Class 08: Breast Cancer Analysis Project

Rachel Field (PID A69042948)

Table of contents

Background

The goal of this mini-project is to explore a complete analysis using the unsupervised learning techniques covered in the last class. We will extend what we learned by combining PCA as a preprocessing step to clustering using data that consist of measurements of cell nuclei of human breast masses.

The data itself comes from the Wisconsin Breast Cancer Diagnostic Data Set first reported by K. P. Benne and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets".

Values in this data set describe characteristics of the cell nuclei present in digitized images of a fine needle aspiration (FNA) of a breast mass.

Data import

row.names to make the patient identifier number no longer a column

```
wisc.df <- read.csv("WisconsinCancer.csv", row.names=1)
```

Make sure I do not include sample id or diagnosis columns in the data that we analyze below.

-1 removes the first column, diagnosis.

```
diagnosis <- as.factor(wisc.df$diagnosis)
wisc.data <- wisc.df[, -1]
dim(wisc.data)
```

```
[1] 569  30
```

```
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
```

Exploratory Data Analysis

Q1. How many observations are in this dataset?

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

```
[1] 569
```

There are 569 observations in this dataset.

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

```
sum(wisc.df$diagnosis == "M")
```

```
[1] 212
```

```
table(wisc.df$diagnosis)
```

```
  B   M
357 212
```

212 observations have a malignant diagnosis.

Q3. How many variables/features in the data are suffixed with _mean? The functions dim(), nrow(), table(), length() and grep() may be useful for answering the first 3 questions above.

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

```
[1] 10
```

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

```
[1] 10
```

10 variables/feature in the data are suffixed with _mean.

Principal Component Analysis

The main function in base R for PCA is called `prcomp()`. An optional argument `scale` should nearly always be switched to `scale = TRUE` for this function.

prcomp(x, scale=F, center = F)

We want to scale and center our data prior to PCA to ensure that each feature contributes to the analysis, preventing variables with larger values from dominanting the principal components.

```
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
```

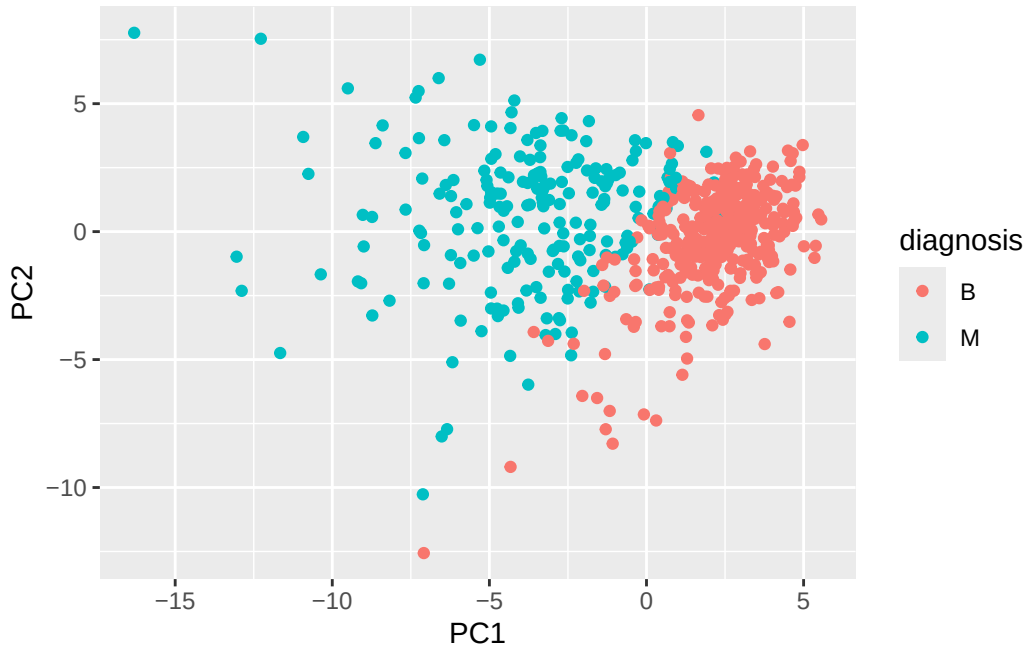
```
wisc.pr <- prcomp(wisc.data, scale=TRUE)
summary(wisc.pr)
```

```
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$x
#ran but not printing in pdf
```

```
library(ggplot2)
ggplot(wisc.pr$x) +
  aes(PC1, PC2, col=diagnosis) +
  geom_point()
```



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 PC1.

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

We need PC1, PC2, and PC3 to get a cumulative proportion of 72.64%, meaning that 72.64% of the variance is described.

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

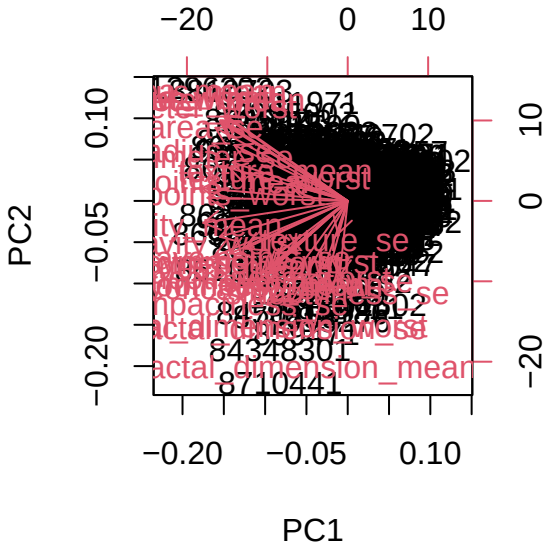
We need PC1-PC7 to describe a cumulative proportion of 91%, meaning that 91% of the variance is described.

Interpreting PCA results

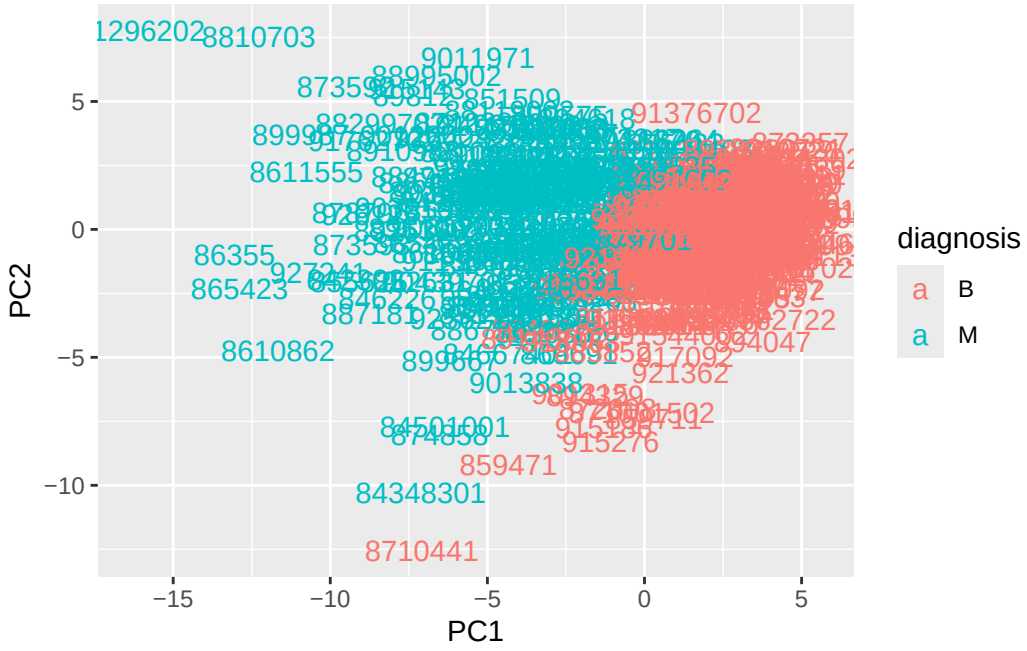
> Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?

It is difficult to understand. Everything is on top of each other because of the text labels.

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

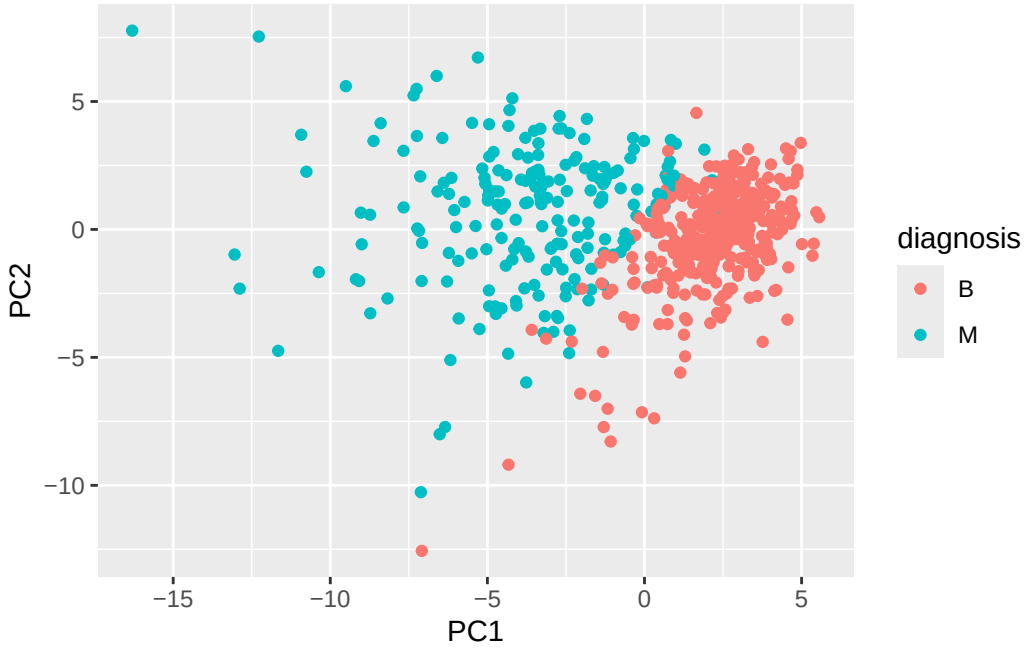


```
ggplot(wisc.pr$x) +
  aes(PC1, PC2, col = diagnosis) +
  geom_text(label = rownames(wisc.pr$x))
```



Lets generate a more standard scatter plot of each observation along principal components 1 and 2 (i.e. a plot of PC1 vs PC2 available as the first two columns of wisc.pr$x) and color the points by the diagnosis (available in the diagnosis vector you created earlier).

```
ggplot(wisc.pr$x) +
  aes(PC1, PC2, col = diagnosis) +
  geom_point()
```

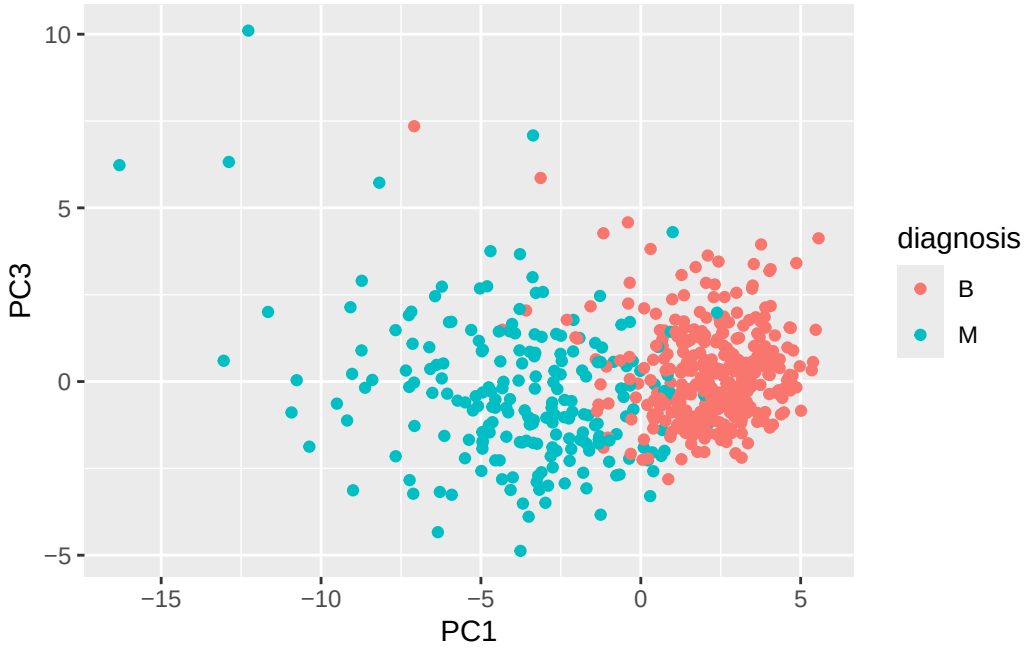


Q8. Generate a similar plot for principal components 1 and 3. What do you notice about these plots?

These plots are much easier to understand because of the color-coding and the fact that the points are dots rather than text.

Also, the PC1 vs PC3 plot has a less clear division of the diagnosis-separated points than the PC1 vs PC2 plot.

```
ggplot(wisc.pr$x) +
  aes(PC1, PC3, col = diagnosis) +
  geom_point()
```



Variance explained

In this exercise, you will produce scree plots showing the proportion of variance explained as the number of principal components increases. The data from PCA must be prepared for these plots, as there is not a built-in function in base R to create them directly from the PCA model.

As you look at these plots, ask yourself if there's an 'elbow' in the amount of variance explained that might lead you to pick a natural number of principal components. If an obvious elbow does not exist, as is typical in some real-world datasets, consider how else you might determine the number of principal components to retain based on the scree plot.

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.

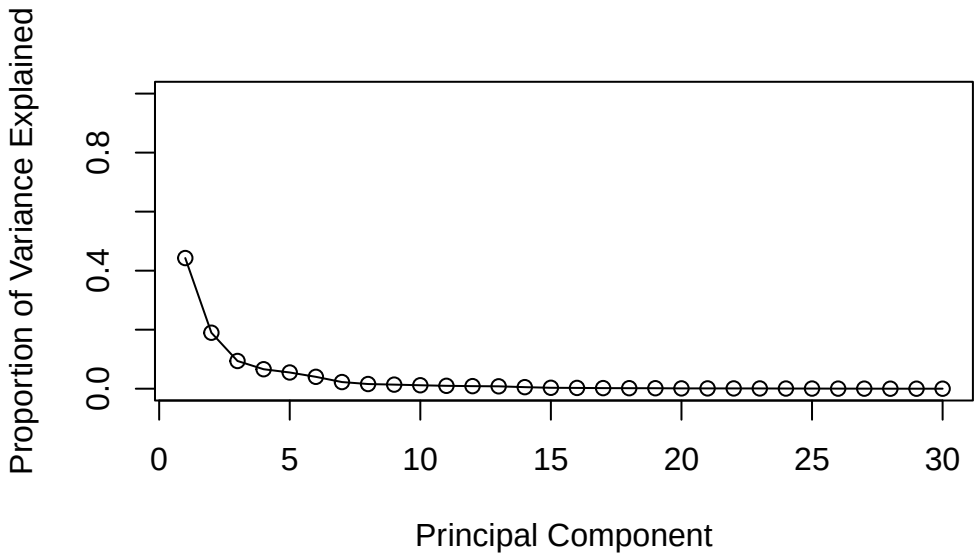
```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

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

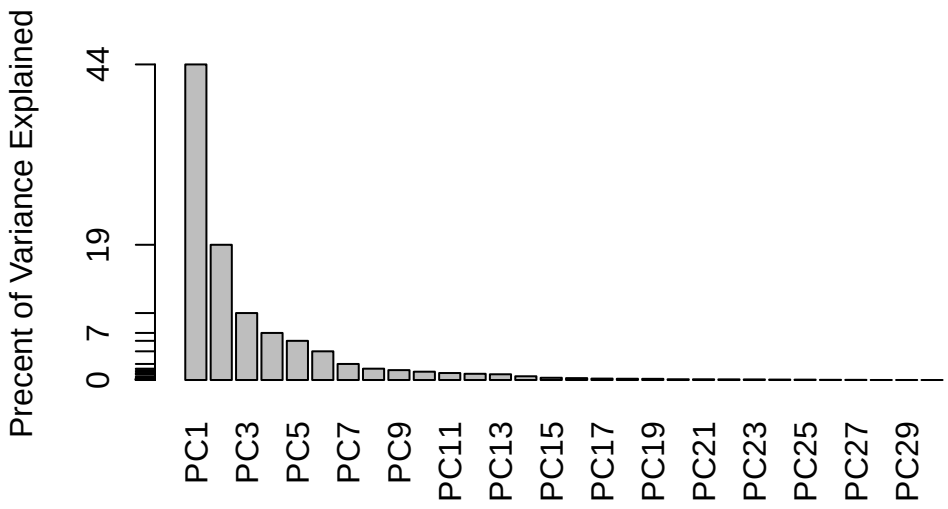
Calculate the variance explained by each principal component by dividing by the total variance explained of all principal components. Assign this to a variable called pve and create a plot of variance explained for each principal component.

```
# 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

In this section we will check your understanding of the PCA results, in particular the loadings and variance explained. The loadings, represented as vectors, explain the mapping from the original features to the principal components. The principal components are naturally ordered from the most variance explained to the least 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? This tells us how much this original feature contributes to the first PC.

The value is -0.26085376.

```
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
```

3. Hierarchical clustering

The goal of this section is to do hierarchical clustering of the original data. Recall from class that this type of clustering does not assume in advance the number of natural groups that exist in the data (unlike K-means clustering).

As part of the preparation for hierarchical clustering, the distance between all pairs of observations are computed. Furthermore, there are different ways to link clusters together, with single, complete, and average being the most common "linkage methods".

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

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

Calculate the (Euclidean) distances between all pairs of observations in the new scaled dataset and assign the result to data.dist.

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

Create a hierarchical clustering model using complete linkage. Manually specify the method argument to hclust() and assign the results to wisc.hclust.

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

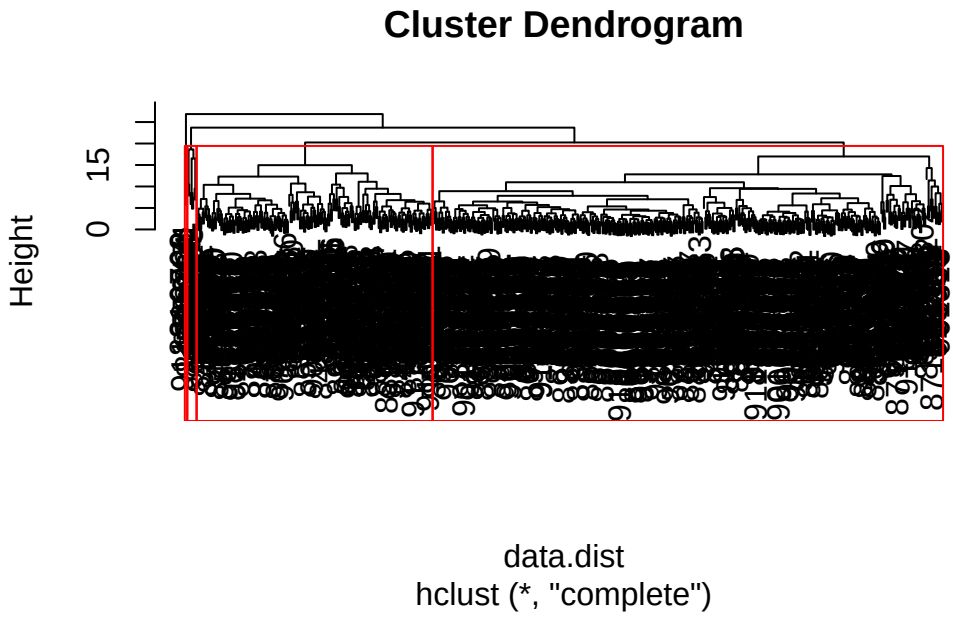
##Results of hierarchical clustering Let's use the hierarchical clustering model you just created to determine a height (or distance between clusters) where a certain number of clusters exists.

Q10. Using the plot() and abline() functions, what is the height at which the clustering model has 4 clusters?

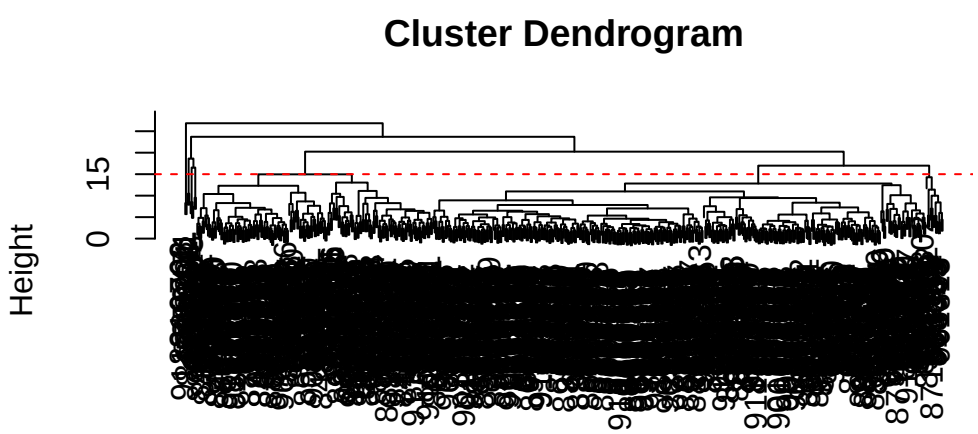
At height 15, the clustering model has 4 clusters.

```
plot(wisc.hclust)

plot(wisc.hclust)
rect.hclust(wisc.hclust, k = 4, border = "red")
```




```
plot(wisc.hclust)
abline(h=15, col="red", lty=2)
```



Selecting number of clusters

In this section, you will compare the outputs from your hierarchical clustering model to the actual diagnoses. Normally when performing unsupervised learning like this, a target variable (i.e. known answer or labels) isn't available. We do have it with this dataset, however, so it can be used to check the performance of the clustering model.

When performing supervised learning - that is, when you're trying to predict some target variable of interest and that target variable is available in the original data - using clustering to create new features may or may not improve the performance of the final model.

This exercise will help you determine if, in this case, hierarchical clustering provides a promising new feature.

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)
```

We can use the 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
```

Here we picked four clusters and see that cluster 1 largely corresponds to malignant cells (with diagnosis values of 1) whilst cluster 3 largely corresponds to benign cells (with diagnosis values of 0).

Before moving on, explore how different numbers of clusters affect the ability of the hierarchical clustering to separate the different diagnoses.

> Q11. OPTIONAL: Can you find a better cluster vs diagnoses match by cutting into a different number of clusters between 2 and 10? How do you judge the quality of your result in each case?

Using different methods

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".

> Q12. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning.

I like ward.D2 because it minimizes the variance within clusters, so I feel like this is the most accurate way to do this to believe that the clusters separate by similarity.

One of the problems with Cluster Analysis is that different methods may produce different results – There is generally no universally accepted "best" method. The good news is that if your data really has clear groups all methods will likely find them and give you similar results. However, in more challenging cases like this one it is best to try multiple algorithms and see what groups logically make sense. A common approach is use a smaller dummy dataset with pre-determined groups that you can use to see which algorithm best recreates what you expect.

Clustering on PCA results

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.

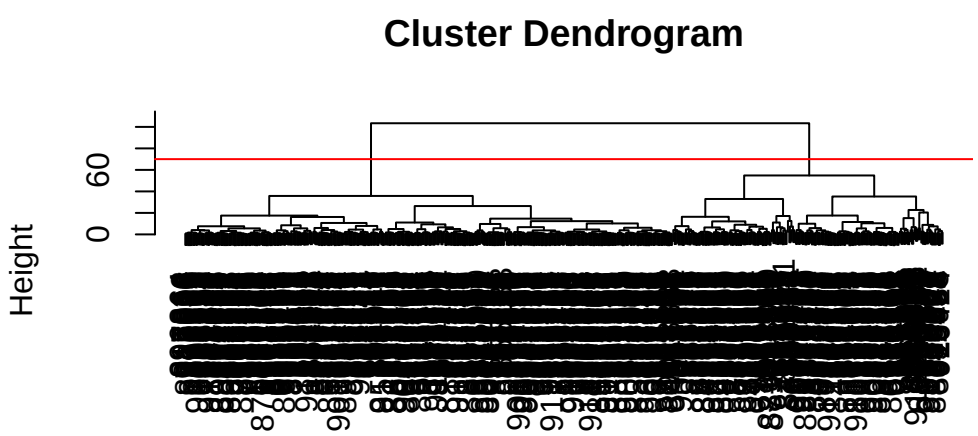
Recall from earlier sections that the PCA model required significantly fewer features to describe 70%, 80% and 95% of the variability of the data. In addition to normalizing data and potentially avoiding over-fitting, PCA also uncorrelates the variables, sometimes improving the performance of other modeling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

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

Using the minimum number of principal components required to describe at least 90% of the variability in the data, create a hierarchical clustering model with the linkage method="ward.D2". We use Ward's criterion here because it is based on multidimensional variance like principal components analysis. Assign the results to wisc.pr.hclust.

Combining PCA and CLustering

```
d <- dist(wisc.pr$x[,1:3])
wisc.pr.hclust <- hclust(d, method = "ward.D2")
plot(wisc.pr.hclust)
abline(h=70, col = "red")
```



Get my cluster membership vector

```
grps <- cutree(wisc.pr.hclust, h=70)
table(grps)
```

```
grps
  1   2
203 366
```

```
#how many patients in each cluster
```

```
table(diagnosis)
```

```
diagnosis
  B   M
357 212
```

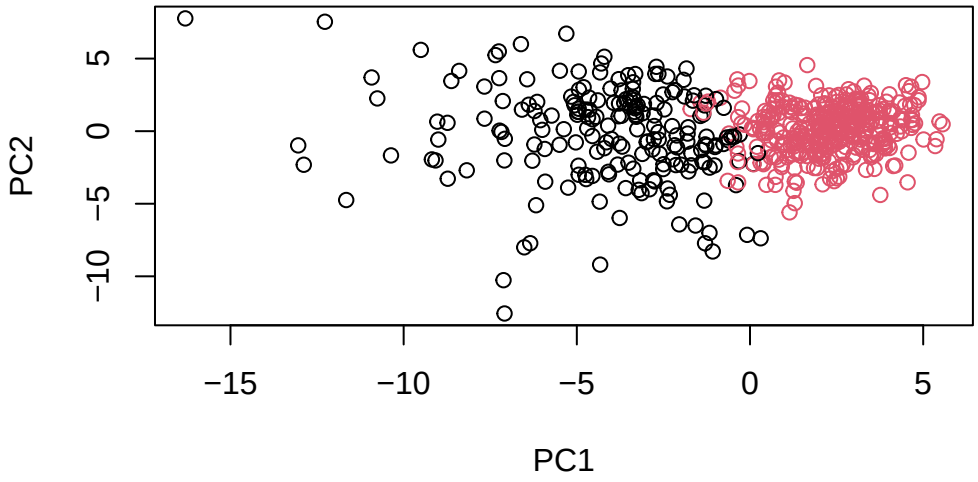
Make a wee “cross-table”

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

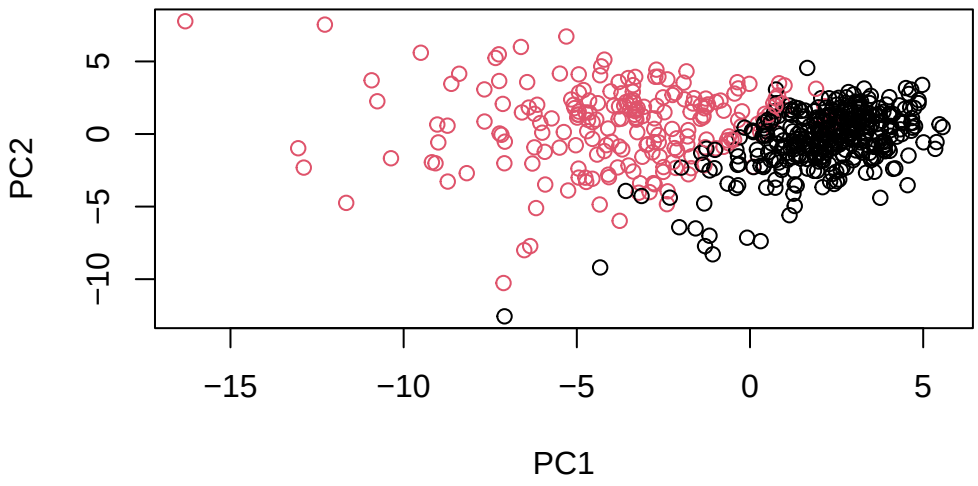
```
    diagnosis
grps   B   M
   1  24 179
   2 333  33
```

TP: 179 FP: 24 TN: 333 FN: 33

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




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



OPTIONAL: Note the color swap here as the hclust cluster 1 is mostly "M" and cluster 2 is mostly "B" as we saw from the results of calling table(grps, diagnosis). To match things up we can turn our groups into a factor and reorder the levels so cluster 2 comes first and thus gets the first color (black) and cluster 1 gets the second color (red).

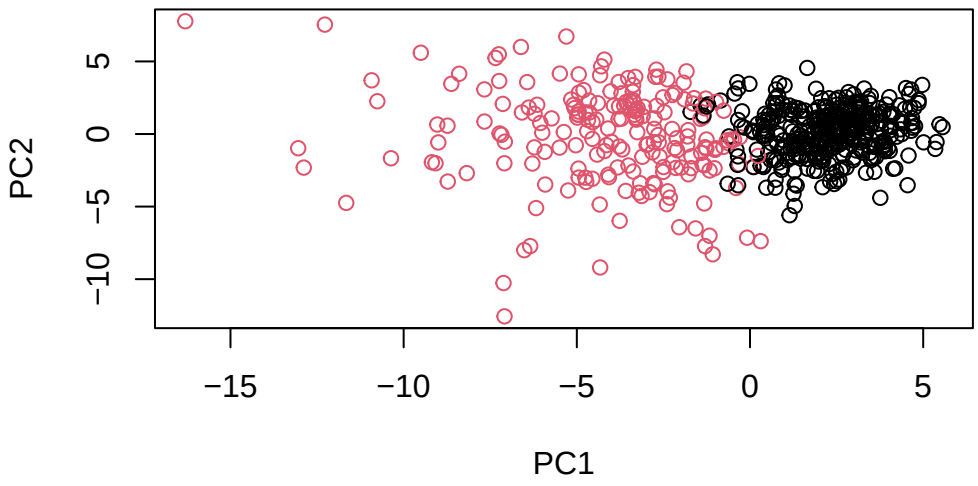
```
g <- as.factor(grps)
levels(g)
```

```
[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)
```



OPTIONAL: We can be fancy and look in 3D with the rgl or plotly packages. Note that this output will not work well with PDF format reports yet so feel free to skip this optional step for your PDF report. If you have difficulty installing the rgl package on mac then you will likely need to install the XQuartz package from here: https://www.xquartz.org. There are also lots of other packages (like plotly) that can make interactive 3D plots.

```
## Use the distance along the first 7 PCs for clustering i.e. wisc.pr$x[, 1:7]
d7 <- dist(wisc.pr$x[,1:7])
wisc.pr.hclust <- hclust(d7, method="ward.D2")
```

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

```
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.

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

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

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

Q13. How well does the newly created model with four clusters separate out the two diagnoses?

It is better than before.

Q14. How well do the 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.

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

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

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

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

The hierarchical clustering models here do a pretty good job, since the resulting clusters have mostly M or mostly B. The numbers in the table here support this.

5. Sensitivity/Specificity

Sensitivity = TP/(TP+FN)

Sensitivity refers to a test's ability to correctly detect ill patients who do have the condition. In our example here the sensitivity is the total number of samples in the cluster identified as predominantly malignant (cancerous) divided by the total number of known malignant samples. In other words: TP/(TP+FN).

Specificity relates to a test's ability to correctly reject healthy patients without a condition. In our example specificity is the proportion of benign (not cancerous) samples in the cluster identified as predominantly benign that are known to be benign. In other words: TN/(TN+FN).

Q15. OPTIONAL: Which of your analysis procedures resulted in a clustering model with the best specificity? How about sensitivity?