

```

1  library(dplyr)
2  library(MASS)
3  library(glmnet)
4  library(randomForest)
5  library(e1071)
6  library(ggplot2)
7  library(reshape)
8  library(gridExtra)
9
10 options(expressions = 5e5)
11
12 a = read.csv("pdata.csv")
13 dim(a)
14 colnames(a)[dim(a)[2]] = c("Class")
15 colMeans(a)
16 t.data = a
17
18 # standardize the dataset
19 t.data[, -dim(t.data)[2]] = scale(t.data[, -dim(t.data)[2]])
20 colMeans(t.data)
21 apply(t.data, 2, sd)
22 mean(t.data$Class)
23 attach(t.data)
24
25 # data structure and rates
26 n1 = dim(t.data %>% filter(Class == 1))[1]
27 n0 = dim(t.data %>% filter(Class == 0))[1]
28 n = n1 + n0
29 p = dim(t.data)[2]-1
30
31 # Modelling factors
32 iterations = 100
33 Dlearn_rate = 0.5
34 sampling.rate = 1
35 weight = c("0" = 1/n0, "1" = 1/n1)
36
37 # train and test error rate matrix
38 train_error = matrix(0, nrow = iterations, ncol = 7)
39 colnames(train_error) = c("RF", "R-SVM", "Log", "Log-LASSO", "Log-Ridge", "OOB", "OOBsd")
40
41 cv_error = matrix(0, nrow = iterations, ncol = 3)
42 colnames(cv_error) = c("R-SVM", "Log-LASSO", "Log-Ridge")
43

```

```

44 test_error = matrix(0, nrow = iterations, ncol = 5)
45 colnames(test_error) = c("RF", "R-SVM", "Log", "Log-LASSO", "Log-Ridge")
46
47 lasso.coef = matrix(0, ncol = iterations, nrow = p+1)
48 ridge.coef = matrix(0, ncol = iterations, nrow = p+1)
49
50 # convert to data frame
51 train_error = data.frame(train_error)
52 test_error = data.frame(test_error)
53
54 # time of cv and fit
55 time.cv = matrix(0, nrow = iterations, ncol = 3)
56 colnames(time.cv) = c("SVM", "LASSO", "Ridge")
57
58 time.fit = matrix(0, nrow = iterations, ncol = 4)
59 colnames(time.fit) = c("SVM", "LASSO", "Ridge", "RF")
60
61 # rf importance
62 rf.importace = matrix(0, nrow = iterations, ncol = 400)
63
64 # sampling from t.data
65 sampling = sample(n, n*sampling.rate)
66 sampling.data = data.frame(t.data[sampling,])
67 sampling.n = dim(sampling.data)[1]
68
69 # preparation for lasso and ridge
70 X = model.matrix(Class ~., sampling.data)[,-1]
71 y = sampling.data$Class
72
73
74 # 100 iteration for error rates, time, and coefficients
75 for(m in 1:iterations){
76
77
78     # create a training data vector for dividing the data set.
79     train = sample(sampling.n, sampling.n*Dlearn_rate)
80
81     dat    = data.frame(sampling.data[train,])
82     datt   = data.frame(sampling.data[-train,])
83
84     # svm
85     # record svm cv time
86     ptm = proc.time()
87     tune.svm = tune(svm, as.factor(Class)~., data=dat,
88                     ranges = list(cost = 10^seq(-2,2,length.out = 5),
89                                     gamma = 10^seq(-2,2,length.out = 5)), scale = F)

```

```

90   ptm = proc.time() - ptm
91   time.cv[m,1] = ptm["elapsed"]
92
93   # record cv error
94   cv_error[m,1] = tune.svm$best.performance
95   # tune.svm$performances
96   # tune.svm$best.parameters[1]
97   # record svm fit time
98   ptm = proc.time()
99   # svm.fit = svm(as.factor(Class)~., data = dat,
100  #               cost = tune.svm$best.parameters[1], gamma = tune.svm$best.parameters[2])
101  svm.fit = tune.svm$best.model
102  ptm = proc.time() - ptm
103  time.fit[m,1] = ptm["elapsed"]
104
105  svm.pred = predict(svm.fit, dat, type = "class")
106  train_error[m,2] = mean(dat[,dim(sampling.data)[2]] != svm.pred)
107  # table(dat[,dim(sampling.data)[2]],svm.pred)
108  svm.pred = predict(svm.fit, datt, type = "class")
109  test_error[m,2] = mean(datt[,dim(sampling.data)[2]] != svm.pred)
110  # table(datt[,dim(sampling.data)[2]], svm.pred)
111
112  # logistic regression
113  log.mod = glm(Class ~., data = dat, family = "binomial",
114               weights = ifelse(dat$Class == 0, 1/n0, 1/n1))
115  log.pred = predict(log.mod, newdata = sampling.data[train,], type = "response")
116  log.pred = ifelse(log.pred > 0.5, 1, 0)
117  train_error[m,3] = mean( sampling.data[train,dim(sampling.data)[2]] != log.pred)
118  log.pred = predict(log.mod, newdata = sampling.data[-train,], type = "response")
119  log.pred = ifelse(log.pred > 0.5, 1, 0)
120  test_error[m,3] = mean( sampling.data[-train,dim(sampling.data)[2]] != log.pred)
121
122  # lasso cross validation and tune lambda
123  # record lasso cv time
124  ptm = proc.time()
125  cv.lasso = cv.glmnet(X[train,], y[train], alpha = 1, family = "binomial",
126                     intercept = T, type.measure="class",
127                     weights = ifelse(y[train] == 0, 1/n0, 1/n1))
128  ptm = proc.time() - ptm
129  time.cv[m,2] = ptm["elapsed"]
130
131  cv_error[m,2] = min(cv.lasso$cvm)
132  bestlam = cv.lasso$lambda.min
133
134  # record lasso fit time
135  ptm = proc.time()

```

```

136 lasso.mod = glmnet(X[train,], y[train], alpha = 1, family = "binomial",
137                   intercept = T, lambda = bestlam,
138                   standardize = F)
139 ptm = proc.time() - ptm
140 time.fit[m,2] = ptm["elapsed"]
141
142 lasso.coef[,m] = coef(lasso.mod)[,1]
143 lasso.pred = predict(lasso.mod, s = bestlam, newx = X[train,], type = "class")
144 train_error[m,4] = mean(y[train] != lasso.pred)
145 lasso.pred = predict(lasso.mod, s = bestlam, newx = X[-train,], type = "class",)
146 test_error[m,4] = mean(y[-train] != lasso.pred)
147
148 # ridge cross validation and tune lambda
149 # record ridge cv time
150 ptm = proc.time()
151 cv.ridge = cv.glmnet(X[train,], y[train], alpha = 0, family = "binomial",
152                    intercept = T, type.measure = "class",
153                    weights = ifelse(y[train] == 0, 1/n0, 1/n1))
154 ptm = proc.time() - ptm
155 time.cv[m,3] = ptm["elapsed"]
156
157 cv_error[m,3] = min(cv.ridge$cvm)
158 bestlam = cv.ridge$lambda.min
159
160 # record ridge fit time
161 ptm = proc.time()
162 ridge.mod = glmnet(X[train,], y[train], alpha = 0, family = "binomial",
163                  intercept = T, lambda = bestlam,
164                  standardize = F)
165 ptm = proc.time() - ptm
166 time.fit[m,3] = ptm["elapsed"]
167
168 ridge.coef[,m] = as.matrix(coef(ridge.mod))
169 ridge.pred = predict(ridge.mod, s = bestlam, newx = X[train,], type = "class")
170 train_error[m,5] = mean(y[train] != ridge.pred)
171 ridge.pred = predict(ridge.mod, s = bestlam, newx = X[-train,], type = "class")
172 test_error[m,5] = mean(y[-train] != ridge.pred)
173
174 #random forest with 500 bootstrapped trees
175 ptm = proc.time()
176 rf = randomForest(x = sampling.data[train,-dim(sampling.data)[2]],
177                  y = as.factor(sampling.data[train,dim(sampling.data)[2]]), data =
sampling.data[train,],
178                  mtry = sqrt(p), classwt = weight)
179 ptm = proc.time() - ptm
180 time.fit[m,4] = ptm["elapsed"]

```

```

181
182 rf.pred = predict(rf, sampling.data[train,-dim(sampling.data)[2]], type = "class")
183 train_error[m,1] = mean(sampling.data[train,dim(sampling.data)[2]] != rf.pred)
184 train_error[m,6] = mean(rf$err.rate[,1])
185 train_error[m,7] = sd(rf$err.rate[,1])
186 rf.pred = predict(rf, sampling.data[-train,], type = "class")
187 test_error[m,1] = mean(sampling.data[-train,dim(sampling.data)[2]] != rf.pred)
188
189
190 }
191
192 #####
193 #####
194
195 # store error rate and coef
196 write.csv(ridge.coef, file = "D50_rdige_coef.csv")
197 write.csv(lasso.coef, file = "D50_lasso_coef.csv")
198
199 write.csv(cv_error, file = "D50_cv_error.csv")
200 write.csv(test_error, file = "D50_test_error.csv")
201 write.csv(train_error, file = "D50_train_error.csv")
202
203 write.csv(time.cv, file = "D50_time_cv.csv")
204 write.csv(time.fit, file = "D50_time_fit.csv")
205
206 # read csv files of project results
207 D5.r.coef = read.csv("D50_rdige_coef.csv")
208 D5.l.coef = read.csv("D50_lasso_coef.csv")
209
210 D5.cv.error = read.csv("D50_cv_error.csv")
211 D5.test.error = read.csv("D50_test_error.csv")
212 D5.train.error = read.csv("D50_train_error.csv")
213 colnames(D5.train.error)[7:8] = c("RF.00B", "RF.00Bsd")
214
215 D5.time.cv = read.csv("D50_time_cv.csv")
216 D5.time.fit = read.csv("D50_time_fit.csv")
217
218 D9.r.coef = read.csv("D90_rdige_coef.csv")
219 D9.l.coef = read.csv("D90_lasso_coef.csv")
220
221 D9.cv.error = read.csv("D90_cv_error.csv")
222 D9.test.error = read.csv("D90_test_error.csv")
223 D9.train.error = read.csv("D90_train_error.csv")
224 colnames(D9.train.error)[7:8] = c("RF.00B", "RF.00Bsd")
225
226 D9.time.cv = read.csv("D90_time_cv.csv")

```

```

227 D9.time.fit = read.csv("D90_time_fit.csv")
228
229 # boxplot of error rates for each nlearn
230 f1_1 = ggplot(melt(D5.train.error[,2:7]), aes(x = variable, y = value, color = variable)) +
231   geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") +
232   scale_color_brewer(palette="Dark2") +
233   labs(x = element_blank(), y = "Train Error Rate", title =
234     expression(n[learn] ~ "=" ~ n/2 ~ Train ~ Error ~ Rate))
235 f1_2 = ggplot(melt(D5.test.error[,2:6]), aes(x = variable, y = value, color = variable)) +
236   geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") +
237   scale_color_brewer(palette="Dark2") +
238   labs(x = element_blank(), y = "Test Error Rate", title =
239     expression(n[learn] ~ "=" ~ n/2 ~ Test ~ Error ~ Rate))
240 f1_3 = ggplot(melt(D5.cv.error[,2:4]), aes(x = variable, y = value, color = variable)) +
241   geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") +
242   scale_color_brewer(palette="Dark2") +
243   labs(x = element_blank(), y = "CV Error Rate", title =
244     expression(n[learn] ~ "=" ~ n/2 ~ CV ~ Error ~ Rate))
245
246 f1_4 = ggplot(melt(D9.train.error[,2:7]), aes(x = variable, y = value, color = variable)) +
247   geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") +
248   scale_color_brewer(palette="Dark2") +
249   labs(x = element_blank(), y = "Train Error Rate", title =
250     expression(n[learn] ~ "=" ~ 0.9 ~ n ~ Train ~ Error ~ Rate))
251 f1_5 = ggplot(melt(D9.test.error[,2:6]), aes(x = variable, y = value, color = variable)) +
252   geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") +
253   scale_color_brewer(palette="Dark2") +
254   labs(x = element_blank(), y = "Test Error Rate", title =
255     expression(n[learn] ~ "=" ~ 0.9 ~ n ~ Test ~ Error ~ Rate))
256 f1_6 = ggplot(melt(D9.cv.error[,2:4]), aes(x = variable, y = value, color = variable)) +
257   geom_boxplot() + ylim(0,0.25) + theme(legend.position="none") +
258   scale_color_brewer(palette="Dark2") +
259   labs(x = element_blank(), y = "CV Error Rate", title =
260     expression(n[learn] ~ "=" ~ 0.9 ~ n ~ CV ~ Error ~ Rate))
261
262 print(f1_3)
263
264 f1.1 = grid.arrange(f1_1, f1_2, f1_3, nrow = 1, widths = c(1.5,1.5,1))
265 f1.2 = grid.arrange(f1_4, f1_5, f1_6, nrow = 1, widths = c(1.5,1.5,1))
266
267 # cv ridge error
268 cv.ridge$lambda
269 length(cv.ridge$lambda)
270 which.min(cv.ridge$cvm)
271
272 cv.ridge.coef = matrix(0, nrow = length(cv.ridge$lambda), ncol = 1)
273 for (i in 1:length(cv.ridge$lambda)) {
274   cv.ridge.coef[i,] = sqrt(sum(coef(cv.ridge, s = cv.ridge$lambda[i])[-1,]^2))

```

```

261 }
262 cv.ridge.rate.coef = round(cv.ridge.coef/cv.ridge.coef[length(cv.ridge$lambda),1],3)
263 cv.ridge.error.and.rate.coef = data.frame(cbind(cv.ridge.rate.coef,cv.ridge$cvm))
264
265 # cv lasso error
266 length(cv.lasso$lambda)
267 which.min(cv.lasso$cvm)
268 cv.lasso.coef = matrix(0, nrow = length(cv.lasso$lambda), ncol = 1)
269 for (j in 1:length(cv.lasso$lambda)) {
270   cv.lasso.coef[j,] = sqrt(sum(coef(cv.lasso, s = cv.lasso$lambda[j])[-1,]^2))
271 }
272 cv.lasso.rate.coef = round(cv.lasso.coef/cv.lasso.coef[length(cv.lasso$lambda),1],3)
273 cv.lasso.error.and.rate.coef = data.frame(cbind(cv.lasso.rate.coef,cv.lasso$cvm))
274
275 D9.cv.lasso.error.and.rate.coef = read.csv("D90_cv.lasso.error.and.rate.coef.csv")
276 D9.cv.ridge.error.and.rate.coef = read.csv("D90_cv.ridge.error.and.rate.coef.csv")
277
278 # svm cv error rate
279 cv.svm.error = data.frame(tune.svm$performances)
280 D9.cv.svm.error = read.csv("D9_cv.svm.error.csv")
281
282 # plot cv error rate for lasso, ridge, and svm
283 f2_1 = ggplot() +
284   geom_line(data = cv.lasso.error.and.rate.coef, aes(x = X1, y = X2, color = "LASSO")) +
285   geom_point(data = cv.lasso.error.and.rate.coef, aes(x = X1[which.min(X2)],
286     y = min(X2), color = "LASSO")) +
287   geom_line(data = cv.ridge.error.and.rate.coef, aes(x = X1, y = X2, color = "Ridge")) +
288   geom_point(data = cv.ridge.error.and.rate.coef, aes(x = X1[which.min(X2)],
289     y = min(X2), color = "Ridge")) +
290   labs(x = "L2 Norm Beta Hat Ratio", y = "CV Error Rate",
291     title = expression(n[learn]~"="~n/2~CV~LASSO~and~Ridge~Error~Rate)) +
292   scale_color_manual(name = element_blank(), values = c("LASSO" = "red", "Ridge" = "blue")) +
293   ylim(0,.6)
294 print(f2_1)
295 f2_2 = ggplot(data = cv.svm.error, aes(as.factor(cost), as.factor(gamma), fill = error)) +
296   geom_tile()+
297   labs(x = "cost", y = "gamma",
298     title = expression(n[learn]~"="~n/2~CV~SVM~Error~Rate),fill = "CV Error Rate")
299 f2_3 = ggplot() +
300   geom_line(data = D9.cv.lasso.error.and.rate.coef, aes(x = X1, y = X2, color = "LASSO")) +
301   geom_point(data = cv.lasso.error.and.rate.coef, aes(x = X1[which.min(X2)],
302     y = min(X2), color = "LASSO")) +
303   geom_line(data = D9.cv.ridge.error.and.rate.coef, aes(x = X1, y = X2, color = "Ridge")) +
304   geom_point(data = D9.cv.ridge.error.and.rate.coef, aes(x = X1[which.min(X2)],
305     y = min(X2), color = "Ridge")) +
306   labs(x = "L2 Norm Beta Hat Ratio", y = "CV Error Rate",

```

```

307     title = expression(n[learn]~"~0.9~n~~CV~LASSO~and~Ridge~Error~Rate)) +
308     scale_color_manual(name = element_blank(), values = c("LASSO" = "red", "Ridge" = "blue")) +
309     ylim(0,.6)
310 f2_4 = ggplot(data = D9.cv.svm.error, aes(as.factor(cost), as.factor(gamma), fill = error)) +
311     geom_tile()+
312     labs(x = "cost", y = "gamma",
313          title = expression(n[learn]~"~0.9~n~~CV~SVM~Error~Rate),fill = "CV Error Rate")
314 f2 = grid.arrange(f2_1, f2_2, f2_3, f2_4, nrow = 2 , widths = c(1,1))
315
316 # time analysis
317 colMeans(D5.time.cv)
318 apply(D5.time.cv, 2, sd)
319 colMeans(D9.time.cv)
320 apply(D9.time.cv, 2, sd)
321
322 colMeans(D5.time.fit)
323 apply(D5.time.fit, 2, sd)
324 colMeans(D9.time.fit)
325 apply(D9.time.fit, 2, sd)
326
327 colMeans(D5.test.error)
328 apply(D5.test.error, 2, sd)
329 colMeans(D9.test.error)
330 apply(D9.test.error, 2, sd)
331
332 # variable importance
333 D5.l.coef = D5.l.coef[,-1]
334 D5.l.variable.importance = data.frame(t(abs(rowMeans(D5.l.coef))))
335 sort(D5.l.variable.importance, decreasing = T)[1:10]
336
337 D5.r.coef = D5.r.coef[,-1]
338 D5.r.variable.importance = data.frame(t(abs(rowMeans(D5.r.coef))))
339 sort(D5.r.variable.importance, decreasing = T)[1:10]
340
341 D5.rf.variable.importance = data.frame(t(read.csv("D5_rf.variable.importance.csv")[-1]))
342 sort(D5.rf.variable.importance, decreasing = T)[1:10]
343
344 D9.l.coef = D9.l.coef[,-1]
345 D9.l.variable.importance = data.frame(t(abs(rowMeans(D9.l.coef))))
346 sort(D9.l.variable.importance, decreasing = T)[1:10]
347
348 D9.r.coef = D9.r.coef[,-1]
349 D9.r.variable.importance = data.frame(t(abs(rowMeans(D9.r.coef))))
350 sort(D9.r.variable.importance, decreasing = T)[1:10]
351
352 D9.rf.variable.importance = data.frame(t(read.csv("D9_rf.variable.importance.csv")[-1]))

```



```

353 sort(D9.rf.variable.importance, decreasing = T)[1:10]
354
355 # names of important variables
356 important.variables = rbind(names(sort(D5.l.variable.importance, decreasing = T)[1:10]),
357 names(sort(D5.r.variable.importance, decreasing = T)[1:10]),
358 names(sort(D5.rf.variable.importance, decreasing = T)[1:10]),
359 names(sort(D9.l.variable.importance, decreasing = T)[1:10]),
360 names(sort(D9.r.variable.importance, decreasing = T)[1:10]),
361 names(sort(D9.rf.variable.importance, decreasing = T)[1:10]))
362 write.csv(important.variables, file = "important.variables.csv")
363
364
365 f3_1 = ggplot(melt(D5.l.variable.importance), aes(x = variable, y = value, color = variable))
+
366   geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
367   labs(x = element_blank(), y = "Absolute Value of Coefficients",
368        title = expression(n[learn]~"="~n/2~~LASSO~Variable~Importance)) +
369   theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
370 f3_2 = ggplot(melt(D5.r.variable.importance), aes(x = variable, y = value, color = variable))
+
371   geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
372   labs(x = element_blank(), y = "Absolute Value of Coefficients",
373        title = expression(n[learn]~"="~n/2~~Ridge~Variable~Importance)) +
374   theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
375 f3_3 = ggplot(melt(D5.rf.variable.importance), aes(x = variable, y = value, color = variable))
+
376   geom_bar(stat="identity") + ylim(0,20) + theme(legend.position="none")+
377   labs(x = element_blank(), y = "Variable Importance",
378        title = expression(n[learn]~"="~n/2~~RF~Variable~Importance)) +
379   theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
380 f3_4 = ggplot(melt(D9.l.variable.importance), aes(x = variable, y = value, color = variable))
+
381   geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
382   labs(x = element_blank(), y = "Absolute Value of Coefficients",
383        title = expression(n[learn]~"="~0.9~n~~LASSO~Variable~Importance)) +
384   theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
385 f3_5 = ggplot(melt(D9.r.variable.importance), aes(x = variable, y = value, color = variable))
+
386   geom_bar(stat="identity") + ylim(0,1) + theme(legend.position="none")+
387   labs(x = element_blank(), y = "Absolute Value of Coefficients",
388        title = expression(n[learn]~"="~0.9~n~~Ridge~Variable~Importance)) +
389   theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
390 f3_6 = ggplot(melt(D9.rf.variable.importance), aes(x = variable, y = value, color = variable))
+
391   geom_bar(stat="identity") + ylim(0,20) + theme(legend.position="none")+
392   labs(x = element_blank(), y = "Variable Importance",

```

```

393     title = expression(n[learn]~"~0.9~n~RF~Variable~Importance)) +
394     theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())
395 f3 = grid.arrange(f3_1, f3_2, f3_3, f3_4, f3_5, f3_6, nrow = 2)
396
397 D5.importance = matrix(0,nrow = 400,ncol = 6)
398 D5.importance[,1:6] = cbind(t(D5.l.variable.importance)[,1], t(D5.r.variable.importance)[,1],
399                             t(D5.rf.variable.importance)[,1],t(D9.l.variable.importance)[,1],
400                             t(D9.r.variable.importance)[,1],
401                             t(D9.rf.variable.importance)[,1])
402 colMeans(D5.importance)
403 D9.l.variable.importance = data.frame(t(abs(rowMeans(D9.l.coef))))
404 sort(D9.l.variable.importance, decreasing = T)[1:10]
405
406 D9.r.coef = D9.r.coef[-1,-1]
407 D9.r.variable.importance = data.frame(t(abs(rowMeans(D9.r.coef))))
408 sort(D9.r.variable.importance, decreasing = T)[1:10]
409
410 D9.rf.variable.importance = data.frame(t(read.csv("D9_rf.variable.importance.csv")[-1]))
411 sort(D9.rf.variable.importance, decreasing = T)[1:10]
412
413
414 x = rep(1:20,20)
415 y = c(rep(1,20),rep(2,20),rep(3,20),rep(4,20),rep(5,20),
416       rep(6,20),rep(7,20),rep(8,20),rep(9,20),rep(10,20),
417       rep(11,20),rep(12,20),rep(13,20),rep(14,20),rep(15,20),
418       rep(16,20),rep(17,20),rep(18,20),rep(19,20),rep(20,20))
419
420 w = data.frame(cbind(x,y,D5.importance[,1:6]))
421 f5_1 = ggplot(w, aes(x = x, y = y, fill = w[,3])) +
422     geom_tile() + labs(x = "X pixel", y = "Y pixel",
423                       title = "n/2, Estimated Position of Nucleus by LASSO",fill = "V.Imp")
424 f5_2 = ggplot(w, aes(x = x, y = y, fill = w[,4])) +
425     geom_tile() + labs(x = "X pixel", y = "Y pixel",
426                       title = "n/2, Estimated Position of Nucleus by Ridge",fill = "V.Imp")
427 f5_3 = ggplot(w, aes(x = x, y = y, fill = w[,5])) +
428     geom_tile() + labs(x = "X pixel", y = "Y pixel",
429                       title = "n/2, Estimated Position of Nucleus by RF",fill = "V.Imp")
430 f5_4 = ggplot(w, aes(x = x, y = y, fill = w[,6])) +
431     geom_tile() + labs(x = "X pixel", y = "Y pixel",
432                       title = "0.9n, Estimated Position of Nucleus by LASSO",fill = "V.Imp")
433 f5_5 = ggplot(w, aes(x = x, y = y, fill = w[,7])) +
434     geom_tile() + labs(x = "X pixel", y = "Y pixel",
435                       title = "0.9n, Estimated Position of Nucleus by Ridge",fill = "V.Imp")
436 f5_6 = ggplot(w, aes(x = x, y = y, fill = w[,8])) +
437     geom_tile() + labs(x = "X pixel", y = "Y pixel",
438                       title = "0.9n, Estimated Position of Nucleus by RF",fill = "V.Imp")

```

```
439 f5 = grid.arrange(f5_1, f5_2, f5_3, f5_4, f5_5, f5_6, nrow = 2)
440
441
442 # two test error rate difference t.test
443 test.error = cbind(D5.test.error, D9.test.error[, -1])
444 test.error.difference = matrix(0, nrow = 100, ncol = 5)
445 test.error.difference = test.error[, 2:6] - test.error[, 7:11]
446 t.test(test.error.difference$RF, mu = 0)
447 t.test(test.error.difference$R.SVM, mu = 0)
448 t.test(test.error.difference$Log, mu = 0)
449 t.test(test.error.difference$Log.LASSO, mu = 0)
450 t.test(test.error.difference$Log.Ridge, mu = 0)
451
```