run-lmmlite

Fred Yu

2022-07-18

Load libraries:

```
# install.packages("devtools")
library(devtools)
```

```
## Loading required package: usethis
```

```
# install.packages("remotes")
# library(remotes)
# install_github("kbroman/lmmlite")

library(lmmlite)

library(knitr)
```

Read in the data:

```
pheno = read.csv("../../data/BXDpheno.csv")
geno = read.csv("../../data/BXDgeno.csv")
K = read.csv("../../data/BXDkinship.csv")
```

```
pheno_1 = pheno[, 1]
pheno_2 = pheno[, 2]
K_mat = data.matrix(K)
geno_mat = data.matrix(geno)
```

Run lmmlite:

Here we will only consider scanning the second quantitative trait for comparing results:

```
# e_null = eigen_rotation(K_mat, pheno_1, NULL)
e_null = eigen_rotation(K_mat, pheno_2, NULL)
```

By default, lmmlite will estimate the variance components parameters using restricted maximum likelihood estimators (REML).

In this notebook, we can run the following code twice - for the first time use line 77 and comment out line 78, save output as a CSV file to get REML result, and for the second time we do the other way around to get ML result and save it.

```r
n = nrow(geno_mat)
p = ncol(geno_mat)


params_null = fitLMM(e_null$Kva, e_null$y, e_null$X, reml = T)
## params_null = fitLMM(e_null$Kva, e_null$y, e_null$X, reml = F)
est_hsq = params_null$hsq
est_sigmasq = params_null$sigmasq
```

```r
## Helper functions:


## Function to construct the design matrix for each marker G_j, j = 1,..., p.
construct_Gj = function(geno, intercept = TRUE){

  list_G = list()
  n = nrow(geno) ## number of individuals
  p = ncol(geno) ## number of markers

  intercept = rep(1, n)

  for(j in 1:p){
    Gj = cbind(intercept, geno[, j])
    list_G[[j]] = Gj
  }

  return(list_G)

}

K_eVects = e_null$Kve_t # left eigen-vectors of kinship matrix
K_eVals = e_null$Kva # eigen-values of kinship matrix

run_model = function(K_eVals, K_eVects, Gj, y, hsq){

  # Rotate data:
  y_star1 = K_eVects %*% y # may not need to do everytime; needs refinement
  Gj_star1 = K_eVects %*% Gj

  # Get RSS:
  ml_soln = getMLsoln(hsq, K_eVals, y_star1, Gj_star1, reml = T)
  # ml_soln = getMLsoln(hsq, K_eVals, y_star1, Gj_star1, reml = F)



  return(ml_soln)

}

rss2Lod = function(rss_null, rss_mod, n){

  lod = (n/2)*(log10(rss_null) - log10(rss_mod))

  return(lod)
}
```

Essentially, what the lmmlite program does is for a given h^2, it estimates the σ_e^2 and the fixed effects of based on optimizing the likelihood function given y and each marker G_j, $j = 1, ..., p$. So, one of the variance component σ_e^2 will be re-estimated for every marker.

We use lmmlite by first estimate the heritability $\hat{h}_0^2$ for the null model that includes only the intercept, and use it for the estimations of other parameters (fixed effects, variance due to noises) for every marker:

```r
# Run model for each marker:

list_Gj = construct_Gj(geno)


### Manually give the hsq estimated from BulkLMM flmm to getMLsoln
## est_hsq =
###

results_null = getMLsoln(est_hsq, e_null$Kva, e_null$y, e_null$X, reml = T)
# results_null = getMLsoln(est_hsq, e_null$Kva, e_null$y, e_null$X, reml = F)


list_RSS = rep(NA, p+1)
rss_null = attributes(results_null)$rss
list_RSS[1] = rss_null

list_ml_solns = list(results_null)

for(j in 1:p){

# ml_soln = run_model(K_eVals, K_eVects, list_Gj[[j]], pheno_1, est_hsq)
  ml_soln = run_model(K_eVals, K_eVects, list_Gj[[j]], pheno_2, est_hsq)
  rss = attributes(ml_soln)$rss

  list_ml_solns[[j+1]] = ml_soln
  list_RSS[j+1] = rss

}
```

After we got the estimated fixed effects for each marker, we can then compute the RSSs and finally compute the LOD scores:

```r
list_LOD = sapply(list_RSS[2:length(list_RSS)],
       function(x) rss2Lod(list_RSS[1], x, n))
list_LOD = c(NA, list_LOD)

list_Sigma_e = rep(NA, length(list_ml_solns))
list_Beta_0 = rep(NA, length(list_ml_solns))
list_Beta_1 = rep(NA, length(list_ml_solns))

for(j in 1:length(list_ml_solns)){

  list_Beta_0[j] = list_ml_solns[[j]][[1]][1]
  list_Beta_1[j] = list_ml_solns[[j]][[1]][2]
  list_Sigma_e[j] = list_ml_solns[[j]][2]

}

list_Beta_0 = unlist(list_Beta_0)
list_Beta_1 = unlist(list_Beta_1)
list_Sigma_e = unlist(list_Sigma_e)

head(list_LOD)
```

```
## [1]         NA 0.04594157 0.04594157 0.04594157 0.04594157 0.04594157
```

```
tail(list_LOD)
```

```
## [1] 0.04626362 0.09097675 0.08559189 0.14504912 0.25375088 0.25375088
```

```
head(list_Sigma_e)
```

```
## [1] 0.05662303 0.05720499 0.05720499 0.05720499 0.05720499 0.05720499
```

```
tail(list_Sigma_e)
```

```
## [1] 0.05720391 0.05705501 0.05707292 0.05687545 0.05651619 0.05651619
```

Saved results is a CSV table that has fields of the estimated fixed effects (intercept + marker effect), estimated σ_e^2 and the LOD score for each marker.

```
results_lmmlite = data.frame(cbind(list_Beta_0, list_Beta_1,
                                   list_Sigma_e, list_LOD))
colnames(results_lmmlite) = c("Est_Beta_0", "Est_Beta_1",
                              "Est_Sigma_e", "LOD")
rownames = c("Null")

for(j in 1:p){
  rownames = c(rownames, paste("G_", j))
}

rownames(results_lmmlite) = rownames
kable(head(results_lmmlite))
```

	Est_Beta_0	Est_Beta_1	Est_Sigma_e	LOD
Null	8.937538	NA	0.056623	NA
G_ 1	8.923832	0.0245998	0.057205	0.0459416
G_ 2	8.923832	0.0245998	0.057205	0.0459416
G_ 3	8.923832	0.0245998	0.057205	0.0459416
G_ 4	8.923832	0.0245998	0.057205	0.0459416
G_ 5	8.923832	0.0245998	0.057205	0.0459416

```
# write.csv(results_lmmlite, "../output/result.lmmlite_REML.csv")
# write.csv(results_lmmlite, "../output/result.lmmlite_ML.csv")
```