Goal

This package provides three R2s for statistical models with correlated errors including classes: 'lmerMod' (LMM), 'glmerMod' (GLMM), 'phylolm' (Phylogenetic GLS), and 'binaryPGLMM/phyloglm' (Phylogenetic Logistic Regression). Detailed technical descriptions can be found in Ives 2017, preprint.

Installation

This package can be installed with:

```
install.packages("devtools")
devtools::install_github("arives/rr2")
library(rr2)
```

Package structure

This package has three main functions: `R2.ls()`, `R2.lr()`, and `R2.ce()`. You can use them individually in the form of, e.g., `R2.ls(mod, mod.r)` where `mod` is the full model and `mod.r` is the reduce model for partial R2s. If you do not include the reduced model `mod.r`, then the appropriate model with just the intercept is used to give the total R2. When using `R2.ls` and `R2.ce` with PGLS, you need to include the phylo object containing a phylogenetic tree, e.g., `R2.ls(mod, mod.r, phy = phy)`.

You can calculate all three R2s at the same time with `R2(mod, mod.r)`. You can also specify which R2(s) to calculate within this function by turning off unwanted methods, e.g., `R2(mod, mod.r, ce = FALSE)` or `R2(mod, mod.r, ls = FALSE)`.

This package also has some helper functions such as `inv.logit()`, `partialR2()`, and `partialR2adj()`.

Models	Available.R2s
LM	partialR2, partialR2adj
LMM: lmerMod	R2.ce, R2.ls, R2.lr
GLMM: glmerMod	R2.ce, R2.ls, R2.lr
PGLS: phylolm	R2.ce, R2.ls, R2.lr
PLOG: binaryPGLMM	R2.ce, R2.ls, —
PLOG: phyloglm	—, —, R2.lr

Usage: calculating R2s for regression models

First, let's simulate data that will be used to fit various models

```
# data
set.seed(123)
p1 <- 10; nsample <- 10; n <- p1 * nsample
d <- data.frame(x1 = rnorm(n = n),
                x2 = rnorm(n = n),
                u1 = rep(1:p1, each = nsample),
                u2 = rep(1:p1, times = nsample))
d$u1 <- as.factor(d$u1); d$u2 <- as.factor(d$u2)
```

```r
# LMM: y with random intercept
b1 <- 1; b2 <- -1; sd1 <- 1.5
d$y_re_intercept <- b1 * d$x1 + b2 * d$x2 +
  rep(rnorm(n = p1, sd = sd1), each = nsample) +  # random intercept u1
  rep(rnorm(n = p1, sd = sd1), times = nsample) + # random intercept u2
  rnorm(n = n)

# LMM: y with random slope
b1 <- 0; sd1 <- 1; sd.x1 <- 2
d$y_re_slope <- b1 * d$x1 +
  rep(rnorm(n = p1, sd = sd1), each = nsample) + # random intercept u1
  d$x1 * rep(rnorm(n = p1, sd = sd.x1), times = nsample) + # random slope u1
  rnorm(n = n)

# GLMM
b1 <- 1; sd1 <- 1.5
prob <- rr2::inv.logit(b1 * d$x1 + rep(rnorm(n = p1, sd = sd1), each = nsample))
# random intercept u1
d$y_binary <- rbinom(n = n, size = 1, prob = prob)

# PGLS
b1 <- 1.5; signal <- 0.7
phy <- ape::compute.brlen(ape::rtree(n = n), method = "Grafen", power = 1)
phy.x <- ape::compute.brlen(phy, method = "Grafen", power = .0001)
x_trait <- ape::rTraitCont(phy.x, model = "BM", sigma = 1)
e <- signal^0.5 * ape::rTraitCont(phy, model = "BM", sigma = 1) + (1-signal)^0.5 * rnorm(n=n)
d$x_trait <- x_trait[match(names(e), names(x_trait))]
d$y_pgls <- b1 * x_trait + e
rownames(d) <- phy$tip.label

# Phylogenetic Logistic Regression
b1 <- 1.5; signal <- 2
e <- signal * ape::rTraitCont(phy, model = "BM", sigma = 1)
e <- e[match(phy$tip.label, names(e))]
d$y_phy_binary <- rbinom(n = n, size = 1, prob = rr2::inv.logit(b1 * d$x1 + e))

head(d)
```

```
##              x1          x2 u1 u2 y_re_intercept y_re_slope y_binary
## t31 -0.56047565 -0.71040656  1  1       3.053041 -0.2790159        1
## t37 -0.23017749  0.25688371  1  2       3.794671  1.7435372        0
## t8   1.55870831 -0.24669188  1  3       8.062178 -0.3410566        1
## t70  0.07050839 -0.34754260  1  4       3.649759  0.5076822        0
## t53  0.12928774 -0.95161857  1  5       2.526704  0.2830316        0
## t13  1.71506499 -0.04502772  1  6       7.631604 -8.5551981        0
##         x_trait     y_pgls y_phy_binary
## t31 -2.07597968 -2.8257102            0
## t37 -0.31921893 -0.6918108            0
## t8  -0.24097587 -0.3359352            1
## t70 -0.08278377 -0.6383157            0
## t53 -1.60010819 -1.3718365            0
## t13 -1.52297135 -2.0347222            1
```

Then, let's fit some models and calculate their R2s.

LMM

```
library(rr2)
```

```
## Loading required package: Matrix
```

```
z.f.lmm <- lme4::lmer(y_re_intercept ~ x1 + x2 + (1 | u1) + (1 | u2), data = d, REML = F)
z.x.lmm <- lme4::lmer(y_re_intercept ~ x1 + (1 | u1) + (1 | u2), data = d, REML = F)
z.v.lmm <- lme4::lmer(y_re_intercept ~ 1 + (1 | u2), data = d, REML = F)
z.0.lmm <- lm(y_re_intercept ~ 1, data = d)

R2(mod = z.f.lmm, mod.r = z.x.lmm)
```

```
##     R2s     value
## 1 R2_lr 0.5356524
## 2 R2_ls 0.6036311
## 3 R2_ce 0.6087728
```

```
R2(mod = z.f.lmm, mod.r = z.v.lmm)
```

```
##     R2s     value
## 1 R2_lr 0.7441745
## 2 R2_ls 0.8373347
## 3 R2_ce 0.8559029
```

```
R2(mod = z.f.lmm, mod.r = z.0.lmm)
```

```
##     R2s     value
## 1 R2_lr 0.7762978
## 2 R2_ls 0.8767789
## 3 R2_ce 0.8991618
```

```
R2(mod = z.f.lmm) # if omit mod.r, default will be the simplest model, such as z.0.lmm here.
```

```
##     R2s     value
## 1 R2_lr 0.7762978
## 2 R2_ls 0.8767789
## 3 R2_ce 0.8991618
```

GLMM

```
z.f.glmm <- lme4::glmer(y_binary ~ x1 + (1 | u1), data = d, family = "binomial")
z.x.glmm <- lme4::glmer(y_binary ~ 1 + (1 | u1), data = d, family = "binomial")
z.v.glmm <- glm(y_binary ~ x1, data = d, family = "binomial")

R2(mod = z.f.glmm, mod.r = z.x.glmm)
```

```
##     R2s     value
## 1 R2_lr 0.1170588
## 2 R2_ls 0.2254840
## 3 R2_ce 0.1373521
```

```
R2(mod = z.f.glmm, mod.r = z.v.glmm)
```

```
##     R2s     value
## 1 R2_lr 0.1990563
```

```
## 2 R2_ls 0.4246935
## 3 R2_ce 0.3545240
```

```
R2(mod = z.f.glmm)
```

```
##     R2s     value
## 1 R2_lr 0.2406380
## 2 R2_ls 0.4530099
## 3 R2_ce 0.3792381
```

PGLS

```
z.x.pgls <- phylolm::phylolm(y_pgls ~ 1, phy = phy, data = d, model = "lambda")
lam.x.pgls <- round(z.x.pgls$optpar, digits = 4)
z.f.pgls <- phylolm::phylolm(y_pgls ~ x_trait, phy = phy, data = d, model = "lambda",
                             starting.value = 0.98 * lam.x.pgls + 0.01)
z.v.pgls <- lm(y_pgls ~ x_trait, data = d)

# phy is needed for phylogenetic models' R2.ls and R2.ce
R2(mod = z.f.pgls, mod.r = z.v.pgls, phy = phy)
```

```
##     R2s     value
## 1 R2_lr 0.2353912
## 2 R2_ls 0.3590018
## 3 R2_ce 0.3114035
```

```
R2(mod = z.f.pgls, phy = phy)
```

```
##     R2s     value
## 1 R2_lr 0.8642865
## 2 R2_ls 0.8862266
## 3 R2_ce 0.8777782
```

Phylogenetic Logistic Regression

```
z.f.plog <- rr2::binaryPGLMM(y_phy_binary ~ x1, data = d, phy = phy)
z.x.plog <- rr2::binaryPGLMM(y_phy_binary ~ 1, data = d, phy = phy)
z.v.plog <- glm(y_phy_binary ~ x1, data = d, family = "binomial")

# R2.lr can't be used with binaryPGLMM because it is not a ML method
R2(mod = z.f.plog, mod.r = z.x.plog, lr = FALSE)
```

```
##     R2s     value
## 1 R2_ls 0.7124029
## 2 R2_ce 0.3344832
```

```
R2(mod = z.f.plog, lr = FALSE)
```

```
##     R2s     value
## 1 R2_ls 0.7291481
## 2 R2_ce 0.5531285
```

```
z.f.plog2 <- phylolm::phyloglm(y_phy_binary ~ x1, data = d, start.alpha = 1, phy = phy)
z.x.plog2 <- phylolm::phyloglm(y_phy_binary ~ 1, data = d, phy = phy,
```

```
                                start.alpha = min(20, z.f.plog2$alpha))
z.v.plog2 <- glm(y_phy_binary ~ x1, data = d, family = "binomial")

# R2.ls and R2.ce do not apply for phyloglm
R2(z.f.plog2, z.x.plog2, ls = FALSE, ce = FALSE)
```

```
##     R2s     value
## 1 R2_lr 0.3853273
```

```
# alternate
R2.lr(z.f.plog2, z.x.plog2)
```

```
## [1] 0.3853273
```

Citation

Please cite the following paper if you find this package useful:

> Anthony Ives. 2017. R2s for Correlated Data: Phylogenetic Models, LMMs, and GLMMs. bioRxiv 144170. doi: https://doi.org/10.1101/144170