It just happened to use *MCMCglmm* as an example of an R package that can get confused by tibble-style data frames. To make that example, I simulated some pedigree and trait data. Just for fun, let’s look at the simulation code, and use *MCMCglmm* and *AnimalINLA* to get heritability estimates.

First, here is some *AlphaSimR* code that creates a small random mating population, and collects trait and pedigree:

library(AlphaSimR)

## Founder population

FOUNDERPOP <- runMacs(nInd = 100,

nChr = 20,

inbred = FALSE,

species = "GENERIC")

## Simulation parameters

SIMPARAM <- SimParam$new(FOUNDERPOP)

SIMPARAM$addTraitA(nQtlPerChr = 100,

mean = 100,

var = 10)

SIMPARAM$setGender("yes\_sys")

SIMPARAM$setVarE(h2 = 0.3)

## Random mating for 9 more generations

generations <- vector(mode = "list", length = 10)

generations[[1]] <- newPop(FOUNDERPOP,

simParam = SIMPARAM)

for (gen in 2:10) {

generations[[gen]] <- randCross(generations[[gen - 1]],

nCrosses = 10,

nProgeny = 10,

simParam = SIMPARAM)

}

## Put them all together

combined <- Reduce(c, generations)

## Extract phentoypes

pheno <- data.frame(animal = combined@id,

pheno = combined@pheno[,1])

## Extract pedigree

ped <- data.frame(id = combined@id,

dam = combined@mother,

sire =combined@father)

ped$dam[ped$dam == 0] <- NA

ped$sire[ped$sire == 0] <- NA

## Write out the files

write.csv(pheno,

file = "sim\_pheno.csv",

row.names = FALSE,

quote = FALSE)

write.csv(ped,

file = "sim\_ped.csv",

row.names = FALSE,

quote = FALSE)

In turn, we:

1. Set up a founder population with a AlphaSimR’s generic livestock-like population history, and 20 chromosomes.
2. Choose simulation parameters: we have an organism with separate sexes, a quantitative trait with an additive polygenic architecture, and we want an environmental variance to give us a heritability of 0.3.
3. We store away the founders as the first generation, then run a loop to give us nine additional generations of random mating.
4. Combine the resulting generations into one population.
5. Extract phenotypes and pedigree into their own data frames.
6. Optionally, save the latter data frames to files (for the last post).

Now that we have some data, we can fit a quantitative genetic pedigree model (”animal model”) to estimate genetic parameters. We’re going to try two methods to fit it: Markov Chain Monte Carlo and (the unfortunately named) Integrated Nested Laplace Approximation. MCMC explores the posterior distribution by sampling; I’m not sure where I heard it described as ”exploring a mountain by random teleportation”. INLA makes approximations to the posterior that can be integrated numerically; I guess it’s more like building a sculpture of the mountain.

First, a Gaussian animal model in *MCMCglmm*:

library(MCMCglmm)

## Gamma priors for variances

prior\_gamma <- list(R = list(V = 1, nu = 1),

G = list(G1 = list(V = 1, nu = 1)))

## Fit the model

model\_mcmc <- MCMCglmm(scaled ~ 1,

random = ~ animal,

family = "gaussian",

prior = prior\_gamma,

pedigree = ped,

data = pheno,

nitt = 100000,

burnin = 10000,

thin = 10)

## Calculate heritability for heritability from variance components

h2\_mcmc\_object <- model\_mcmc$VCV[, "animal"] /

(model\_mcmc$VCV[, "animal"] + model\_mcmc$VCV[, "units"])

## Summarise results from that posterior

h2\_mcmc <- data.frame(mean = mean(h2\_mcmc\_object),

lower = quantile(h2\_mcmc\_object, 0.025),

upper = quantile(h2\_mcmc\_object, 0.975),

method = "MCMC",

stringsAsFactors = FALSE)

And here is a similar animal model in *AnimalINLA*:

library(AnimalINLA)

## Format pedigree to AnimalINLA's tastes

ped\_inla <- ped

ped\_inla$id <- as.numeric(ped\_inla$id)

ped\_inla$dam <- as.numeric(ped\_inla$dam)

ped\_inla$dam[[is.na](http://is.na)(ped\_inla$dam)] <- 0

ped\_inla$sire <- as.numeric(ped\_inla$sire)

ped\_inla$sire[[is.na](http://is.na)(ped\_inla$sire)] <- 0

## Turn to relationship matrix

A\_inv <- compute.Ainverse(ped\_inla)

## Fit the model

model\_inla <- animal.inla(response = scaled,

genetic = "animal",

Ainverse = A\_inv,

type.data = "gaussian",

data = pheno,

verbose = TRUE)

## Pull out summaries from the model object

summary\_inla <- summary(model\_inla)

## Summarise results

h2\_inla <- data.frame(mean = summary\_inla$summary.hyperparam["Heritability", "mean"],

lower = summary\_inla$summary.hyperparam["Heritability", "0.025quant"],

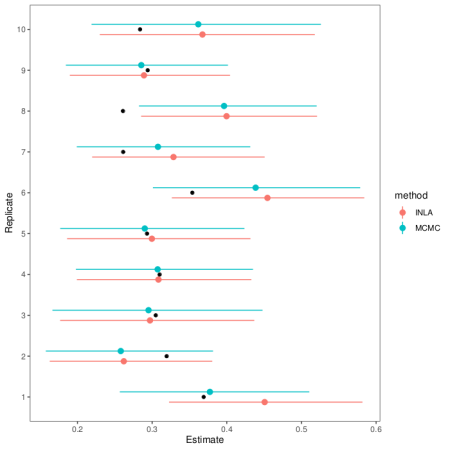
upper = summary\_inla$summary.hyperparam["Heritability", "0.975quant"],

method = "INLA",

stringsAsFactors = FALSE)

If we wrap this all in a loop, we can see how the estimation methods do on replicate data. Here are estimates and intervals from ten replicates (black dots show the actual heritability in the first generation). Here is the full script:

|  |
| --- |
| library(AlphaSimR) |
|  | library(AnimalINLA) |
|  | library(ggplot2) |
|  | library(MCMCglmm) |
|  |  |
|  |  |
|  | results <- vector(mode = "list", |
|  | length = 10) |
|  |  |
|  | resulting\_h2 <- numeric(10) |
|  |  |
|  | for (sim\_ix in 1:10) { |
|  |  |
|  | FOUNDERPOP <- runMacs(nInd = 100, |
|  | nChr = 20, |
|  | inbred = FALSE, |
|  | species = "GENERIC") |
|  |  |
|  |  |
|  | ## Simulation parameters |
|  |  |
|  | SIMPARAM <- SimParam$new(FOUNDERPOP) |
|  | SIMPARAM$addTraitA(nQtlPerChr = 100, |
|  | mean = 100, |
|  | var = 10) |
|  | SIMPARAM$setGender("yes\_sys") |
|  | SIMPARAM$setVarE(h2 = 0.3) |
|  |  |
|  |  |
|  | generations <- vector(mode = "list", length = 10) |
|  | generations[[1]] <- newPop(FOUNDERPOP, |
|  | simParam = SIMPARAM) |
|  |  |
|  |  |
|  | for (gen in 2:10) { |
|  |  |
|  | generations[[gen]] <- randCross(generations[[gen - 1]], |
|  | nCrosses = 10, |
|  | nProgeny = 10, |
|  | simParam = SIMPARAM) |
|  |  |
|  | } |
|  |  |
|  | combined <- Reduce(c, generations) |
|  |  |
|  | resulting\_h2[sim\_ix] <- varG(generations[[1]])/varP(generations[[1]]) |
|  |  |
|  | pheno <- data.frame(animal = combined@id, |
|  | pheno = combined@pheno[,1], |
|  | stringsAsFactors = FALSE) |
|  |  |
|  | ped <- data.frame(id = combined@id, |
|  | dam = combined@mother, |
|  | sire =combined@father, |
|  | stringsAsFactors = FALSE) |
|  | ped$dam[ped$dam == 0] <- NA |
|  | ped$sire[ped$sire == 0] <- NA |
|  |  |
|  | rm(generations) |
|  | rm(combined) |
|  |  |
|  |  |
|  | ##write.csv(pheno, |
|  | ## file = "sim\_pheno.csv", |
|  | ## row.names = FALSE, |
|  | ## quote = FALSE) |
|  |  |
|  | ##write.csv(ped, |
|  | ## file = "sim\_ped.csv", |
|  | ## row.names = FALSE, |
|  | ## quote = FALSE) |
|  |  |
|  |  |
|  |  |
|  | pheno$scaled <- scale(pheno$pheno) |
|  |  |
|  |  |
|  | ##MCMCglmm |
|  |  |
|  | prior\_gamma <- list(R = list(V = 1, nu = 1), |
|  | G = list(G1 = list(V = 1, nu = 1))) |
|  |  |
|  | model\_mcmc <- MCMCglmm(scaled ~ 1, |
|  | random = ~ animal, |
|  | family = "gaussian", |
|  | prior = prior\_gamma, |
|  | pedigree = ped, |
|  | data = pheno, |
|  | nitt = 100000, |
|  | burnin = 10000, |
|  | thin = 10) |
|  |  |
|  | h2\_mcmc\_object <- model\_mcmc$VCV[, "animal"]/(model\_mcmc$VCV[, "animal"] + model\_mcmc$VCV[, "units"]) |
|  |  |
|  | h2\_mcmc <- data.frame(mean = mean(h2\_mcmc\_object), |
|  | lower = quantile(h2\_mcmc\_object, 0.025), |
|  | upper = quantile(h2\_mcmc\_object, 0.975), |
|  | method = "MCMC", |
|  | stringsAsFactors = FALSE) |
|  |  |
|  |  |
|  | rm(model\_mcmc) |
|  |  |
|  |  |
|  | ## INLA |
|  |  |
|  |  |
|  | ped\_inla <- ped |
|  | ped\_inla$id <- as.numeric(ped\_inla$id) |
|  | ped\_inla$dam <- as.numeric(ped\_inla$dam) |
|  | ped\_inla$dam[is.na(ped\_inla$dam)] <- 0 |
|  | ped\_inla$sire <- as.numeric(ped\_inla$sire) |
|  | ped\_inla$sire[is.na(ped\_inla$sire)] <- 0 |
|  |  |
|  |  |
|  | A\_inv <- compute.Ainverse(ped\_inla) |
|  |  |
|  |  |
|  | model\_inla <- animal.inla(response = scaled, |
|  | genetic = "animal", |
|  | Ainverse = A\_inv, |
|  | type.data = "gaussian", |
|  | data = pheno, |
|  | verbose = TRUE) |
|  |  |
|  | summary\_inla <- summary(model\_inla) |
|  |  |
|  | h2\_inla <- data.frame(mean = summary\_inla$summary.hyperparam["Heritability", "mean"], |
|  | lower = summary\_inla$summary.hyperparam["Heritability", "0.025quant"], |
|  | upper = summary\_inla$summary.hyperparam["Heritability", "0.975quant"], |
|  | method = "INLA", |
|  | stringsAsFactors = FALSE) |
|  |  |
|  | rm(model\_inla) |
|  |  |
|  |  |
|  | results[[sim\_ix]] <- rbind(h2\_mcmc, |
|  | h2\_inla) |
|  |  |
|  | results[[sim\_ix]]$rep <- sim\_ix |
|  |  |
|  | } |
|  |  |
|  | results\_combined <- Reduce(rbind, results) |
|  |  |
|  | h2 <- data.frame(rep = 1:10, |
|  | h2 = resulting\_h2) |
|  |  |
|  | plot\_estimates <- ggplot() + |
|  | geom\_pointrange(aes(x = factor(rep), y = mean, ymin = lower, ymax = upper, colour = method), |
|  | data = results\_combined, |
|  | position = position\_dodge(0.5)) + |
|  | geom\_point(aes(x = factor(rep), |
|  | y = h2), |
|  | data = h2) + |
|  | xlab("Replicate") + |
|  | ylab("Estimate") + |
|  | coord\_flip() + |
|  | theme\_bw() + |
|  | theme(panel.grid = element\_blank()) |



As you can see, the MCMC and INLA estimates agree pretty well and mostly hit the mark. In the one replicate dataset where they falter, they falter together.