Package: QGglmm (via r-universe)

September 18, 2024

Type Package

Title Estimate Quantitative Genetics Parameters from Generalised Linear Mixed Models

Version 0.7.5

Date 2020-01-07

Author Pierre de Villemereuil <pierre.devillemereuil@ephe.psl.eu>

Maintainer Pierre de Villemereuil <pierre.devillemereuil@ephe.psl.eu>

BugReports https://github.com/devillemereuil/qgglmm/issues

Description Compute various quantitative genetics parameters from a Generalised Linear Mixed Model (GLMM) estimates. Especially, it yields the observed phenotypic mean, phenotypic variance and additive genetic variance.

Imports cubature (>= 1.4)

License GPL-2

Repository https://devillemereuil.r-universe.dev

RemoteUrl https://github.com/devillemereuil/qgglmm

RemoteRef HEAD

RemoteSha 60efc4d54464df4cf8ca840235b826048be909de

Contents

`QGglmm-package` *Estimate Quantitative Genetics Parameters from Generalised Linear Mixed Models*

Description

Compute various quantitative genetics parameters from a Generalised Linear Mixed Model (GLMM) estimates. Especially, it yields the observed phenotypic mean, phenotypic variance and additive genetic variance.

Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

This package gives the values on the observed scale for several quantitative genetics parameter using estimates from a Generalised Linear Mixed Model (GLMM). If a fitness function is assumed or measured, it also predicts the evolutionary response to selection on the observed scale.

The two main functions of this package are `QGparams` and `QGpred`. The first allows to compute the quantitative genetics parameters on the observed scale for any given GLMM and its estimates. The second allows to compute a predicted response to evolution on the observed scale using GLMM estimates and an assumed/measured/inferred fitness function.

For some distribution/link models (e.g. Binomial/probit and Poisson and Negative Binomial with logartihm or square-root link), a closed form solutions of the integrals computed by this package are available. They are automatially used by `QGparams` and this function only.

Author(s)

Pierre de Villemereuil <pierre.devillemereuil@ephe.psl.eu>

Maintainer: Pierre de Villemereuil <pierre.devillemereuil@ephe.psl.eu>

References

de Villemereuil, P., Schielzeth, H., Nakagawa, S., and Morrissey, M.B. (2016). General methods for evolutionary quantitative genetic inference from generalised mixed models. Genetics 204, 1281-1294.

QGicc *Intra - Class Correlation coefficients (ICC) on the observed data scale*

Description

Function to estimate the Intra - Class Correlation coefficients (ICC, a.k.a. repeatability - like estimates) on the observed scale based on estimates on the latent scale. For a specific variance component, the function yields a data.frame which includes the phenotypic mean and variance, as well as the variance component and associated ICC, on the observed data scale.

Usage

```
QGicc(mu = NULL, var.comp, var.p, model = "", width = 10, predict = NULL,
closed.form = TRUE, custom.model = NULL, n.obs = NULL, theta = NULL, verbose = TRUE)
```

Arguments

mu	Latent intercept estimated from a GLMM (ignored if predict is not NULL). (numeric of length 1)
var.comp	Latent variance component for which ICC needs to be computed, estimated from a GLMM. (numeric of length 1)
var.p	Latent total phenotypic variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
model	Name of the used model, i.e. distribution.link. Ignored if custom.model is not NULL. See QGlink.funcs for a complete list of model available. (character)
width	Parameter for the integral computation. The integral is evaluated from -width * sqrt(var.comp) to width * sqrt(var.comp). The default value is 10, which should be sensible for most models. (numeric)
predict	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
closed.form	When available, should closed forms be used instead of integral computations? (boolean)
custom.model	If the model used is not available using the model argument, a list of functions describing the model can be provided. (list of functions, see Details)
n.obs	Number of "trials" for the "binomN" distribution. (numeric)
theta	Dispersion parameter for the Negative Binomial distribution. The parameter theta should be such as the variance of the distribution is mean + mean^2 / theta. (numeric)
verbose	Should the function be verbose? (boolean)

Details

The function typically uses precise integral numerical approximation to compute parameters on the observed scale, from latent estimates yielded by a GLMM. If closed form solutions for the integrals are available, it uses them if `closed.form = TRUE`.

Only the most typical distribution/link function couples are implemented in the function. If you used an "exotic" GLMM, you can use the `custom.model` argument. It should take the form of a list of functions. The first function should be the inverse of the link function named `inv.link`, the second function should be the "distribution variance" function named `var.func` and the third function should be the derivative of the inverse link function named `d.inv.link` (see Example below).

Some distributions require extra-arguments. This is the case for "binomN", which require the number of trials N, passed with the argument `n.obs`. The distribution "negbin" requires a dispersion parameter `theta`, such as the variance of the distribution is `mean + mean^2 / theta` (mean/dispersion parametrisation).

If fixed effects (apart from the intercept) have been included in the GLMM, they can be included as marginal predicted values, i.e. predicted values excluding the random effects, which can be calculated as the matrix product **Xb** where **X** is the design matrix and **b** is the vector of fixed effects estimates. To do so, provide the vector of marginal predicted values using the argument `predict`. Note this can considerably slow down the algorithm, especially when no closed form is used.

Value

The function yields a data.frame containing the following values:

`mean.obs`	Phenotypic mean on the observed scale.
`var.obs`	Phenotypic variance on the observed scale.
`var.comp.obs`	Component variance on the observed scale.
`icc.obs`	ICC on the observed scale.

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGparams`, `QGpred`, `QGlink.funcs`, `QGmean`, `QGvar.dist`, `QGvar.exp`, `QGpsi`

Examples

```
## Example using Poisson count data
# Parameters
mu <- 0
va <- 0.5
vm <- 0.2  # Maternal effect
vp <- 1

# Simulating data l = mu + a + e
lat <- mu +
```

```
       rnorm(1000, 0, sqrt(va)) +
       rnorm(1000, 0, sqrt(vm)) +
       rnorm(1000, 0, sqrt(vp - (va + vm)))
y   <- rpois(1000, exp(lat))

# Computing the broad - sense heritability
QGicc(mu = mu, var.p = vp, var.comp = va, model = "Poisson.log")
# Computing the maternal effect ICC
QGicc(mu = mu, var.p = vp, var.comp = vm, model = "Poisson.log")

# Using integral computation
QGicc(mu = mu, var.p = vp, var.comp = vm, model = "Poisson.log", closed.form = FALSE)
# Note that the "approximation" is exactly equal to the results obtained with the closed form

# Let's create a custom model
custom <- list(inv.link = function(x){exp(x)},
    var.func = function(x){exp(x)},
    d.inv.link = function(x){exp(x)})

QGicc(mu = mu, var.p = vp, var.comp = vm, custom.model = custom)
# Again, exactly equal

# Integrating over a posterior distribution
# e.g. output from MCMCglmm named "model"
# df <- data.frame(mu = model$Sol[, 'intercept'],
#                  vm = model$VCV[, 'mother'],
#                  vp = rowSums(model$VCV))
# params <- apply(df, 1, function(row){
#        QGicc(mu = row$mu, var.comp = row$vm, var.p = row$vp, model = "Poisson.log")
# })
```

`QGlink.funcs` *List of functions according to a distribution and a link function*

Description

Function yielding different functions (inverse-link, variance function, derivative of the inverse-link) according to a distribution and link function.

Usage

```
QGlink.funcs(name, n.obs = NULL, theta = NULL)
```

Arguments

`name`	Name of the distribution.link couple. (character) Available models are :

- "Gaussian" Gaussian distribution with identity link (e.g. LMM)
- "binom1.probit" Binomial with 1 trial (binary data) with a probit link

- "binomN.probit" Binomial with N tria with a probit link (require the parameter `n.obs`)
- "binom1.logit" Binomial with 1 trial (binary) with a logit link
- "binomN.logit" Binomial with N trial with a logit link (require the parameter `n.obs`)
- "Poisson.log" Poisson distribution with a log link
- "Poisson.sqrt" Poisson distribution with a square-root link
- "negbin.log" Negative - Binomial distribution wiht a log link (require the parameter `theta`)
- "negbin.sqrt" Negative - Binomial distribution with a square-root link (require the parameter `theta`)
- "ordinal" Multiple threshold model for ordinal categorical traits (require the parameter `cut.points`)

`n.obs` Optional parameter required for "binomN" distributions (number of "trials"). See `QGparams`. (numeric)

`theta` Optional parameter required for "negbin" distributions (dispersion parameter). See `QGparams`. (numeric)

Details

This function takes the name of a distribution.link couple and yields several important functions such as the inverse-link function and its derivative, as well as the "distribution variance function".

The inverse-link function is the inverse function of the link function. For example, if the link function is the natural logarithm (typically for a Poisson distribution), then the inverse-link function is the exponential.

The distribution variance function is a function yielding the variance of the distribution for a given latent trait. For a Poisson distribution, the variance is equal to the mean, hence the variance function is equal to the inverse-link function. For a binomial distribution, the variance is N * p(l) * (1 - p(l)), where p is the inverse-link function.

For some distributions, such as "binomN" and "negbin", some extra-parameters are required.

Value

This function yields a list of function:

- `inv.link`Inverse function of the link function. (function)
- `var.func`Distribution variance function. (function)
- `inv.link`Derivative of the inverse-link function. (function)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGparams`, `QGmvparams`

Examples

```
## Getting the functions for a Poisson.log model
QGlink.funcs("Poisson.log")
# Note that because the variance is equal to the mean in a Poisson distribution
# and the derivative of exp is exp
# all functions are the same!

## Getting the functions for a binom1.probit model
QGlink.funcs("binom1.probit")

## The function QGparams automatically computes these functions
QGparams(mu = 0, var.p = 2, var.a = 1, model = "binom1.logit")
# Hence this is the same as using the custom.model argument with QGlink.funcs
QGparams(mu = 0, var.p = 2, var.a = 1, custom.model = QGlink.funcs("binom1.logit"))

## We can create our own custom set of functions
# Let's create a custom model exactly identical to QGlink.funcs("binom1.logit")
custom <- list(inv.link = function(x){plogis(x)},
               var.func = function(x){plogis(x) * (1 - plogis(x))},
               d.inv.link = function(x){dlogis(x)})

QGparams(mu = 0, var.p = 2, var.a = 1, custom.model = custom)
```

`QGmean` *Compute the phenotypic mean on the observed scale*

Description

This function calculates the phenotypic mean on the observed scale from the latent mean and variance.

Usage

```
QGmean(mu = NULL, var, link.inv, predict = NULL, width = 10)
```

Arguments

`mu`	Latent intercept estimated from a GLMM (ignored if predict is not `NULL`). (numeric of length 1)
`var`	Latent total variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
`link.inv`	Inverse function of the link function. (function)
`predict`	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`width`	Parameter for the integral computation. The integral is evaluated from `mu - width * sqrt(var)` to `mu + width * sqrt(var)`. The default value is 10, which should be sensible for most models. (numeric)

Details

This function needs the latent population mean (mu) or the marginal predicted values (predict) and the total latent variance (i.e. total latent variance var) to compute the observed phenotypic mean. To do so, it also requires the inverse function of the link function.

For example, if the link function is the natural logarithm, the inverse-link function will be the exponential. The inverse-link functions for many models are yielded by the QGlink.funcs function.

Contrary to QGparams, QGmean.obs never uses the closed form solutions, but always compute the integrals.

Value

This function yields the phenotypic mean on the observed scale. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGmvmean, QGparams, QGpred, QGlink.funcs, QGvar.dist, QGvar.exp, QGpsi

Examples

```
## Computing the observed mean for a probit link
QGmean(mu = 0.3, var = 1, link.inv = pnorm)
# The theoretical expectation is
1 - pnorm(0, 0.3, sqrt(1 + 1))

# Or, using the QGlink.funcs function
QGmean(mu = 0.3, var = 1, link.inv = QGlink.funcs(name = "binom1.probit")$inv.link)

## Computing the observed mean for a logarithm link
QGmean(mu = 1, var = 1, link.inv = exp)
# The theoretical expectation is
exp(1 + 0.5 * 1)

# This computation is automatically performed by QGparams
# but directly using the closed form solution when available
QGparams(mu = 1, var.p = 1, var.a = 0.5, model = "Poisson.log")
```

QGmvicc *Intra - Class Correlation coefficients (ICC) on the observed data scale (multivariate analysis).*

Description

Function to estimate the variance-covariance matrix of a variance component on the observed scale based on estimates on the latent scale. Contrary to the univariate function, this one cannot use the analytical closed forms and yields a list of paramaters instead of a data.frame.

Usage

```
QGmvicc(mu = NULL, vcv.comp, vcv.P, models, predict = NULL, rel.acc = 0.001,
width = 10, n.obs = NULL, theta = NULL, verbose = TRUE, mask = NULL)
```

Arguments

`mu`	Vector of latent intercepts estimated from a GLMM (ignored if predict is not `NULL`). (numeric)
`vcv.comp`	Component variance-covariance matrix (G-matrix - like). (numeric)
`vcv.P`	Total phenotypic variance-covariance matrix. Usually, the sum of all the estimated variance-covariance matrices. (numeric)
`models`	A vector containing the names of the model used or a list which elements contain the list of the functions needed (inverse-link, distribution variance and derivative of the inverse-link). See `QGlink.funcs` for a complete list of model available or the naming of the list of functions. (character vector or list of lists of functions)
`rel.acc`	Relative accuracy of the integral approximation. (numeric)
`width`	Parameter for the integral computation. The default value is 10, which should be sensible for most models. (numeric)
`predict`	Optional matrix of predicted values on the latent scale (each trait in each column). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`n.obs`	Number of "trials" for each "binomN" distribution. (numeric, length equal to the number of "binomN" models)
`theta`	Dispersion parameter for the Negative Binomial distribution. The parameter `theta` should be such as the variance of the distribution is `mean + mean^2 / theta`. (numeric, length equal to the number of "negbin" models)
`verbose`	Should the function be verbose? (boolean)
`mask`	Masking filter for removing predictions that don't exist in the population (e.g. female predictions for males for a sex-based bivariate model). Should the same dimensions as `predict` and values should be `FALSE` when the predictions should be filtered out.

Details

The function typically uses integral numerical approximation provided by the R2Cuba package to compute multivariate quantitative genetics parameters on the observed scale, from latent estimates yielded by a GLMM. It cannot use closed form solutions.

Only the most typical distribution/link function couples are implemented through the `models` argument. If you used an "exotic" GLMM, you can provide a list containg lists of functions corresponding to the model. The list of functions should be implemented as is the output of `QGlink.funcs`, i.e.

three elements: the inverse link functions named `inv.link`, the derivative of this function named `d.inv.link` and the distribution variance named `var.func` (see Example below).

Some distributions require extra-arguments. This is the case for "binomN", which require the number of trials N, passed with the argument `n.obs`. The distribution "negbin" requires a dispersion parameter `theta`, such as the variance of the distribution is `mean + mean^2 / theta` (mean/dispersion parametrisation). For now, the arguments `n.obs` and `theta` can be used for ONE distribution only.

If fixed effects (apart from the intercept) have been included in the GLMM, they can be included through the argument `predict` as a matrix of the marginal predicted values, i.e. predicted values excluding the random effects, for each trait (one trait per column of the matrix, see Example below).Note that computation can be extremely slow in that case.

Value

The function yields a list containing the following values:

`mean.obs`	Vector of phenotypic means on the observed scale.
`vcv.P.obs`	Phenotypic variance-covariance matrix on the observed scale.
`vcv.comp.obs`	Component variance-covariance (G-matrix - like, but broad - sense) on the observed scale.

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGmvparams`, `QGlink.funcs`, `QGmvmean`, `QGvcov`, `QGmvpsi`

Examples

```
## Example using a bivariate model (Binary trait/Gaussian trait)
# Parameters
mu <- c(0, 1)
G <- diag(c(0.5, 2))
M <- diag(c(0.2, 1))  # Maternal effect VCV matrix
P <- diag(c(1, 4))

# Broad - sense "G-matrix" on observed data scale
## Not run: QGmvicc(mu = mu, vcv.comp = G, vcv.P = P, models = c("binom1.probit", "Gaussian"))
# Maternal effect VCV matrix on observed data scale
## Not run: QGmvicc(mu = mu, vcv.comp = M, vcv.P = P, models = c("binom1.probit", "Gaussian"))
# Reminder: the results are the same here because we have no correlation between the two traits

# Defining the model "by hand" using the list
list.models = list(
  model1 = list(inv.link = function(x){pnorm(x)},
                d.inv.link = function(x){dnorm(x)},
                var.func = function(x){pnorm(x) * (1 - pnorm(x))}),
  model2 = list(inv.link = function(x){x},
                d.inv.link = function(x){1},
```

```
              var.func = function(x){0})
)
# Running the same analysis than above
## Not run: QGmvicc(mu = mu, vcv.comp = M, vcv.P = P, models = list.models)

# Using predicted values
# Say we have 100  individuals
n <- 100
# Let's simulate predicted values
p <- matrix(c(runif(n), runif(n)), ncol = 2)
# Note that p has as many as columns as we have traits (i.e. two)
# Multivariate analysis with predicted values
## Not run: QGmvicc(predict = p, vcv.comp = M, vcv.P = P, models = c("binom1.probit", "Gaussian"))
# That can be a bit long to run!
```

`QGmvmean` *Compute the multivariate phenotypic mean on the observed scale*

Description

This function calculates the multivariate phenotypic mean on the observed scale from multivariate latent mean and variance-covariance matrix.

Usage

```
QGmvmean(mu = NULL, vcov, link.inv, predict = NULL,
         rel.acc = 0.001, width = 10, mask = NULL)
```

Arguments

`mu`	Vector of latent intercepts estimated from a GLMM (ignored if predict is not `NULL`). (numeric)
`vcov`	Latent total phenotypic variance-covariance matrix estimated from a GLMM. Usually, the sum of all the estimated variance-covariance matrices. (numeric)
`link.inv`	Inverse functions of the link functions. This function should accept a vector and yield a vector of the same length, see Details and Example below. (function)
`predict`	Optional matrix of predicted values on the latent scale (each trait in each column). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`rel.acc`	Relative accuracy of the integral approximation. (numeric)
`width`	Parameter for the integral computation. The default value is 10, which should be sensible for most models. (numeric)
`mask`	Masking filter for removing predictions that don't exist in the population (e.g. female predictions for males for a sex-based bivariate model). Should the same dimensions as `predict` and values should be `FALSE` when the predictions should be filtered out.

Details

This function needs the multivariate latent population mean (mu) or the marginal predicted values (predict) and the total latent variance-covariance matrix (vcov) to compute the observed phenotypic mean.

To do so, it also requires the inverse functions of the link functions (link.inv). For an analysis with d traits, the function given to the link.inv argument should use a vector of length d and yield a vector of length d (see Example below).

Value

This function yields the mutlivariate phenotypic mean on the observed scale. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGmean, QGmvparams, QGlink.funcs, QGvcov, QGmvpsi

Examples

```
## Example using a bivariate model (Binary trait/Gaussian trait)
# Parameters
mu <- c(0, 1)
P <- diag(c(1, 4))

# Note: no phenotypic, nor genetic correlations, hence should be equal to univariate case!

# Setting up the link functions
# Note that since the use of "cubature" to compute the integrals,
# the functions must use a matrix as input and yield a matrix as output,
# each row corresponding to a trait
inv.links <- function(mat) {matrix(c(pnorm(mat[1, ]), mat[2, ]), nrow = 2, byrow = TRUE)}
# probit link and identity link respectively

# Computing the multivariate mean on observed scale
QGmvmean(mu = mu, vcov = P, link.inv = inv.links)
QGmean(mu = 0, var = 1, link.inv = pnorm)     # Same result than trait 1!
QGmean(mu = 1, var = 4, link.inv = function(x){x})     # Same result than trait 2!
# Reminder: the results are the same here because we have no correlation between the two traits
```

QGmvparams *Quantitative Genetics parameters from GLMM estimates (multivariate analysis).*

Description

Function to estimate the multivariate quantitative genetics parameters on the observed scale based on estimates on the latent scale. Contrary to the univariate function, this one cannot use the analytical closed forms and yields a list of paramaters instead of a data.frame.

Usage

```
QGmvparams(mu = NULL, vcv.G, vcv.P, models, predict = NULL, rel.acc = 0.001,
width = 10, n.obs = NULL, theta = NULL, verbose = TRUE, mask = NULL)
```

Arguments

`mu`	Vector of latent intercepts estimated from a GLMM (ignored if predict is not `NULL`). (numeric)
`vcv.G`	Genetic additive variance-covariance matrix (a.k.a. G-matrix). (numeric)
`vcv.P`	Total phenotypic variance-covariance matrix. Usually, the sum of all the estimated variance-covariance matrices. (numeric)
`models`	A vector containing the names of the model used or a list which elements contain the list of the functions needed (inverse-link, distribution variance and derivative of the inverse-link). See `QGlink.funcs` for a complete list of model available or the naming of the list of functions. (character vector or list of lists of functions)
`rel.acc`	Relative accuracy of the integral approximation. (numeric)
`width`	Parameter for the integral computation. The default value is 10, which should be sensible for most models. (numeric)
`predict`	Optional matrix of predicted values on the latent scale (each trait in each column). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`n.obs`	Number of "trials" for the "binomN" distribution. (numeric, length equal to the number of "negbin" models)
`theta`	Dispersion parameter for the Negative Binomial distribution. The parameter `theta` should be such as the variance of the distribution is `mean + mean^2 / theta`. (numeric, length equal to the number of "negbin" models)
`verbose`	Should the function be verbose? (boolean)
`mask`	Masking filter for removing predictions that don't exist in the population (e.g. female predictions for males for a sex-based bivariate model). Should the same dimensions as `predict` and values should be `FALSE` when the predictions should be filtered out.

Details

The function typically uses integral numerical approximation provided by the R2Cuba package to compute multivariate quantitative genetics parameters on the observed scale, from latent estimates yielded by a GLMM. It cannot use closed form solutions.

Only the most typical distribution/link function couples are implemented through the `models` argument. If you used an "exotic" GLMM, you can provide a list containg lists of functions corresponding to the model. The list of functions should be implemented as is the output of `QGlink.funcs`, i.e.

three elements: the inverse link functions named `inv.link`, the derivative of this function named `d.inv.link` and the distribution variance named `var.func` (see Example below).

Some distributions require extra-arguments. This is the case for "binomN", which require the number of trials N, passed with the argument `n.obs`. The distribution "negbin" requires a dispersion parameter `theta`, such as the variance of the distribution is `mean + mean^2 / theta` (mean/dispersion parametrisation). For now, the arguments `n.obs` and `theta` can be used for ONE distribution only.

If fixed effects (apart from the intercept) have been included in the GLMM, they can be included through the argument `predict` as a matrix of the marginal predicted values, i.e. predicted values excluding the random effects, for each trait (one trait per column of the matrix, see Example below).Note this can considerably slow down the algorithm, especially when no closed form is used.

Value

The function yields a list containing the following values:

`mean.obs`	Vector of phenotypic means on the observed scale.
`vcv.P.obs`	Phenotypic variance-covariance matrix on the observed scale.
`vcv.G.obs`	Additive genetic variance-covariance (a.k.a. G-matrix) on the observed scale.

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGparams`, `QGlink.funcs`, `QGmvmean`, `QGvcov`, `QGmvpsi`

Examples

```
## Example using a bivariate model (Binary trait/Gaussian trait)
# Parameters
mu <- c(0, 1)
G <- diag(c(0.5, 2))
P <- diag(c(1, 4))

# Note: no phenotypic, nor genetic correlations, hence should be equal to univariate case!

# Multivariate analysis
QGmvparams(mu = mu, vcv.G = G, vcv.P = P, models = c("binom1.probit", "Gaussian"))
QGparams(mu = 0, var.a = 0.5, var.p = 1, model = "binom1.probit") # Consistent results!
# Reminder: the results are the same here because we have no correlation between the two traits

# Defining the model "by hand" using the list
list.models = list(
  model1 = list(inv.link = function(x){pnorm(x)},
                d.inv.link = function(x){dnorm(x)},
                var.func = function(x){pnorm(x) * (1 - pnorm(x))}),
  model2 = list(inv.link = function(x){x},
                d.inv.link = function(x){1},
                var.func = function(x){0})
```

```
)
# Running the same analysis than above
QGmvparams(mu = mu, vcv.G = G, vcv.P = P, models = list.models) # Same results!

# Using predicted values
# Say we have 100  individuals
n <- 100
# Let's simulate predicted values
p <- matrix(c(runif(n), runif(n)), ncol = 2)
# Note that p has as many as columns as we have traits (i.e. two)
# Multivariate analysis with predicted values
## Not run: QGmvparams(predict = p, vcv.G = G, vcv.P = P, models = c("binom1.probit", "Gaussian"))
```

`QGmvpred` *Predict the evolutionary response to selection on the observed scale*

Description

This function uses an assumed or measured fitness function to compute evolutionary response to selection on the observed scale. To do so a latent fitness function must be provided to the function. This fitness function is used to compute the evolutionary response on the latent scale.

Usage

```
QGmvpred(mu = NULL, vcv.G, vcv.P, fit.func, d.fit.func,
         predict = NULL, rel.acc = 0.001, width = 10,
         verbose = TRUE, mask = NULL)
```

Arguments

`mu`	Vector of latent intercepts estimated from a GLMM (ignored if predict is not `NULL`). (numeric)
`vcv.G`	Genetic additive variance-covariance matrix (a.k.a. G-matrix). (numeric)
`vcv.P`	Total phenotypic variance-covariance matrix. Usually, the sum of all the estimated variance-covariance matrices. (numeric)
`fit.func`	Function giving the expected fitness on the observed scale for a given latent trait (see Example). (function)
`d.fit.func`	Derivative of the expected fitness to the latent trait. This function should return a vector containing the partial derivative to each trait (see Example). (function)
`rel.acc`	Relative accuracy of the integral approximation. (numeric)
`width`	Parameter for the integral computation. The default value is 10, which should be sensible for most models. (numeric)
`predict`	Optional matrix of predicted values on the latent scale (each trait in each column). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`verbose`	Should the function be verbose? (boolean)

`mask` Masking filter for removing predictions that don't exist in the population (e.g. female predictions for males for a sex-based bivariate model). Should the same dimensions as `predict` and values should be `FALSE` when the predictions should be filtered out.

Details

The function uses the latent fitness function (`fitness.func`) and latent quantitative genetics parameters to compute the expected selection differential and response on the latent scale.

There is no argument to describe the model used as it is already and implicitely contained in the calculation of `fit.func` and `d.fit.func` (see Example below).

If fixed effects were included during the estimation of the quantitative genetics parameters, they can be included as marginal predicted values, i.e. predicted values excluding the random effects, which can be calculated as the matrix product **Xb** where **X** is the design matrix and **b** is the vector of fixed effects estimates. To do so, provide the vector of marginal predicted values using the argument `predict`. Note this will considerably slow down the algorithm.

The predictions can be transposed on the observed scale by using the `QGmvmean` function (see Example below).

Value

The function yields a data.frame containing:

- `mean.lat.fitness`Average latent fitness. (numeric)
- `lat.grad`Latent selection gradient. (numeric)
- `lat.sel`Latent selection differential. (numeric)
- `lat.resp`Latent evolutionary response to selection. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGparams`, `QGlink.funcs`, `QGmean`, `QGvar.dist`, `QGvar.exp`, `QGpsi`

Examples

```
## Bivariate example with a binary trait and a Gaussian one

# Assume a bivariate GLMM with Binomial(probit)/Gaussian distributions with:
mu <- c(0, 10)
G <- matrix(c(0.5, 0, 0, 1), nrow = 2)
P <- matrix(c(1, 0, 0, 2), nrow = 2)

# Link functions
inv.links = function(vec){c(pnorm(vec[1]), vec[2])}

# Creating the expected fitness function
```

```
# i.e. expected fitness given a latent trait vector l
# Say if the binary trait is 1, then the fitness is 0.5 * "the Gaussian trait"
# But if the binary trait is 0, then the fitness is 0
lat.fit <- function(mat) {pnorm(mat[1, ]) * 0.5 * mat[2, ]}
# Derivative of the above function
# This function yields a vector which elements are the derivative according to each trait
d.lat.fit <- function(mat) {matrix(c(dnorm(mat[1, ]) * 0.5 * mat[2, ], pnorm(mat[1, ]) * 0.5),
                           nrow = 2,
                           byrow = TRUE)}

# Predicting the latent evolutionary response
pred<- QGmvpred(mu = mu, vcv.P = P, vcv.G = G, fit.func = lat.fit, d.fit.func = d.lat.fit)


# Predicting the observed evolutionary response
# Current observed phenotypic mean
QGmvmean(mu = mu, vcov = P, link.inv = inv.links)
# Predicted observed phenotypic mean after selection
QGmvmean(mu = mu + pred$lat.resp, vcov = P, link.inv = inv.links)
```

`QGmvpsi` *Compute a multivariate "Psi" (used to compute the additive genetic variance on the observed scale).*

Description

This function computes a multivariate version of the parameter "Psi" which relates the additive genetic variance-covariance matrix on the latent scale to the additive genetic variance-covariance matrix on the observed scale: G.obs = Psi %*% G %*% t(Psi)

Usage

```
QGmvpsi(mu = NULL, vcov, d.link.inv, predict = NULL,
        rel.acc = 0.001, width = 10, mask = NULL)
```

Arguments

`mu`	Vector of latent intercepts estimated from a GLMM (ignored if predict is not `NULL`). (numeric)
`vcov`	Latent total phenotypic variance-covariance matrix estimated from a GLMM. Usually, the sum of all the estimated variance-covariance matrices. (numeric)
`d.link.inv`	Derivative of the inverse-link functions. This function should accept a vector and yield a vector of the same length, see Details and Example below. (function)
`predict`	Optional matrix of predicted values on the latent scale (each trait in each column). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`rel.acc`	Relative accuracy of the integral approximation. (numeric)

width	Parameter for the integral computation. The default value is 10, which should be sensible for most models. (numeric)
mask	Masking filter for removing predictions that don't exist in the population (e.g. female predictions for males for a sex-based bivariate model). Should the same dimensions as predict and values should be FALSE when the predictions should be filtered out.

Details

The multivariate parameter "Psi" is a diagonal matrix which elements are the average of the derivative of the inverse-link function. The additive genetic variance-covariance matrix on the latent scale G is linked to the additive genetic variance-covariance matrix on the observed scale G.obs through Psi: G.obs = Psi %*% G %*% t(Psi).

This function requires the derivatives of the inverse-link functions (d.link.inv). For an analysis with d traits, the function given to the d.link.inv argument should use a vector of length d and yield a vector of length d (see Example below).

Value

This function yields the matrix "Psi". (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGpsi, QGmvparams, QGlink.funcs, QGvcov, QGmvpsi

Examples

```
## Example using a bivariate model (Binary trait/Gaussian trait)
# Parameters
mu <- c(0, 1)
G <- diag(c(0.5, 2))
P <- diag(c(1, 4))

# Setting up the derivatives of the inverse-link functions
dinvs <- function(mat) {matrix(c(dnorm(mat[1, ]), rep(1, length(mat[2, ]))),
                               nrow = 2,
                               byrow = TRUE)}
# The derivative of pnorm() is dnorm(), and the derivative of the identity is 1

# Computing Psi
Psi <- QGmvpsi(mu = mu, vcov = P, d.link.inv = dinvs)
# Computing genetic additive variance-covariance matrix on the observed scale
Psi
G.obs <- Psi %*% G %*% t(Psi)

QGparams(mu = 0, var.a = 0.5, var.p = 1, model = "binom1.probit")
# Same additive variance than trait 1
```

```
# Reminder: the results are the same here because we have no correlation between the two traits
```

QGparams *Quantitative Genetics parameters from GLMM estimates.*

Description

Function to estimate the quantitative genetics parameters on the observed scale based on estimates on the latent scale. The function yields a data.frame which includes the phenotypic mean and variance, as well as the additive genetic variance and heritability, on the observed scale.

Usage

```
QGparams(mu, var.a, var.p, model = "", width = 10, predict = NULL,
         closed.form = TRUE, custom.model = NULL, n.obs = NULL,
         cut.points = NULL, theta = NULL, verbose = TRUE)
```

Arguments

mu	Latent intercept estimated from a GLMM (ignored if predict is not NULL). (numeric of length 1)
var.a	Latent additive genetic variance estimated from a GLMM. (numeric of length 1)
var.p	Latent total phenotypic variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
model	Name of the used model, i.e. distribution.link. Ignored if custom.model is not NULL. See QGlink.funcs for a complete list of model available. (character)
width	Parameter for the integral computation. The integral is evaluated from mu - width * sqrt(var.p) to mu + width * sqrt(var.p). The default value is 10, which should be sensible for most models. (numeric)
predict	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
closed.form	When available, should closed forms be used instead of integral computations? (boolean, ignored if model = "ordinal")
custom.model	If the model used is not available using the model argument, a list of functions describing the model can be provided. (list of functions, see Details)
n.obs	Number of "trials" for the "binomN" distribution. (numeric)
cut.points	Values for the "cut points" in the multiple threshold model ("ordinal"). Should be a vector of length equal to the number of categories plus one, starting with the value -Inf and ending with the value Inf. (numeric)
theta	Dispersion parameter for the Negative Binomial distribution. The parameter theta should be such as the variance of the distribution is mean + mean^2 / theta. (numeric)
verbose	Should the function be verbose? (boolean)

Details

The function typically uses precise integral numerical approximation to compute quantitative genetics parameters on the observed scale, from latent estimates yielded by a GLMM. If closed form solutions for the integrals are available, it uses them if `closed.form = TRUE`.

Only the most typical distribution/link function couples are implemented in the function. If you used an "exotic" GLMM, you can use the `custom.model` argument. It should take the form of a list of functions. The first function should be the inverse of the link function named `inv.link`, the second function should be the "distribution variance" function named `var.func` and the third function should be the derivative of the inverse link function named `d.inv.link` (see Example below).

Some distributions require extra-arguments. This is the case for "binomN", which require the number of trials N, passed with the argument `n.obs`. The distribution "negbin" requires a dispersion parameter `theta`, such as the variance of the distribution is `mean + mean^2 / theta` (mean/dispersion parametrisation).

If fixed effects (apart from the intercept) have been included in the GLMM, they can be included as marginal predicted values, i.e. predicted values excluding the random effects, which can be calculated as the matrix product **Xb** where **X** is the design matrix and **b** is the vector of fixed effects estimates. To do so, provide the vector of marginal predicted values using the argument `predict`. Note this can considerably slow down the algorithm, especially when no closed form is used.

Ordinal model is different from the other models, because it yields multivariate inference on the observed data scale, even though the latent scale is not multivariate. As a consequence, this model can only be accessed using the function `QGparams` and has an output similar to the one of `QGmvparams`.

Value

The function yields a data.frame containing the following values:

`mean.obs`	Phenotypic mean on the observed scale.
`var.obs`	Phenotypic variance on the observed scale.
`var.a.obs`	Additive genetic variance on the observed scale.
`h2.obs`	Heritability on the observed scale.

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGmvparams`, `QGpred`, `QGlink.funcs`, `QGmean`, `QGvar.dist`, `QGvar.exp`, `QGpsi`

Examples

```
## Example using binary data
# Parameters
mu <- 0
va <- 1
vp <- 2
```

```
# Simulating data l = mu + a + e
lat<- mu + rnorm(1000, 0, sqrt(va)) + rnorm(1000, 0, sqrt(vp - va))
y<- rbinom(1000, 1, pnorm(lat))

# Expected results
QGparams(mu = 0, var.p = 2, var.a = 1, model = "binom1.probit")
# Simulated results for mean and variance
mean(y)
var(y)

# Using integral approximations
QGparams(mu = 0, var.p = 2, var.a = 1, model = "binom1.probit", closed.form = FALSE)
# Note that the approximation is exactly equal to the results obtained with the closed form

# Let's create a custom model
custom <- list(inv.link = function(x){pnorm(x)},
    var.func = function(x){pnorm(x) * (1 - pnorm(x))},
    d.inv.link = function(x){dnorm(x)})

QGparams(mu = 0, var.p = 2, var.a = 1, custom.model = custom)

# Using an ordinal model (with 4 categories)
QGparams(mu = 0.1, var.a = 1, var.p = 2, cut.points = c( - Inf, 0, 0.5, 1, Inf), model = "ordinal")
# Note the slightly different output (see QGmvparams)

# Integrating over a posterior distribution
# e.g. output from MCMCglmm named "model"
# df <- data.frame(mu = model$Sol[, 'intercept'],
#                  va = model$VCV[, 'animal'],
#                  vp = rowSums(model$VCV))
# params <- apply(df, 1, function(row){
#        QGparams(mu = row$mu, var.a = row$va, var.p = row$vp, model = "Poisson.log")
# })
```

QGpred *Predict the evolutionary response to selection on the observed scale*

Description

This function uses an assumed or measured fitness function to compute evolutionary response to selection on the observed scale. To do so a latent fitness function must be provided to the function. This fitness function is used to compute the evolutionary response on the latent scale.

Usage

```
QGpred(mu = NULL, var.a, var.p, fit.func, d.fit.func, width = 10,
       predict = NULL, verbose = TRUE)
```

Arguments

`mu`	Latent intercept estimated from a GLMM (set to 0 if `predict` is not `NULL`). (numeric of length 1)
`var.a`	Latent additive genetic variance estimated from a GLMM. (numeric of length 1)
`var.p`	Latent total phenotypic variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
`fit.func`	Function giving the expected fitness on the observed scale for a given latent trait (see Example). (function)
`d.fit.func`	Derivative of the expected fitness to the latent trait (see Example). (function)
`width`	Parameter for the integral computation. The integral is evaluated from `mu - width * sqrt(var.p)` to `mu + width * sqrt(var.p)`. The default value is 10, which should be sensible for most models. (numeric)
`predict`	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`verbose`	Should the function be verbose? (boolean)

Details

The function uses the latent fitness function (`fit.func`) and latent quantitative genetics parameters to compute the expected selection differential and response on the latent scale.

There is no argument to describe the model used as it is already and implicitely contained in the calculation of `fit.func`.

If fixed effects were included during the estimation of the quantitative genetics parameters, they can be included as marginal predicted values, i.e. predicted values excluding the random effects, which can be calculated as the matrix product **Xb** where **X** is the design matrix and **b** is the vector of fixed effects estimates. To do so, provide the vector of marginal predicted values using the argument `predict`. Note this will considerably slow down the algorithm.

The predictions can be transposed on the observed scale by using the `QGmean` function (see Example below).

Value

The function yields a data.frame containing:

- `mean.lat.fitness`Average latent fitness. (numeric)
- `lat.grad`Latent selection gradient. (numeric)
- `lat.sel`Latent selection differential. (numeric)
- `lat.resp`Latent evolutionary response to selection. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGparams, QGlink.funcs, QGmean, QGvar.dist, QGvar.exp, QGpsi

Examples

```
## Example with binary traits and a fitness measurement
# Let's assume we dispose of a binary trait measurement
# and associated fitness of trait 0 (say 1) and trait 1 (say 1.86)
# We further assume a GLMM with Binomial distribution and probit link with:
mu <- -0.1
va <- 2
vp <- 2.5  # note that the latent heritability is very high

# Creating the latent fitness function
# i.e. expected fitness given a latent trait l
# We have a trait 1 with probability pnorm(l) with fitness 1.86
# We have a trait 0 with probability (1 - pnorm(l)) with fitness 1
lat.fit<- function(l){(1 - pnorm(l)) * 1 + pnorm(l) * 1.86}
# Derivate of the fitnes function
d.lat.fit<- function(l){- dnorm(l) * 1 + dnorm(l) * 1.86}

# Predicting the latent evolutionary response
pred <- QGpred(mu = mu, var.p = vp, var.a = va, fit.func = lat.fit, d.fit.func = d.lat.fit)

# Predicting the observed evolutionary response
# Current observed phenotypic mean
QGmean(mu = mu, var = vp, link.inv = QGlink.funcs("binom1.probit")$inv.link)
# Predicted observed phenotypic mean after selection
QGmean(mu = mu + pred$lat.resp, var = vp, link.inv = QGlink.funcs("binom1.probit")$inv.link)
```

QGpsi *Compute "Psi" (used to compute the additive genetic variance on the observed scale).*

Description

This function computes the parameter "Psi" which relates the additive genetic variance on the latent scale to the additive genetic variance on the observed scale: Va.obs = (Psi^2) * Va

Usage

```
QGpsi(mu = NULL, var, d.link.inv, predict = NULL, width = 10)
```

Arguments

mu	Latent intercept estimated from a GLMM (set to 0 if predict is not NULL). (numeric of length 1)

`var`	Latent total phenotypic variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
`d.link.inv`	Derivative of the inverse-link function. (function)
`predict`	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`width`	Parameter for the integral computation. The integral is evaluated from `mu - width * sqrt(var)` to `mu + width * sqrt(var)`. The default value is 10, which should be sensible for most models. (numeric)

Details

The parameter "Psi" is the average of the derivative of the inverse-link function. The additive genetic variance on the observed scale is linked to the additive genetic variance on the latent scale by : Va.obs = (Psi^2) * Va.lat.

Value

This function yields the "Psi" parameter. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

`QGmvpsi`, `QGparams`, `QGpred`, `QGlink.funcs`, `QGmean`, `QGvar.dist`, `QGvar.exp`

Examples

```
## Example using binom1.probit model
mu <- 0
va <- 1
vp <- 2
# The inverse-link for a probit is the CDF of a standard Gaussian
# Hence its derivative is the PDF of a standard Gaussian
dinv <- function(x){dnorm(x)}

# Computing Psi
Psi <- QGpsi(mu = 0, var = 2, d.link.inv = dinv)
# Computing additive variance on the observed scale
(Psi^2) * va

# This function is used by QGparams to obtain var.a.obs
QGparams(mu = 0, var.p = vp, var.a = va, model = "binom1.probit")
# Same results as above!
```

QGvar.dist *Compute the distribution variance*

Description

This function computes the variance emerging from the error distribution around the individual expected value. This variance, added to the variance of the individual expected values themselves (see QGvar.exp) yields the total observed phenotypic variance.

Usage

```
QGvar.dist(mu = NULL, var, var.func, predict = NULL, width = 10)
```

Arguments

mu	Latent intercept estimated from a GLMM (ignored if predict is not NULL). (numeric of length 1)
var	Latent total phenotypic variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
var.func	Function giving the variance of the distribution according to a given latent value. (function)
predict	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
width	Parameter for the integral computation. The integral is evaluated from mu - width * sqrt(var) to mu + width * sqrt(var). The default value is 10, which should be sensible for most models. (numeric)

Details

The distribution variance is the part of the observed variance emerging from the error distribution. It is calculated as an average error variance over all possible latent values. The distribution variance added to the variance of the expected values gives the total phenotypic variance on the observed scale.

The variance function (var.func) is a function giving the variance of the error distribution of the GLMM according to a given latent value.

Using a Poisson distribution with a logarithm link, this function is exp(x), because the variance of a Poisson is its mean. Using a Negative Binomial distribution with a logarithm link, this function will now be exp(x) + exp(2 * x) / theta. Note that the dispersion parameter theta is necessary for a Negative Binomial distribution.

The var.func function is yielded by QGlink.funcs according to a given distribution.link model (see Example below).

Contrary to QGparams, QGvar.exp never uses the closed form solutions, but always compute the integrals.

Value

This function yields the distribution variance. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGvar.exp, QGparams, QGpred, QGlink.funcs, QGmean, QGpsi

Examples

```
## Example using Poisson.log model
mu <- 1
va <- 0.2
vp <- 0.5

# The variance function is simply the inverse-link function
# because the variance of a Poisson is its mean
varfunc <- function(x) { exp(x) }

QGvar.dist(mu = mu, var = vp, var.func = varfunc)

# The QGlink.funcs gives a ready - to - use var.func
funcs <- QGlink.funcs(name = "Poisson.log")

# Calculating the distribution variance
vdist <- QGvar.dist(mu = mu, var = vp, var.func = funcs$var.func)

vdist           # Same value as above

# Calculating the variance of the expected values
vexp <- QGvar.exp(mu = mu, var = vp, link.inv = funcs$inv.link)

# The phenotypic variance on the observed scale is then:
vexp + vdist

# This computation is automatically performed by QGparams
# but directly using the closed form solutions when available
QGparams(mu = mu, var.p = vp, var.a = va, model = "Poisson.log")
# var.obs is equal to the sum above
```

QGvar.exp *Compute the variance of expected values (i.e. the latent values after inverse-link transformation.)*

Description

This function computes the variance of the expected values, i.e. the variance of the latent values after transformation through the inverse-link function. This variance, added to the distribution variance, yields to the phenotypic variance on the observed scale.

Usage

```
QGvar.exp(mu = NULL, var, link.inv, obs.mean = NULL, predict = NULL, width = 10)
```

Arguments

`mu`	Latent intercept estimated from a GLMM (ignored if predict is not `NULL`). (numeric of length 1)
`var`	Latent total phenotypic variance estimated from a GLMM. Usually, the sum of the estimated variances of the random effects, plus the "residual" variance. (numeric of length 1)
`link.inv`	Inverse function of the link function. (function)
`obs.mean`	Optional parameter giving the phenotypic mean on the observed scale. Automatically computed if not provided. (numeric)
`predict`	Optional vector of predicted values on the latent scale (i.e. matrix product **Xb**). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
`width`	Parameter for the integral computation. The integral is evaluated from `mu - width * sqrt(var)` to `mu + width * sqrt(var)`. The default value is 10, which should be sensible for most models. (numeric)

Details

The variance of the expected values is the variance that directly arise from the variance of the latent values, but after transformation through the inverse-link function. For example, using a logarithm link, this is the variance of exp(l) where l is the latent trait.

To compute the variance, the function needs the phenotypic mean on the observed scale. If this was already computed, it can be provided using the optional argument `obs.mean`, which will save computing time. Otherwise (default), the function will compute the mean on the observed scale before computing the variance.

This variance, when added to the distribution variance (see `QGvar.dist`) yields the phenotypic variance on the observed scale.

The function required for `link.inv` is yielded by `QGlink.funcs` according to a given distribution.link model (see Example below).

Contrary to `QGparams`, `QGvar.dist` never uses the closed form solutions, but always compute the integrals.

Value

This function yields the variance of the expected values. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGvar.dist, QGparams, QGpred, QGlink.funcs, QGmean, QGpsi

Examples

```
## Example using Poisson.log model
mu <- 1
va <- 0.2
vp <- 0.5

# The inverse-link for a logarithm link is the exponential
inv.link<- function(x){exp(x)}

# We can then calculate the variance of expected values
QGvar.exp(mu = mu, var = vp, link.inv = inv.link)

# The mean on the observed scale can be computed beforehand
y_bar <- QGmean(mu = mu, var = vp, link.inv = inv.link)
QGvar.exp(mu = mu, var = vp, obs.mean = y_bar, link.inv = inv.link)

# The QGlink.funcs gives a ready - to - use inverse-link function
funcs<- QGlink.funcs(name = "Poisson.log")

# Calculating the distribution variance
vexp <- QGvar.exp(mu = mu, var = vp, obs.mean = y_bar, link.inv = funcs$var.func)

vexp           # Same value as above

# Calculating the associated distribution variance
vdist <- QGvar.dist(mu = mu, var = vp, var.func = funcs$var.func)

# The phenotypic variance on the observed scale is then:
vexp + vdist

# This computation is automatically performed by QGparams
# but directly using the closed form solutions when available
QGparams(mu = mu, var.p = vp, var.a = va, model = "Poisson.log")
# var.obs is equal to the sum above
```

QGvcov *Compute the phenotypic variance-covariance matrix on the observed / expected scale*

Description

This function computes the total phenotypic variance-covariance matrix on the observed or expected scales.

Usage

```
QGvcov(mu = NULL, vcov, link.inv, var.func, mvmean.obs = NULL,
      predict = NULL, rel.acc = 0.001, width = 10, exp.scale = FALSE, mask = NULL)
```

Arguments

mu	Vector of latent intercepts estimated from a GLMM (ignored if predict is not NULL). (numeric)
vcov	Latent total phenotypic variance-covariance matrix estimated from a GLMM. Usually, the sum of all the estimated variance-covariance matrices. (numeric)
link.inv	Inverse functions of the link functions. This function should accept a vector and yield a vector of the same length, see Details and Example below. (function)
var.func	Function giving the variance function for each trait. This function should accept a vector and yield a vector of the same length, see Details and Example below. (function)
mvmean.obs	Optional parameter giving the multivariate phenotypic mean on the observed scale. Automatically computed if not provided. (numeric)
predict	Optional matrix of predicted values on the latent scale (each trait in each column). The latent predicted values must be computed while only accounting for the fixed effects (marginal to the random effects). (numeric)
rel.acc	Relative accuracy of the integral approximation. (numeric)
width	Parameter for the integral computation. The default value is 10, which should be sensible for most models. (numeric)
exp.scale	Should the variance-covariance matrix be computed on the expected scale? FALSE by default, which means the variance-covariance matrix is computed on the observed scale. (boolan)
mask	Masking filter for removing predictions that don't exist in the population (e.g. female predictions for males for a sex-based bivariate model). Should the same dimensions as predict and values should be FALSE when the predictions should be filtered out.

Details

This function needs the multivariate latent population mean (mu) or the marginal predicted values (predict) and the total latent variance-covariance matrix (vcov) to compute the phenotypic variance-covariance matrix on the observed scale (or on the expected scale if exp.scale is TRUE).

To do so, it also requires the inverse functions of the link functions (link.inv) and the distribution variance functions (var.func). For an analysis with d traits, the function given to these arguments should use a vector of length d and yield a vector of length d (see Example below).

Value

This function yields the phenotypic variance-covariance on the observed or expected scale. (numeric)

Author(s)

Pierre de Villemereuil & Michael B. Morrissey

See Also

QGvar.exp, QGvar.dist, QGmvparams, QGlink.funcs, QGmvpsi

Examples

```
## Example using a bivariate model (Binary trait/Gaussian trait)
# Parameters
mu <- c(0, 1)
P <- diag(c(1, 4))

# Note: no phenotypic, nor genetic correlations, hence should be equal to univariate case!

# Setting up the link functions
# Note that since the use of "cubature" to compute the integrals,
# the functions must use a matrix as input and yield a matrix as output,
# each row corresponding to a trait
inv.links <- function(mat) {matrix(c(pnorm(mat[1, ]), mat[2, ]), nrow = 2, byrow = TRUE)}

# Setting up the distribution variance functions
var.funcs <- function(mat) {matrix(c(pnorm(mat[1, ]) * (1 - pnorm(mat[1, ])), 0 * mat[2, ]),
                                   nrow = 2,
                                   byrow = TRUE)}
# The first row is p * (1 - p) (variance of a binomial)
# The second row is 0 because no extra distribution is assumed for a Gaussian trait

# Computing the multivariate mean on observed scale
# Phenotypic VCV matrix on observed scale
QGvcov(mu = mu, vcov = P, link.inv = inv.links, var.func = var.funcs)
# Phenotypic VCV matrix on the expected scale
QGvcov(mu = mu, vcov = P, link.inv = inv.links, var.func = var.funcs, exp.scale = TRUE)

QGvar.exp(mu = 0, var = 1, link.inv = pnorm) # Same variance on the expected scale
QGvar.exp(mu = 0, var = 1, link.inv = pnorm) +
    QGvar.dist(mu = 0, var = 1, var.func = function(x){pnorm(x) * (1 - pnorm(x))})
# Same variance on the observed scale
# Reminder: the results are the same here because we have no correlation between the two traits
```

Index