Class09: Mini Project

Tasnia Sharia (PID A15931128)

10/26/2021

In this project, we will be analyzing data describing the characteristics of cell nuclei found in breast cancer

We first need to download and import the data that is already located in the project directory

```r
#save the data as a variable
data <- "WisconsinCancer.csv"
#inputting data and ensuring column names are set correctly
wisc.df <- read.csv(data, row.names = 1)
View(wisc.df)
```

The diagnosis column will not be used in the in our analysis so we will omit the first column

```r
wisc.data <- wisc.df[,-1]
View(wisc.data)
```

We will save the data in the diagnosis column in a vector to be used later in our analysis

```r
diagnosis <- wisc.df[,1]
View(diagnosis)
```

Q1. How many observations are in this dataset?

```r
#dim() gives number of rows and columns.
dim(wisc.data)
```

```
## [1] 569  30
```

```r
#length() outputs length of vectors, lists, or factors.
length(wisc.data)
```

```
## [1] 30
```

```r
#nrow() outputs number of rows
nrow(wisc.data)
```

```
## [1] 569
```

```r
length(diagnosis)
```

```
## [1] 569
```

The wisc.data has 569 rows and 30 columns of data. The diagnosis vector has 569 data values total.

> **Q2. How many of the observations have a malignant diagnosis?**

```r
#table() outputs a contingency table of displaying the amount of repeated inputs
table(diagnosis)
```

```
## diagnosis
##   B   M
## 357 212
```

There are 212 observations that have a malignant diagnosis.

> **Q3. How many variables/features in the data are suffixed with _mean?**

```r
#grep() finds specific matches to the argument pattern in each element of character vectors
#This outputs which columns have the suffix"_mean"
grep("_mean", colnames(wisc.data))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
length(grep("_mean", colnames(wisc.data)))
```

```
## [1] 10
```

Using length() we can see that there are 10 observation suffized with "_mean".

#Let's do a PCA analysis on this dataset!

We need to check the mean and standard deviation of the features (columns) of the wisc.data to determine if the data should be scaled.

```r
# Check column means and standard deviations
colMeans(wisc.data)
```

```
##             radius_mean            texture_mean          perimeter_mean
##            1.412729e+01            1.928965e+01            9.196903e+01
##               area_mean         smoothness_mean        compactness_mean
##            6.548891e+02            9.636028e-02            1.043410e-01
##          concavity_mean     concave.points_mean           symmetry_mean
##            8.879932e-02            4.891915e-02            1.811619e-01
##  fractal_dimension_mean               radius_se              texture_se
##            6.279761e-02            4.051721e-01            1.216853e+00
##            perimeter_se                 area_se           smoothness_se
##            2.866059e+00            4.033708e+01            7.040979e-03
##          compactness_se            concavity_se       concave.points_se
##             2.547814e-02            3.189372e-02            1.179614e-02
##              symmetry_se    fractal_dimension_se            radius_worst
##             2.054230e-02            3.794904e-03            1.626919e+01
##            texture_worst         perimeter_worst              area_worst
##             2.567722e+01            1.072612e+02            8.805831e+02
##         smoothness_worst       compactness_worst         concavity_worst
##             1.323686e-01            2.542650e-01            2.721885e-01
##     concave.points_worst          symmetry_worst fractal_dimension_worst
##             1.146062e-01            2.900756e-01            8.394582e-02
```

```
apply(wisc.data,2,sd)
```

```
##              radius_mean            texture_mean          perimeter_mean
##             3.524049e+00            4.301036e+00            2.429898e+01
##                area_mean         smoothness_mean        compactness_mean
##             3.519141e+02            1.406413e-02            5.281276e-02
##           concavity_mean     concave.points_mean           symmetry_mean
##             7.971981e-02            3.880284e-02            2.741428e-02
##   fractal_dimension_mean               radius_se              texture_se
##             7.060363e-03            2.773127e-01            5.516484e-01
##             perimeter_se                 area_se           smoothness_se
##             2.021855e+00            4.549101e+01            3.002518e-03
##           compactness_se            concavity_se       concave.points_se
##             1.790818e-02            3.018606e-02            6.170285e-03
##              symmetry_se    fractal_dimension_se            radius_worst
##             8.266372e-03            2.646071e-03            4.833242e+00
##            texture_worst         perimeter_worst              area_worst
##             6.146258e+00            3.360254e+01            5.693570e+02
##         smoothness_worst       compactness_worst         concavity_worst
##             2.283243e-02            1.573365e-01            2.086243e-01
##     concave.points_worst          symmetry_worst fractal_dimension_worst
##             6.573234e-02            6.186747e-02            1.806127e-02
```

We need to use scale=TRUE in this case for the PCA analysis as the columns data are on different scales.

```
# Perform PCA on wisc.data
wisc.pr <- prcomp(wisc.data, scale=TRUE)
```

Now we will take a look at the summary of the results

```
summary(wisc.pr)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                            PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                           PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

Q4. From your results, what proportion of the original variance is captured by the first principal components (PC1)?

Based on the summary results, there is a 44.27% cumulative proportion captured by PC1.

Q5. How many principal components (PCs) are required to describe at least 70% of the original variance in the data?

Based on the summary results, about 3 PCs are required to describe at least 70% of the original variance in the data.

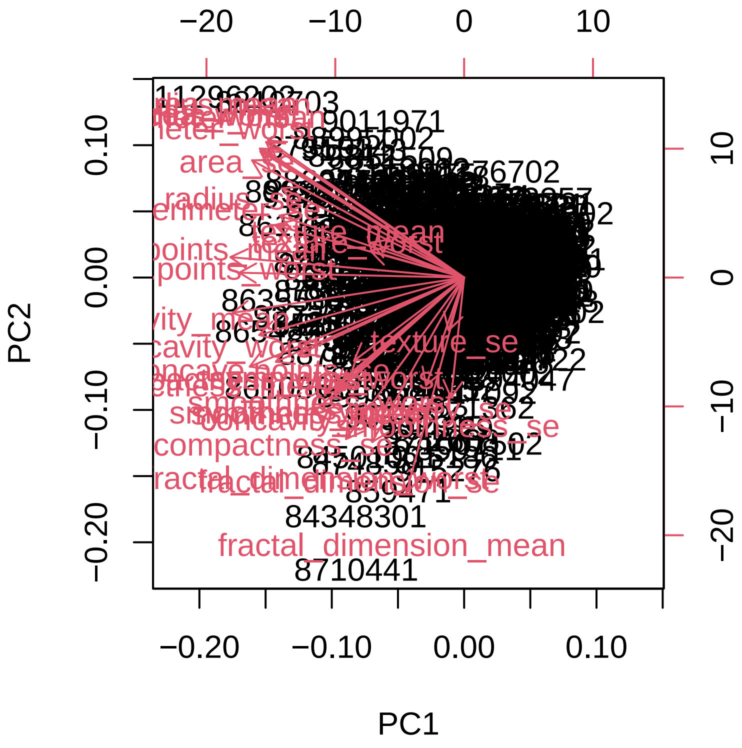
Q6. How many principal components (PCs) are required to describe at least 90% of the original variance in the data?

Based on the summary results, about 7 PCs are required to describe at least 90% of the original variance in the data.

#Interpretting PCA Results

We will create a some visualizations to help understand the PCA results. We will create a biplot.

```
biplot(wisc.pr)
```

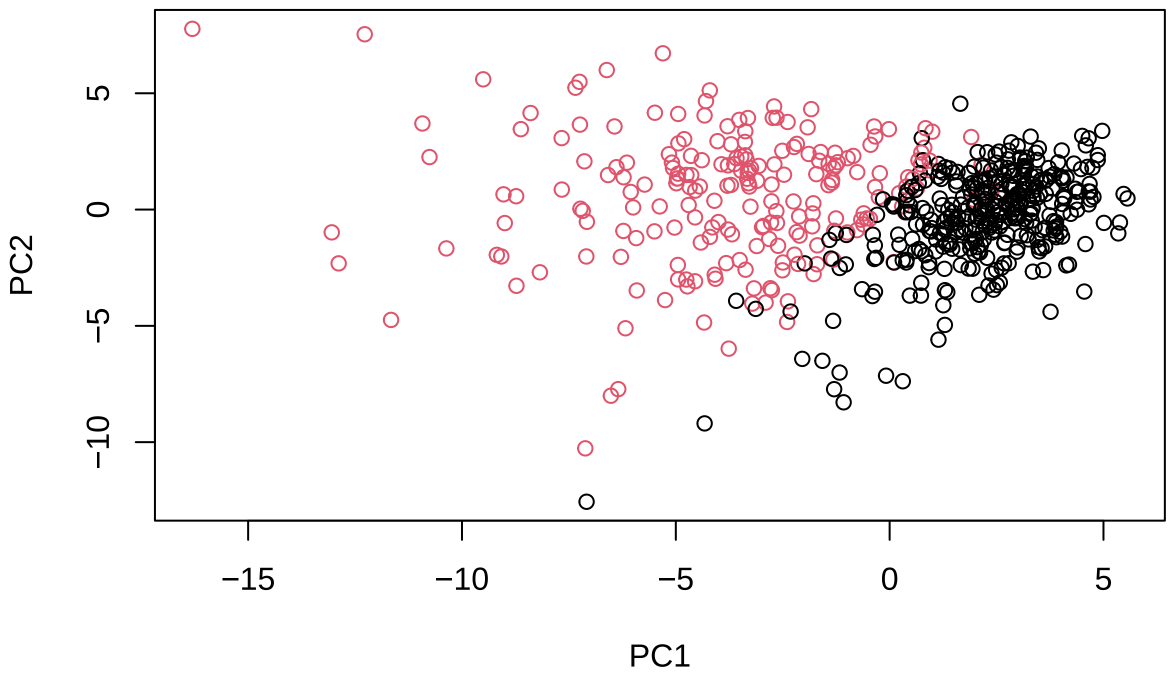


> **Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?**

There is a big black blob of data inputs with the columns names spread out and pointing in the center. It is not easy to understand because I do not know what the axis are scaled too to represent.

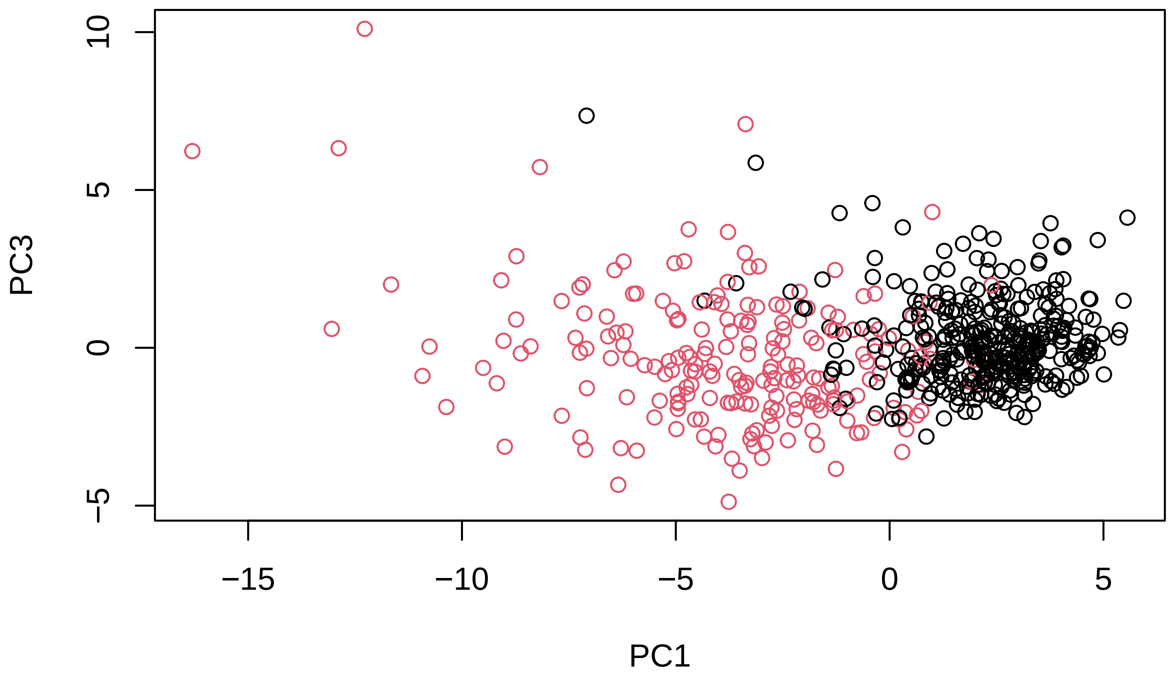
We will generate our own scatterplot to make sense of the PCA results

```
# Scatter plot observations by components 1 and 2
plot(wisc.pr$x[,1], wisc.pr$x[,2], col=as.factor(diagnosis), xlab = "PC1",
     ylab ="PC2")
```



Q8. Generate a similar plot for principal components 1 and 3. What do you notice about these plots?

```
# Scatter plot observations by components 1 and 3
plot(wisc.pr$x[,1], wisc.pr$x[,3], col=as.factor(diagnosis), xlab = "PC1",
     ylab ="PC3")
```



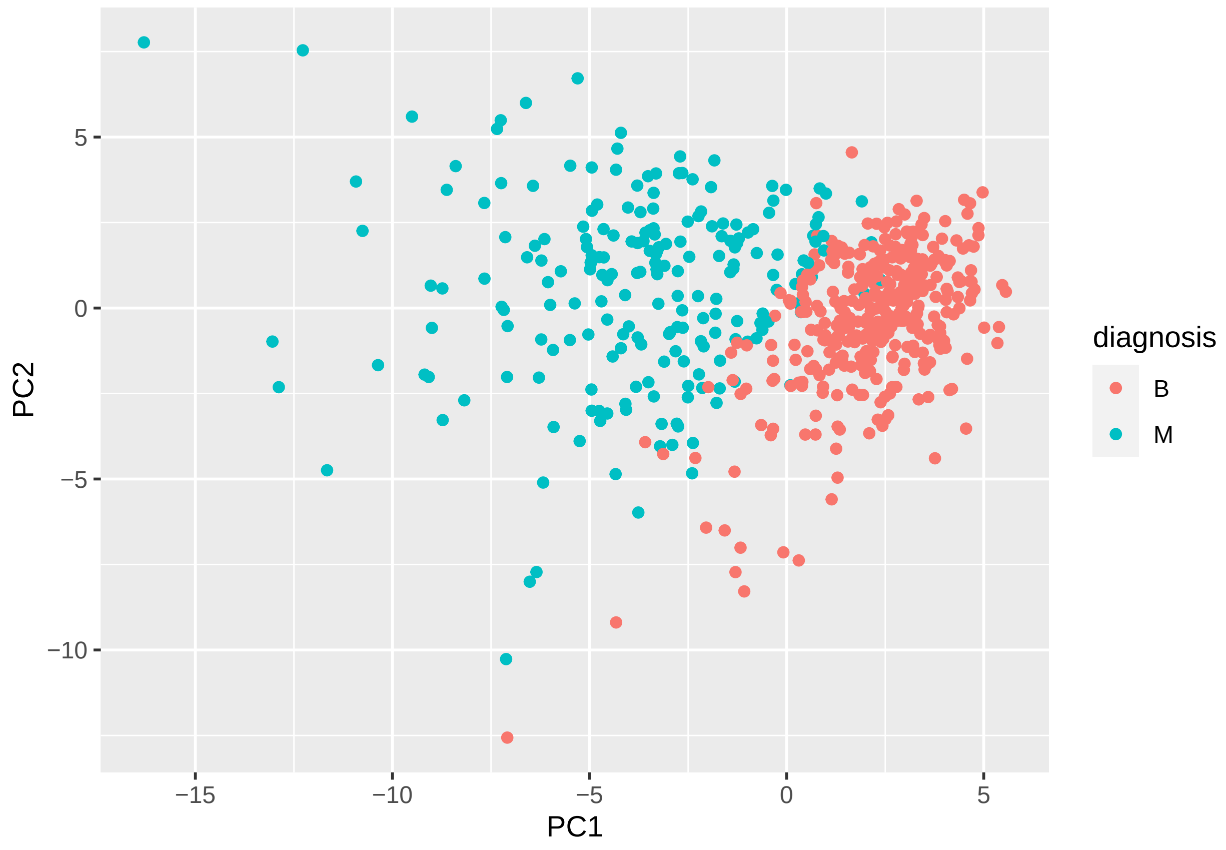
PC2 captures more variance in the original data than PC3, so the first plot appears better for separating the two subgroups of benign (black) and malignant (red) samples.

Create a more aesthetic figure using ggplot!

```r
# Create a data.frame for ggplot
df <- as.data.frame(wisc.pr$x)
df$diagnosis <- diagnosis

# Load the ggplot2 package
library(ggplot2)

# Make a scatter plot colored by diagnosis
ggplot(df) +
  aes(PC1, PC2, col=diagnosis) +
  geom_point()
```



Variance explained

We will produce scree plots showing the proportion of variance explained as the number of PCs increases.

First, we calculate the variance of each PC by squaring the sdev component of wisc.pr

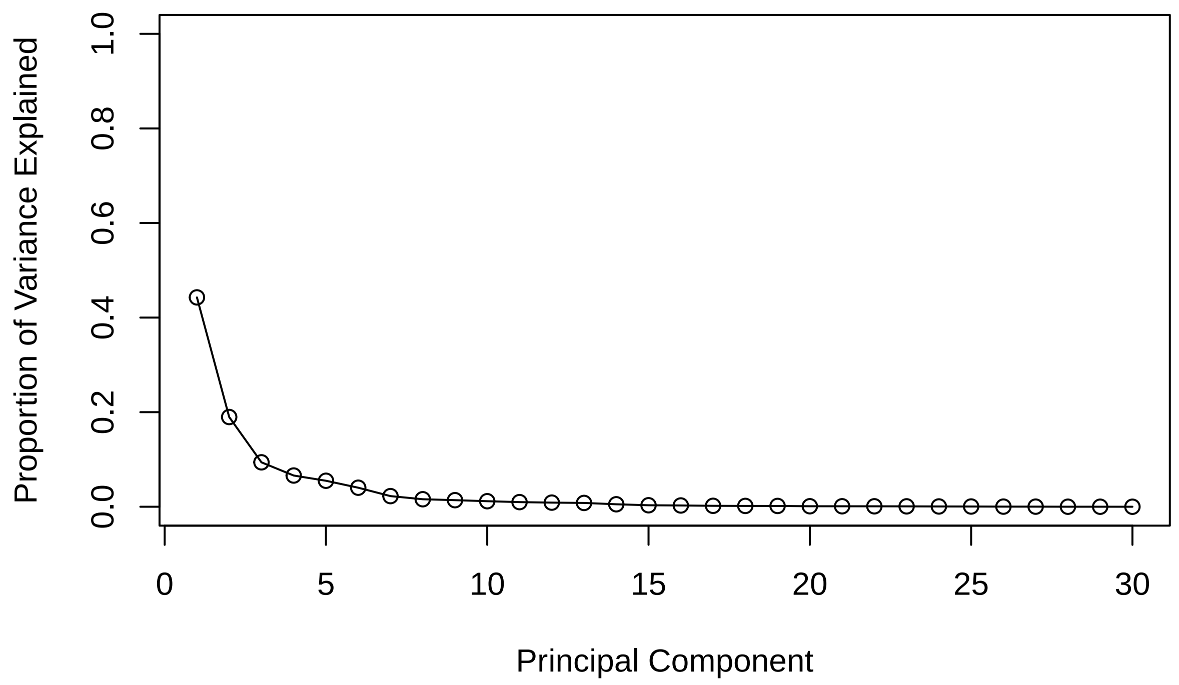
```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

```
## [1] 13.281608  5.691355  2.817949  1.980640  1.648731  1.207357
```

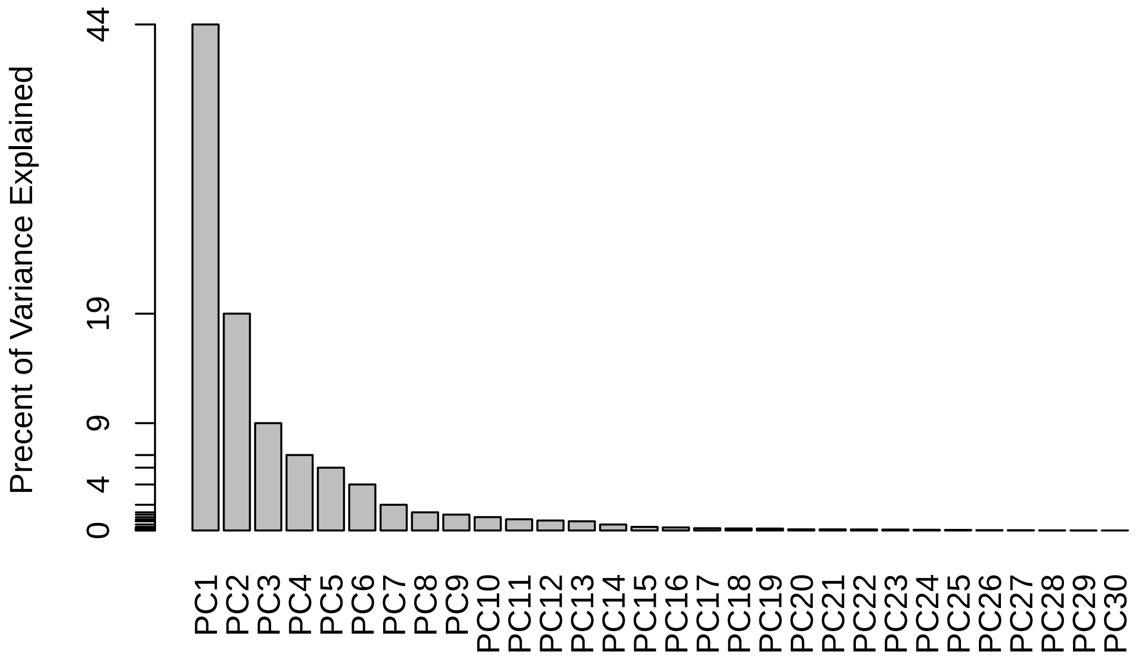
Then, we calculate the variance explained by each PC by dividing by the total variance explained of all PCs.

```
# Variance explained by each principal component: pve
pve <- pr.var / sum(pr.var)

# Plot variance explained for each principal component
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "o")
```



```
# Alternative scree plot of the same data, note data driven y-axis
barplot(pve, ylab = "Precent of Variance Explained",
     names.arg=paste0("PC",1:length(pve)), las=2, axes = FALSE)
axis(2, at=pve, labels=round(pve,2)*100 )
```



Communicating PCA Results

We will check our understanding of the PCA results like the loadings and variance explained.

> **Q9. For the first principal component, what is the component of the loading vector (i.e. wisc.pr$rotation[,1]) for the feature concave.points_mean?**

```
wisc.pr$rotation["concave.points_mean", 1]
```

```
## [1] -0.2608538
```

The component for the concave.points_mean is -0.2608538.

> **Q10. What is the minimum number of principal components required to explain 80% of the variance of the data?**

```
summary(wisc.pr)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759 0.91010
##                            PC8    PC9    PC10   PC11    PC12    PC13    PC14
## Standard deviation     0.69037 0.6457 0.59219 0.5421 0.51104 0.49128 0.39624
## Proportion of Variance 0.01589 0.0139 0.01169 0.0098 0.00871 0.00805 0.00523
## Cumulative Proportion  0.92598 0.9399 0.95157 0.9614 0.97007 0.97812 0.98335
##                           PC15    PC16    PC17    PC18    PC19    PC20   PC21
## Standard deviation     0.30681 0.28260 0.24372 0.22939 0.22244 0.17652 0.1731
## Proportion of Variance 0.00314 0.00266 0.00198 0.00175 0.00165 0.00104 0.0010
## Cumulative Proportion  0.98649 0.98915 0.99113 0.99288 0.99453 0.99557 0.9966
##                           PC22    PC23   PC24    PC25    PC26    PC27    PC28
## Standard deviation     0.16565 0.15602 0.1344 0.12442 0.09043 0.08307 0.03987
## Proportion of Variance 0.00091 0.00081 0.0006 0.00052 0.00027 0.00023 0.00005
## Cumulative Proportion  0.99749 0.99830 0.9989 0.99942 0.99969 0.99992 0.99997
##                           PC29    PC30
## Standard deviation     0.02736 0.01153
## Proportion of Variance 0.00002 0.00000
## Cumulative Proportion  1.00000 1.00000
```

Based on the summary, we would need minimum 4 PCs to capture at least 80% of the variance of the data.

Hierarchical Clustering

The distance between all pairs of observations are computed.

```
# Scale the wisc.data data using the "scale()" function
data.scaled <- scale(wisc.data)
```

Calculate the distance between all pairs in the new scaled dataset

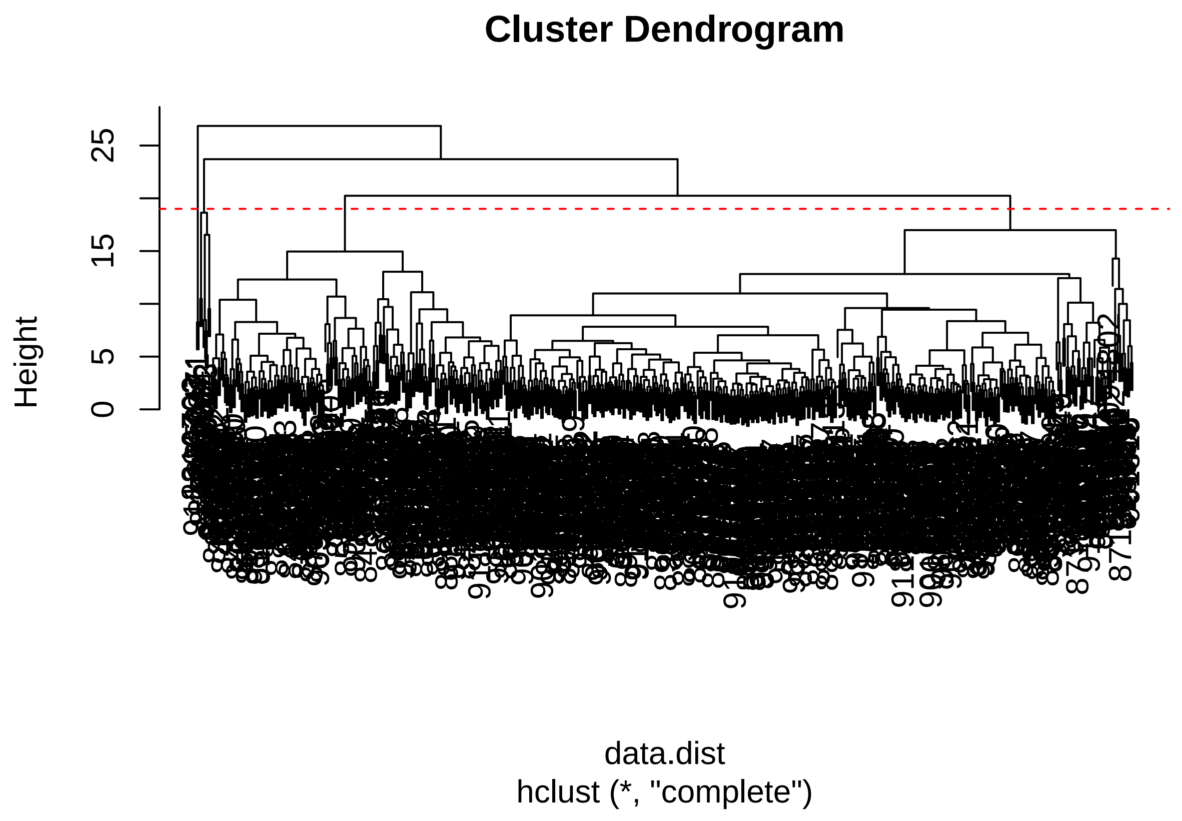
```
data.dist <- dist(data.scaled)
```

Create a hierarchical clustering model using complete linkage.

```
wisc.hclust <- hclust(data.dist)
```

> **Q11. Using the plot() and abline() functions, what is the height at which the clustering model has 4 clusters?**

```
# Viewing the plot
plot(wisc.hclust)
#adding a line to view height at which 4 clusters are made
abline(h=19, col="red", lty=2)
```



Around height 19, the clustering model has 4 clusters

Selecting number of clusters

We will compare the outputs from your hierarchical clustering model to the actual diagnoses.

```r
#using cutree to cut the tree to make 4 clusters
wisc.hclust.clusters <- cutree(wisc.hclust, k=4)

#use table() function to compare the cluster membership to the actual diagnoses
table(wisc.hclust.clusters, diagnosis)
```

```
##                     diagnosis
## wisc.hclust.clusters   B   M
##                    1  12 165
##                    2   2   5
##                    3 343  40
##                    4   0   2
```

> **Q12. Can you find a better cluster vs diagnoses match by cutting into a different number of clusters between 2 and 10?**

```r
wisc.hclust.clusters2 <- cutree(wisc.hclust, k=2)
table(wisc.hclust.clusters2, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters2   B   M
##                     1 357 210
##                     2   0   2
```

```
wisc.hclust.clusters3 <- cutree(wisc.hclust, k=3)
table(wisc.hclust.clusters3, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters3   B   M
##                     1 355 205
##                     2   2   5
##                     3   0   2
```

```
wisc.hclust.clusters5 <- cutree(wisc.hclust, k=5)
table(wisc.hclust.clusters5, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters5   B   M
##                     1  12 165
##                     2   0   5
##                     3 343  40
##                     4   2   0
##                     5   0   2
```

```
wisc.hclust.clusters6 <- cutree(wisc.hclust, k=6)
table(wisc.hclust.clusters6, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters6   B   M
##                     1  12 165
##                     2   0   5
##                     3 331  39
##                     4   2   0
##                     5  12   1
##                     6   0   2
```

```
wisc.hclust.clusters7 <- cutree(wisc.hclust, k=7)
table(wisc.hclust.clusters7, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters7   B   M
##                     1  12 165
##                     2   0   3
##                     3 331  39
##                     4   2   0
##                     5  12   1
##                     6   0   2
##                     7   0   2
```

```r
wisc.hclust.clusters8 <- cutree(wisc.hclust, k=8)
table(wisc.hclust.clusters8, diagnosis)
```

```
##                      diagnosis
## wisc.hclust.clusters8   B   M
##                     1  12  86
##                     2   0  79
##                     3   0   3
##                     4 331  39
##                     5   2   0
##                     6  12   1
##                     7   0   2
##                     8   0   2
```

```r
wisc.hclust.clusters9 <- cutree(wisc.hclust, k=9)
table(wisc.hclust.clusters9, diagnosis)
```

```
##                      diagnosis
## wisc.hclust.clusters9   B   M
##                     1  12  86
##                     2   0  79
##                     3   0   3
##                     4 331  39
##                     5   2   0
##                     6  12   0
##                     7   0   2
##                     8   0   2
##                     9   0   1
```

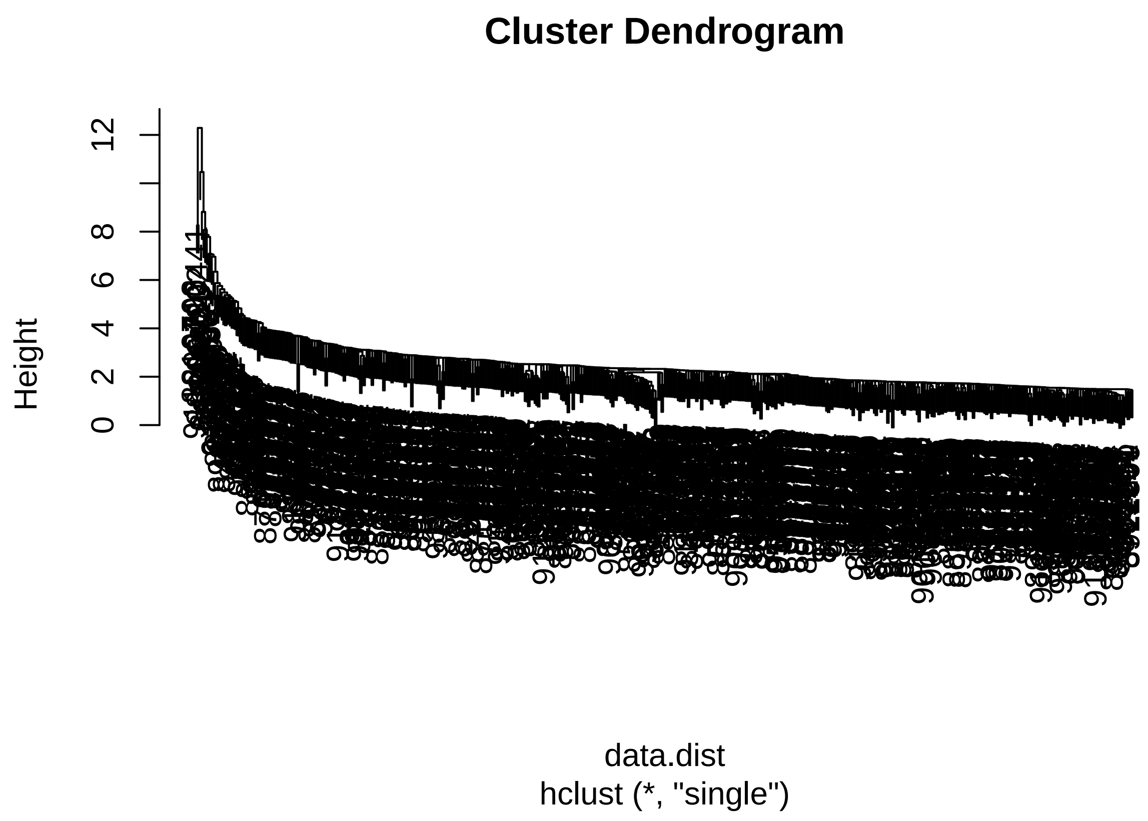
```r
wisc.hclust.clusters10 <- cutree(wisc.hclust, k=10)
table(wisc.hclust.clusters10, diagnosis)
```

```
##                       diagnosis
## wisc.hclust.clusters10   B   M
##                     1   12  86
##                     2    0  59
##                     3    0   3
##                     4  331  39
##                     5    0  20
##                     6    2   0
##                     7   12   0
##                     8    0   2
##                     9    0   2
##                     10   0   1
```

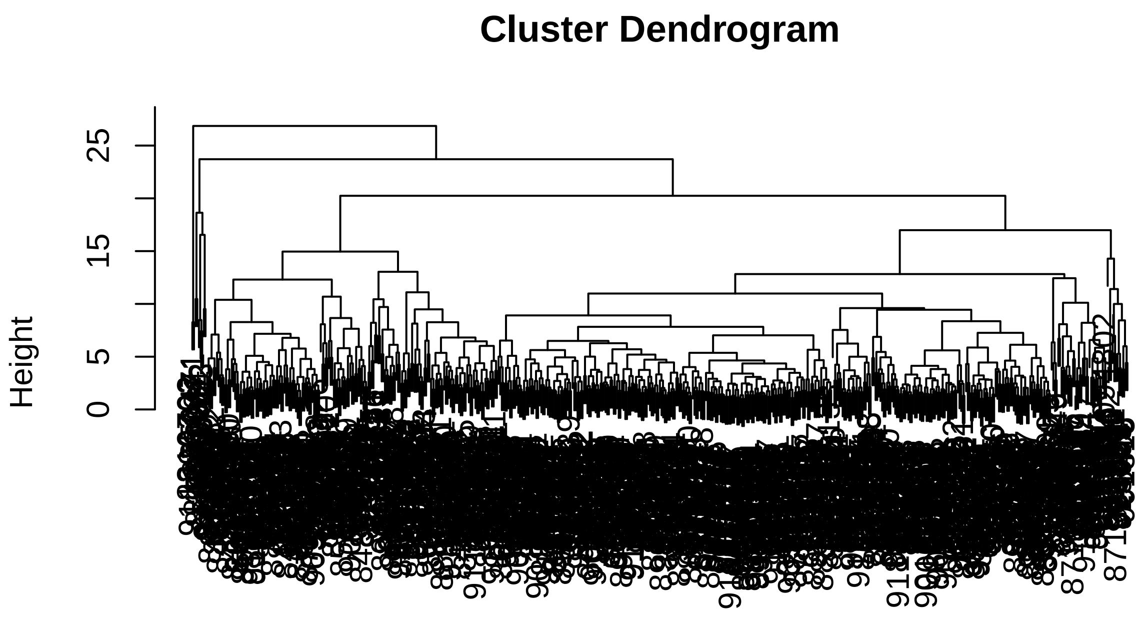
A lower number of clusters would provide better analyses. 2-5 clusters seem appropriate.

> **Q13. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning.**

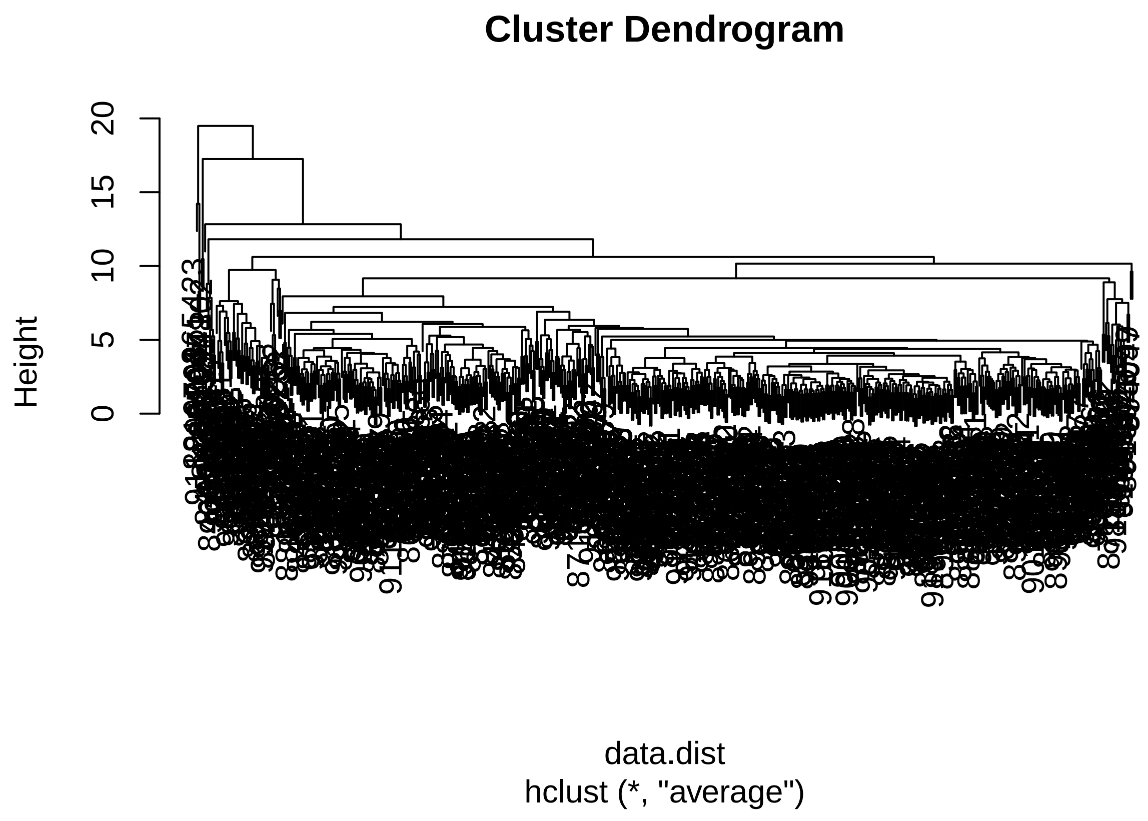
```
# single method
wisc.hclust.single <- hclust(data.dist, method= "single" )
plot(wisc.hclust.single)
```



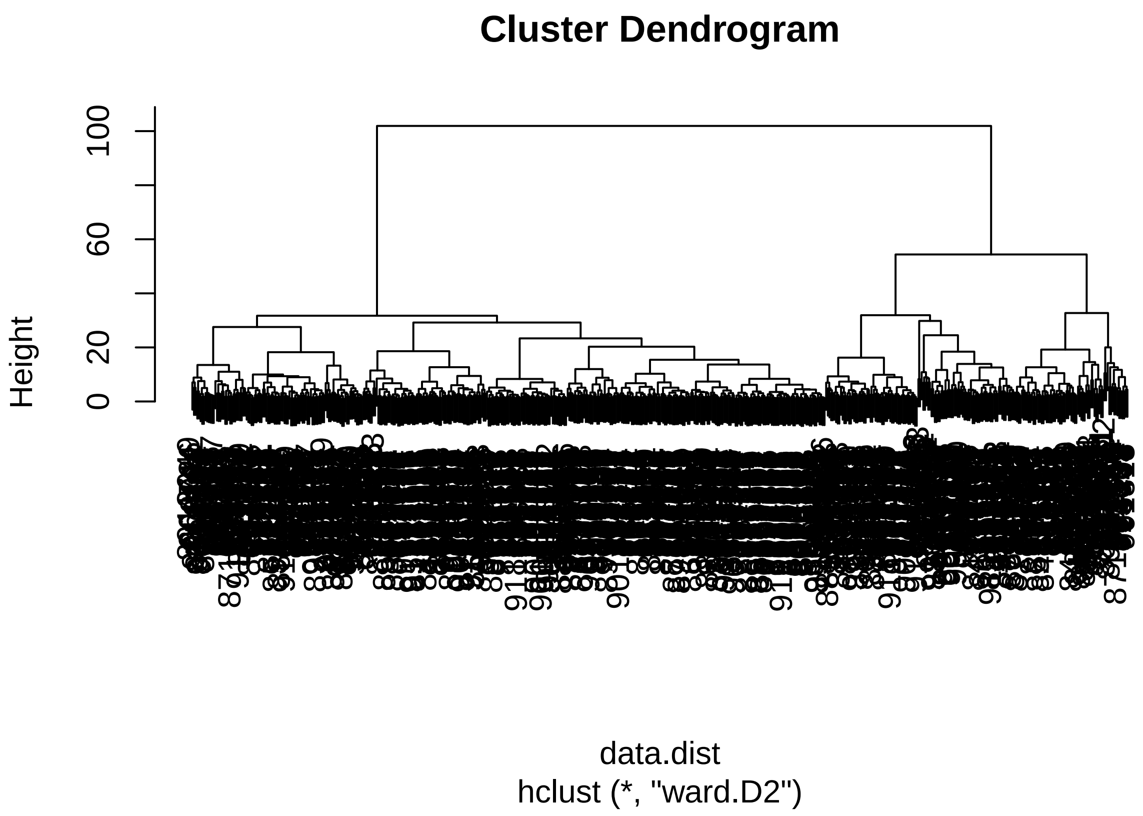
```
# Complete method
wisc.hclust.complete <- hclust(data.dist, method= "complete" )
plot(wisc.hclust.complete)
```



```
# Average method
wisc.hclust.average <- hclust(data.dist, method= "average" )
plot(wisc.hclust.average)
```



```
# Ward.D2 method
wisc.hclust.ward.D2 <- hclust(data.dist, method= "ward.D2" )
plot(wisc.hclust.ward.D2)
```



They all appear crowded, others to a more confusing extent, but overall the ward.D2 method seems most similar in appearance and more clean.

K-means clustering

We will create a k-means clustering model on the data and compare the results to the actual diagnoses and results of the hierarchical clustering model.

```
#creating k-means with the scaled data created for the hierarchical clustering
#Making 2 clusters and running algorithm 20 times
wisc.km <- kmeans(data.scaled, centers=2, nstart= 20)

#use table() function to compare the cluster membership of the k-means model to the actual diagnoses co
table(wisc.km$cluster, diagnosis)
```

```
##    diagnosis
##        B   M
##   1 343  37
##   2  14 175
```

Q14. How well does k-means separate the two diagnoses? How does it compare to your hclust results?

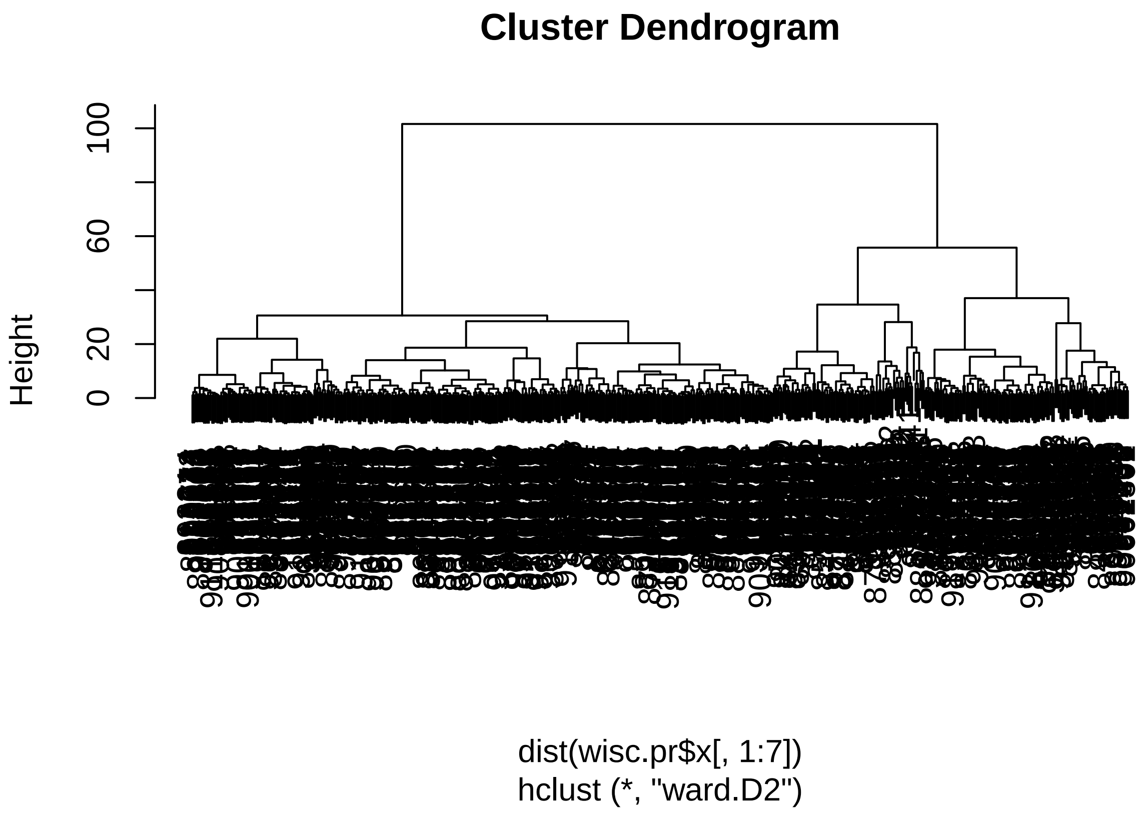
```
table(wisc.km$cluster, wisc.hclust.clusters)
```

```
##    wisc.hclust.clusters
##       1   2   3   4
##   1  17   0 363   0
##   2 160   7  20   2
```

Clusters 1, 2, and 4 from the hierarchical clustering model can be interpreted as the cluster 1 for the k-means algorithm. Cluster 3 from the hierarchical clustering can be interpreted as the cluster 2 for k-means.

#Combining methods We will apply PCA results to hierarchical clustering.

```
wisc.pr.hclust <- hclust(dist(wisc.pr$x[,1:7]), method="ward.D2")
plot(wisc.pr.hclust)
```



This appears better than our previous clustering results on the original scaled data. There are 2 main branches in this dendrogram indicating two clusters that could possibly represent the malignant and benign samples.

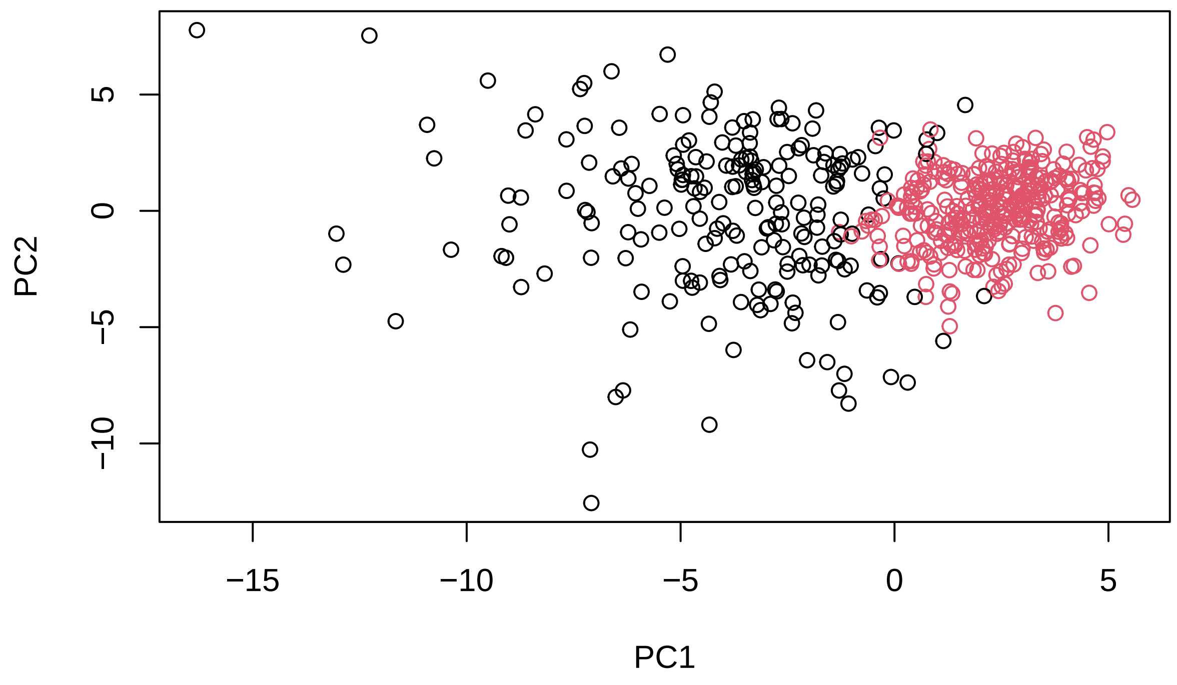
```
#creating 2 clusters and a table to view what samples are in each cluster
grps <- cutree(wisc.pr.hclust, k=2)
table(grps)
```

```
## grps
##   1   2
## 216 353
```

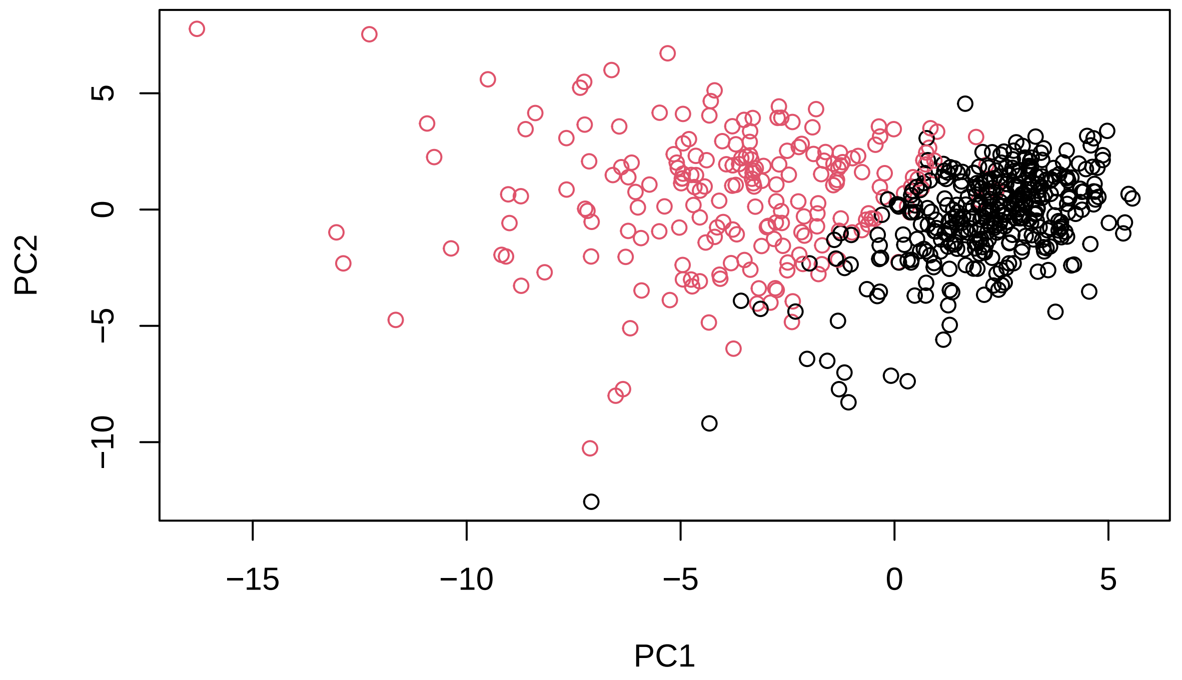
```r
#seeing if the 2 branches represent M and B samples
table(grps, diagnosis)
```

```
##     diagnosis
## grps   B   M
##    1  28 188
##    2 329  24
```

```r
#plotting the results using grps to color
plot(wisc.pr$x[,1:2], col=grps)
```



```r
#plotting the results using diagnosis vector to color
plot(wisc.pr$x[,1:2], col=as.factor(diagnosis))
```



To match things, we can turn our groups into a factor and reorder the levels so cluster 2 comes first and gets the first color (black) and cluster 1 gets the second color (red).

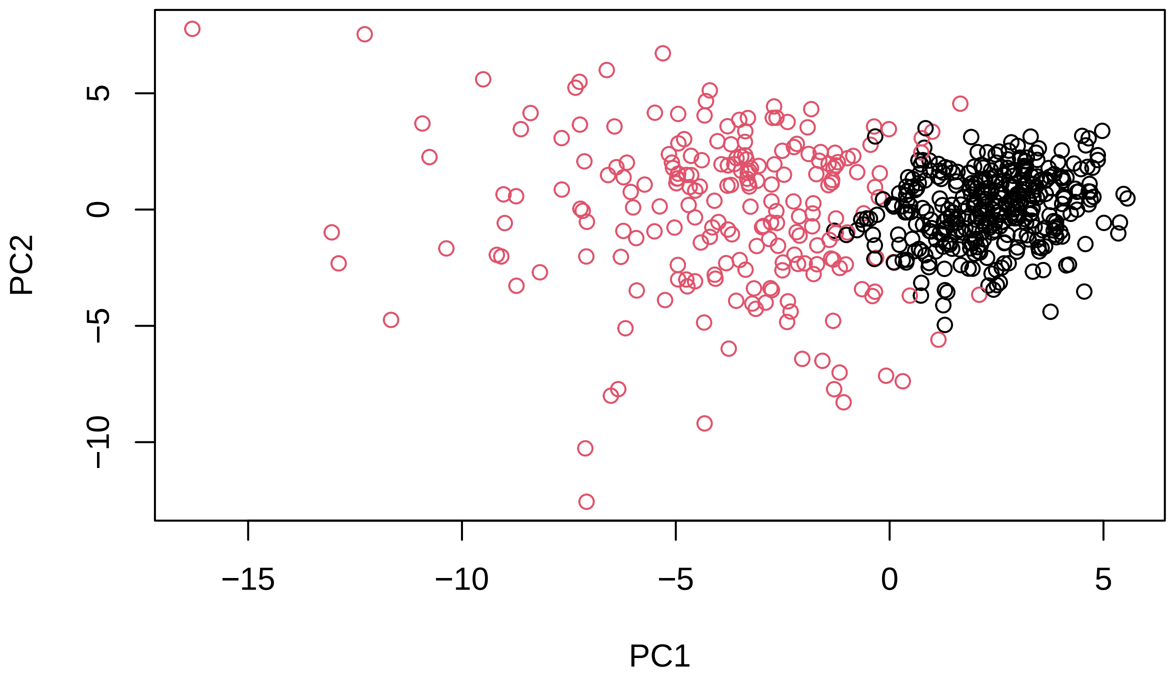
```
g <- as.factor(grps)
levels(g)
```

```
## [1] "1" "2"
```

```
g <- relevel(g,2)
levels(g)
```

```
## [1] "2" "1"
```

```
# Plot using our re-ordered factor
plot(wisc.pr$x[,1:2], col=g)
```



```r
#Use the distance along the first 7 PCs for clustering
wisc.pr.hclust <- hclust(dist(wisc.pr$x[,1:7]), method="ward.D2")

#cut 2 clusters
wisc.pr.hclust.clusters <- cutree(wisc.pr.hclust, k=2)
```

> **Q15. How well does the newly created model with four clusters separate out the two diagnoses?**

```r
# Compare to actual diagnoses
table(wisc.pr.hclust.clusters, diagnosis)
```

```
##                          diagnosis
## wisc.pr.hclust.clusters   B   M
##                        1  28 188
##                        2 329  24
```

Cluster 1 contains more malignant samples and cluster 2 has more benign.

> **Q16. How well do the k-means and hierarchical clustering models you created in previous sections (i.e. before PCA) do in terms of separating the diagnoses? Again, use the table() function to compare the output of each model (wisc.km$cluster and wisc.hclust.clusters) with the vector containing the actual diagnoses.**

```r
table(wisc.km$cluster, diagnosis)
```

```
##    diagnosis
##       B   M
##   1 343  37
##   2  14 175
```

```r
table(wisc.hclust.clusters, diagnosis)
```

```
##                     diagnosis
## wisc.hclust.clusters   B   M
##                    1  12 165
##                    2   2   5
##                    3 343  40
##                    4   0   2
```

```r
#relooking at what the actual amount of M and B samples exist
table(diagnosis)
```

```
## diagnosis
##   B   M
## 357 212
```

The k-means and hierarchical clustering both do significantly well in separating the M and B samples. K-means seems more similar to the separation of the actual diagnosis

Sensitivity/Specificity

> **Q17. Which of your analysis procedures resulted in a clustering model with the best specificity? How about sensitivity?'**

```r
table(wisc.hclust.clusters, diagnosis)
```

```
##                     diagnosis
## wisc.hclust.clusters   B   M
##                    1  12 165
##                    2   2   5
##                    3 343  40
##                    4   0   2
```

Best analysis for specificity would be the k-means model. And the best analysis for sensitivity would be the clustering model.

Prediction

We will use the predict() function that will take our PCA model from the breat cancer dataset and new cancer cell data and project that data

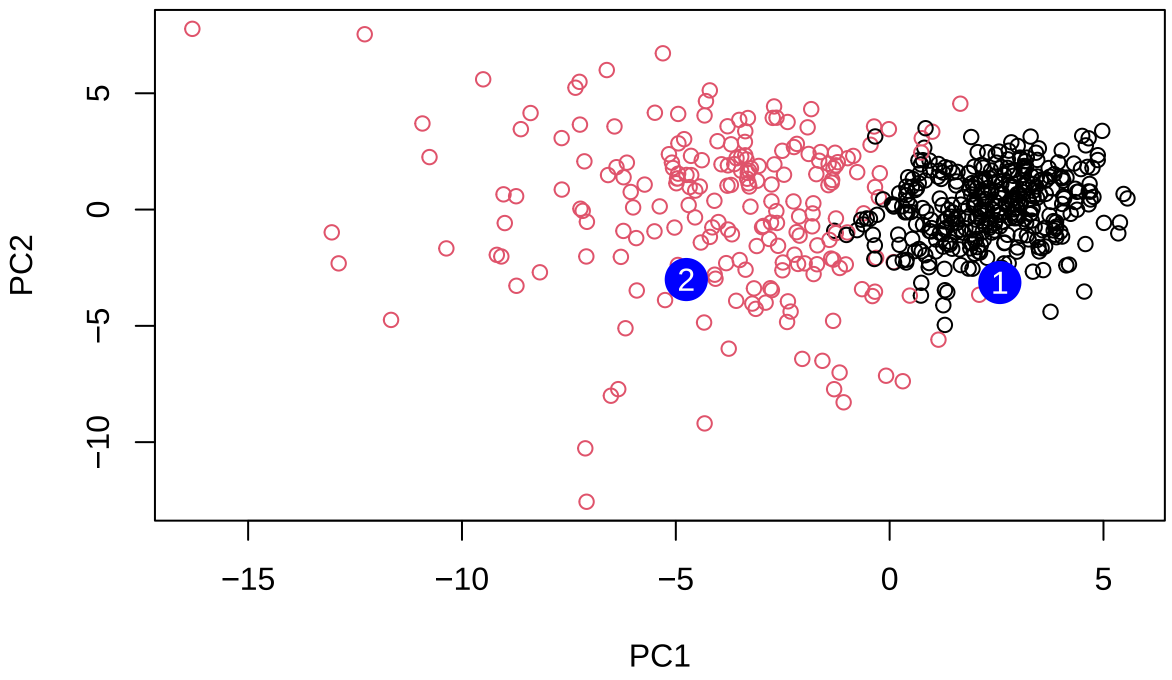
```
#first we need to import the new data
new <- read.csv("new_samples.csv")

#predicting the data
npc <- predict(wisc.pr, newdata=new)
npc
```

```
##            PC1       PC2        PC3        PC4       PC5        PC6        PC7
## [1,]  2.576616 -3.135913  1.3990492 -0.7631950  2.781648 -0.8150185 -0.3959098
## [2,] -4.754928 -3.009033 -0.1660946 -0.6052952 -1.140698 -1.2189945  0.8193031
##             PC8       PC9       PC10      PC11      PC12      PC13     PC14
## [1,] -0.2307350 0.1029569 -0.9272861 0.3411457  0.375921 0.1610764 1.187882
## [2,] -0.3307423 0.5281896 -0.4855301 0.7173233 -1.185917 0.5893856 0.303029
##           PC15       PC16        PC17        PC18        PC19       PC20
## [1,] 0.3216974 -0.1743616 -0.07875393 -0.11207028 -0.08802955 -0.2495216
## [2,] 0.1299153  0.1448061 -0.40509706  0.06565549  0.25591230 -0.4289500
##            PC21       PC22       PC23       PC24        PC25         PC26
## [1,]  0.1228233 0.09358453 0.08347651  0.1223396  0.02124121  0.078884581
## [2,] -0.1224776 0.01732146 0.06316631 -0.2338618 -0.20755948 -0.009833238
##              PC27        PC28         PC29         PC30
## [1,]  0.220199544 -0.02946023 -0.015620933  0.005269029
## [2,] -0.001134152  0.09638361  0.002795349 -0.019015820
```

Creating a new plot to compare the prediction

```
plot(wisc.pr$x[,1:2], col=g)
points(npc[,1], npc[,2], col="blue", pch=16, cex=3)
text(npc[,1], npc[,2], c(1,2), col="white")
```



Q18. Which of these new patients should we prioritize for follow up based on your results?

Patient 2