Package 'rr2'

August 21, 2023

Type Package

Title R2s for Regression Models

Version 1.1.1

Description Three methods to calculate R2 for models with correlated errors, including Phylogenetic GLS, Phylogenetic Logistic Regression, Linear Mixed Models (LMMs), and Generalized Linear Mixed Models (GLMMs). See details in Ives 2018 <doi:10.1093/sysbio/syy060>.

License GPL-3

Depends R (>= 3.0), stats

Encoding UTF-8

Imports lme4, phylolm (>= 2.6.2), ape, utils, Matrix, nlme, phyr (>= 1.0.3)

RoxygenNote 7.2.3

Suggests testthat, mvtnorm

URL https://github.com/arives/rr2

BugReports https://github.com/arives/rr2/issues

NeedsCompilation no

Author Anthony Ives [aut, cre], Daijiang Li [ctb]

Maintainer Anthony Ives <arives@wisc.edu>

Repository CRAN

Date/Publication 2023-08-21 14:20:04 UTC

R topics documented:

`binaryPGLMM` *Phylogenetic GLM for binary data*

Description

Fitting phylogenetic generalized linear models for binary data (0 and 1).

Usage

```
binaryPGLMM(
  formula,
  data = list(),
  phy,
  s2.init = 0.1,
  B.init = NULL,
  tol.pql = 10^-6,
  maxit.pql = 200,
  maxit.reml = 100
)
```

Arguments

`formula`	Regression formula.
`data`	Data frame to fit the model with.
`phy`	Phylogenetic tree of type phylo with branch lengths.
`s2.init`	Initial variance values for random terms, default is 0.1.
`B.init`	Initial coefficient values for fixed terms, if not provided, will use those from `lm`.
`tol.pql`	Tolerance value, default is 10^-6.
`maxit.pql`	The number of iterations, default is 200.
`maxit.reml`	The number of iterations for optim, default is 100.

Value

A large list with class as `binaryPGLMM`.

inv.logit *Invert logit function*

Description

Convert numeric values between 0 and 1.

Usage

```
inv.logit(x)
```

Arguments

x	A numeric vector.

partialR2 *Partial R2*

Description

Get partial R2 by comparing a model and its reduced model.

Usage

```
partialR2(mod, mod.r)
```

Arguments

mod	A linear regression model.
mod.r	A reduced model based on mod.

Value

R2 value between 0 and 1.

partialR2adj *Adjusted partial R2*

Description

Get adjusted partial R2 by comparing a model and its reduced model.

Usage

```
partialR2adj(
  mod,
  df.f = summary(mod)$df[1],
  mod.r,
  df.r = summary(mod.r)$df[1]
)
```

Arguments

mod	A linear regression model.
df.f	Degree of freedom of the mod.
mod.r	A reduced model based on mod.
df.r	Degree of freedom of the reduced mod.r.

Value

A list of both R2 and adjusted R2 , the latter is not necessary to be between 0 and 1.

R2 *Calculate R2_lik, R2_resid, and R2_pred*

Description

This is a wrapper for calculating three R2s – R2_lik, R2_resid, and R2_pred – for LMMs and GLMMs, and phylogenetic LMMs (PLMMs) and GLMMs (PGLMMs). Note that the individual functions R2_lik(), R2_resid(), and R2_pred() can be called separately. This is preferrable if you are only interested in one R2; for example, for phylolm() called from 'R2' you need to specify 'phy' (phylo object for the phylogeny), while R2_lik() does not require this.

Usage

```
R2(
  mod = NULL,
  mod.r = NULL,
  phy = NULL,
  sigma2_d = c("s2w", "NS", "rNS"),
  lik = TRUE,
  resid = TRUE,
  pred = TRUE,
  ...
)
```

Arguments

mod	A regression model with one of the following classes: 'lm', 'glm', lmerMod', glmerMod', 'phylolm', 'gls', 'pglmm', 'pglmm_compare', binaryPGLMM', or 'communityPGLMM'.
mod.r	A reduced model; if not provided, the total R2 will be given by setting 'mod.r' to the model corresponding to 'mod' with the intercept as the only predictor.
phy	The phylogeny for phylogenetic models (as a 'phylo' object), which is not required to be specified for `R2_lik()` or non-phylogenetic models.
sigma2_d	Distribution-specific variance σ_d^2 (see Details) used in `R2_resid()`. For binomial GLMs, GLMMs and PGLMMs with logit link functions, options are c('s2w', 'NS', 'rNS'). For binomial GLMs, GLMMs and PGLMMs with probit link functions, options are c('s2w', 'NS'). Other families use 's2w'.
lik	Whether to calculate R2_lik; default is TRUE.
resid	Whether to calculate R2_resid; default is TRUE.
pred	Whether to calculate R2_pred; default is TRUE.
...	Additional arguments for 'R2_pred()'. 'gaussian.pred = "tip_rm"' or 'gaussian.pred = "nearest_node"'.

Details

Details about the methods are provided under the separate functions for `R2_lik()`, `R2_resid()`, and `R2_pred()`. There are also many worked examples.

Value

A vector, with all three R2s by default.

Author(s)

Daijiang Li and Anthony R. Ives

References

Ives A.R. and Li D. 2018. rr2: An R package to calculate R2s for regression models. Journal of Open Source Software. DOI:10.21105/joss.01028

Ives A.R. 2018. R2s for Correlated Data: Phylogenetic Models, LMMs, and GLMMs. Systematic Biology, Volume 68, Issue 2, March 2019, Pages 234-251. DOI:10.1093/sysbio/syy060

See Also

MuMIn, lme4, ape, phylolm, phyr, pez

Examples

```
library(ape)
library(phylolm)
library(lme4)
library(nlme)
library(phyr)

set.seed(12345)
p1 <- 10
nsample <- 10
n <- p1 * nsample

d <- data.frame(x1 = 0, x2 = 0, u1 = rep(1:p1, each = nsample),
                u2 = rep(1:p1, times = nsample))
d$u1 <- as.factor(d$u1)
d$u2 <- as.factor(d$u2)

b1 <- 1
b2 <- -1
sd1 <- 1.5

d$x1 <- rnorm(n = n)
d$x2 <- rnorm(n = n)
d$y.lmm <- b1 * d$x1 + b2 * d$x2 +
  rep(rnorm(n = p1, sd = sd1), each = nsample) +
  rep(rnorm(n = p1, sd = sd1), times = nsample) +
  rnorm(n = n)

prob <- inv.logit(b1 * d$x1 + rep(rnorm(n = p1, sd = sd1), each = nsample))
d$y.glmm <- rbinom(n = n, size = 1, prob = prob)

# LMM with two fixed and two random effects ----
z.f <- lmer(y.lmm ~ x1 + x2 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.x <- lmer(y.lmm ~ x1 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.v <- lmer(y.lmm ~ 1 + (1 | u2), data = d, REML = FALSE)
z.0 <- lm(y.lmm ~ 1, data = d)

R2(z.f, z.x)
R2(z.f, z.v)
R2(z.f)
```

```
# GLMM with one fixed and one random effect ----
z.f <- glmer(y.glmm ~ x1 + (1 | u1), data = d, family = 'binomial')
z.x <- glmer(y.glmm ~ 1 + (1 | u1), data = d, family = 'binomial')
z.v <- glm(y.glmm ~ x1, data = d, family = 'binomial')

R2(z.f, z.x)
R2(z.f, z.v)
R2(z.f)

# These give different results for R2_resid.
R2(z.f, sigma2_d = 's2w')
R2(z.f, sigma2_d = 'NS')
R2(z.f, sigma2_d = 'rNS')

# GLS {nlme} with one fixed effect and autocorrelated errors among 6 groups ----
nT <- 10
nseries <- 6
n <- nT * nseries

d <- data.frame(x = 0, y = 0, u = rep(1:nseries, each = nT), e = rnorm(1))
d$u <- as.factor(d$u)
d$x <- rnorm(n = n)
ar1 <- .5
for(t in 2:n) d$e[t] <- ar1*d$e[t-1] + rnorm(1)

b1 <- 1
d$y <- b1 * d$x + d$e

z.f <- gls(y ~ x + u, correlation = corAR1(form = ~1 | u), data = d)
z.x <- gls(y ~ 1, correlation = corAR1(form = ~1 | u), data = d)
z.ar <- lm(y ~ x + u, data = d)

R2(z.f, z.x)
R2(z.f, z.ar)
R2(z.f)

# PGLS with a single fixed effect ----
n <- 100
d <- data.frame(x = rep(0, n), y = 0)

b1 <- 1.5
signal <- 0.7

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rTraitCont(phy.x, model = 'BM', sigma = 1)
e <- signal ^ 0.5 * rTraitCont(phy, model = 'BM', sigma = 1) +
  (1 - signal) ^ 0.5 * rnorm(n = n)
d$x <- x[match(names(e), names(x))]
d$y <- b1 * x + e
```

```
rownames(d) <- phy$tip.label
d$sp <- phy$tip.label

# Fit with phylolm() in {phylolm}
z.f <- phylolm(y ~ x, phy = phy, data = d, model = 'lambda')
z.x <- phylolm(y ~ 1, phy = phy, data = d, model = 'lambda')
z.v <- lm(y ~ x, data = d)

R2(z.f, z.x, phy = phy)
R2(z.f, z.v, phy = phy)
R2(z.f, phy = phy)

# These data can also be fit with pglmm_compare in {phyr}
# Note that pglmm_compare will be renamed to pglmm_compare in the next version
z.f <- pglmm_compare(y ~ x, data = d, phy = phy, REML=FALSE)
z.x <- pglmm_compare(y ~ 1, data = d, phy = phy, REML=FALSE)
z.v <- glm(y ~ x, data = d)

R2(z.f, z.x)
R2(z.f, z.v)
R2(z.f)

# This also works for models fit with gls() in {nlme}
z.f <- gls(y ~ x, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.x <- gls(y ~ 1, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.v <- lm(y ~ x, data = d)

R2(z.f, z.x)
R2(z.f, z.v)
R2(z.f)

# But note that you need to define weights for gls() with non-ultrametric trees;
# if not, you will get a error from R2_resid,  "Matrix is not block-diagonal"

phy.nu <- rtree(n = n)
# Generate random data
e <- signal ^ 0.5 * rTraitCont(phy.nu, model = 'BM', sigma = 1) +
  (1 - signal) ^ 0.5 * rnorm(n = n)
d$x <- x[match(names(e), names(x))]
d$y <- b1 * x + e
rownames(d) <- phy.nu$tip.label
d$sp <- phy.nu$tip.label

weights <- diag(vcv.phylo(phy.nu))
z.f <- gls(y ~ x, data = d,
           correlation = corPagel(1, phy.nu, form = ~sp),
           weights = varFixed(~weights), method = "ML")
z.x <- gls(y ~ 1, data = d,
           correlation = corPagel(1, phy.nu, form = ~sp),
           weights = varFixed(~weights), method = "ML")
z.v <- lm(y ~ x, weights = 1/weights, data = d)

R2(z.f, z.x)
```

```
R2(z.f, z.v)
R2(z.f)

# PGLMM with one fixed effect ----
n <- 100
b1 <- 1.5
signal <- 2

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rnorm(n)
d <- data.frame(x = x, y = 0)

e <- signal * rTraitCont(phy, model = 'BM', sigma = 1)
e <- e[match(phy$tip.label, names(e))]

d$y <- rbinom(n = n, size = 1, prob = inv.logit(b1 * d$x + e))
rownames(d) <- phy$tip.label
# Use the function phyloglm() from the phylolm package.
z.f <- phyloglm(y ~ x, data = d, start.alpha = 1, phy = phy)
z.x <- phyloglm(y ~ 1, data = d, phy = phy, start.alpha = min(20, z.f$alpha))
z.v <- glm(y ~ x, data = d, family = 'binomial')

R2(z.f, z.x)
R2(z.f, z.v)
R2(z.f)

# Use the function pglmm_compare() from the phyr package. Note that this is a
# different model from phyloglm()
z.f <- pglmm_compare(y ~ x, data = d, family = 'binomial', phy = phy, REML = FALSE)
z.x <- pglmm_compare(y ~ 1, data = d, family = 'binomial', phy = phy, REML = FALSE)
z.v <- glm(y ~ x, data = d, family = 'binomial')

R2(z.f, z.x)
R2(z.f, z.v)
R2(z.f)

# A community example of pglmm {phyr} ----
library(mvtnorm)
nspp <- 6
nsite <- 4

# Simulate a phylogeny that has a lot of phylogenetic signal (power = 1.3)
phy <- compute.brlen(rtree(n = nspp), method = "Grafen", power = 1.3)

# Simulate species means
sd.sp <- 1
mean.sp <- rTraitCont(phy, model = "BM", sigma=sd.sp ^ 2)

# Replicate values of mean.sp over sites
Y.sp <- rep(mean.sp, times = nsite)
```

```
# Simulate site means
sd.site <- 1
mean.site <- rnorm(nsite, sd = sd.site)

# Replicate values of mean.site over sp
Y.site <- rep(mean.site, each = nspp)

# Compute a covariance matrix for phylogenetic attraction
sd.attract <- 1
Vphy <- vcv(phy)

# Standardize the phylogenetic covariance matrix to have determinant = 1.
# (For an explanation of this standardization, see subsection 4.3.1 in Ives (2018))
Vphy <- Vphy/(det(Vphy)^(1/nspp))

# Construct the overall covariance matrix for phylogenetic attraction.
# (For an explanation of Kronecker products, see subsection 4.3.1 in the book)
V <- kronecker(diag(nrow = nsite, ncol = nsite), Vphy)
Y.attract <- array(t(rmvnorm(n = 1, sigma = sd.attract^2*V)))

# Simulate residual errors
sd.e <- 1
Y.e <- rnorm(nspp * nsite, sd = sd.e)

# Construct the dataset
d <- data.frame(sp = rep(phy$tip.label, times = nsite), site = rep(1:nsite, each = nspp))

# Simulate abundance data
d$Y <- Y.sp + Y.site + Y.attract + Y.e

# Full and reduced models
z.f <- pglmm(Y ~ 1 + (1|sp__) + (1|site) + (1|sp__@site),
             data = d, cov_ranef = list(sp = phy), REML = FALSE)
z.nested <- pglmm(Y ~ 1 + (1|sp__) + (1|site),
                  data = d, cov_ranef = list(sp = phy), REML = FALSE)
z.sp <- pglmm(Y ~ 1 + (1|sp) + (1|site),
              data = d, cov_ranef = list(sp = phy), REML = FALSE)

R2(z.f, z.nested)
R2(z.nested, z.sp)
R2(z.f)
```

R2_lik *Calculate R2_lik*

Description

Calculate partial and total R2s for LMM, GLMM, PGLS, and PGLMM using R2_lik, an R2 based on the likelihood of the fitted model.

Usage

```
R2_lik(mod = NULL, mod.r = NULL)
```

Arguments

mod	A regression model with one of the following classes: 'lm', 'glm', 'lmerMod', 'glmerMod', 'phylolm', 'phyloglm', 'gls', 'pglmm', pglmm_compare' or 'communityPGLMM'.
mod.r	A reduced model; if not provided, the total R2 will be given by setting 'mod.r' to the model corresponding to 'mod' with the intercept as the only predictor.

Details

`R2_lik()` is implemented as

$$partialR2 = 1 - exp(-2/n * (logLik(mod.f) - logLik(mod.r)))$$

where 'mod.f' and 'mod.r' are the full and reduced models, respectively. The total R2 is given when 'mod.r' is the model corresponding to mod.f that contains only the intercept. For GLMMs and PGLMMs, `R2_lik()` is standardized to have a maximum of one following Nagelkerke (1991). Although you can use R2_lik with models fit with REML, you really shouldn't, because this makes it impossible to compare values when reduced models differ in independent variables (fixed effects).

`R2_lik()` is also computed for LMMs and GLMMs in the MuMIn package.

Value

R2_lik value.

Author(s)

Anthony R. Ives

References

Ives A.R. and Li D. 2018. rr2: An R package to calculate R2s for regression models. Journal of Open Source Software. DOI:10.21105/joss.01028

Ives A.R. 2018. R2s for Correlated Data: Phylogenetic Models, LMMs, and GLMMs. Systematic Biology, Volume 68, Issue 2, March 2019, Pages 234-251. DOI:10.1093/sysbio/syy060

Nagelkerke 1991. A note on a general definition of the coefficient of determination. Biometrika 78:691–692.

See Also

MuMIn, lme4, ape, phylolm, phyr, pez

Examples

```
library(ape)
library(phylolm)
library(lme4)
library(nlme)
library(phyr)

set.seed(12345)
p1 <- 10
nsample <- 10
n <- p1 * nsample

d <- data.frame(x1 = 0, x2 = 0, u1 = rep(1:p1, each = nsample),
                u2 = rep(1:p1, times = nsample))
d$u1 <- as.factor(d$u1)
d$u2 <- as.factor(d$u2)

b1 <- 1
b2 <- -1
sd1 <- 1.5

d$x1 <- rnorm(n = n)
d$x2 <- rnorm(n = n)
d$y.lmm <- b1 * d$x1 + b2 * d$x2 +
  rep(rnorm(n = p1, sd = sd1), each = nsample) +
  rep(rnorm(n = p1, sd = sd1), times = nsample) +
  rnorm(n = n)

prob <- inv.logit(b1 * d$x1 + rep(rnorm(n = p1, sd = sd1), each = nsample))
d$y.glmm <- rbinom(n = n, size = 1, prob = prob)

# LMM with two fixed and two random effects ----
z.f <- lmer(y.lmm ~ x1 + x2 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.x <- lmer(y.lmm ~ x1 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.v <- lmer(y.lmm ~ 1 + (1 | u2), data = d, REML = FALSE)
z.0 <- lm(y.lmm ~ 1, data = d)

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)

# GLMM with one fixed and one random effect ----
z.f <- glmer(y.glmm ~ x1 + (1 | u1), data = d, family = 'binomial')
z.x <- glmer(y.glmm ~ 1 + (1 | u1), data = d, family = 'binomial')
z.v <- glm(y.glmm ~ x1, data = d, family = 'binomial')

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)

# PGLS with a single fixed effect ----
n <- 100
```

```
d <- data.frame(x = array(0, dim = n), y = 0)

b1 <- 1.5
signal <- 0.5

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rTraitCont(phy.x, model = 'BM', sigma = 1)
e <- signal^0.5 * rTraitCont(phy, model = 'BM', sigma = 1) + (1-signal)^0.5 * rnorm(n = n)
d$x <- x[match(names(e), names(x))]
d$y <- b1 * x + e
rownames(d) <- phy$tip.label
d$sp <- phy$tip.label

z.f <- pglmm_compare(y ~ x, data = d, phy = phy, REML=FALSE)
z.x <- pglmm_compare(y ~ 1, data = d, phy = phy, REML=FALSE)
z.v <- glm(y ~ x, data = d)

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)

# These data can also be fit with phylolm() in {phylolm}
z.f <- phylolm(y ~ x, phy = phy, data = d, model = 'lambda')
z.x <- phylolm(y ~ 1, phy = phy, data = d, model = 'lambda')
z.v <- lm(y ~ x, data = d)

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)

# This also works for models fit with gls() in {nlme}
z.f <- gls(y ~ x, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.x <- gls(y ~ 1, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.v <- lm(y ~ x, data = d)

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)

# PGLMM with one fixed effect ----
n <- 100
b1 <- 1.5
signal <- 2

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rnorm(n)
d <- data.frame(x = x, y = 0)
```

```
e <- signal * rTraitCont(phy, model = 'BM', sigma = 1)
e <- e[match(phy$tip.label, names(e))]

d$y <- rbinom(n = n, size = 1, prob = inv.logit(b1 * d$x + e))
rownames(d) <- phy$tip.label

z.f <- phyloglm(y ~ x, data = d, start.alpha = 1, phy = phy)
z.x <- phyloglm(y ~ 1, data = d, phy = phy, start.alpha = min(20, z.f$alpha))
z.v <- glm(y ~ x, data = d, family = 'binomial')

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)

# These data can also be fit with pglmm_compare(), although note that
# this is a different model from phyloglm()

z.f <- pglmm_compare(y ~ x, data = d, family = "binomial", phy = phy, REML=FALSE)
z.x <- pglmm_compare(y ~ 1, data = d, family = "binomial", phy = phy, REML=FALSE)
z.v <- glm(y ~ x, data = d, family = "binomial")

R2_lik(z.f, z.x)
R2_lik(z.f, z.v)
R2_lik(z.f)
```

`R2_pred` *Calculate R2_pred*

Description

Calculate partial and total R2s for LMM, GLMM, PGLS, and PGLMM using R2_pred, an R2 based on the variance of the difference between the observed and predicted values of a fitted model.

Usage

```
R2_pred(mod = NULL, mod.r = NULL, gaussian.pred = "tip_rm", phy = NULL)
```

Arguments

`mod`	A regression model with one of the following classes: 'lm', 'glm', 'lmerMod', 'glmerMod', 'phylolm', 'gls', 'pglmm', pglmm_compare', 'binaryPGLMM', or 'communityPGLMM'.
`mod.r`	A reduced model; if not provided, the total R2 will be given by setting 'mod.r' to the model corresponding to 'mod' with the intercept as the only predictor.
`gaussian.pred`	For models of classes 'pglmm' and 'pglmm_compare' when family = gaussian, which type of prediction to calculate.
`phy`	The phylogeny for phylogenetic models (as a 'phylo' object), which must be specified for models of class 'phylolm'.

Details

R2_pred works with classes 'lm', 'glm', 'lmerMod', 'glmerMod', 'phylolm', 'phyloglm', 'gls', 'pglmm', 'pglmm_compare', binaryPGLMM', and 'communityPGLMM' (family = gaussian and binomial).

LMM (lmerMod), GLMM (glmerMod), PGLMM (pglmm, pglmm_compare, binaryPGLMM and communityPGLMM):

$$partialR2 = 1 - var(y - y.fitted.f)/var(y - y.fitted.r)$$

where y are the observed data, and y.fitted.f and y.fitted.r are the fitted (predicted) values from the full and reduced models. For GLMMs and PGLMMs, the values of y.fitted are in the space of the raw data (as opposed to the 'Normal' or 'latent' space). When the reduced model 'mod.r' is not specified, the total R2 is computing using the reduced model with only the intercept.

For pglmm and pglmm_compare with gaussian models, the default method for computing predicted values is "nearest_node" to correspond to predicted values in lmer, although the method "tip_rm" can be specified to correspond to the analyses in Ives (2018).

Note that the version of `binaryPGLMM()` in the package ape is replaced by a version contained within rr2 that outputs all of the required information for the calculation of R2_resid.

PGLS (phylolm and gls):

For PGLS, the total R2_pred is computed by removing each datum one at a time, predicting its value from the fitted model, repeating this for all data points, and then calculating the variance of the difference between observed and fitted values. The predictions are calculated as

$$res.predicted[j] = V[j, -j]solve(V[-j, -j])res[-j]$$

where res[-j] is a vector of residuals with datum j removed, V[-j,-j] is the phylogenetic covariance matrix with row and column j removed, and V[j, -j] is row j of covariance matrix V with element j removed. The partial R2_pred is calculated from the total R2_pred from full and reduced models as

$$partialR2 = 1 - (1 - R2_predred.f)/(1 - R2_predred.r)$$

Note that `phylolm()` can have difficulties in finding solutions when there is no phylogenetic signal; when the estimate indicates no phylogenetic signal, you should refit the model with the corresponding LM.

LM (lm) and GLM (glm):

For compatibility and generating reduced models, rr2 will compute R2_pred for LM and GLM that correspond to LMM/PGLS and GLMM/PGLMM.

Author(s)

Anthony R. Ives

References

Ives A.R. and Li D. 2018. rr2: An R package to calculate R2s for regression models. Journal of Open Source Software. DOI:10.21105/joss.01028

Ives A.R. 2018. R2s for Correlated Data: Phylogenetic Models, LMMs, and GLMMs. Systematic Biology, Volume 68, Issue 2, March 2019, Pages 234-251. DOI:10.1093/sysbio/syy060

See Also

MuMIn, lme4, ape, phylolm, pez

Examples

```
library(ape)
library(phylolm)
library(lme4)
library(nlme)
library(phyr)

set.seed(12345)
p1 <- 10
nsample <- 10
n <- p1 * nsample

d <- data.frame(x1 = 0, x2 = 0, u1 = rep(1:p1, each = nsample),
                u2 = rep(1:p1, times = nsample))
d$u1 <- as.factor(d$u1)
d$u2 <- as.factor(d$u2)

b1 <- 1
b2 <- -1
sd1 <- 1.5

d$x1 <- rnorm(n = n)
d$x2 <- rnorm(n = n)
d$y.lmm <- b1 * d$x1 + b2 * d$x2 +
  rep(rnorm(n = p1, sd = sd1), each = nsample) +
  rep(rnorm(n = p1, sd = sd1), times = nsample) +
  rnorm(n = n)

prob <- inv.logit(b1 * d$x1 + rep(rnorm(n = p1, sd = sd1), each = nsample))
d$y.glmm <- rbinom(n = n, size = 1, prob = prob)

# LMM with two fixed and two random effects ----
z.f <- lmer(y.lmm ~ x1 + x2 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.x <- lmer(y.lmm ~ x1 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.v <- lmer(y.lmm ~ 1 + (1 | u2), data = d, REML = FALSE)
z.0 <- lm(y.lmm ~ 1, data = d)

R2_pred(z.f, z.x)
R2_pred(z.f, z.v)
R2_pred(z.f)
```

```
# GLMM with one fixed and one random effect ----
z.f <- glmer(y.glmm ~ x1 + (1 | u1), data = d, family = 'binomial')
z.x <- glmer(y.glmm ~ 1 + (1 | u1), data = d, family = 'binomial')
z.v <- glm(y.glmm ~ x1, data = d, family = 'binomial')

R2_pred(z.f, z.x)
R2_pred(z.f, z.v)
R2_pred(z.f)

# PGLS with a single fixed effect ----
n <- 100
d <- data.frame(x = array(0, dim = n), y = 0)

b1 <- 1.5
signal <- 0.7

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rTraitCont(phy.x, model = 'BM', sigma = 1)
e <- signal ^ 0.5 * rTraitCont(phy, model = 'BM', sigma = 1) +
  (1 - signal) ^ 0.5 * rnorm(n = n)
d$x <- x[match(names(e), names(x))]
d$y <- b1 * x + e
rownames(d) <- phy$tip.label
d$sp <- phy$tip.label

z.x <- gls(y ~ 1, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.f <- gls(y ~ x, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.v <- lm(y ~ x, data = d)

R2_pred(z.f, z.x)
R2_pred(z.f, z.v)
R2_pred(z.f)

# PGLMM with one fixed effect ----
n <- 100
b1 <- 1.5
signal <- 2

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rnorm(n)
d <- data.frame(x = x, y = 0)

e <- signal * rTraitCont(phy, model = 'BM', sigma = 1)
e <- e[match(phy$tip.label, names(e))]

d$y <- rbinom(n = n, size = 1, prob = inv.logit(b1 * d$x + e))
```

```
rownames(d) <- phy$tip.label

z.f <- pglmm_compare(y ~ x, data = d, family = "binomial", phy = phy)
z.x <- pglmm_compare(y ~ 1, data = d, family = "binomial", phy = phy)
z.v <- glm(y ~ x, data = d, family = "binomial")

R2_pred(z.f, z.x)
R2_pred(z.f, z.v)
R2_pred(z.f)

#' #################
# A community example of pglmm {phyr} contrasting R2_pred when bayes = TRUE and bayes = F

library(mvtnorm)
nspp <- 6
nsite <- 4

# Simulate a phylogeny that has a lot of phylogenetic signal (power = 1.3)
phy <- compute.brlen(rtree(n = nspp), method = "Grafen", power = 1.3)

# Simulate species means
sd.sp <- 1
mean.sp <- rTraitCont(phy, model = "BM", sigma=sd.sp^2)

# Replicate values of mean.sp over sites
Y.sp <- rep(mean.sp, times=nsite)

# Simulate site means
sd.site <- 1
mean.site <- rnorm(nsite, sd=sd.site)

# Replicate values of mean.site over sp
Y.site <- rep(mean.site, each=nspp)

# Compute a covariance matrix for phylogenetic attraction
sd.attract <- 1
Vphy <- vcv(phy)

# Standardize the phylogenetic covariance matrix to have determinant = 1.
# (For an explanation of this standardization, see subsection 4.3.1 in Ives (2018))
Vphy <- Vphy/(det(Vphy)^(1/nspp))

# Construct the overall covariance matrix for phylogenetic attraction.
# (For an explanation of Kronecker products, see subsection 4.3.1 in the book)
V <- kronecker(diag(nrow = nsite, ncol = nsite), Vphy)
Y.attract <- array(t(rmvnorm(n = 1, sigma = sd.attract^2*V)))

# Simulate residual errors
sd.e <- 1
Y.e <- rnorm(nspp*nsite, sd = sd.e)

# Construct the dataset
d <- data.frame(sp = rep(phy$tip.label, times = nsite), site = rep(1:nsite, each = nspp))
```

```
# Simulate abundance data
d$Y <- Y.sp + Y.site + Y.attract + Y.e

# Full and reduced models
z.f <- pglmm(Y ~ 1 + (1|sp__) + (1|site) + (1|sp__@site),
        data = d, cov_ranef = list(sp = phy), REML = FALSE)
z.nested <- pglmm(Y ~ 1 + (1|sp__) + (1|site),
        data = d, cov_ranef = list(sp = phy), REML = FALSE)
z.sp <- pglmm(Y ~ 1 + (1|sp) + (1|site),
        data = d, cov_ranef = list(sp = phy), REML = FALSE)

R2_pred(z.f, z.nested)
R2_pred(z.nested, z.sp)
R2_pred(z.f)

# vector - matrix
# These are generally larger when gaussian.pred = "nearest_node"
R2_pred(z.f, z.nested, gaussian.pred = "nearest_node")
R2_pred(z.nested, z.sp, gaussian.pred = "nearest_node")
R2_pred(z.f, gaussian.pred = "nearest_node")

# # When bayes = TRUE, gaussian.pred does not work.
# # Commented out because INLA is not on CRAN
# z.f.bayes <- pglmm(Y ~ 1 + (1|sp__) + (1|site) + (1|sp__@site),
#                data = d, cov_ranef = list(sp = phy), bayes = TRUE)
# z.nested.bayes <- pglmm(Y ~ 1 + (1|sp__) + (1|site),
#                data = d, cov_ranef = list(sp = phy), bayes = TRUE)
# z.sp.bayes <- pglmm(Y ~ 1 + (1|sp) + (1|site),
#                data = d, cov_ranef = list(sp = phy), bayes = TRUE)
#
# R2_pred(z.f.bayes, z.nested.bayes)
# R2_pred(z.nested.bayes, z.sp.bayes)
# R2_pred(z.f.bayes)
```

`R2_resid` *Calculate R2_resid*

Description

Calculate partial and total R2s for LMM, GLMM, PGLS, and PGLMM using R2_resid, an extension of ordinary least-squares (OLS) R2s. For LMMs and GLMMs, R2_resid is related to the method proposed by Nakagawa and Schielzeth (2013).

Usage

```
R2_resid(
  mod = NULL,
  mod.r = NULL,
  sigma2_d = c("s2w", "NS", "rNS"),
```

```
  phy = NULL
)
```

Arguments

mod	A regression model with one of the following classes: 'lm', 'glm', 'lmerMod', 'glmerMod', 'phylolm', 'gls', 'pglmm_compare' or 'binaryPGLMM'. For 'glmerMod', only family = c('binomial', 'poisson') are supported.
mod.r	A reduced model; if not provided, the total R2 will be given by setting 'mod.r' to the model corresponding to 'mod' with the intercept as the only predictor.
sigma2_d	Distribution-specific variance σ^2_d (see Details). For binomial GLMs, GLMMs and PGLMMs with logit link functions, options are c('s2w', 'NS', 'rNS'). For binomial GLMs, GLMMs and PGLMMs with probit link functions, options are c('s2w', 'NS'). Other families use 's2w'.
phy	The phylogeny for phylogenetic models (as a 'phylo' object), which must be specified for models of class 'phylolm'.

Details

R2_resid works with classes 'lm', 'glm', 'lmerMod', 'glmerMod', 'phylolm', 'pglmm_compare', and 'binaryPGLMM'.

LMM (lmerMod):

$$partialR^2 = 1 - \sigma^2_{e.f}/\sigma^2_{e.r}$$

$$totalR^2 = 1 - \sigma^2_{e.f}/var(y)$$

where $\sigma^2_{e.f}$ and $\sigma^2_{e.r}$ are the estimated residual variances from the full and reduced LMM, and var(y) is the total variance of the response (dependent) variable.

GLMM (glmerMod):

$$totalR^2 = 1 - \sigma^2_d/(\sigma^2_x + \sigma^2_b + \sigma^2_d)$$

where σ^2_x and σ^2_b are the estimated variances associated with the fixed and random effects. σ^2_d is a term that scales the implied 'residual variance' of the GLMM (see Ives 2018, Appendix 1). The default used for σ^2_d is σ^2_w which is computed from the iterative weights of the GLMM. Specifically,

$$\sigma^2_w = var(g'(\mu) * (y - \mu))$$

where g'() is the derivative of the link function, and $(y - \mu)$ is the difference between the data y and their predicted values μ. This is the default option specified by sigma2_d = 's2w'. For binomial models with a logit link function, sigma2_d = 'NS' gives the scaling $\sigma^2_d = \pi^2/3$ from Nakagawa and Schielzeth (2013), and sigma2_d = 'rNS' gives $\sigma^2_d = 0.8768809 * \pi^2/3$ which is a "corrected" version of 'NS' (see Ives 2018, Appendix 1). For binomial models with a probit link function, sigma2_d = 'NS' gives the scaling $\sigma^2_d = 1$. In general, option sigma2_d = 's2w' will give values lower than sigma2_d = 'NS' and 'rNS', but the values will be closer to `R2_lik()` and `R2_pred()`.

For other forms of sigma2_d from Nakagawa and Schielzeth (2013) and Nakagawa et al. (2017), see the MuMIn package.

Partial R2s are given by the standard formula

$$partialR^2 = 1 - (1 - R^2_{.f})/(1 - R^2_{.r})$$

where R2.f and R2.r are the total R2s for full and reduced models, respectively.

PGLS (phylolm, pglmm_compare):

$$partialR^2 = 1 - c.f * \sigma^2_{.f}/(c.r * \sigma^2_{.r})$$

where $\sigma^2_{.f}$ and $\sigma^2_{.r}$ are the variances estimated for the PGLS full and reduced models, and c.f and c.r are the scaling values for full and reduce models that equal the total sum of phylogenetic branch length estimates. Note that the phylogeny needs to be specified in R2_resid.

`phylolm()` can have difficulties in finding solutions when there is no phylogenetic signal; when the estimate indicates no phylogenetic signal, you should refit the model with the corresponding LM.

PGLMM (pglmm_compare, binaryPGLMM):

The R2_resid for PGLMMs is computed in the same way as the GLMM (glmer), with options sigma_d = c('s2w', 'NS', 'rNS'). The estimated variance of the random effect associated with the phylogeny, σ^2_b, is multiplied by the diagonal elements of the phylogenetic covariance matrix. For binary models, this covariance matrix should be standardized so that all diagonal elements are the same (a contemporaneous or ultrametric phylogenetic tree) (Ives and Garland 2014). In case this is not done, however, the code takes the geometric average of the diagonal elements.

Note that the version of `binaryPGLMM()` in the package ape is replaced by a version contained within rr2 that outputs all of the required information for the calculation of `R2_resid()`

LM (lm) and GLM (glm):

For compatibility and generating reduced models, rr2 will compute `R2_resid()` for LM and GLM that correspond to LMM/PGLS and GLMM/PGLMM.

Value

R2_resid value.

Author(s)

Anthony R. Ives

References

Ives A.R. and Li D. 2018. rr2: An R package to calculate R2s for regression models. Journal of Open Source Software. DOI:10.21105/joss.01028

Ives A.R. 2018. R2s for Correlated Data: Phylogenetic Models, LMMs, and GLMMs. Systematic Biology, Volume 68, Issue 2, March 2019, Pages 234-251. DOI:10.1093/sysbio/syy060

Ives A. R., Garland T., Jr. 2014. Phylogenetic regression for binary dependent variables. In: Garamszegi LZ editor. Modern Phylogenetic Comparative Methods and Their Application in Evolutionary Biology. Berlin Heidelberg, Springer-Verlag, p. 231-261.

Nakagawa S., Schielzeth H. 2013. A general and simple method for obtaining R2 from generalized linear mixed-effects models. Methods in Ecology and Evolution, 4:133-142.

Nakagawa S., Johnson P. C. D., Schielzeth H. 2017. The coefficient of determination R2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. Journal of the Royal Society Interface, 14.

See Also

MuMIn, lme4, ape, phylolm, pez

Examples

```
library(ape)
library(phylolm)
library(lme4)
library(nlme)
library(phyr)

set.seed(12345)
p1 <- 10
nsample <- 10
n <- p1 * nsample

d <- data.frame(x1 = 0, x2 = 0, u1 = rep(1:p1, each = nsample),
                u2 = rep(1:p1, times = nsample))
d$u1 <- as.factor(d$u1)
d$u2 <- as.factor(d$u2)

b1 <- 1
b2 <- -1
sd1 <- 1.5

d$x1 <- rnorm(n = n)
d$x2 <- rnorm(n = n)
d$y.lmm <- b1 * d$x1 + b2 * d$x2 +
  rep(rnorm(n = p1, sd = sd1), each = nsample) +
  rep(rnorm(n = p1, sd = sd1), times = nsample) +
  rnorm(n = n)

prob <- inv.logit(b1 * d$x1 + rep(rnorm(n = p1, sd = sd1), each = nsample))
d$y.glmm <- rbinom(n = n, size = 1, prob = prob)

# LMM with two fixed and two random effects ----
z.f <- lmer(y.lmm ~ x1 + x2 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.x <- lmer(y.lmm ~ x1 + (1 | u1) + (1 | u2), data = d, REML = FALSE)
z.v <- lmer(y.lmm ~ 1 + (1 | u2), data = d, REML = FALSE)
z.0 <- lm(y.lmm ~ 1, data = d)

R2_resid(z.f, z.x)
R2_resid(z.f, z.v)
R2_resid(z.f)
```

```
# GLMM with one fixed and one random effect ----
z.f <- glmer(y.glmm ~ x1 + (1 | u1), data = d, family = 'binomial')
z.x <- glmer(y.glmm ~ 1 + (1 | u1), data = d, family = 'binomial')
z.v <- glm(y.glmm ~ x1, data = d, family = 'binomial')

R2_resid(z.f, z.x)
R2_resid(z.f, z.v)
R2_resid(z.f)

# PGLS with a single fixed effect ----
n <- 100
d <- data.frame(x = array(0, dim = n), y = 0)

b1 <- 1.5
signal <- 0.7

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rTraitCont(phy.x, model = 'BM', sigma = 1)
e <- signal ^ 0.5 * rTraitCont(phy, model = 'BM', sigma = 1) +
  (1 - signal) ^ 0.5 * rnorm(n = n)
d$x <- x[match(names(e), names(x))]
d$y <- b1 * x + e
rownames(d) <- phy$tip.label
d$sp <- phy$tip.label

z.x <- pglmm_compare(y ~ 1, phy = phy, data = d, REML=FALSE)
z.f <- pglmm_compare(y ~ x, phy = phy, data = d, REML=FALSE)
z.v <- lm(y ~ x, data = d)

R2_resid(z.f, z.x)
R2_resid(z.f, z.v)
R2_resid(z.f)

z.x <- phylolm(y ~ 1, phy = phy, data = d, model = 'lambda')
z.f <- phylolm(y ~ x, phy = phy, data = d, model = 'lambda')
z.v <- lm(y ~ x, data = d)

R2_resid(z.f, z.x, phy = phy)
R2_resid(z.f, z.v, phy = phy)
R2_resid(z.f, phy = phy)

# This also works for models fit with gls() in {nlme}
z.f <- gls(y ~ x, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.x <- gls(y ~ 1, data = d, correlation = corPagel(1, phy, form = ~sp), method = "ML")
z.v <- lm(y ~ x, data = d)

R2_resid(z.f, z.x)
R2_resid(z.f, z.v)
R2_resid(z.f)
```

```
# But note that you need to define weights for gls() with non-ultrametric trees;
# if not, you will get a error "Matrix is not block-diagonal"

phy.nu <- rtree(n = n)

# Generate random data
e <- signal ^ 0.5 * rTraitCont(phy.nu, model = 'BM', sigma = 1) +
  (1 - signal) ^ 0.5 * rnorm(n = n)
d$x <- x[match(names(e), names(x))]
d$y <- b1 * x + e
rownames(d) <- phy.nu$tip.label
d$sp <- phy.nu$tip.label

weights <- diag(vcv.phylo(phy.nu))
z.x <- gls(y ~ 1,data = d,
         correlation = corPagel(1, phy.nu, form = ~sp),
         weights=varFixed(~weights), method = "ML")
z.f <- gls(y ~ x,data = d,
         correlation = corPagel(1, phy.nu, form = ~sp),
         weights=varFixed(~weights), method = "ML")
z.v <- lm(y ~ x, weights = 1/weights, data = d)

R2_resid(z.f, z.x)
R2_resid(z.f, z.v)
R2_resid(z.f)

# PGLMM with one fixed effect ----

n <- 100
b1 <- 1.5
signal <- 2

phy <- compute.brlen(rtree(n = n), method = 'Grafen', power = 1)
phy.x <- compute.brlen(phy, method = 'Grafen', power = .0001)

# Generate random data
x <- rnorm(n)
d <- data.frame(x = x, y = 0)

e <- signal * rTraitCont(phy, model = 'BM', sigma = 1)
e <- e[match(phy$tip.label, names(e))]

d$y <- rbinom(n = n, size = 1, prob = inv.logit(b1 * d$x + e))
rownames(d) <- phy$tip.label

# Use the function pglmm_compare in {phyr}.
z.f <- pglmm_compare(y ~ x, data = d, family = "binomial", phy = phy)
z.x <- pglmm_compare(y ~ 1, data = d, family = "binomial", phy = phy)
z.v <- glm(y ~ x, data = d, family = 'binomial')

R2_resid(z.f, z.x)
R2_resid(z.f, z.v)
R2_resid(z.f)
```

rr2 *rr2: An R package to calculate R2s for regression models*

Description

The rr2 package provides methods to calculate R2 for models with correlated errors, including Phylogenetic GLS, Phylogenetic Logistic Regression, LMMs, GLMM, and PGLMM.

transf_phy *Transform a phylogeny based on a phylolm model*

Description

Using a fitted phylolm model to transform branch lengths of a phylogeny

Usage

```
transf_phy(phylolmMod, phy)
```

Arguments

phylolmMod	A fitted phylolm model.
phy	A phylogeny with class 'phylo'.

Value

A transformed phylogeny.

Index