Class 8: Machine Learning Mini Project

Chloe Do

1. Exploratory data analysis

Data Import

I downloaded this file and saved it into my project directory

```
# Save your input data file into your Project directory
fna.data <- "WisconsinCancer (1).csv"

# Complete the following code to input the data and store as wisc.df
wisc.df <- read.csv(fna.data, row.names=1)
```

Lets create a new data.frame that omits this first column because we don't want to include this data in our analysis

```
# We can use -1 here to remove the first column
wisc.data <- wisc.df[,-1]
```

Finally, setup a separate new vector called diagnosis that contains the data from the diagnosis column of the original dataset. We will store this as a factor (useful for plotting) and use this later to check our results.

```
# Create diagnosis vector for later
diagnosis <- as.factor(wisc.df[,1])
```

> Q1. How many observations are in this dataset?

```
nrow(wisc.data)
```

```
[1] 569
```

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

There are 212 malignant diagnosis

```
table(diagnosis)
```

```
diagnosis
  B   M 
357 212 
```

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

There are 10 variables/feature in the data that are suffixed with _mean

```
colnames(wisc.data)
```

```
 [1] "radius_mean"             "texture_mean"           
 [3] "perimeter_mean"          "area_mean"              
 [5] "smoothness_mean"         "compactness_mean"       
 [7] "concavity_mean"          "concave.points_mean"    
 [9] "symmetry_mean"           "fractal_dimension_mean" 
[11] "radius_se"               "texture_se"             
[13] "perimeter_se"            "area_se"                
[15] "smoothness_se"           "compactness_se"         
[17] "concavity_se"            "concave.points_se"      
[19] "symmetry_se"             "fractal_dimension_se"   
[21] "radius_worst"            "texture_worst"          
[23] "perimeter_worst"         "area_worst"             
[25] "smoothness_worst"        "compactness_worst"      
[27] "concavity_worst"         "concave.points_worst"   
[29] "symmetry_worst"          "fractal_dimension_worst"
```

The function `grep()` could be useful here. How does it work?

```
matches <- grep("_mean",colnames(wisc.data))
length(matches)
```

```
[1] 10
```

2. Principal Component Analysis

Performing PCA

Check the mean and standard deviation of the features.

```
# 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
```

Execute PCA with the prcomp() function on the wisc.data, scaling if appropriate, and assign the output model to wisc.pr.

```r
# Perform PCA on wisc.data by completing the following code
wisc.pr <- prcomp(wisc.data, scale. = TRUE)
```

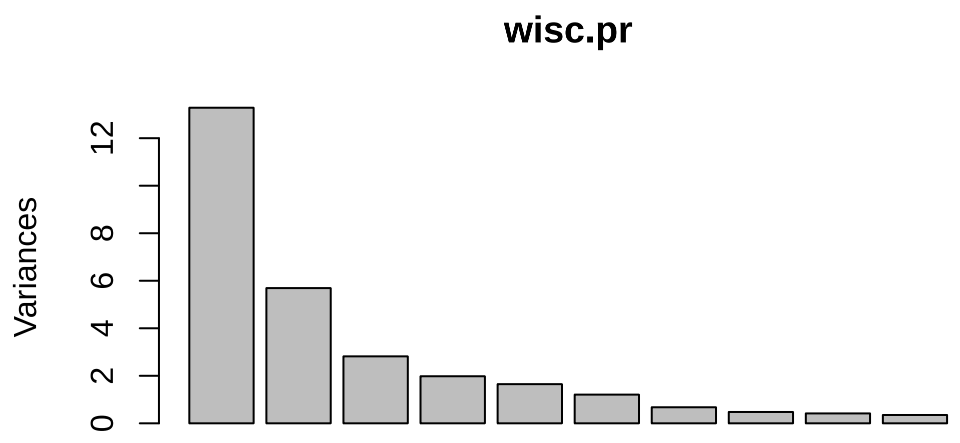
Find the summary of the results

```r
# Look at summary of 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
```

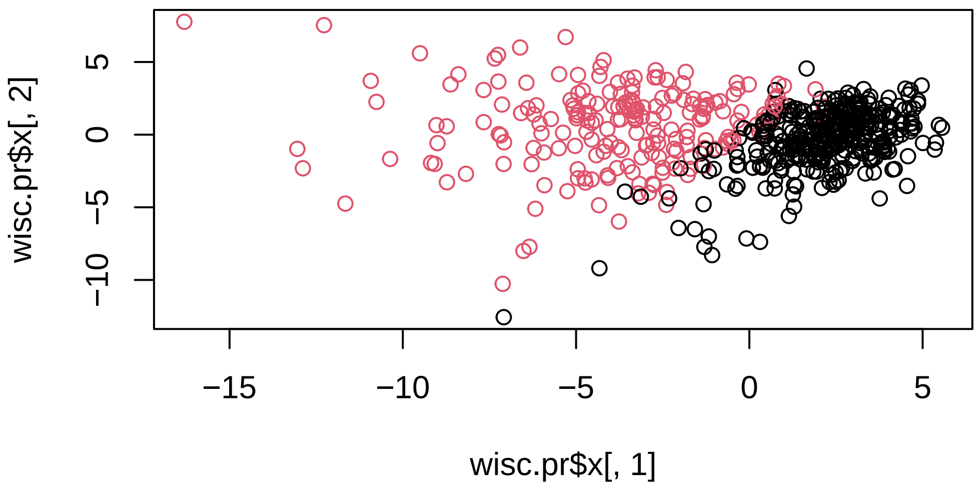
```
Proportion of Variance 0.00002 0.00000
Cumulative Proportion  1.00000 1.00000
```

```
plot(wisc.pr)
```



Let's make a PC plot (a.k.a "score plot" or "PC1 vs PC2" etc plot)

```
plot(wisc.pr$x[,1], wisc.pr$x[,2], col = diagnosis)
```



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

44.27% of the original variance is captured by the first principle component

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

Three principal component 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?

Seven principal component are required to describe at least 90& of the original variance in the data.

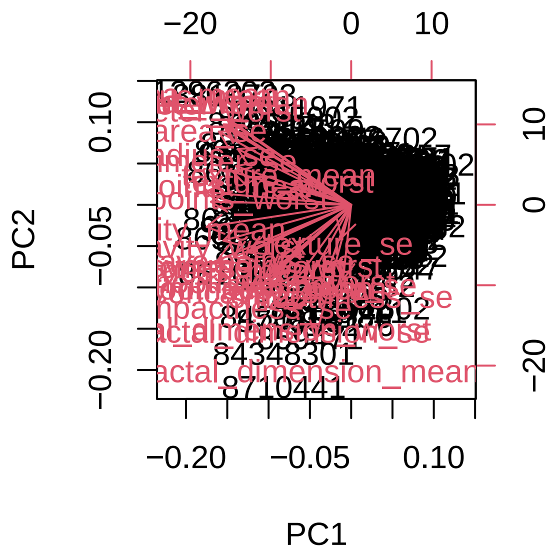
Interpreting PCA results

Create a biplot of the wisc.pr using the biplot() function.

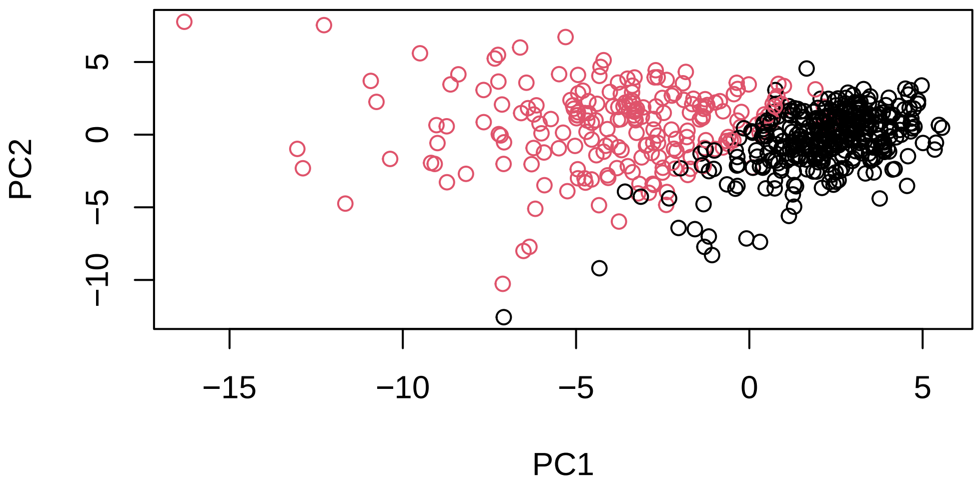
> Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?

What stands out to me is the big chunk of data that is very difficult to understand. All of the data points are overlapping each other.

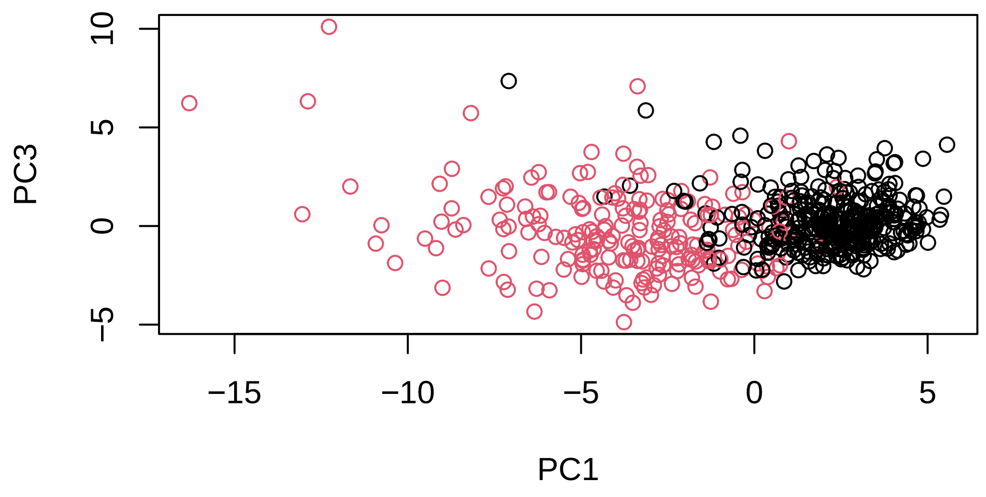
```r
biplot(wisc.pr)
```



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



```
# Repeat for components 1 and 3
plot(wisc.pr$x[, 1], wisc.pr$x[, 3], col = diagnosis,
     xlab = "PC1", ylab = "PC3")
```



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

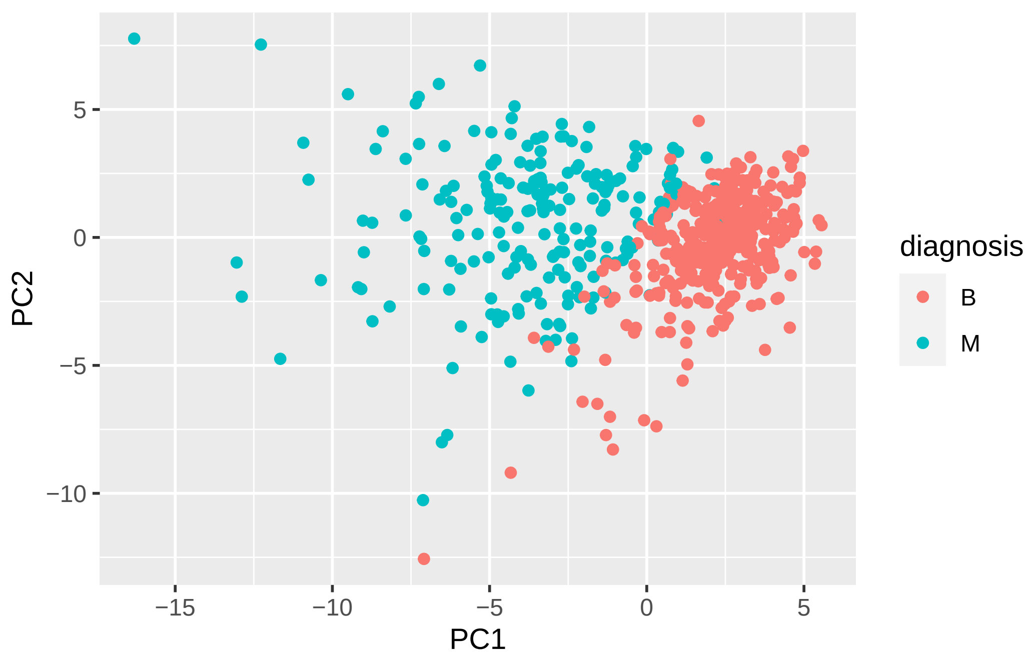
Plot for PC1 vs PC2 seems cleaner compared to plot for PC1 vs PC3. There are more overlapping in PC1 vs PC3 plot and in plot for PC1 vs PC2, the data are more spread out.

Let's use ggplot to create better plot!

```
# 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

Calculate the variance of each principal component by squaring the sdev component of wisc.pr (i.e. wisc.pr$sdev^2). Save the result as an object called pr.var.

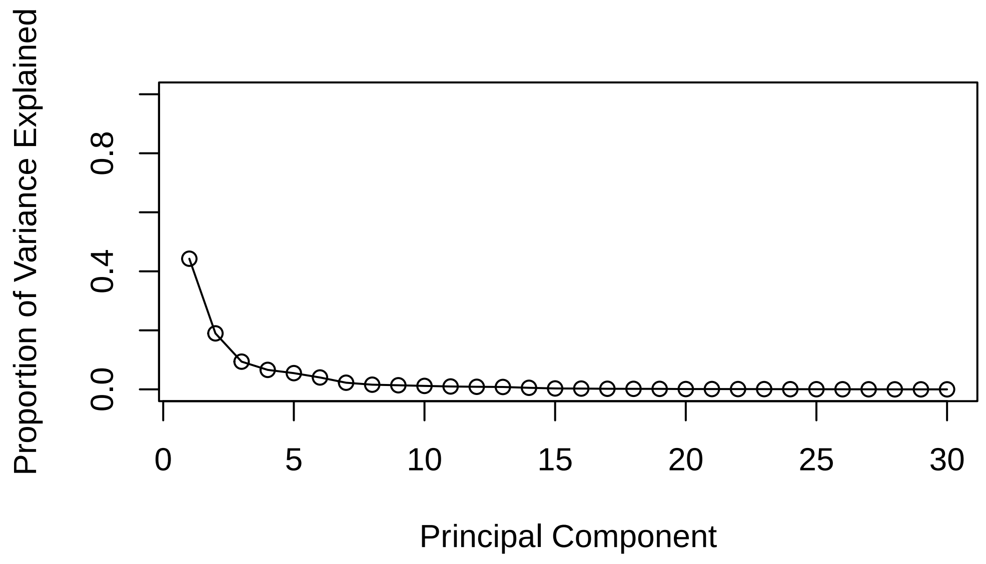
```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

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

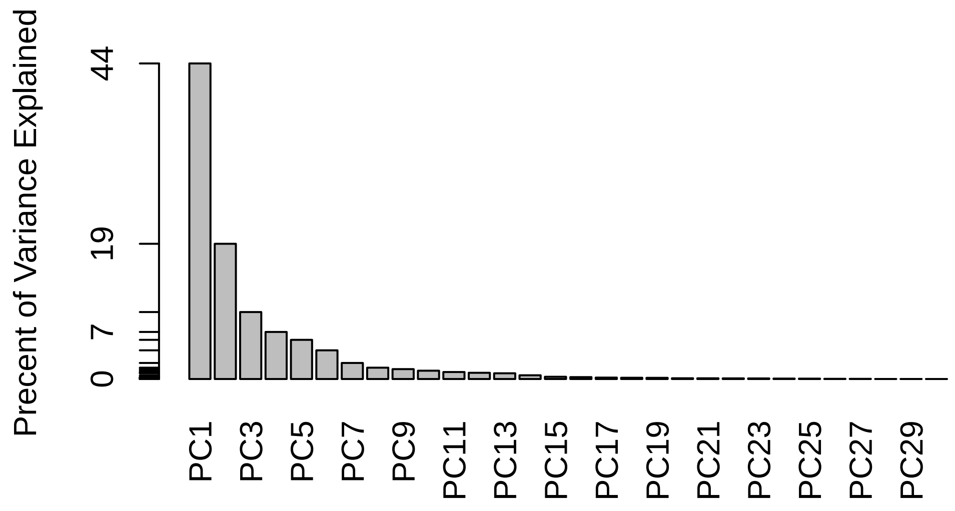
Calculate the variance explained by each principal component by dividing by the total variance explained of all principal components

```
# 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

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?

-0.26085376 is the component of the loading vector for the feature concave.points_mean

```
wisc.pr$rotation[,1]
```

```
            radius_mean            texture_mean          perimeter_mean
            -0.21890244             -0.10372458             -0.22753729
              area_mean         smoothness_mean        compactness_mean
            -0.22099499             -0.14258969             -0.23928535
         concavity_mean     concave.points_mean           symmetry_mean
            -0.25840048             -0.26085376             -0.13816696
 fractal_dimension_mean               radius_se              texture_se
            -0.06436335             -0.20597878             -0.01742803
           perimeter_se                 area_se           smoothness_se
            -0.21132592             -0.20286964             -0.01453145
         compactness_se            concavity_se       concave.points_se
            -0.17039345             -0.15358979             -0.18341740
            symmetry_se    fractal_dimension_se            radius_worst
            -0.04249842             -0.10256832             -0.22799663
          texture_worst         perimeter_worst              area_worst
            -0.10446933             -0.23663968             -0.22487053
       smoothness_worst       compactness_worst         concavity_worst
            -0.12795256             -0.21009588             -0.22876753
   concave.points_worst          symmetry_worst fractal_dimension_worst
            -0.25088597             -0.12290456             -0.13178394
```

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

5 is the minimum number of principal components required to explain 80% of the variance of the data

```r
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
```

3. Hierarchical clustering

First scale the wisc.data data and assign the result to data.scaled

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

Find the (Euclidean) distances between all pairs of observations

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

Create a hierarchical clustering model using complete linkage.

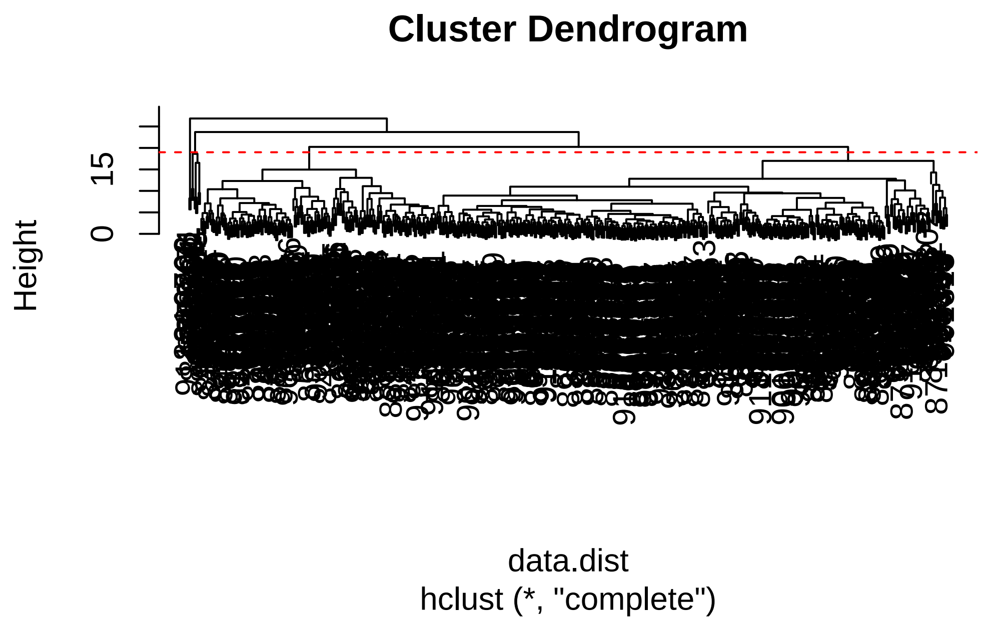
```r
wisc.hclust <- hclust(data.dist, method = "complete")
```

Results of hierarchical clustering

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

At the height of 19, there are 4 clusters

```r
plot(wisc.hclust)
abline(h=19, col="red", lty=2)
```



Selecting number of clusters

Use cutree() to cut the tree so that it has 4 clusters. Assign the output to the variable wisc.hclust.clusters.

```r
wisc.hclust.clusters <- cutree(wisc.hclust, k = 4)
```

We can use the table() function to compare the cluster membership to the actual diagnoses.

```r
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?

There is no better way to cut the tree than cutting it into 4 clusters because I tried other numbers and they all seem to be overlapping the two diagnoses.

```r
wisc.hclust.clusters <- cutree(wisc.hclust, k = 2)
```

Using different methods

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

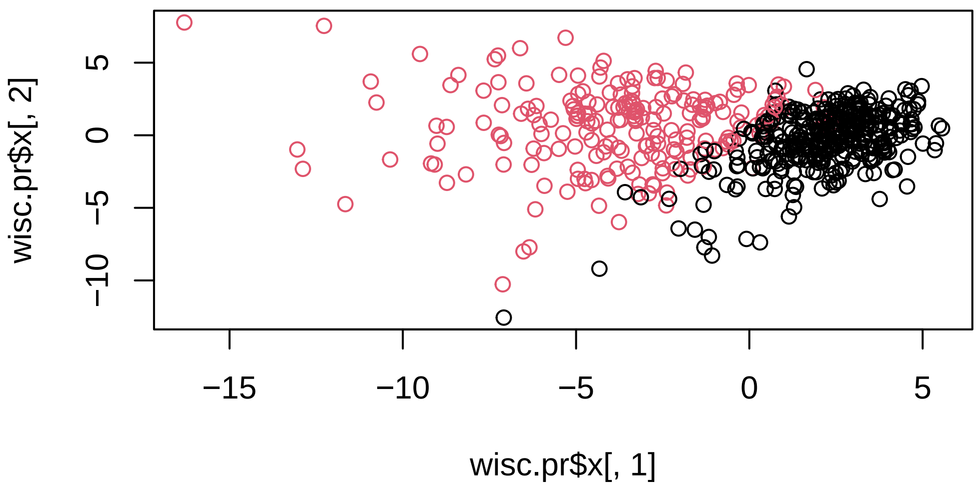
My favorite method is "ward.D2" because it creates groups with minimized variance between clusters

5. Combining methods

Clustering on PCA results

I want to cluster in "PC space".

```
plot(wisc.pr$x[,1], wisc.pr$x[,2], col = diagnosis)
```



The `hclust()` function wants a distance matrix as input…

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

Find my cluster membership vector with the `cutree()` function.

```
grps <- cutree(wisc.pr.hclust, k=2)
table(grps)
```

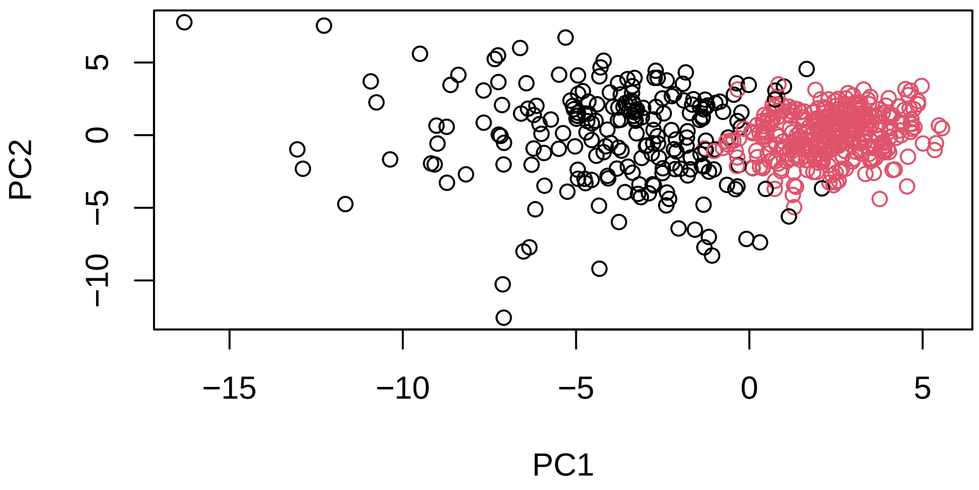
```
grps
  1   2
216 353
```

```
table(grps, diagnosis)
```

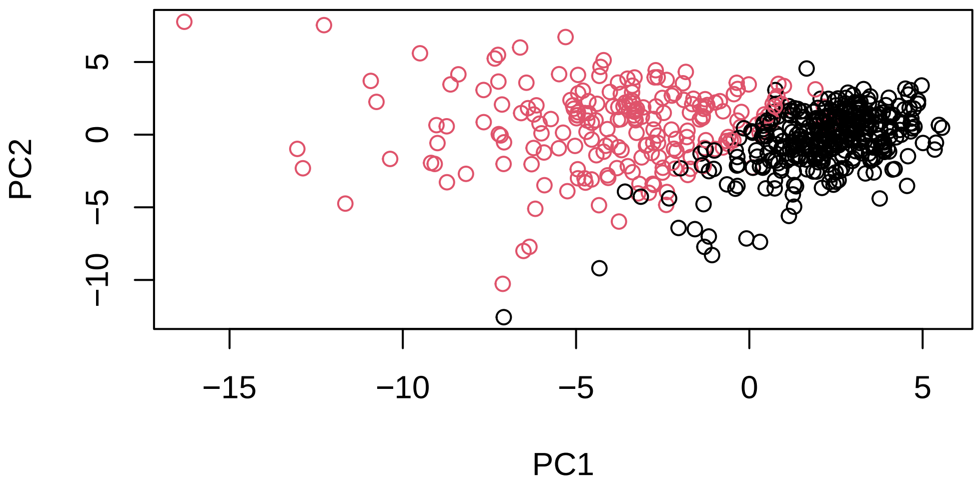
```
    diagnosis
```

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

```
plot(wisc.pr$x[,1:2], col=grps)
```



```
plot(wisc.pr$x[,1:2], col=diagnosis)
```



```r
## Use the distance along the first 7 PCs for clustering i.e. wisc.pr$x[, 1:7]

wisc.pr.hclust <- hclust(d, method="ward.D2")
```

Cut this hierarchical clustering model into 2 clusters and assign the results to wisc.pr.hclust.clusters.

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

Using table(), compare the results from your new hierarchical clustering model with the actual diagnoses.

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

The newly created model with two clusters separate pretty well because all of the benign separated into one group and malignant separated into a different group.

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

```
                       diagnosis
wisc.pr.hclust.clusters   B   M
```

```
                      1  28 188
                      2 329  24
```

> 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.

I think the "wisc.pr.hclust.clusters" are better at separating because it show more distinctness between the two groups.

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

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

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

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

6. Sensitivity/Specificity

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

`wisc.hclust.clusters` provide the best specificity

`wisc.hclust.clusters` provide the best sensitivity

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

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

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

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

Sensitivity: TP/(TP+FN) For `wisc.pr.hclust.clusters`

```r
188/(188+28)
```

```
[1] 0.8703704
```

For `wisc.hclust.clusters`

```r
165/(165+12)
```

```
[1] 0.9322034
```

Specificity: TN/(TN+FN) For `wisc.pr.hclust.clusters`

```r
329/(329+28)
```

```
[1] 0.9215686
```

For `wisc.hclust.clusters`

```r
343/(343+12)
```

```
[1] 0.9661972
```