curatedMetagenomicData Data Application - Scenario 1

Load packages.

```
library(curatedMetagenomicData)
library(plyr)
library(FSelector)
library(glmnet)
library(data.table)
library(nlme)
library(lme4)
```

Load data.

```
metadata = curatedMetagenomicData("QinJ_2012.marker_abundance.stool", dryrun = FALSE)
meta.merged = mergeData(metadata)
rm(list=c("metadata"))
```

Add clinical variables to marker abundance data and restrict to female patients.

```
meta.exp = data.frame(t(exprs(meta.merged)))
meta.exp$cholesterol = meta.merged$cholesterol
meta.exp$age = meta.merged$age
meta.exp = meta.exp[which(meta.merged$gender=="female"),]
meta.exp = meta.exp[complete.cases(meta.exp),]
```

Divide dataset into training and test sets.

```
set.seed(123)
# number of training datasets
ndat = 4
ind.train = sample(1:nrow(meta.exp), ceiling(nrow(meta.exp)*(ndat)/(ndat+1)))
qin.train = meta.exp[ind.train,]
qin.test = meta.exp[-ind.train,]
qin.train$group = sample(1:ndat, nrow(qin.train), replace=T)
qin.test$group = ndat+1
group = qin.train$group
```

Use the top 5 marker abundances most highly correlated with the outcome in the training set as the predictors.

```
qin.train = qin.train[, which(colSums(qin.train)!=0)]

# remove features that are very sparse in the training data
min.samples = 4
```

```
qin.train = qin.train[vapply(qin.train,
                             function(x) length(unique(x))>min.samples, logical(1L))]

# calculate correlation for each feature
feature.list = lapply(split(as.list(as.data.frame(qin.train[, which(!names(qin.train) %in%
                                                                      c("cholesterol", "age"))])),
                            cut(1:ncol(qin.train[, which(!names(qin.train) %in%
                                                           c("cholesterol", "age"))]), 20)),
                      as.data.frame)
weight.list = lapply(feature.list, function(x) linear.correlation(qin.train$cholesterol ~., x))
weight.df = rbind.fill(weight.list)
weight.df$feature = unlist(lapply(weight.list, row.names))
weight.df = weight.df[order(weight.df$attr_importance, decreasing = T), ]
# get top 5 features
max.features = 5
qin.train = cbind(qin.train[, which(names(qin.train) %in% c(weight.df$feature[1:max.features],
                                                             c("cholesterol", "age")))], group)
qin.test = qin.test[, which(names(qin.test) %in% c(weight.df$feature[1:max.features],
                                                    c("cholesterol", "age", "group")))]
```

Set up design matrices.

```
qin.all = rbind(qin.train, qin.test)
parts = split(qin.all, qin.all$group)
edat_train = parts[1:ndat]
edat_test = parts[ndat+1]
train = rbindlist(edat_train)
test = rbindlist(edat_test)
```

Estimate random effect variances and variance of residuals using REML via a linear mixed effects model.

```
features = names(train)[which(!names(train) %in% c("cholesterol", "group"))]
lm.formula = as.formula(paste("cholesterol~", paste0(features, collapse="+")))
feature.cols = which(names(train) %in% c(features))

lmer.formula = as.formula(paste("cholesterol~ (1|group) +",
                                paste0(unique(names(train)[feature.cols]), collapse="+"), "+",
                                paste0("(0+", names(train)[feature.cols], "|group)", collapse="+")))
tol = 1e-10
fit.lmer = lmer(lmer.formula, data=train)
ind.re = which(as.data.frame(VarCorr(fit.lmer))[2:(length(feature.cols)+1), 4]>tol)
sigma.eps = summary(fit.lmer)$sigma
as.data.frame(VarCorr(fit.lmer))[1:(length(feature.cols)+1), 4]

## [1] 3.292003e-05 5.073568e-05 0.000000e+00 0.000000e+00 6.950433e-05
## [6] 0.000000e+00 0.000000e+00

sigma2.bar = mean(as.data.frame(VarCorr(fit.lmer))[1:(length(feature.cols)+1), 4])
sigma2.bar

## [1] 2.188001e-05
```

Estimate optimal LS weights.

```
source("transition_point_fns.R")
parts2 = split(cbind(rep(1, nrow(qin.all)), qin.all[, feature.cols]),
               qin.all$group)
edat_train2 = parts2[1:ndat]
edat_test2 = parts2[ndat+1]

# estimate optimal LS weights
vec.re = sqrt(as.data.frame(VarCorr(fit.lmer))[1:(length(feature.cols)+1), 4])
wk.ols = optimal_weights(edat_train2, edat_test2, sigma.eps, vec.re)
wk.ols

## [1] 0.454403415 0.304337930 0.231621326 0.009637329
```

Tune ridge regression regularization parameters.

```
# choose regularization parameter
set.seed(1)
cv.ridge.merged = cv.glmnet(data.matrix(train)[, feature.cols], train$cholesterol,
                            alpha = 0,
                            intercept=T, lambda=2^seq(-8, 8, length=100), standardize=F)
sd.y = sqrt(var(train$cholesterol)*(length(train$cholesterol)-1)/length(train$cholesterol))
# compute regularization parameter for formulation of ridge regression objective function assumed by
lam = cv.ridge.merged$lambda.min*nrow(train)/sd.y

lamk = rep(NA, ndat)

for (i in 1:ndat) {
  dataset = edat_train[[i]]

  set.seed(1)
  cv.ridge = cv.glmnet(data.matrix(dataset)[, feature.cols], dataset$cholesterol,
                       alpha = 0,
                       intercept=T, lambda=2^seq(-8, 8, length=100),
                       standardize=F, nfolds=5)
  sd.y = sqrt(var(dataset$cholesterol)*(length(dataset$cholesterol)-1)/length(dataset$cholesterol))
  lamk[i] = cv.ridge$lambda.min*nrow(dataset)/sd.y
}
```

Train and validate merged LS model.

```
fit.ols.merged = lm(lm.formula, data=train)
pred.ols.merged = predict(fit.ols.merged, newdata=test)
err.ols.merged = mean((pred.ols.merged-test$cholesterol)^2)
sqrt(err.ols.merged)

## [1] 53.62371
```

Train and validate merged ridge regression model.

```
fit.ridge.merged = glmnet(data.matrix(train)[, feature.cols], train$cholesterol,
                          alpha = 0,
                          lambda = cv.ridge.merged$lambda.min,
                          intercept=T, standardize=F)
pred.ridge.merged = predict(fit.ridge.merged, newx=data.matrix(test)[, feature.cols])
err.ridge.merged = mean((pred.ridge.merged - test$cholesterol)^2)
sqrt(err.ridge.merged)

## [1] 59.07932
```

Calculate transition intervals using optimal LS weights for the CSLs.

```
if (as.data.frame(VarCorr(fit.lmer))[1, 4] > tol) {
  ind.re = c(0, ind.re)
}

clist = as.list(ind.re+1)

wk.eq = rep(1, ndat)/ndat

ls.bounds = tau_ls_range(edat_train2, edat_test2, wk.eq, sigma.eps, cols_re_list=clist)
ls.bounds

## [1] 7.089301e-02 1.373470e+04

ridge.bounds = tau_r_range(edat_train2, edat_test2, wk.eq, sigma.eps, lambda=lam, lambdak=lamk,
                           beta=fit.ols.merged$coefficients, cols_re_list=clist)
ridge.bounds

## [1] 2.449549e-03 6.241744e+02

sigma2.bar < ls.bounds[1]

## [1] TRUE

sigma2.bar < ridge.bounds[1]

## [1] TRUE
```

Train and validate CSLs.

```
beta.ols.mat = matrix(data=NA, nrow=ndat, ncol=length(fit.ols.merged$coefficients))
beta.ols.mat.opt = beta.ols.mat

beta.ridge.mat = beta.ols.mat
ols.mat = matrix(data=NA, nrow=ndat, ncol=nrow(test))
ridge.mat = ols.mat

# fit models to each study
for (i in 1:ndat) {

  dataset = edat_train[[i]]
```

```
  # OLS
  fit.ols = lm(lm.formula, data=dataset)
  beta.ols.mat[i, ] = fit.ols$coefficients
  ols.mat[i, ] = predict(fit.ols, newdata=test)

  # ridge
  sd.y = sqrt(var(dataset$cholesterol)*(length(dataset$cholesterol)-1)/length(dataset$cholesterol))
  fit.ridge = glmnet(data.matrix(dataset)[, feature.cols], dataset$cholesterol,
                     alpha = 0,
                     lambda = lamk[i]*sd.y/nrow(dataset),
                     intercept=T, standardize=F)
  beta.ridge.mat[i, ] = c(fit.ridge$a0, as.vector(fit.ridge$beta))
  ridge.mat[i, ] = predict(fit.ridge, newx=data.matrix(test)[, feature.cols])
}

pred.ols.csl.eq = wk.eq %*% ols.mat
err.ols.csl.eq = mean((pred.ols.csl.eq-test$cholesterol)^2)
sqrt(err.ols.csl.eq)

## [1] 202.1047

pred.ols.csl = wk.ols %*% ols.mat
err.ols.csl = mean((pred.ols.csl-test$cholesterol)^2)
sqrt(err.ols.csl)

## [1] 85.69544

pred.ridge.csl.eq = wk.eq %*% ridge.mat
err.ridge.csl.eq = mean((pred.ridge.csl.eq-test$cholesterol)^2)
sqrt(err.ridge.csl.eq)

## [1] 69.92978

wk.ridge = optimal_weights_ridge(edat_train2, edat_test2, sigma.eps, vec.re, lamk, summary(fit.lmer)$
pred.ridge.csl = wk.ridge %*% ridge.mat
err.ridge.csl = mean((pred.ridge.csl-test$cholesterol)^2)
sqrt(err.ridge.csl)

## [1] 67.81052
```

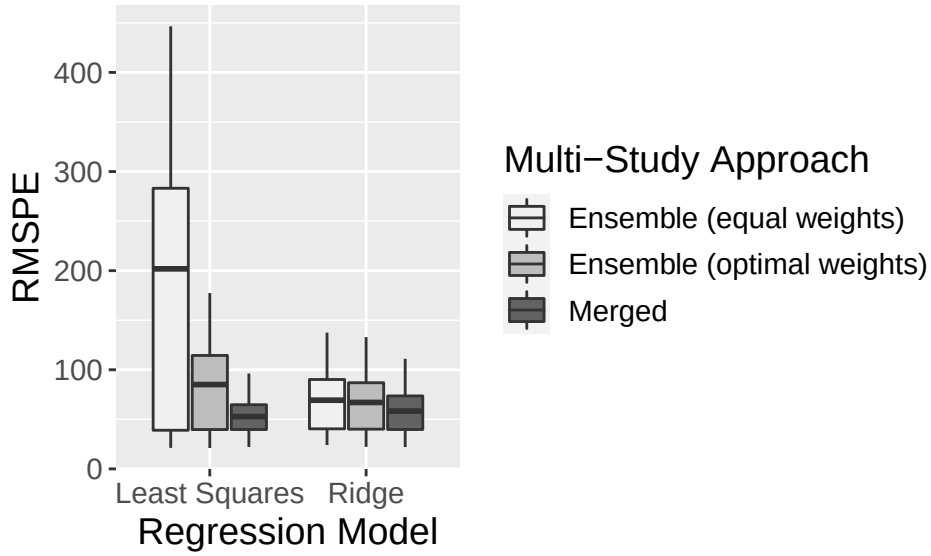
Get bootstrap confidence intervals for prediction error.

```
nboot = 500

set.seed(1)
err.ridge = data.frame(merged=rep(NA, nboot), csl=NA, csl.opt=NA)
err.ols = data.frame(merged=rep(NA, nboot), csl=NA, csl.opt=NA)
for (i in 1:nboot) {
  ind.boot = sample(1:length(pred.ridge.merged), length(pred.ridge.merged), replace=T)
  edat_test2.boot = edat_test2
  edat_test2.boot[[1]] = edat_test2.boot[[1]][ind.boot,]
  wk.ols.boot = optimal_weights(edat_train2, edat_test2.boot, sigma.eps, vec.re)
  wk.ridge.boot = optimal_weights_ridge(edat_train2, edat_test2.boot, sigma.eps, vec.re, lamk, summar
```

```
  err.ridge$merged[i] = mean((pred.ridge.merged[ind.boot] - test$cholesterol[ind.boot])^2)
  err.ridge$csl[i] = mean((pred.ridge.csl.eq[ind.boot] - test$cholesterol[ind.boot])^2)
  err.ridge$csl.opt[i] = mean((wk.ridge.boot %*% ridge.mat[, ind.boot] - test$cholesterol[ind.boot])^
  err.ols$merged[i] = mean((pred.ols.merged[ind.boot] - test$cholesterol[ind.boot])^2)
  err.ols$csl[i] = mean((pred.ols.csl.eq[ind.boot] - test$cholesterol[ind.boot])^2)
  err.ols$csl.opt[i] = mean((wk.ols.boot %*% ols.mat[, ind.boot] - test$cholesterol[ind.boot])^2)
}
save(err.ridge, err.ols, file="cholesterol_single.RData")
```

Make boxplots of prediction error.

```
library(ggplot2)
library(reshape2)

err.ridge = sqrt(err.ridge)
err.ols = sqrt(err.ols)
names(err.ridge) = c("R,M", "R,E", "R,E (O)")
names(err.ols) = c("LS,M", "LS,E", "LS,E (O)")
err2 = melt(cbind(err.ols, err.ridge))
names(err2) = c("Learner", "RMSPE")
err2$model = "Least Squares"
err2$model[which(err2$Learner %in% c("R,M", "R,E", "R,E (O)"))] = "Ridge"
err2$type = "Ensemble (equal weights)"
err2$type[which(err2$Learner %in% c("LS,M", "R,M"))] = "Merged"
err2$type[which(err2$Learner %in% c("LS,E (O)", "R,E (O)"))] = "Ensemble (optimal weights)"
names(err2) = c("Learner", "RMSPE", "Regression Model", "Multi-Study Approach")
ggplot(err2, aes(`Regression Model`, RMSPE, fill=`Multi-Study Approach`)) + geom_boxplot() + theme(te
```




```
# save figure
library(gridExtra)
png('rmse_scenario1.png', width=800, height=500, res=100)
ggplot(err2, aes(`Regression Model`, RMSPE, fill=`Multi-Study Approach`)) + geom_boxplot() + theme(te
dev.off()
```

```
## pdf
##   2
```