Anusha Muppalla
16286311

LAB Assignment8,9

1.We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features.

(a) (5 points) Generate a data set with n = 500 and p = 2, such that the observations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows:
> x1=runif (500) -0.5

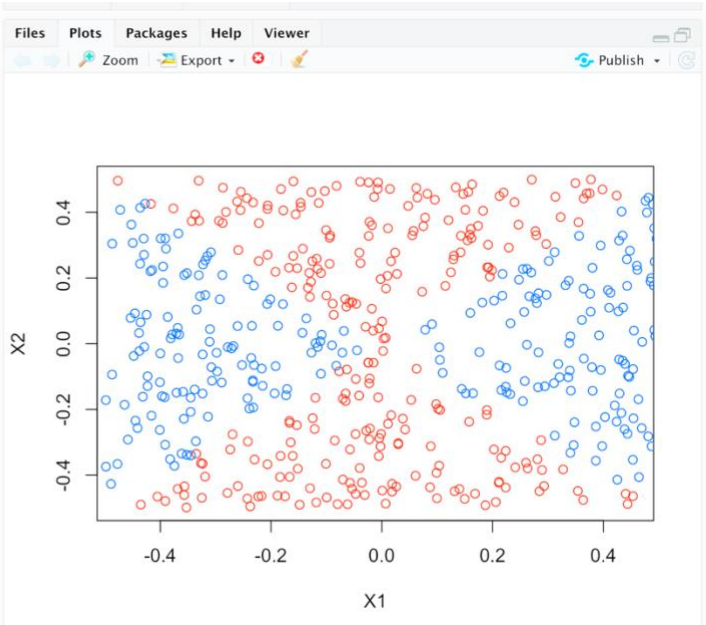
> x2=runif (500) -0.5

> y = 1*(x1□2-x2□2 > 0)

```
> set.seed(1)
> x1 <- runif(500) - 0.5
> x2 <- runif(500) - 0.5
> y <- as.integer(x1 ^ 2 - x2 ^ 2 > 0)
>
```

(b) (5 points) Plot the observations, colored according to their class labels. Your plot should display X_1 on the x-axis, and X_2 on the yaxis.

```
> set.seed(1)
> x1 <- runif(500) - 0.5
> x2 <- runif(500) - 0.5
> y <- as.integer(x1 ^ 2 - x2 ^ 2 > 0)
> plot(x1[y == 0], x2[y == 0], col = "red", xlab = "X
1", ylab = "X2")
> points(x1[y == 1], x2[y == 1], col = "blue")
>
```

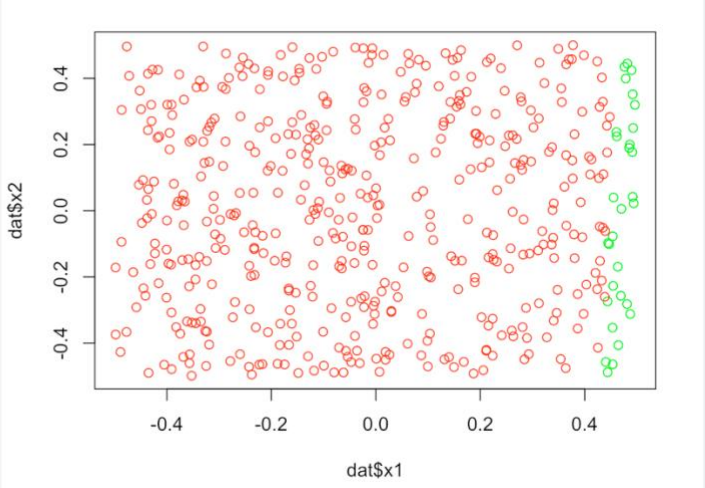


(c) (5 points) Fit a logistic regression model to the data, using X_1 and X_2 as predic- tors.

```
> points(x1[y == 1], x2[y == 1], col = "blue")
> dat <- data.frame(x1 = x1, x2 = x2, y = as.factor(y))
> lr.fit <- glm(y ~ ., data = dat, family = 'binomial')
>
```

(d) (5 points) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.

```
> points(x1[y == 1], x2[y == 1], col = "blue")
> dat <- data.frame(x1 = x1, x2 = x2, y = as.factor(y))
> lr.fit <- glm(y ~ ., data = dat, family = 'binomial')
> lr.prob <- predict(lr.fit, newdata = dat, type = 'res
ponse')
> lr.pred <- ifelse(lr.prob > 0.5, 1, 0)
> plot(dat$x1, dat$x2, col = lr.pred + 2)
>
```

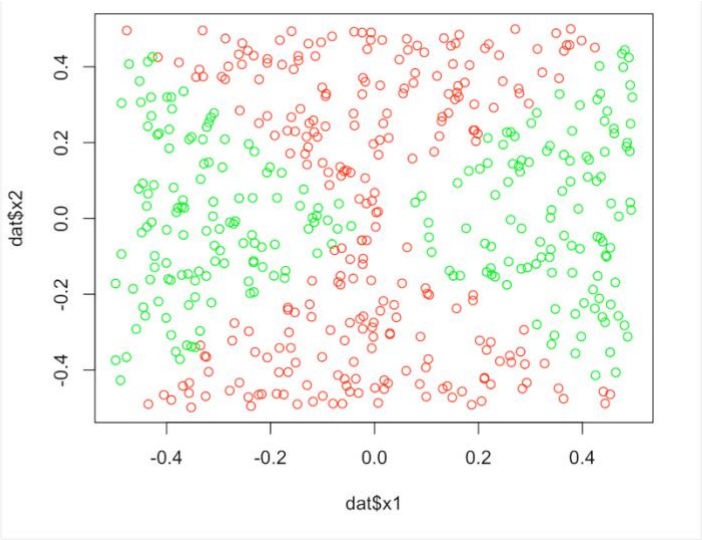


(e) (5 points) Now fit a logistic regression model to the data using non-linear functions of X_1 and X_2 as predictors (e.g. X_1^2, $X_1 \times X_2$, $\log(X_2)$, and so forth).

```
> summary(lr.nl)

Call:
glm(formula = y ~ poly(x1, 2) + poly(x2, 2), family =
  "binomial",
    data = dat)

Deviance Residuals:
       Min          1Q      Median          3Q
-1.079e-03  -2.000e-08  -2.000e-08   2.000e-08
       Max
 1.297e-03

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)     -94.48    2963.78  -0.032    0.975
poly(x1, 2)1   3442.52  104411.28   0.033    0.974
poly(x1, 2)2  30110.74  858421.66   0.035    0.972
poly(x2, 2)1    162.82   26961.99   0.006    0.995
poly(x2, 2)2 -31383.76  895267.48  -0.035    0.972

(Dispersion parameter for binomial family taken to be
 1)

    Null deviance: 6.9218e+02  on 499  degrees of freed
om
Residual deviance: 4.2881e-06  on 495  degrees of freed
om
AIC: 10

Number of Fisher Scoring iterations: 25
```

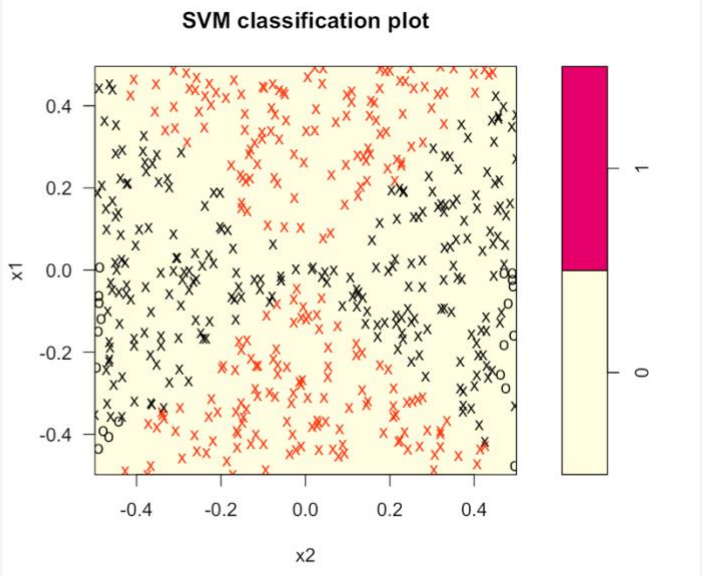
(f) (5 points) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be obviously non- linear. If it is not, then repeat (a)-(e) until you come up with an example in which the predicted class labels are obviously non-linear.

```
 1.297e-03

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)     -94.48    2963.78  -0.032    0.975
poly(x1, 2)1   3442.52  104411.28   0.033    0.974
poly(x1, 2)2  30110.74  858421.66   0.035    0.972
poly(x2, 2)1    162.82   26961.99   0.006    0.995
poly(x2, 2)2 -31383.76  895267.48  -0.035    0.972

(Dispersion parameter for binomial family taken to be
 1)

    Null deviance: 6.9218e+02  on 499  degrees of freed
om
Residual deviance: 4.2881e-06  on 495  degrees of freed
om
AIC: 10

Number of Fisher Scoring iterations: 25

> lr.prob.nl <- predict(lr.nl, newdata = dat, type = 'r
esponse')
> lr.pred.nl <- ifelse(lr.prob.nl > 0.5, 1, 0)
> plot(dat$x1, dat$x2, col = lr.pred.nl + 2)
>
```



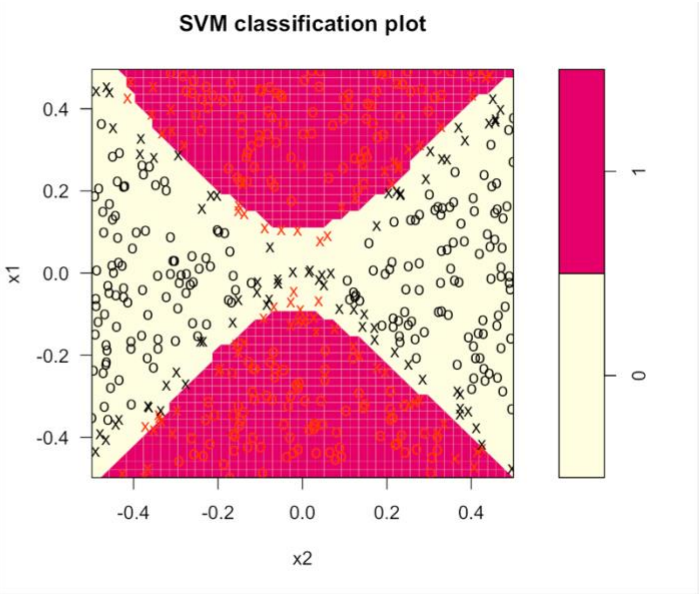
(g) (5 points) Fit a support vector classifier to the data with X_1 and X_2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
> install.packages(e1071)
Error in install.packages : object 'e1071' not found
> install.packages('e1071')
trying URL 'https://cran.rstudio.com/bin/macosx/el-capi
tan/contrib/3.6/e1071_1.7-3.tgz'
Content type 'application/x-gzip' length 900756 bytes
 (879 KB)
==================================================
downloaded 879 KB


The downloaded binary packages are in
        /var/folders/w5/msff5yg55hvgnrf5gxk_qk680000gn/
T//RtmpxjKuGh/downloaded_packages
> svm.lin <- svm(y ~ ., data = dat, kernel = 'linear',
 cost = 0.01)
Error in svm(y ~ ., data = dat, kernel = "linear", cost
 = 0.01) :
  could not find function "svm"
> plot(svm.lin, dat)
Error in plot(svm.lin, dat) : object 'svm.lin' not foun
d
> library(e1071
+ )
> library(e1071)
> svm.lin <- svm(y ~ ., data = dat, kernel = 'linear',
 cost = 0.01)
> plot(svm.lin, dat)
>
```



(h) (5 points) Fit a SVM using a non-linear kernel to the data. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
trying URL 'https://cran.rstudio.com/bin/macosx/el-capi
tan/contrib/3.6/e1071_1.7-3.tgz'
Content type 'application/x-gzip' length 900756 bytes
 (879 KB)
==================================================
downloaded 879 KB


The downloaded binary packages are in
        /var/folders/w5/msff5yg55hvgnrf5gxk_qk680000gn/
T//RtmpxjKuGh/downloaded_packages
> svm.lin <- svm(y ~ ., data = dat, kernel = 'linear',
 cost = 0.01)
Error in svm(y ~ ., data = dat, kernel = "linear", cost
 = 0.01) :
  could not find function "svm"
> plot(svm.lin, dat)
Error in plot(svm.lin, dat) : object 'svm.lin' not foun
d
> library(e1071
+ )
> library(e1071)
> svm.lin <- svm(y ~ ., data = dat, kernel = 'linear',
 cost = 0.01)
> plot(svm.lin, dat)
> svm.nl <- svm(y ~ ., data = dat, kernel = 'radial', g
amma = 1)
> plot(svm.nl, data = dat)
>
```



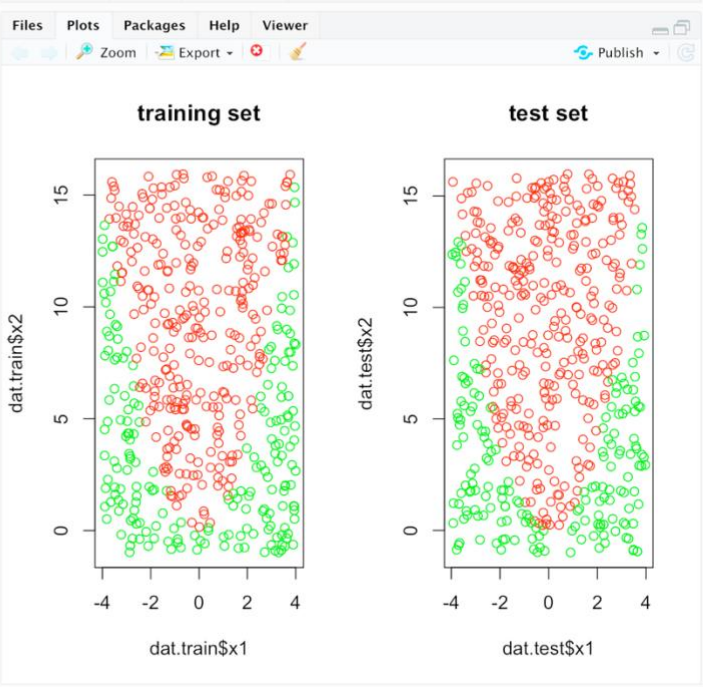
(i) (5 points) Comment on your results.

The effect of linear logistic regression on nonlinear boundary is poor and the effect of svm as linear kernel is small and the effect of nonlinear logistic regression and svm on nonlinear boundary is good

2.At the end of Section 9.6.1, it is claimed that in the case of data that is just barely linearly separable, a support vector classifier with a small value of cost that misclassifies a couple of training observations may perform better on test data than one with a huge value of cost that does not misclassify any training observations. You will now investigate this claim.

(a) (5 points) Generate two-class data with p = 2 in such a way that the classes are just barely linearly separable.

```
> set.seed(1)
> obs = 1000
> x1 <- runif(obs, min = -4, max = 4)
> x2 <- runif(obs, min = -1, max = 16)
> y <- ifelse(x2 > x1 ^ 2, 0, 1)
> dat <- data.frame(x1 = x1, x2 = x2, y = as.factor(y))
> train <- sample(obs, obs/2)
> dat.train <- dat[train, ]
> dat.test <- dat[-train, ]
> par(mfrow = c(1,2))
> plot(dat.train$x1, dat.train$x2, col = as.integer(da
t.train$y) + 1, main = 'training set')
> plot(dat.test$x1, dat.test$x2, col = as.integer(dat.t
est$y) + 1, main = 'test set')
>
```

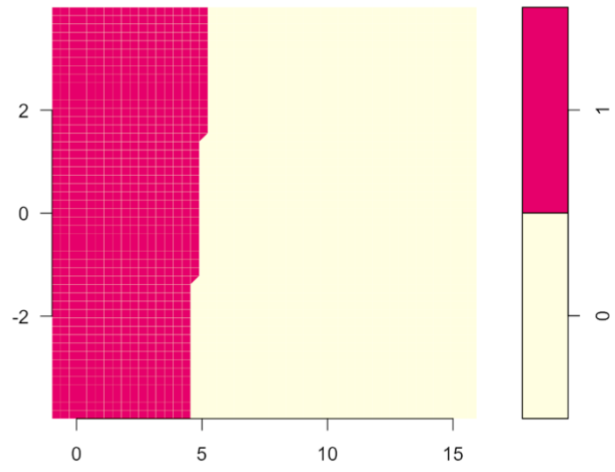


(b) (5 points) Compute the cross-validation error rates for support vector classifiers with a range of cost values. How many training errors are misclassified for each value of cost considered, and how does this relate to the cross-validation errors obtained?

```
> set.seed(1)
> cost.grid <- c(0.001, 0.01, 0.1, 1, 5, 10, 100, 1000
0)
> tune.out <- tune(svm, y ~., data = dat.train, kernel
  = 'linear', ranges = list(cost = cost.grid))
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
WARNING: reaching max number of iterations
> summary(tune.out)

Parameter tuning of 'svm':
```

```
WARNING: reaching max number of iterations

WARNING: reaching max number of iterations
> summary(tune.out)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 cost
    5

- best performance: 0.25

- Detailed performance results:
   cost error dispersion
1 1e-03 0.392 0.04825856
2 1e-02 0.256 0.05947922
3 1e-01 0.252 0.05902918
4 1e+00 0.252 0.06051630
5 5e+00 0.250 0.06271629
6 1e+01 0.250 0.06271629
7 1e+02 0.250 0.06271629
8 1e+04 0.390 0.05270463

>
```

```
- best parameters:
 cost
    5

- best performance: 0.25

- Detailed performance results:
   cost error dispersion
1 1e-03 0.392 0.04825856
2 1e-02 0.256 0.05947922
3 1e-01 0.252 0.05902918
4 1e+00 0.252 0.06051630
5 5e+00 0.250 0.06271629
6 1e+01 0.250 0.06271629
7 1e+02 0.250 0.06271629
8 1e+04 0.390 0.05270463

> err.rate.train <- rep(NA, length(cost.grid))
> for (cost in cost.grid) {
+     svm.fit <- svm(y ~ ., data = dat.train, kernel =
 'linear', cost = cost)
+     plot(svm.fit, data = dat.train)
+     res <- table(prediction = predict(svm.fit, newdat
a = dat.train), truth = dat.train$y)
+     err.rate.train[match(cost, cost.grid)] <- (res[2,
1] + res[1,2]) / sum(res)
+ }
```



```
WARNING: reaching max number of iterations
> err.rate.train
[1] 0.392 0.246 0.248 0.248 0.248 0.248 0.248 0.392
> paste('The cost', cost.grid[which.min(err.rate.train)], 'has the minimum training error:', min(er
r.rate.train))
[1] "The cost 0.01 has the minimum training error: 0.246"
>
```

(c) (5 points) Generate an appropriate test data set, and compute the test errors corresponding to each of the values of cost considered. Which value of cost leads to the fewest test errors, and how does this compare to the values of cost that yield the fewest training errors and the fewest cross-validation errors?

```
> err.rate.test <- rep(NA, length(cost.grid))
> for (cost in cost.grid) {
+     svm.fit <- svm(y ~ ., data = dat.train, kernel = 'linear', cost = cost)
+     res <- table(prediction = predict(svm.fit, newdata = dat.test), truth = dat.test$y)
+     err.rate.test[match(cost, cost.grid)] <- (res[2,1] + res[1,2]) / sum(res)
+ }

WARNING: reaching max number of iterations
> err.rate.test
[1] 0.362 0.216 0.216 0.218 0.220 0.220 0.218 0.362
> paste('The cost', cost.grid[which.min(err.rate.test)], 'has the minimum test error:', min(err.rat
e.test))
[1] "The cost 0.01 has the minimum test error: 0.216"
>
```

(d) (5 points) Discuss your results.

I have observed the overfitting for linear kernel because of noisy points.

3.(10 points total) In Section 10.2.3, a formula for calculating PVE was given in Equation 10.8. We also saw that the PVE can be obtained using the sdev output of the prcomp() function. On the USArrests data, calculate PVE in two ways:

(a) (5 points) Using the sdev output of the prcomp() function, as was done in Section 10.2.3.

```
> packages <- c('ISLR', 'tidyverse', 'knitr', 'kableExtra')
> sapply(packages, library, character.only = TRUE, quietly = TRUE, verbose =
 FALSE)

Attaching package: ‘kableExtra’

The following object is masked from ‘package:dplyr’:

    group_rows

$ISLR
 [1] "knitr"     "forcats"   "stringr"   "dplyr"     "purrr"
 [6] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
[11] "ISLR"      "e1071"     "stats"     "graphics"  "grDevices"
[16] "utils"     "datasets"  "methods"   "base"

$tidyverse
 [1] "knitr"     "forcats"   "stringr"   "dplyr"     "purrr"
 [6] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
[11] "ISLR"      "e1071"     "stats"     "graphics"  "grDevices"
[16] "utils"     "datasets"  "methods"   "base"

$knitr
 [1] "knitr"     "forcats"   "stringr"   "dplyr"     "purrr"
 [6] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
[11] "ISLR"      "e1071"     "stats"     "graphics"  "grDevices"
[16] "utils"     "datasets"  "methods"   "base"

$kableExtra
 [1] "kableExtra" "knitr"      "forcats"    "stringr"    "dplyr"
 [6] "purrr"      "readr"      "tidyr"      "tibble"     "ggplot2"
[11] "tidyverse"  "ISLR"       "e1071"      "stats"      "graphics"
[16] "grDevices"  "utils"      "datasets"   "methods"    "base"
```

```
$tidyverse
 [1] "knitr"     "forcats"   "stringr"   "dplyr"     "purrr"
 [6] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
[11] "ISLR"      "e1071"     "stats"     "graphics"  "grDevices"
[16] "utils"     "datasets"  "methods"   "base"

$knitr
 [1] "knitr"     "forcats"   "stringr"   "dplyr"     "purrr"
 [6] "readr"     "tidyr"     "tibble"    "ggplot2"   "tidyverse"
[11] "ISLR"      "e1071"     "stats"     "graphics"  "grDevices"
[16] "utils"     "datasets"  "methods"   "base"

$kableExtra
 [1] "kableExtra" "knitr"      "forcats"    "stringr"    "dplyr"
 [6] "purrr"      "readr"      "tidyr"      "tibble"     "ggplot2"
[11] "tidyverse"  "ISLR"       "e1071"      "stats"      "graphics"
[16] "grDevices"  "utils"      "datasets"   "methods"    "base"

> data(USArrests)
> USArrests <- scale(USArrests)
> comp.arrests <- prcomp(USArrests)
> comp.var <- comp.arrests$sdev^2
> comp.var
[1] 2.4802416 0.9897652 0.3565632 0.1734301
> comp.var/sum(comp.var)
[1] 0.62006039 0.24744129 0.08914080 0.04335752
>
```

(b) (5 points) By applying Equation 10.8 directly. That is, use the prcomp() func- tion to compute the principal component loadings. Then, use those loadings in Equation 10.8 to obtain the PVE.

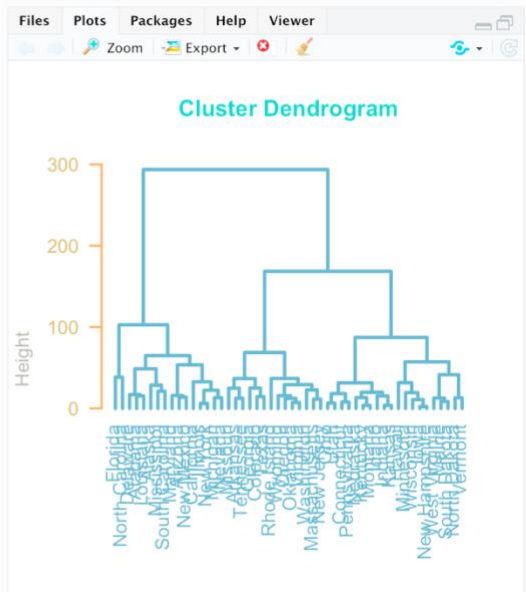
```
> head(USArrests)
               Murder   Assault   UrbanPop         Rape
Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
Arizona    0.07163341 1.4788032  0.9989801  1.042878388
Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
California 0.27826823 1.2628144  1.7589234  2.067820292
Colorado   0.02571456 0.3988593  0.8608085  1.864967207
>
```

```
> comp.arrests$rotation
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
> dim(USArrests)
[1] 50  4
> dim(comp.arrests$rotation)
[1] 4 4
> USArrests %*%
+     comp.arrests$rotation %>%
+     (function(x) x^2) %>%
+     as.tibble %>%
+     rowwise() %>%
+     map_df(sum) %>%
+     ungroup() %>%
+     (function(x) x/sum(USArrests^2))
        PC1       PC2       PC3        PC4
1 0.6200604 0.2474413 0.0891408 0.04335752
```

4.(20 points total) Consider the USArrests data. We will now perform hierarchical clustering on the states.

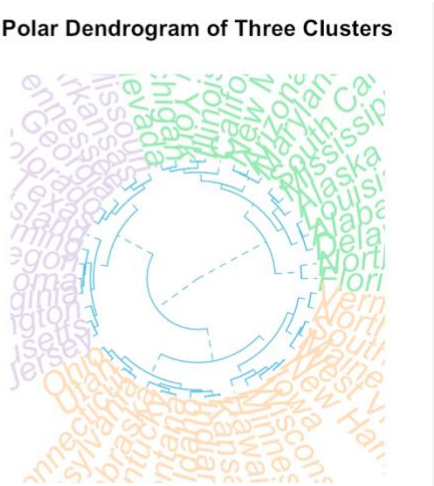
(a) (5 points) Using hierarchical clustering with complete linkage and Euclidean distance, cluster the states.

```
> require(ISLR); require(tidyverse); require(ggthemes)
Loading required package: ggthemes
> theme_set(theme_tufte(base_size = 14))
> data(USArrests)
>
> usa.dist <- dist(USArrests)
> usa.hclust <- hclust(usa.dist)
>
> plot(usa.hclust, col = "#487AA1", col.main = "#45ADA8", col.lab = "#7C807
1",
+      col.axis = "#F38630", lwd = 3, lty = 1, sub = "", hang = -1, axes = FA
LSE,
+      xlab = '')
>
> axis(side = 2, at = seq(0, 300, length.out = 4), col = "#F38630", labels =
 FALSE,
+      lwd = 2)
>
> mtext(seq(0, 300, length.out = 4), side = 2, at = seq(0, 300, length.out =
 4), line = 1,
+       col = "#A38630", las = 2)
>
```



(b) (5 points) Cut the dendrogram at a height that results in three distinct clusters. Which states belong to which clusters?

```
  there is no package called 'ape'
Loading required package: RColorBrewer
>
> plot(as.phylo(usa.hclust), type = 'fan',
+      tip.color = brewer.pal(3, 'Accent')[cutree(usa.hclust, 3)],
+      edge.color = 'steelblue', edge.lty = 2,
+      cex = 1.8,
+      main = 'Polar Dendrogram of Three Clusters')
Error in as.phylo(usa.hclust) : could not find function "as.phylo"
> install.packages("ape")
trying URL 'https://cran.rstudio.com/bin/macosx/el-capitan/contrib/3.6/ape_5.
3.tgz'
Content type 'application/x-gzip' length 2612810 bytes (2.5 MB)
==================================================
downloaded 2.5 MB


The downloaded binary packages are in
        /var/folders/w5/msff5yg55hvgnrf5gxk_qk680000gn/T//RtmpxjKuGh/download
ed_packages
> require(ape); require(RColorBrewer)
Loading required package: ape
>
> plot(as.phylo(usa.hclust), type = 'fan',
+      tip.color = brewer.pal(3, 'Accent')[cutree(usa.hclust, 3)],
+      edge.color = 'steelblue', edge.lty = 2,
+      cex = 1.8,
+      main = 'Polar Dendrogram of Three Clusters')
>
```

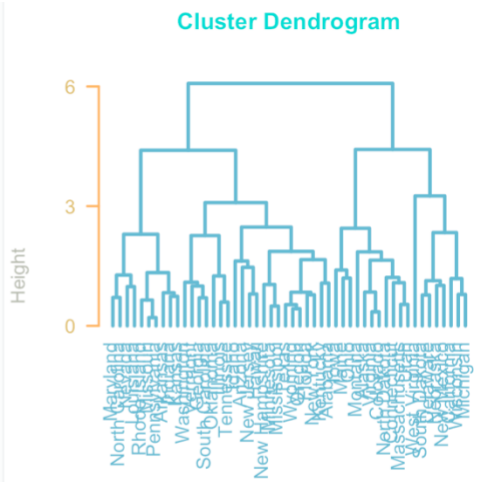


(c) (5 points) Hierarchically cluster the states using complete linkage and Euclidean distance, after scaling the variables to have standard deviation one.

```
> scaling <- function(x) (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
>
> clust.scaled <- USArrests %>%
+     map_df(scaling) %>%
+     dist(method = 'euclidean') %>%
+     hclust(method = 'complete')
>
> clust.scaled$labels <- row.names(USArrests)[clust.scaled$order]
>
```

(d) (5 points) What effect does scaling the variables have on the hierarchical clustering obtained? In your opinion, should the variables be scaled before the inter-observation dissimilarities are computed? Provide a justification for your answer.

```
Loading required package: ape
>
> plot(as.phylo(usa.hclust), type = 'fan',
+      tip.color = brewer.pal(3, 'Accent')[cutree(usa.hclust, 3)],
+      edge.color = 'steelblue', edge.lty = 2,
+      cex = 1.8,
+      main = 'Polar Dendrogram of Three Clusters')
> scaling <- function(x) (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
>
> clust.scaled <- USArrests %>%
+     map_df(scaling) %>%
+     dist(method = 'euclidean') %>%
+     hclust(method = 'complete')
>
> clust.scaled$labels <- row.names(USArrests)[clust.scaled$order]
> plot(clust.scaled, col = "#487AA1", col.main = "#45ADA8", col.lab = "#7C807
1",
+      col.axis = "#F38630", lwd = 3, lty = 1, sub = "", hang = -1, axes = FA
LSE,
+      xlab = '')
>
> axis(side = 2, at = seq(0, 6, length.out = 3), col = "#F38630", labels = FA
LSE,
+      lwd = 2)
>
> mtext(seq(0, 6, length.out = 3), side = 2, at = seq(0, 6, length.out = 3),
 line = 1,
+       col = "#A38630", las = 2)
>
```




```
> scaling <- function(x) (x - mean(x, na.rm = TRUE)) / sd(x, na.rm = TRUE)
>
> clust.scaled <- USArrests %>%
+     map_df(scaling) %>%
+     dist(method = 'euclidean') %>%
+     hclust(method = 'complete')
>
> clust.scaled$labels <- row.names(USArrests)[clust.scaled$order]
> plot(clust.scaled, col = "#487AA1", col.main = "#45ADA8", col.lab = "#7C807
1",
+      col.axis = "#F38630", lwd = 3, lty = 1, sub = "", hang = -1, axes = FA
LSE,
+      xlab = '')
>
> axis(side = 2, at = seq(0, 6, length.out = 3), col = "#F38630", labels = FA
LSE,
+      lwd = 2)
>
> mtext(seq(0, 6, length.out = 3), side = 2, at = seq(0, 6, length.out = 3),
 line = 1,
+       col = "#A38630", las = 2)
> require(ape); require(RColorBrewer)
>
> plot(as.phylo(clust.scaled), type = 'fan',
+      tip.color = brewer.pal(3, 'Accent')[cutree(clust.scaled, 3)],
+      edge.color = 'steelblue', edge.lty = 2,
+      cex = 1.8,
+      main = 'Polar Dendrogram of Three Clusters')
>
```