Homework 5

Daniel Dulaney

November 7, 2020

```r
library(tidyverse)
```

```
## -- Attaching packages ----------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts -------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(here)
```

```
## here() starts at C:/Users/dgdul/OneDrive/Documents/grad-school/st-563
```

```r
library(ggrepel)
library(ISLR)
library(e1071)
```

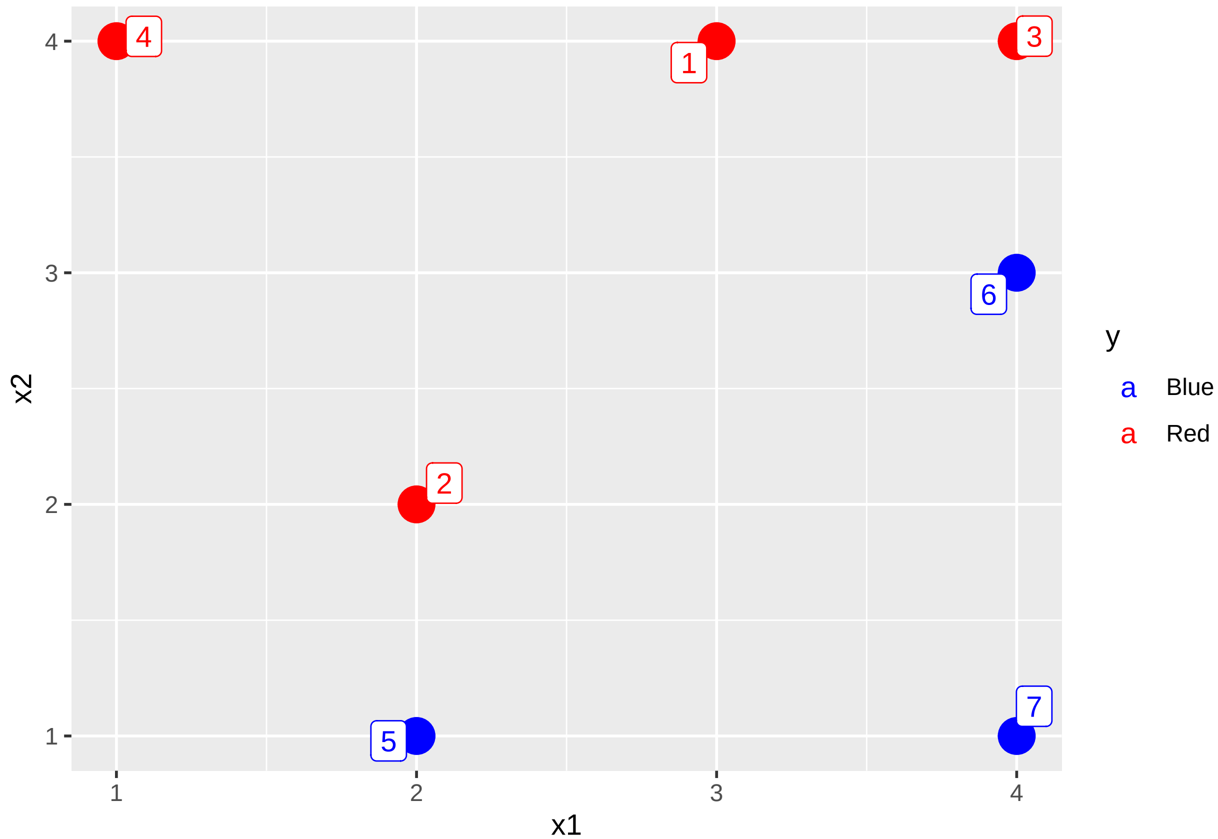
```
## Warning: package 'e1071' was built under R version 4.0.3
```

(9.3)

(a)

```r
df_9.3 <- tribble(
  ~obs, ~x1, ~x2, ~y,
  1, 3, 4, "Red",
  2, 2, 2, "Red",
  3, 4, 4, "Red",
  4, 1, 4, "Red",
  5, 2, 1, "Blue",
  6, 4, 3, "Blue",
  7, 4, 1, "Blue"
```

```
)

points_9.3 <- df_9.3 %>%
  ggplot(aes(x1, x2, color = y)) +
  geom_point(size = 6) +
  geom_label_repel(aes(label = obs)) +
  scale_color_manual(values = c("Blue", "Red"))

points_9.3
```



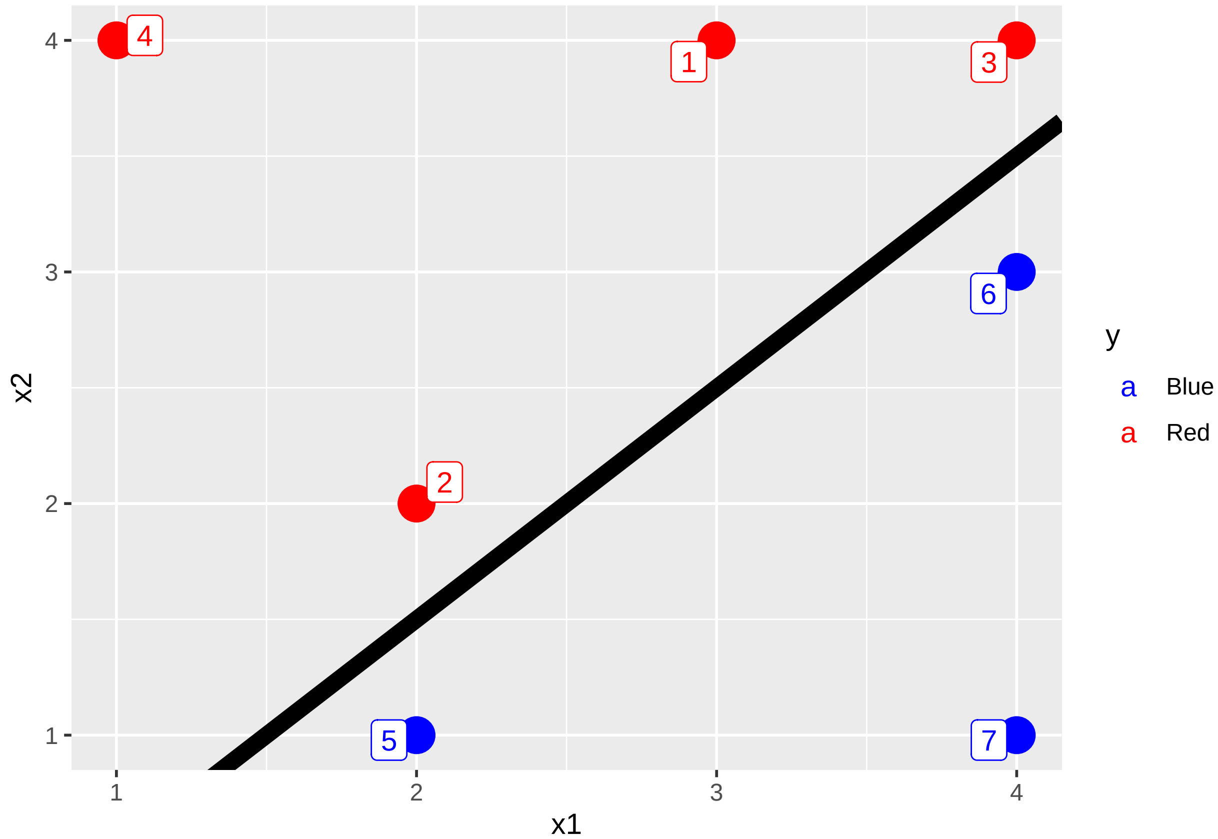
(b)

Based on the plot, we know we can construct a line that will separate the classes perfectly.

The maximal margin classifier will be a line between Blue points 5 (2, 1) and 6 (4, 3), and red points 2 (2, 2) and 3 (4, 4). The line will need to pass through the two points (4, 3.5) and (2, 1.5).

The slope of the line equals $\frac{3.5-1.5}{4-2} = 1$ and the intercept equals 1.5-2 = -0.5

```
points_9.3 +
  geom_abline(intercept = -.5,
              slope = 1,
              size = 3)
```



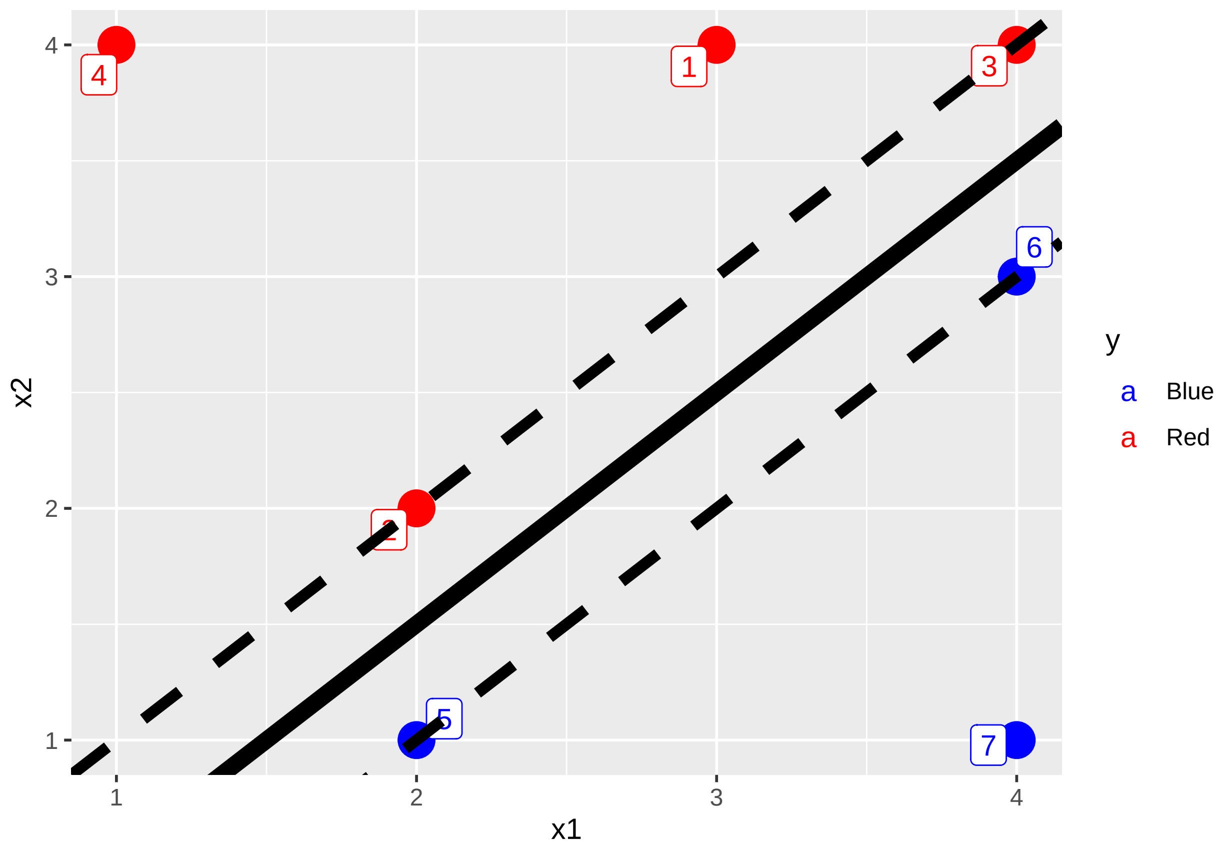
(c)

0.5 - $X_1 + X_2 > 0$

(d)

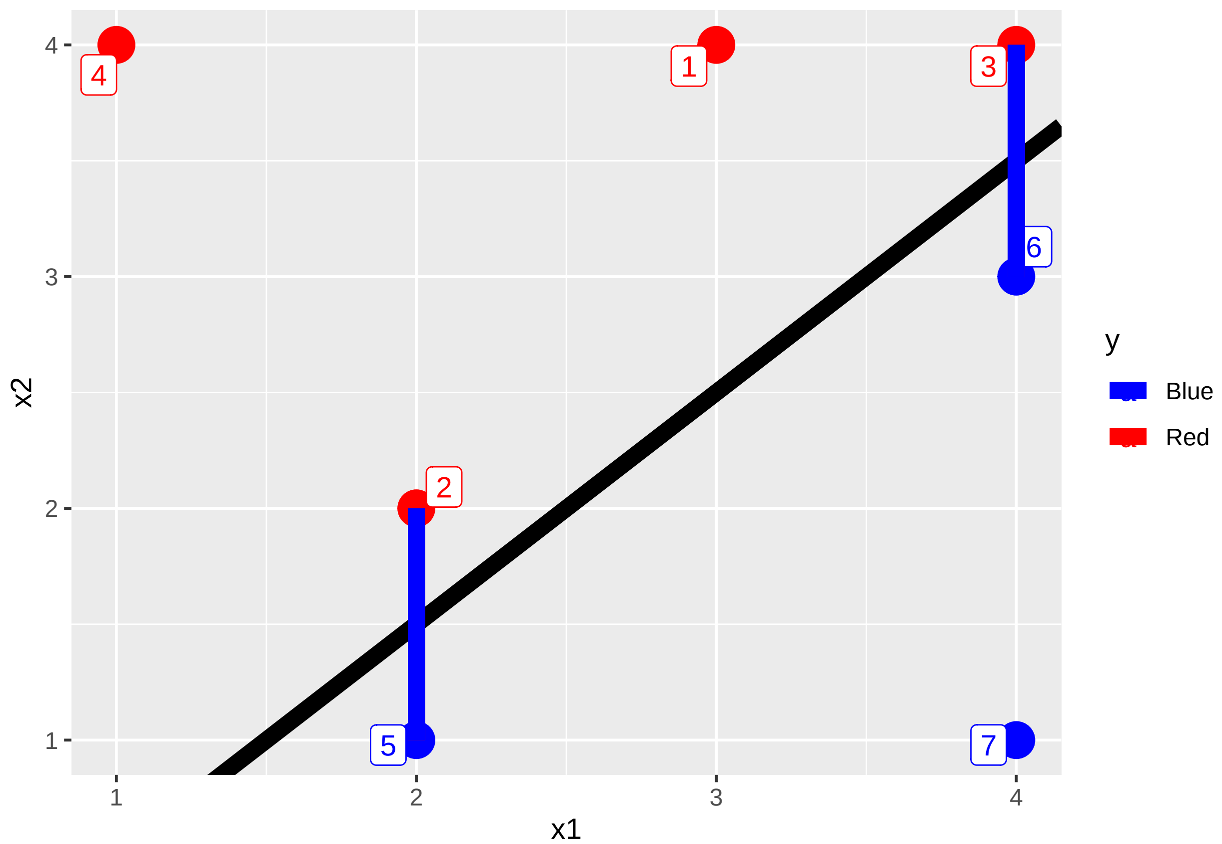
The line passing through Red points 2 and 3 has slope 1 and intercept -1, and the line passing through Blue points 5 and 6 has slope 1 and intercept 0

```
points_9.3 +
  geom_abline(intercept = -.5,
              slope = 1,
              size = 3) +
  geom_abline(intercept = -1,
              slope = 1,
              size = 2,
              linetype = "dashed") +
  geom_abline(intercept = 0,
              slope = 1,
              size = 2,
              linetype = "dashed")
```



(e)

```
points_9.3 +
  geom_abline(intercept = -.5,
              slope = 1,
              size = 3) +
  geom_segment(aes(x = 2,
                   y = 2,
                   xend = 2,
                   yend = 1),
               size = 3) +
  geom_segment(aes(x = 4,
                   y = 3,
                   xend = 4,
                   yend = 4),
               size = 3)
```

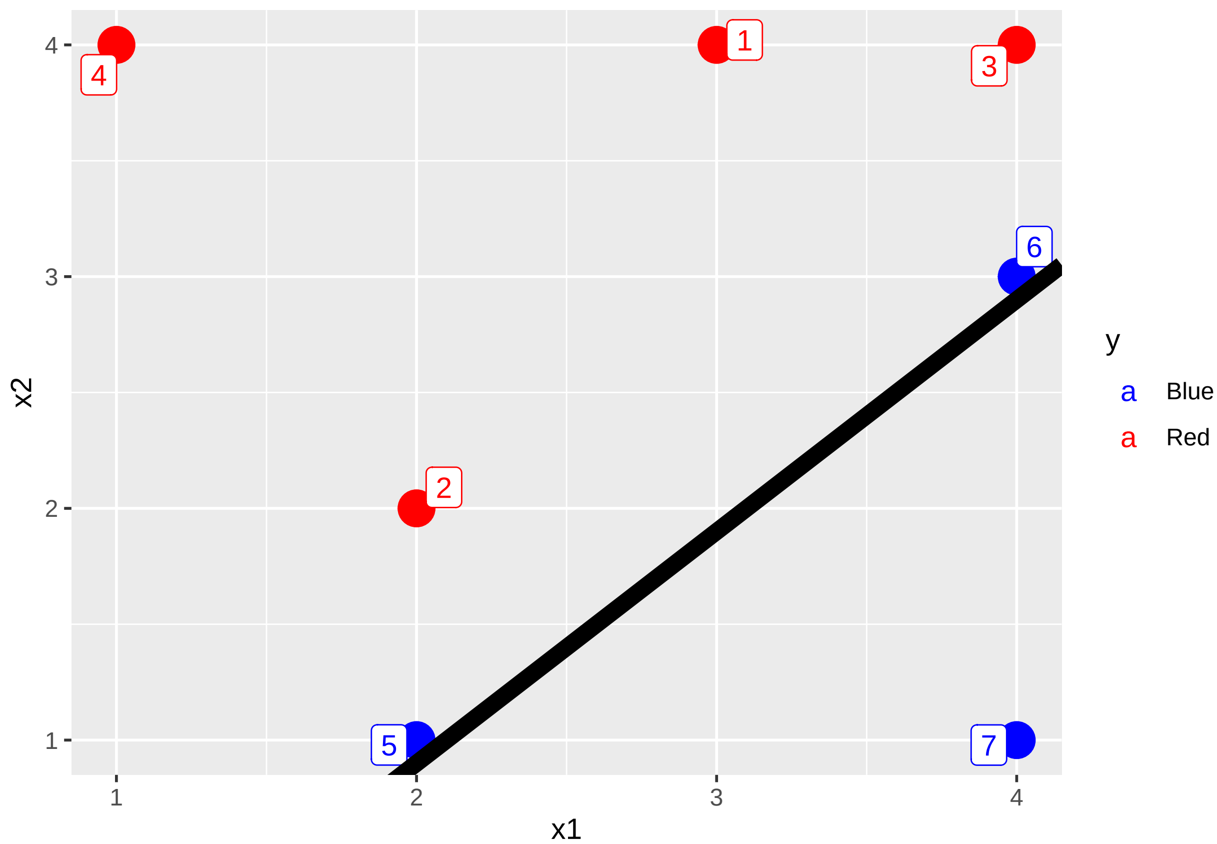


(f)

Since observation 7 is outside of the margin, which is the area between the dotted lines, moving it wouldn't affect the maximal margin hyperplane.

(g)

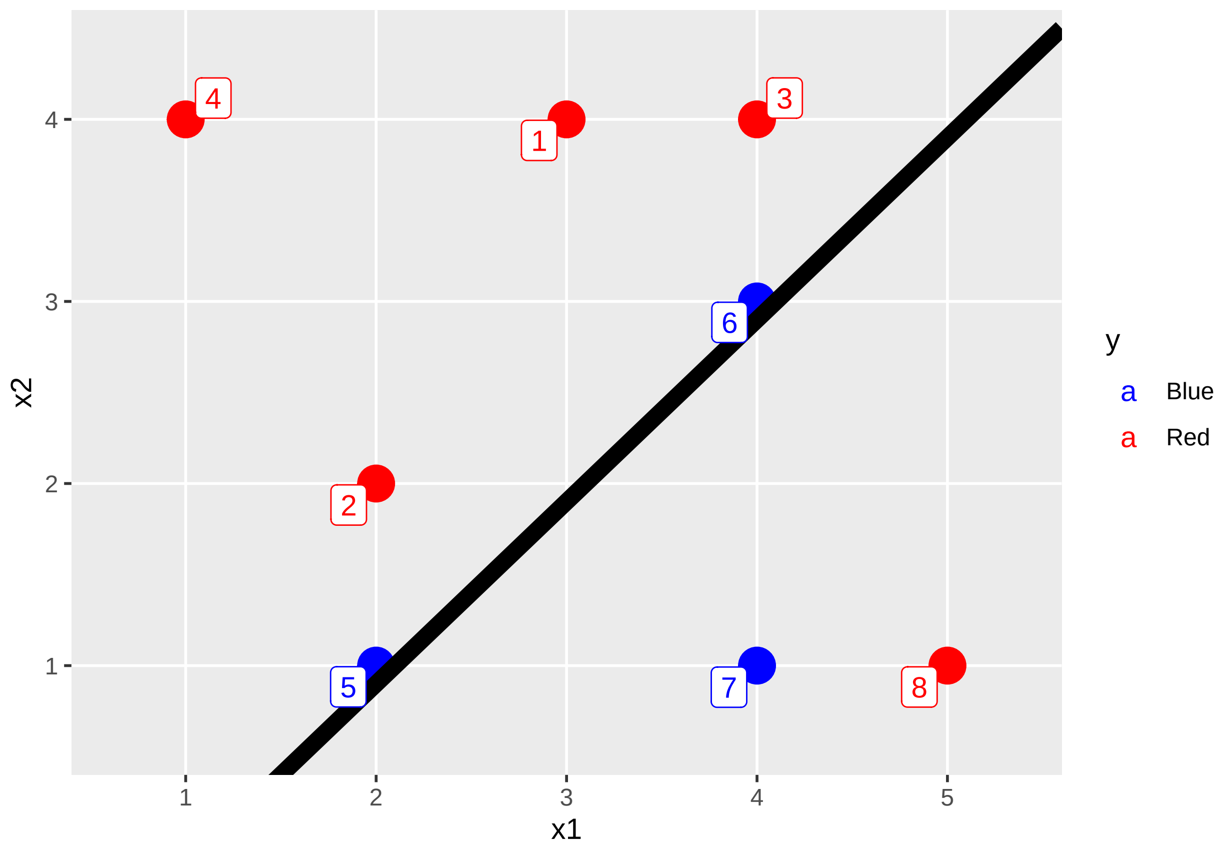
```
points_9.3 +
  geom_abline(intercept = -1.1,
              slope = 1,
              size = 3)
```



$-1.1 - X_1 + X_2 > 0$

(h)

```
df_9.3 %>%
  rbind(c(8, 5, 1, "Red")) %>%
  ggplot(aes(x1, x2, color = y)) +
  geom_point(size = 6) +
  geom_abline(intercept = -1.1,
              slope = 1,
              size = 3) +
  geom_label_repel(aes(label = obs)) +
  scale_color_manual(values = c("Blue", "Red"))
```



(9.7)

```
auto <- as_tibble(Auto)
```

(a)

```
mpg_median <- median(auto$mpg)

auto <- auto %>%
  mutate(high_mileage = ifelse(mpg > mpg_median, 1, 0))
```

(b)

```
tune_linear <- tune(svm, high_mileage ~ ., data = auto, kernel = "linear", ranges = list(cost = c(.01,

summary(tune_linear)
```

```
##
```

```
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.07467893
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-02 0.08392114 0.02695357
## 2 1e-01 0.07855388 0.02526868
## 3 1e+00 0.07467893 0.01853852
## 4 5e+00 0.08364994 0.02389411
## 5 1e+01 0.09145801 0.02547538
## 6 1e+02 0.12656543 0.04668958
```

Errors for various costs in the table above. The lowest error was at cost = 1.

(c)

```r
tune_poly <- tune(svm, high_mileage ~ ., data = auto, kernel = "polynomial", ranges = list(cost = c(0.1

summary(tune_poly)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    15      2
##
## - best performance: 0.2952512
##
## - Detailed performance results:
##    cost degree     error dispersion
## 1   0.1      2 0.4802780 0.08644157
## 2   0.5      2 0.4700671 0.08509175
## 3   1.0      2 0.4577319 0.08386686
## 4   3.0      2 0.4130300 0.08263519
## 5  15.0      2 0.2952512 0.07346642
## 6   0.1      3 0.4819385 0.08665578
## 7   0.5      3 0.4782763 0.08599366
## 8   1.0      3 0.4737479 0.08520332
## 9   3.0      3 0.4560220 0.08238401
## 10 15.0      3 0.3629959 0.07409030
## 11  0.1      4 0.4828261 0.08681829
```

```
## 12  0.5      4 0.4827537 0.08681089
## 13  1.0      4 0.4826632 0.08680169
## 14  3.0      4 0.4823016 0.08676546
## 15 15.0      4 0.4800871 0.08654456
## 16  0.1      5 0.4828416 0.08681982
## 17  0.5      5 0.4828311 0.08681853
## 18  1.0      5 0.4828180 0.08681691
## 19  3.0      5 0.4827656 0.08681045
## 20 15.0      5 0.4824515 0.08677195
```

Best cost for type polynomial is at cost = 15.

```r
tune_radial <- tune(svm, high_mileage ~ ., data = auto, kernel = "radial", ranges = list(cost = c(0.1,

summary(tune_radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    15      2
##
## - best performance: 0.07251967
##
## - Detailed performance results:
##    cost degree      error dispersion
## 1   0.1      2 0.09852772 0.02127592
## 2   0.5      2 0.08359851 0.02670191
## 3   1.0      2 0.08021264 0.02951717
## 4   3.0      2 0.07758403 0.03351797
## 5  15.0      2 0.07251967 0.03489359
## 6   0.1      3 0.09852772 0.02127592
## 7   0.5      3 0.08359851 0.02670191
## 8   1.0      3 0.08021264 0.02951717
## 9   3.0      3 0.07758403 0.03351797
## 10 15.0      3 0.07251967 0.03489359
## 11  0.1      4 0.09852772 0.02127592
## 12  0.5      4 0.08359851 0.02670191
## 13  1.0      4 0.08021264 0.02951717
## 14  3.0      4 0.07758403 0.03351797
## 15 15.0      4 0.07251967 0.03489359
## 16  0.1      5 0.09852772 0.02127592
## 17  0.5      5 0.08359851 0.02670191
## 18  1.0      5 0.08021264 0.02951717
## 19  3.0      5 0.07758403 0.03351797
## 20 15.0      5 0.07251967 0.03489359
```

Best cost for type radial is at cost = 15.

(d)

```
svm_linear <- svm(high_mileage ~ ., data = auto, kernel = "linear", cost = 1)

plot(svm_linear, auto, as.formula(mpg~cylinders))
```

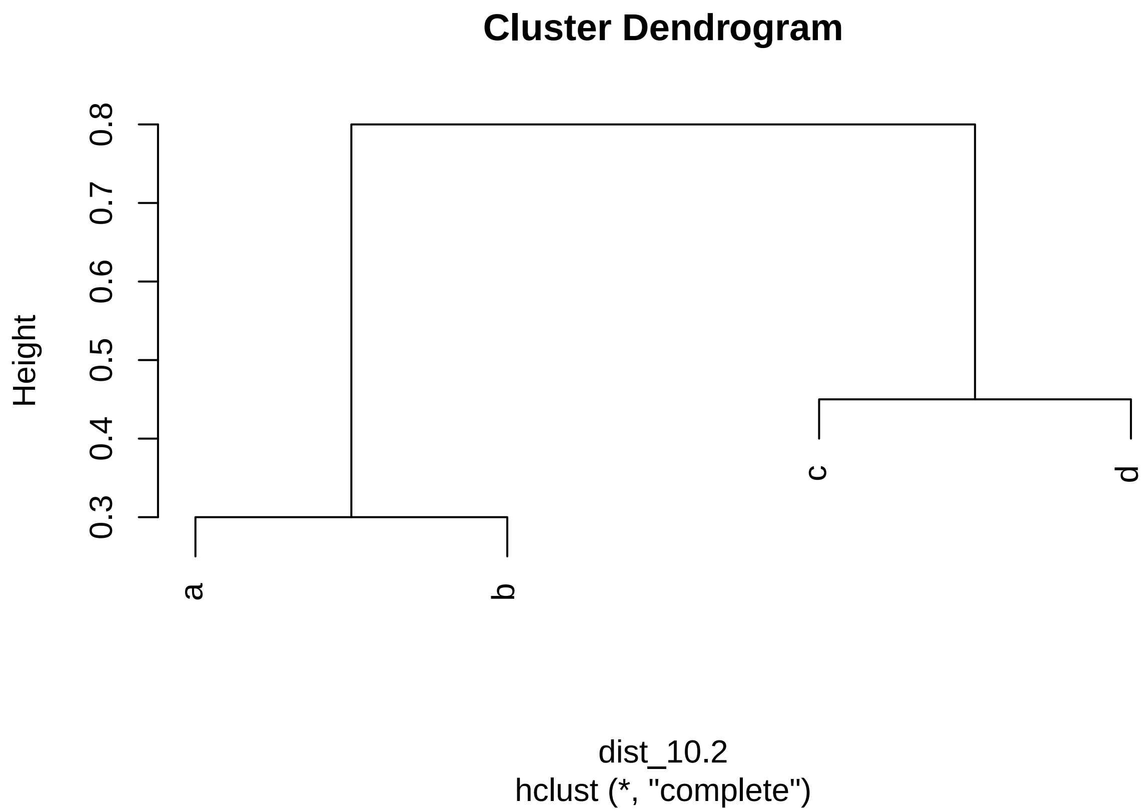
This code follows what is in the book but it's not producing the plot here. Not sure what the issue is.

(10.2)

Will use the `hclust()` function from an earlier chapter to create these trees.

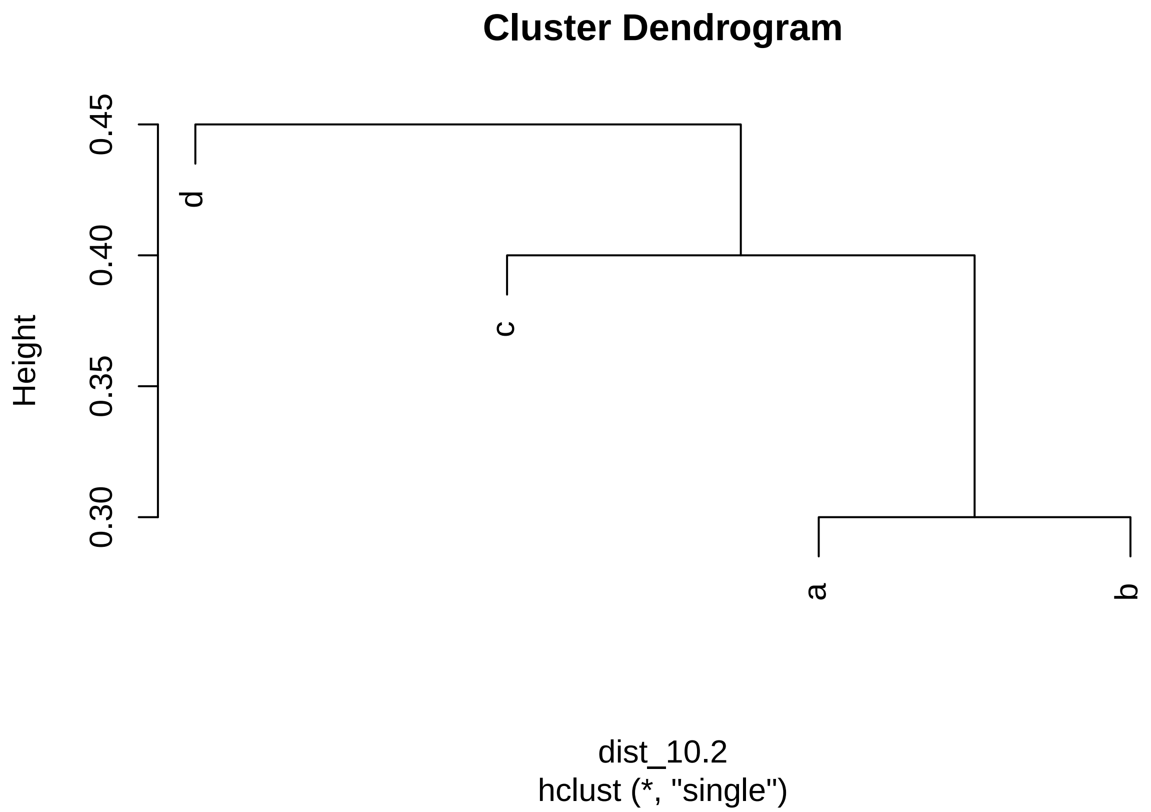
(a)

```
matrix_10.2 <- as.matrix(tribble(
  ~a, ~b, ~c, ~d,
  0, 0.3, 0.4, 0.7,
  0.3, 0, 0.5, 0.8,
  0.4, 0.5, 0, 0.45,
  0.7, 0.8, 0.45, 0
))

dist_10.2 <- as.dist(matrix_10.2)

plot(hclust(dist_10.2, method = "complete"))
```



(b)

```
plot(hclust(dist_10.2, method = "single"))
```



(c)

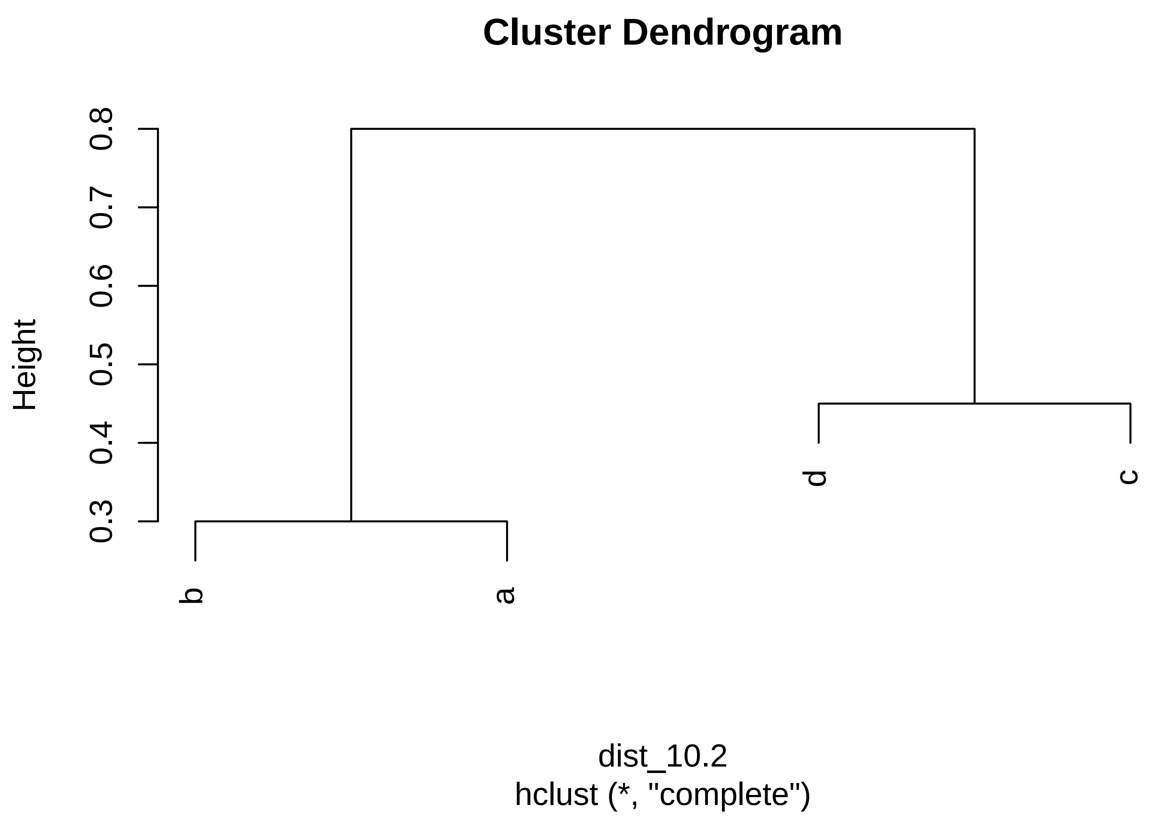
They are split into (a, b) and (c, d)

(d)

They are split into (d) and (c, b, a)

(e)

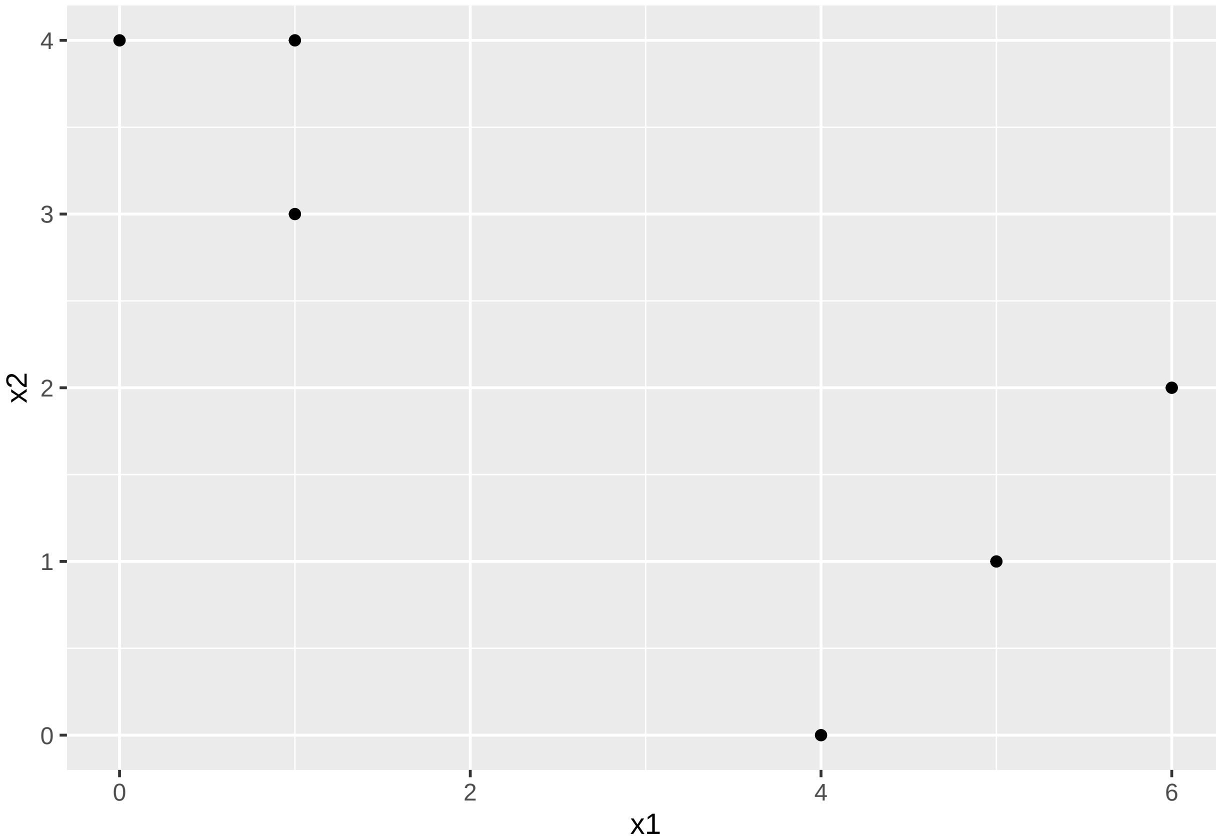
```r
plot(hclust(dist_10.2, method = "complete"),
     labels = c("b", "a", "d", "c"))
```



(10.3)

(a)

```
df_10.3 <- tribble(
  ~obs, ~x1, ~x2,
  1, 1, 4,
  2, 1, 3,
  3, 0, 4,
  4, 5, 1,
  5, 6, 2,
  6, 4, 0
)

df_10.3 %>%
  ggplot(aes(x1, x2)) +
  geom_point()
```



(b)

```
df_10.3 <- df_10.3 %>%
  mutate(cluster = sample(2, 6, replace = TRUE))

df_10.3
```

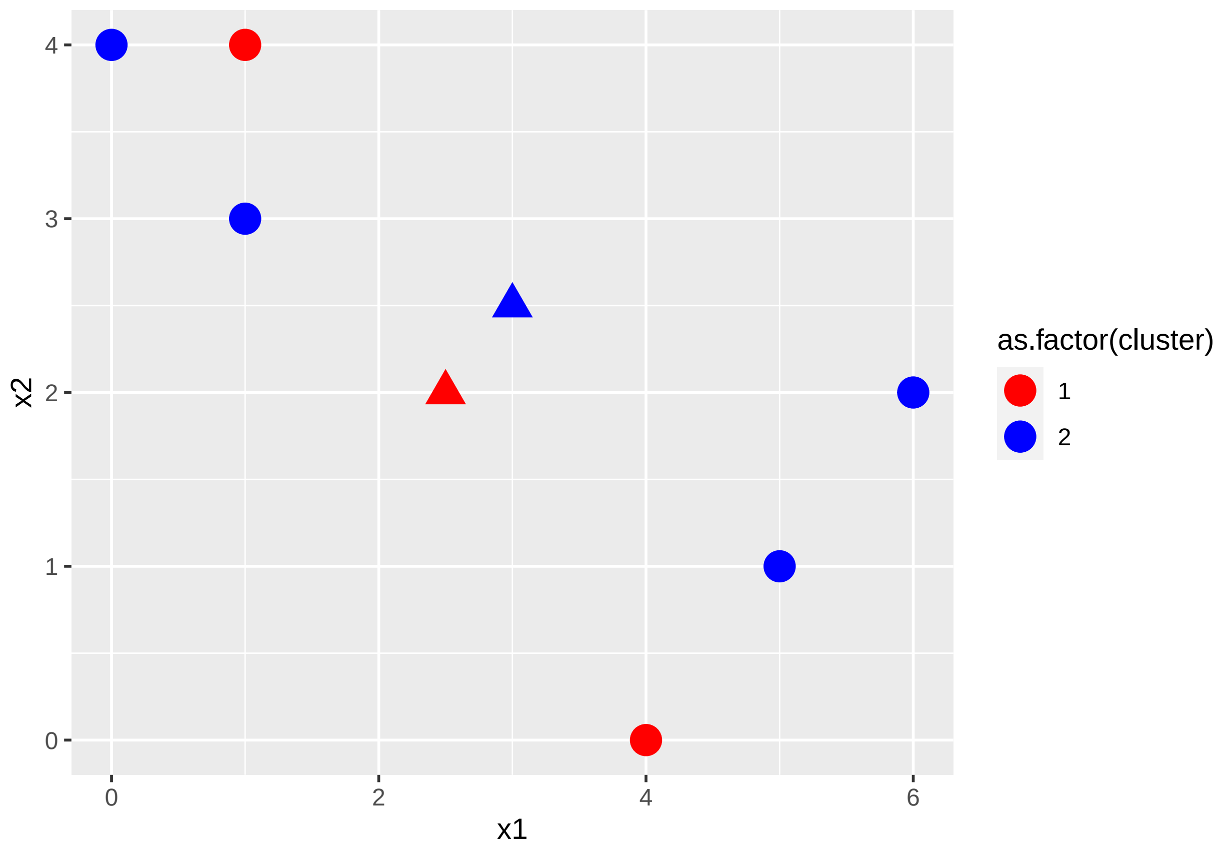
```
## # A tibble: 6 x 4
##     obs    x1    x2 cluster
##   <dbl> <dbl> <dbl>   <int>
## 1     1     1     4       1
## 2     2     1     3       2
## 3     3     0     4       2
## 4     4     5     1       2
## 5     5     6     2       2
## 6     6     4     0       1
```

Clusters in last column.

(c)

```r
# calculate centroids
clust1_x1 <- df_10.3 %>%
  filter(cluster == 1) %>%
  summarise(mean(x1)) %>%
  pull()

clust1_x2 <- df_10.3 %>%
  filter(cluster == 1) %>%
  summarise(mean(x2)) %>%
  pull()

clust2_x1 <- df_10.3 %>%
  filter(cluster == 2) %>%
  summarise(mean(x1)) %>%
  pull()

clust2_x2 <- df_10.3 %>%
  filter(cluster == 2) %>%
  summarise(mean(x2)) %>%
  pull()
```

```r
df_10.3 %>%
  ggplot() +
  geom_point(aes(x1, x2, color = as.factor(cluster)),
             data = df_10.3,
             size = 5) +
  geom_point(aes(x1, x2),
             data = tribble(~x1, ~x2,
                            clust1_x1, clust1_x2),
             color = "red",
             size = 5,
             shape = "triangle") +
  geom_point(aes(x1, x2),
             data = tribble(~x1, ~x2,
                            clust2_x1, clust2_x2),
             color = "blue",
             size = 5,
             shape = "triangle") +
  scale_color_manual(values = c("red", "blue"))
```

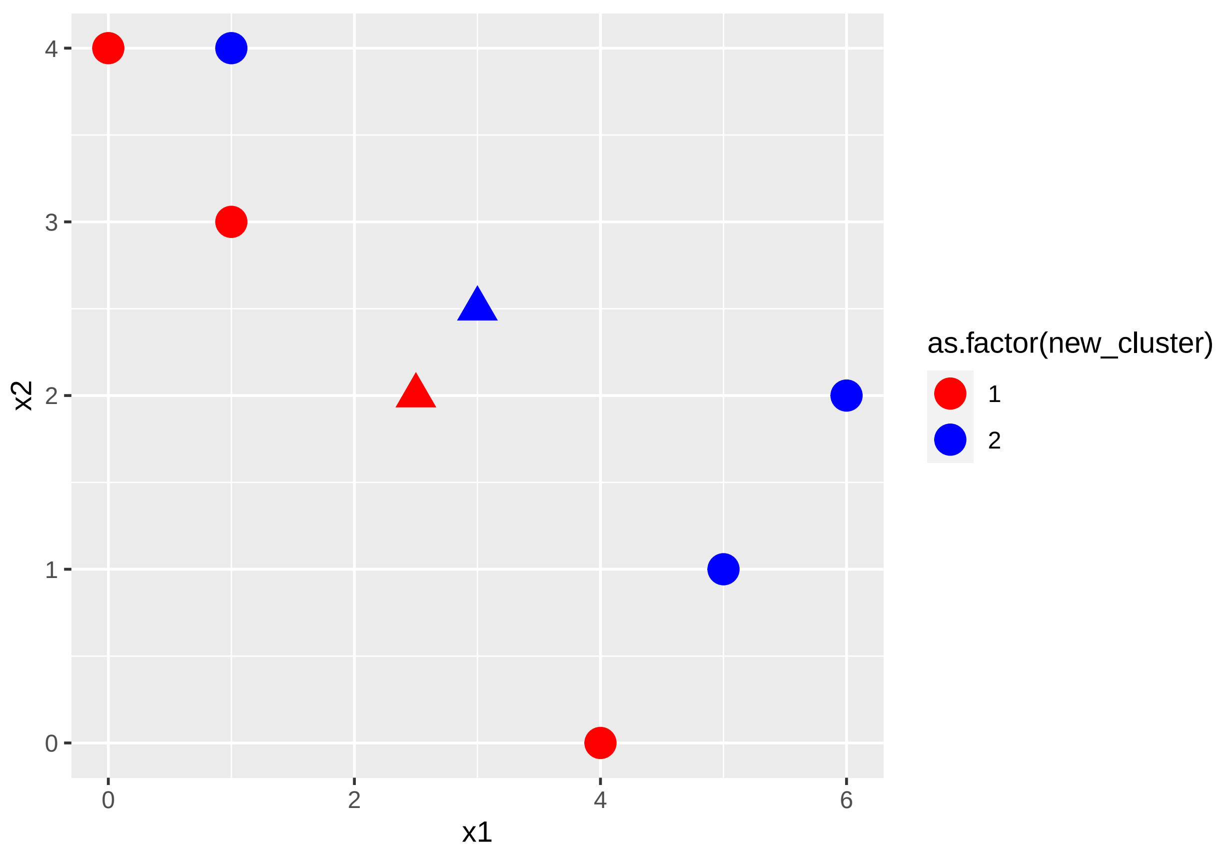


(d)

```
df_10.3 <- df_10.3 %>%
  mutate(center_clust1_x1 = clust1_x1,
         center_clust1_x2 = clust1_x2,
         center_clust2_x1 = clust2_x1,
         center_clust2_x2 = clust2_x2,
         dist_to_centerclust1 = sqrt((center_clust1_x1 - x1)^2 + (center_clust1_x2 - x2)^2),
         dist_to_centerclust2 = sqrt((center_clust2_x1 - x1)^2 + (center_clust2_x2 - x2)^2),
         new_cluster = ifelse(dist_to_centerclust1 < dist_to_centerclust2, 1, 2))

df_10.3
```

```
## # A tibble: 6 x 11
##     obs    x1    x2 cluster center_clust1_x1 center_clust1_x2 center_clust2_x1
##   <dbl> <dbl> <dbl>   <int>            <dbl>            <dbl>            <dbl>
## 1     1     1     4       1              2.5                2                3
## 2     2     1     3       2              2.5                2                3
## 3     3     0     4       2              2.5                2                3
## 4     4     5     1       2              2.5                2                3
## 5     5     6     2       2              2.5                2                3
## 6     6     4     0       1              2.5                2                3
## # ... with 4 more variables: center_clust2_x2 <dbl>,
## #   dist_to_centerclust1 <dbl>, dist_to_centerclust2 <dbl>, new_cluster <dbl>
```

```r
df_10.3 %>%
  ggplot() +
  geom_point(aes(x1, x2, color = as.factor(new_cluster)),
             data = df_10.3,
             size = 5) +
  geom_point(aes(x1, x2),
             data = tribble(~x1, ~x2,
                            clust1_x1, clust1_x2),
             color = "red",
             size = 5,
             shape = "triangle") +
  geom_point(aes(x1, x2),
             data = tribble(~x1, ~x2,
                            clust2_x1, clust2_x2),
             color = "blue",
             size = 5,
             shape = "triangle") +
  scale_color_manual(values = c("red", "blue"))
```



(e)

```r
# calculate centroids
clust1new_x1 <- df_10.3 %>%
```

```r
  filter(new_cluster == 1) %>%
  summarise(mean(x1)) %>%
  pull()

clust1new_x2 <- df_10.3 %>%
  filter(new_cluster == 1) %>%
  summarise(mean(x2)) %>%
  pull()

clust2new_x1 <- df_10.3 %>%
  filter(new_cluster == 2) %>%
  summarise(mean(x1)) %>%
  pull()

clust2new_x2 <- df_10.3 %>%
  filter(new_cluster == 1) %>%
  summarise(mean(x2)) %>%
  pull()
```

(f)

(10.9)

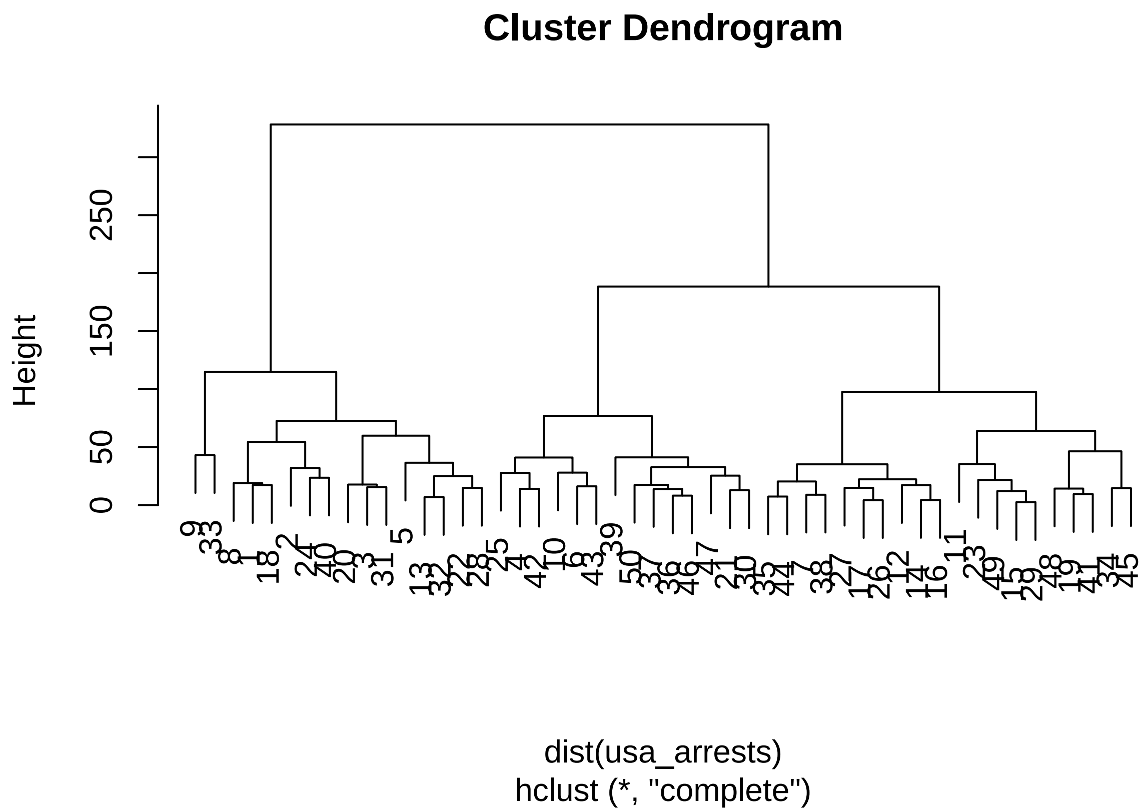
(a)

```r
usa_arrests <- as_tibble(USArrests %>%
                           rownames_to_column(var = "state"))
```

```r
hc_complete <- hclust(dist(usa_arrests), method = "complete")
```

```
## Warning in dist(usa_arrests): NAs introduced by coercion
```

```r
plot(hc_complete)
```



(b)

```r
usa_arrests <- usa_arrests %>%
  mutate(cluster = cutree(hc_complete, 3))

# show each cluster individually
usa_arrests %>%
  filter(cluster == 1)
```

```
## # A tibble: 16 x 6
##    state          Murder Assault UrbanPop  Rape cluster
##    <chr>           <dbl>   <int>    <int> <dbl>   <int>
##  1 Alabama          13.2     236       58  21.2       1
##  2 Alaska           10       263       48  44.5       1
##  3 Arizona           8.1     294       80  31         1
##  4 California        9       276       91  40.6       1
##  5 Delaware          5.9     238       72  15.8       1
##  6 Florida          15.4     335       80  31.9       1
##  7 Illinois         10.4     249       83  24         1
##  8 Louisiana        15.4     249       66  22.2       1
##  9 Maryland         11.3     300       67  27.8       1
## 10 Michigan         12.1     255       74  35.1       1
## 11 Mississippi      16.1     259       44  17.1       1
## 12 Nevada           12.2     252       81  46         1
```

```
## 13 New Mexico       11.4      285        70  32.1        1
## 14 New York         11.1      254        86  26.1        1
## 15 North Carolina   13        337        45  16.1        1
## 16 South Carolina   14.4      279        48  22.5        1
```

```r
usa_arrests %>%
  filter(cluster == 2)
```

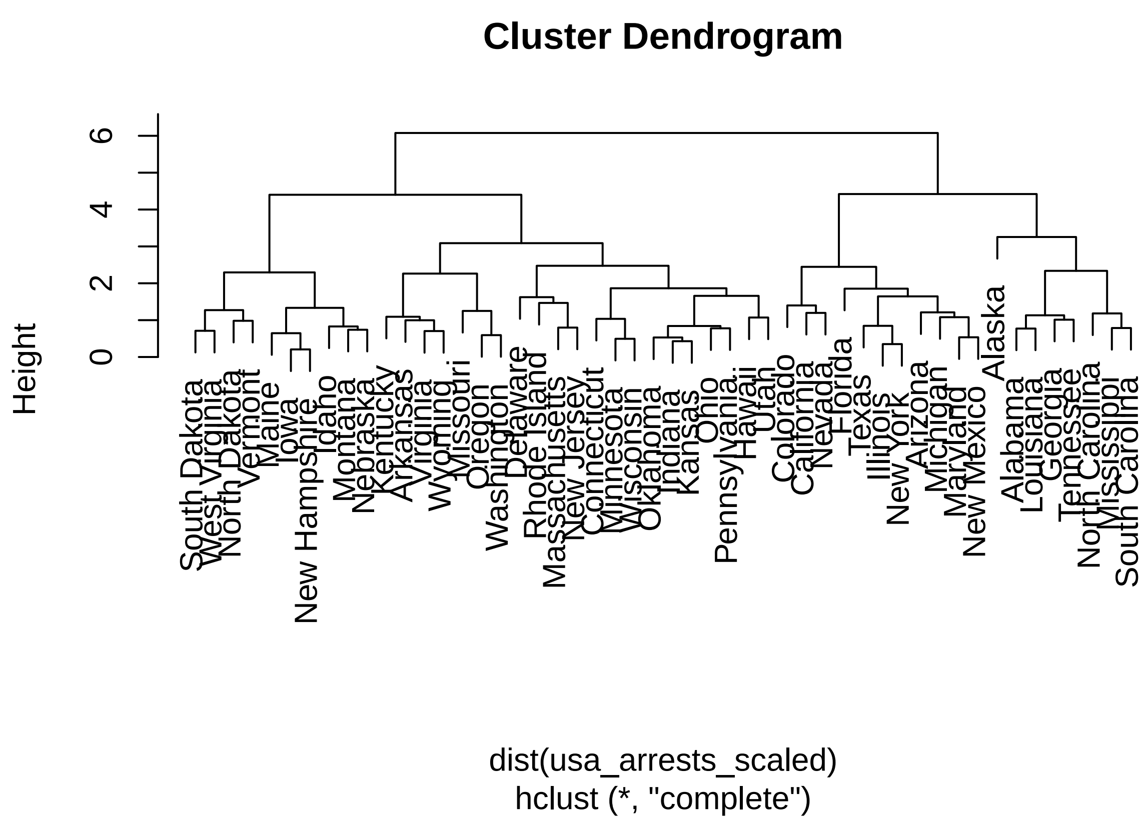
```
## # A tibble: 14 x 6
##    state          Murder Assault UrbanPop  Rape cluster
##    <chr>           <dbl>   <int>    <int> <dbl>   <int>
##  1 Arkansas          8.8     190       50  19.5       2
##  2 Colorado          7.9     204       78  38.7       2
##  3 Georgia          17.4     211       60  25.8       2
##  4 Massachusetts     4.4     149       85  16.3       2
##  5 Missouri          9       178       70  28.2       2
##  6 New Jersey        7.4     159       89  18.8       2
##  7 Oklahoma          6.6     151       68  20         2
##  8 Oregon            4.9     159       67  29.3       2
##  9 Rhode Island      3.4     174       87   8.3       2
## 10 Tennessee        13.2     188       59  26.9       2
## 11 Texas            12.7     201       80  25.5       2
## 12 Virginia          8.5     156       63  20.7       2
## 13 Washington        4       145       73  26.2       2
## 14 Wyoming           6.8     161       60  15.6       2
```

```r
usa_arrests %>%
  filter(cluster == 3)
```

```
## # A tibble: 20 x 6
##    state          Murder Assault UrbanPop  Rape cluster
##    <chr>           <dbl>   <int>    <int> <dbl>   <int>
##  1 Connecticut       3.3     110       77  11.1       3
##  2 Hawaii            5.3      46       83  20.2       3
##  3 Idaho             2.6     120       54  14.2       3
##  4 Indiana           7.2     113       65  21         3
##  5 Iowa              2.2      56       57  11.3       3
##  6 Kansas            6       115       66  18         3
##  7 Kentucky          9.7     109       52  16.3       3
##  8 Maine             2.1      83       51   7.8       3
##  9 Minnesota         2.7      72       66  14.9       3
## 10 Montana           6       109       53  16.4       3
## 11 Nebraska          4.3     102       62  16.5       3
## 12 New Hampshire     2.1      57       56   9.5       3
## 13 North Dakota      0.8      45       44   7.3       3
## 14 Ohio              7.3     120       75  21.4       3
## 15 Pennsylvania      6.3     106       72  14.9       3
## 16 South Dakota      3.8      86       45  12.8       3
## 17 Utah              3.2     120       80  22.9       3
## 18 Vermont           2.2      48       32  11.2       3
## 19 West Virginia     5.7      81       39   9.3       3
## 20 Wisconsin         2.6      53       66  10.8       3
```

(c)

```
usa_arrests_scaled <- scale(usa_arrests %>%
                              column_to_rownames("state") %>%
                              select(-cluster))

hc_scaled <- hclust(dist(usa_arrests_scaled), method = "complete")

plot(hc_scaled)
```



(d)

(10.10)

(a)

```
mat_10.10 <- matrix(nrow = 60, ncol = 50)

for (i in 1:20) {
  mat_10.10[i, ] <- c(rnorm(n = 50, mean = 0))
}
```

```r
for (i in 21:40) {
  mat_10.10[i, ] <- c(rnorm(n = 50, mean = 10))
}

for (i in 41:60) {
  mat_10.10[i, ] <- c(rnorm(n = 50, mean = 20))
}

# df_10.10 <- as_tibble(mat_10.10) %>%
#   mutate(cluster = c(rep(1, 20), rep(2, 20), rep(3, 20)))
#
# df_10.10
```

(b)

```r
pca_out <- prcomp(mat_10.10)

summary(pca_out)
```

```
## Importance of components:
##                            PC1     PC2    PC3     PC4    PC5     PC6     PC7
## Standard deviation     58.2488 1.81124 1.7623 1.67961 1.6562 1.63688 1.57092
## Proportion of Variance  0.9858 0.00095 0.0009 0.00082 0.0008 0.00078 0.00072
## Cumulative Proportion   0.9858 0.98675 0.9877 0.98847 0.9893 0.99005 0.99076
##                            PC8     PC9    PC10    PC11    PC12   PC13    PC14
## Standard deviation     1.47709 1.46836 1.38370 1.37133 1.33518 1.3090 1.25871
## Proportion of Variance 0.00063 0.00063 0.00056 0.00055 0.00052 0.0005 0.00046
## Cumulative Proportion  0.99140 0.99202 0.99258 0.99313 0.99364 0.9941 0.99460
##                           PC15   PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.21778 1.1776 1.12766 1.10267 1.04338 1.03407 0.96604
## Proportion of Variance 0.00043 0.0004 0.00037 0.00035 0.00032 0.00031 0.00027
## Cumulative Proportion  0.99503 0.9954 0.99581 0.99616 0.99648 0.99679 0.99706
##                           PC22    PC23    PC24    PC25   PC26    PC27    PC28
## Standard deviation     0.93948 0.92643 0.89280 0.86282 0.8283 0.80702 0.78794
## Proportion of Variance 0.00026 0.00025 0.00023 0.00022 0.0002 0.00019 0.00018
## Cumulative Proportion  0.99731 0.99756 0.99779 0.99801 0.9982 0.99840 0.99858
##                           PC29    PC30    PC31    PC32   PC33   PC34    PC35
## Standard deviation     0.76807 0.75548 0.68248 0.64086 0.5902 0.5742 0.54906
## Proportion of Variance 0.00017 0.00017 0.00014 0.00012 0.0001 0.0001 0.00009
## Cumulative Proportion  0.99875 0.99892 0.99905 0.99917 0.9993 0.9994 0.99946
##                           PC36    PC37    PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.53631 0.50252 0.49289 0.42975 0.41670 0.38113 0.34721
## Proportion of Variance 0.00008 0.00007 0.00007 0.00005 0.00005 0.00004 0.00004
## Cumulative Proportion  0.99954 0.99961 0.99968 0.99974 0.99979 0.99983 0.99987
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.32571 0.32002 0.27220 0.24514 0.21687 0.19454 0.15301
## Proportion of Variance 0.00003 0.00003 0.00002 0.00002 0.00001 0.00001 0.00001
## Cumulative Proportion  0.99990 0.99993 0.99995 0.99996 0.99998 0.99999 1.00000
##                          PC50
## Standard deviation     0.1112
## Proportion of Variance 0.0000
```

```
## Cumulative Proportion  1.0000
```

(c)

```r
km_out <- kmeans(mat_10.10, 3, nstart = 20)

km_out$cluster
```

```
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

As in the dataset, points are clustered into different classes 20 by 20 by 20. They are perfectly learned here as I gave them a large mean separation.

(d)

```r
km_out2 <- kmeans(mat_10.10, 2, nstart = 20)

km_out2$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

The second class which was previously "2" is now a part of class "1", while the other class (previously 3) is now all 2.

(e)

```r
km_out4 <- kmeans(mat_10.10, 4, nstart = 20)

km_out4$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
## [39] 4 4 3 2 3 3 3 2 2 3 2 3 2 2 2 3 3 3 3 2 2 3
```

The cluster which was originally 2, rows 21:40 from the data, is split into two different clusters now. The other two remain the same as original, perfectly separated into their own clusters.

(f)

```r
km_outpca <- kmeans(pca_out$x[, 1:2], 3, nstart = 20)

km_outpca$cluster
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Like we saw at the start, split perfectly into 3 clusters.

(g)

```
mat_10.10_scaled <- scale(mat_10.10)

km_outscaled <- kmeans(mat_10.10_scaled, 3, nstart = 20)

km_outscaled$cluster
```

```
##  [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [39] 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

They are still perfectly separated.