $\bar{n} = \sum_h^{n_h} n_h / H$.

If algorithm is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u] = \left[n_m^* \hat{\lambda} - q_l^{calib} \sqrt{n_m^* \frac{\hat{\lambda} + \hat{\kappa}\bar{n}\hat{\lambda}}{\bar{n}H} + (n_m^* \hat{\lambda} + \hat{\kappa} n_m^{*2} \hat{\lambda}^2)}, \quad n_m^* \hat{\lambda} + q_u^{calib} \sqrt{n_m^* \frac{\hat{\lambda} + \hat{\kappa}\bar{n}\hat{\lambda}}{\bar{n}H} + (n_m^* \hat{\lambda} + \hat{\kappa} n_m^{*2} \hat{\lambda}^2)} \right]$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

Value

`neg_bin_pi()` returns an object of class `c("predint", "negativeBinomialPI")` with prediction intervals or limits in the first entry (`$prediction`).

References

Menssen et al. (2025): Prediction Intervals for Overdispersed Poisson Data and Their Application in Medical and Pre-Clinical Quality Control. Pharmaceutical Statistics doi:10.1002/pst.2447

Examples

```
# HCD from the Ames test
ames_HCD

# Pointwise prediction interval for one future number of revertant colonies
# obtained in three petridishes
pred_int <- neg_bin_pi(histdat=ames_HCD, newoffset=3, nboot=100)
summary(pred_int)

# Simultaneous prediction interval for the numbers of revertant colonies obtained in
# the control and three treatment groups of a future trial
pred_int_w <- neg_bin_pi(histdat=ames_HCD, newoffset=c(3, 3, 3, 3), adjust="within", nboot=100)
summary(pred_int_w)

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

---

normal_pi *Simple uncalibrated prediction intervals for normal distributed data*

Description

`normal_pi()` is a helper function that is internally called by the `lmer_pi_...()` functions. It calculates simple uncalibrated prediction intervals for normal distributed observations.

Usage

```
normal_pi(
  mu,
  pred_se,
  m = 1,
  q = qnorm(1 - 0.05/2),
  alternative = "both",
  futmat_list = NULL,
  futvec = NULL,
  newdat = NULL,
  histdat = NULL,
  algorithm = NULL
)
```

Arguments

mu	overall mean
pred_se	standard error of the prediction
m	number of future observations
q	quantile used for interval calculation
alternative	either "both", "upper" or "lower" alternative specifies, if a prediction interval or an upper or a lower prediction limit should be computed
futmat_list	used to add the list of future design matrices to the output if called via lmer_pi_futmat()
futvec	used to add the vector of the historical row numbers that define the future experimental design to the output if called via lmer_pi_futmat()
newdat	additional argument to specify the current data set
histdat	additional argument to specify the historical data set
algorithm	used to define the algorithm for calibration if called via lmer_pi_...(). This argument is not of interest for the calculation of simple uncalibrated intervals

Details

This function returns a simple uncalibrated prediction interval as given in Menssen and Schaarschmidt 2022

$$[l, u] = \hat{\mu} \pm q\sqrt{\widehat{var}(\hat{\mu}) + \sum_{c=1}^{C+1} \hat{\sigma}_c^2}$$

with $\hat{\mu}$ as the expected future observation (historical mean) and $\hat{\sigma}_c^2$ as the $c = 1, 2, ..., C$ variance components and $\hat{\sigma}_{C+1}^2$ as the residual variance and q as the quantile used for interval calculation.

The direct application of this uncalibrated prediction interval to real life data is not recommended. Please use the lmer_pi_...() functions for real life applications.

Value

normal_pi() returns an object of class c("predint", "normalPI") with prediction intervals or limits in the first entry ($prediction).

References

Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica, doi:10.1111/stan.12260

Examples

```
# simple PI
norm_pred <- normal_pi(mu=10, pred_se=3, m=1)
summary(norm_pred)
```

pi_rho_est *Estimation of the binomial proportion and the intra class correlation.*

Description

`pi_rho_est()` estimates the overall binomial proportion $\hat{\pi}$ and the intra class correlation $\hat{\rho}$ of data that is assumed to follow the beta-binomial distribution. The estimation of $\hat{\pi}$ and $\hat{\rho}$ is done following the approach of Lui et al. 2000.

Usage

```
pi_rho_est(dat)
```

Arguments

`dat`	a `data.frame` with two columns (successes and failures)

Value

a vector containing estimates for π and ρ

References

Lui, K.-J., Mayer, J.A. and Eckhardt, L: Confidence intervals for the risk ratio under cluster sampling based on the beta-binomial model. Statistics in Medicine.2000;19:2933-2942. doi:10.1002/10970258(20001115)19:21<2933::AIDSIM591>3.0.CO;2Q

Examples

```
# Estimates for bb_dat1
pi_rho_est(bb_dat1)
```

plot.predint *Plots of* `predint` *objects*

Description

This function provides methodology for plotting the prediction intervals or limits that are calculated using the functionality of the **predint** package.

Usage

```
## S3 method for class 'predint'
plot(x, ..., size = 4, width = 0.05, alpha = 0.5)
```

Arguments

x	object of class predint
...	arguments handed over to ggplot2::aes()
size	size of the dots
width	margin of jittering
alpha	opacity of dot colors

Value

Since plot.predint() is based on ggplot2::ggplot, it returns an object of class c("gg", "ggplot").

Examples

```
### PI for quasi-Poisson data
pred_int <- quasi_pois_pi(histdat=ames_HCD,
                          newoffset=3,
                          nboot=100,
                          traceplot = FALSE)

### Plot the PI
plot(pred_int)

### Since plot.predint is based on ggplot, the grafic can be altered using
# the methodology provided via ggplot2
plot(pred_int)+
     theme_classic()
```

---

print.predint *Print objects of class* predint

Description

Print objects of class predint

Usage

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

Arguments

x	an object of class predint
...	additional arguments passed over to base::cbind() and base::data.frame()

Value

prints output to the console

qb_dat1 *Quasi-binomial data (example 1)*

Description

This data set contains sampled quasi-binomial data from 10 clusters each of size 50. The data set was sampled with `rqbinom(n=10, size=50, prob=0.1, phi=3)`.

Usage

```
qb_dat1
```

Format

A `data.frame` with 3 rows and 2 columns:

succ numbers of success

fail numbers of failures

qb_dat2 *Quasi-binomial data (example 2)*

Description

This data set contains sampled quasi binomial data from 3 clusters with different size.The data set was sampled with `rqbinom(n=3, size=c(40, 50, 60), prob=0.1, phi=3)`.

Usage

```
qb_dat2
```

Format

A `data.frame` with 3 rows and 2 columns:

succ numbers of success

fail numbers of failures

qb_pi *Simple uncalibrated prediction intervals for quasi-binomial data*

Description

`qb_pi()` is a helper function that is internally called by `quasi_bin_pi()`. It calculates simple uncalibrated prediction intervals for binary data with constant overdispersion (quasi-binomial assumption).

Usage

```
qb_pi(
  newsize,
  histsize,
  pi,
  phi,
  q = qnorm(1 - 0.05/2),
  alternative = "both",
  newdat = NULL,
  histdat = NULL,
  algorithm = NULL
)
```

Arguments

`newsize`	number of experimental units in the historical clusters.
`histsize`	number of experimental units in the future clusters.
`pi`	binomial proportion
`phi`	dispersion parameter
`q`	quantile used for interval calculation
`alternative`	either "both", "upper" or "lower" `alternative` specifies, if a prediction interval or an upper or a lower prediction limit should be computed
`newdat`	additional argument to specify the current data set
`histdat`	additional argument to specify the historical data set
`algorithm`	used to define the algorithm for calibration if called via `quasi_bin_pi`. This argument is not of interest for the calculation of simple uncalibrated intervals

Details

This function returns a simple uncalibrated prediction interval

$$[l,u]_m = n_m^*\hat{\pi} \pm q\sqrt{\hat{\phi}n_m^*\hat{\pi}(1-\hat{\pi}) + \frac{\hat{\phi}n_m^{*2}\hat{\pi}(1-\hat{\pi})}{\sum_h n_h}}$$

with n_m^* as the number of experimental units in the $m = 1, 2, ..., M$ future clusters, $\hat{\pi}$ as the estimate for the binomial proportion obtained from the historical data, $\hat{\phi}$ as the estimate for the dispersion parameter and n_h as the number of experimental units per historical cluster.

The direct application of this uncalibrated prediction interval to real life data is not recommended. Please use the `beta_bin_pi()` functions for real life applications.

Value

`qb_pi` returns an object of class `c("predint", "quasiBinomailPI")`.

Examples

```
qb_pred <- qb_pi(newsize=50, pi=0.3, phi=3, histsize=c(50, 50, 30), q=qnorm(1-0.05/2))
summary(qb_pred)
```

`qp_dat1` *Quasi-Poisson data (example 1)*

Description

This data set contains sampled quasi-Poisson data for 10 clusters. The data set was sampled with `rqpois(n=10, lambda=50, phi=3)`.

Usage

```
qp_dat1
```

Format

A data.frame with two columns

y numbers of eventzs

offset size of experimental units

qp_dat2 *Quasi-Poisson data (example 2)*

Description

This data set contains sampled quasi-Poisson data for 3 clusters. The data set was sampled with `rqpois(n=3, lambda=50, phi=3)`.

Usage

```
qp_dat2
```

Format

A data.frame with two columns

y numbers of eventzs

offset size of experimental units

qp_pi *Simple uncalibrated prediction intervals for quasi-Poisson data*

Description

`qp_pi()` is a helper function that is internally called by `quasi_pois_pi()`. It calculates simple uncalibrated prediction intervals for Poisson data with constant overdispersion (quasi-Poisson assumption).

Usage

```
qp_pi(
  newoffset,
  histoffset,
  lambda,
  phi,
  q = qnorm(1 - 0.05/2),
  alternative = "both",
  adjust = NULL,
  newdat = NULL,
  histdat = NULL,
  algorithm = NULL
)
```

Arguments

newoffset	number of experimental units in the future clusters
histoffset	number of experimental units in the historical clusters
lambda	overall Poisson mean
phi	dispersion parameter
q	quantile used for interval calculation
alternative	either "both", "upper" or "lower" alternative specifies, if a prediction interval or an upper or a lower prediction limit should be computed
adjust	only important if called via quasi_pois_pi()
newdat	additional argument to specify the current data set
histdat	additional argument to specify the historical data set
algorithm	used to define the algorithm for calibration if called via quasi_pois_pi(). This argument is not of interest for the calculation of simple uncalibrated intervals

Details

This function returns a simple uncalibrated prediction interval

$$[l,u]_m = n_m^*\hat{\lambda} \pm q\sqrt{n_m^*\hat{\phi}\hat{\lambda} + \frac{n_m^{*2}\hat{\phi}\hat{\lambda}}{\sum_h n_h}}$$

with n_m^* as the number of experimental units in the $m = 1, 2, ..., M$ future clusters, $\hat{\lambda}$ as the estimate for the Poisson mean obtained from the historical data, $\hat{\phi}$ as the estimate for the dispersion parameter and n_h as the number of experimental units per historical cluster.

The direct application of this uncalibrated prediction interval to real life data is not recommended. Please use the quasi_pois_pi_pi() functions for real life applications.

Value

qp_pi returns an object of class c("predint", "quasiPoissonPI").

Examples

```
# Prediction interval
qp_pred <- qp_pi(newoffset=3, lambda=3, phi=3, histoffset=1:9, q=qnorm(1-0.05/2))
summary(qp_pred)
```

quasi_bin_pi *Prediction intervals for quasi-binomial data*

Description

quasi_bin_pi() calculates bootstrap calibrated prediction intervals for binomial data with constant overdispersion (quasi-binomial assumption).

Usage

```
quasi_bin_pi(
  histdat,
  newdat = NULL,
  newsize = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22mod"
)
```

Arguments

histdat	a data.frame with two columns (success and failures) containing the historical data
newdat	a data.frame with two columns (success and failures) containing the future data
newsize	a vector containing the future cluster sizes
alternative	either "both", "upper" or "lower". alternative specifies if a prediction interval or an upper or a lower prediction limit should be computed
alpha	defines the level of confidence (1-alpha)
nboot	number of bootstraps
delta_min	lower start value for bisection
delta_max	upper start value for bisection
tolerance	tolerance for the coverage probability in the bisection
traceplot	if TRUE: Plot for visualization of the bisection process
n_bisec	maximal number of bisection steps
algorithm	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If `algorithm` is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u]_m = n_m^* \hat{\pi} \pm q^{calib} \hat{se}(Y_m - y_m^*)$$

where

$$\hat{se}(Y_m - y_m^*) = \sqrt{\hat{\phi} n_m^* \hat{\pi}(1 - \hat{\pi}) + \frac{\hat{\phi} n_m^{*2} \hat{\pi}(1 - \hat{\pi})}{\sum_h n_h}}$$

with n_m^* as the number of experimental units in the future clusters, $\hat{\pi}$ as the estimate for the binomial proportion obtained from the historical data, q^{calib} as the bootstrap-calibrated coefficient, $\hat{\phi}$ as the estimate for the dispersion parameter and n_h as the number of experimental units per historical cluster.

If `algorithm` is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u] = \left[n_m^* \hat{\pi} - q_l^{calib} \hat{se}(Y_m - y_m^*), \quad n_m^* \hat{\pi} + q_u^{calib} \hat{se}(Y_m - y_m^*) \right]$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

Value

`quasi_bin_pi` returns an object of class `c("predint", "quasiBinomialPI")` with prediction intervals or limits in the first entry (`$prediction`).

References

Menssen and Schaarschmidt (2019): Prediction intervals for overdispersed binomial data with application to historical controls. Statistics in Medicine. doi:10.1002/sim.8124
Menssen and Schaarschmidt (2022): Prediction intervals for all of M future observations based on linear random effects models. Statistica Neerlandica, doi:10.1111/stan.12260

Examples

```
# Pointwise prediction interval
pred_int <- quasi_bin_pi(histdat=mortality_HCD, newsize=40, nboot=100)
summary(pred_int)

# Pointwise upper prediction limit
pred_u <- quasi_bin_pi(histdat=mortality_HCD, newsize=40, alternative="upper", nboot=100)
summary(pred_u)
```

```
# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

---

quasi_pois_pi *Prediction intervals for quasi-Poisson data*

Description

quasi_pois_pi() calculates bootstrap calibrated prediction intervals for Poisson data with constant overdispersion (quasi-Poisson).

Usage

```
quasi_pois_pi(
  histdat,
  newdat = NULL,
  newoffset = NULL,
  alternative = "both",
  adjust = "within",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30,
  algorithm = "MS22mod"
)
```

Arguments

histdat	a data.frame with two columns. The first has to contain the historical observations. The second has to contain the number of experimental units per study (offsets).
newdat	a data.frame with two columns. The first has to contain the future observations. The second has to contain the number of experimental units per study (offsets).
newoffset	vector with future number of experimental units per historical study.
alternative	either "both", "upper" or "lower". alternative specifies if a prediction interval or an upper or a lower prediction limit should be computed
adjust	specifies if simultaneous prediction should be done for several control groups of different studies (between), or for the outcome of the current control and some treatment groups within the same trial
alpha	defines the level of confidence $(1 - \alpha)$
nboot	number of bootstraps

`delta_min`	lower start value for bisection
`delta_max`	upper start value for bisection
`tolerance`	tolerance for the coverage probability in the bisection
`traceplot`	if `TRUE`: Plot for visualization of the bisection process
`n_bisec`	maximal number of bisection steps
`algorithm`	either "MS22" or "MS22mod" (see details)

Details

This function returns bootstrap-calibrated prediction intervals as well as lower or upper prediction limits.

If `algorithm` is set to "MS22", both limits of the prediction interval are calibrated simultaneously using the algorithm described in Menssen and Schaarschmidt (2022), section 3.2.4. The calibrated prediction interval is given as

$$[l, u]_m = n_m^* \hat{\lambda} \pm q^{calib} \sqrt{n_m^* \hat{\phi} \hat{\lambda} + \frac{n_m^{*2} \hat{\phi} \hat{\lambda}}{\sum_h n_h}}$$

with n_m^* as the number of experimental units in the future clusters, $\hat{\lambda}$ as the estimate for the Poisson mean obtained from the historical data, q^{calib} as the bootstrap-calibrated coefficient, $\hat{\phi}$ as the estimate for the dispersion parameter and n_h as the number of experimental units per historical cluster.

If `algorithm` is set to "MS22mod", both limits of the prediction interval are calibrated independently from each other. The resulting prediction interval is given by

$$[l, u] = \left[n_m^* \hat{\lambda} - q_l^{calib} \sqrt{n_m^* \hat{\phi} \hat{\lambda} + \frac{n_m^{*2} \hat{\phi} \hat{\lambda}}{\sum_h n_h}}, \quad n_m^* \hat{\lambda} + q_u^{calib} \sqrt{n_m^* \hat{\phi} \hat{\lambda} + \frac{n_m^{*2} \hat{\phi} \hat{\lambda}}{\sum_h n_h}} \right]$$

Please note, that this modification does not affect the calibration procedure, if only prediction limits are of interest.

Value

`quasi_pois_pi` returns an object of class `c("predint", "quasiPoissonPI")` with prediction intervals or limits in the first entry (`$prediction`).

References

Menssen et al. (2025): Prediction Intervals for Overdispersed Poisson Data and Their Application in Medical and Pre-Clinical Quality Control. Pharmaceutical Statistics doi:10.1002/pst.2447

Examples

```
#' # Historical data
qp_dat1

# Future data
qp_dat2

# Pointwise prediction interval
pred_int <- quasi_pois_pi(histdat=ames_HCD, newoffset=3, nboot=100)
summary(pred_int)

# Pointwise upper prediction
pred_u <- quasi_pois_pi(histdat=ames_HCD, newoffset=3, alternative="upper", nboot=100)
summary(pred_u)

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

---

rbbinom *Sampling of beta-binomial data*

Description

`rbbinom()` samples beta-binomial data according to Menssen and Schaarschmidt (2019).

Usage

```
rbbinom(n, size, prob, rho)
```

Arguments

`n`	defines the number of clusters (i)
`size`	integer vector defining the number of trials per cluster (n_i)
`prob`	probability of success on each trial (π)
`rho`	intra class correlation (ρ)

Details

For beta binomial data with $i = 1, ...I$ clusters, the variance is

$$var(y_i) = n_i\pi(1-\pi)[1 + (n_i - 1)\rho]$$

with ρ as the intra class correlation coefficient

$$\rho = 1/(1 + a + b).$$

For the sampling $(a + b)$ is defined as

$$(a + b) = (1 - \rho)/\rho$$

where $a = \pi(a + b)$ and $b = (a + b) - a$. Then, the binomial proportions for each cluster are sampled from the beta distribution

$$\pi_i \sim Beta(a, b)$$

and the number of successes for each cluster are sampled to be

$$y_i \sim Bin(n_i, \pi_i).$$

In this parametrization $E(\pi_i) = \pi = a/(a + b)$ and $E(y_i) = n_i\pi$. Please note, that $1 + (n_i - 1)\rho$ is a constant if all cluster sizes are the same and hence, in this special case, also the quasi-binomial assumption is fulfilled.

Value

a `data.frame` with two columns (succ, fail)

References

Menssen M, Schaarschmidt F.: Prediction intervals for overdispersed binomial data with application to historical controls. Statistics in Medicine. 2019;38:2652-2663. doi:10.1002/sim.8124

Examples

```
# Sampling of example data
set.seed(234)
bb_dat1 <- rbbinom(n=10, size=50, prob=0.1, rho=0.06)
bb_dat1


set.seed(234)
bb_dat2 <- rbbinom(n=3, size=c(40, 50, 60), prob=0.1, rho=0.06)
bb_dat2
```

---

rnbinom *Sampling of negative binomial data*

Description

`rnbinom()` samples negative-binomial data. The following description of the sampling process is based on the parametrization used by Gsteiger et al. 2013.

Usage

```
rnbinom(n, lambda, kappa, offset = NULL)
```

Arguments

`n`	defines the number of clusters (I)
`lambda`	defines the overall Poisson mean (λ)
`kappa`	dispersion parameter (κ)
`offset`	defines the number of experimental units per cluster (n_i)

Details

The variance of the negative-binomial distribution is

$$var(Y_i) = n_i\lambda(1 + \kappa n_i\lambda).$$

Negative-biomial observations can be sampled based on predefined values of κ, λ and n_i:
Define the parameters of the gamma distribution as $a = \frac{1}{\kappa}$ and $b_i = \frac{1}{\kappa n_i \lambda}$. Then, sample the Poisson means for each cluster

$$\lambda_i \sim Gamma(a, b_i).$$

Finally, the observations y_i are sampled from the Poisson distribution

$$y_i \sim Pois(\lambda_i)$$

Value

`rnbinom()` returns a `data.frame` with two columns: `y` as the observations and `offset` as the number of offsets per observation.

References

Gsteiger, S., Neuenschwander, B., Mercier, F. and Schmidli, H. (2013): Using historical control information for the design and analysis of clinical trials with overdispersed count data. Statistics in Medicine, 32: 3609-3622. doi:10.1002/sim.5851

Examples

```
# Sampling of negative-binomial observations
# with different offsets
set.seed(123)
rnbinom(n=5, lambda=5, kappa=0.13, offset=c(3,3,2,3,2))
```

`rqbinom` *Sampling of overdispersed binomial data with constant overdispersion*

Description

rqbinom samples overdispersed binomial data with constant overdispersion from the beta-binomial distribution such that the quasi-binomial assumption is fulfilled.

Usage

```
rqbinom(n, size, prob, phi)
```

Arguments

`n`	defines the number of clusters (i)
`size`	integer vector defining the number of trials per cluster (n_i)
`prob`	probability of success on each trial (π)
`phi`	dispersion parameter (Φ)

Details

It is assumed that the dispersion parameter (Φ) is constant for all $i = 1, ...I$ clusters, such that the variance becomes

$$var(y_i) = \Phi n_i \pi (1 - \pi).$$

For the sampling $(a + b)_i$ is defined as

$$(a + b)_i = (\Phi - n_i)/(1 - \Phi)$$

where $a_i = \pi(a + b)_i$ and $b_i = (a + b)_i - a_i$. Then, the binomial proportions for each cluster are sampled from the beta distribution

$$\pi_i \sim Beta(a_i, b_i)$$

and the numbers of success for each cluster are sampled to be

$$y_i \sim Bin(n_i, \pi_i).$$

In this parametrization $E(\pi_i) = \pi$ and $E(y_i) = n_i\pi$. Please note, the quasi-binomial assumption is not in contradiction with the beta-binomial distribution if all cluster sizes are the same.

Value

a `data.frame` with two columns (succ, fail)

Examples

```
# Sampling of example data
set.seed(456)
qb_dat1 <- rqbinom(n=10, size=50, prob=0.1, phi=3)
qb_dat1

set.seed(456)
qb_dat2 <- rqbinom(n=3, size=c(40, 50, 60), prob=0.1, phi=3)
qb_dat2
```

---

rqpois *Sampling of overdispersed Poisson data with constant overdispersion*

Description

`rqpois()` samples overdispersed Poisson data with constant overdispersion from the negative-binomial distribution such that the quasi-Poisson assumption is fulfilled. The following description of the sampling process is based on the parametrization used by Gsteiger et al. 2013.

Usage

```
rqpois(n, lambda, phi, offset = NULL)
```

Arguments

`n`	defines the number of clusters (I)
`lambda`	defines the overall Poisson mean (λ)
`phi`	dispersion parameter (Φ)
`offset`	defines the number of experimental units per cluster (n_i)

Details

It is assumed that the dispersion parameter (Φ) is constant for all $i = 1, ...I$ clusters, such that the variance becomes

$$var(y_i) = \Phi n_i \lambda$$

For the sampling κ_i is defined as

$$\kappa_i = (\Phi - 1)/(n_i \lambda)$$

where $a_i = 1/\kappa_i$ and $b_i = 1/(\kappa_i n_i \lambda)$. Then, the Poisson means for each cluster are sampled from the gamma distribution

$$\lambda_i \sim Gamma(a_i, b_i)$$

and the observations per cluster are sampled to be

$$y_i \sim Pois(\lambda_i).$$

Please note, that the quasi-Poisson assumption is not in contradiction with the negative-binomial distribution, if the data structure is defined by the number of clusters only (which is the case here) and the offsets are all the same $n_h = n_h = n$.

Value

a data.frame containing the sampled observations and the offsets

References

Gsteiger, S., Neuenschwander, B., Mercier, F. and Schmidli, H. (2013): Using historical control information for the design and analysis of clinical trials with overdispersed count data. Statistics in Medicine, 32: 3609-3622. doi:10.1002/sim.5851

Examples

```
# set.seed(123)
qp_dat1 <- rqpois(n=10, lambda=50, phi=3)
qp_dat1

# set.seed(123)
qp_dat2 <- rqpois(n=3, lambda=50, phi=3)
qp_dat2
```

---

summary.predint *Summarizing objects of class* predint

Description

This function gives a summary about the prediction intervals (and limits) computed with **predint**.

Usage

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

Arguments

object	object of class predint
...	further arguments passed over to base::cbind() and base::data.frame()

Value

A data.frame containing the current data (if provided via newdat), the prediction interval (or limit), the expected value for the future observation, the bootstrap calibrated coefficient(s), the prediction standard error and a statement about the coverage for each future observation, if new observations were provided via newdat.

Examples

```
# Fitting a random effects model based on c2_dat1
fit <- lme4::lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)


# Prediction interval using c2_dat2 as future data
pred_int <- lmer_pi_futmat(model=fit, newdat=c2_dat2, alternative="both", nboot=100)
summary(pred_int)

#------------------------------------------------------------------------------

# Please note that nboot was set to 100 in order to decrease computing time
# of the example. For a valid analysis set nboot=10000.
```

Index