Take Home Final

Yigao Li

May 3, 2018

Part A: Ozone Data

Preparation

```
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 3.4.4
```

```
library(tree)
```

```
## Warning: package 'tree' was built under R version 3.4.4
```

```
library(boot)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.3
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.4.3
```

```
## Loaded glmnet 2.0-13
```

```
data(Ozone)
ozone <- Ozone
ozone <- ozone[,-c(2,3)]
colnames(ozone) <- c("month", "dailymax", "pressure", "windlax", "humlax", "tsand", "telmonte",
                     "invlax", "pressgrad", "invbasetemplax", "vislax")
ozone.clean <- na.omit(ozone)
ozone.dailymax <- ozone[!is.na(ozone$dailymax),]
```

A.1

```
month.table <- table(ozone.clean$month)
miss.month <- c(31,29,31,30,31,30,31,31,30,31,30,31) - month.table
chisq.test(miss.month)
```

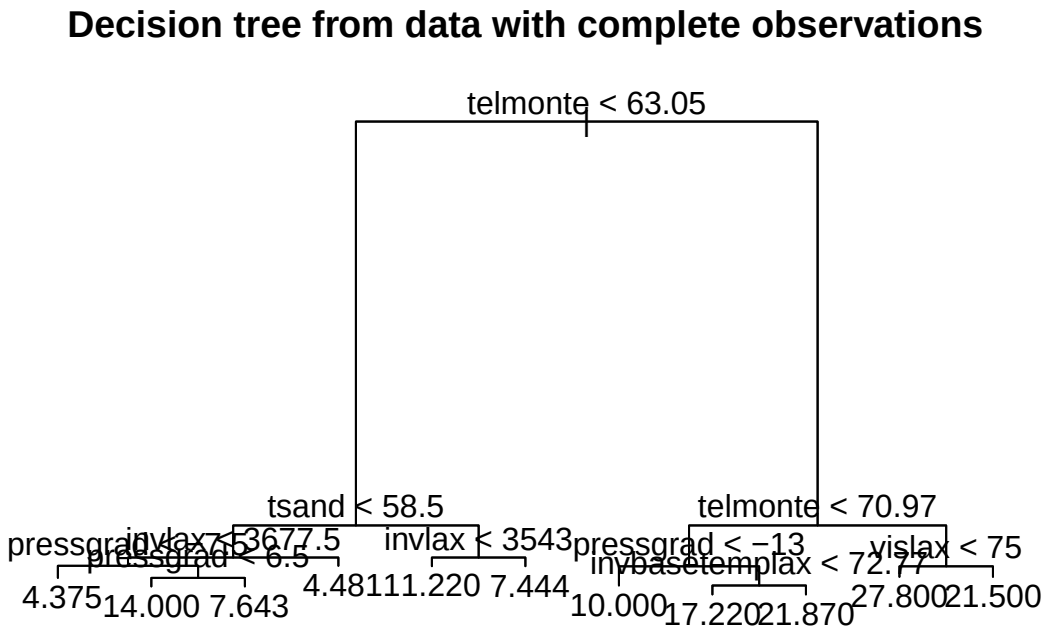
```
##
##  Chi-squared test for given probabilities
##
## data:  miss.month
## X-squared = 9.638, df = 11, p-value = 0.5632
```

Convert the month column to a contigency table. The data has 366 rows, which means that February has 29 days this year. By subtracting the total number of days in a month by its corresponding days in contigency table, we get the number of missing days in the data. Then use built-in `chisq.test` to test our hypothesis. The null hypothesis is that the days with missing data are uniformly distributed and alternative hypothesis is

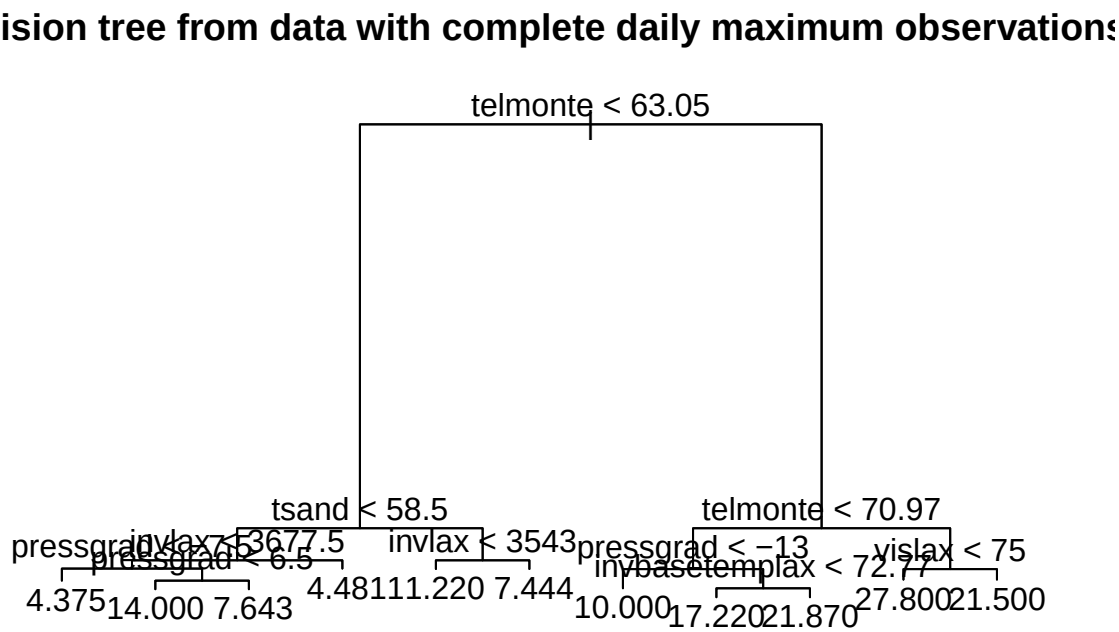
that the number of missing days in at least one month is different from other months. The χ^2 test statistics is 9.638 with degrees of freedom 11. The p-value is 0.5632, which is larger than 5%. We fail to reject our null hypothesis. Therefore, the days with missing data are uniformly distributed.

A.2

```
tree.clean <- tree(dailymax ~ . - month, data = ozone.clean)
plot(tree.clean)
text(tree.clean)
title("Decision tree from data with complete observations")
```




```
tree.dailymax <- tree(dailymax ~ . - month, data = ozone.dailymax)
plot(tree.dailymax)
text(tree.dailymax)
title("Decision tree from data with complete daily maximum observations")
```



According to the instruction of the package `tree`, function `tree` takes parameter `na.action` equals to `na.pass` by default, which means "to do nothing ... by dropping them down the tree as far as possible" (tree, 14). Therefore, when applying to data with missing values, it passes through these `na`s.

A.3

```
set.seed(6007)
n <- dim(ozone.dailymax)[1]
nfold <- 10
fold.index <- sample(rep(1:nfold, length.out = n))
se <- 0
for (i in 1:nfold){
  test <- which(fold.index == i)
  train <- -test
  train.data <- ozone.dailymax[train,]
  test.data <- ozone.dailymax[test,]
  month.mean <- c()
  for (j in 1:12){
    month.mean[j] <- mean(train.data$dailymax[train.data$month == j])
  }
  se <- se + sum((test.data$dailymax - month.mean[test.data$month])^2)
}
sqrt(se/n)
```

```
## [1] 6.3997
```

I wrote my own code for cross-validation. The dataset I chose was the complete daily maximum observations. I got errors of all 361 observations through 10-fold CV and then calculate root mean square error of all. The RMS error is 6.3997.

A.4

```
set.seed(7758)
ozone.linear <- glm(dailymax ~ . - month, data = ozone.clean)
summary(ozone.linear)
```

```
##
## Call:
## glm(formula = dailymax ~ . - month, data = ozone.clean)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -11.2055   -2.7232   -0.2741    3.0891   13.3442
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    59.9517553 38.3286940   1.564 0.119421
## pressure       -0.0139111  0.0072511  -1.918 0.056527 .
## windlax         0.0276862  0.1741433   0.159 0.873847
## humlax          0.0808740  0.0237694   3.402 0.000812 ***
## tsand           0.1503404  0.0692994   2.169 0.031272 *
## telmonte        0.5253439  0.1247136   4.212 3.87e-05 ***
## invlax         -0.0010052  0.0003944  -2.549 0.011586 *
## pressgrad       0.0049796  0.0147772   0.337 0.736501
## invbasetemplax -0.1543882  0.1192917  -1.294 0.197140
## vislax         -0.0033951  0.0048963  -0.693 0.488883
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 20.08543)
##
##     Null deviance: 13549.5  on 202  degrees of freedom
## Residual deviance:  3876.5  on 193  degrees of freedom
## AIC: 1196.8
##
## Number of Fisher Scoring iterations: 2
```

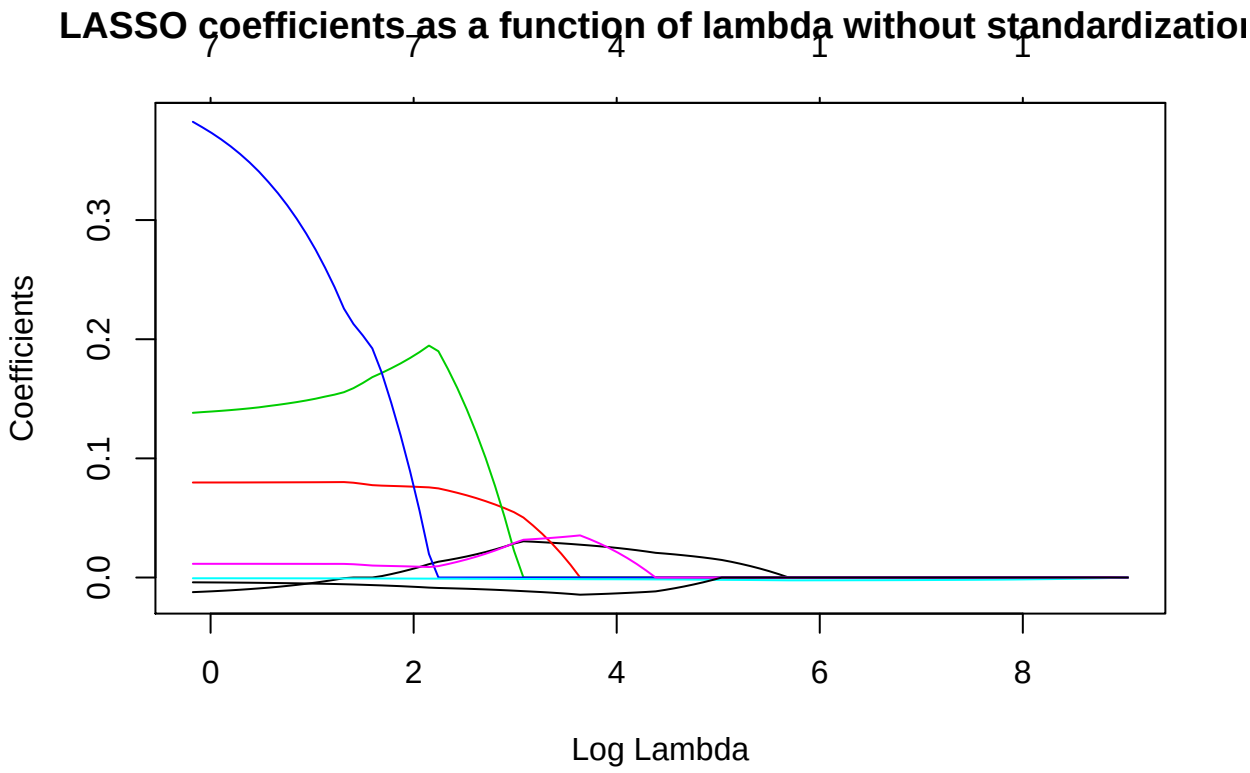
```
cv.1 <- cv.glm(data = ozone.clean, glmfit = ozone.linear, K = 10)
sqrt(cv.1$delta[1])
```

```
## [1] 4.589724
```

The dataset is the one with complete observations. Using `glm` to build linear model to predict daily ozone maximum from all other variables except `month`, and then apply `cv.glm` to do 10-fold CV and find mean square error `delta` from `cv.glm` output. Significant variables are `humlax`, `tsand`, `telmonte` and `invlax`. Root mean square error can be calculated by taking square root of MSE. The RMS error is 4.589724.

A.5

```
X <- model.matrix(dailymax ~ . - month, data = ozone.clean)
ozone.unscale.lasso <- glmnet(X, ozone.clean$dailymax, alpha = 1, standardize = FALSE)
plot(ozone.unscale.lasso, xvar = "lambda",
     main = "LASSO coefficients as a function of lambda without standardization")
```




```
ozone.unscale.lasso$beta[,36:59]
```

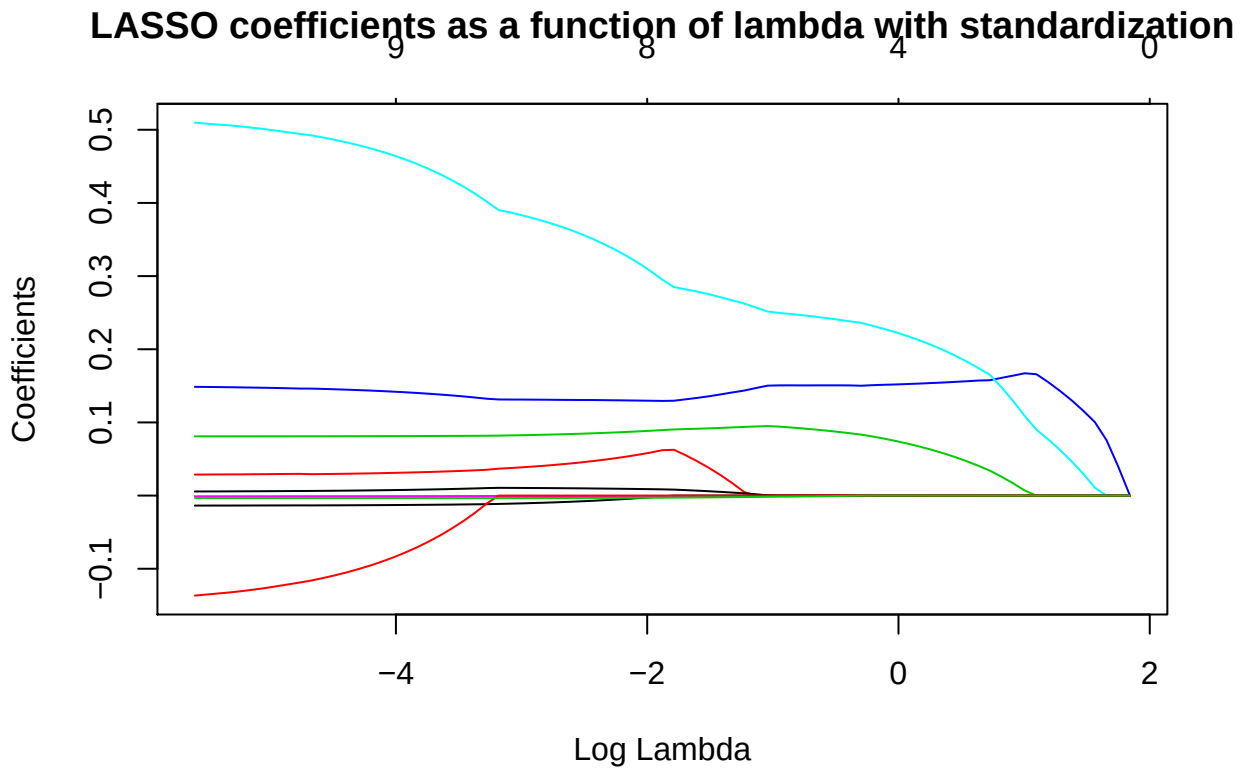
```
## 10 x 24 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 24 column names 's35', 's36', 's37' ... ]]
##
## (Intercept)     .            .             .            .
## pressure        .            0.0002284664  0.002923610  0.005378789
## windlax         .            .             .            .
## humlax          .            .             .            .
## tsand           .            .             .            .
## telmonte        .            .             .            .
## invlax         -0.002345223 -0.0023465071 -0.002270762 -0.002201763
## pressgrad       .            .             .            .
## invbasetemplax  .            .             .            .
## vislax          .            .             .            .
##
## (Intercept)     .            .             .            .
## pressure        0.007615856  0.009654189   0.011511441  0.013203701
```

```
## windlax         .            .            .            .
## humlax          .            .            .            .
## tsand           .            .            .            .
## telmonte        .            .            .            .
## invlax         -0.002138894 -0.002081610 -0.002029414 -0.001981856
## pressgrad       .            .            .            .
## invbasetemplax  .            .            .            .
## vislax          .            .            .            .
##
## (Intercept)     .            .            .            .
## pressure        0.014745624  0.015870048  0.016881059  0.017801023
## windlax         .            .            .            .
## humlax          .            .            .            .
## tsand           .            .            .            .
## telmonte        .            .            .            .
## invlax         -0.001938522 -0.001871588 -0.001809739 -0.001753375
## pressgrad       .            .            .            .
## invbasetemplax  .            .            .            .
## vislax          .           -0.002083967 -0.004061405 -0.005863841
##
## (Intercept)     .            .            .            .
## pressure        0.018639253  0.019403017  0.020098930  0.020733020
## windlax         .            .            .            .
## humlax          .            .            .            .
## tsand           .            .            .            .
## telmonte        .            .            .            .
## invlax         -0.001702019 -0.001655225 -0.001612588 -0.001573739
## pressgrad       .            .            .            .
## invbasetemplax  .            .            .            .
## vislax         -0.007506154 -0.009002568 -0.010366046 -0.011608395
##
## (Intercept)     .            .            .            .
## pressure        0.021872288  0.022928014  0.023886930  0.024763417
## windlax         .            .            .            .
## humlax          .            .            .            .
## tsand           .            .            .            .
## telmonte        .            .            .            .
## invlax         -0.001545146 -0.001519123 -0.001495567 -0.001473959
## pressgrad       0.005854453  0.011337948  0.016333768  0.020886187
## invbasetemplax  .            .            .            .
## vislax         -0.012096094 -0.012523740 -0.012913464 -0.013268543
##
## (Intercept)     .           .           .            .
## pressure        0.02556204  0.02628971  0.026952744  2.755687e-02
## windlax         .           .           .            .
## humlax          .           .           .            5.711783e-05
## tsand           .           .           .            .
## telmonte        .           .           .            .
## invlax         -0.00145427 -0.00143633 -0.001419984 -1.404932e-03
## pressgrad       0.02503418  0.02881368  0.032257411  3.537382e-02
## invbasetemplax  .           .           .            .
## vislax         -0.01359208 -0.01388687 -0.014155478 -1.439610e-02
```

The plot of the coefficient trajectories as a function of the regularization parameter λ with nonstandardized

data is shown above. From the matrix of coefficients given by `glmnet()$beta`, the last five variables that leave the model as λ increases are `invlax`, `pressure`, `vislax`, `pressgrad` and `humlax`.

```
ozone.scale.lasso <- glmnet(X, ozone.clean$dailymax, alpha = 1, standardize = TRUE)
plot(ozone.scale.lasso, xvar = "lambda",
     main = "LASSO coefficients as a function of lambda with standardization")
```




```
ozone.scale.lasso$beta[,3:24]
```

```
## 10 x 22 sparse Matrix of class "dgCMatrix"
##    [[ suppressing 22 column names 's2', 's3', 's4' ... ]]
##
## (Intercept)    .          .          .          .          .
## pressure       .          .          .          .          .
## windlax        .          .          .          .          .
## humlax         .          .          .          .          .
## tsand          0.07560921 0.10039753 0.11602657 0.13033973 0.14325928
## telmonte       .          0.01081845 0.02974332 0.04690673 0.06268036
## invlax         .          .          .          .          .
## pressgrad      .          .          .          .          .
## invbasetemplax .          .          .          .          .
## vislax         .          .          .          .          .
##
## (Intercept)    .          .          .          .          .
## pressure       .          .          .          .          .
## windlax        .          .          .          .          .
```

```
## humlax          .          .          0.006916519 0.01686534 0.02594493
## tsand           0.15515163 0.16586805 0.167160433 0.16392946 0.16086044
## telmonte        0.07691942 0.09002559 0.108737034 0.12929096 0.14815175
## invlax          .          .          .           .          .
## pressgrad       .          .          .           .          .
## invbasetemplax  .          .          .           .          .
## vislax          .          .          .           .          .
##
## (Intercept)     .             .             .             .
## pressure        .             .             .             .
## windlax         .             .             .             .
## humlax          3.420772e-02  4.096735e-02  4.719470e-02  0.0528609755
## tsand           1.576589e-01  1.572267e-01  1.562352e-01  0.1553971026
## telmonte        1.653610e-01  1.745122e-01  1.835251e-01  0.1916654853
## invlax         -4.782540e-06 -5.322749e-05 -9.706349e-05 -0.0001370331
## pressgrad       .             .             .             .
## invbasetemplax  .             .             .             .
## vislax          .             .             .             .
##
## (Intercept)     .             .             .             .
## pressure        .             .             .             .
## windlax         .             .             .             .
## humlax          0.0580379616  0.0627416828  0.0670406865  0.0709446698
## tsand           0.1545163506  0.1538248402  0.1530854893  0.1525207510
## telmonte        0.1992116428  0.2059651748  0.2122390867  0.2178356806
## invlax         -0.0001734019 -0.0002065872 -0.0002367778 -0.0002643328
## pressgrad       .             .             .             .
## invbasetemplax  .             .             .             .
## vislax          .             .             .             .
##
## (Intercept)     .             .             .             .
## pressure        .             .             .             .
## windlax         .             .             .             .
## humlax          0.0745146082  0.0777545581  0.0807191159  8.332365e-02
## tsand           0.1519000324  0.1514411578  0.1509197452  1.501503e-01
## telmonte        0.2230519871  0.2276874061  0.2320247700  2.360960e-01
## invlax         -0.0002893946 -0.0003122754 -0.0003330796 -3.513653e-04
## pressgrad       .             .             .             .
## invbasetemplax  .             .             .             .
## vislax          .             .             .            -9.710811e-05
```

Similarly, make the same kind of plot for scaled data. From the matrix of coefficients, the 5 most important variables are `tsand`, `telmonte`, `humlax`, `invlax` and `vislax`.

A.6

Model	pressure	windlax	humlax	tsand	telmonte	invlax	pressgrad	invbasetemplax	vislax
Decision Tree (**A.2**)				✓	✓	✓	✓	✓	✓
Linear (**A.4**)			✓	✓	✓	✓			
LASSO (unscale) (**A.5**)	✓		✓			✓	✓		✓

Model	pressure	windlax	humlax	tsand	telmonte	invlax	pressgrad	invbasetemplax	vislax
LASSO (scale) (**A.5**)			✓	✓	✓	✓			✓

The above table displays variables that decision tree, linear model and LASSO use. We can see that `invlax` is used in all models, and `humlax`, `tsand`, `telmonte` and `vislax` appear in most models. `windlax` does not appear in any model and `pressure` and `invbasetemplax` are only used in a few models.

Part B: Artificial Data

Preparation

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.4.4
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:boot':
##
##     aml
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

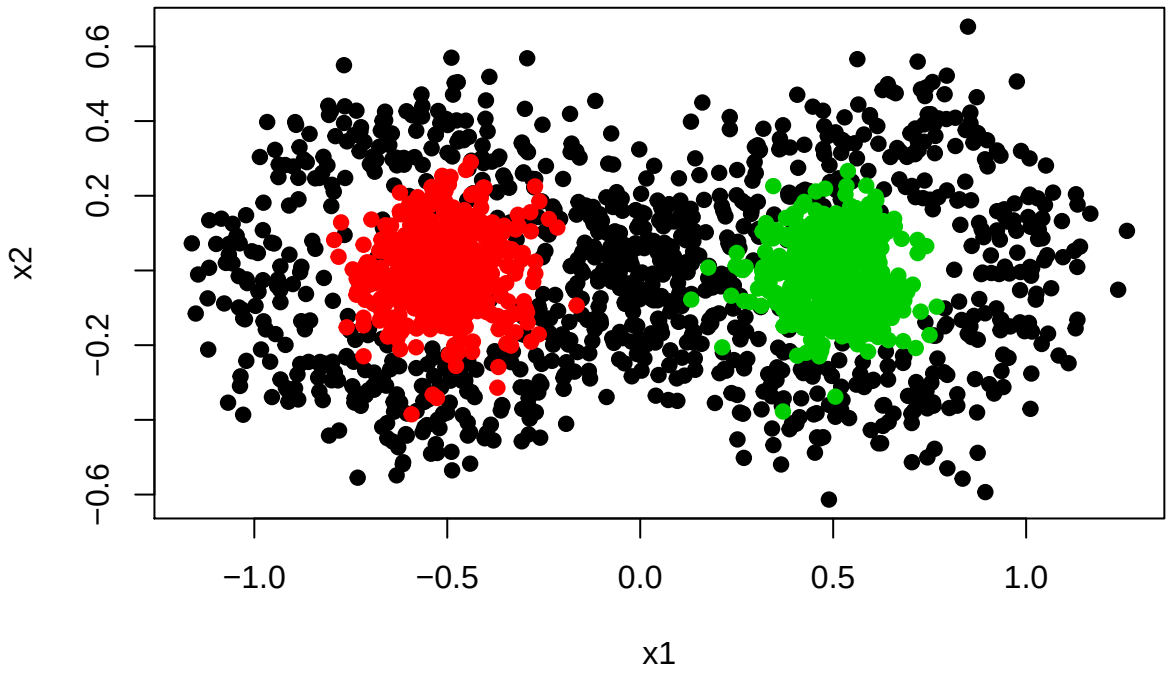
```
## Loading required package: splines
```

```
## Loading required package: parallel
```

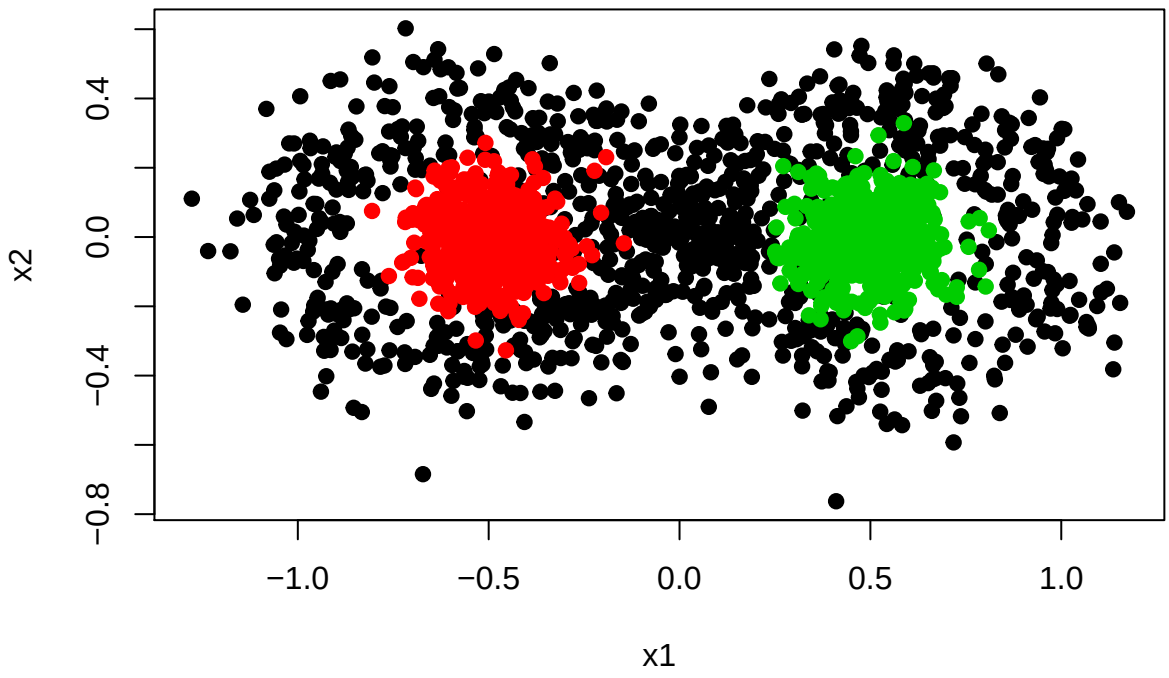
```
## Loaded gbm 2.1.3
```

```
set.seed(2019)
make.eight = function(N, spread = .1, makeplot = T){
  # Make N points: N/2 points in horizontal figure 8
  # N/4 points each inside the holes of the figure 8
  # spread = noise parameter
  # return data frame with coordinates x, y for each point
  # Classification variables in the data frame:
  # charlabel = eight or left or right
  # label = 0 (for points on the figure 8) or = 1 (for points inside the holes)
  # plot with marked points if makeplot = T
  # Try these examples:
  # mydf <- make.eight(200)
  # mydf <- make.eight(100,spread = .05)
  # mydf <- make.eight(300,spread = .1, makeplot = F)
  circ0 = runif(N/2)*2*pi
```

```
  circ = matrix(c(cos(circ0)/(1 + sin(circ0)^2),rep(-.5,N/4),rep(.5,N/4),
  cos(circ0)*sin(circ0)/(1 + sin(circ0)^2),rep(0,N/2)),ncol = 2)
  x = circ + spread*matrix(rnorm(2*N),N,2)
  y=rep(c(0,1),c(N/2,N/2))
  if(makeplot){plot(x,col = c(rep(1,N/2),rep(2,N/4),rep(3,N/4)),pch=19,
  xlab = "x1", ylab = "x2")}
  A = data.frame(x = x[,1], y = x[,2], label = as.factor(y),
  charlabel = c(rep("eight",N/2),rep("left",N/4), rep("right",N/4)))
  return(A)
}
eight.train <- make.eight(2000)
```



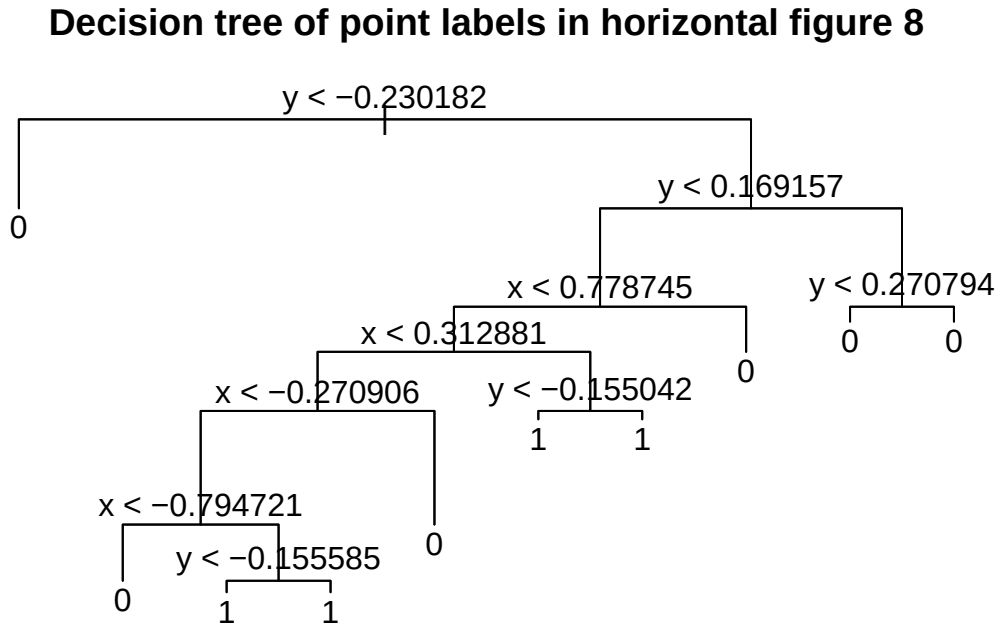

```
eight.test <- make.eight(2000)
```



B.1

(a)

```
eight.tree <- tree(label ~ x + y, data = eight.train)
plot(eight.tree)
text(eight.tree)
title("Decision tree of point labels in horizontal figure 8")
```




```
pred.tree.train <- predict(eight.tree, newdata = eight.train, type = "class")
table(eight.train$label, pred.tree.train)
```

```
##    pred.tree.train
##       0   1
##   0 936  64
##   1  60 940
```

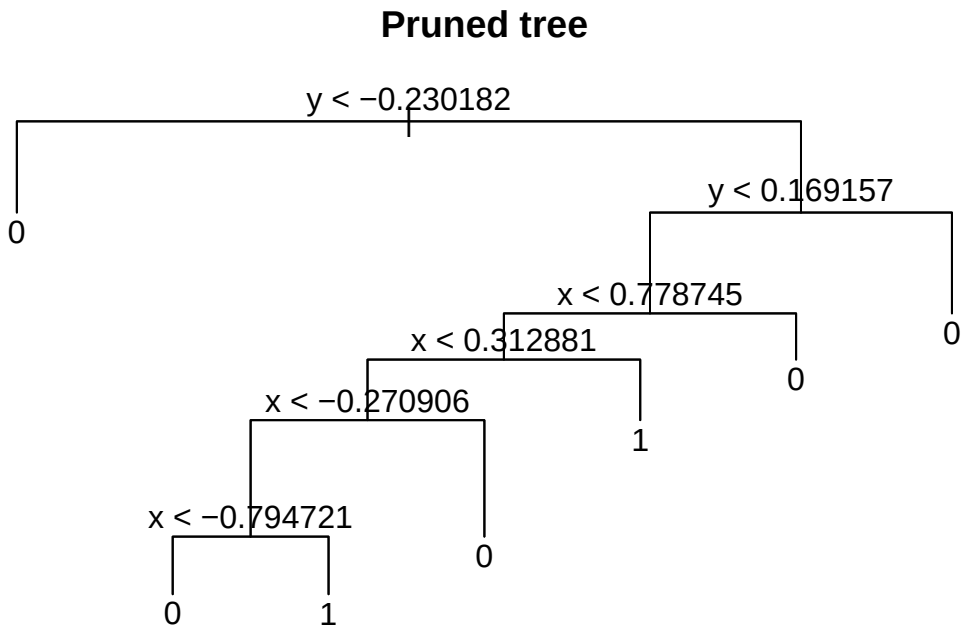
```
pred.tree.test <- predict(eight.tree, newdata = eight.test, type = "class")
table(eight.test$label, pred.tree.test)
```

```
##    pred.tree.test
##       0   1
##   0 912  88
##   1  72 928
```

Training error is 6.2% and testing error is 8.0%.

(b)

```
eight.prune <- prune.tree(eight.tree, best = 7)
plot(eight.prune)
text(eight.prune)
title("Pruned tree")
```




```
pred.prune.train <- predict(eight.prune, newdata = eight.train, type = "class")
table(eight.train$label, pred.prune.train)
```

```
##    pred.prune.train
##       0   1
##   0 936  64
##   1  60 940
```

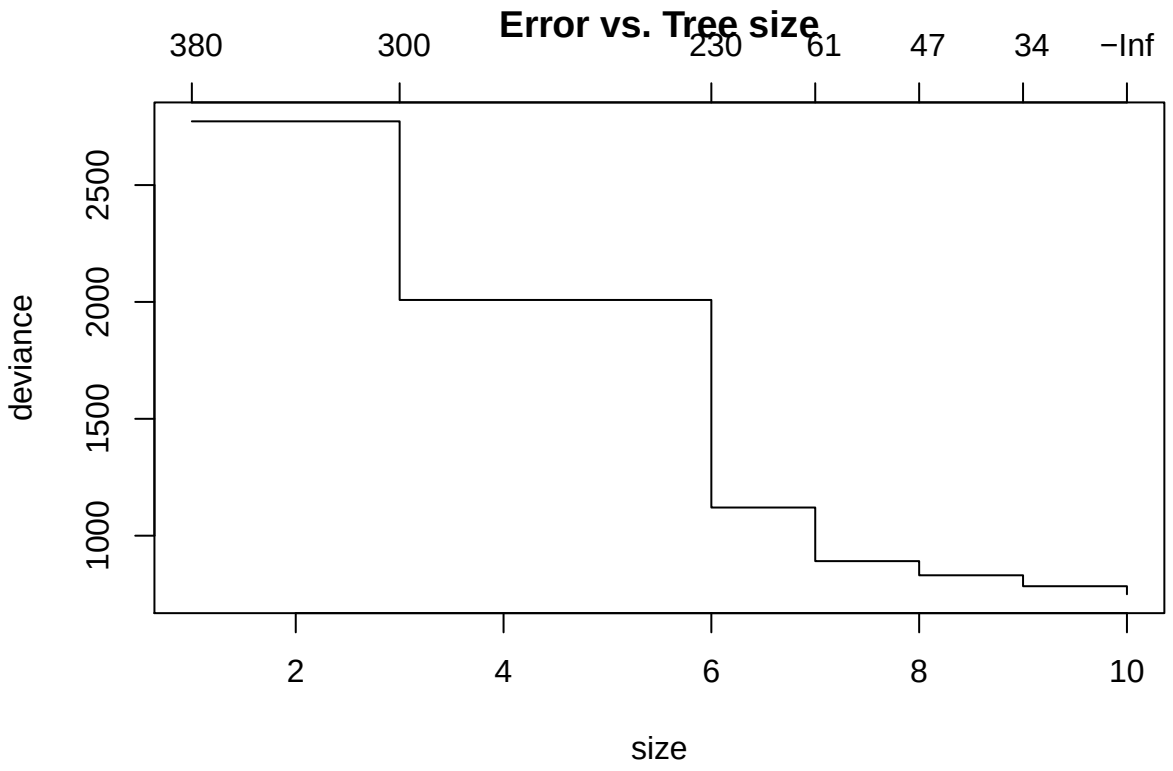
```
pred.prune.test <- predict(eight.prune, newdata = eight.test, type = "class")
table(eight.test$label, pred.prune.test)
```

```
##    pred.prune.test
##       0   1
##   0 912  88
##   1  72 928
```

Original tree has 10 terminal nodes and there are 3 subtrees that return the same result for both branches. Therefore, pruned tree uses only 7 terminal nodes. After building the pruned tree and the confusion matrices of predictions.

(c)

```
eight.fewer6 <- prune.tree(eight.tree)
plot(eight.fewer6)
title("Error vs. Tree size")
```



As tree size decreases, the error rate increases.

B.2

(a)

```
eight.train$label <- as.numeric(eight.train$label)-1
set.seed(2019)
eight.boost <- gbm(label ~ x + y, distribution = "bernoulli", data = eight.train, n.trees = 20000,
                   interaction.depth = 2, shrinkage = 0.01)
pred.boost.train <- predict(eight.boost, eight.train, n.trees = 20000, response = "class")
table(eight.train$label, pred.boost.train > 0.5)
```

```
##
##     FALSE TRUE
##   0   998    2
##   1    18  982
```

```
pred.boost.test <- predict(eight.boost, eight.test, n.trees = 20000, response = "class")
table(eight.test$label, pred.boost.test > 0.5)
```

```
##
##     FALSE TRUE
##   0   917   83
##   1    72  928
```

In the boosted model, training error rate is 1.00% and test error rate is 7.75%. Prediction error cannot be lower than 1% because boosting training error converges. Since the number of trees is set to 20000 to achieve small training error, this large number of trees causes overfitting and significantly large prediction error in test data.

(b)

```
set.seed(2019)
eight.d3 <- gbm(label ~ x + y, distribution = "bernoulli", data = eight.train, n.trees = 1000,
                interaction.depth = 8, shrinkage = 0.2)
pred.d3.train <- predict(eight.d3, eight.train, n.trees = 1000, response = "class")
table(eight.train$label, pred.d3.train > 0.5)
```

```
##
##     FALSE TRUE
##   0  1000    0
##   1     0 1000
```

```
pred.d3.test <- predict(eight.d3, eight.test, n.trees = 1000, response = "class")
table(eight.test$label, pred.d3.test > 0.5)
```

```
##
##     FALSE TRUE
##   0   923   77
##   1    66  934
```

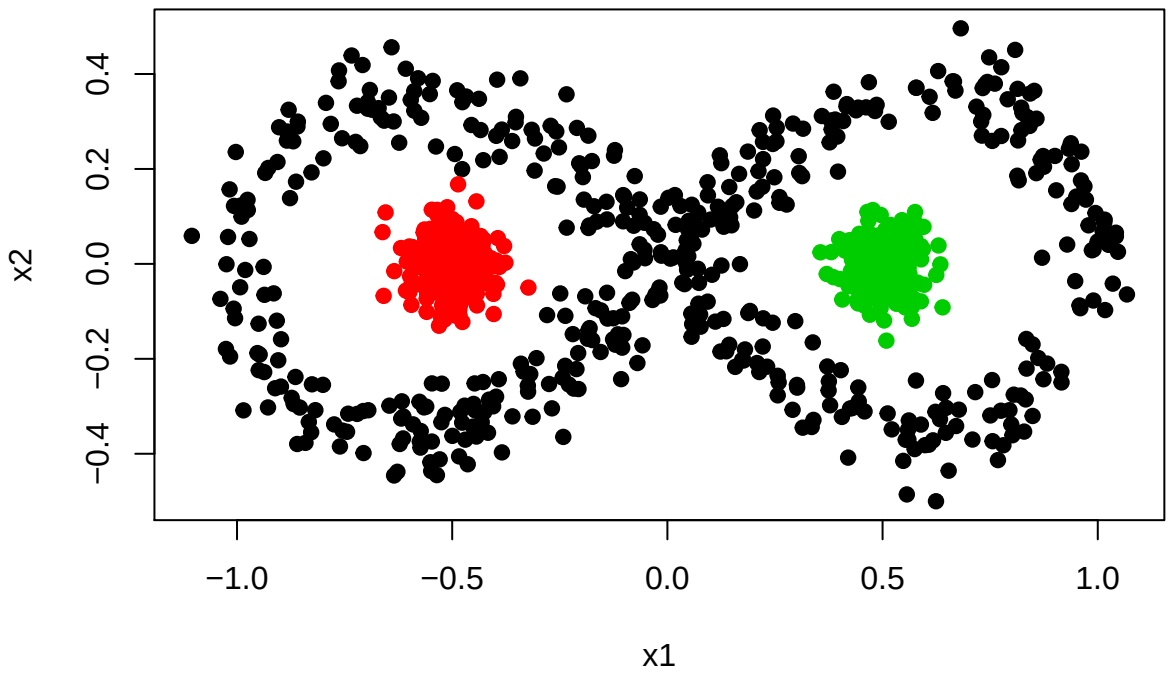
In the boosted model with 4 splits in each tree, training error rate is 0 and test error rate is 7.15%. The model in part (b) is slightly better than that in part (a), but the improvement is not significant. Thus, neither of them should be used for new data because these models are overfitting.

B.3

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.4.4
```

```
set.seed(2019)
eightdense <- make.eight(1000, spread = 0.05)
```



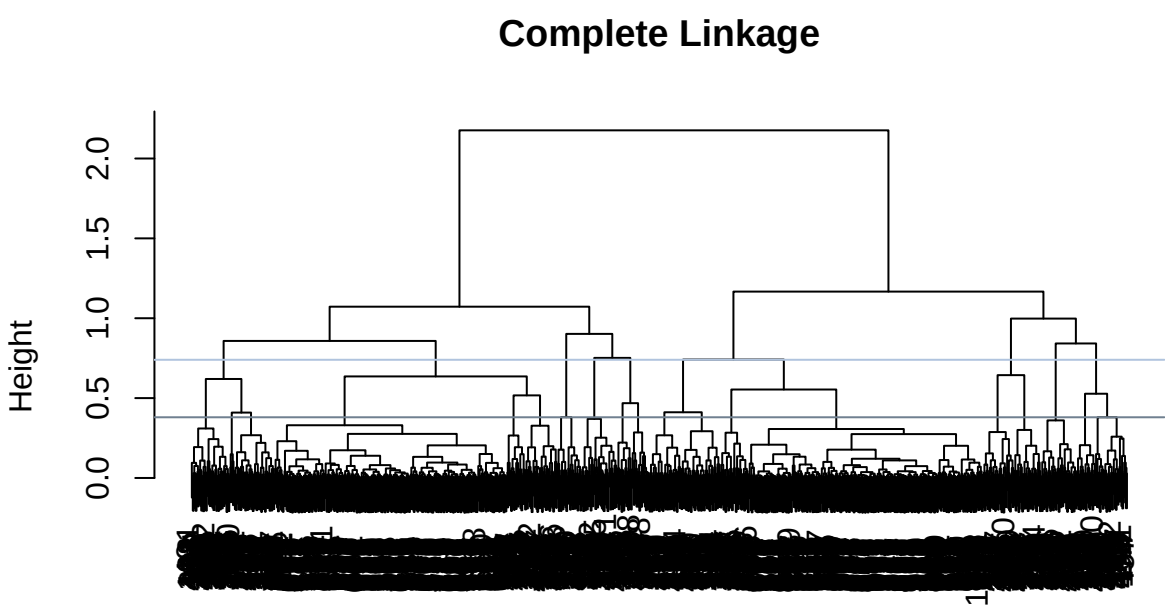

```
eightdense <- subset(eightdense, select = -label)
```

(a)

```
eight.dist <- dist(eightdense[,1:2], diag = TRUE, upper = TRUE)
hc.complete <- hclust(eight.dist, method = "complete")
hc.complete$height[980:999]
```

```
##  [1] 0.3799486 0.3802956 0.4093505 0.4113162 0.4682846 0.5170263 0.5273921
##  [8] 0.5536861 0.6197647 0.6356746 0.6437483 0.7413649 0.7517767 0.8420140
## [15] 0.8579969 0.9020657 0.9978527 1.0722778 1.1668879 2.1763102
```

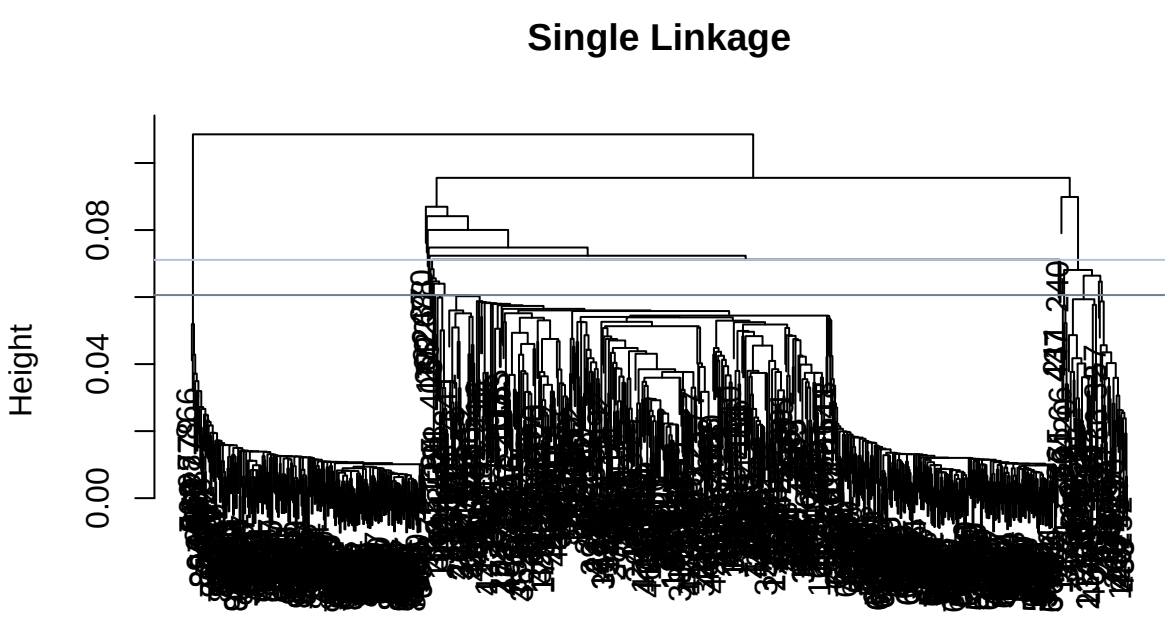
```
plot(hc.complete, main = "Complete Linkage", xlab = "")
abline(h = 0.74, col = "lightsteelblue")
abline(h = 0.38, col = "slategray")
```




```
hc.single <- hclust(eight.dist, method = "single")
hc.single$height[980:999]
```

```
##  [1] 0.06055039 0.06060832 0.06399683 0.06463443 0.06466890 0.06559857
##  [7] 0.06639064 0.06808642 0.06834353 0.06927017 0.07074986 0.07116745
## [13] 0.07233364 0.07479034 0.08002902 0.08413931 0.08695046 0.08986223
## [19] 0.09557843 0.10855473
```

```
plot(hc.single, main = "Single Linkage", xlab = "")
abline(h = 0.0711, col = "lightsteelblue")
abline(h = 0.0606, col = "slategray")
```




```
complete.10 <- cutree(hc.complete, k = 10)
table(eightdense$charlabel, complete.10)
```

```
##         complete.10
##            1   2   3   4   5   6   7   8   9  10
##   eight   76  53  88  38  31  62  38  34  54  26
##   left     0 249   1   0   0   0   0   0   0   0
##   right    0   0   0   0 250   0   0   0   0   0
```

```
complete.20 <- cutree(hc.complete, k = 20)
table(eightdense$charlabel, complete.20)
```

```
##         complete.20
##            1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
##   eight   34  35  40  18  38  31  33  38  42  43  16  20  29  15  34   5
##   left     0   0   1   1   0   0   0   0   0   0   0   0   0   0   0   0
##   right    0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##         complete.20
##           17  18  19  20
##   eight   18  11   0   0
##   left     0   0 248   0
##   right    0   0   0 250
```

```
single.10 <- cutree(hc.single, k = 10)
table(eightdense$charlabel, single.10)
```

```
##         single.10
##            1   2   3   4   5   6   7   8   9  10
```

```
##   eight 422  70   1   3   1   1   1   1   0   0
##   left  249   0   0   0   0   0   0   0   1   0
##   right   0   0   0   0   0   0   0   0   0 250
```

```
single.20 <- cutree(hc.single, k = 20)
table(eightdense$charlabel, single.20)
```

```
##        single.20
##           1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16
##   eight 416  30  32   1   7   1   1   1   1   1   1   1   2   1   1   1
##   left  249   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   right   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##        single.20
##          17  18  19  20
##   eight   1   1   0   0
##   left    0   0   1   0
##   right   0   0   0 250
```

This dataset has 1000 points. 500 points labeled `eight`, 250 points labeled `left` and the rest 250 points are labeled `right`.

Clustering result with complete linkage, cut at $k = 10$, 249 `left` points are in cluster **2** and all 250 `right` points are in cluster **5**.

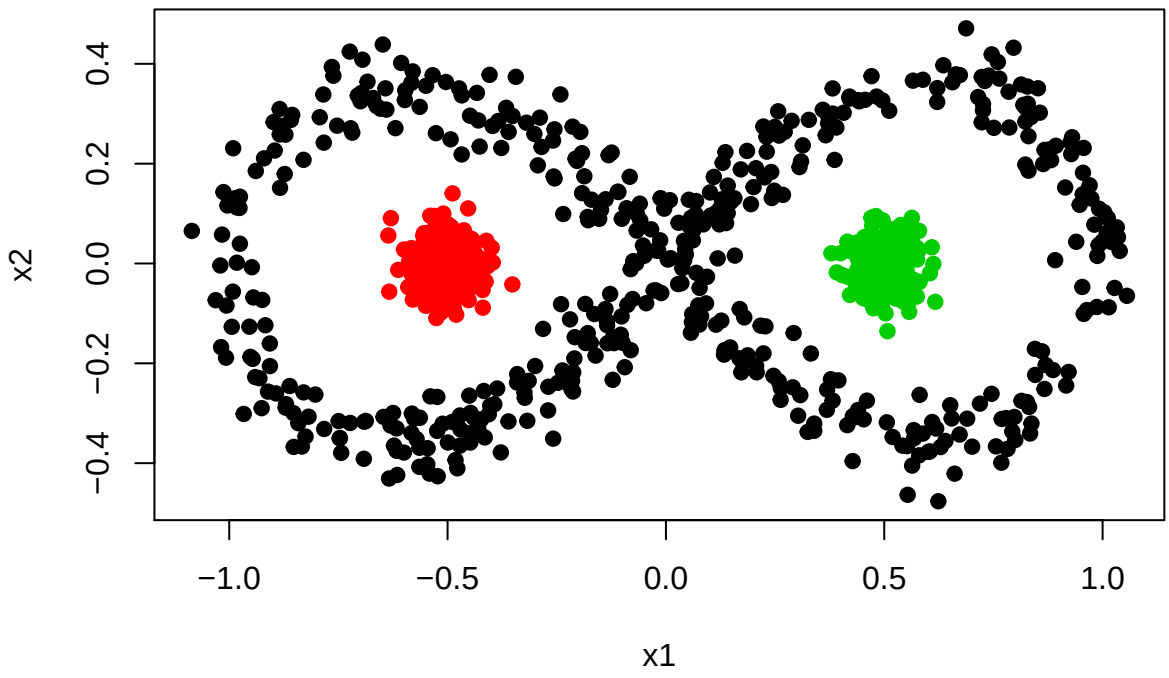
Clustering result with complete linkage, cut at $k = 20$, 248 `left` points are in cluster **19** and all 250 `right` points are in cluster **20**.

Clustering result with single linkage, cut at $k = 10$, 249 `left` points are in cluster **1** and all 250 `right` points are in cluster **10**.

Clustering result with single linkage, cut at $k = 20$, 249 `left` points are in cluster **1** and all 250 `right` points are in cluster **20**.

(b)

```
set.seed(2019)
sp <- .0419
eightperfect <- make.eight(1000, spread = sp)
```




```
eight.dist.p <- dist(eightperfect[,1:2], diag = TRUE, upper = TRUE)
hc.single.p <- hclust(eight.dist.p, method = "single")
single.3 <- cutree(hc.single.p, k = 3)
table(eightperfect$charlabel, single.3)
```

```
##         single.3
##            1   2   3
##   eight  500   0   0
##   left     0 250   0
##   right    0   0 250
```

After several trials, hierarchical clustering with single linkage can perfectly identify 3 clusters when data point spread is as small or smaller than 0.0419.