MLStats: Unsupervised Learning

Darshan Patel

3/12/2019

In this assignment, mimic the lab exercises from ISLR Chapter 10: Unsupervised Learning.

Libraries

Load the following libraries.

```
rm(list = ls())
library(MASS)
library(ISLR)
library(tidyverse)
```

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

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

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

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

```
library(gridExtra)
library(ggdendro)
library(ggfortify)
```

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

```
library(iCluster)
```

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

Dataset

In this assignment, several datasets will be utilized. The dataset that will be used for the first part of this assignment is is `CC General.csv`. In this dataset, there is information about credit card holders and their buying behaviors. The goal of this is to make a marketing strategy for different sorts of credit card holders based on their attributes.

(Source: https://www.kaggle.com/arjunbhasin2013/ccdata)

```
df = read_delim("CC General.csv", delim = ',')
```

```
## Parsed with column specification:
## cols(
##   CUST_ID = col_character(),
##   BALANCE = col_double(),
##   BALANCE_FREQUENCY = col_double(),
##   PURCHASES = col_double(),
##   ONEOFF_PURCHASES = col_double(),
##   INSTALLMENTS_PURCHASES = col_double(),
##   CASH_ADVANCE = col_double(),
##   PURCHASES_FREQUENCY = col_double(),
```

```
##   ONEOFF_PURCHASES_FREQUENCY = col_double(),
##   PURCHASES_INSTALLMENTS_FREQUENCY = col_double(),
##   CASH_ADVANCE_FREQUENCY = col_double(),
##   CASH_ADVANCE_TRX = col_integer(),
##   PURCHASES_TRX = col_integer(),
##   CREDIT_LIMIT = col_integer(),
##   PAYMENTS = col_double(),
##   MINIMUM_PAYMENTS = col_double(),
##   PRC_FULL_PAYMENT = col_double(),
##   TENURE = col_integer()
## )
```

```
df = na.omit(df)
```

The number of observations in this dataset are

```
nrow(df)
```

```
## [1] 8627
```

The columns in this dataset are

```
colnames(df)
```

```
##  [1] "CUST_ID"                          "BALANCE"
##  [3] "BALANCE_FREQUENCY"                "PURCHASES"
##  [5] "ONEOFF_PURCHASES"                 "INSTALLMENTS_PURCHASES"
##  [7] "CASH_ADVANCE"                     "PURCHASES_FREQUENCY"
##  [9] "ONEOFF_PURCHASES_FREQUENCY"       "PURCHASES_INSTALLMENTS_FREQUENCY"
## [11] "CASH_ADVANCE_FREQUENCY"           "CASH_ADVANCE_TRX"
## [13] "PURCHASES_TRX"                    "CREDIT_LIMIT"
## [15] "PAYMENTS"                         "MINIMUM_PAYMENTS"
## [17] "PRC_FULL_PAYMENT"                 "TENURE"
```

Drop the `CUST_ID` column as it is only identification values.

```
df = df %>% subset(select = -CUST_ID)
```

To identify patterns in credit cardholders behaviors, perform different unsupervised learning methods.

Principal Components Analysis

Look at the means of the columns.

```
as.data.frame(apply(df, 2, mean))
```

```
##                                  apply(df, 2, mean)
## BALANCE                              1601.9254630
## BALANCE_FREQUENCY                       0.8953108
## PURCHASES                            1025.6986079
## ONEOFF_PURCHASES                      605.1071230
## INSTALLMENTS_PURCHASES                420.9029060
## CASH_ADVANCE                          994.6942648
## PURCHASES_FREQUENCY                     0.4960409
## ONEOFF_PURCHASES_FREQUENCY              0.2059465
## PURCHASES_INSTALLMENTS_FREQUENCY        0.3688670
## CASH_ADVANCE_FREQUENCY                  0.1376801
## CASH_ADVANCE_TRX                        3.3158688
```

```
## PURCHASES_TRX                          15.0403385
## CREDIT_LIMIT                         4521.8053785
## PAYMENTS                             1782.8592625
## MINIMUM_PAYMENTS                      864.8472800
## PRC_FULL_PAYMENT                        0.1591395
## TENURE                                 11.5342529
```

It is noticed here that different columns of data are on different scales. The `BALANCE` column runs in the thousands while many columns only have values less than one.

Look at the variances of the columns.

```
as.data.frame(apply(df, 2, var))
```

```
##                                  apply(df, 2, var)
## BALANCE                              4.393775e+06
## BALANCE_FREQUENCY                    4.305954e-02
## PURCHASES                            4.700684e+06
## ONEOFF_PURCHASES                     2.839436e+06
## INSTALLMENTS_PURCHASES               8.420431e+05
## CASH_ADVANCE                         4.504061e+06
## PURCHASES_FREQUENCY                  1.610900e-01
## ONEOFF_PURCHASES_FREQUENCY           9.006723e-02
## PURCHASES_INSTALLMENTS_FREQUENCY     1.585407e-01
## CASH_ADVANCE_FREQUENCY               4.074582e-02
## CASH_ADVANCE_TRX                     4.782266e+01
## PURCHASES_TRX                        6.346202e+02
## CREDIT_LIMIT                         1.339858e+07
## PAYMENTS                             8.464167e+06
## MINIMUM_PAYMENTS                     5.634577e+06
## PRC_FULL_PAYMENT                     8.769045e-02
## TENURE                               1.719526e+00
```

The variances also deviate a lot from column to column. The `BALANCE` column has a variance of 4392775 while the `BALANCE_FREQUENCY` column has a variance of `0.04305`. These differ by several powers of 10.

Due to the fact that there are huge differences in the mean and variance of cardholder informations, it will help to standardize the data when performing principal components analysis.

Perform PCA on the dataset and report the means and standard deviations used for PCA after scaling the dataset.

```
df_pca = prcomp(df, scale = TRUE)
as.data.frame(cbind("mean" = df_pca$center, "sd" = df_pca$scale))
```

```
##                                       mean           sd
## BALANCE                         1601.9254630 2096.1333839
## BALANCE_FREQUENCY                  0.8953108    0.2075079
## PURCHASES                       1025.6986079 2168.1061328
## ONEOFF_PURCHASES                 605.1071230 1685.0626466
## INSTALLMENTS_PURCHASES           420.9029060  917.6290915
## CASH_ADVANCE                     994.6942648 2122.2774064
## PURCHASES_FREQUENCY                0.4960409    0.4013602
## ONEOFF_PURCHASES_FREQUENCY         0.2059465    0.3001120
## PURCHASES_INSTALLMENTS_FREQUENCY   0.3688670    0.3981717
## CASH_ADVANCE_FREQUENCY             0.1376801    0.2018559
## CASH_ADVANCE_TRX                   3.3158688    6.9153929
## PURCHASES_TRX                     15.0403385   25.1916695
```

```
## CREDIT_LIMIT                     4521.8053785 3660.4076026
## PAYMENTS                         1782.8592625 2909.3241321
## MINIMUM_PAYMENTS                  864.8472800 2373.7263129
## PRC_FULL_PAYMENT                    0.1591395    0.2961257
## TENURE                             11.5342529    1.3113068
```

The values appear to look more manageable now that the values are scaled down.

Look at the principal component loading vector.

```
as.data.frame(round(t(df_pca$rotation), 4))
```

```
##       BALANCE BALANCE_FREQUENCY PURCHASES ONEOFF_PURCHASES
## PC1    0.0921            0.1098    0.4121           0.3468
## PC2    0.4059            0.1276    0.0494           0.0698
## PC3   -0.1744           -0.4587    0.2426           0.3687
## PC4    0.2595            0.1593    0.0640           0.1231
## PC5    0.0757           -0.4512   -0.0104          -0.1968
## PC6   -0.0363            0.0145   -0.1957          -0.1729
## PC7    0.2638           -0.0993   -0.2018          -0.1131
## PC8    0.1993           -0.1280    0.0057          -0.1231
## PC9   -0.0623           -0.6711   -0.1007          -0.0688
## PC10   0.0450           -0.0253    0.0592          -0.1651
## PC11   0.1506           -0.1389    0.1964           0.4465
## PC12  -0.4761            0.0673    0.0790          -0.0494
## PC13   0.5374           -0.1687    0.1089          -0.0106
## PC14  -0.1426            0.0240    0.2245           0.2230
## PC15  -0.2195           -0.0433    0.0636           0.0688
## PC16  -0.0057           -0.0093    0.0015          -0.0052
## PC17   0.0000            0.0000    0.7489          -0.5821
##       INSTALLMENTS_PURCHASES CASH_ADVANCE PURCHASES_FREQUENCY
## PC1                   0.3370      -0.0304              0.3236
## PC2                  -0.0115       0.4373             -0.1867
## PC3                  -0.1039      -0.0017             -0.3559
## PC4                  -0.0748      -0.2655             -0.2216
## PC5                   0.3373       0.0995             -0.0887
## PC6                  -0.1448       0.1326              0.0856
## PC7                  -0.2693       0.0388              0.1582
## PC8                   0.2398       0.0043             -0.0270
## PC9                  -0.1118       0.0187              0.1912
## PC10                  0.4433      -0.3743             -0.2578
## PC11                 -0.3560      -0.3531              0.1259
## PC12                  0.2774      -0.1739              0.1615
## PC13                  0.2768       0.0112              0.1935
## PC14                  0.1210       0.5962             -0.0105
## PC15                  0.0238       0.2393              0.0238
## PC16                  0.0140      -0.0083              0.6788
## PC17                 -0.3168       0.0000              0.0002
##       ONEOFF_PURCHASES_FREQUENCY PURCHASES_INSTALLMENTS_FREQUENCY
## PC1                       0.2947                           0.2772
## PC2                      -0.0148                          -0.1737
## PC3                       0.1050                          -0.4502
## PC4                       0.0555                          -0.2651
## PC5                      -0.5213                           0.1750
## PC6                       0.0959                           0.0481
## PC7                       0.3061                          -0.0428
```

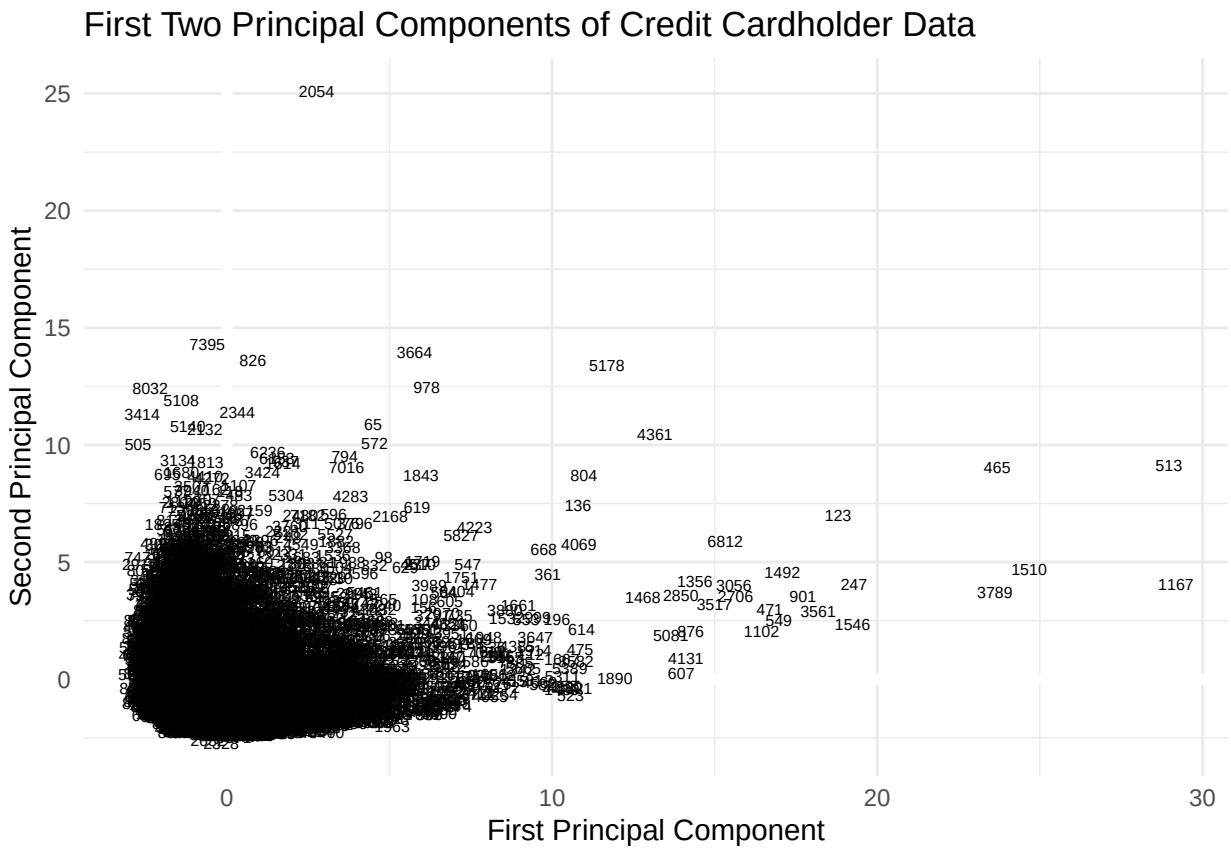
```
## PC8                    -0.2004                          0.1291
## PC9                     0.3621                          0.0820
## PC10                    0.0892                         -0.2555
## PC11                   -0.3705                          0.2962
## PC12                    0.1655                         -0.0174
## PC13                    0.2478                         -0.0410
## PC14                   -0.0440                         -0.0436
## PC15                    0.0153                          0.0672
## PC16                   -0.3416                         -0.6329
## PC17                   -0.0001                         -0.0002
##      CASH_ADVANCE_FREQUENCY CASH_ADVANCE_TRX PURCHASES_TRX CREDIT_LIMIT
## PC1                 -0.0990          -0.0568        0.3910       0.2101
## PC2                  0.4301           0.4165       -0.0121       0.2438
## PC3                 -0.0875          -0.0870       -0.0797       0.0952
## PC4                 -0.2665          -0.3325       -0.0240       0.1228
## PC5                 -0.1600          -0.0898       -0.0526       0.1324
## PC6                 -0.0313           0.0903       -0.0778       0.3116
## PC7                 -0.1374          -0.1968       -0.1041       0.5454
## PC8                 -0.0765          -0.1789        0.0463       0.3659
## PC9                  0.0875           0.2154        0.2555      -0.0949
## PC10                 0.2903           0.2079        0.2294       0.1604
## PC11                 0.2131           0.2085       -0.2030       0.1516
## PC12                 0.0375           0.2038       -0.5935       0.3208
## PC13                 0.0424          -0.0940       -0.5299      -0.4019
## PC14                -0.3393           0.1207       -0.0797       0.0305
## PC15                 0.6464          -0.6482       -0.0382       0.0541
## PC16                 0.0417          -0.0144        0.1349       0.0187
## PC17                 0.0000           0.0000        0.0001       0.0000
##      PAYMENTS MINIMUM_PAYMENTS PRC_FULL_PAYMENT  TENURE
## PC1    0.2641           0.0593           0.1307  0.0779
## PC2    0.2641           0.1704          -0.1957 -0.0046
## PC3    0.2874          -0.2490           0.1841 -0.0653
## PC4   -0.0976           0.3521          -0.4184  0.4286
## PC5    0.1888           0.4171           0.2017  0.1174
## PC6    0.0660          -0.3410           0.2873  0.7466
## PC7   -0.1691           0.2016           0.2793 -0.3991
## PC8   -0.0485          -0.6136          -0.4822 -0.1685
## PC9   -0.1364           0.1484          -0.3924  0.1437
## PC10  -0.4594          -0.0162           0.2672  0.0414
## PC11  -0.2597          -0.0224           0.0497  0.0655
## PC12   0.1179           0.1595          -0.2465 -0.0307
## PC13  -0.0429          -0.1410           0.1130  0.0774
## PC14  -0.6041           0.0236          -0.0083  0.0479
## PC15  -0.1398           0.0721          -0.0106  0.1042
## PC16   0.0111          -0.0146          -0.0209  0.0203
## PC17   0.0000           0.0000           0.0000  0.0000
```

A total of 17 principal components are found. This is accurate because generally, there are $\min(n-1, p)$ informative principal components in a dataset with n observations and p variables. Here n is very large and so there are p principal components by default. (Remember that the `CUST_ID` column was dropped!)

The first two principal components can be plotted as below.

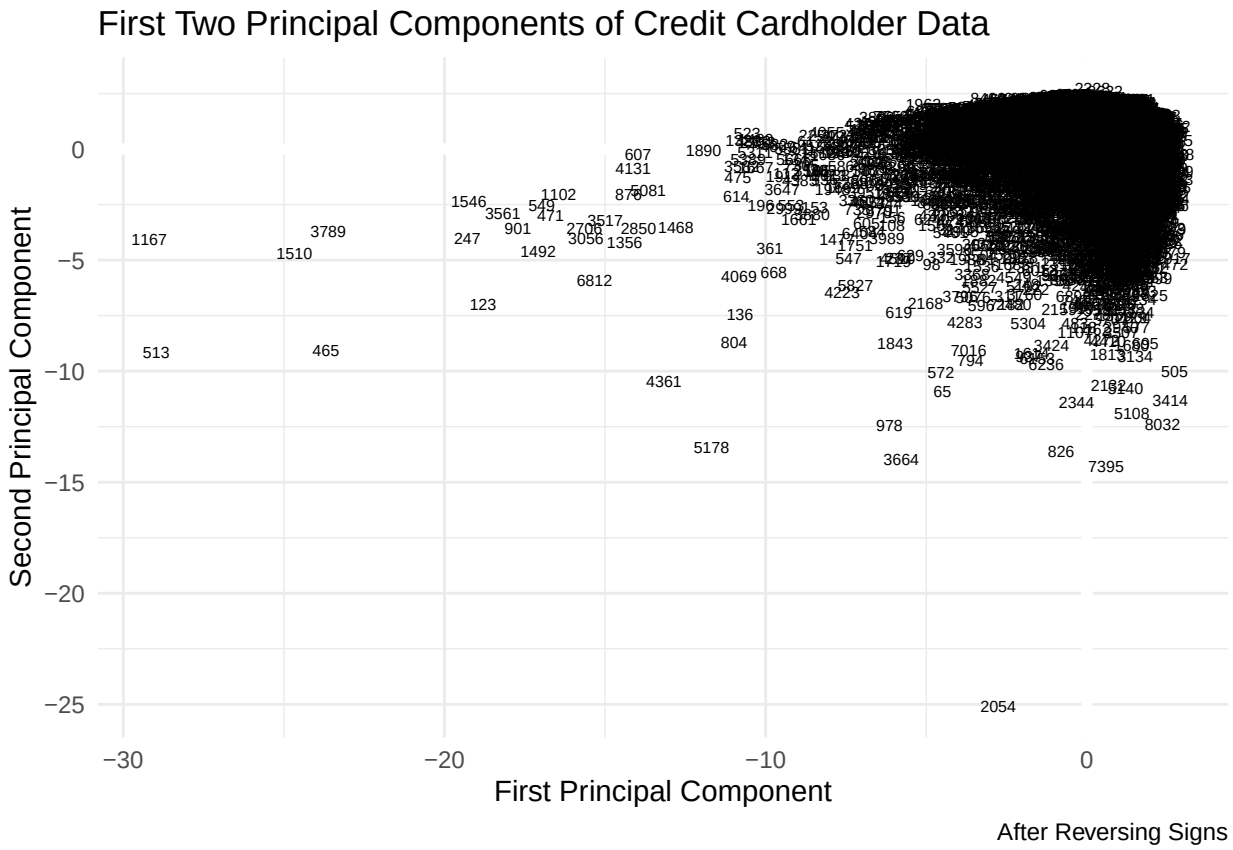
```
ggplot(df_pca$x[,1:2], aes(x = PC1, y = PC2)) +
  modelr::geom_ref_line(h = 0) +
```

```
  modelr::geom_ref_line(v = 0) +
  geom_text(aes(label = seq(1,nrow(df),1)), size = 2) +
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  ggtitle("First Two Principal Components of Credit Cardholder Data") +
  theme_minimal()
```



Principal components are unique up to a sign change. Reproduce the mirror image of the plot above. Do this by reversing the signs of the principal components vectors and loadings.

```
df_pca$rotation = -df_pca$rotation
df_pca$x = -df_pca$x

ggplot(df_pca$x[,1:2], aes(x = PC1, y = PC2)) +
  modelr::geom_ref_line(h = 0) +
  modelr::geom_ref_line(v = 0) +
  geom_text(aes(label = seq(1,nrow(df),1)), size = 2) +
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  ggtitle("First Two Principal Components of Credit Cardholder Data") +
  labs(caption = "After Reversing Signs") +
  theme_minimal()
```



The standard deviations of the principal components are as follows.

```
df_pca$sdev
```

```
##  [1] 2.151907405 1.860898889 1.231073494 1.134943816 1.032850443
##  [6] 0.985212363 0.914137838 0.846015410 0.791463438 0.723744992
## [11] 0.633946213 0.549242734 0.492499984 0.447269533 0.414001647
## [16] 0.214720738 0.003425255
```

There does not appear to be a lot of variation in the componenets calculated.

Now, to understand how much the principal components explain the variance, compute the proportion of variance explained by each principal component.

```
pca_var = df_pca$sdev^2
pve = pca_var / sum(pca_var)
pve
```

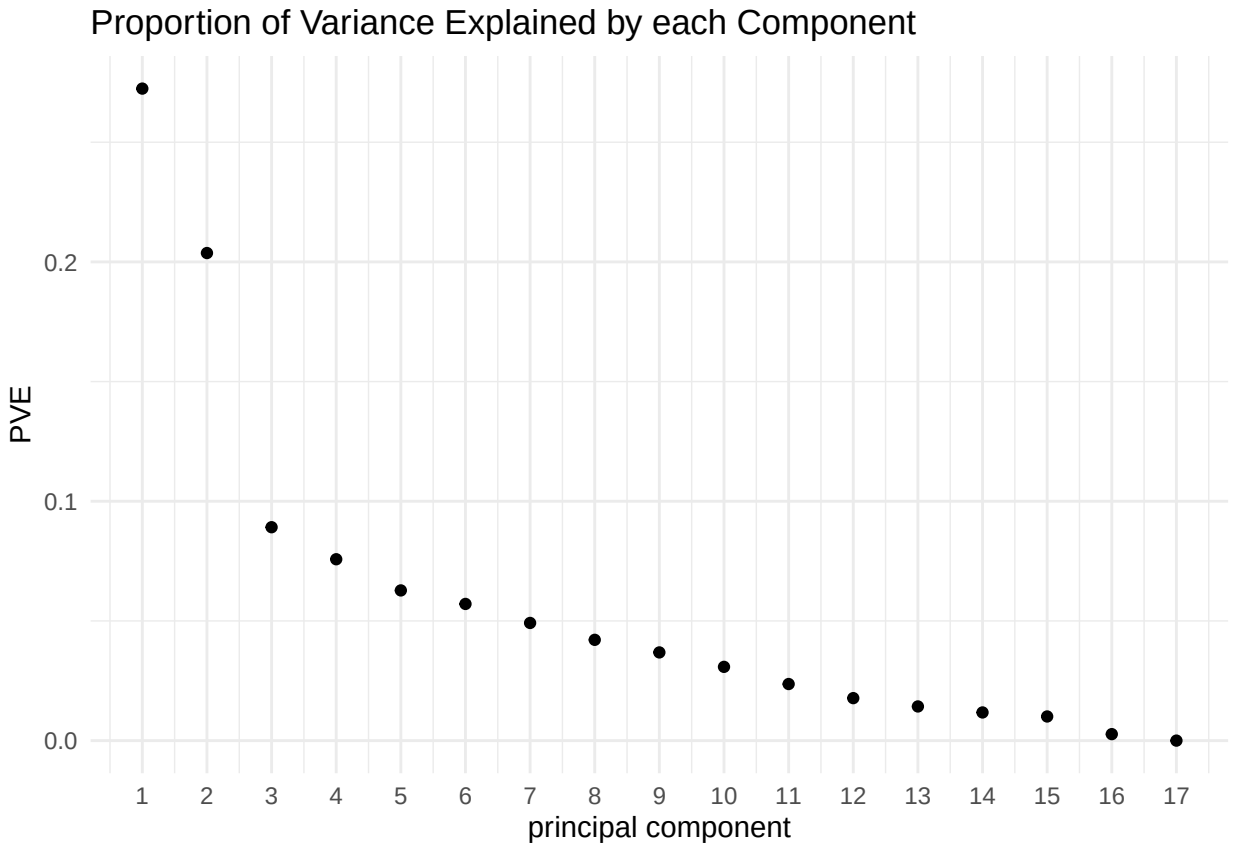
```
##  [1] 2.723944e-01 2.037026e-01 8.914953e-02 7.577044e-02 6.275177e-02
##  [6] 5.709667e-02 4.915576e-02 4.210247e-02 3.684790e-02 3.081217e-02
## [11] 2.364046e-02 1.774515e-02 1.426801e-02 1.176765e-02 1.008220e-02
## [16] 2.712059e-03 6.901394e-07
```

According to this, the first principal component explains 27.23% of the variance in the data; the next principal component explains 20.37% of the variance in the data.

Plot the proportion of variance explained, PVE, by each component.

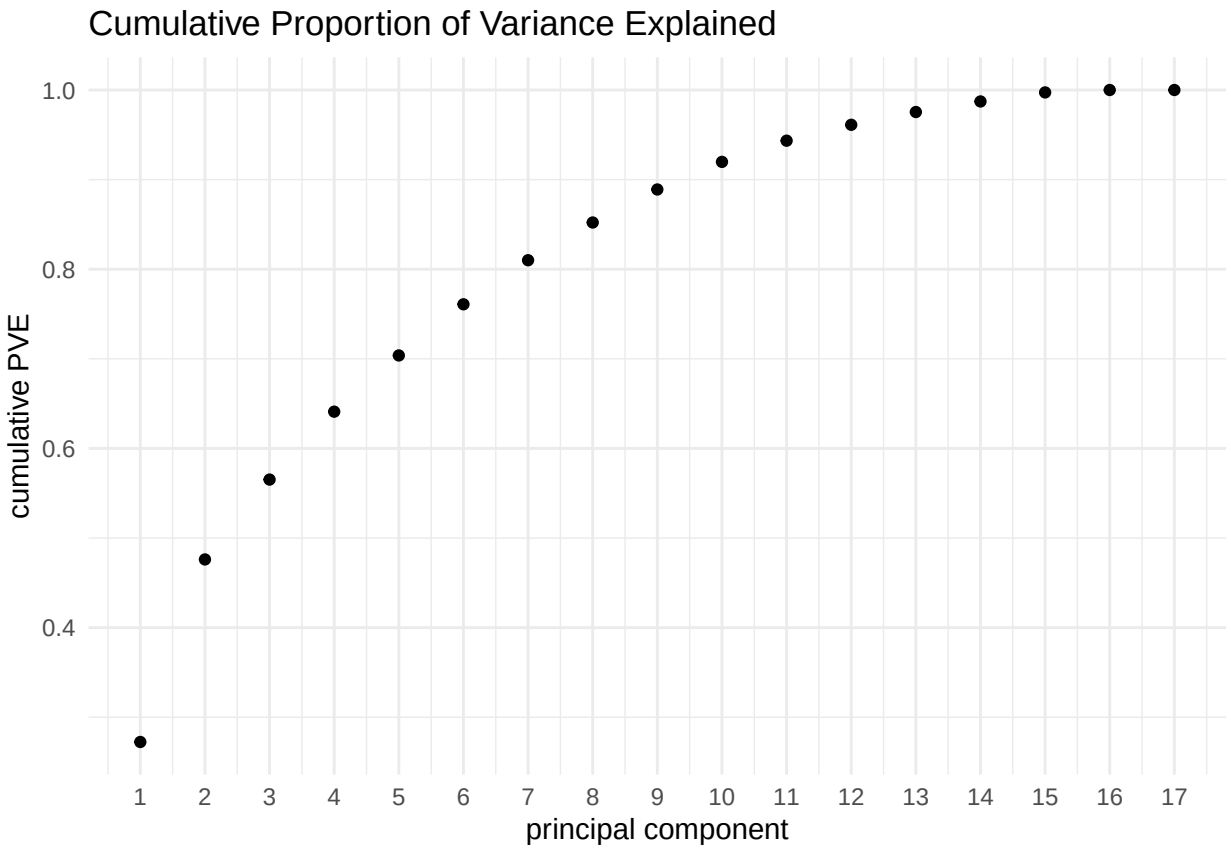
```
pve_df = as.data.frame(pve) %>% mutate("cs_pve" = cumsum(pve))
ggplot(pve_df, aes(x = 1:17, y = pve)) + geom_point() +
  scale_x_continuous(breaks = 1:17) +
```

```
  labs(x = "principal component", y = "PVE") +
  ggtitle("Proportion of Variance Explained by each Component") +
  theme_minimal()
```



Plot the cumulative proportion of variance explained as additional components are added.

```
ggplot(pve_df, aes(x = 1:17, y = cs_pve)) + geom_point() +
  scale_x_continuous(breaks = 1:17) +
  labs(x = "principal component", y = "cumulative PVE") +
  ggtitle("Cumulative Proportion of Variance Explained") +
  theme_minimal()
```



After the 7 principal component is made, 80% of the variance in the data is explained. By the 10th principal componenet, 90% of the variance in the data is explained.

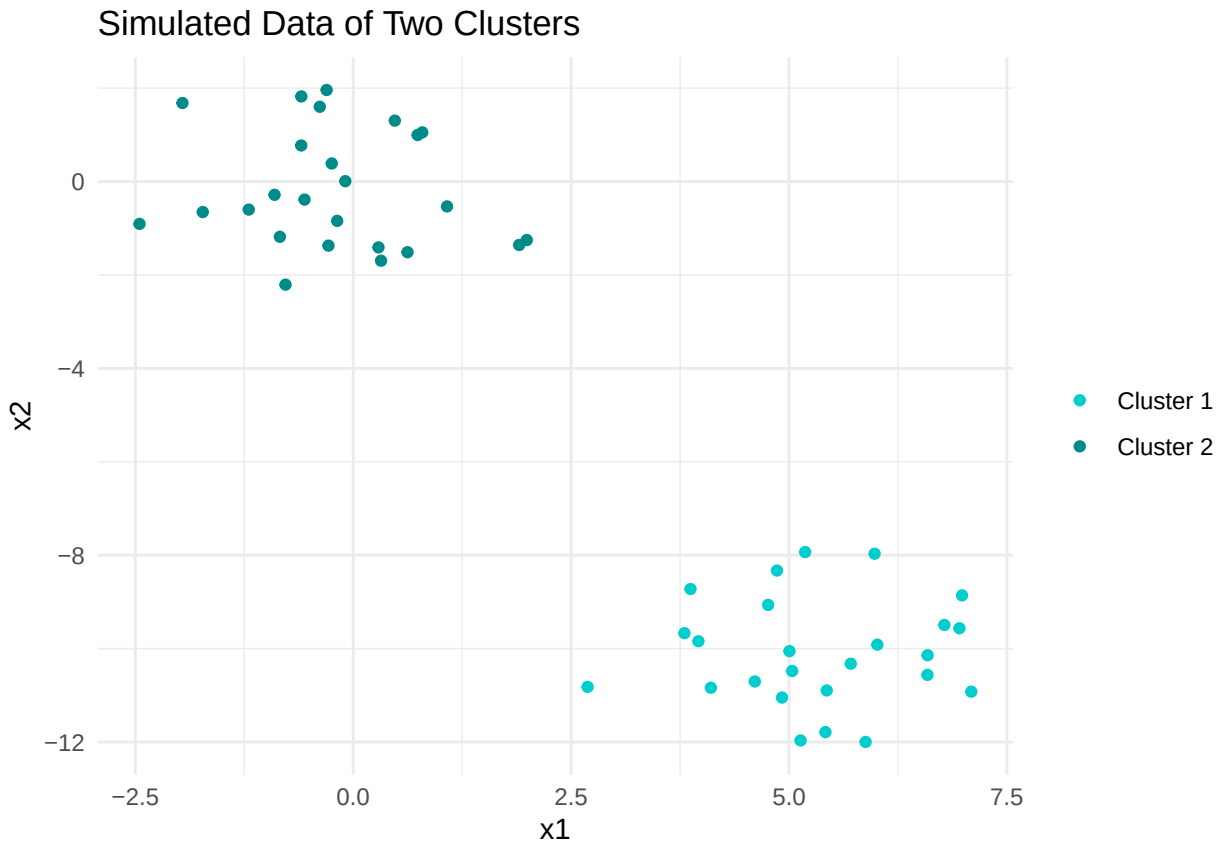
Another unsupervised learning technique that can be used is clustering. There are many different types of clustering algorithms, two of which are k-means clustering and hierarchical clustering. Let's see those in action on the credit cardholders dataset.

K-Means Clustering

To best understand the functionality of k-means clustering, it helps to use data that can be separated into classes.

First simulate a dataset where two distinct clusters can be seen.

```
set.seed(2)
X = matrix(rnorm(100), ncol = 2)
X[1:25, 1] = X[1:25, 1] + 5
X[1:25, 2] = X[1:25, 2] - 10

df = as.data.frame(X)
ggplot(df, aes(x = V1, y = V2, color = V1 < 2.5)) + geom_point() +
  scale_colour_manual(labels = c("Cluster 1", "Cluster 2"),
                      values = c("cyan3", "darkcyan"),
                      name = NULL) +
  labs(x = "x1", y = "x2") +
  ggtitle("Simulated Data of Two Clusters") +
  theme_minimal()
```



The two clusters can be visually seen.

Now perform K-means clustering with $K = 2$.

```
km_cluster_2 = kmeans(df, centers = 2)
```

The cluster assignments of the 100 observations can be seen.

```
km_cluster_2$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 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

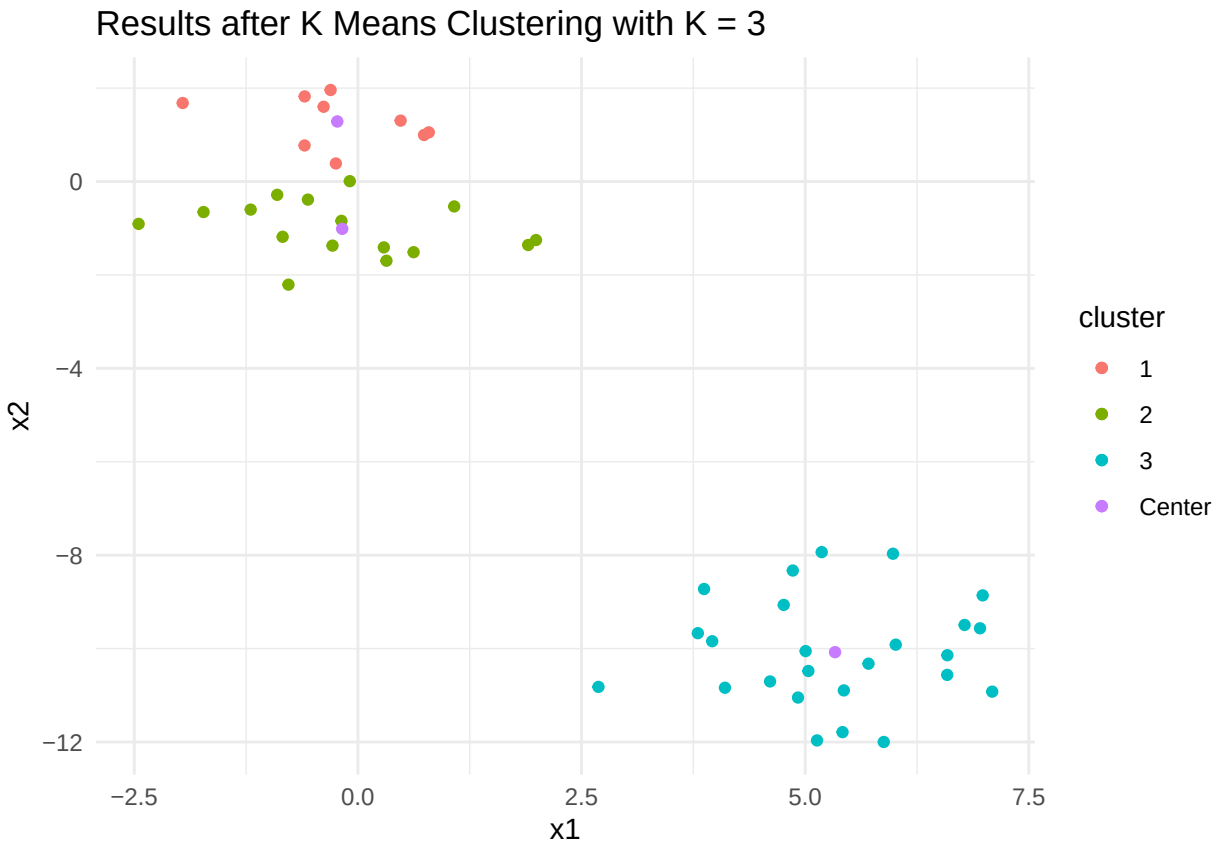
It does appear that two clusters were made of equal size, as defined. Seeing this visually can only help to determine if the observations got properly clustered.

```
df$cluster = factor(km_cluster_2$cluster)
centers = as.data.frame(km_cluster_2$centers)

ggplot(df, aes(x = V1, y = V2, color = cluster)) +
  geom_point() +
  geom_point(data = centers, aes(x = V1, y = V2, color = 'Center')) +
  geom_point(data = centers, aes(x = V1, y = V2, color = 'Center'),
             alpha = 0.3) +
  labs(x = "x1", y = "x2") +
  ggtitle("Results after K Means Clustering with K = 2") +
  theme_minimal()
```

Two clusters are properly made.

Now try with $K = 3$ and plot the clusters.

```
km_cluster_3 = kmeans(df, centers = 3)
df$cluster = factor(km_cluster_3$cluster)
centers = as.data.frame(km_cluster_3$centers)

ggplot(df, aes(x = V1, y = V2, color = cluster)) +
  geom_point() +
  geom_point(data = centers, aes(x = V1, y = V2, color = 'Center')) +
  geom_point(data = centers, aes(x = V1, y = V2, color = 'Center'),
             alpha = 0.3) +
  labs(x = "x1", y = "x2") +
  ggtitle("Results after K Means Clustering with K = 3") +
  theme_minimal()
```



With the inclusion of the third cluster, one of the clusters that was previously properly in 1 cluster is now broken into two smaller clusters.

Look at the details of the clusters.

```
km_cluster_3
```

```
## K-means clustering with 3 clusters of sizes 9, 16, 25
##
## Cluster means:
##           V1         V2 cluster
## 1 -0.2309472   1.285948       1
## 2 -0.1758700  -1.012217       1
## 3  5.3339737 -10.076191       2
##
## Clustering vector:
##  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 1 1 1 2 1 2 2 2 2
## [36] 1 2 2 2 1 1 1 2 2 2 2 1 2 2 2
##
## Within cluster sum of squares by cluster:
## [1]  7.979261 26.982150 63.205951
##  (between_SS / total_SS =  94.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

The cluster sizes are 16, 9 and 25.

Now run K means clustering with multiple initial cluster assignments to find the best results.

```
set.seed(20)
kmeans(df, centers = 3, nstart = 1)$tot.withins
```

```
## [1] 103.9393
```

```
kmeans(df, centers = 3, nstart = 20)$tot.withins
```

```
## [1] 98.16736
```

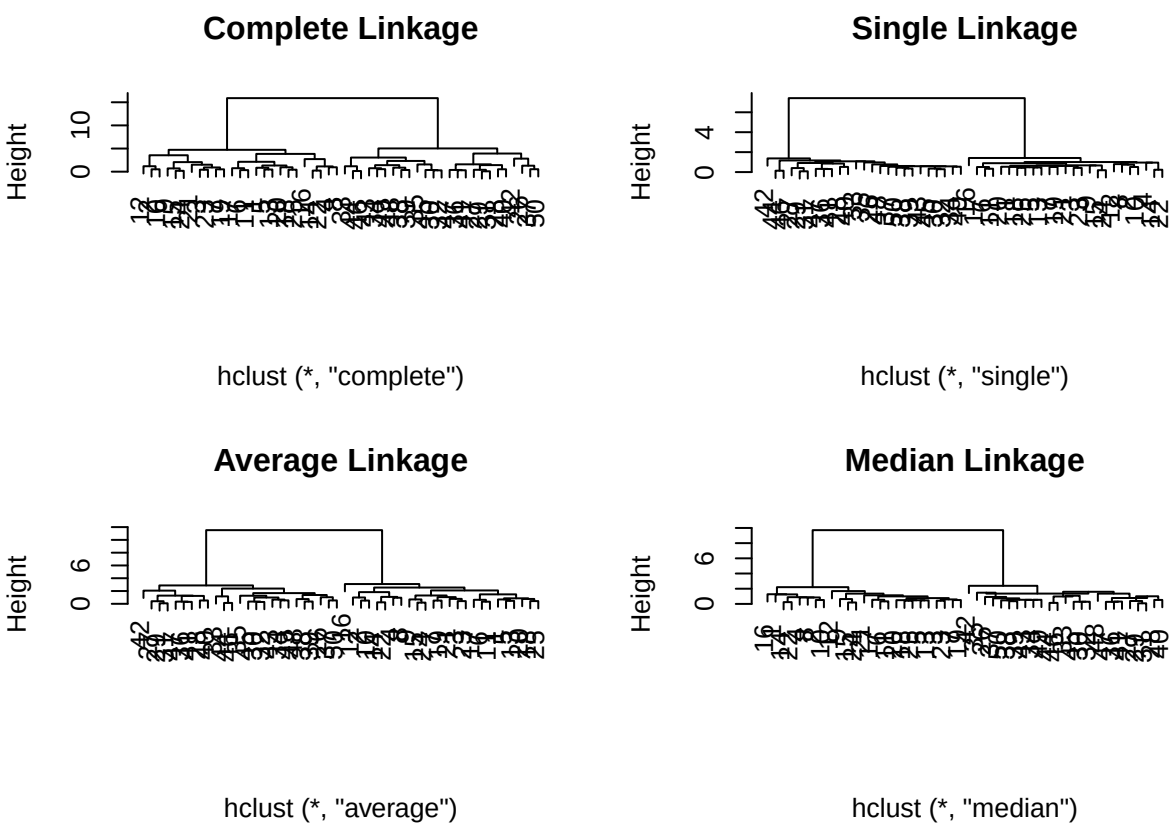
When utilizing more assignments, a more optimal result can be found here. Above the individual within-cluster sum of squares is shown. When more more assignments were used, a better clustering was made that lessened the inter-observation distances in the clusters.

Now let's use hierarchical clustering on this simulated dataset.

Hierarchical Clustering

Perform hierarchical clustering with the following linkages: complete, single, average, median and plots its respective dendrogram.

```
hc_complete = hclust(dist(df), method = "complete")
hc_single = hclust(dist(df), method = "single")
hc_average = hclust(dist(df), method = "average")
hc_median = hclust(dist(df), method = "median")

par(mfrow = c(2,2))
plot(hc_complete, main = "Complete Linkage", xlab = "")
plot(hc_single, main = "Single Linkage", xlab = "")
plot(hc_average, main = "Average Linkage", xlab = "")
plot(hc_median, main = "Median Linkage", xlab = "")
```



It is viually apparent that two clusters were made using each linkage type. Here the different types of linkages had no effect on the clustering.

The cluster labels of each observation associated with the given cut of 2 of the dendrogram can also be looked at.

```
cutree(hc_complete, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
cutree(hc_single, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
cutree(hc_average, 2)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
cutree(hc_median, 2)
```

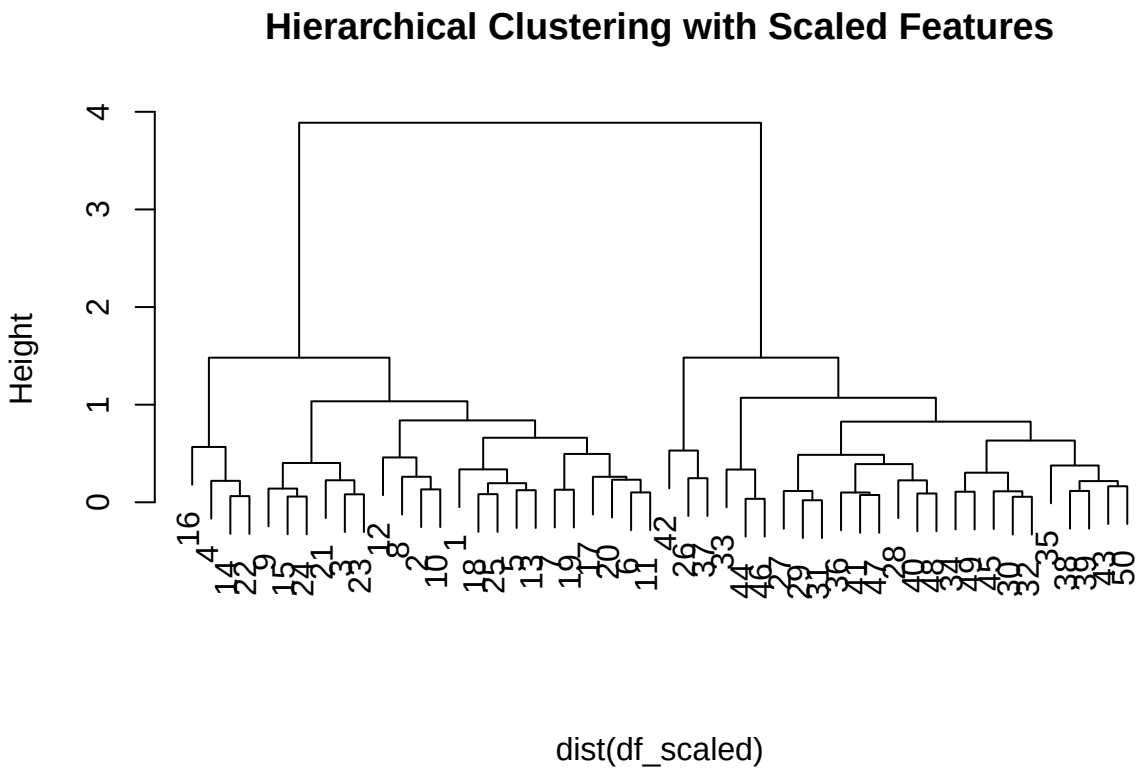
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Despite the different linkages used, the same clusters are made with the same observations.

Try scaling the variables to see what changes. Use complete linkage and plot the dendrogram.

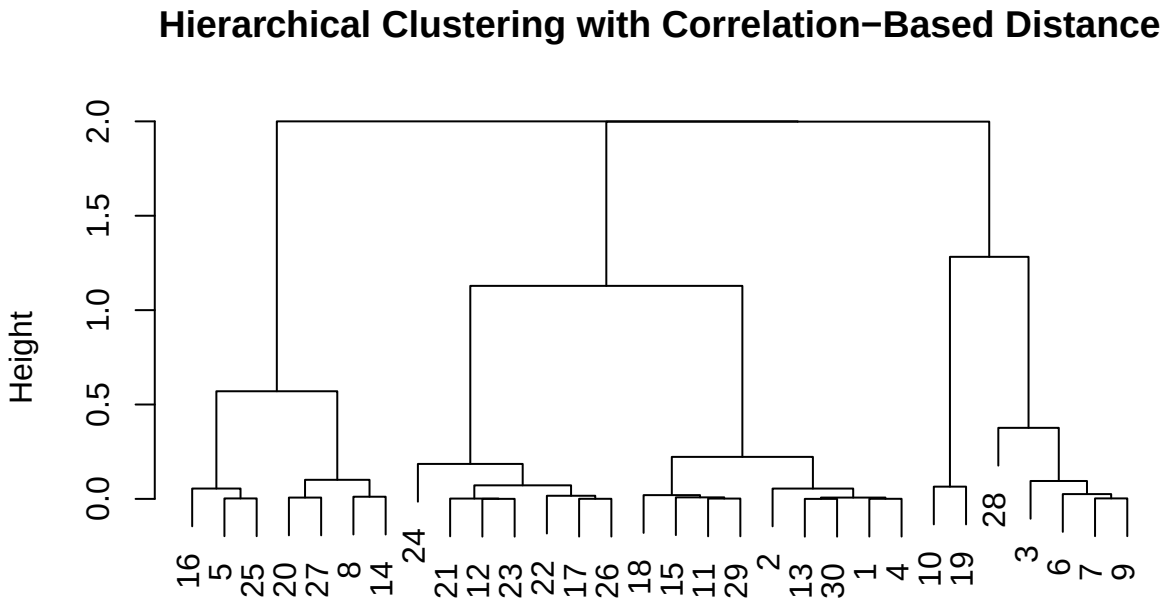
```
df = df %>% select(V1, V2)
df_scaled = scale(df)
```

```r
hc_complete_scaled = hclust(dist(df_scaled), method = "complete")
plot(hc_complete_scaled, main = "Hierarchical Clustering with Scaled Features", sub = "")
```



Try hierarchical clustering for a simulated dataset with 3 features.

```r
set.seed(3)
X = matrix(rnorm(30 * 3), ncol = 3)
X_dist = as.dist(1 - cor(t(X)))
plot(hclust(X_dist, method = "complete"), main = "Hierarchical Clustering with Correlation-Based Distan
```



It is visually apparent that three distinct clusters are made directly from the entire dataset.

Unsupervised clustering is often used in the analysis of genomic data. In particular, PCA and hierarchical clustering are the most popular tools for this. Try doing this on a genomic dataset from the internet.

Genomic Data

The dataset that will be used here is `tissuesGeneExpression` which is from the `genomicsclass` package. Code is presented below to obtain the dataset. (When asked whether or not to update packages, say `n`.)

First create the distance matrix and obtain the labels of the tissues.

```
df = t(e)
group = as.fumeric(tab$Tissue)
```

The different types of tissues are

```
table(tab$Tissue)
```

```
##
##  cerebellum       colon endometrium hippocampus      kidney       liver
##          38          34          15          31          39          26
##    placenta
##           6
```

The dimensions of the dataset is

```
dim(df)
```

```
## [1]   189 22215
```

That's a staggering 22215 different gene expression measurements, on only 189 cancer cells.

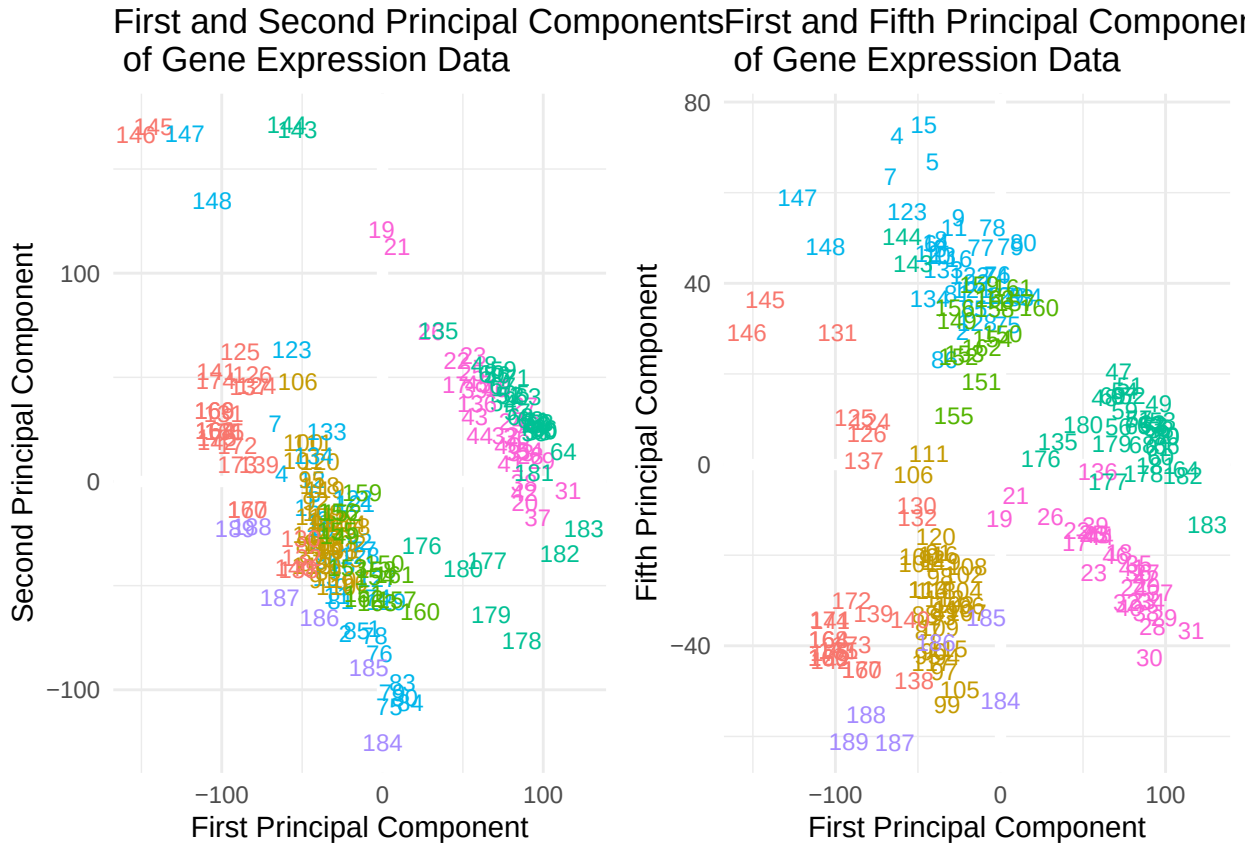
For the first part of this investigation, perform PCA on the scaled dataset.

```
gene_pca = prcomp(df, scale = TRUE)
```

The first few principal component score vectors are plotted.

```
Cols = function(vec){
  cols = rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))])
}


g1 = ggplot(gene_pca$x[,1:2], aes(x = PC1, y = PC2, color = Cols(group))) +
  modelr::geom_ref_line(h = 0) +
  modelr::geom_ref_line(v = 0) +
  geom_text(aes(label = seq(1,nrow(df),1)), size = 3) +
  xlab("First Principal Component") +
  ylab("Second Principal Component") +
  scale_color_discrete(guide = FALSE) +
  ggtitle("First and Second Principal Components \n of Gene Expression Data") +
  theme_minimal()
g2 = ggplot(gene_pca$x[,c(1,5)], aes(x = PC1, y = PC5, color = Cols(group))) +
  modelr::geom_ref_line(h = 0) +
  modelr::geom_ref_line(v = 0) +
  geom_text(aes(label = seq(1,nrow(df),1)), size = 3) +
  xlab("First Principal Component") +
  ylab("Fifth Principal Component") +
  scale_color_discrete(guide = FALSE) +
```

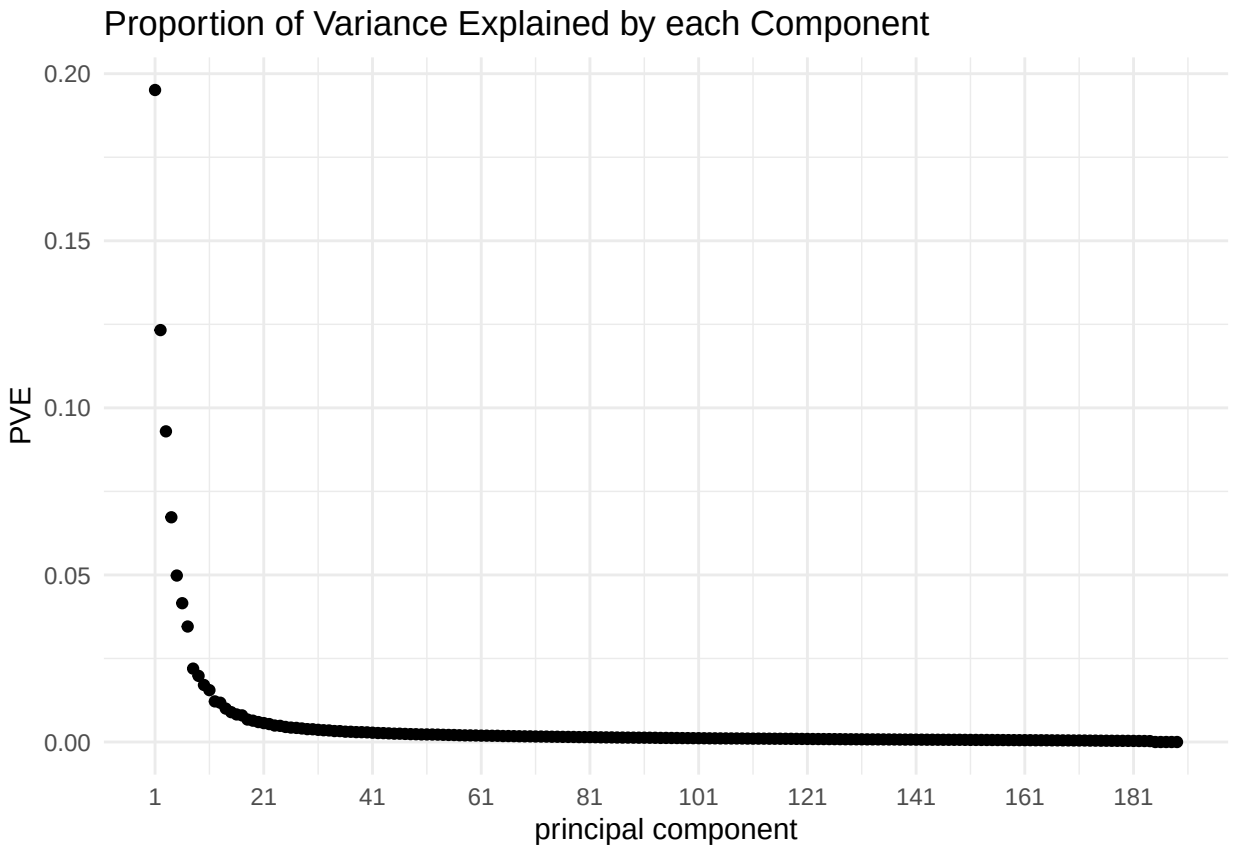
```
  ggtitle("First and Fifth Principal Components \n of Gene Expression Data") +
  theme_minimal()
grid.arrange(g1, g2, ncol = 2)
```



Values that have the same color indicate the same cancer types. It is clear from the plot that genes that exhibit the same cancer type tend to have similar gene expression levels.

A summary of the proportion of variance of the first few principal components can be displayed in a tabular form but since there are more than a hundred principal components, it might be helpful to just visualize them.

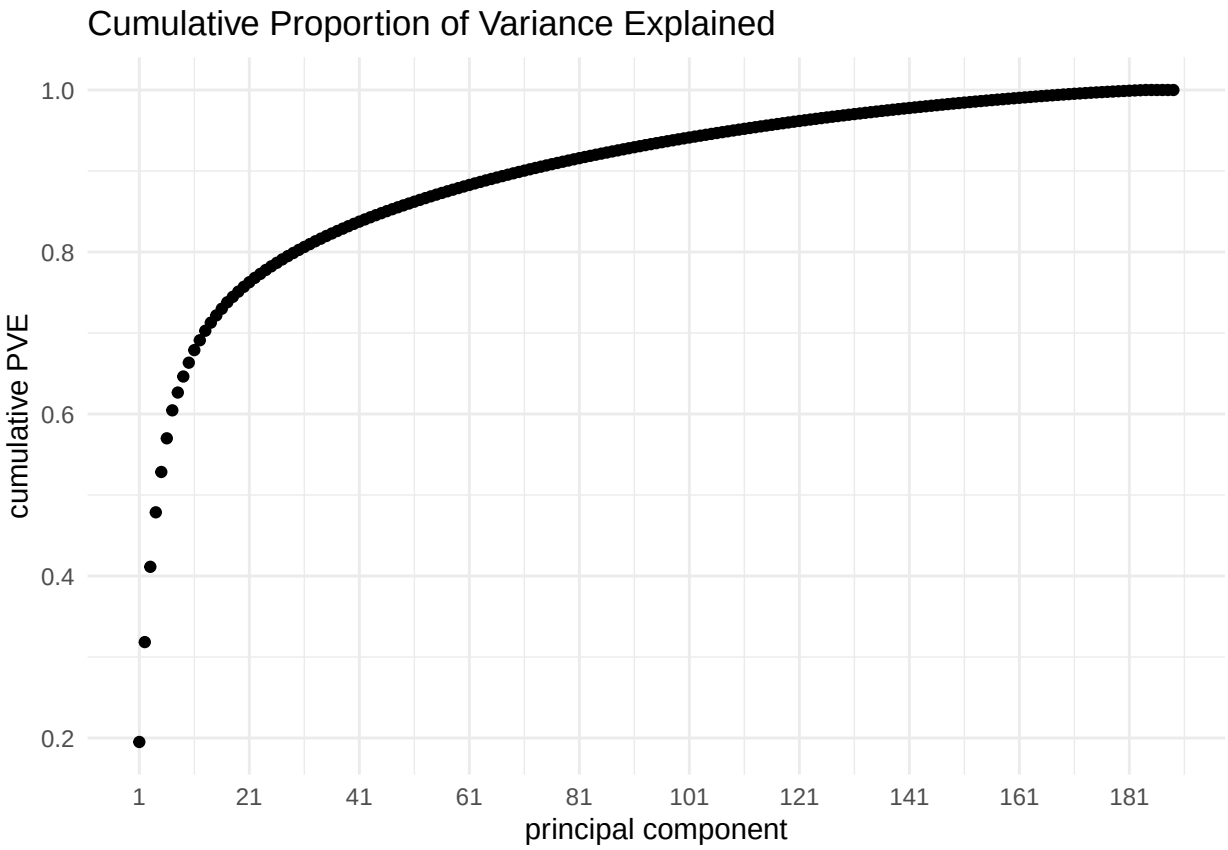
```
pca_var = gene_pca$sdev^2
pve = pca_var / sum(pca_var)
pve_df = as.data.frame(pve) %>% mutate("cs_pve" = cumsum(pve))
ggplot(pve_df, aes(x = 1:189, y = pve)) + geom_point() +
  scale_x_continuous(breaks = seq(1,189, by = 20),
                     labels = seq(1,189, by = 20)) +
  labs(x = "principal component", y = "PVE") +
  ggtitle("Proportion of Variance Explained by each Component") +
  theme_minimal()
```



After the 31st principal component, little to none of the variance is explained by the principal components created.

The cumulative principal of variance explained is plotted below.

```
ggplot(pve_df, aes(x = 1:189, y = cs_pve)) + geom_point() +
  scale_x_continuous(breaks = seq(1,189, by = 20),
                     labels = seq(1,189, by = 20)) +
  labs(x = "principal component", y = "cumulative PVE") +
  ggtitle("Cumulative Proportion of Variance Explained") +
  theme_minimal()
```



After the 31st principal component, 80% of the variance in the data is explained. After the 71st principal component, 90% of the variance in the data is explained.

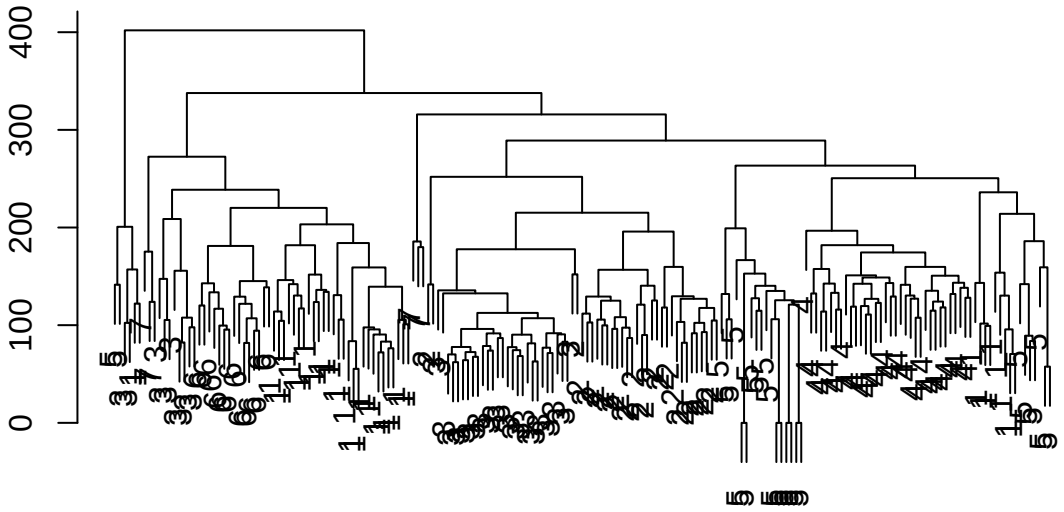
Now let's try hierarchically custering the cell lines. First scale the data.

```
scaled_df = scale(df)
```

Perform hierarchical clustering on the data using complete, single, average and median linkage and plot the dendrograms.

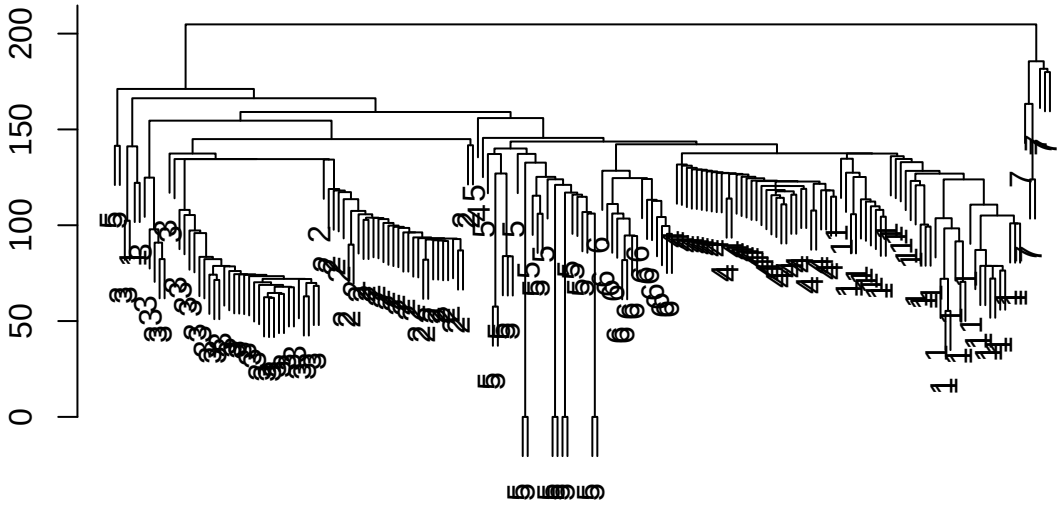
```
dist_df = dist(scaled_df)
plot(hclust(dist_df, method = "complete"), labels = group,
     main = "Complete Linkage", xlab = "", sub = "", ylab = "")
```

Complete Linkage

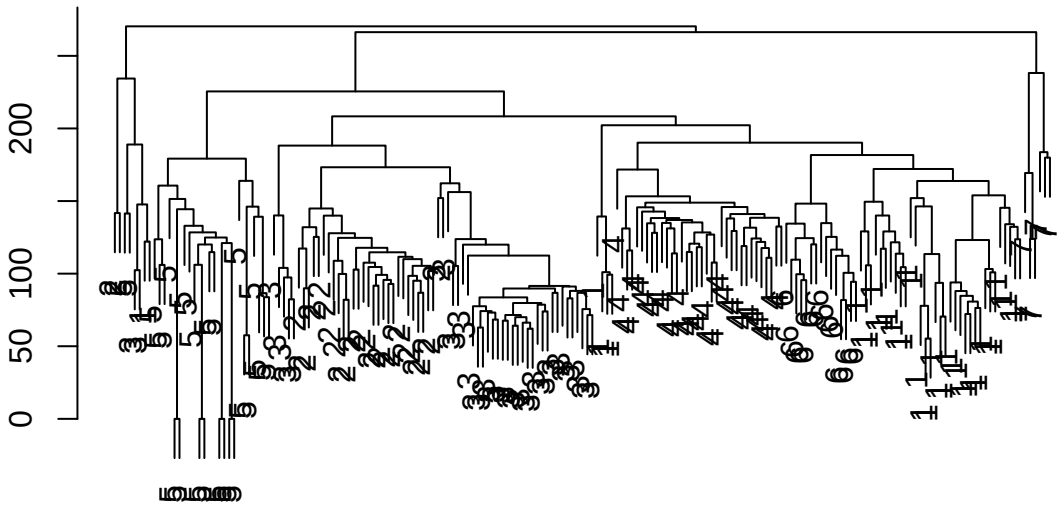



```
plot(hclust(dist_df, method = "single"), labels = group,
     main = "Single Linkage", xlab = "", sub = "", ylab = "")
```

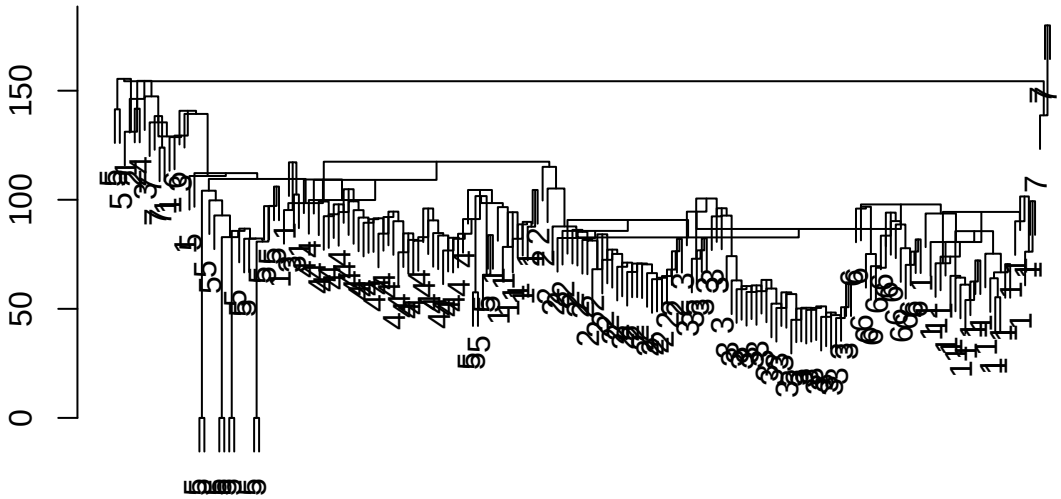
Single Linkage




```
plot(hclust(dist_df, method = "average"), labels = group,
     main = "Average Linkage", xlab = "", sub = "", ylab = "")
```




```
plot(hclust(dist_df, method = "median"), labels = group,
     main = "Median Linkage", xlab = "", sub = "", ylab = "")
```



Depending on the type of linkages used, clusters were made differently. In the clustering done by complete linkage, the clusters appear to look balanced. In the clustering done by single linkage, there are many trailing clusters of size one. Average linkage created a balanced dendrogram and median linkage created a strange looking representation of clusters.

The dendrogram can be cut at the height of number of distinct tissue groups to see if the clusters correlate to the cancer tissue groups.

```
gene_hc = hclust(dist(scaled_df))
clusters = cutree(gene_hc, 7)
table("predicted" = clusters, "actual" = tab$Tissue)
```

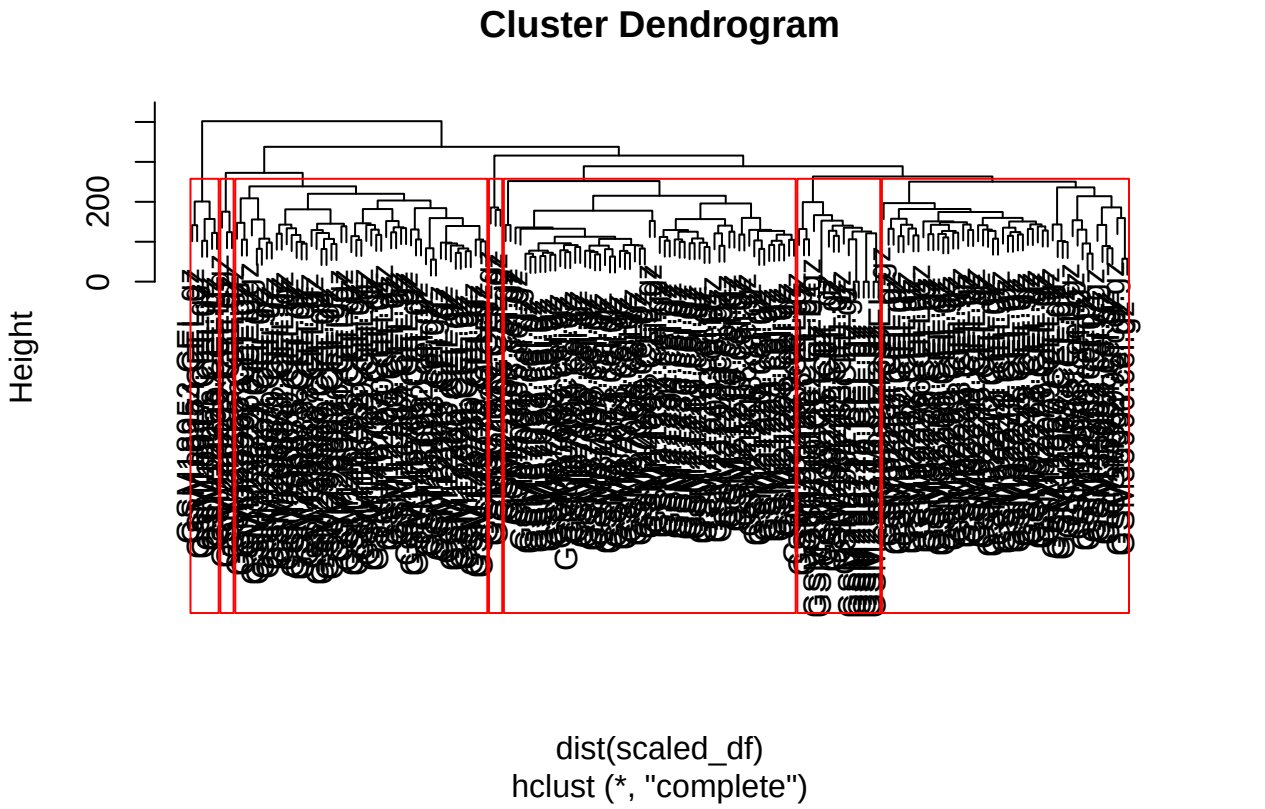
```
##           actual
## predicted cerebellum colon endometrium hippocampus kidney liver placenta
```

```
##          1          8          0         15          0         28          0          0
##          2          0         34          0          0          9          7          0
##          3         28          0          0         31          0          0          0
##          4          0          0          0          0          0         17          0
##          5          2          0          0          0          2          2          0
##          6          0          0          0          0          0          0          3
##          7          0          0          0          0          0          0          3
```

Of the 7 different tissue types, the colon tissues, endometrium tissues and hippocampus tissues all got grouped into its own clusters. However these clusters also had other types of tissues within it. Hierarchical clustering created 3 distinct clusters. No cluster is absolutely clean and properly made.

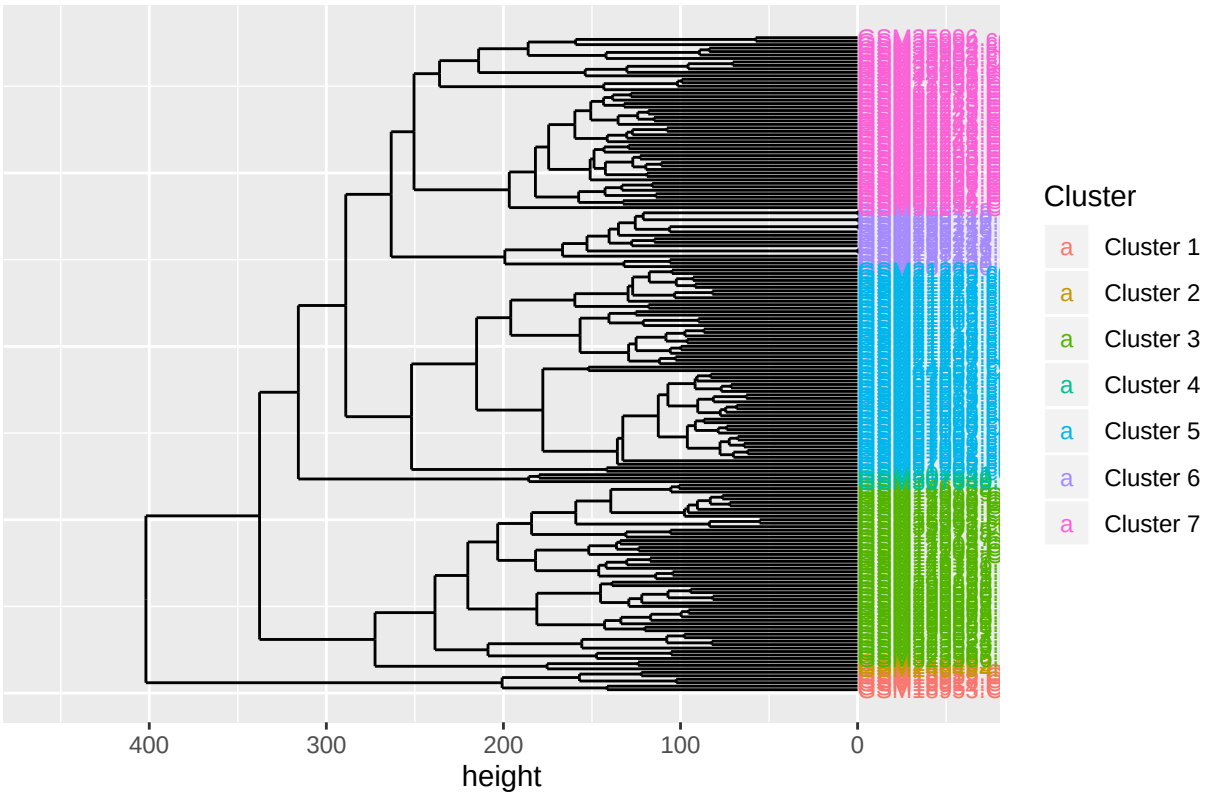
The cluster dendrogram can be plotted.

```
dend_data = dendro_data(gene_hc, type = "rectangle")
plot(gene_hc)
g3 = rect.hclust(gene_hc, k = 7)
```




```
clustered_df = data.frame(num = unlist(g3),
                          clust=rep(c("Cluster 1","Cluster 2", "Cluster 3",
                                      "Cluster 4","Cluster 5", "Cluster 6",
                                      "Cluster 7"),
                                    times = sapply(g3,length)))
text_df = merge(label(dend_data), clustered_df,
                by.x = "label", by.y = "row.names")
ggplot() +
  geom_segment(data=segment(dend_data),
               aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data=text_df,
            aes(x = x, y = y, label = label, hjust = 0, color = clust),
```

```
            size = 3) +
  scale_color_discrete(name = "Cluster") +
  labs(y = "height", x = NULL,
       title = "Scaled Hierarchical Clustering into 7 Clusters") +
  coord_flip() + scale_y_reverse(expand = c(0.2, 0)) +
    theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



The output of the clusters give a summary of the object.

```
gene_hc
```

```
##
## Call:
## hclust(d = dist(scaled_df))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 189
```

The clusters were made using complete linkage and computing the Euclidean distances between features.

Moving on from hierarchical clustering, try doing k-means clustering on the dataset to see if the same numbers of clusters yield the same results. Use $K = 7$.

```
set.seed(7)
gene_km = kmeans(scaled_df, 7, nstart = 20)
km_cluster_gene = gene_km$cluster
```

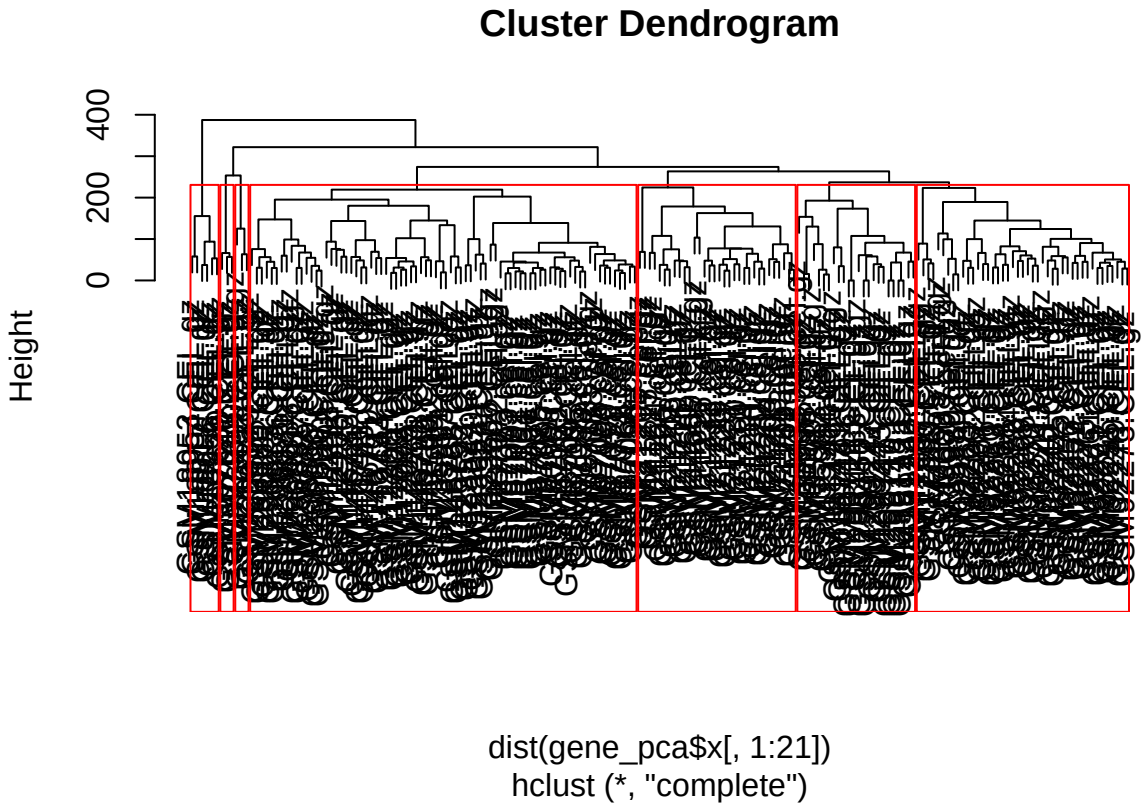
```
table(km_cluster_gene, clusters)
```

```
##                 clusters
## km_cluster_gene  1  2  3  4  5  6  7
##               1  0  7  0 17  0  0  0
##               2  3  0 28  0  0  0  0
##               3  0 34  0  0  0  0  0
##               4  0  0  0  0  0  3  3
##               5 48  8  0  0  0  0  0
##               6  0  1  0  0  6  0  0
##               7  0  0 31  0  0  0  0
```

Interestingly enough, by using k-means clustering, only 2 tissue types got properly clustered into its own cluster with no disturbance. This contrasts with hierarchical clustering where 3 distinct clusters are made.

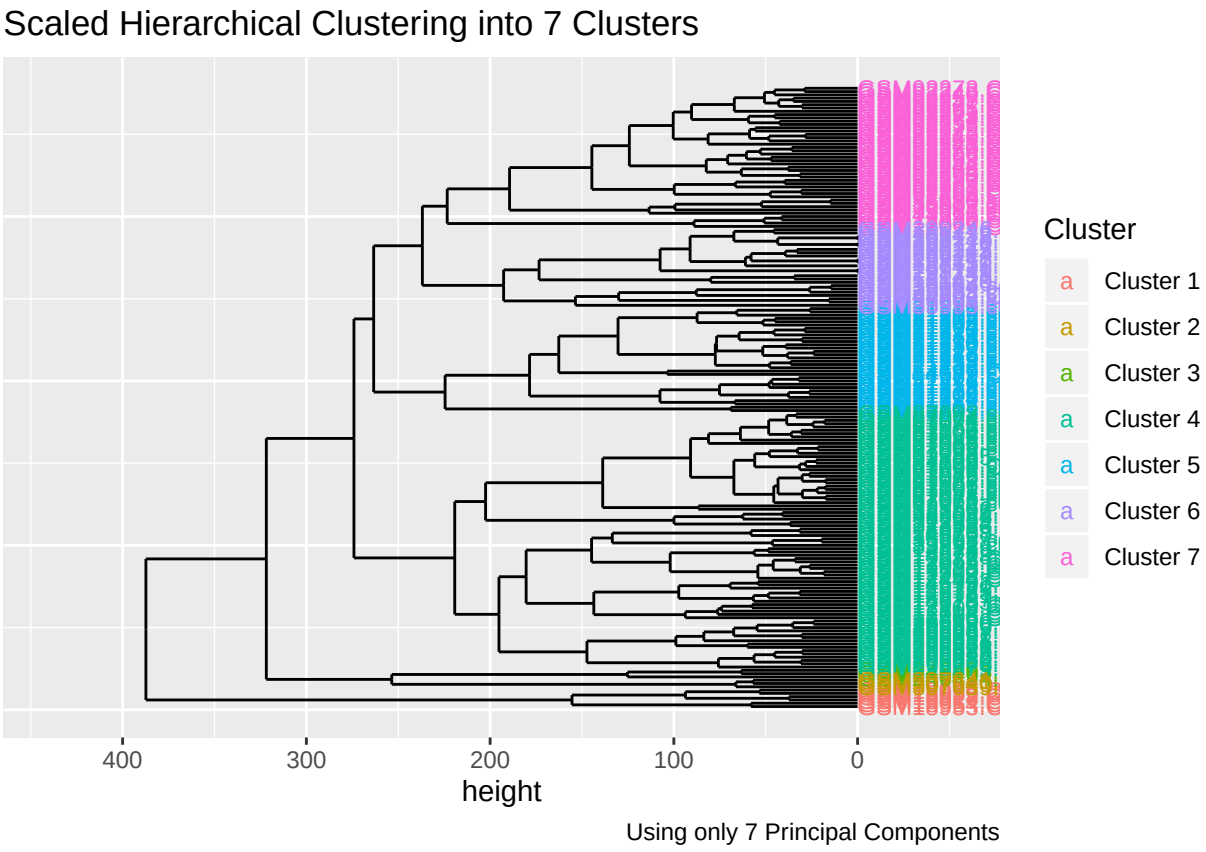
Rather than performing hierarchical clustering on the entire dataset, it can be performed on the first principal score vectors as follows.

```
gene_hc_partial = hclust(dist(gene_pca$x[,1:21]))
dend_data = dendro_data(gene_hc_partial, type = "rectangle")
plot(gene_hc_partial)
g3 = rect.hclust(gene_hc_partial, k = 7)
```




```
clustered_df = data.frame(num = unlist(g3),
                          clust=rep(c("Cluster 1","Cluster 2", "Cluster 3",
                                      "Cluster 4","Cluster 5", "Cluster 6",
                                      "Cluster 7"),
                                    times = sapply(g3,length)))
text_df = merge(label(dend_data), clustered_df,
                by.x = "label", by.y = "row.names")
```

```
ggplot() +
  geom_segment(data=segment(dend_data),
               aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data=text_df,
            aes(x = x, y = y, label = label, hjust = 0, color = clust),
            size = 3) +
  scale_color_discrete(name = "Cluster") +
  labs(y = "height", x = NULL,
       title = "Scaled Hierarchical Clustering into 7 Clusters",
       caption = "Using only 7 Principal Components") +
  coord_flip() + scale_y_reverse(expand = c(0.2, 0)) +
    theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```



By only using a few of the principal components, the clusters are formed differently.

All of the lab instructions in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.