

curatedMetagenomicData Data Application - Scenario 2

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(paste0(c("KarlssonFH_2013.", "Heitz-BuschartA_2016.",
      "QinJ_2012."), "marker_abundance.stool"), dryrun = FALSE)
meta.merged = mergeData(metadata)
meta.exp = data.frame(t(exprs(meta.merged)))
rm(list = c("metadata"))
```

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

```
meta.exp = meta.exp[, which(colSums(meta.exp) != 0)]
# remove features that are very sparse in the training data
min.samples = floor(length(which(meta.merged$studyID == "KarlssonFH_2013.marker_abundance.stool" &
  meta.merged$gender == "female"))/10)
meta.exp = meta.exp[vapply(meta.exp, function(x) length(unique(x[which(meta.merged$studyID ==
  "KarlssonFH_2013.marker_abundance.stool" & meta.merged$gender == "female")]))) >
  min.samples & length(unique(x[which(meta.merged$studyID == "QinJ_2012.marker_abundance.stool" &
  meta.merged$gender == "female")]))) > min.samples, logical(1L))]
meta.exp$cholesterol = meta.merged$cholesterol
meta.exp$age = meta.merged$age
meta.exp$group = as.numeric(as.factor(meta.merged$studyID))
meta.exp = meta.exp[which(meta.merged$gender == "female"), ]
meta.exp = meta.exp[complete.cases(meta.exp), ]
```

Divide dataset into training and test sets.

```
meta.train = meta.exp[which(meta.exp$group %in% 2:3), ]
meta.test = meta.exp[which(meta.exp$group %in% 1), ]
```

Use the top 20 marker abundances most highly correlated with the outcome in the training set as the predictors.

```

feature.list = lapply(split(as.list(as.data.frame(meta.train[, which(!names(meta.train) %in%
  c("cholesterol", "age", "group"))])), cut(1:ncol(meta.train[, which(!names(meta.train) %in%
  c("cholesterol", "age", "group"))]), 20)), as.data.frame)
weight.list = lapply(feature.list, function(x) linear.correlation(meta.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$attr_importance, decreasing = T), ]
max.features = 20
meta.train = meta.train[, which(names(meta.train) %in% c(weight.df$feature[1:max.features],
  c("cholesterol", "age", "group")))]
meta.test = meta.test[, which(names(meta.test) %in% c(weight.df$feature[1:max.features],
  c("cholesterol", "age", "group")))]

```

Set up design matrices.

```

ndat = 2
meta.all = rbind(meta.train, meta.test)
parts = split(meta.all, meta.all$group)
edat_train = parts[2:3]
edat_test = parts[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] 2.970851e-03 7.385479e+03 0.000000e+00 0.000000e+00 7.558953e-04
## [6] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
## [11] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 7.009743e-02
## [16] 0.000000e+00 8.634381e-04 0.000000e+00 0.000000e+00 4.019086e-04
## [21] 0.000000e+00 0.000000e+00

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

## [1] 335.707

```

Estimate optimal LS weights.

```
source("../simulations/transition_point_fns.R")
parts2 = split(cbind(rep(1, nrow(meta.all)), meta.all[, feature.cols]), meta.all$group)
edat_train2 = parts2[2:3]
edat_test2 = parts2[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] 0.4999645 0.5000355
```

Tune 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 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 = 1)

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

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

# lasso
cv.lasso = cv.glmnet(data.matrix(dataset)[, feature.cols], dataset$cholesterol,
  alpha = 1, intercept = T, lambda = 2^seq(-8, 8, length = 100), standardize = F)
sd.y = sqrt(var(dataset$cholesterol) * (length(dataset$cholesterol) - 1)/length(dataset$cholesterol))
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$group = NULL
pp = preProcess(dataset, method = "range")
train.norm = predict(pp, dataset)
test.norm = predict(pp, test)
nn_default <- train(lm.formula, data = train.norm, method = "neuralnet",
  tuneGrid = tuneGrid.nn, trControl = control)
sizek[i] = unlist(nn_default$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] 38.65778

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] 38.54562

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] 42.77903

# 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] 43.66871

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] 45.80401

```

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] 0.01787156 232.48502656

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] 0.001405357 19.564912992

```

```
sigma2.bar > ls.bounds[2]

## [1] TRUE

sigma2.bar > ridge.bounds[2]

## [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(10)
# 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])

  # 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] 34.35825

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] 35.21786

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

## [1] 50.07753

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

## [1] 28.95443

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

## [1] 49.33207

```

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])^2)

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])^2)

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_multi_otherlearners.RData")

```

Make boxplots of prediction error.

```

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

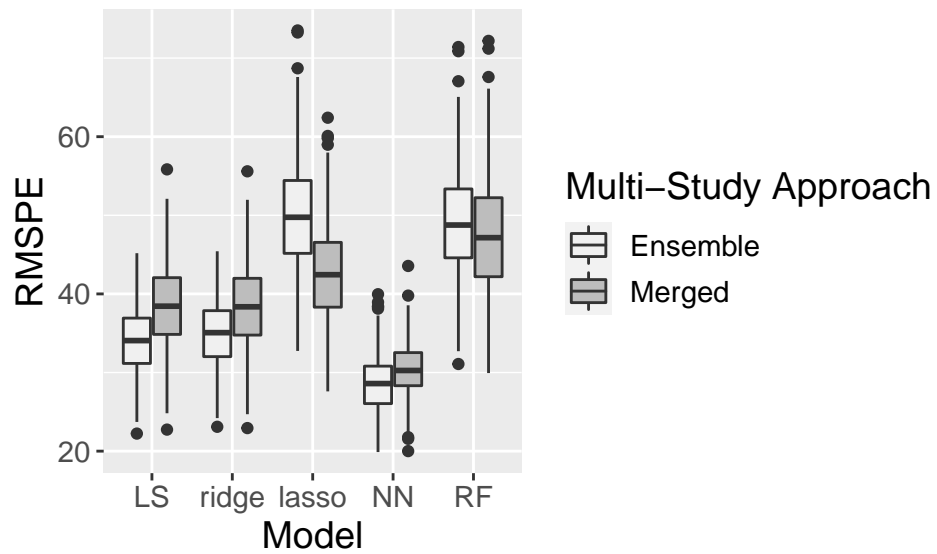
err2 = reshape2::melt(sqrt(err.df))
names(err2) = c("Learner", "RMSPE")
err2$Learner = as.character(err2$Learner)

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

err2$type = "Ensemble"
err2$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")

```

```
# save figure
library(gridExtra)
png("rmse_scenario2_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
```