curatedMetagenomicData Data Application - Scenario 1

Load packages.

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

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("../simulations/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_weights0(edat_train2, edat_test2, sigma.eps, vec.re)
wk.ols

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

Tune regularization parameters.

```
# choose regularization parameters
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 transition_point_fns.R
# (https://stats.stackexchange.com/questions/129179/why-is-glmnet-ridge-regression-giving-me-a-differ
lam = cv.ridge.merged$lambda.min * nrow(train)/sd.y

set.seed(1)
cv.lasso.merged = cv.glmnet(data.matrix(train)[, feature.cols], train$cholesterol,
    alpha = 1, intercept = T, lambda = 2^seq(-8, 8, length = 100), standardize = F)
lam.lasso = cv.lasso.merged$lambda.min

lamk.lasso = rep(NA, ndat)

lamk = rep(NA, ndat)

mtryk = rep(NA, ndat)
sizek = rep(NA, ndat)
decayk = rep(NA, ndat)

# cross-validation setup for random forest and nn
control = trainControl(method = "repeatedcv", number = 5, repeats = 2)

nvar = length(features)
tunegrid = expand.grid(mtry = seq(floor(nvar/3), nvar))
tunegrid.nn = expand.grid(layer1 = c(2, 5, 10), layer2 = 0, layer3 = 0)
```

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

    set.seed(1)

    # ridge
    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$cholesterol) - 1)/length(dataset$cholester
    lamk[i] = cv.ridge$lambda.min * nrow(dataset)/sd.y

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

    # random forest
    rf_default <- train(lm.formula, data = dataset, method = "rf", tuneGrid = tunegrid,
        trControl = control)
    mtryk[i] = unlist(rf_default$bestTune)

    # nn
    dataset$group = NULL
    # normalize
    pp = preProcess(dataset, method = "range")
    train.norm = predict(pp, dataset)
    test.norm = predict(pp, test)
    nn_default <- train(lm.formula, data = train.norm, method = "neuralnet",
        tuneGrid = tunegrid.nn, trControl = control)
    sizek[i] = unlist(nn_default$bestTune)[1]
}


set.seed(123)
rf_default <- train(lm.formula, data = train, method = "rf", tuneGrid = tunegrid,
    trControl = control)
mtry = unlist(rf_default$bestTune)


# normalize
pp = preProcess(train, method = "range")
train.norm = predict(pp, train)
test.norm = predict(pp, test)

set.seed(123)
nn_default <- train(lm.formula, data = train.norm, method = "neuralnet", tuneGrid = tunegrid.nn,
    trControl = control)
size = unlist(nn_default$bestTune)[1]
```

Train and validate merged models.

```
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

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

fit.lasso.merged = glmnet(data.matrix(train)[, feature.cols], train$cholesterol,
    alpha = 1, lambda = cv.lasso.merged$lambda.min, intercept = T, standardize = F)
pred.lasso.merged = predict(fit.lasso.merged, newx = data.matrix(test)[, feature.cols])
err.lasso.merged = mean((pred.lasso.merged - test$cholesterol)^2)
sqrt(err.lasso.merged)

## [1] 63.14949

# merged nn
set.seed(10)
nn = neuralnet(lm.formula, data = train.norm, hidden = size)
pred.nn.merged.norm = predict(nn, newdata = test.norm)
pred.nn.merged = pred.nn.merged.norm * (pp$ranges[, "cholesterol"][2] - pp$ranges[,
    "cholesterol"][1]) + pp$ranges[, "cholesterol"][1]
err.nn.merged = mean((pred.nn.merged - test$cholesterol)^2)
sqrt(err.nn.merged)

## [1] 39.67863

# merged random forest
set.seed(10)
fit.rf = randomForest(lm.formula, data = train, mtry = mtry)
pred.rf.merged = predict(fit.rf, newdata = test)
err.rf.merged = mean((pred.rf.merged - test$cholesterol)^2)
sqrt(err.rf.merged)

## [1] 48.97911
```

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.ridge.mat = beta.ols.mat
beta.lasso.mat = beta.ols.mat

ols.mat = matrix(data = NA, nrow = ndat, ncol = nrow(test))
ridge.mat = ols.mat
lasso.mat = ols.mat
nn.mat = ols.mat
rf.mat = ols.mat

set.seed(1)
# 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$cholester
    fit.ridge = glmnet(data.matrix(dataset)[, feature.cols], dataset$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])

    # lasso
    fit.lasso = glmnet(data.matrix(dataset)[, feature.cols], dataset$cholesterol,
```

```
        alpha = 1, lambda = lamk.lasso[i], intercept = T, standardize = F)
    beta.lasso.mat[i, ] = c(fit.lasso$a0, as.vector(fit.lasso$beta))
    lasso.mat[i, ] = predict(fit.lasso, newx = data.matrix(test)[, feature.cols])

    ## nn
    dataset$group = NULL
    pp = preProcess(dataset, method = "range")
    train.norm = predict(pp, dataset)
    test.norm = predict(pp, test)

    nn = neuralnet(lm.formula, data = train.norm, hidden = sizek[i])
    pred.nn.merged.norm = predict(nn, newdata = test.norm)
    pred.nn.merged = pred.nn.merged.norm * (pp$ranges[, "cholesterol"][2] -
        pp$ranges[, "cholesterol"][1]) + pp$ranges[, "cholesterol"][1]
    nn.mat[i, ] = pred.nn.merged

    ## random forest
    fit.rf = randomForest(lm.formula, data = dataset, mtry = mtryk[i])
    pred.rf.merged = predict(fit.rf, newdata = test)
    rf.mat[i, ] = pred.rf.merged

}

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

## [1] 85.69544

wk.ridge = optimal_weights_ridge0(edat_train2, edat_test2, sigma.eps, vec.re,
    lamk, summary(fit.lmer)$coefficients[, 1])
pred.ridge.csl = wk.ridge %*% ridge.mat
err.ridge.csl = mean((pred.ridge.csl - test$cholesterol)^2)
sqrt(err.ridge.csl)

## [1] 67.81052

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

## [1] 60.5789

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

## [1] 49.24639

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

## [1] 48.35378
```

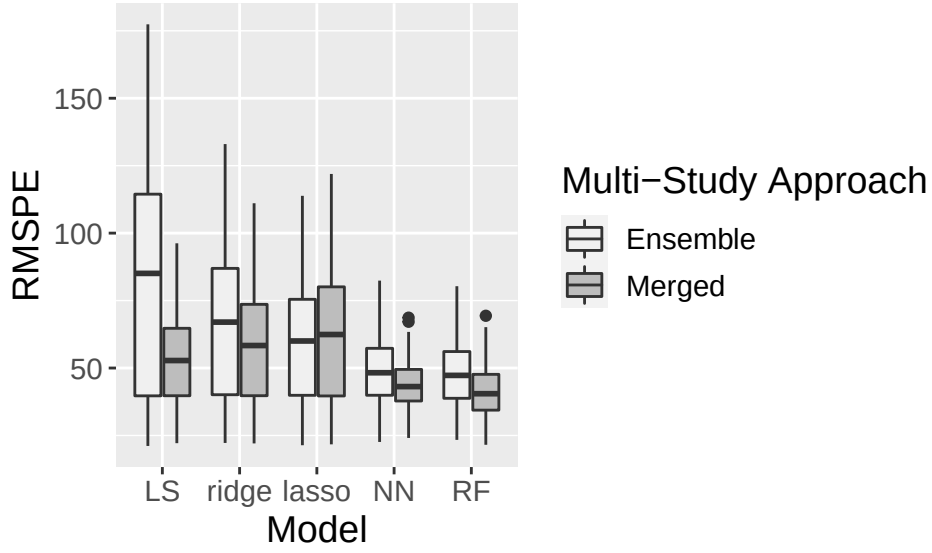
Get bootstrap confidence intervals for prediction error.

```
nboot = 500

set.seed(1)
err.df = data.frame(merged.ols = rep(NA, nboot), csl.ols = NA)
err.df[, paste0("merged.", c("ridge", "lasso", "nn", "rf"))] = NA
err.df[, paste0("csl.", c("ridge", "lasso", "nn", "rf"))] = 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_weights0(edat_train2, edat_test2.boot, sigma.eps,
        vec.re)
    wk.ridge.boot = optimal_weights_ridge0(edat_train2, edat_test2.boot, sigma.eps,
        vec.re, lamk, summary(fit.lmer)$coefficients[, 1])

    err.df$merged.ols[i] = mean((pred.ols.merged[ind.boot] - test$cholesterol[ind.boot])^2)
    err.df$csl.ols[i] = mean((wk.ols.boot %*% ols.mat[, ind.boot] - test$cholesterol[ind.boot])^2)

    err.df$merged.ridge[i] = mean((pred.ridge.merged[ind.boot] - test$cholesterol[ind.boot])^2)
    err.df$csl.ridge[i] = mean((wk.ridge.boot %*% ridge.mat[, ind.boot] - test$cholesterol[ind.boot])

    err.df$merged.lasso[i] = mean((pred.lasso.merged[ind.boot] - test$cholesterol[ind.boot])^2)
    err.df$csl.lasso[i] = mean((wk.ridge.boot %*% lasso.mat[, ind.boot] - test$cholesterol[ind.boot])

    err.df$merged.nn[i] = mean((pred.nn.merged[ind.boot] - test$cholesterol[ind.boot])^2)
    err.df$csl.nn[i] = mean((wk.ridge.boot %*% nn.mat[, ind.boot] - test$cholesterol[ind.boot])^2)

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

Make boxplots of prediction error.

```
library(ggplot2)
library(reshape2)
library(data.table)

err2 = reshape2::melt(sqrt(err.df))
names(err2) = c("Learner", "RMSPE")
err2$Learner = as.character(err2$Learner)

err2$model = NA
err2$model = gsub("merged.", "", err2$Learner)
err2$model = gsub("csl.", "", err2$model)

err2$type = "Ensemble"
```

```r
err2$type[which(grepl("merged", err2$Learner))] = "Merged"

names(err2) = c("Learner", "RMSPE", "Model", "Multi-Study Approach")
err2$Model[which(err2$Model %in% "ols")] = "LS"
err2$Model[which(err2$Model %in% "nn")] = "NN"
err2$Model[which(err2$Model %in% "rf")] = "RF"
err2$Model = factor(err2$Model, levels = c("LS", "ridge", "lasso", "NN", "RF"))
ggplot(err2, aes(Model, RMSPE, fill = `Multi-Study Approach`)) + geom_boxplot() +
    theme(text = element_text(size = 14)) + scale_fill_brewer(palette = "Greys")
```




```r
# save figure
library(gridExtra)
png("rmse_scenario1_otherlearners.png", width = 800, height = 300, res = 100)
ggplot(err2, aes(Model, RMSPE, fill = `Multi-Study Approach`)) + geom_boxplot() +
    theme(text = element_text(size = 14)) + scale_fill_brewer(palette = "Greys")
dev.off()

## pdf
##   2
```