Homework 3 Solutions

Alexander McLain

Question One

For this question, we'll use the dataset parkinsons data used in the previous homework. Recall, the data is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring.

For all of the following, ignore that multiple observations come from the same person. This should not be done in practice, especially when we're doing any type of inference.

Question One part a (3 points)

Expand the first 16 voice measurements using B-splines with 4 degrees of freedom and degree 3, i.e., use `bs( , df=4, degree = 3)`. Use this data for the rest of the problem.

```r
library(tidyverse)
library(readr)
library(splines)
parkinsons <- read_csv("parkinsons_updrs.txt",
                       show_col_types = FALSE) %>%
  na.omit()
# only use 16 voice measurements to predict total_UPDRS
parkinsons_voice <- parkinsons[, 7:22]
```

```r
# Using dplyr to mutate new B-spline columns
parkinsons_voice_bs <- parkinsons_voice %>%
  mutate(
    across(1:16, ~ bs(.x, df = 4, degree = 3),
           .names = "bs_{.col}",
           .unpack = "{inner}")
    )
# add outcome variable back into dataset and only keep the BS variables
parkinsons_bs <- parkinsons_voice_bs %>%
  add_column(total_UPDRS = parkinsons$total_UPDRS) %>%
  dplyr::select(total_UPDRS,
              starts_with("bs_"))
```

Question One part b (7 points)

Analyze the data using the grouped lasso model, similar to the last homework. However, this time use data splitting to get p-values and confidence intervals of all coefficients retained in the grouped lasso model. Note the "inference set'' analyses will be based on standard least squares.

Here, we'll be splitting the data. Also, recall that with the `gglasso` *model scaling has to be done outside the function.*

The correct way to do this will be to center and scale the training or ***selection*** *X dataset. Note that we don't have to center and scale the* ***inference*** *set because the function we'll use (`lm`) doesn't require centering or scaling the data.*

```r
# Data Splitting
set.seed(172389)
sample_idx <- sample(seq_len(nrow(parkinsons_bs)), size = 0.6 * nrow(parkinsons_bs))
selection_set <- parkinsons_bs[sample_idx, ]
inference_set <- parkinsons_bs[-sample_idx, ]

# Define your predictors and response variables for the selection set
X_selection <- as.matrix(
  selection_set %>%
    dplyr::select(-"total_UPDRS")
                         )
y_selection <- as.vector(selection_set$total_UPDRS)

# Center and scale the X variables from the selection set
X_selection <- scale(X_selection, center = TRUE, scale = TRUE)
```

```r
library(gglasso)
# Get the group's (very important step)
groups <- rep(1:16, each = 4)

# Fit the grouped lasso model
fit_grouped_lasso <- cv.gglasso(X_selection, y_selection,
                                group = groups, loss = "ls",
                                pred.loss = "L2", nfolds=5)
```

```r
# Get the coefficients from the grouped lasso fit
lasso_coef <- coef(fit_grouped_lasso, s = fit_grouped_lasso$lambda.1se)[-1]

# Identify the non-zero coefficients (features retained by the lasso)
non_zero_coef <- which(lasso_coef != 0)
colnames(X_selection)[non_zero_coef]
```

```
##  [1] "bs_Jitter(Abs).1"   "bs_Jitter(Abs).2"   "bs_Jitter(Abs).3"
##  [4] "bs_Jitter(Abs).4"   "bs_Jitter:PPQ5.1"   "bs_Jitter:PPQ5.2"
##  [7] "bs_Jitter:PPQ5.3"   "bs_Jitter:PPQ5.4"   "bs_Shimmer:APQ3.1"
## [10] "bs_Shimmer:APQ3.2"  "bs_Shimmer:APQ3.3"  "bs_Shimmer:APQ3.4"
## [13] "bs_Shimmer:APQ11.1" "bs_Shimmer:APQ11.2" "bs_Shimmer:APQ11.3"
## [16] "bs_Shimmer:APQ11.4" "bs_Shimmer:DDA.1"   "bs_Shimmer:DDA.2"
## [19] "bs_Shimmer:DDA.3"   "bs_Shimmer:DDA.4"   "bs_NHR.1"
## [22] "bs_NHR.2"           "bs_NHR.3"           "bs_NHR.4"
## [25] "bs_HNR.1"           "bs_HNR.2"           "bs_HNR.3"
## [28] "bs_HNR.4"           "bs_RPDE.1"          "bs_RPDE.2"
## [31] "bs_RPDE.3"          "bs_RPDE.4"          "bs_DFA.1"
## [34] "bs_DFA.2"           "bs_DFA.3"           "bs_DFA.4"
## [37] "bs_PPE.1"           "bs_PPE.2"           "bs_PPE.3"
## [40] "bs_PPE.4"
```

From here, there are two ways to proceed.

The first, we'll ignore the grouping for the rest of the problem and run a standard linear model. Here's the code for it:

```r
# Create inference set data only keeping the variables of interest
# Turning this into a matrix so all columns can be indexed.
# Adding 1 since inference set has the Y in the first column.
# Turning it into a data.frame so it can be used with lm
inference_set_lm <- data.frame(
  as.matrix(inference_set)[,c(1,non_zero_coef+1)]
)

# Fit a least squares model using only the features retained by the grouped lasso
fit_least_squares <- lm(total_UPDRS ~ . , data = inference_set_lm)
```

The second will use the grouping. You may have noticed that when you use `across` above the data looks like it only has 16 X variables when in actuality it has 64. For example, notice that

```r
dim(inference_set)
```

```
## [1] 2350   17
```

```r
dim(as.matrix(inference_set))
```

```
## [1] 2350   65
```

The `as.matrix` removes the grouping, which is why I used it above (a lot of times it's easier to work with the data when the grouping is removed). This happens because `across` generates a data which keeps the grouping structure.

Here's how to run sample splitting with the grouping structure intact.

```r
#First, figure out which groups were selected:
sel_groups <- unique(sort(groups[non_zero_coef]))
sel_groups
```

```
##  [1]  2  4  8 10 11 12 13 14 15 16
```

```r
# Keep all the groups
# Adding 1 since inference set has the Y in the first column.
inference_set_lm_gp <- inference_set[,c(1,sel_groups+1)]
# Fit a least squares model using only the features retained by the grouped lasso
fit_least_squares_gp <- lm(total_UPDRS ~ . , data = inference_set_lm_gp)

# Get summary statistics, including p-values and confidence intervals
summary_fit_gp <- summary(fit_least_squares_gp)
```

The results of this will look the same. The difference is that we can use the `anova` function to get group level p-values (type III tests). That's not possible when using the first method.

Take a look at the p-values and confidence intervals:

```r
# First, let's look at some p-values for the type III tests
anova(fit_least_squares_gp)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
`bs_Jitter(Abs)`	4	8492.8445	2123.21112	22.503415	0.0000000
`bs_Jitter:PPQ5`	4	1708.6393	427.15983	4.527367	0.0012085
`bs_Shimmer:APQ3`	4	1996.7279	499.18198	5.290712	0.0003059
`bs_Shimmer:APQ11`	4	4737.1179	1184.27946	12.551899	0.0000000
`bs_Shimmer:DDA`	4	592.7346	148.18365	1.570564	0.1794329
bs_NHR	4	1531.8943	382.97357	4.059047	0.0027790

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
bs_HNR	4	9202.0619	2300.51548	24.382623	0.0000000
bs_RPDE	4	795.0760	198.76899	2.106706	0.0775021
bs_DFA	4	13535.2083	3383.80209	35.864123	0.0000000
bs_PPE	4	3682.8593	920.71482	9.758440	0.0000001
Residuals	2309	217855.5741	94.35062	NA	NA

```r
# Second, look at the individual level p-values and CI's
round(cbind(summary_fit_gp$coefficients, confint(fit_least_squares_gp)), 3)
```

	Estimate	Std. Error	t value	Pr(>|t|)	2.5 %	97.5 %
(Intercept)	39.158	18.744	2.089	0.037	2.402	75.915
bs_Jitter(Abs)1	15.269	4.057	3.764	0.000	7.313	23.225
bs_Jitter(Abs)2	1.735	5.755	0.301	0.763	-9.551	13.021
bs_Jitter(Abs)3	4.377	12.882	0.340	0.734	-20.883	29.638
bs_Jitter(Abs)4	14.259	13.122	1.087	0.277	-11.473	39.992
bs_Jitter:PPQ51	-3.054	4.803	-0.636	0.525	-12.473	6.365
bs_Jitter:PPQ52	24.226	11.397	2.126	0.034	1.877	46.575
bs_Jitter:PPQ53	-33.260	27.822	-1.195	0.232	-87.819	21.298
bs_Jitter:PPQ54	11.092	23.111	0.480	0.631	-34.229	56.413
bs_Shimmer:APQ31	-193.993	1856.358	-0.105	0.917	-3834.295	3446.309
bs_Shimmer:APQ32	7323.852	7386.670	0.991	0.322	-7161.349	21809.053
bs_Shimmer:APQ33	-46027.044	38902.283	-1.183	0.237	-122314.107	30260.018
bs_Shimmer:APQ34	131961.912	90050.325	1.465	0.143	-44626.047	308549.870
bs_Shimmer:APQ111	6.896	4.571	1.509	0.132	-2.068	15.860
bs_Shimmer:APQ112	44.771	6.938	6.453	0.000	31.165	58.376
bs_Shimmer:APQ113	-9.119	16.439	-0.555	0.579	-41.356	23.117
bs_Shimmer:APQ114	57.836	24.688	2.343	0.019	9.422	106.249
bs_Shimmer:DDA1	172.188	1854.341	0.093	0.926	-3464.160	3808.537
bs_Shimmer:DDA2	-7358.024	7385.995	-0.996	0.319	-21841.901	7125.852
bs_Shimmer:DDA3	45993.029	38901.181	1.182	0.237	-30291.871	122277.930
bs_Shimmer:DDA4	-131996.679	90054.802	-1.466	0.143	-308593.419	44600.060
bs_NHR1	-4.167	1.996	-2.088	0.037	-8.080	-0.253
bs_NHR2	-10.111	6.213	-1.627	0.104	-22.295	2.072
bs_NHR3	5.328	12.780	0.417	0.677	-19.734	30.390
bs_NHR4	-7.916	14.849	-0.533	0.594	-37.036	21.203
bs_HNR1	-18.300	18.346	-0.997	0.319	-54.276	17.677
bs_HNR2	22.733	14.939	1.522	0.128	-6.562	52.028
bs_HNR3	-21.355	16.856	-1.267	0.205	-54.410	11.699
bs_HNR4	-32.417	18.252	-1.776	0.076	-68.209	3.375
bs_RPDE1	15.575	10.161	1.533	0.125	-4.349	35.500
bs_RPDE2	9.920	6.820	1.455	0.146	-3.453	23.293
bs_RPDE3	13.466	9.889	1.362	0.173	-5.926	32.857
bs_RPDE4	19.540	10.310	1.895	0.058	-0.677	39.758
bs_DFA1	2.269	3.172	0.715	0.475	-3.952	8.490
bs_DFA2	-9.726	2.182	-4.457	0.000	-14.005	-5.446
bs_DFA3	-1.845	4.064	-0.454	0.650	-9.814	6.125
bs_DFA4	-34.131	4.910	-6.952	0.000	-43.759	-24.503
bs_PPE1	-16.368	4.600	-3.558	0.000	-25.389	-7.346
bs_PPE2	-16.652	4.143	-4.019	0.000	-24.777	-8.527
bs_PPE3	2.689	7.398	0.364	0.716	-11.819	17.197
bs_PPE4	-15.773	10.762	-1.466	0.143	-36.878	5.332

	Estimate	Std. Error	t value	Pr(>|t|)	2.5 %	97.5 %

Question One part c (7 points)

Analyze the data using a lasso model. Use Selective Inference Tools to get p-values and confidence intervals on the variables retained by the lasso model.

Recall that the selective inference package requires that X should be centered, I'm also going to scale them here.

```
library(glmnet)
library(selectiveInference)
# Fit the  lasso model
parkinsons_X <- as.matrix(
  parkinsons_bs %>%
    dplyr::select(-"total_UPDRS")
                          )
parkinsons_Y <- as.vector(parkinsons_bs$total_UPDRS)

parkinsons_X <- scale(parkinsons_X, TRUE, FALSE)

fit_lasso <- glmnet(parkinsons_X,
                    parkinsons_Y,
                    alpha = 1,
                    standardize = FALSE,
                    intercept = TRUE,
                    lambda = exp(seq(-7, 3, 0.1))
                    )
lambda <- exp(-3)

# extract coef for a given lambda minus the intercept term
beta <- coef(fit_lasso, x=parkinsons_X, y= parkinsons_Y, s = lambda, exact = TRUE)[-1]
```

Now run the selective inference functions.

```
inf <- fixedLassoInf(x=parkinsons_X,
                     y= parkinsons_Y,
                     beta,
                     lambda = lambda * nrow(parkinsons_X),
                     alpha = 0.05
                     )
```

Let's look at the p-values and confidence intervals.

```
print(inf)
```

```
##
## Call:
## fixedLassoInf(x = parkinsons_X, y = parkinsons_Y, beta = beta,
##     lambda = lambda * nrow(parkinsons_X), alpha = 0.05)
##
## Standard deviation of noise (specified or estimated) sigma = 9.719
##
## Testing results at lambda = 292.499, with alpha = 0.050
##
##  Var    Coef Z-score P-value LowConfPt UpConfPt LowTailArea UpTailArea
##    5   8.406   6.888   0.000     6.013   10.844       0.025      0.024
##   33  -8.512  -6.247   0.000   -14.473   -4.630       0.025      0.025
##   38   3.137   1.388   0.472   -15.608    7.355       0.000      0.025
##   50  30.037  12.200   0.000    24.744   34.943       0.024      0.025
##   51   3.149   1.813   0.845    -3.420    8.169       0.025      0.025
##   53  -1.379  -1.308   0.287    -3.430    3.301       0.024      0.025
##   57   2.938   1.596   0.083    -1.409    7.006       0.025      0.025
##   59  -6.134  -2.676   0.013   -11.342   -0.865       0.024      0.025
##   60 -14.864  -6.820   0.000   -19.146   -9.584       0.025      0.025
##   61 -16.068 -13.995   0.000   -18.497  -13.811       0.024      0.025
##
## Note: coefficients shown are partial regression coefficients
```

```r
inf$vars
```

```
##   bs_Jitter(Abs).1  bs_Shimmer:APQ5.1 bs_Shimmer:APQ11.2           bs_HNR.2
##                  5                 33                 38                 50
##           bs_HNR.3          bs_RPDE.1           bs_DFA.1           bs_DFA.3
##                 51                 53                 57                 59
##           bs_DFA.4           bs_PPE.1
##                 60                 61
```

Question One part d (3 points)

Compare the results from the previous two models. Which do you think is a better approach and why?

Main points to this answer:

- *The selective inference method has the benefit of using the whole sample. Here, that's not much of a plus since there are a lot of observations.*
- *The sample splitting method can keep the grouping of the variables intact for the selection and the inference portions of the analysis (see above).*

Overall, since there are a lot of observations I would use sample splitting in this scenario.

Question One part e (8 points)

Using conformal inference, get prediction intervals for 100 observations left out of the analysis. That is, extract 100 observations then use the rest to train the model and get the conformal scores. Use a lasso regression with the same λ you used above.

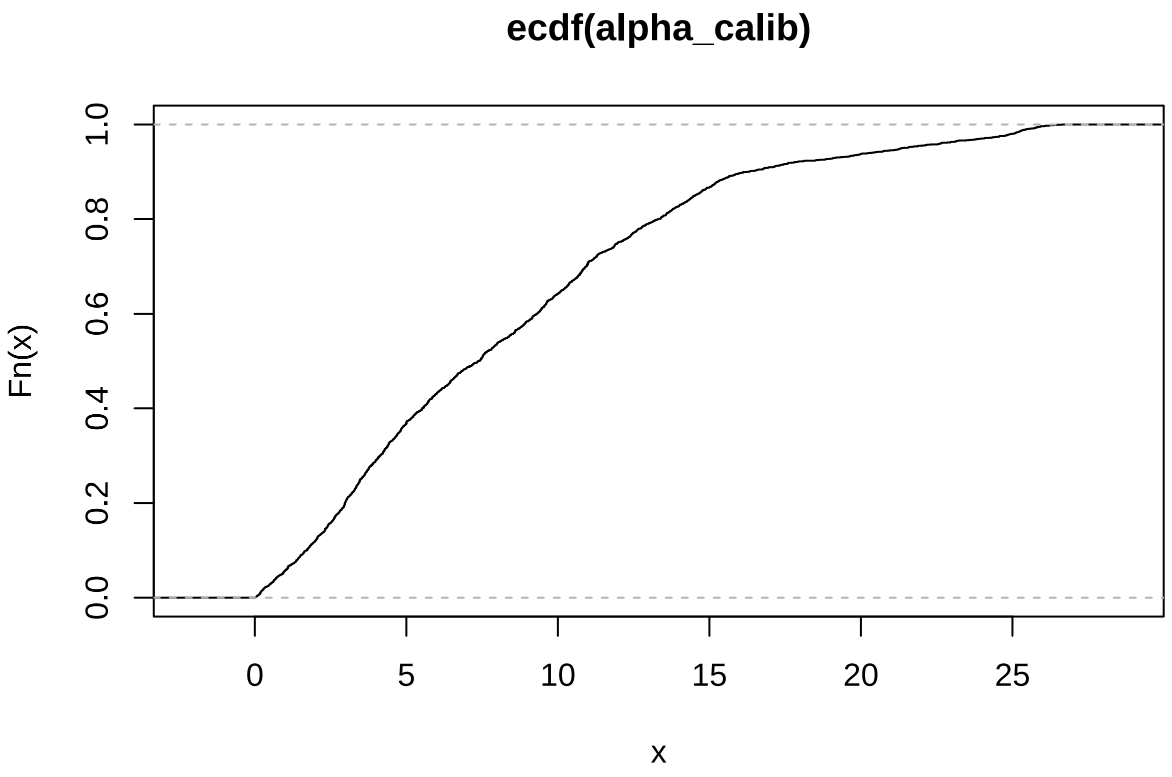
The following code follows the example we did in class.

First, split the data up.

```r
# extract 100 observations from the data set
test_indices <- sample(1:nrow(parkinsons_bs), 100)
tests_data <- parkinsons_bs[test_indices,]
rest_data <- parkinsons_bs[-test_indices,]
# split the rest of data into training and calibration sets
calib_indices <- sample(1:nrow(rest_data),
                        ceiling((nrow(rest_data)-1)*2/3))

train_data <- rest_data[calib_indices,]
calib_data <- rest_data[-calib_indices,]
```

Second, run the model and get the conformal scores.

```r
# Fit a LASSO model to the training data
X_train <- as.matrix(train_data[, -1])
y_train <- train_data$total_UPDRS
conformal_lasso_fit <- glmnet(X_train, y_train, alpha=1, lambda=1)

# Predict on the calibration set
X_calib <- as.matrix(calib_data[, -1])
y_calib <- calib_data$total_UPDRS
predictions_calib <- predict(conformal_lasso_fit, s=0.1, newx=X_calib)

# Compute nonconformity scores for the calibration set (absolute residuals)
alpha_calib <- abs(y_calib - predictions_calib)
plot(ecdf(alpha_calib))
```



Third, get all the prediction intervals.

```r
# Get the (n+1)*(1-alpha) largest nonconformity score
(ind_0.95 <- ceiling((length(alpha_calib)+1)*(0.95)))
```

```
## [1] 1830
```

```r
(alpha_0.95 <- sort(alpha_calib)[ind_0.95])
```

```
## [1] 21.42171
```

```r
# Predict on the test set
# New observation
X_pi <- as.matrix(tests_data[,-1])
y_pi <- tests_data$total_UPDRS
predicted_new <- predict(conformal_lasso_fit, s=0.1, newx=as.matrix(X_pi))
```

```r
# Use alpha_0.95 to form an 80% prediction interval.
conf_pi_lower <- predicted_new - alpha_0.95
conf_pi_upper <- predicted_new + alpha_0.95

table_conformal_interval <- data.frame(y_pi, conf_pi_lower, conf_pi_upper)
head(table_conformal_interval)
```

y_pi	s1	s1.1
24.702	8.185805	51.02923
39.273	8.132931	50.97635
21.722	6.213409	49.05683
34.807	7.374983	50.21840
38.428	6.748402	49.59182
33.428	8.632777	51.47620

Question One part f (2 points)

How many of the prediction intervals contain the true Y value?

```r
# Indicator that the interval contains the true value.
PI_ind <- 1*I(
  table_conformal_interval$s1 <= y_pi &
    table_conformal_interval$s1.1 >= y_pi
)
sum(PI_ind)
```

```
## [1] 98
```

```r
mean(PI_ind)
```

```
## [1] 0.98
```

98% of the true values are in the prediction interval.

Question Two

The dataset "pima.indians.diabetes3.xlsx" contains data from 532 females whom are at least 21 years of age and of Pima Indian heritage. The dataset consists of: npregnant: Number of times pregnant, glucose: Plasma glucose concentration a 2 hours in an oral glucose tolerance test, diastolic.bp: Diastolic blood pressure (mm Hg), skinfold.thickness: Triceps skin fold thickness (mm), bmi: Body mass index (weight in kg/(height in m)^2), pedigree: Diabetes pedigree function, age: Age (years), classdigit: Numerical class variable, and class: Alphanumeric class variable.

The point of this study was to indicate whether the patient shows signs of diabetes mellitus according to World Health Organization criteria (i.e., if the 2 hour post-load plasma glucose was at least 200 mg/dl at any survey examination or if found during routine care). The two class variables indicate diabetes status using a blood tests of 2-hour serum insulin. The goal is to use the other variables to predict diabetes status since getting information on them is easier for the patients, less invasive and cheaper than a blood-test. Use all other variables in each analysis.

Question Two part a (10 points)

Compute a LDA, draw overlaying histograms of the first LDF coordinate by diabetes status (similar to the histograms of the first PC done in the example in class).

Load in the data and take a look at it.

```
pima_dat <- data.frame(read.csv("pima.indians.diabetes3.csv",as.is=TRUE,header=TRUE))
dim(pima_dat)
```

```
## [1] 532   9
```

```
head(pima_dat)
```

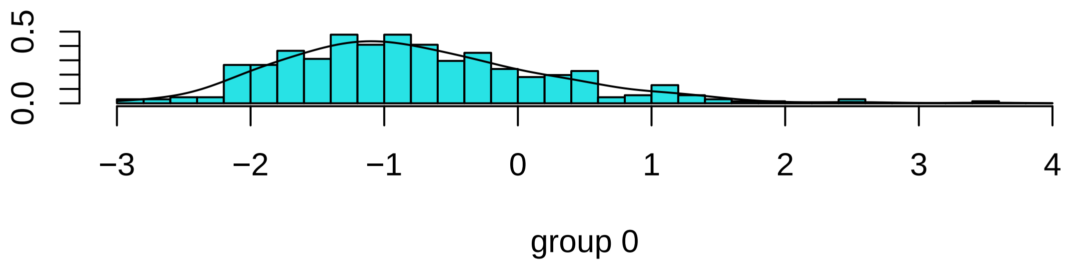
npregnant	glucose	diastolic.bp	skinfold.thickness	bmi	pedigree	age	classdigit	class
6	148	72	35	33.6	0.627	50	1	diabetic
1	85	66	29	26.6	0.351	31	0	normal
1	89	66	23	28.1	0.167	21	0	normal
0	137	40	35	43.1	2.288	33	1	diabetic
3	78	50	32	31.0	0.248	26	1	diabetic
2	197	70	45	30.5	0.158	53	1	diabetic

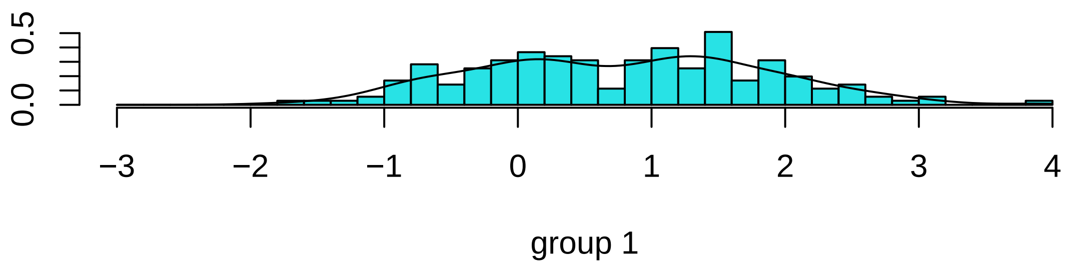
Now we'll fit and plot the results from a LDA.

```
library(MASS)
pima_dat <- pima_dat[,-9]
LDA_pima <- lda(classdigit ~.,data=pima_dat)
LDA_pima
```

```
## Call:
## lda(classdigit ~ ., data = pima_dat)
##
## Prior probabilities of groups:
##         0         1
## 0.6672932 0.3327068
##
## Group means:
##   npregnant  glucose diastolic.bp skinfold.thickness      bmi  pedigree
## 0  2.926761 110.0169     69.91268           27.29014 31.42958 0.4463155
## 1  4.700565 143.1186     74.70056           32.97740 35.81977 0.6165876
##        age
## 0 29.22254
## 1 36.41243
##
## Coefficients of linear discriminants:
##                             LD1
## npregnant           0.089453937
## glucose             0.026797651
## diastolic.bp       -0.004028686
## skinfold.thickness  0.002668636
## bmi                 0.052832219
## pedigree            0.801972183
## age                 0.019100341
```

```r
plot(LDA_pima, dimen=1, type="both")
```





Note that the LDA function will classify the above using a cutoff of zero, but we don't have to use this value. We could look for the best cutoff ourselves. This is actually a good thing to do when you think your data might not be normally distributed.

Question Two part b (5 points)

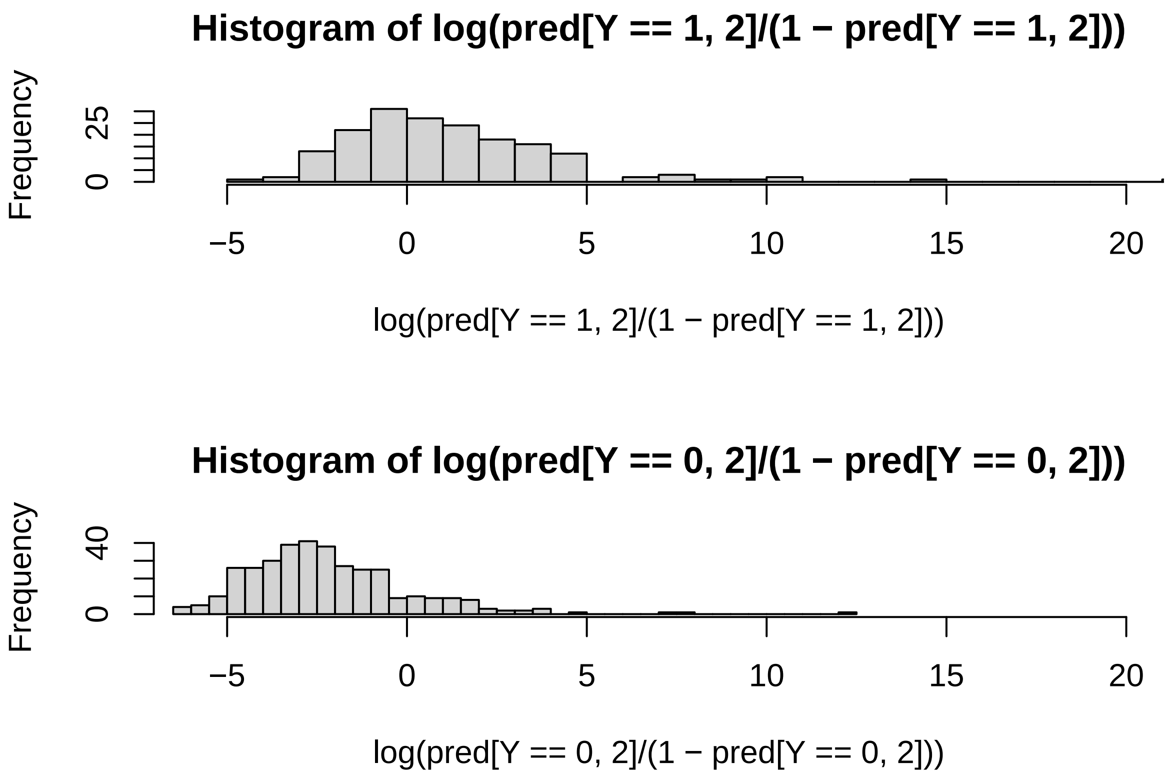
Compute a QDA to the data, draw histograms of the first LDF coordinate by diabetes status.

```r
QDA_pima <- qda(classdigit ~.,data=pima_dat)
QDA_pima
```

```
## Call:
## qda(classdigit ~ ., data = pima_dat)
##
## Prior probabilities of groups:
```

```
##         0         1 
## 0.6672932 0.3327068 
## 
## Group means:
##   npregnant  glucose diastolic.bp skinfold.thickness      bmi  pedigree
## 0  2.926761 110.0169     69.91268           27.29014 31.42958 0.4463155
## 1  4.700565 143.1186     74.70056           32.97740 35.81977 0.6165876
##        age
## 0 29.22254
## 1 36.41243
```

```
qfit <- qda(classdigit~.,
            data=pima_dat)
pred <- predict(qfit)$posterior
#Use a logit transform to get them on the linear scale.
par(mfrow = c(2,1))
Y <- pima_dat$classdigit
hist(log(pred[Y==1,2]/(1-pred[Y==1,2])), breaks = 30, xlim = c(-6,20))
hist(log(pred[Y==0,2]/(1-pred[Y==0,2])), breaks = 30, xlim = c(-6,20))
```



```
par(mfrow = c(1,1))
```

Question Two part c (7 points)

Perform a logistic regression. Draw overlaying histograms of the estimated logit function by diabetes status.

First, we'll perform a logistic regression on the data. Then we'll plot the predicted logit scores, i.e., $\hat{\beta}\boldsymbol{X}$, *and then plot a histogram separetly by diabetes status.*

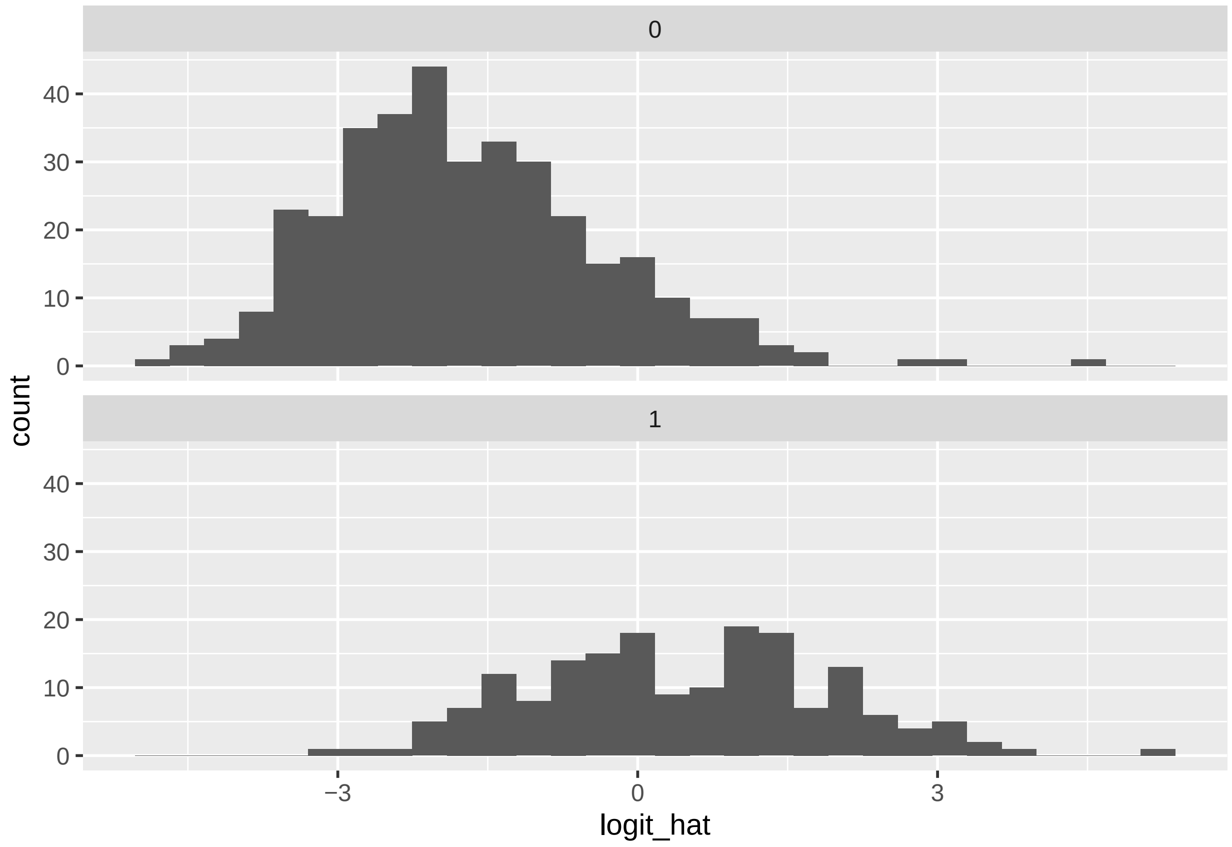
```r
Logis_pima <- glm(classdigit ~.,data=pima_dat,family=binomial)
summary(Logis_pima)
```

```
##
## Call:
## glm(formula = classdigit ~ ., family = binomial, data = pima_dat)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -9.554651   0.994217  -9.610  < 2e-16 ***
## npregnant            0.122517   0.043743   2.801 0.005097 **
## glucose              0.035321   0.004244   8.322  < 2e-16 ***
## diastolic.bp        -0.007695   0.010314  -0.746 0.455602
## skinfold.thickness   0.006774   0.014759   0.459 0.646242
## bmi                  0.082678   0.023334   3.543 0.000395 ***
## pedigree             1.308708   0.364040   3.595 0.000324 ***
## age                  0.026375   0.014000   1.884 0.059581 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.79  on 531  degrees of freedom
## Residual deviance: 466.32  on 524  degrees of freedom
## AIC: 482.32
##
## Number of Fisher Scoring iterations: 5
```

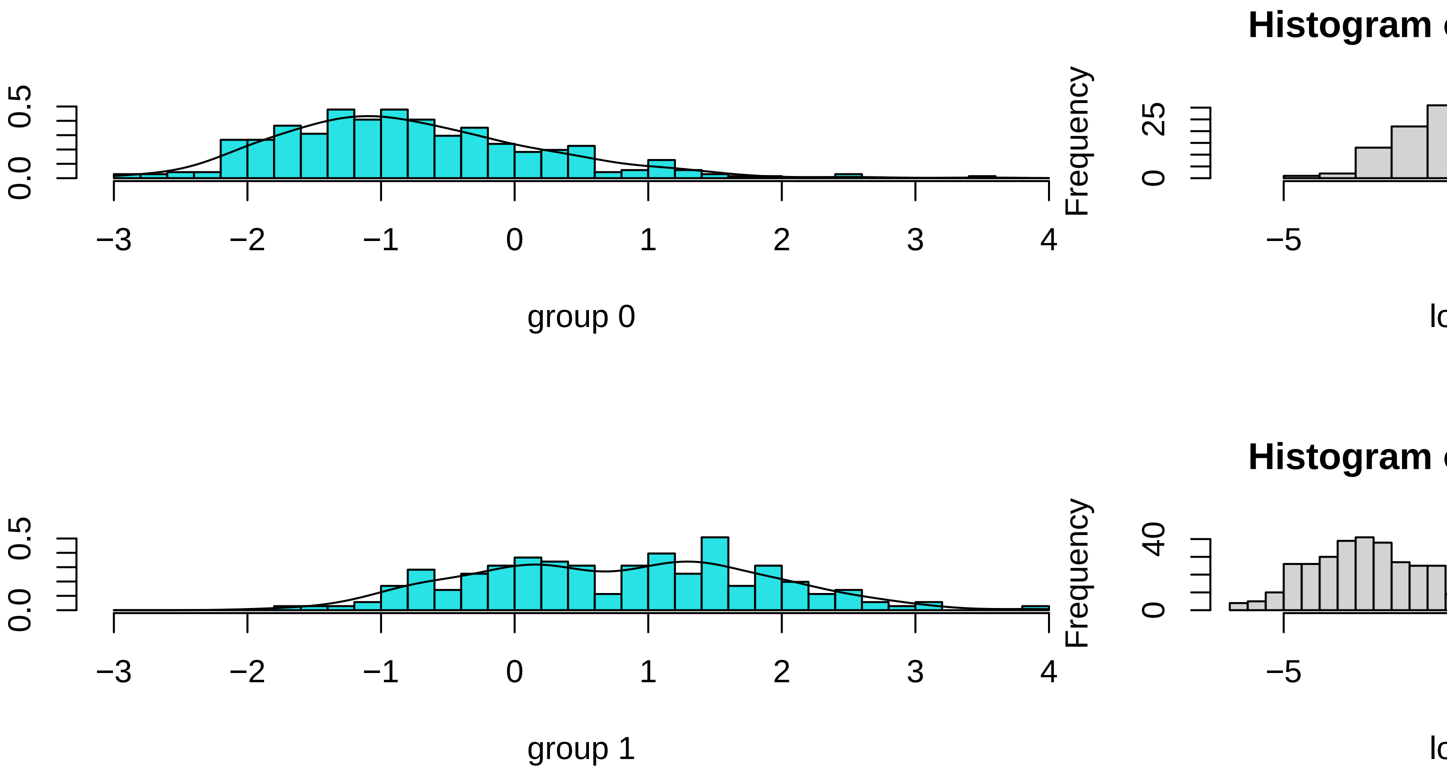
```r
logit_hat <- predict.glm(Logis_pima)
pima_dat_log <- cbind(pima_dat,logit_hat)
```

```r
library(ggplot2)
ggplot(pima_dat_log,aes(x=logit_hat)) + geom_histogram() +
  facet_wrap(vars(classdigit), nrow=2)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Compare with the LDA/QDA histrograms.



Question Two part d (3 points)

Comment on the figures. Which do you think will perform better classification?

Overall, the histograms appear to be pretty darn similar. I think the logistic might be slightly better, but it's difficult to tell just by looking at the.

Question Two part e (8 points)

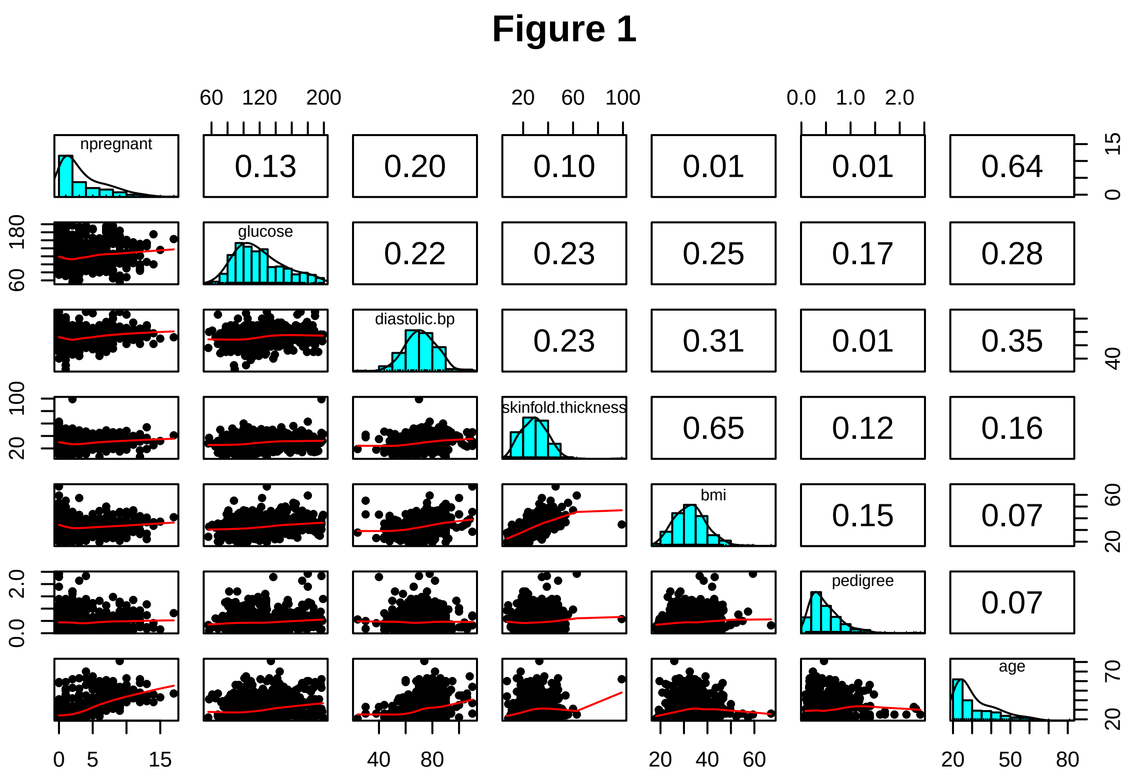
Using the pros and cons of LDA versus QDA versus logistic regression: which procedure do you think is the most appropriate for this data?

Here, are the general rules:

- *MLE is valid under general exponential family assumptions → Logistic is more robust.*
- *If no Gaussian or equal variances the logistic likely better than LDA.*
- *Logistic is less sensitive to outliers.*
- *LDA is more efficient than logistic.*
- *To get "good discrimination" logistic requires larger sample size than LDA.*
- *If the variables seem normally distributed with unequal variances by group then QDA is a good bet.*

Let's check normality and outliers in the variables.

```r
library(psych)
pairs.panels(pima_dat[,1:7], smooth = TRUE, scale = FALSE, density=TRUE,ellipses=FALSE,
digits = 2,method="pearson", pch = 20,lm=FALSE,main="Figure 1",
cor=TRUE,jiggle=FALSE,factor=2,hist.col="cyan",show.points=TRUE,rug=TRUE)
```



Based on the above figures, the normality assumption seems questionable for most of the variables. As a result, logistic regression is a safer choice.

Whether LDA or QDA is more appropriate depends on the homogeneity of the variances. See MANOVA Assumptions (particularly the "multivariate normality" section) for methods of evaluating multivariate *normality and homogeneity of covariance matrices. Recall that for LDA and QDA the b_1 coefficients are robust to normality. If we find normality is suspect we should choose the b_0 coefficient based on the data (e.g., using CV).*

For example, if we wanted to check the homogeneity of the variances we could use:

```
library(heplots)
boxM(pima_dat[, 1:7], pima_dat[, 8])
```

```
##
##  Box's M-test for Homogeneity of Covariance Matrices
##
## data:  pima_dat[, 1:7]
## Chi-Sq (approx.) = 119.27, df = 28, p-value = 3.11e-13
```

This is highly significant indicating there is sufficient evidence that the homogeneity of covariance assumption does not hold. However, since the normality assumption doesn't hold - and QDA is more sensitive to the normality assumption - QDA is not the procedure I would use.

Overall, this points to logistic regression being the best choice.

Question Two part f (3 points)

Perform a leave one out cross-validation to see which method (LDA vs QDA vs Logistic) more accurately predicts the response.

Perfoming a leave-one-out CV is actually pretty simple for LDA and QDA.

```
LDA_loo <- lda(classdigit ~.,data=pima_dat,CV=TRUE)
#Assess the accuracy of the prediction percent correct for each category
ct <- table(pima_dat$class, LDA_loo$class)
ct
```

/	0	1
0	315	40
1	77	100

```
QDA_loo <- qda(classdigit ~.,data=pima_dat,CV=TRUE)
#Assess the accuracy of the prediction percent correct for each category
qda_ct <- table(pima_dat$class, QDA_loo$class)
qda_ct
```

/	0	1
0	301	54
1	72	105

For logistic regression we need to do a little more work.

```
n <- length(pima_dat$classdigit)
logis_loo_class <- NULL
logis_loo_prob <- NULL
for(i in 1:n) {
pima_dat_i <- pima_dat[-i,]
logis_loo_i <- glm(classdigit ~.,data=pima_dat_i,family=binomial)
p.hat <- predict(logis_loo_i,newdata=pima_dat[i,],type="response")
logis_loo_prob <- c(logis_loo_prob,p.hat)
logis_loo_class <- c(logis_loo_class,1*I(p.hat>0.5))
}
#Assess the accuracy of the prediction percent correct for each category
logis_ct <- table(pima_dat$class, logis_loo_class)
logis_ct
```

/logis_loo_class	0	1
0	315	40
1	78	99

```
# total percent correct
data.frame(Method = c("LDA", "QDA", "Logistic"),
           Per.correct =
             c(sum(diag(prop.table(ct))),
               sum(diag(prop.table(logis_ct))),
               sum(diag(prop.table(qda_ct))))
           )
```

Method	Per.correct
LDA	0.7800752
QDA	0.7781955
Logistic	0.7631579

As we can see LDA has the highest number of percent correct, and therefore appears to be the best classifier.

Question Two part g (5 points)

Saying a subject doesn't have diabetes when they actually do is dangerous. Ignored diabetes can lead to complications that include low blood sugar, cardiovascular disease along with damage to the eyes, kidneys and nerves. An expert panel suggested that saying someone doesn't have diabetes when they actually do is 4 times worse than saying they do when they actually don't. Update parts (a-c) using this cost function.

Here's we'll do the same thing as above, but instead of comparing the posterior probabilities to 0.5 *(which assumes a cost ratio of* 1*), we'll compare them to* 0.8 *since* $0.8/0.2 = 4$*. That is, we must think the probability a person* ***does not*** *have diabetes is* 0.8 *to say they don't have diabetes.*

For QDA and LDA I'll use the predicted posterior probabilities from the model fitted in (a) and (b).

```
LDA_pred <- predict(LDA_pima)
#Posterior probability a person doesn't have diabetes
LDA_nodiab_post <- LDA_pred$posterior[,1]
#Predicted class using the cost ratio
LDA_cost_class <- 1-1*I(LDA_nodiab_post>=0.8)
ct_cost <- table(pima_dat$class, LDA_cost_class)
```

Same for QDA.

```
QDA_pred <- predict(QDA_pima)
#Posterior probability a person doesn't have diabetes
QDA_nodiab_post <- QDA_pred$posterior[,1]
#Predicted class using the cost ratio
```

```r
QDA_cost_class <- 1-1*I(QDA_nodiab_post>=0.8)
ct_QDA_cost <- table(pima_dat$class, QDA_cost_class)
```

Now let's do this for logistic. I'll use the predicted logit values from part (c)

```r
#Posterior probability a person doesn't have diabetes
Logis_pred <- 1-exp(logit_hat)/(1+exp(logit_hat))
#Predicted class using the cost ratio
Logis_cost_class <- 1-1*I(Logis_pred>=0.8)
ct_Logis_cost <- table(pima_dat$class, Logis_cost_class)
```

Summarizing the results.

```r
# The number of false positives (said they have diabetes and they didn't)
FPs <- c(ct_cost[2], ct_QDA_cost[2], ct_Logis_cost[2])
# true negatives (said they don't have diabetes when they do.)
TNs <- c(ct_cost[3], ct_QDA_cost[3], ct_Logis_cost[3])

# Expected cost of misclassification using cost function
ECM_cost <- c((4*ct_cost[2]+ct_cost[3])/n,
              (4*ct_QDA_cost[2]+ct_QDA_cost[3])/n,
              (4*ct_Logis_cost[2]+ct_Logis_cost[3])/n)

# Expected cost of misclassification not using cost function
ECM_nocost <- c((4*ct[2]+ct[3])/n,
                (4*qda_ct[2]+qda_ct[3])/n,
                (4*logis_ct[2]+logis_ct[3])/n)
```

Here are the full results:

```r
data.frame(Method = c("LDA", "QDA", "Logistic"),
           FPs = FPs, TNs = TNs,
           ECM_cost = round(ECM_cost,3),
           ECM_nocost = round(ECM_nocost, 3))
```

Method	FPs	TNs	ECM_cost	ECM_nocost
LDA	25	116	0.406	0.654
QDA	29	104	0.414	0.643
Logistic	21	132	0.406	0.662

We'll do this again below (evaluating the ECM properly), but for now the LDA and logistic methods work the best at minimizing the ECM.

Question Two part h (4 points)

Perform a leave one out cross-validation to see which method (LDA vs QDA vs Logistic) leads to a lower expected cost due to misclassification. Use $1 and $4 (as appropriate) as the misclassification costs.

The only thing that we'll change here is the predicted probabilities. For each, we'll use the leave-one-out posterior probabilities calculated in part (f).

```r
#Posterior probability a person doesn't have diabetes
LDA_nodiab_post_loo <- LDA_loo$posterior[,1]
#Predicted class using the cost ratio
```

```r
LDA_cost_class_loo <- 1-1*I(LDA_nodiab_post_loo>=0.8)
ct_cost_loo <- table(pima_dat$class, LDA_cost_class_loo)
ct_cost_loo
```

/LDA_cost_class_loo	0	1
0	237	118
1	26	151

Same for QDA

```r
#Posterior probability a person doesn't have diabetes
QDA_nodiab_post_loo <- QDA_loo$posterior[,1]
#Predicted class using the cost ratio
QDA_cost_class_loo <- 1-1*I(QDA_nodiab_post_loo>=0.8)
ct_QDA_cost_loo <- table(pima_dat$class, QDA_cost_class_loo)
ct_QDA_cost_loo
```

/QDA_cost_class_loo	0	1
0	246	109
1	32	145

Now let's do this for logistic. I'll use the predicted logit values from before.

```r
#Posterior probability a person doesn't have diabetes
Logis_loo_pred <- 1-logis_loo_prob
#Predicted class using the cost ratio
Logis_cost_class_loo <- 1-1*I(Logis_loo_pred>=0.8)
ct_Logis_cost_loo <- table(pima_dat$class, Logis_cost_class_loo)
ct_Logis_cost_loo
```

/Logis_cost_class_loo	0	1
0	221	134
1	25	152

Summarizing the results.

```r
# The number of false positives (said they have diabetes and they didn't)
FPs <- c(ct_cost_loo[2], ct_QDA_cost_loo[2], ct_Logis_cost_loo[2])
# true negatives (said they don't have diabetes when they do.)
TNs <- c(ct_cost_loo[3], ct_QDA_cost_loo[3], ct_Logis_cost_loo[3])

# Expected cost of misclassification using cost function (loo)
ECM_cost <- c((4*ct_cost_loo[2]+ct_cost_loo[3])/n,
              (4*ct_QDA_cost_loo[2]+ct_QDA_cost_loo[3])/n,
              (4*ct_Logis_cost_loo[2]+ct_Logis_cost_loo[3])/n)
# Expected cost of misclassification not using cost function (loo)
ECM_nocost <- c((4*ct[2]+ct[3])/n,
                (4*qda_ct[2]+qda_ct[3])/n,
                (4*logis_ct[2]+logis_ct[3])/n)
```

Here are the full results:

Method	FPs	TNs	ECM_cost	ECM_nocost
LDA	26	118	0.417	0.654
QDA	32	109	0.445	0.643
Logistic	25	134	0.440	0.662

The LDA does the best job at minimizing the cost. Further, when incorporating the cost function we do a better job at minimizing the ECM. For example, in the LDA analysis the ECM was $0.417 when using the cost function and $0.654 when not using the cost function (56% higher).

Question Three

For this problem we'll use the "Communities and Crime Data Set" we used on the previous homework, however, you must re-download the data (see `Communities.xlsx`) as the variables have been streamlined to minimize missing data. The goal is to predict the variable "ViolentCrimesPerPop" (Y in the data) using the first 100 predictors (V1–V100).

Question Three part a (5 points)

Assuming the predictor variables for this data are economic and socio-demographic indicators, discuss whether `local' or`global' structure should be the focus of dimension reduction efforts.

Either answer is acceptable with appropriate reasoning. Here's what I would say for each of the options

- **For local:** *local structure should be the focus of our dimension reduction because there are groups of communities that are similar (in terms of* ***all*** *their economic and socio-demographic indicators) and knowing the data related to these similarities is the most important thing when predicting the violent crime rate.*
- **For global:** *global structure should be the focus of our dimension reduction because there are a few major economic and socio-demographic trends that really matter when predicting the violent crime rate. We don't need to keep data close based on* ***all*** *the predictor variables, only those the predictor variables that really matter. As a result, we should use global methods.*

Overall, I think global is a better answer but this is subjective. I really could see the reasoning either way.

Question Three part b (9 points)

Apply 3 of the dimension reduction techniques to the predictor variables.

Here, I'm going to apply PCA, MDS, Isomap, UMAP, and t-SNE.

```r
library(dplyr)
library(tidyverse)
library(MASS)
library(readxl)
library(vegan)
library(Rtsne)
library(umap)
# Load the dataset
communities <- read_excel("Communities.xlsx") %>% na.omit()

set.seed(8329)
### PCA
# Extract predictor variables
X <- communities[, 1:100]

# Perform PCA
pca_result <- prcomp(X, center = TRUE, scale. = TRUE)

# Get the first few principal components
pca_data <- pca_result$x[, 1:2]  # first 2 PCs


### MDS
# Calculate the dissimilarity matrix
dissimilarity_matrix <- dist(X)
# Perform classical MDS
```

```r
mds_result <- cmdscale(dissimilarity_matrix, eig = TRUE, k = 2)  # k is the number of dimensions

### Isomap

# Perform Isomap
isomap_result <- isomap(dissimilarity_matrix, k = 3)  # ndim is the number of dimensions

# Extract the coordinates
isomap_coords <- as.matrix(isomap_result$points)

# t-SNE (I tried some different perplexities)
tsne_results50_0.5 <- Rtsne(X, dims = 2, perplexity = 50, theta = 0.5, verbose = TRUE)
```

```
## Performing PCA
## Read the 1993 x 50 data matrix successfully!
## Using no_dims = 2, perplexity = 50.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.31 seconds (sparsity = 0.107344)!
## Learning embedding...
## Iteration 50: error is 71.699391 (50 iterations in 0.20 seconds)
## Iteration 100: error is 69.265118 (50 iterations in 0.22 seconds)
## Iteration 150: error is 69.014558 (50 iterations in 0.22 seconds)
## Iteration 200: error is 68.872197 (50 iterations in 0.19 seconds)
## Iteration 250: error is 68.871220 (50 iterations in 0.19 seconds)
## Iteration 300: error is 1.504908 (50 iterations in 0.19 seconds)
## Iteration 350: error is 1.331713 (50 iterations in 0.21 seconds)
## Iteration 400: error is 1.270831 (50 iterations in 0.19 seconds)
## Iteration 450: error is 1.252173 (50 iterations in 0.18 seconds)
## Iteration 500: error is 1.242067 (50 iterations in 0.18 seconds)
## Iteration 550: error is 1.235660 (50 iterations in 0.18 seconds)
## Iteration 600: error is 1.230022 (50 iterations in 0.18 seconds)
## Iteration 650: error is 1.225741 (50 iterations in 0.18 seconds)
## Iteration 700: error is 1.221907 (50 iterations in 0.18 seconds)
## Iteration 750: error is 1.218860 (50 iterations in 0.18 seconds)
## Iteration 800: error is 1.215441 (50 iterations in 0.18 seconds)
## Iteration 850: error is 1.212122 (50 iterations in 0.18 seconds)
## Iteration 900: error is 1.209755 (50 iterations in 0.18 seconds)
## Iteration 950: error is 1.207145 (50 iterations in 0.18 seconds)
## Iteration 1000: error is 1.205330 (50 iterations in 0.18 seconds)
## Fitting performed in 3.76 seconds.
```

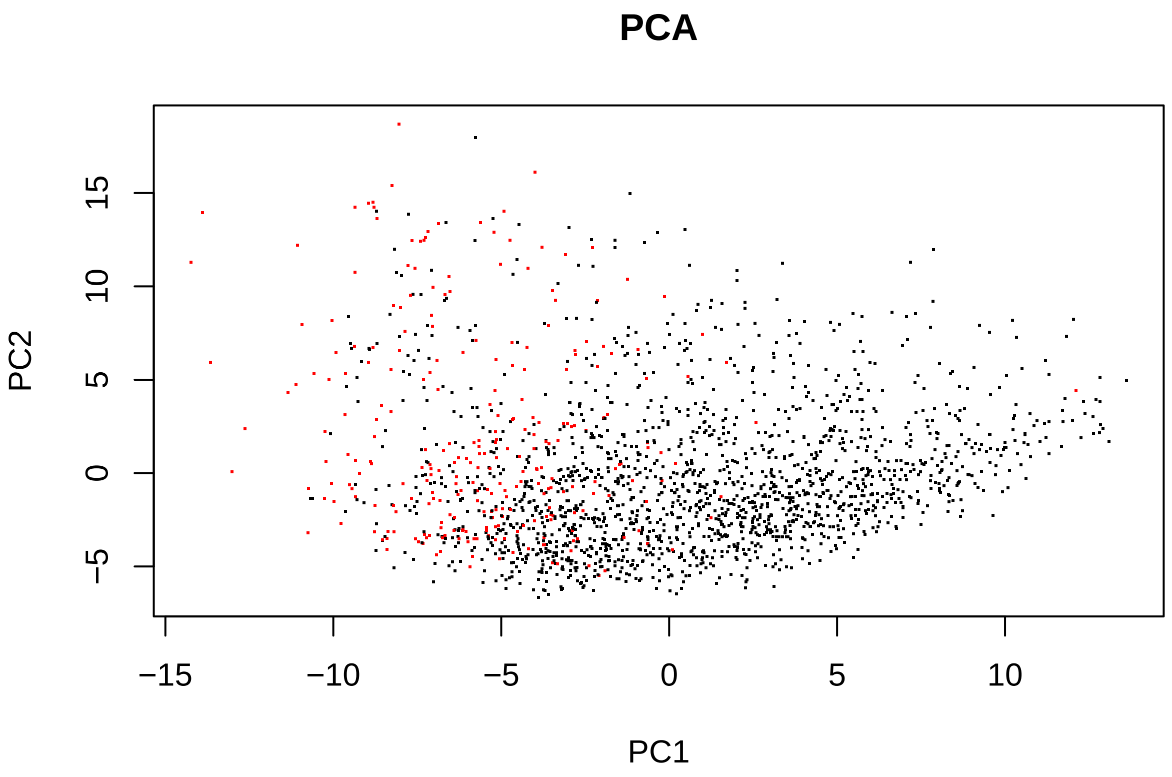
```r
# UMAP
umap_result <- umap(X)
```

Question Three part c (3 points)

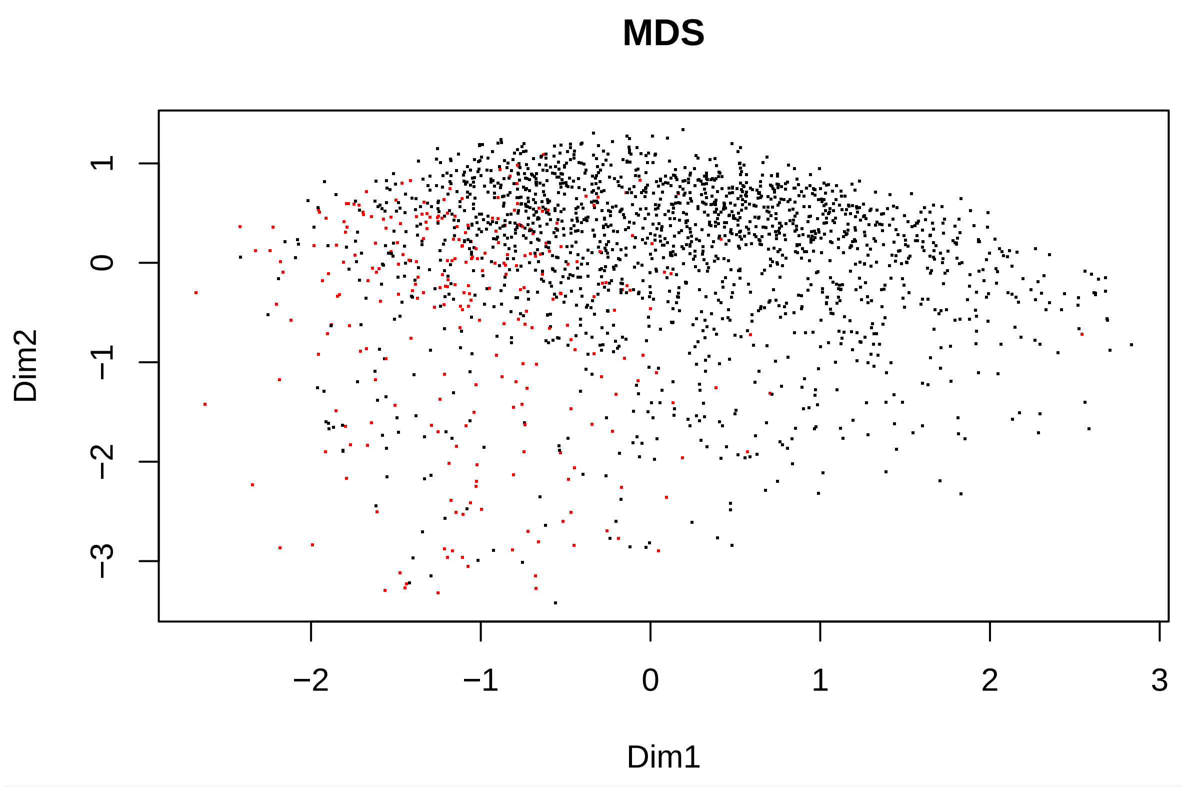
Plot the first two coordinates of the methods applied in the previous question on separate plots (3 plots total). Make the color of the points equal to red if $Y > 0.5$ and black if $Y \leq 0.5$.

```r
# Plot PCA
# Create a color variable based on Y
color_var <- ifelse(communities$Y > 0.5, "red", "black")

# PCA Plot
```

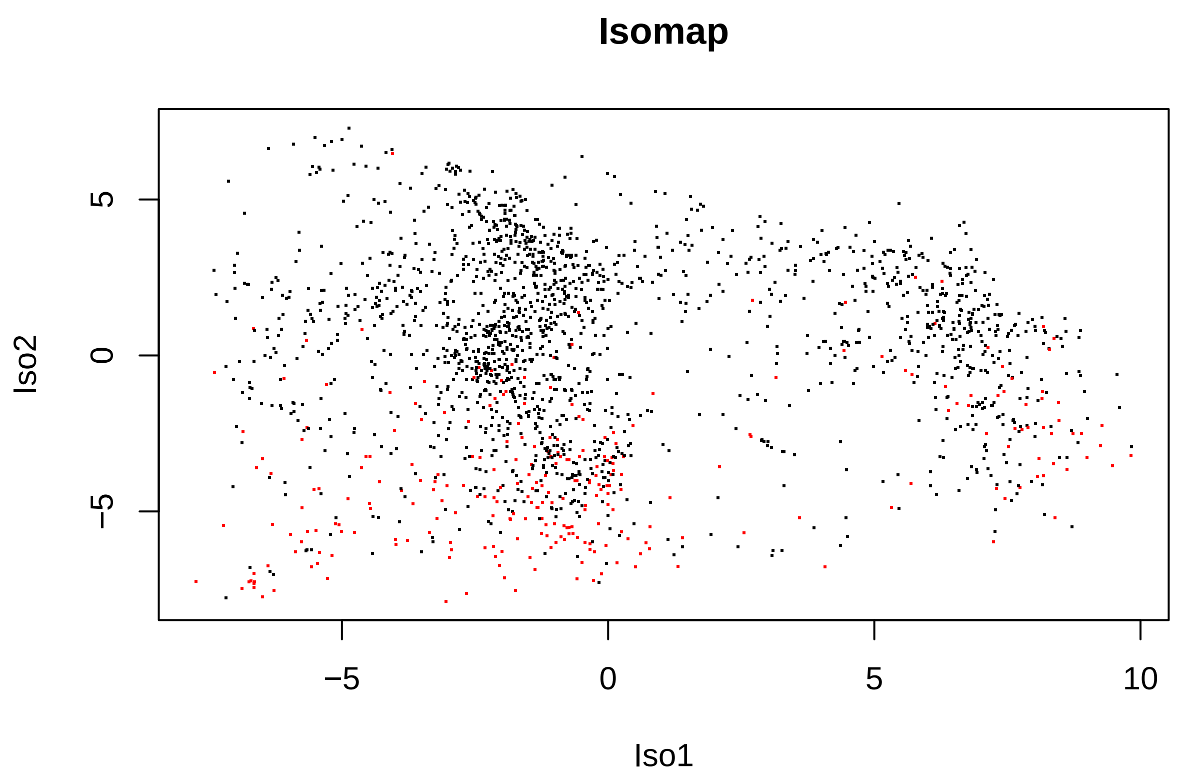
```
plot(pca_data[,1], pca_data[,2], col=color_var, pch = ".",
     xlab="PC1", ylab="PC2", main="PCA")
```



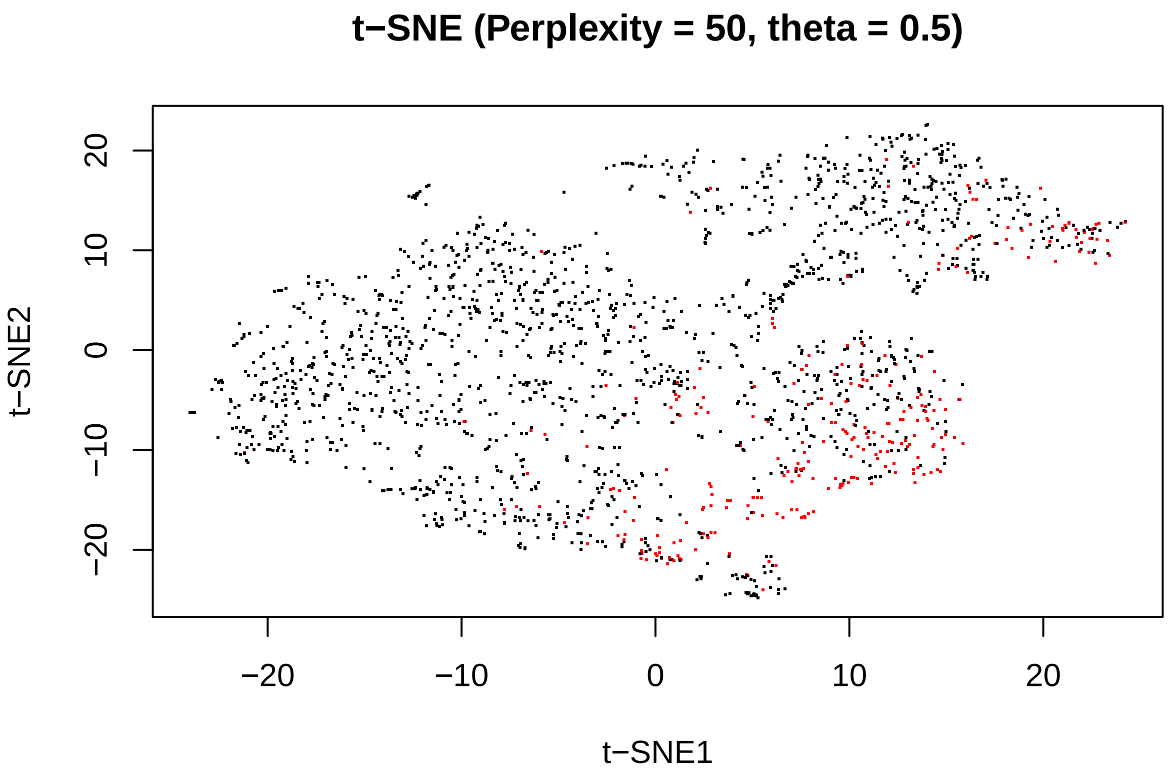
```
# MDS Plot
plot(mds_result$points, col = color_var, pch = ".",
     xlab="Dim1", ylab="Dim2", main="MDS")
```



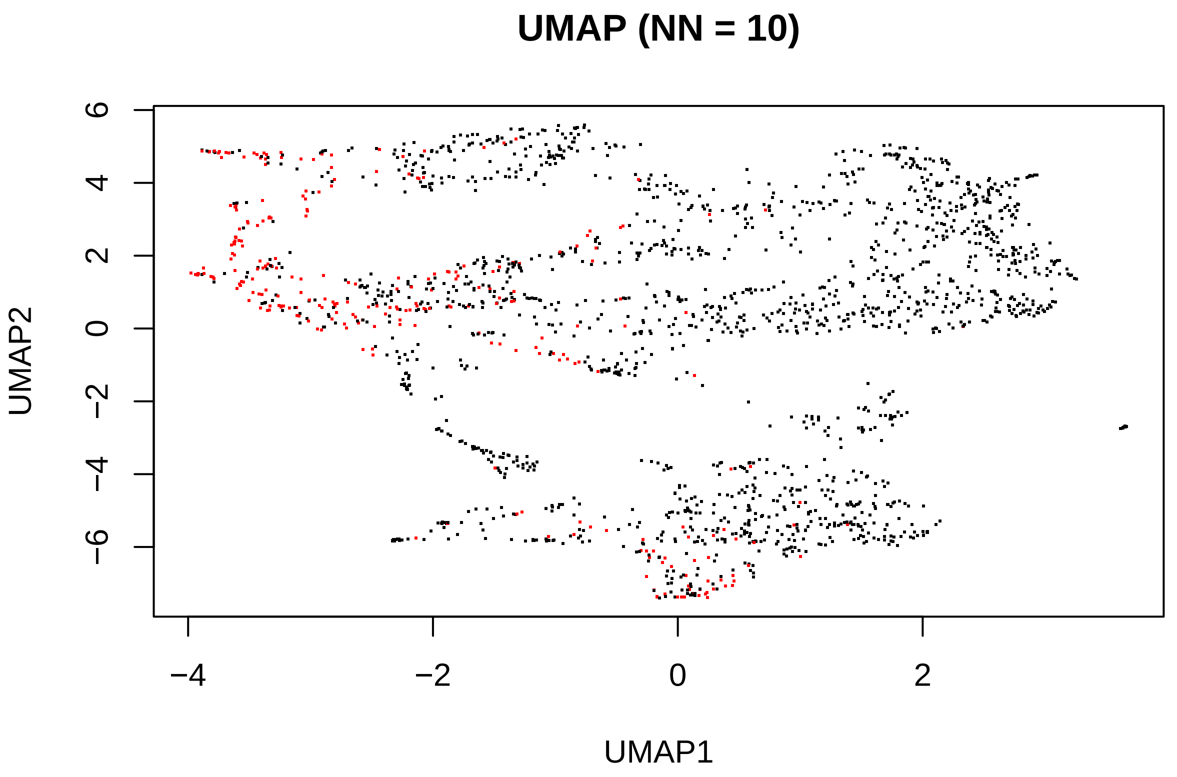
```r
# Isomap Plot
plot(isomap_result$points, col = color_var, pch = ".",
     xlab="Iso1", ylab="Iso2", main="Isomap")
```



```r
# t-SNE
plot(tsne_results50_0.5$Y, col = color_var, pch = ".",
     xlab="t-SNE1", ylab="t-SNE2",
     main = "t-SNE (Perplexity = 50, theta = 0.5)")
```



```
# UMAP Plot
plot(umap_result$layout, col = color_var, pch = ".",
    xlab="UMAP1", ylab="UMAP2", main = "UMAP (NN = 10)")
```



Question Three part d (3 points)

Which of the dimension reduction techniques appears to separate $Y > 0.5$ vs $Y \leq 0.5$ most effectively?

In my view, it appears that Isomap, and UMAP are the most effective and PCA or MDS are the worst. More specifically, the second component of Isomap ($Iso2$) seems to do the best job out of anything that is plotted here ($UMAP1$ is probably second best). It seems that you could classify anything with $Iso2 < -4$ as a "violent community" ($Y > 0.5$) and be pretty accurate. However, the top left portion of the UMAP graph seems to be the area with the most densely populated with red.

Question Three part e (5 points)

Which dimension reduction technique would you use to classify $Y > 0.5$ vs $Y \leq 0.5$? Use the pros and cons discussed in class along with your answer to the previous question.

The real benefit of PCA is that it's linear and you can transform parameters (i.e., β's) back to the original scale. However, if it doesn't appear to be an accurate method then you shouldn't use it. As I said above, I think it appears that PCA does not perform well in terms of separating the classes.

So, with PCA off the list of options, all of the other methods are non-linear. Among this group I would go with UMAP for the following reasons: a) this data set could be one where global or local structure should be focused on, since UMAP can capture both it seems like a safe bet, b) I did try out some different numbers of nearest neighbors and the results were similar, and c) it seems to have the best separation between the two classes.