Lab 9

Ravi Patel | ID: 400133223

20/03/2021

```
library(tidyverse)
```

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

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

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

```
seeds <- read.table(
  "https://archive.ics.uci.edu/ml/machine-learning-databases/00236/seeds_dataset.txt"
  )
colnames(seeds) <- c("area",
                     "perimeter",
                     "compactness",
                     "length_of_kernel",
                     "width_of_kernel",
                     "asy_coeff",
                     "length_of_kernel_groove",
                     "Class")
summary(seeds)
```

```
##       area         perimeter      compactness     length_of_kernel
##  Min.   :10.59   Min.   :12.41   Min.   :0.8081   Min.   :4.899
##  1st Qu.:12.27   1st Qu.:13.45   1st Qu.:0.8569   1st Qu.:5.262
##  Median :14.36   Median :14.32   Median :0.8734   Median :5.524
##  Mean   :14.85   Mean   :14.56   Mean   :0.8710   Mean   :5.629
##  3rd Qu.:17.30   3rd Qu.:15.71   3rd Qu.:0.8878   3rd Qu.:5.980
##  Max.   :21.18   Max.   :17.25   Max.   :0.9183   Max.   :6.675
##  width_of_kernel   asy_coeff       length_of_kernel_groove     Class
##  Min.   :2.630   Min.   :0.7651   Min.   :4.519           Min.   :1
##  1st Qu.:2.944   1st Qu.:2.5615   1st Qu.:5.045           1st Qu.:1
##  Median :3.237   Median :3.5990   Median :5.223           Median :2
##  Mean   :3.259   Mean   :3.7002   Mean   :5.408           Mean   :2
##  3rd Qu.:3.562   3rd Qu.:4.7687   3rd Qu.:5.877           3rd Qu.:3
##  Max.   :4.033   Max.   :8.4560   Max.   :6.550           Max.   :3
```

```r
cor(dplyr::select(seeds, -Class))
```

```
##                              area  perimeter compactness length_of_kernel
## area                    1.0000000  0.9943409   0.6082884        0.9499854
## perimeter               0.9943409  1.0000000   0.5292436        0.9724223
## compactness             0.6082884  0.5292436   1.0000000        0.3679151
## length_of_kernel        0.9499854  0.9724223   0.3679151        1.0000000
## width_of_kernel         0.9707706  0.9448294   0.7616345        0.8604149
## asy_coeff              -0.2295723 -0.2173404  -0.3314709       -0.1715624
## length_of_kernel_groove 0.8636927  0.8907839   0.2268248        0.9328061
##                        width_of_kernel   asy_coeff length_of_kernel_groove
## area                         0.9707706 -0.22957233              0.86369275
## perimeter                    0.9448294 -0.21734037              0.89078390
## compactness                  0.7616345 -0.33147087              0.22682482
## length_of_kernel             0.8604149 -0.17156243              0.93280609
## width_of_kernel              1.0000000 -0.25803655              0.74913147
## asy_coeff                   -0.2580365  1.00000000             -0.01107902
## length_of_kernel_groove      0.7491315 -0.01107902              1.00000000
```

```r
dim(seeds)
```

```
## [1] 210   8
```

```r
x <- seeds %>%
  dplyr::select(-Class) %>%
  scale()

set.seed(1)

seeds_train_index <- seeds %>%
  mutate(ind = 1:nrow(seeds)) %>%
  group_by(Class) %>%
  mutate(n = n()) %>%
  sample_frac(size = .75, weight = n) %>%
  ungroup() %>%
  pull(ind)

library(nnet)
class_labels <- pull(seeds, Class) %>%
  class.ind()


seeds_train <- x[seeds_train_index, ]
train_class <- class_labels[seeds_train_index,]
seeds_test <- x[-seeds_train_index, ]
test_class <- class_labels[-seeds_train_index,]

nn_seeds <- nnet(
  x = seeds_train,
  y = train_class,
  size = 4,
  decay = 0,
```

```r
  softmax = TRUE,
  maxit=500
  )
```

```
## # weights:  47
## initial  value 179.079752
## iter  10 value 10.357187
## iter  20 value 0.304073
## iter  30 value 0.002143
## iter  40 value 0.000138
## iter  40 value 0.000061
## iter  40 value 0.000061
## final  value 0.000061
## converged
```

```r
nn_pred <- predict(nn_seeds, seeds_test,
                   type="class")

tab_seeds <- table(slice(
  seeds,
  -seeds_train_index) %>% pull(Class),
  nn_pred)

1-sum(diag(tab_seeds))/sum(tab_seeds)
```

```
## [1] 0.1111111
```

```
library(nnet)
library(MASS)
```

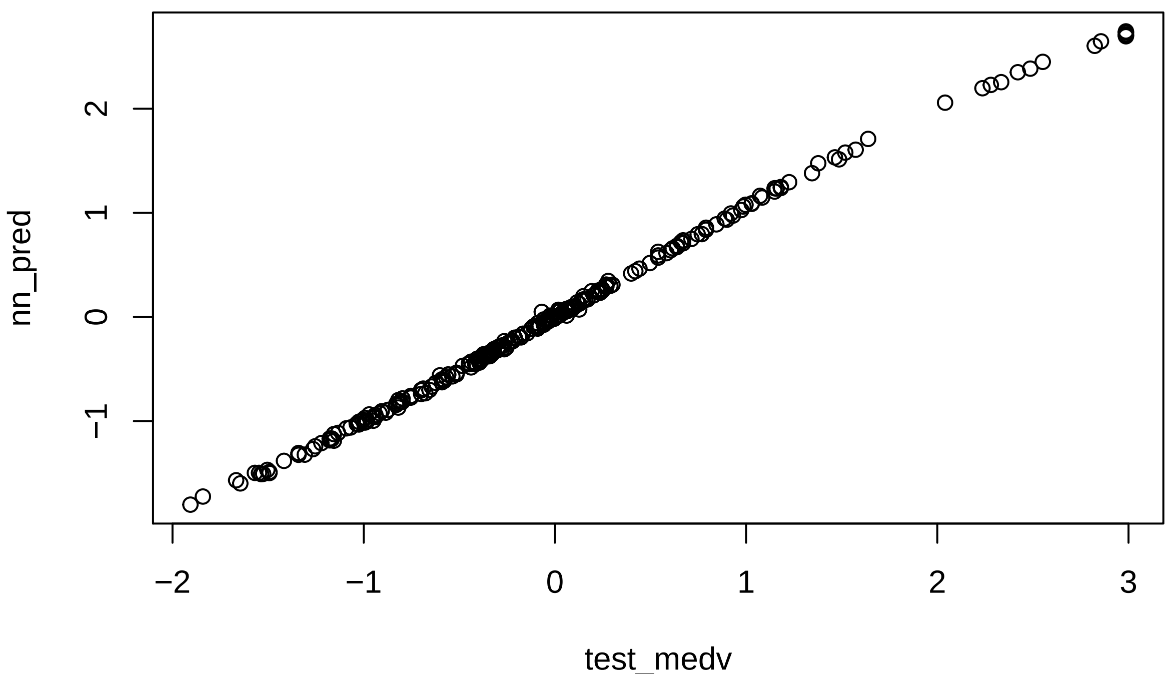
```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
train_Boston <- sample(
  1:nrow(Boston),
  nrow(Boston)/2
  )

x <- scale(Boston)

Boston_train <- x[train_Boston, ]
train_medv <- x[train_Boston, "medv"]
Boston_test <- x[-train_Boston, ]
test_medv <- x[-train_Boston, "medv"]

nn_Boston <- nnet(
  Boston_train,
  train_medv,
  size=10,
  decay=1,
  softmax=FALSE,
  maxit=1000,
  linout=TRUE
  )
```

```
## # weights:  161
## initial  value 469.211580
## iter  10 value 39.116735
## iter  20 value 22.164051
## iter  30 value 17.626264
## iter  40 value 14.619830
## iter  50 value 12.655570
## iter  60 value 11.292161
## iter  70 value 10.583592
## iter  80 value 10.254760
## iter  90 value 10.097962
## iter 100 value 10.015590
## iter 110 value 9.948224
## iter 120 value 9.917840
## iter 130 value 9.905889
## iter 140 value 9.901823
## iter 150 value 9.900874
## iter 160 value 9.900509
## iter 170 value 9.900410
## iter 180 value 9.900337
```

```
## iter 190 value 9.900141
## iter 200 value 9.900086
## iter 210 value 9.900065
## final  value 9.900062
## converged
```

```
nn_pred <- predict(
  nn_Boston,
  Boston_test,
  type="raw"
  )

plot(test_medv, nn_pred)
```




```
mean((test_medv - nn_pred)^2)
```

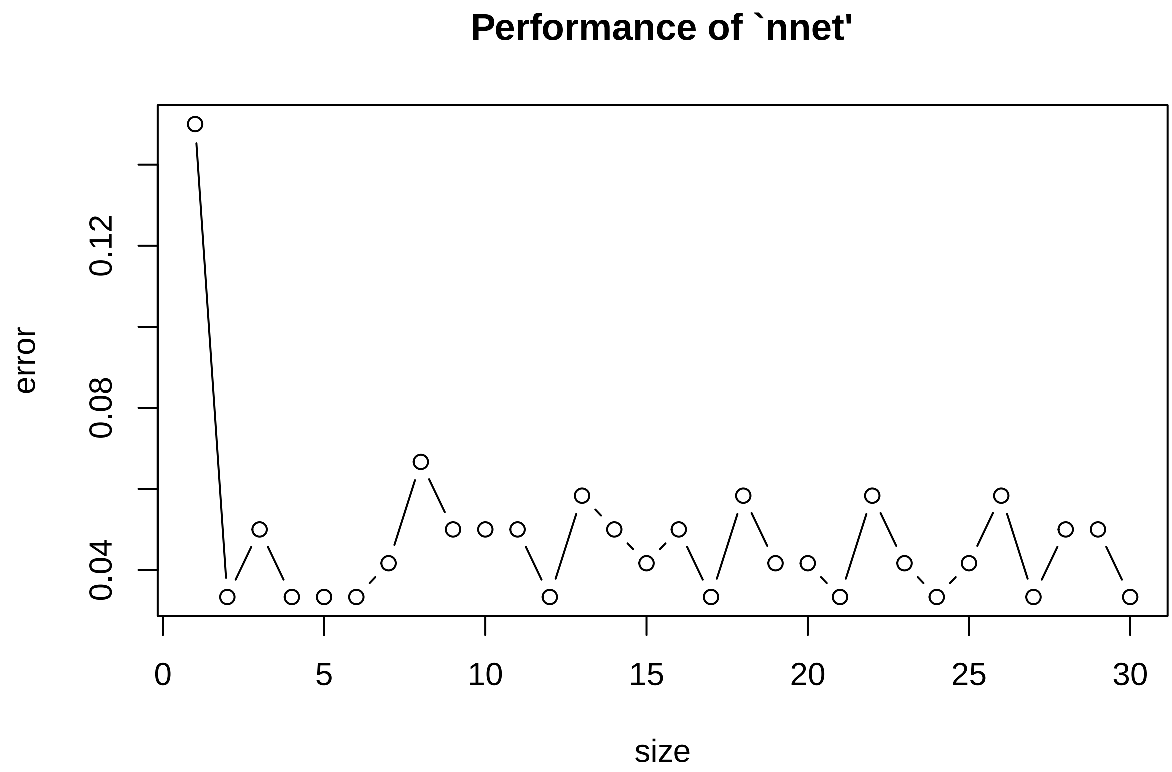
```
## [1] 0.003687532
```

```
library(e1071)
library(cluster)
set.seed(1)

data("iris")

Species <- pull(iris, Species)

xy <- dplyr::select(iris, -Species) %>%
  scale() %>%
  data.frame() %>%
  mutate(Species = Species) # scale predictors

iris_train_index <- iris %>%
  mutate(ind = 1:nrow(iris)) %>%
  group_by(Species) %>%
  mutate(n = n()) %>%
  sample_frac(size = .8, weight = n) %>%
  ungroup() %>%
  pull(ind)

iris_train <- slice(xy, iris_train_index)
iris_test <- slice(xy, -iris_train_index)
class_labels <- pull(xy, Species) %>%
  class.ind()

iris_nnet1 <- tune.nnet(
  Species~.,
  data = iris_train,
  size = 1:30,
  tunecontrol = tune.control(sampling = "cross",cross=5)
  )

head(summary(iris_nnet1))
```

```
## $best.parameters
##   size
## 2    2
##
## $best.performance
## [1] 0.03333333
##
## $method
## [1] "nnet"
##
## $nparcomb
## [1] 30
##
## $train.ind
## $train.ind$`(0.881,24.8]`
##  [1]  40  83  90  35 111 112 120  78  22  70  28  37  61  46  67  71 116  44  49
## [20] 117  56  89  50   7  20 100  80  99  16   2 118  65  79 101  41  77 107  13
## [39] 109 114  82  19  17  57  11  31 115  74  95  55  45  52  68 119   9  97  81
```

```
## [58] 113 108  85  32  87  94  12  30  14  62   6  72  64  38 102  91   3 104  69
## [77]  54   5  88  33  84  47   8   4  98  18  27  36  63 110  25  21  66  73  23
## [96]  75
##
## $train.ind$`(24.8,48.6]`
##  [1] 106  96 103  60  51  93  34  10   1  43  59  26  15  58  29  24  42  48  76
## [20]  39 105  53  92  86  20 100  80  99  16   2 118  65  79 101  41  77 107  13
## [39] 109 114  82  19  17  57  11  31 115  74  95  55  45  52  68 119   9  97  81
## [58] 113 108  85  32  87  94  12  30  14  62   6  72  64  38 102  91   3 104  69
## [77]  54   5  88  33  84  47   8   4  98  18  27  36  63 110  25  21  66  73  23
## [96]  75
##
## $train.ind$`(48.6,72.4]`
##  [1] 106  96 103  60  51  93  34  10   1  43  59  26  15  58  29  24  42  48  76
## [20]  39 105  53  92  86  40  83  90  35 111 112 120  78  22  70  28  37  61  46
## [39]  67  71 116  44  49 117  56  89  50   7  95  55  45  52  68 119   9  97  81
## [58] 113 108  85  32  87  94  12  30  14  62   6  72  64  38 102  91   3 104  69
## [77]  54   5  88  33  84  47   8   4  98  18  27  36  63 110  25  21  66  73  23
## [96]  75
##
## $train.ind$`(72.4,96.2]`
##  [1] 106  96 103  60  51  93  34  10   1  43  59  26  15  58  29  24  42  48  76
## [20]  39 105  53  92  86  40  83  90  35 111 112 120  78  22  70  28  37  61  46
## [39]  67  71 116  44  49 117  56  89  50   7  20 100  80  99  16   2 118  65  79
## [58] 101  41  77 107  13 109 114  82  19  17  57  11  31 115  74  91   3 104  69
## [77]  54   5  88  33  84  47   8   4  98  18  27  36  63 110  25  21  66  73  23
## [96]  75
##
## $train.ind$`(96.2,120]`
##  [1] 106  96 103  60  51  93  34  10   1  43  59  26  15  58  29  24  42  48  76
## [20]  39 105  53  92  86  40  83  90  35 111 112 120  78  22  70  28  37  61  46
## [39]  67  71 116  44  49 117  56  89  50   7  20 100  80  99  16   2 118  65  79
## [58] 101  41  77 107  13 109 114  82  19  17  57  11  31 115  74  95  55  45  52
## [77]  68 119   9  97  81 113 108  85  32  87  94  12  30  14  62   6  72  64  38
## [96] 102
##
##
## $sampling
## [1] "5-fold cross validation"
```

```
plot(iris_nnet1)
```




```r
library(nnet)
nn_iris <- nnet(
  x = dplyr::select(iris_train, -Species),
  y = class_labels[iris_train_index, ],
  size = iris_nnet1$best.parameters[1,1],
  decay = 0,
  softmax = TRUE
  )
```

```
## # weights:  19
## initial  value 139.787195
## iter  10 value 51.659855
## iter  20 value 12.382653
## iter  30 value 2.538123
## iter  40 value 0.820028
## iter  50 value 0.000596
## iter  60 value 0.000139
## final  value 0.000086
## converged
```

```r
nn_pred <- predict(
  nn_iris,
  dplyr::select(iris_test, -Species),
  type="class"
  )
```

```r
tab <- table(pull(iris_test, Species),
  nn_pred
  )

tab
```

```
##             nn_pred
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          2         8
```

```r
1- sum(diag(tab))/sum(tab)
```

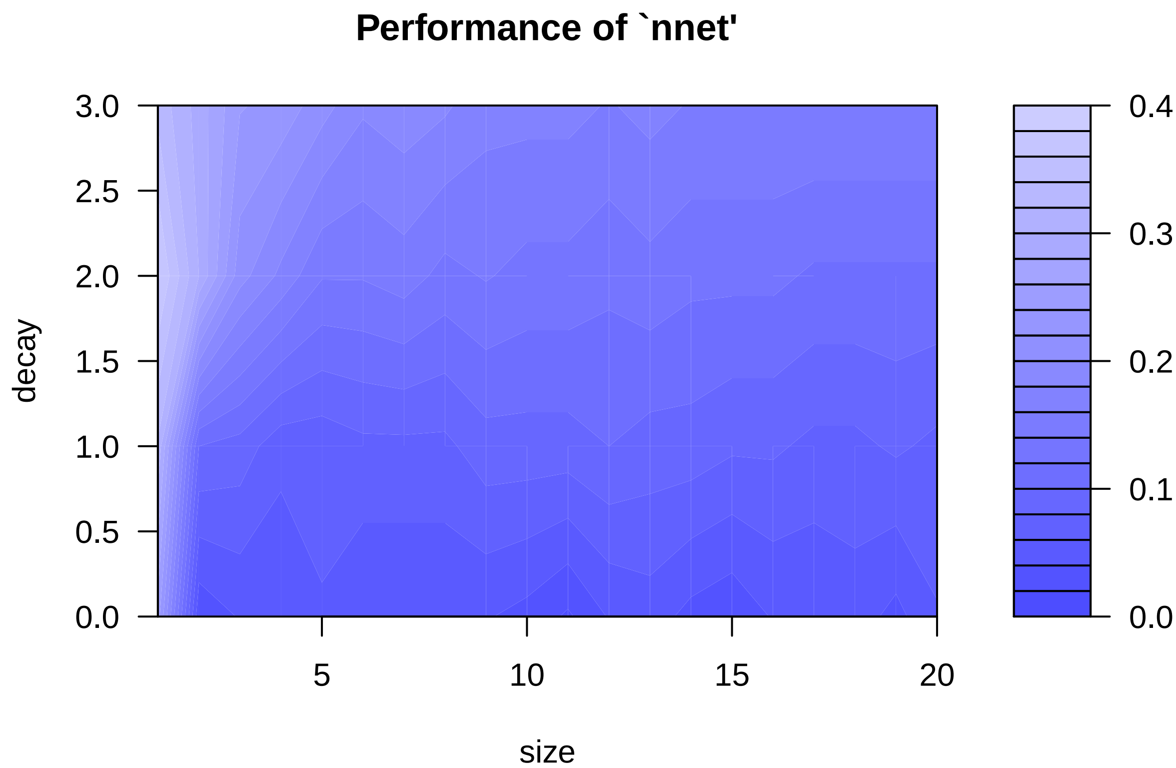
```
## [1] 0.06666667
```

```r
set.seed(1)

iris_nnet2 <- tune.nnet(
  Species~.,
  data = iris_train,
  size = 1:20,
  decay = 0:3,
  tunecontrol = tune.control(sampling = "cross",cross=5)
  )

head(summary(iris_nnet2))
```

```
## $best.parameters
##    size decay
## 11   11     0
##
## $best.performance
## [1] 0.01666667
##
## $method
## [1] "nnet"
##
## $nparcomb
## [1] 80
##
## $train.ind
## $train.ind$`(0.881,24.8]`
##  [1]  99  44 102  33  84  35  70 105  42  38  20  28  86  95  90  40  83  25 113
## [20] 119 111  88   6  24  32 114   2  45  18  22  65  13  81  94  48  63  23  46
## [39]  92  77  29  66  67  56 101  80  62  93  69 108  31 116  17   9  57  60  19
## [58]  26  30  72  53 110  10 118  11  27  75  15  50 103  91  16  47  12 104 112
## [77]   8  49   3  98  64  55  71  96  36   4 115   5  52  41  61 120  78  58 107
## [96]  76
##
## $train.ind$`(24.8,48.6]`
```

```
##  [1]  68  39   1  34  87  43  14  82  59  51  97  85  21 106  54  74   7  73  79
## [20] 109  37  89 100 117  32 114   2  45  18  22  65  13  81  94  48  63  23  46
## [39]  92  77  29  66  67  56 101  80  62  93  69 108  31 116  17   9  57  60  19
## [58]  26  30  72  53 110  10 118  11  27  75  15  50 103  91  16  47  12 104 112
## [77]   8  49   3  98  64  55  71  96  36   4 115   5  52  41  61 120  78  58 107
## [96]  76
##
## $train.ind$`(48.6,72.4]`
##  [1]  68  39   1  34  87  43  14  82  59  51  97  85  21 106  54  74   7  73  79
## [20] 109  37  89 100 117  99  44 102  33  84  35  70 105  42  38  20  28  86  95
## [39]  90  40  83  25 113 119 111  88   6  24  69 108  31 116  17   9  57  60  19
## [58]  26  30  72  53 110  10 118  11  27  75  15  50 103  91  16  47  12 104 112
## [77]   8  49   3  98  64  55  71  96  36   4 115   5  52  41  61 120  78  58 107
## [96]  76
##
## $train.ind$`(72.4,96.2]`
##  [1]  68  39   1  34  87  43  14  82  59  51  97  85  21 106  54  74   7  73  79
## [20] 109  37  89 100 117  99  44 102  33  84  35  70 105  42  38  20  28  86  95
## [39]  90  40  83  25 113 119 111  88   6  24  32 114   2  45  18  22  65  13  81
## [58]  94  48  63  23  46  92  77  29  66  67  56 101  80  62  93  47  12 104 112
## [77]   8  49   3  98  64  55  71  96  36   4 115   5  52  41  61 120  78  58 107
## [96]  76
##
## $train.ind$`(96.2,120]`
##  [1]  68  39   1  34  87  43  14  82  59  51  97  85  21 106  54  74   7  73  79
## [20] 109  37  89 100 117  99  44 102  33  84  35  70 105  42  38  20  28  86  95
## [39]  90  40  83  25 113 119 111  88   6  24  32 114   2  45  18  22  65  13  81
## [58]  94  48  63  23  46  92  77  29  66  67  56 101  80  62  93  69 108  31 116
## [77]  17   9  57  60  19  26  30  72  53 110  10 118  11  27  75  15  50 103  91
## [96]  16
##
##
## $sampling
## [1] "5-fold cross validation"
```

```
plot(iris_nnet2)
```




```
nn_iris_d_s <- nnet(
  x = dplyr::select(iris_train, -Species),
  y = class_labels[iris_train_index, ],
  size = iris_nnet2$best.parameters[1,1],
  decay = iris_nnet2$best.parameters[1,2],
  softmax = TRUE
  )
```

```
## # weights:  91
## initial  value 164.446139
## iter  10 value 15.814895
## iter  20 value 1.891497
## iter  30 value 0.102615
## final  value 0.000056
## converged
```

```
# Compute test error
nn_pred <- predict(
  nn_iris_d_s,
  dplyr::select(iris_test, -Species),
  type="class"
  )

tab <- table(pull(iris_test, Species),
  nn_pred
```

```r
  )

tab
```

```
##             nn_pred
##              setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         0
##   virginica       0          2         8
```

```r
1- sum(diag(tab))/sum(tab)
```

```
## [1] 0.06666667
```

```r
library(cluster)
library(factoextra) # PCA
```

```
## Warning: package 'factoextra' was built under R version 4.0.4
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

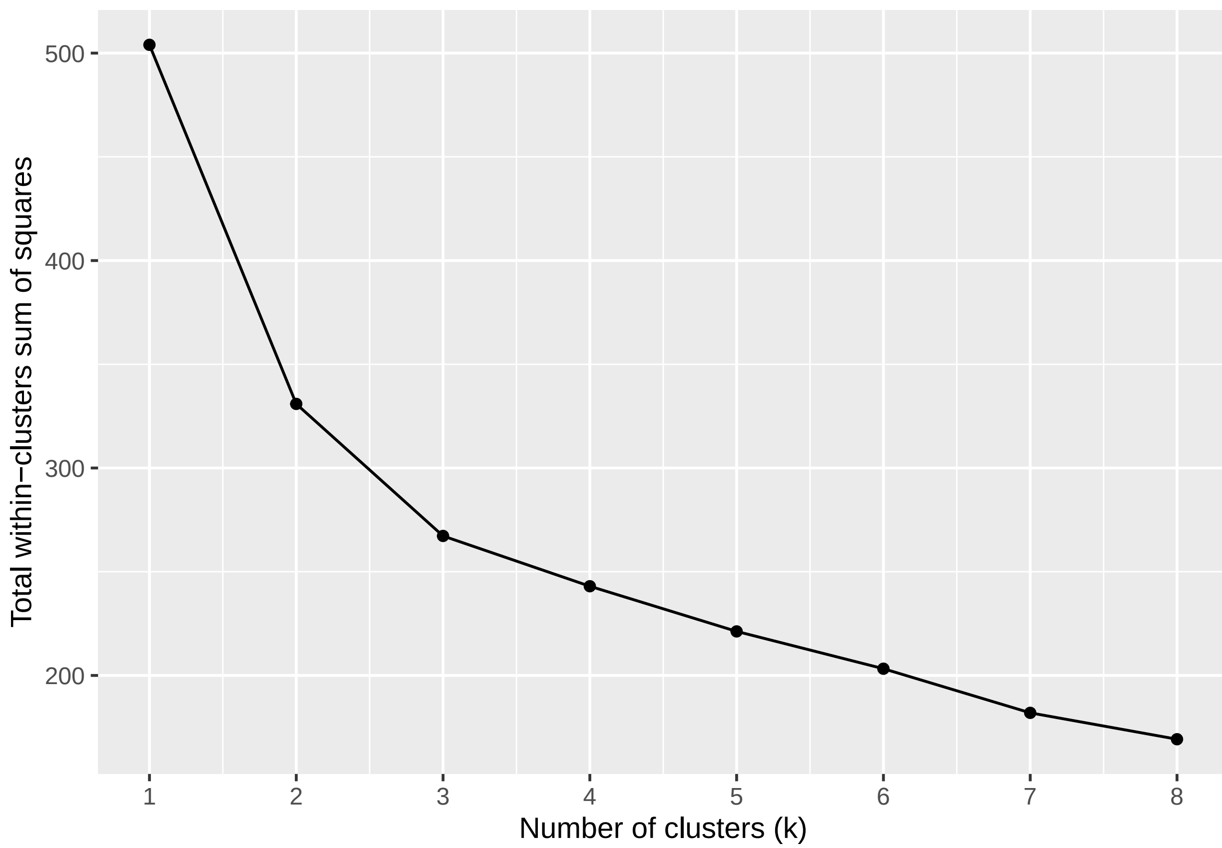
```r
library(pgmm) # coffee data
data("coffee")
set.seed(1)
x <- dplyr::select(coffee, - Variety, - Country)
x_scaled <- scale(x)
kmeans_coffee <- kmeans(x_scaled, 2)
kmeans_coffee$tot.withinss
```

```
## [1] 330.8912
```

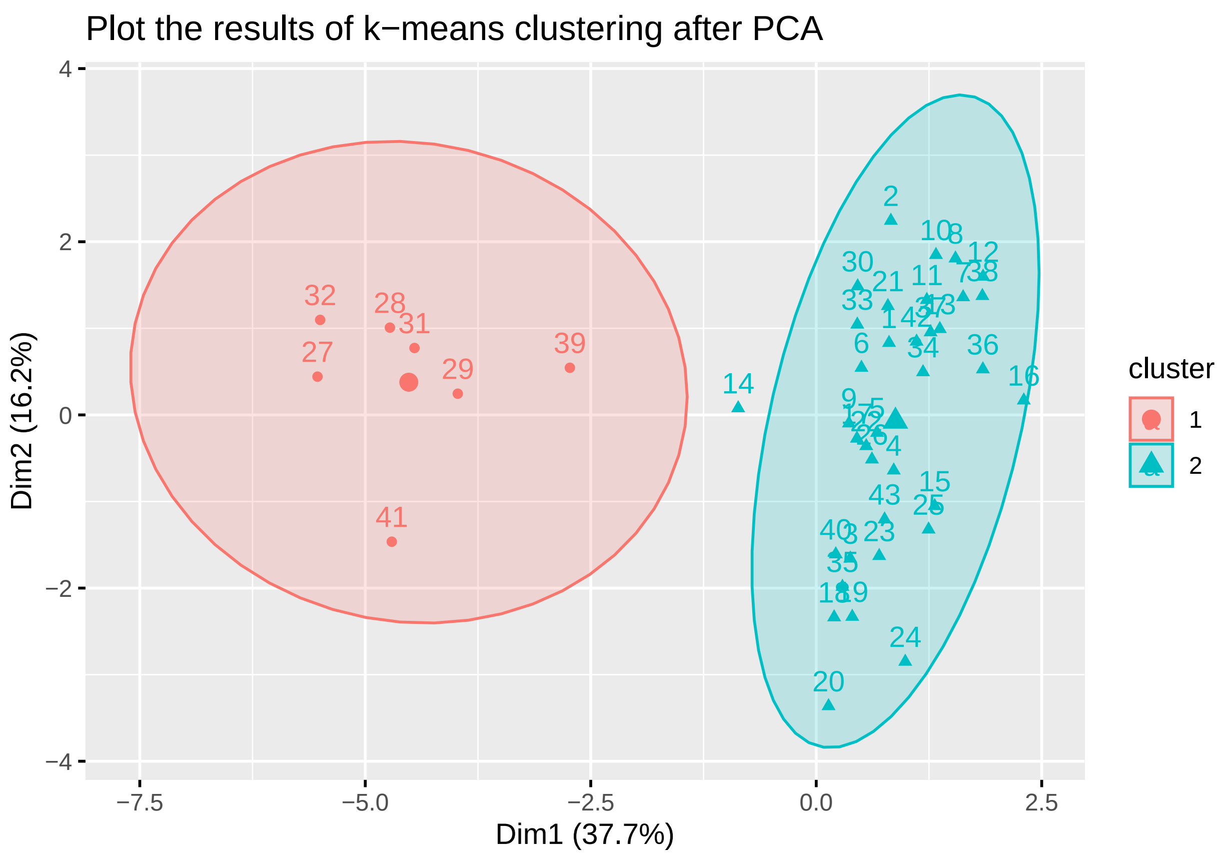
```r
kmeans_coffee <- kmeans(x_scaled, 3)
kmeans_coffee$tot.withinss
```

```
## [1] 267.2453
```

```r
# Let's select K using elbow method
withiclusterss <- function(K,x){
  kmeans(x, K)$tot.withinss
}

K <- 1:8

wcss <- lapply(as.list(K), function(k){
  withiclusterss(k, x_scaled)
}) %>% unlist()

ggplot(tibble(K = K, wcss = wcss), aes(x = K, y = wcss)) +
  geom_point() +
  geom_line() +
  xlab("Number of clusters (k)") +
  ylab("Total within-clusters sum of squares") +
  scale_x_continuous(breaks=c(seq(1,K[length(K)])))
```




```
kmeans_coffee <- kmeans(x_scaled, 2)
fvPCA <- fviz_cluster(kmeans_coffee,
                    x_scaled,
                    ellipse.type = "norm",
                    main = "Plot the results of k-means clustering after PCA")
fvPCA
```




```r
library(cluster)
library(factoextra) # PCA
library(pgmm) # coffee data
data("coffee")
set.seed(1)
x <- dplyr::select(coffee, - Variety, - Country)
x_scaled <- scale(x)
kmeans_coffee <- kmeans(x_scaled, 2)
kmeans_coffee$tot.withinss
```

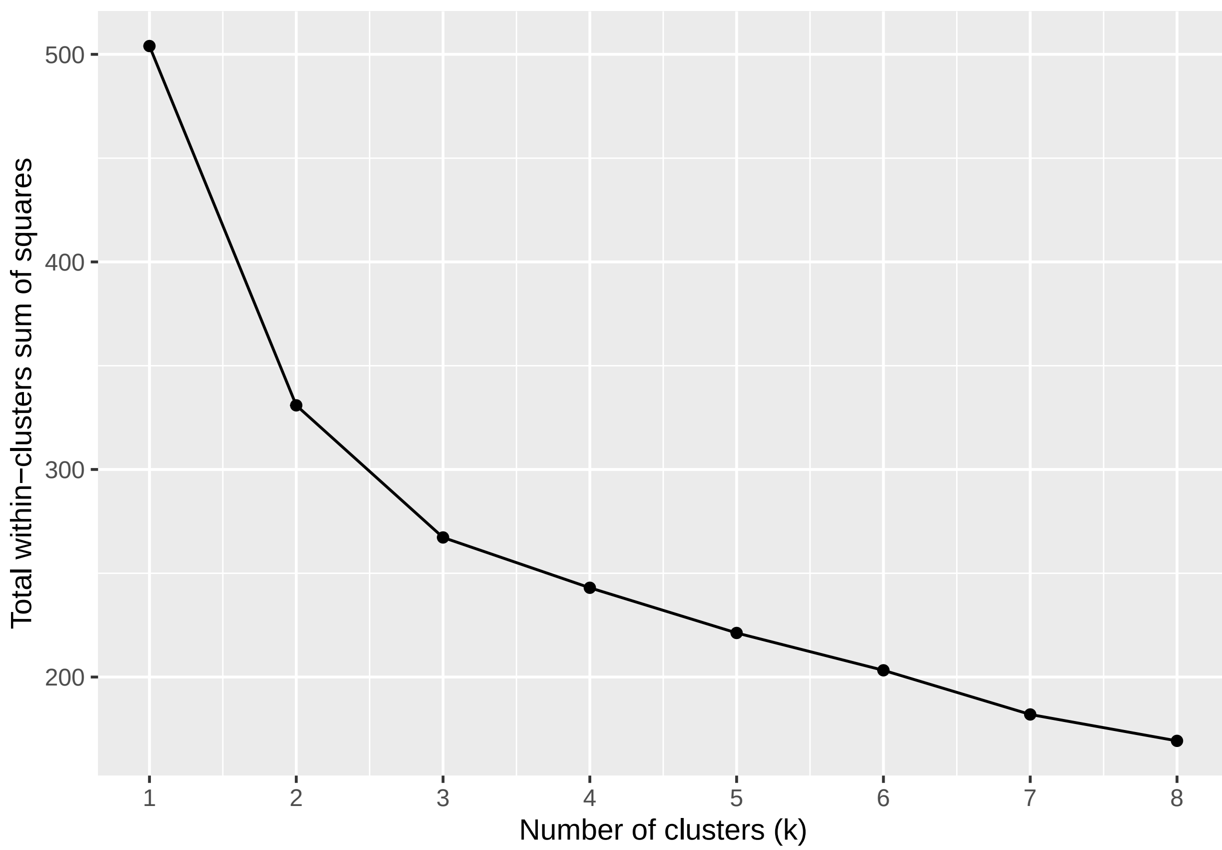
```
## [1] 330.8912
```

```r
kmeans_coffee <- kmeans(x_scaled, 3)
kmeans_coffee$tot.withinss
```

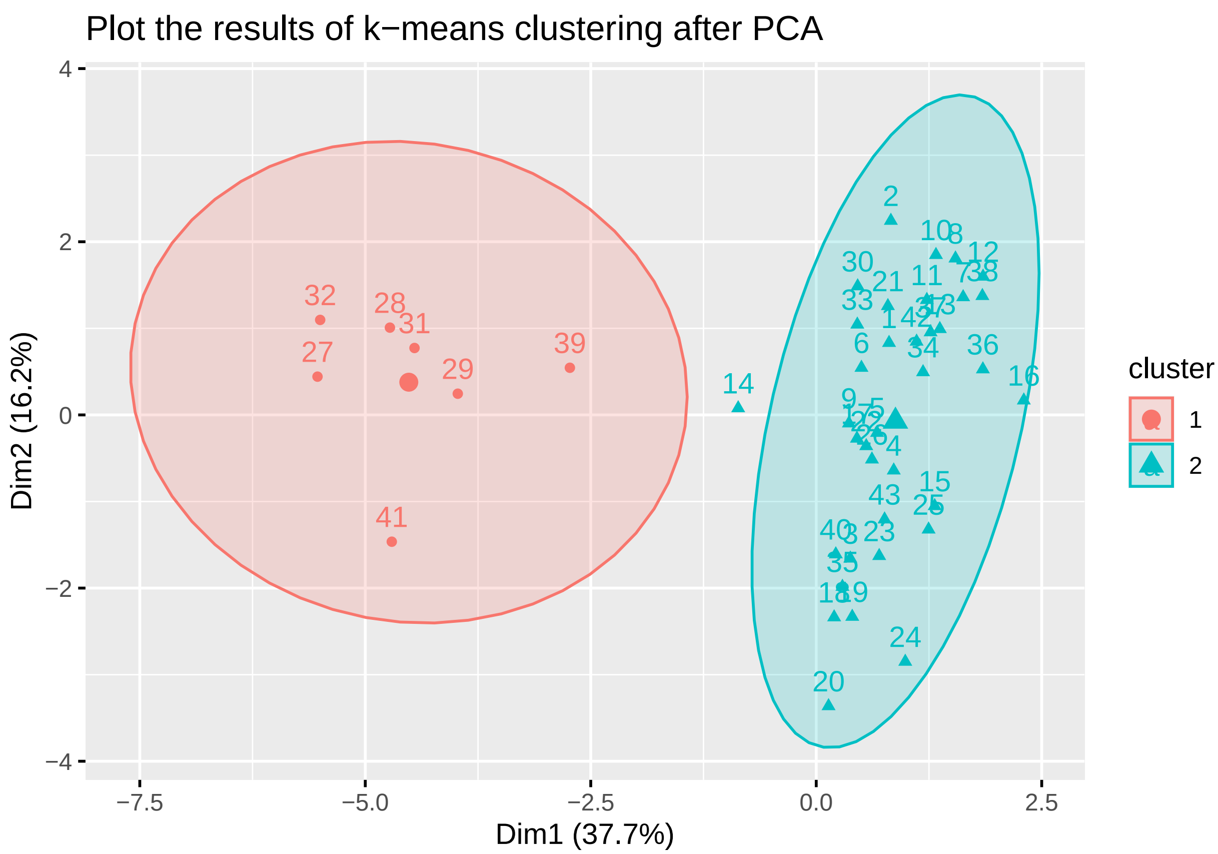
```
## [1] 267.2453
```

```r
# Let's select K using elbow method
withiclusterss <- function(K,x){
  kmeans(x, K)$tot.withinss
}

K <- 1:8
```

```r
wcss <- lapply(as.list(K), function(k){
  withiclusterss(k, x_scaled)
}) %>% unlist()

ggplot(tibble(K = K, wcss = wcss), aes(x = K, y = wcss)) +
  geom_point() +
  geom_line() +
  xlab("Number of clusters (k)") +
  ylab("Total within-clusters sum of squares") +
  scale_x_continuous(breaks=c(seq(1,K[length(K)])))
```




```r
kmeans_coffee <- kmeans(x_scaled, 2)
fvPCA <- fviz_cluster(kmeans_coffee,
                      x_scaled,
                      ellipse.type = "norm",
                      main = "Plot the results of k-means clustering after PCA")
fvPCA
```



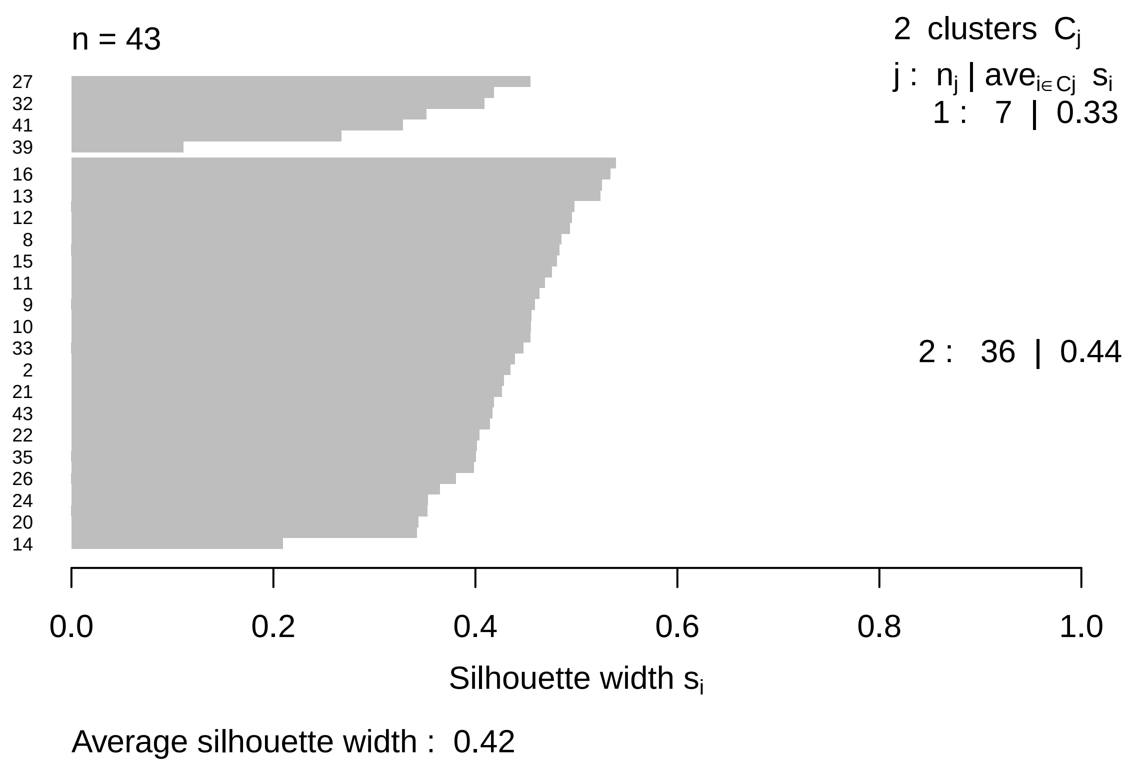

```
si <- silhouette(kmeans_coffee$cluster, dist(x_scaled))
head(si)
```

```
##      cluster neighbor sil_width
## [1,]       2        1 0.5252373
## [2,]       2        1 0.4346060
## [3,]       2        1 0.4143200
## [4,]       2        1 0.4932787
## [5,]       2        1 0.4632535
## [6,]       2        1 0.4832208
```

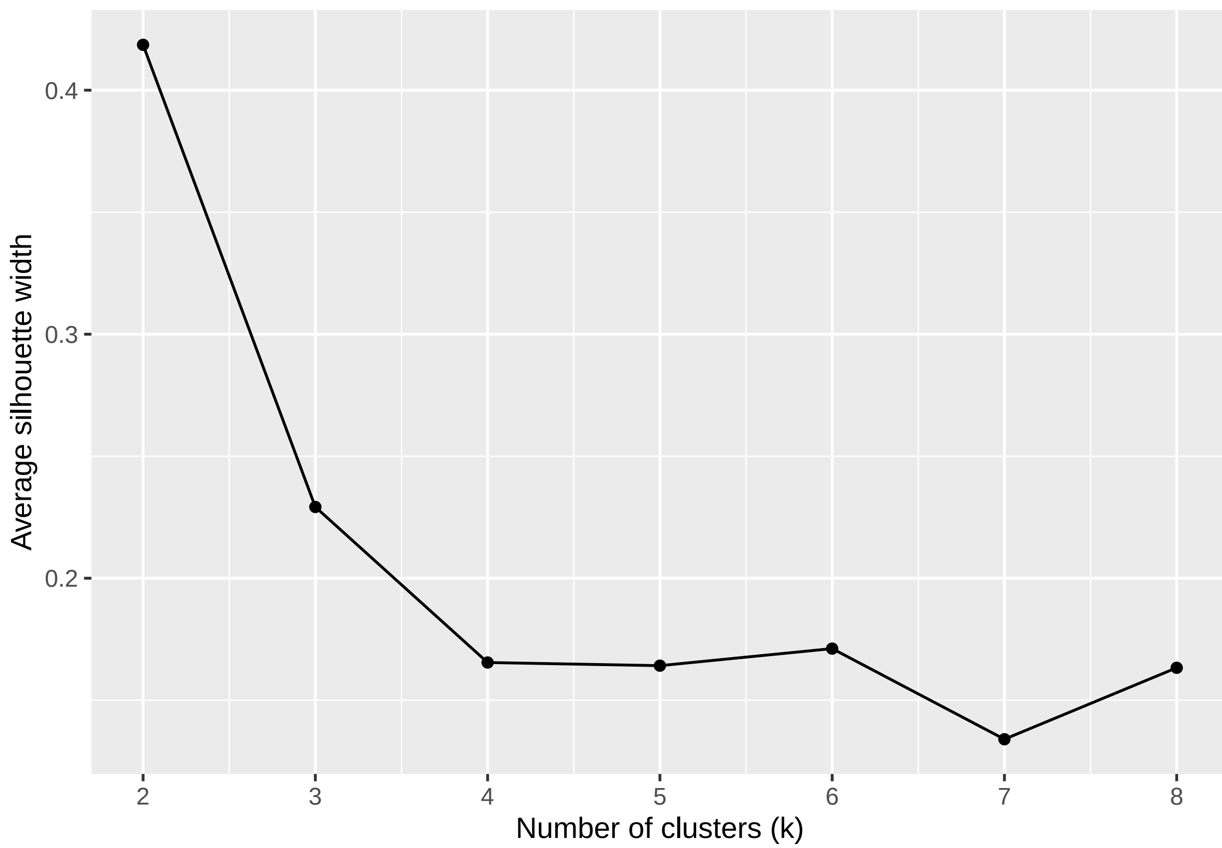
```
#average Silhouette width
mean(si[, 3])
```

```
## [1] 0.4186062
```

```
plot(si, nmax= 80, cex.names=0.6, main = "")
```



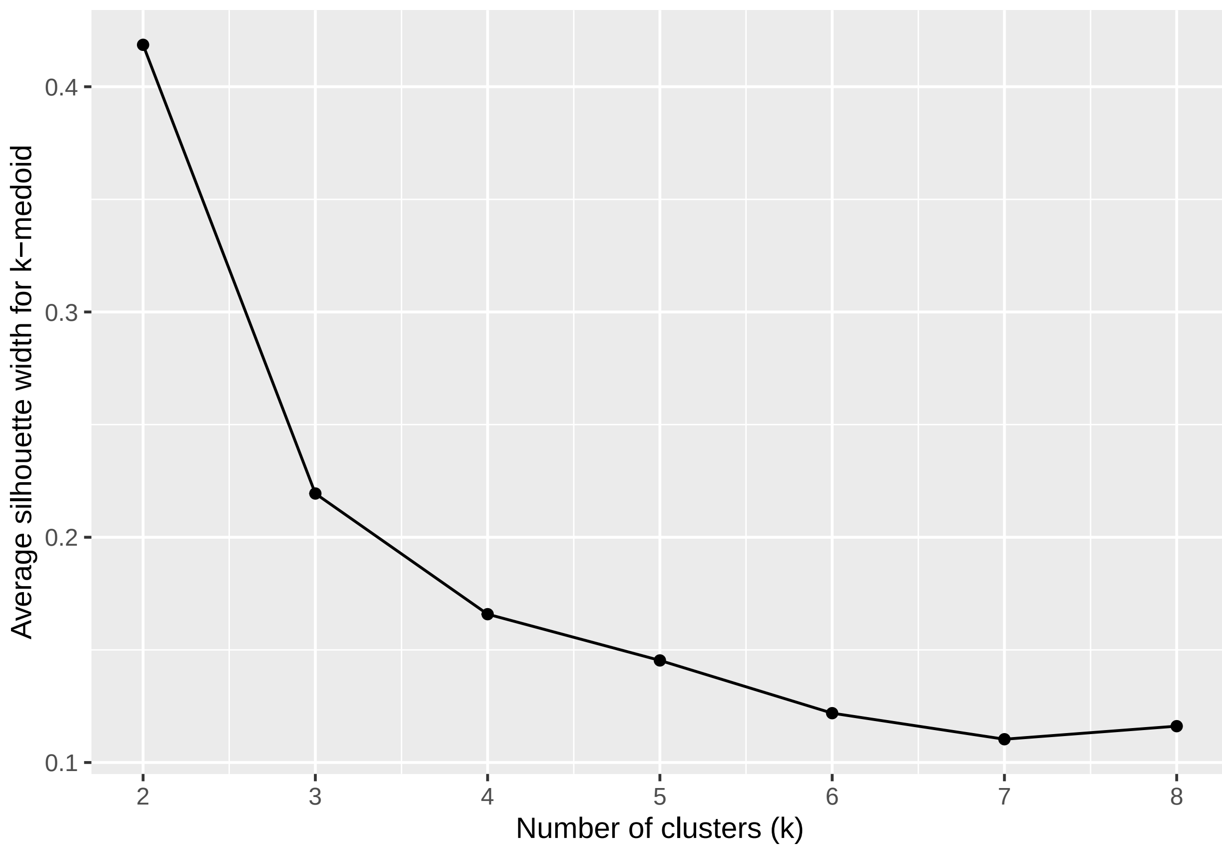

```
# Let's select K using average Silhouette width
avgSilhouette <- function(K,x) {
  km_cl <- kmeans(x, K)
  sil <- silhouette(km_cl$cluster, dist(x))
  return(mean(sil[, 3]))
}

K <- 2:8

avgSil <- numeric()
for(i in K){
  avgSil[(i-1)] <- avgSilhouette(i, x_scaled)
}

ggplot(tibble(K = K, avgSil = avgSil), aes(x = K, y = avgSil)) +
  geom_point() +
  geom_line() +
  xlab("Number of clusters (k)") +
  ylab("Average silhouette width") +
  scale_x_continuous(breaks=c(seq(1,K[length(K)])))
```




```
kmedoid_coffee <- pam(x_scaled, 2)
kmedoid_coffee$silinfo$avg.width
```

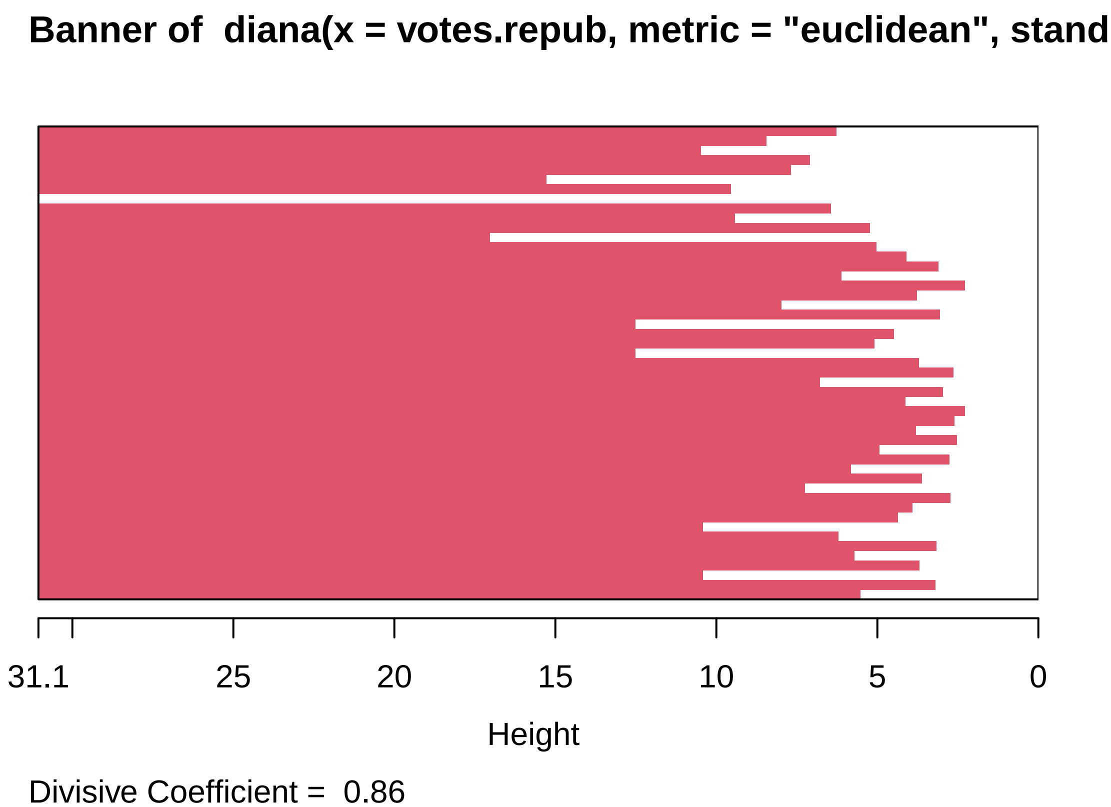
```
## [1] 0.4186062
```

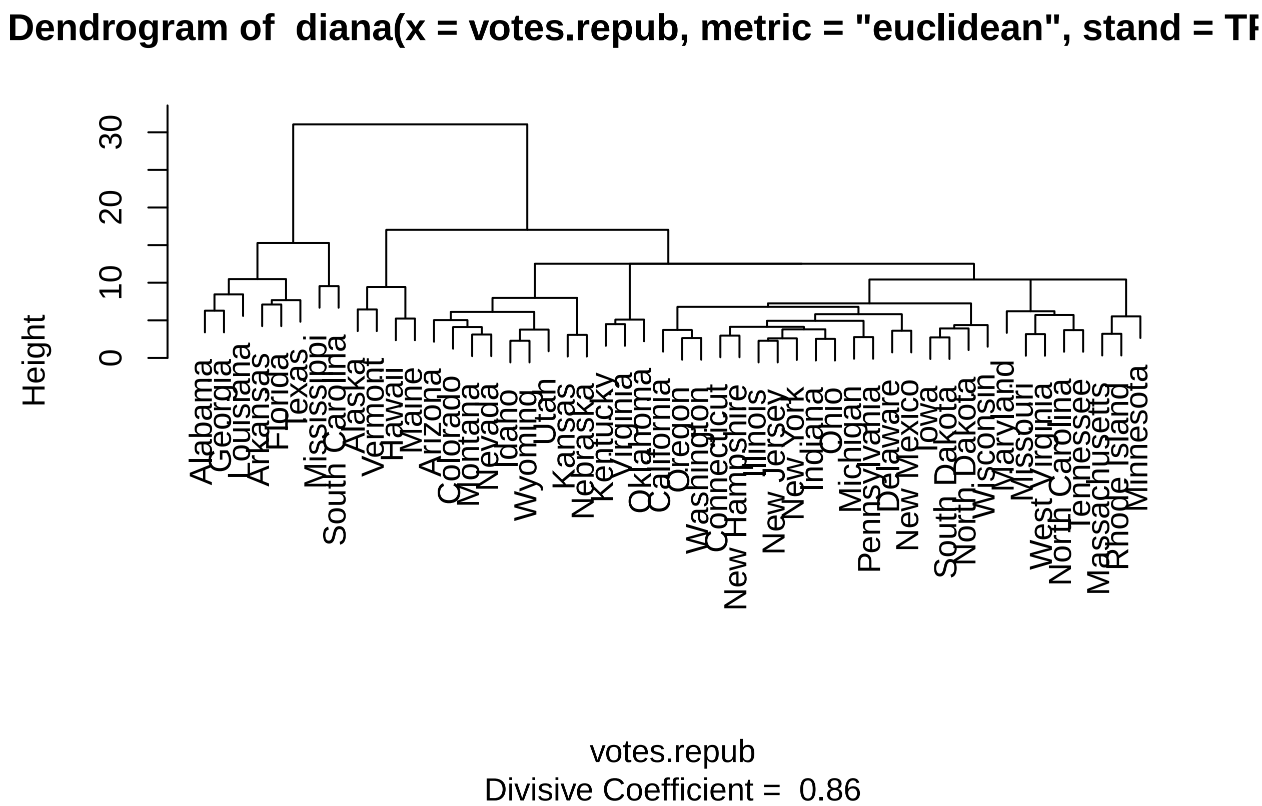
```
avgSil <- lapply(as.list(2:8), function(k){
  kmedoid_coffee <- pam(x_scaled, k)
kmedoid_coffee$silinfo$avg.width
}) %>% unlist()

ggplot(tibble(K = 2:8, avgSil = avgSil), aes(x = K, y = avgSil)) +
  geom_point() +
  geom_line() +
  xlab("Number of clusters (k)") +
  ylab("Average silhouette width for k-medoid") +
  scale_x_continuous(breaks=c(seq(1,K[length(K)])))
```



Average silhouette width for k−medoid
0.4
0.3
0.2
0.1
2
3
4
5
6
7
8
Number of clusters (k)

```
library(cluster)
library(factoextra)

data(votes.repub)
divisive_votes <- diana(
  votes.repub,
  metric = "euclidean",
  stand = TRUE
  )

plot(divisive_votes)
```





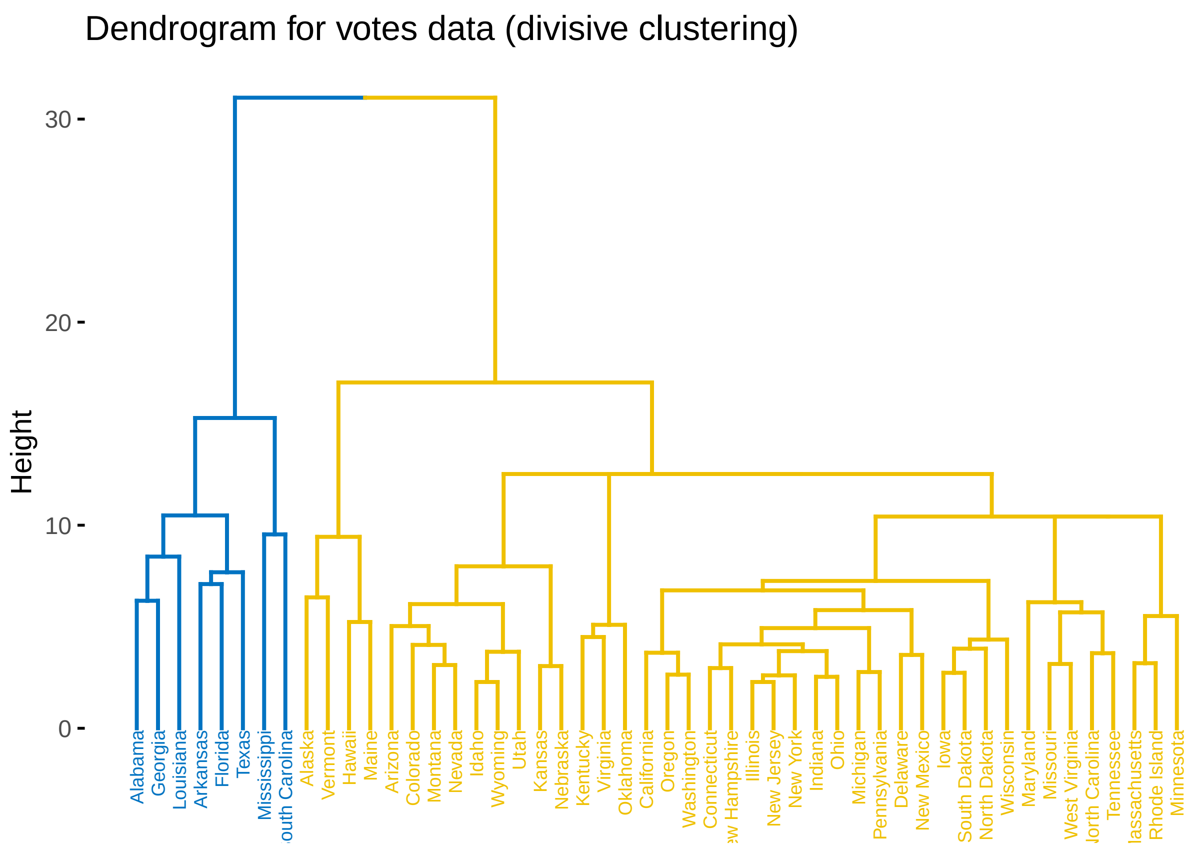

```r
cut_divisive_votes <- cutree(as.hclust(divisive_votes), k = 2)
table(cut_divisive_votes) # 8 and 42 group members
```

```
## cut_divisive_votes
##  1  2
##  8 42
```

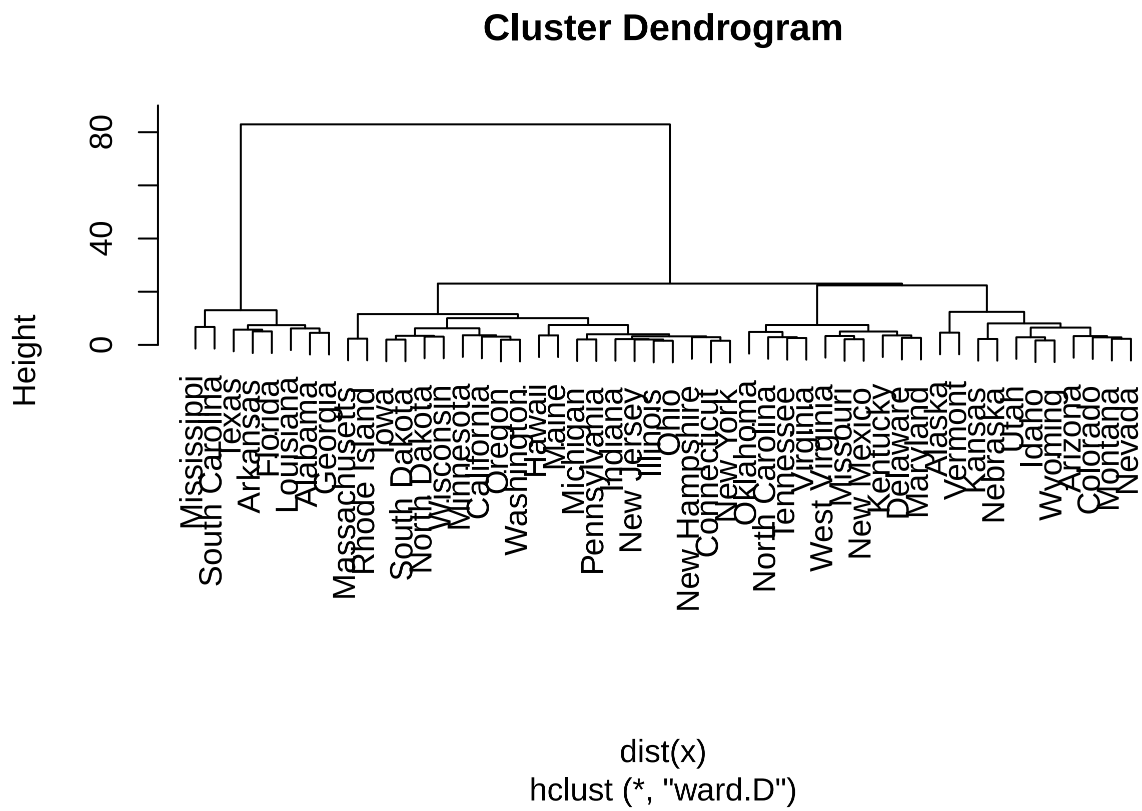
```r
rownames(votes.repub)[cut_divisive_votes == 1]
```

```
## [1] "Alabama"        "Arkansas"       "Florida"        "Georgia"
## [5] "Louisiana"      "Mississippi"    "South Carolina" "Texas"
```

```r
# rownames(votes.repub)[cut_divisive_votes == 2]

#make a nice dendrogram
fviz_dend(
  divisive_votes,
  cex = 0.5,
  k = 2, # Cut in 2 groups
  palette = "jco", # Color palette
  main = "Dendrogram for votes data (divisive clustering)")
```




```
x <- votes.repub %>%
  scale()
hc_vote <- hclust(dist(x), "ward.D")
plot(hc_vote)
```




```
#make a nice dendrogram
fviz_dend(
  hc_vote,
  k = 2, # Cut in 2 groups
  cex = 0.5,
  color_labels_by_k = TRUE,
  rect = TRUE,
  main = "Dendrogram for votes data (agglomerative clustering)"
  )
```

Dendrogram for votes data (agglomerative clustering)