Class-8-mini-project

Alvin Cheng A16840171

##Outline Today we will apply the machine learning methods we introduced in the last class on breast cancer biopsy data from fine needle aspiration (FNA).

##Data Input The data is supplied on CSV format.

For this we can use the read.csv() function to read the CSV (comma-separated values) file containing the data (available from our class website: WisconsinCancer.csv)

Assign the result to an object called wisc.df.

```
# Save your input data file into your Project directory
wisc.df <- read.csv("WisconsinCancer.csv", row.names = 1)
#fna.data <- "WisconsinCancer.csv"
#View(wisc.df)
head(wisc.df)
```

```
         diagnosis radius_mean texture_mean perimeter_mean area_mean
842302           M       17.99        10.38         122.80    1001.0
842517           M       20.57        17.77         132.90    1326.0
84300903         M       19.69        21.25         130.00    1203.0
84348301         M       11.42        20.38          77.58     386.1
84358402         M       20.29        14.34         135.10    1297.0
843786           M       12.45        15.70          82.57     477.1
         smoothness_mean compactness_mean concavity_mean concave.points_mean
842302           0.11840          0.27760         0.3001             0.14710
842517           0.08474          0.07864         0.0869             0.07017
84300903         0.10960          0.15990         0.1974             0.12790
84348301         0.14250          0.28390         0.2414             0.10520
84358402         0.10030          0.13280         0.1980             0.10430
843786           0.12780          0.17000         0.1578             0.08089
         symmetry_mean fractal_dimension_mean radius_se texture_se perimeter_se
842302          0.2419                0.07871    1.0950     0.9053        8.589
842517          0.1812                0.05667    0.5435     0.7339        3.398
```

```
84300903        0.2069                0.05999    0.7456     0.7869        4.585
84348301        0.2597                0.09744    0.4956     1.1560        3.445
84358402        0.1809                0.05883    0.7572     0.7813        5.438
843786          0.2087                0.07613    0.3345     0.8902        2.217
         area_se smoothness_se compactness_se concavity_se concave.points_se
842302    153.40      0.006399        0.04904      0.05373           0.01587
842517     74.08      0.005225        0.01308      0.01860           0.01340
84300903   94.03      0.006150        0.04006      0.03832           0.02058
84348301   27.23      0.009110        0.07458      0.05661           0.01867
84358402   94.44      0.011490        0.02461      0.05688           0.01885
843786     27.19      0.007510        0.03345      0.03672           0.01137
         symmetry_se fractal_dimension_se radius_worst texture_worst
842302       0.03003             0.006193        25.38         17.33
842517       0.01389             0.003532        24.99         23.41
84300903     0.02250             0.004571        23.57         25.53
84348301     0.05963             0.009208        14.91         26.50
84358402     0.01756             0.005115        22.54         16.67
843786       0.02165             0.005082        15.47         23.75
         perimeter_worst area_worst smoothness_worst compactness_worst
842302            184.60     2019.0           0.1622            0.6656
842517            158.80     1956.0           0.1238            0.1866
84300903          152.50     1709.0           0.1444            0.4245
84348301           98.87      567.7           0.2098            0.8663
84358402          152.20     1575.0           0.1374            0.2050
843786            103.40      741.6           0.1791            0.5249
         concavity_worst concave.points_worst symmetry_worst
842302            0.7119               0.2654         0.4601
842517            0.2416               0.1860         0.2750
84300903          0.4504               0.2430         0.3613
84348301          0.6869               0.2575         0.6638
84358402          0.4000               0.1625         0.2364
843786            0.5355               0.1741         0.3985
         fractal_dimension_worst
842302                   0.11890
842517                   0.08902
84300903                 0.08758
84348301                 0.17300
84358402                 0.07678
843786                   0.12440
```

Creating a new data frame to omit the ID

```
# We can use -1 here to remove the first column
wisc.data <- wisc.df[,-1] # this removes ID to prevent bias in data.
head(wisc.data)
```

```
        radius_mean texture_mean perimeter_mean area_mean smoothness_mean
842302        17.99        10.38         122.80    1001.0         0.11840
842517        20.57        17.77         132.90    1326.0         0.08474
84300903      19.69        21.25         130.00    1203.0         0.10960
84348301      11.42        20.38          77.58     386.1         0.14250
84358402      20.29        14.34         135.10    1297.0         0.10030
843786        12.45        15.70          82.57     477.1         0.12780
        compactness_mean concavity_mean concave.points_mean symmetry_mean
842302           0.27760         0.3001             0.14710        0.2419
842517           0.07864         0.0869             0.07017        0.1812
84300903         0.15990         0.1974             0.12790        0.2069
84348301         0.28390         0.2414             0.10520        0.2597
84358402         0.13280         0.1980             0.10430        0.1809
843786           0.17000         0.1578             0.08089        0.2087
        fractal_dimension_mean radius_se texture_se perimeter_se area_se
842302                 0.07871    1.0950     0.9053        8.589  153.40
842517                 0.05667    0.5435     0.7339        3.398   74.08
84300903               0.05999    0.7456     0.7869        4.585   94.03
84348301               0.09744    0.4956     1.1560        3.445   27.23
84358402               0.05883    0.7572     0.7813        5.438   94.44
843786                 0.07613    0.3345     0.8902        2.217   27.19
        smoothness_se compactness_se concavity_se concave.points_se
842302       0.006399        0.04904      0.05373           0.01587
842517       0.005225        0.01308      0.01860           0.01340
84300903     0.006150        0.04006      0.03832           0.02058
84348301     0.009110        0.07458      0.05661           0.01867
84358402     0.011490        0.02461      0.05688           0.01885
843786       0.007510        0.03345      0.03672           0.01137
        symmetry_se fractal_dimension_se radius_worst texture_worst
842302      0.03003             0.006193        25.38         17.33
842517      0.01389             0.003532        24.99         23.41
84300903    0.02250             0.004571        23.57         25.53
84348301    0.05963             0.009208        14.91         26.50
84358402    0.01756             0.005115        22.54         16.67
843786      0.02165             0.005082        15.47         23.75
        perimeter_worst area_worst smoothness_worst compactness_worst
842302           184.60     2019.0           0.1622            0.6656
842517           158.80     1956.0           0.1238            0.1866
```

```
84300903          152.50     1709.0           0.1444            0.4245
84348301           98.87      567.7           0.2098            0.8663
84358402          152.20     1575.0           0.1374            0.2050
843786            103.40      741.6           0.1791            0.5249
         concavity_worst concave.points_worst symmetry_worst
842302            0.7119               0.2654         0.4601
842517            0.2416               0.1860         0.2750
84300903          0.4504               0.2430         0.3613
84348301          0.6869               0.2575         0.6638
84358402          0.4000               0.1625         0.2364
843786            0.5355               0.1741         0.3985
         fractal_dimension_worst
842302                   0.11890
842517                   0.08902
84300903                 0.08758
84348301                 0.17300
84358402                 0.07678
843786                   0.12440
```

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.

```
diagnosis <- as.factor(wisc.df$diagnosis)
diagnosis
```

```
  [1] M M M M M M M M M M M M M M M M M B B B M M M M M M M M M M M M M M
 [38] B M M M M M M M M B M B B B B B M M B M M B B B B M B M M B B B M B M M
 [75] B M B M M B B B M M B M M M B B B M B B M M B B B M M B B B M B B M B B
[112] B B B B B B M M M B M M B B B M M B M B M M B M M B B M B B B M B B B B M B
[149] B B B B B B B B M B B B B M M B M B B M M B B M M B B B B M B B M M M B M
[186] B M B B B M B B M M B M M M M B M M M B M B M B B M B M M M M B B M M B B
[223] B M B B B B B M M B B M B B M M B M B B B B M B B B B B M B M M M M M M M
[260] M M M M M M M B B B B B B M B M B B M B B M B M M B B B B B B B B B B B B
[297] B M B B M B M B B B B B B B B B B B B B B M B B B M B M B B B B M M M B B
[334] B B M B M B M B B B M B B B B B B B M M M B B B B B B B B B B B M M B M M
[371] M B M M B B B B B M B B B B B M B B B M B B M M B B B B B B M B B B B B B
[408] B M B B B B B M B B M B B B B B B B B B B B B M B M M B M B B B B B M B B
[445] M B M B B M B M B B B B B B B B M M B B B B B B M B B B B B B B B B M B
[482] B B B B B B M B M B B M B B B B B B M M B M B M B B B B B M B B M B M B M M
[519] B B B M B B B B B B B B B B B B M B M M B B B B B B B B B B B B B B B B B
[556] B B B B B B B M M M M M B
```

```
Levels: B M
```

Note that the first column here wisc.df$diagnosis is a pathologist provided expert diagnosis. Malignant vs. Benign

Q1. How many observations are in this dataset?

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

```
[1] 569
```

569 patients

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

```
table(wisc.df$diagnosis) # OR sum(wisc.data == "M")
```

```
  B   M 
357 212 
```

212 malignant

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

```
# Use grep to find column names containing "_mean"
mean_column_names <- grep("_mean", names(wisc.data), value = TRUE)

# Count the number of columns with "_mean"
count_mean_columns <- length(mean_column_names)

# Print the count of columns with "_mean"
count_mean_columns
```

```
[1] 10
```

Principal Component Analysis

Check the mean and standard deviation of the features (i.e. columns) of the wisc.data to determine if the data should be scaled.

```
# 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
```

`scale = TRUE` is a common practice in PCA. This is used to find patterns in data adjusted to a scale for better comparison without a larger number dominating the data

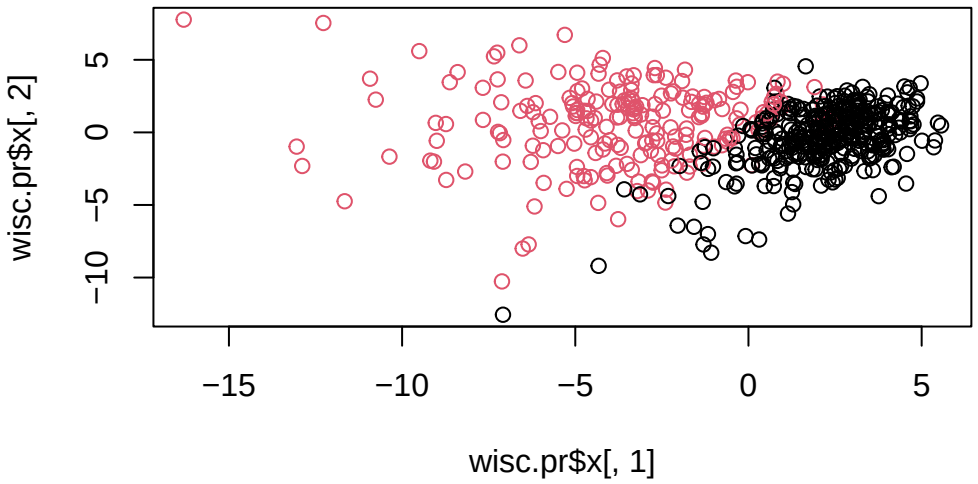
```
# Perform PCA on wisc.data by completing the following code
wisc.pr <- prcomp(wisc.data, scale = TRUE)

#looking at a brief 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
```

```
#wisc.pr
```

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



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

```
standard_deviations <- wisc.pr$sdev #extracting the standard deviations

# Calculate the proportion of variance captured by PC1
proportion_variance_pc1 <- (standard_deviations[1] ^ 2) / sum(standard_deviations ^ 2)
proportion_variance_pc1
```

```
[1] 0.4427203
```

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

```
cumulative_var <- cumsum(wisc.pr$sdev^2/sum(wisc.pr$sdev^2))

num_of_PCs <- which(cumulative_var >= 0.70)[1] # finding the first position in the cumulat

num_of_PCs
```

```
[1] 3
```

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

```
cumulative_var <- cumsum(wisc.pr$sdev^2/sum(wisc.pr$sdev^2))

num_of_PCs <- which(cumulative_var >= 0.90)[1] # finding the first position in the cumulat

num_of_PCs
```

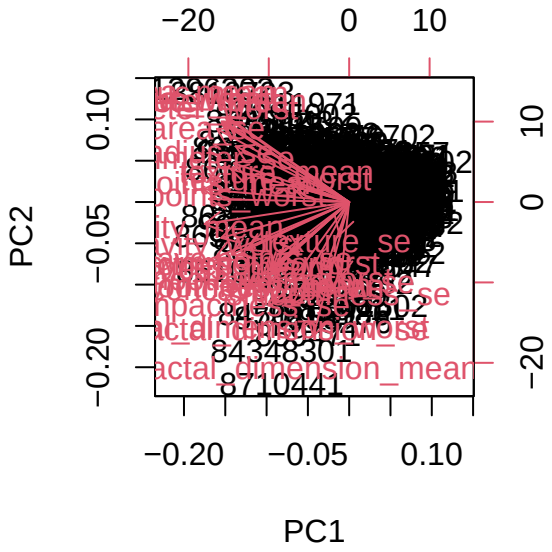
```
[1] 7
```

However, you will often run into some common challenges with using biplots on real-world data containing a non-trivial number of observations and variables. Here we will need to look at some alternative visualizations.

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

The plot is extremely messy and a lot of data points are congregated together. This is hard to understand so we may need a better way of plotting.

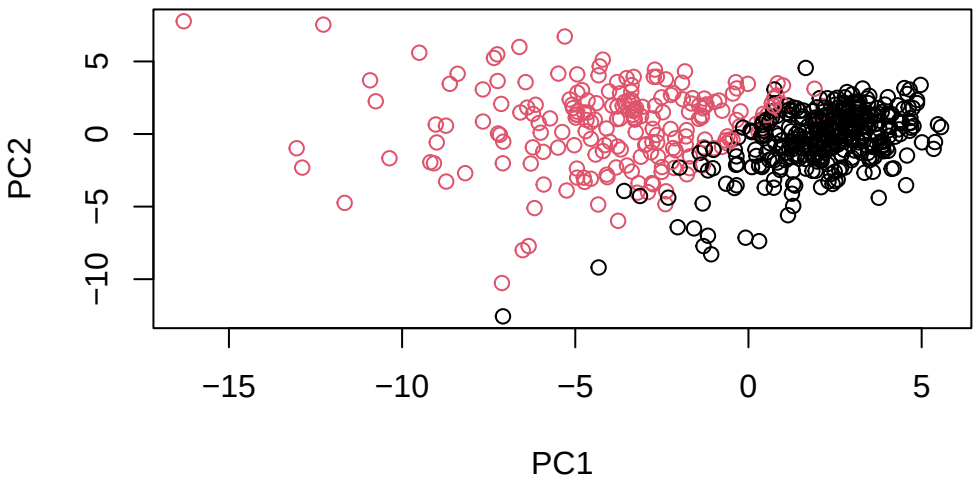
```
biplot(wisc.pr)
```



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

There is a cleaner separation of data despite the axis numbers are the same. The separation of benign and malignant data is better.

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

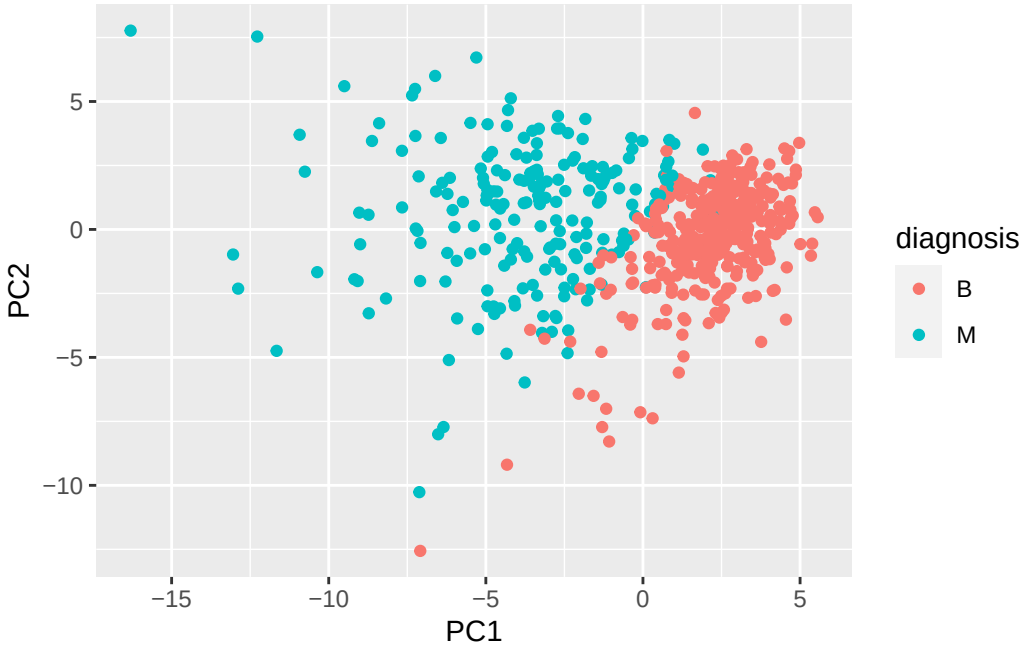


We can make a more fancier figure of the graph above using ggplot

```
# 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

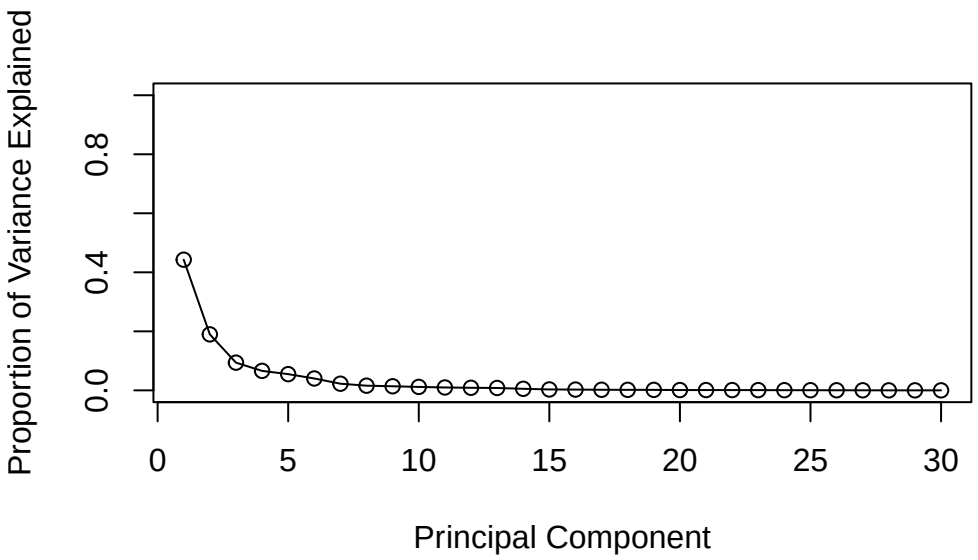
Calculate the variance of each principal component by squaring the sdev component of wisc.pr (i.e. wisc.pr$sdev^2). Save the result as an object called pr.var.

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

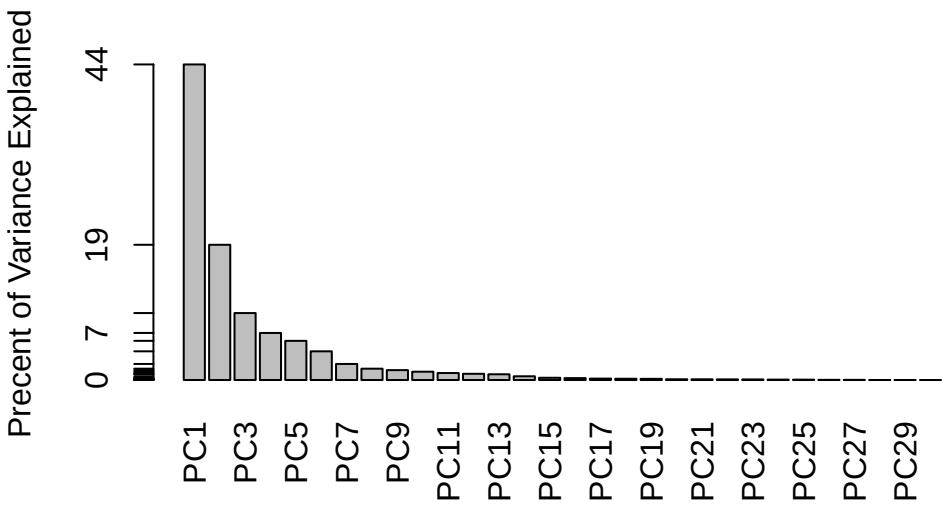
```
[1] 13.281608  5.691355  2.817949  1.980640  1.648731  1.207357
```

```
# 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 )
```



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

```
[1] -0.2608538
```

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

```
cumulative_var_1 <- cumsum(wisc.pr$sdev^2/sum(wisc.pr$sdev^2))

num_of_PCs_1 <- which(cumulative_var_1 >= 0.80)[1] # finding the first position in the cum

num_of_PCs_1
```

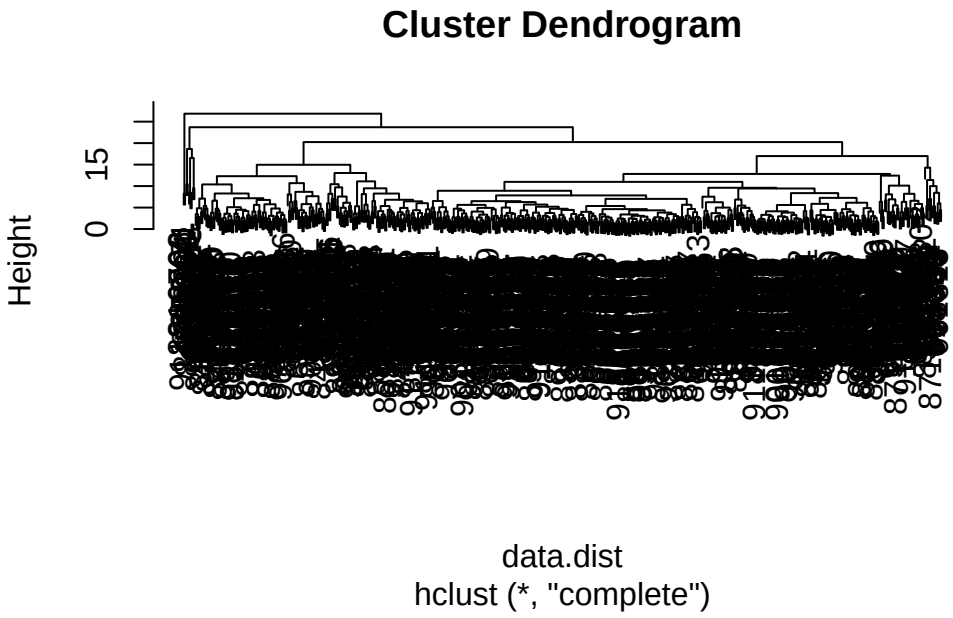
```
[1] 5
```

Hierarchical Clustering

Can we just use clustering on the original data and get some insights into M vs. B?

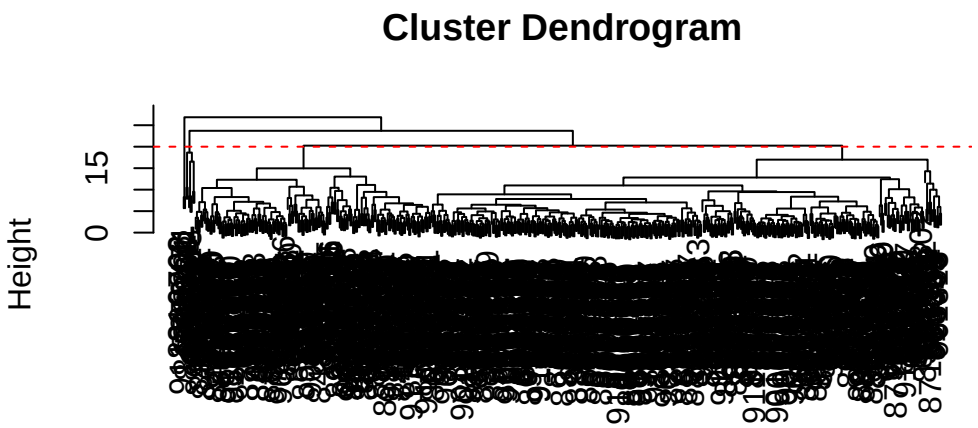
It is rather difficult, this "tree looks like a hot mess...

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



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

```
plot(wisc.hclust)  # Plot the dendrogram
height_11 <- abline(h = 20, col = "red", lty = 2) # continually to change h to see where t
```



at height 19 (but also at 20), the clustering model has 4 clusters based on the abline drawn and what was seen visually

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

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

Yes, when I use 2 cluster, most of the data fits in the first cluster.

```
wisc.hclust.clusters <- cutree(wisc.hclust, k = 2) # change k to something between 2 and 1
#wisc.hclust.clusters
table(wisc.hclust.clusters, diagnosis)
```

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

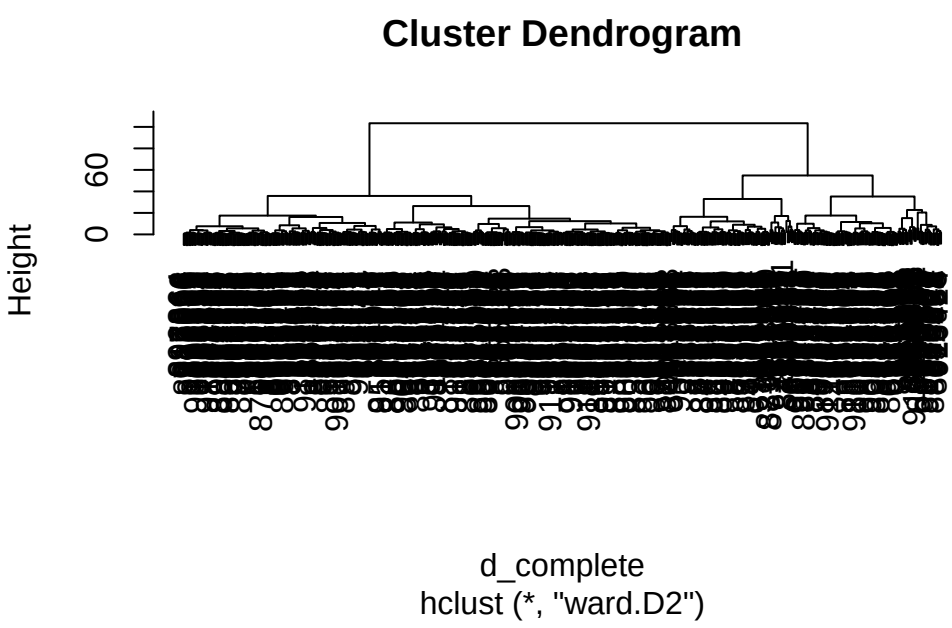
As we discussed in our last class videos there are number of different "methods" we can use to combine points during the hierarchical clustering procedure. These include "single", "complete", "average" and (my favorite) "ward.D2".

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

I like `ward.D2` the best because it divides the data well in which it minimizes the variance in each cluster. Single has too much data at each cluster, complete and average have some stems that looks like it is more showing of the outliers

```
d_complete <- dist( wisc.pr$x[,1:3])
wisc.pr.hclust_complete <- hclust(d_complete,method = "ward.D2") #insert single, complete,

plot(wisc.pr.hclust_complete)
```



5. Combining Methods

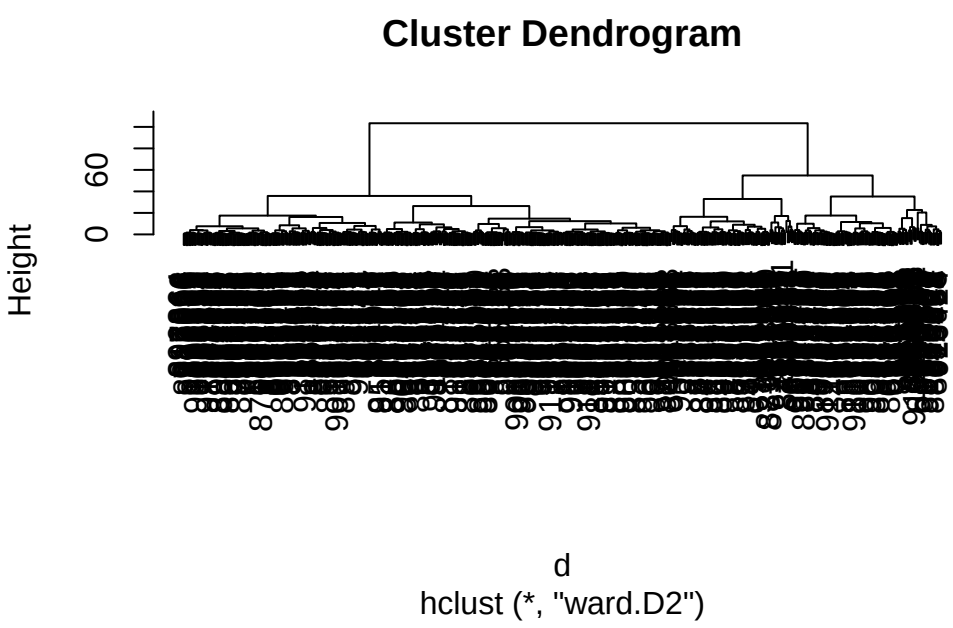
In this final section, you will put together several steps you used earlier and, in doing so, you will experience some of the creativity and open endedness that is typical in unsupervised learning.

Recall from earlier sections that the PCA model required significantly fewer features to describe 70%, 80% and 95% of the variability of the data. In addition to normalizing data and potentially avoiding over-fitting, PCA also uncorrelates the variables, sometimes improving the performance of other modeling techniques.

Let's see if PCA improves or degrades the performance of hierarchical clustering.

This approach will take not original data but our PCA results and work with them.

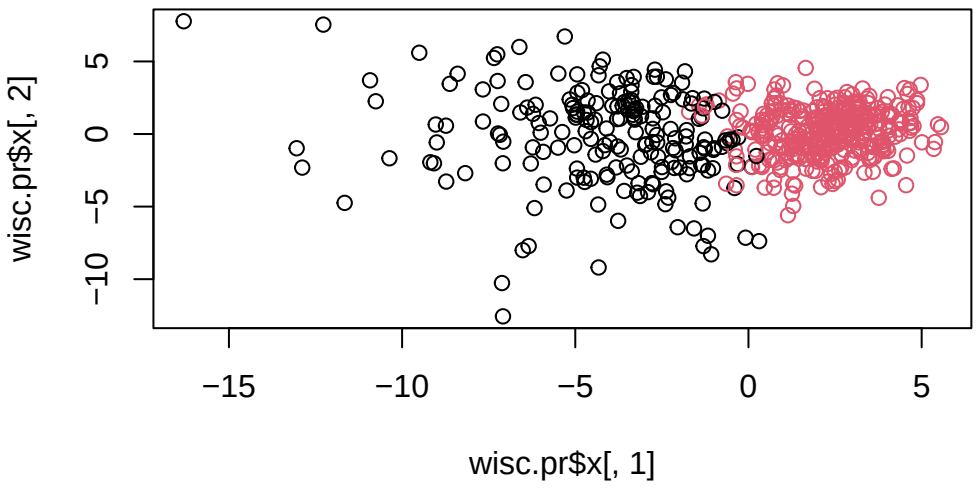
```
d <- dist( wisc.pr$x[,1:3])
wisc.pr.hclust <- hclust(d,method = "ward.D2")
plot(wisc.pr.hclust)
```



Generate 2 cluster groups from this hclust object

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

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




```
table(grps)
```

```
grps
  1   2
203 366
```

```
table(diagnosis)
```

```
diagnosis
  B   M
357 212
```

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

```
         grps
diagnosis   1   2
        B  24 333
        M 179  33
```

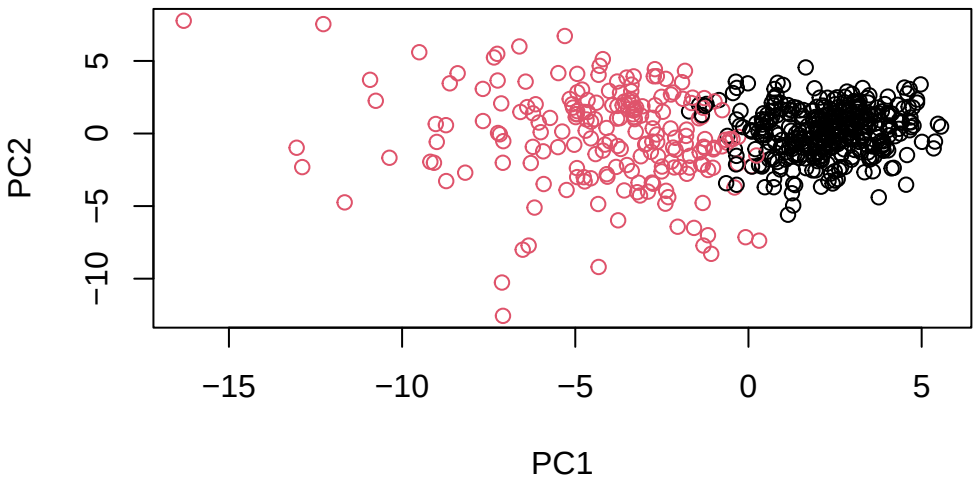
```
g <- as.factor(grps)
levels(g)
```

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

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

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

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



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

The newly created model separates the two diagnoses better with 4 different clusters in than in the table with 2 clusters in which most of the benign and malignant were grouped together. In this model, most of the benign and malignant is separated though there are some benign grouped with maligant and some malignant grouped with benign, presenting the risk of false positives. Nonetheless, this model is still better.

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

# Perform hierarchical clustering
wisc.pr.hclust <- hclust(distance, method = "ward.D2")

wisc.pr.hclust.clusters <- cutree(wisc.pr.hclust, k=2) # Cut this hierarchical clustering

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

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