Computing heritability

Facundo Muñoz

2017-04-12 breedR version: 0.12

Contents

Introduction **1**

Case 1: Explicit `genetic` component with `method = 'ai'` **1**

Case 2: Using custom heritability formulae **2**

Case 3: Using Bootstrap estimation **4**

References **7**

Introduction

Heritability is usually a desired outcome whenever a genetic component is involved in a model. There are three alternative ways for computing it.

1. Simulation from the asymptotic distribution (Meyer and Houle, 2013)
2. Delta Method (Oehlert, 1992)
3. Bootstrap estimation (Effron and Tibshirani, 1993; Davison and Hinkley, 1997)

Not all are possible under all circumstances. Moreover, each one bears different degrees of approximation. The Bootstrap estimation (3) is the most reliable, but the most difficult to use in practice. Simulating from the asymptotic distribution (1) is the default approach, and gives a reasonable approximation whenever N is large and the variance components are not too small. The Delta Method (3) implies yet another approximation on top of the asymptotic assumption, and is to be used as a last resource.

Case 1: Explicit `genetic` component with `method = 'ai'`

This is the easiest case. `breedR` computes and returns the heritability automatically as

$$h^2 = \frac{\sigma_a^2}{\sigma_P^2},$$

where σ_a^2 is the additive-genetic variance and σ_P^2 is the phenotypic variance, computed as the sum of all the variance components in the model.

```
res.animal <- remlf90(fixed  = phe_X ~ 1,
                      random = ~ gg,
                      genetic = list(model = 'add_animal',
                                     pedigree = globulus[, 1:3],
                                     id = 'self'),
                      data = globulus)
summary(res.animal)
```

```
## Formula: phe_X ~ 0 + Intercept + gg + pedigree
##    Data: globulus
##   AIC  BIC logLik
##  5857 5872  -2926
##
## Parameters of special components:
##
##
## Variance components:
##          Estimated variances  S.E.
## gg                     2.356 1.249
## genetic                3.632 1.649
## Residual              14.271 1.561
##
##              Estimate    S.E.
## Heritability   0.1795 0.08253
##
## Fixed effects:
##            value s.e.
## Intercept 14.797 0.47
```

The heritability is shown automatically in the `summary()` along with its standard error. In this case, the formula used was genetic/(gg+genetic+Residual)'.

The calculation method is by simulation from the asymptotic Gaussian joint sampling distribution of the variance components. The approximation makes use of the Average Information matrix, and therefore can only be used when `method = 'ai'`.

In summary, there are two requirements for this approach to work:

- `method = 'ai'`
- explicit `genetic` component in the model

Note that a **progeny trial** with a specific genetic structure can be fitted in this way by manually building the pedigree.

For instance, for a half-sib family structure, you can use the family code as a fictitious mother in a pedigree, and use `NA` for the fathers.

Case 2: Using custom heritability formulae

Under some circumstances, you need to use an alternative formulation for the heritability. For example, you might want to exclude the spatial variance from the denominator, in order to compare the heritability estimates from different sites (with potentially different spatial variances).

You have two alternative approaches for this:

1. Specifying the formula using the option `se_covar_function` in the argument `progsf90.options`.
2. Using the Delta Method with the inverse Average-Information matrix.

2.1 Specifying and explicit function of the variance components

The methodology is the same as in Case 1 (i.e. simulation from the asymptotic joint distribution), but with a different formula.

In this example, I will simulate a half-sib family genetic structure, and use the following formula for the heritability, which excludes the variance of the random blocks from the denominator:

$$h^2 = \frac{4\sigma_f^2}{\sigma_f^2 + \sigma^2}$$

```r
globulus <- transform(globulus,
                      fam = factor(mum, exclude = 0))

h2fml <- '4*G_3_3_1_1/(G_3_3_1_1+R_1_1)'

res.fml <- remlf90(fixed  = phe_X ~ gg,
                   random = ~ bl + fam,
                   progsf90.options = paste('se_covar_function h2', h2fml),
                   data = globulus)
summary(res.fml)
```

```
## Formula: phe_X ~ 0 + gg + bl + fam
##    Data: globulus
##   AIC  BIC logLik
##  5677 5692  -2836
##
## Parameters of special components:
##
##
## Variance components:
##          Estimated variances   S.E.
## bl                     2.644 1.0793
## fam                    1.101 0.4187
## Residual              14.391 0.6642
##
##    Estimate   S.E.
## h2   0.2823 0.1039
##
## Fixed effects:
##        value   s.e.
## gg.1  13.533 0.6157
## gg.2  14.027 0.8311
## gg.3  16.116 0.6618
## gg.4  11.863 0.8546
## gg.5  15.885 0.7221
## gg.6  10.211 1.6500
## gg.7  13.995 1.4954
## gg.8  15.694 0.6422
## gg.9  16.474 0.7308
## gg.10 12.845 1.0978
## gg.11 16.723 0.7603
## gg.12 16.922 1.0493
## gg.13 16.297 1.4954
## gg.14 14.424 0.5262
```

The formula `h2fml` needs to be written without spaces and includes random effects (`G`) and the residual variances (`R`). More specifically, `G_3_3_1_1` means: the variance of *the third effect* in the model for the *first trait*, while `R_1_1` stands for the residual variance for the first trait.

Since `breedR` for the moment supports single trait models only, the last two numbers for all the terms in the formula will always be 1. For the effect numbers count fixed effects also: `gg` (1), `bl` (2), `fam` (3).

For more details, see the `se_covar_function` option in the AIREMLF90 manual.

2.2 Using the Delta Method

This method uses a second-order Taylor expansion of the function of the variance components (i.e. the formula for the heritability) about its mean, and then takes variances. Assuming the Gaussian asymptotic distribution, the covariance matrix is given by the inverse Average-Information matrix.

This matrix can be recovered from a `breedR` result (as long as `method = 'ai'` has been used) as follows:

```
(invAI <- res.fml$reml$invAI)
```

```
##                bl       fam      resid
## bl     1.1648000  0.002172 -0.0070802
## fam    0.0021720  0.175290 -0.0325110
## resid -0.0070802 -0.032511  0.4411100
```

Note that the square root of this matrix's diagonal gives the Standard Errors reported in the previous summary for the variance componens.

```
sqrt(diag(invAI))
```

```
##        bl       fam     resid
## 1.0792590 0.4186765 0.6641611
```

You can use the `R` package `msm` to handle the differentiation details for you.

```
if (require(msm, quietly = TRUE)) {

  h2hat <- with(as.list(res.fml$var[, "Estimated variances"]),
                4*fam/(fam+Residual))
  h2se <- deltamethod(~ 4*x2/(x2+x3),
                      res.fml$var[, "Estimated variances"],
                      invAI)
  cat(paste0('Heritability (delta method): ',
             round(h2hat, 2), ' (',
             round(h2se, 2), ')'))
}
```

```
## Heritability (delta method): 0.28 (0.1)
```

I think there is currently little reason for using this method, as 2.1 is feasible under the same conditions, and is always preferable.

Case 3: Using Bootstrap estimation

Bootstrap estimation is computationally expensive, since it needs to fit the same model many times, and more statistically involved. But is the best option if you can afford it:

- available under all circumstances (even with EM-REML)
- makes no assumptions about the sampling distribution of the variance components

Moreover, it is the only option available to compute heritability in genetic `competition` or spatial `splines` models, which require `method = 'em'` and therefore there is no access to the Average-Information matrix.

Furthermore, for models where the spatial arrangement matters (such as competition or splines), taking subsamples may not be a good idea. The sampling distribution of the variance component REML estimates will likely be wider for sparsely arranged subsamples of your data.

In these cases, rather than resampling from your data, it is better to simulate full datasets from the fitted model.

For computational reasons, I will illustrate with a toy example. For competition or splines models, however, consider parallelization of this code (see R-package `parallel` and/or `?remote` capabilities in `breedR`)

Step 1: Fit the model to your data

I use AIREML here in order to make it faster. Note that with EMREML you would not have any estimate of the Standard Errors. Not to mention the heritability and its S.E.

```r
globulus <- transform(globulus,
                      fam = factor(mum, exclude = 0))

res <- remlf90(fixed  = phe_X ~ 1,
               random = ~ bl + fam,
               data = globulus)
```

Step 2: write a function to simulate data from your fitted model

You can leverage `breedR.sample.phenotype()`.

```r
resample_globulus <- function(N = 1, fit = res) {

  dat <- breedR.sample.phenotype(
    fixed = 1,
    random = list(bl = list(nlevels = 15,
                            sigma2 = fit$var['bl', 1]),
                  fam = list(nlevels = 63,
                             sigma2 = fit$var['fam', 1])),
    residual.variance = fit$var['Residual', 1],
    N = nrow(model.frame(fit))
  )

  # rename variables to keep the original names
  names(dat) <- gsub("rnd\\.", "", names(dat))

  return(dat)
}
```

Step 3: write a function to fit a simulated dataset and extract the target values

```r
sim_target <- function(dat = resample_globulus()) {

  ## Fit the original model
  ## avoiding messages about initial variances spec
  res <- suppressMessages(
    remlf90(fixed  = phenotype ~ 1,
            random = ~ bl + fam,
            data = dat)
  )

  ## Extract point estimates of all variances
  ## and point estimate of heritability
  ans <- res$var[, 'Estimated variances']
  ans <- c(ans, h2 = unname(4*ans['fam']/sum(ans)))

  return(ans)
}
```

Repeated calls to this function will return different estimates, even though the data generating process is the same.

```r
sim_target()
```

```
##          bl         fam   Residual          h2
##   2.3239000   3.0598000 15.2250000   0.5938851
```

```r
sim_target()
```

```
##          bl         fam   Residual          h2
##   1.9541000   3.6643000 13.8770000   0.7518286
```

```r
sim_target()
```

```
##         bl        fam  Residual         h2
##   2.665700   2.836100 14.201000   0.575776
```

The distribution of each these values is known as the **sampling distribution**, and its standard deviation is the Standard Error of the corresponding estimators.

To get an idea of these distributions, we need to sample many times

```r
empirical_dist <- as.data.frame(t(replicate(500, sim_target())))

if (require(tidyr, quietly = TRUE)) {

  plotdat <- gather(empirical_dist[, 1:3], 'variable', 'value')
  qplot(value, data = plotdat, facets = ~ variable)
}
```

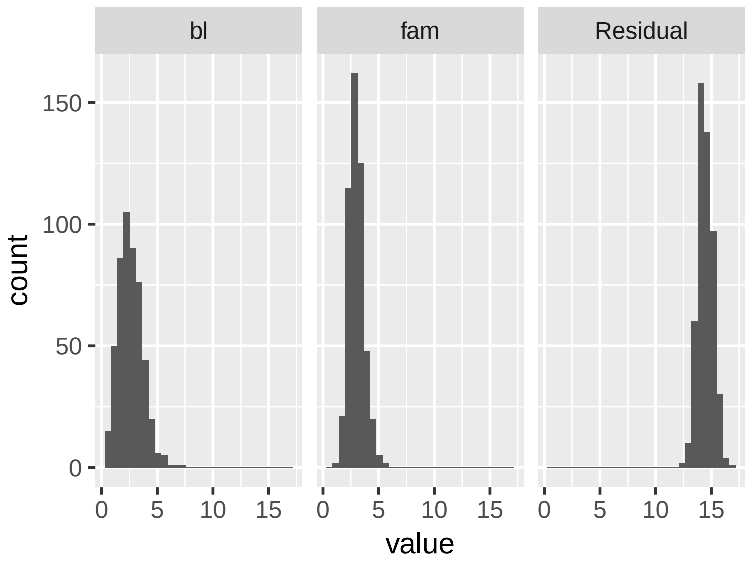

Figure 1: Approximate sampling distribution of variance components

```
qplot(h2, data = empirical_dist, xlim = c(0, 1))
```

```
## Warning: Removed 2 rows containing non-finite values (stat_bin).
```

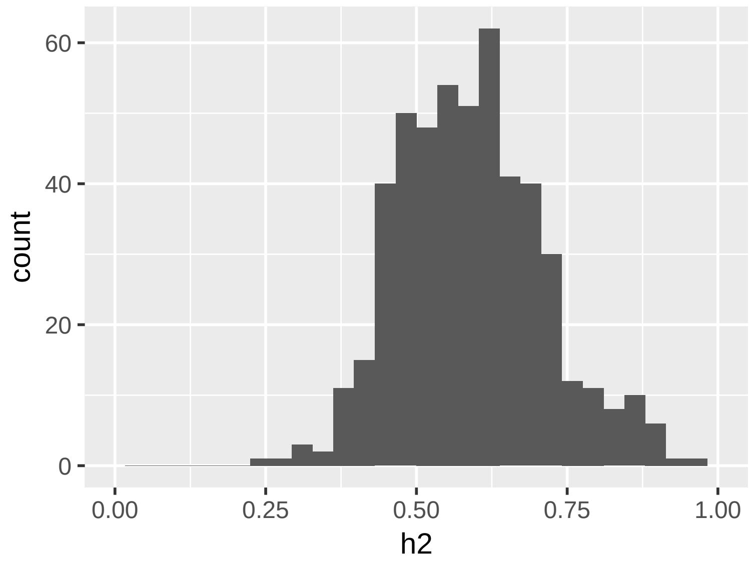

Figure 2: Approximate sampling distribution of heritability

You can compute Standard Errors and Confidence Intervals from numerical summaries of these distributions.

```
cat('Standard Errors:\n')
round(sapply(empirical_dist, sd), 2)
```

```
## Standard Errors:
##       bl      fam Residual       h2
##     1.08     0.71     0.68     0.12
```

```
cat('95% Confidence Intervals:\n')
round(sapply(empirical_dist, quantile, probs = c(.025, .975)), 2)
```

```
## 95% Confidence Intervals:
##         bl  fam Residual   h2
## 2.5%  0.82 1.82    13.26 0.38
## 97.5% 4.95 4.51    15.77 0.86
```

References

Davison, A.C. and Hinkley, D.V. (1997) *Bootstrap Methods and Their Application.* Cambridge University Press.

Efron, B. and Tibshirani, R.J. (1993) *An Introduction to the Bootstrap.* Chapman and Hall.

Meyer, K. and D. Houle (2013) Sampling based approximation of confidence intervals for functions of genetic covariance matrices. *Proceedings of the Twentieth Conference of the Association for the Advancement of Animal Breeding and Genetics*, Number 20, pp. 523-526.

Oehlert, G. W. (1992). A note on the delta method. *The American Statistician* 46 (1), 27-29.