Package 'predint'

June 24, 2022

Type Package

Title Prediction Intervals

Version 1.1.1

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 lme4, stats, graphics, methods

RoxygenNote 7.1.2

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 3.5.0)

NeedsCompilation no

Author Max Menssen [aut, cre]

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

Repository CRAN

Date/Publication 2022-06-24 14:20:02 UTC

R topics documented:

`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

Examples

```
# Upper prediction limit for m=3 future number of successes
# that are based on cluster sizes 40, 50, 60 respectively
beta_bin_pi(histdat=bb_dat1, newsize=c(40, 50, 60), alternative="upper", nboot=100)

# 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_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

Examples

```
# Prediction interval using bb_dat2 as future data
beta_bin_pi(histdat=bb_dat1, newdat=bb_dat2, nboot=100)

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

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
)
```

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` plot for visualization of the bisection process

`n_bisec` maximal number of bisection steps

Details

This function returns bootstrap calibrated prediction intervals

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

with $\hat{y}_m$ as the predicted future number of successes for $m = 1, ..., M$ future clusters, y_m as the observed future number of successes, $\sqrt{v\hat{a}r(\hat{y}_m - y_m)}$ as the prediction standard error and q as the bootstrap calibrated coefficient that approximates a quantile from a multivariate normal distribution. Please note that the predicted future number of successes is based on the future cluster size n_m and the success probability estimated from the historical data π^{hist} such that $\hat{y}_m = \pi^{hist} n_m$. Hence, the prediction intervals $[l, u]_m$ are different for each of the m future clusters, if their size is not the same.

Value

If `newdat` is specified: A `data.frame` that contains the future data, the historical proportion (hist_prob), 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 `newsize` is specified: A `data.frame` that contains the future cluster sizes (total) the historical proportion (hist_prob), 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 bounds are computed instead of a prediction interval.

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

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

Examples

```
# Historical data
bb_dat1

# Future data
bb_dat2

# Prediction interval using bb_dat2 as future data
beta_bin_pi(histdat=bb_dat1, newdat=bb_dat2, nboot=100)

# Upper prediction bound for m=3 future numbers of success
# that are based on cluster sizes 40, 50, 60 respectively
beta_bin_pi(histdat=bb_dat1, newsize=c(40, 50, 60), alternative="upper", nboot=100)

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

c2_dat1 *Cross-classified data (example 1)*

Description

c2_dat1 contains data that is sampled from a balanced cross-classified design.

Usage

```
c2_dat1
```

Format

A data.frame with 27 rows and 3 columns:

y_ijk observations

a treatment a

b treatment b

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat_1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

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

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

c2_dat2 *Cross-classified data (example 2)*

Description

c2_dat2 contains data that was sampled from an unbalanced cross-classified design.

Usage

```
c2_dat2
```

Format

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

y_ijk observations

a treatment a

b treatment b

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat_1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

# Prediction interval using c2_dat2 as future data
lmer_pi_unstruc(model=fit, newdat=c2_dat2, alternative="both", nboot=100)

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

c2_dat3 *Cross-classified data (example 3)*

Description

c2_dat3 contains data that was sampled from a balanced cross-classified design.

Usage

```
c2_dat3
```

Format

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

y_ijk observations

a treatment a

b treatment b

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat_1
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
lmer_pi_futvec(model=fit, futvec=futvec, alternative="both", nboot=100)

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

c2_dat4 *Cross-classified data (example 4)*

Description

c2_dat4 contains data that was sampled from an unbalanced cross-classified design.

Usage

```
c2_dat4
```

Format

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

y_ijk observations

a treatment a

b treatment b

Examples

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat_1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

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

### Prediction interval using c2_dat4 as future data

# c2_dat4 has no replication for b. Hence the list of design matrices can not be
# generated by lme4::lFormula() and has 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 believed that
# both, the historical and the future data descent from the same data generating
# process.

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

# 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_bs — *Sampling of bootstrap data from a given random effects model fit with* `lme4::lmer()`

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 need 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*

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`	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

```
# loading lme4
library(lme4)

# Fitting a random effects model based on c2_dat_1
fit <- lmer(y_ijk~(1|a)+(1|b)+(1|a:b), c2_dat1)
summary(fit)

# Prediction interval using c2_dat2 as future data
lmer_pi(model=fit, newdat=c2_dat2, alternative="both", nboot=100)

# Upper prediction limit for m=3 future observations
lmer_pi(model=fit, m=3, alternative="upper", nboot=100)

# 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_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 sampling structure 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
)
```

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`	plot for visualization of the bisection process
`n_bisec`	maximal number of bisection steps

Details

This function returns a bootstrap calibrated prediction interval

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

with $\hat{y}$ as the predicted future observation, y as the observed future observations, $\sqrt{\hat{var}(\hat{y} - y)}$ 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)`.

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 2021 section 3.2.4 except that the bootstrap calibration values are drawn from bootstrap samples that mimic the future data as described above.

Value

If `newdat` is a `data.frame`: 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 `newdat=1`: A `data.frame` that contains a statement that m=1, the historical mean (hist_mean), the calibrated coefficient (quant_calib), the prediction standard error (pred_se) and the prediction interval (lower and upper).

If `futmat_list` is defined: 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.

References

Menssen, M., Schaarschmidt, F.: 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
lmer_pi_futmat(model=fit, newdat=c2_dat2, alternative="both", nboot=100)

# Upper prediction limit for m=1 future observations
lmer_pi_futmat(model=fit, newdat=1, alternative="upper", 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

# 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 believed that
# both, the historical and the future data descent from the same data generating
# process.

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

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

# 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 sampling structure 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
)
```

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` 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)`.

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 2021 section 3.2.4 except that the bootstrap calibration values are drawn from bootstrap samples that mimic the future data.

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 observations (cover).

If only `futvec` 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 `futvec` is set to 1, the PI is calculated for one future observation.

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.

References

Menssen, M., Schaarschmidt, F.: 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
lmer_pi_futvec(model=fit, futvec=futvec, alternative="both", nboot=100)

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

### Calculating the PI with c2_dat3 printed in the output
lmer_pi_futvec(model=fit, futvec=futvec, newdat=c2_dat3, alternative="both", nboot=100)

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

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

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

# 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 (2021). 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
)
```

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`	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.

References

Menssen and Schaarschmidt (2021): 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
lmer_pi_unstruc(model=fit, newdat=c2_dat2, alternative="both", nboot=100)

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

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

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)
```

qb_dat1 *Quasi-binomial data (example 1)*

Description

This data set contains sampled quasi-binomial data from 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

Examples

```
# Upper prediction limit for m=3 future observations
# that are based on cluster sizes 40, 50, 60 respectively
quasi_bin_pi(histdat=qb_dat1, newsize=c(40, 50, 60), alternative="upper", nboot=100)

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

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

Examples

```
# Prediction interval using qb_dat2 as future data
quasi_bin_pi(histdat=qb_dat1, newdat=qb_dat2, nboot=100)

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

---

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

An integer vector with ten entries containing quasi-Poisson data

Examples

```
# Upper prediction limit for m=3 future observations
quasi_pois_pi(histdat=data.frame(qp_dat1), m=3, alternative="upper", nboot=100)

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

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

An integer vector with three entries containing quasi-Poisson data

Examples

```
# Prediction interval using qp_dat2 as future data
quasi_pois_pi(histdat=data.frame(qp_dat1), newdat=data.frame(qp_dat2), nboot=100)

# 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_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
)
```

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` plot for visualization of the bisection process

`n_bisec` maximal number of bisection steps

Details

This function returns bootstrap calibrated prediction intervals

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

with $\hat{y}_m$ as the predicted future number of successes for $m = 1, ..., M$ future clusters, y_m as the observed future number of successes, $\sqrt{v\hat{a}r(\hat{y}_m - y_m)}$ as the prediction standard error and q as the bootstrap calibrated coefficient that approximates a quantile from a multivariate normal distribution. Please note that the predicted future number of successes is based on the future cluster size n_m and the success probability estimated from the historical data π^{hist} such that $\hat{y}_m = \pi^{hist} n_m$. Hence, the prediction intervals are different for each of the m future clusters, if their size is not the same.

Value

If `newdat` is specified: A `data.frame` that contains the future data, the the historical proportion (hist_prob), 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 `newsize` is specified: A `data.frame` that contains the future cluster sizes (total) the the historical proportion (hist_prob), 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 bounds are computed instead of a prediction interval.

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

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

Examples

```
# Prediction interval using qb_dat2 as future data
quasi_bin_pi(histdat=qb_dat1, newdat=qb_dat2, nboot=100)

# Upper prediction bound for m=3 future observations
# that are based on cluster sizes 40, 50, 60 respectively
quasi_bin_pi(histdat=qb_dat1, newsize=c(40, 50, 60), alternative="upper", nboot=100)

# 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,
  m = NULL,
  alternative = "both",
  alpha = 0.05,
  nboot = 10000,
  delta_min = 0.01,
  delta_max = 10,
  tolerance = 0.001,
  traceplot = TRUE,
  n_bisec = 30
)
```

Arguments

`histdat` a `data.frame` with one column containing the historical data

`newdat` a `data.frame` with one column containing the future data

`m` number of future clusters

`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` 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 error and q as the bootstrap calibrated coefficient that approximates a quantile of a multivariate normal distribution.

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 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 error (pred_se) and the prediction interval (lower and upper).

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

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

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

Examples

```
#' # Historical data
qp_dat1

# Future data
qp_dat2

# Prediction interval using bb_dat2 as future data
quasi_pois_pi(histdat=data.frame(qp_dat1), newdat=data.frame(qp_dat2), nboot=100)

# Upper prediction bound for m=3 future observations
quasi_pois_pi(histdat=data.frame(qp_dat1), m=3, alternative="upper", nboot=100)

# 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
```

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 succes 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)
```

Arguments

`n`	defines the number of clusters (i)
`lambda`	defines the overall poisson mean (λ)
`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

$$var(y_i) = \lambda(1 + \lambda\kappa) = \Phi\lambda.$$

For the sampling κ is defined as

$$\kappa = (\Phi - 1)/(\lambda)$$

where $a = 1/\kappa$ and $b = 1/(\kappa\lambda)$. Then, the poisson means for each cluster are sampled from the gamma distribution

$$\lambda_i \sim Gamma(a, b)$$

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), rather than by a complex randomization structure.

Value

a vector containing the sampled observations (one per cluster)

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. Statist. Med., 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
```

Index