class09_mini_project

Anita Wang

10/26/2021

Class 9 Mini-Project:

Unsupervised Learning Analysis of Human Breast Cancer Cells

Preparing the data

Downloading and importing data into R-Studio

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

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

Examine data to ensure correct column name formatting:

```
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
```

The "diagnosis" column (1st column) provides the answer to our analysis! Let's remove it so as to not accidentally include it 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$diagnosis)
```

#1. Exploratory data analysis

First, let's familiarize ourselves with the data:

Q1. How many observations are in this dataset?

```
dim(wisc.data)
```

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

- There are 569 total observations in the dataset

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

```
length(grep("M", diagnosis))
```

```
## [1] 212
```

- There are 212 observations that have a malignant diagnosis

Q3.How many variables/features in the data 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"
```

```
length(grep("mean", colnames(wisc.data)))
```

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

- There are 10 variables in the data that are suffixed with _mean

#2. Principal Component Analysis

Performing PCA

It is important to check if the data need to be scaled before performing PCA. Recall two common reasons for scaling data include:

- The input variables use different units of measurement.
- The input variables have significantly different variances.

Let's 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
```

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

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

```
# 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
```

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?

- 3 PCs are required to describe at least 70% of the orignal variance in the data

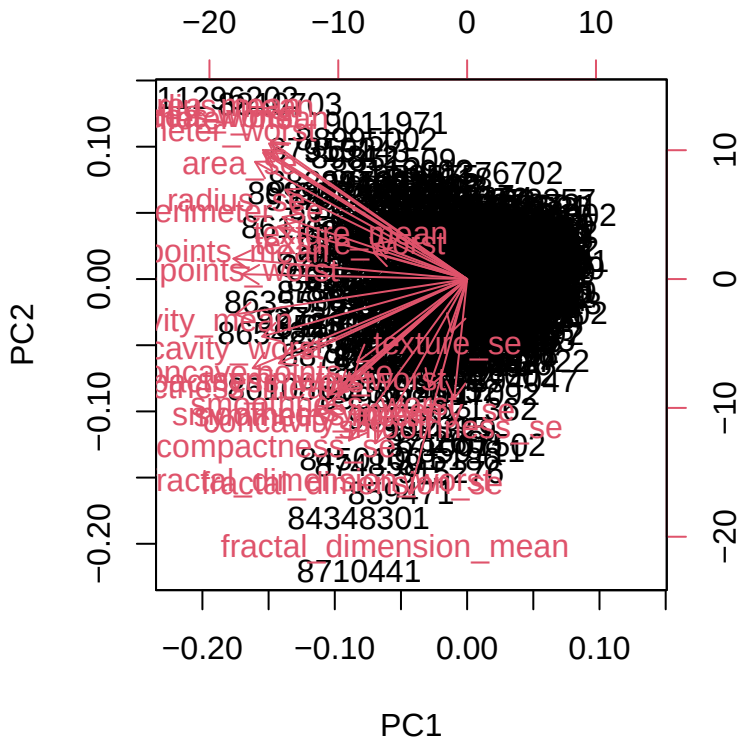
Q6. How many principal components (PCs) are required to describe at least 90% of the original variance in the data?

- 7 PCs are required to describe at least 90% of the orignal variance in the data

Interpreting PCA results

To better understand the PCA model, let's visualize data with a biplot:

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

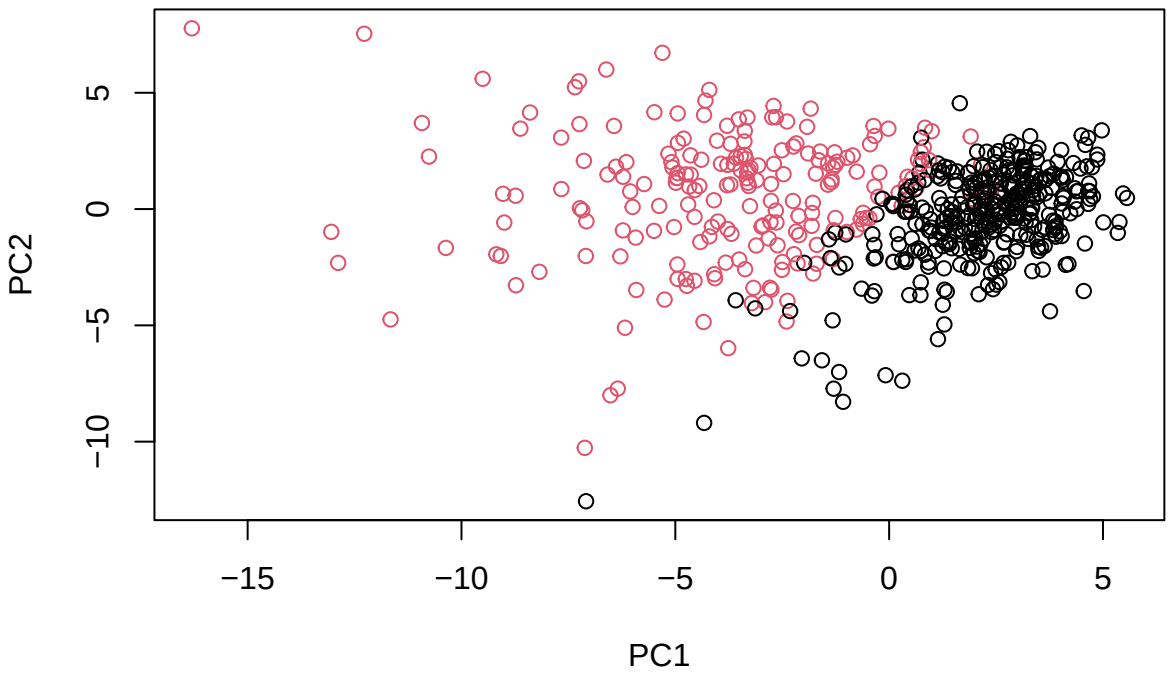


Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?

- This plot is very messy and difficult to understand. I cannot interpret any of the labels and can only see a black ball of text.Any possible trends are virtually invisible as rownames are being used as the plotting character.
- This is a hot mess of a plot and we will need to generate our own plots to make sense of this PCA result.

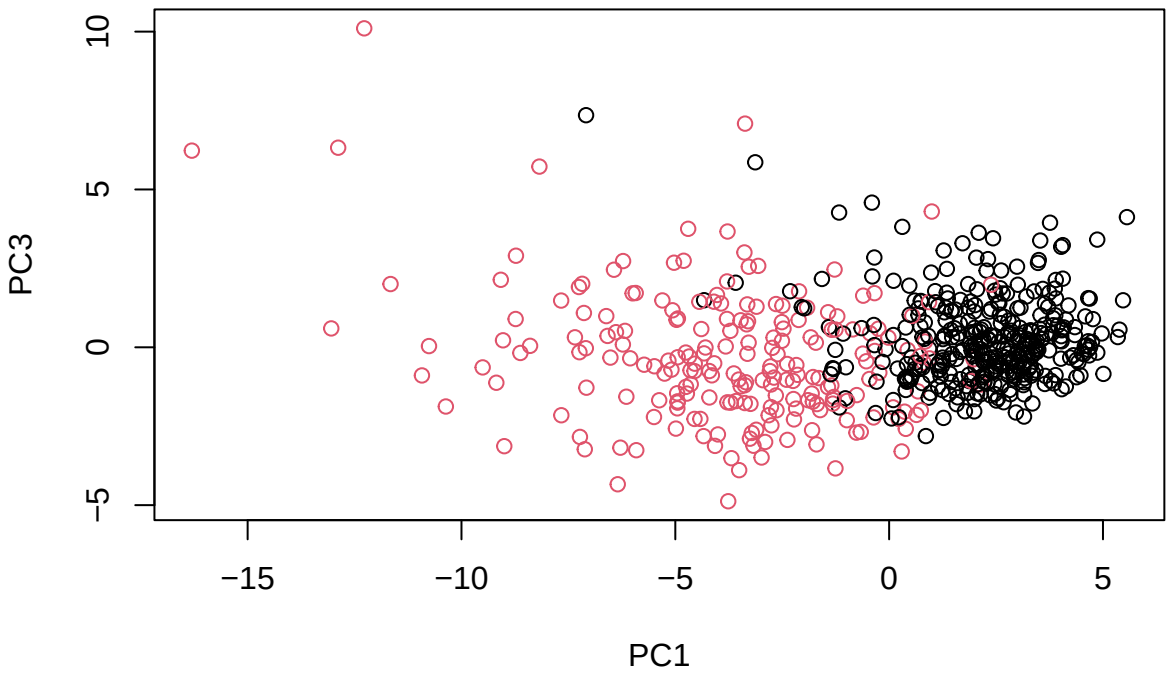
Let's generate a more standard scatter plot of each observation along principal components 1 and 2 (i.e. a plot of PC1 vs PC2 available as the first two columns of wisc.pr$x) and color the points by the diagnosis (available in the diagnosis vector you created earlier):

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



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

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



- When plotting PC1 and PC3 together, there is less distinct separation between the two groups (malignant vs. benign)
- Because principal component 2 explains more variance in the original data than principal component 3, you can see that the first plot has a cleaner cut separating the two subgroups.
- Overall, the plots indicate that principal component 1 is capturing a separation of malignant (red) from benign (black) samples.

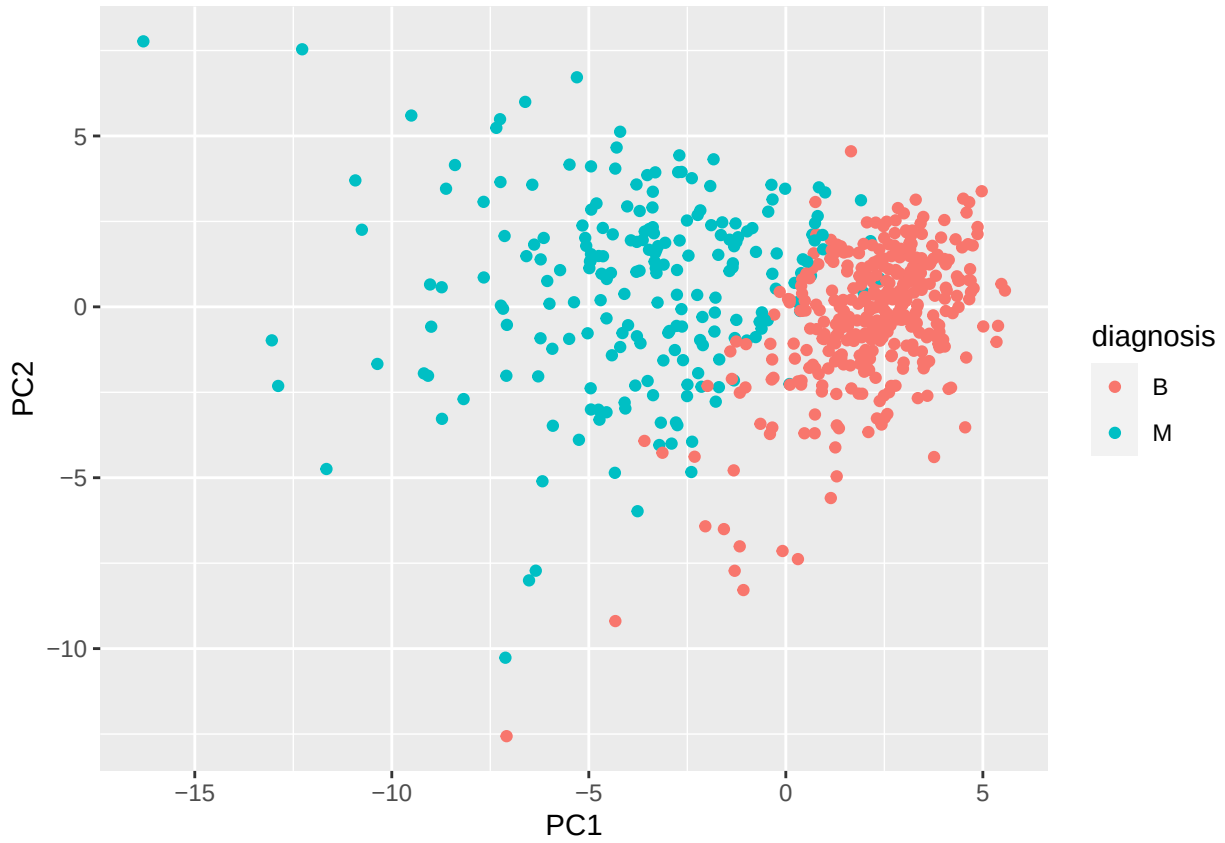
Let's use ggplot2 to make the plots look better!

- Remember! ggplot requires a data.frame as input and we will also need to add our diagnosis vector as a column if we want to use it for mapping to the plot color aesthetic.

```
# 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

Let's produce scree plots that show the proportion of variance explained as the number of principal components increases

You must first prepare the PCA data

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.

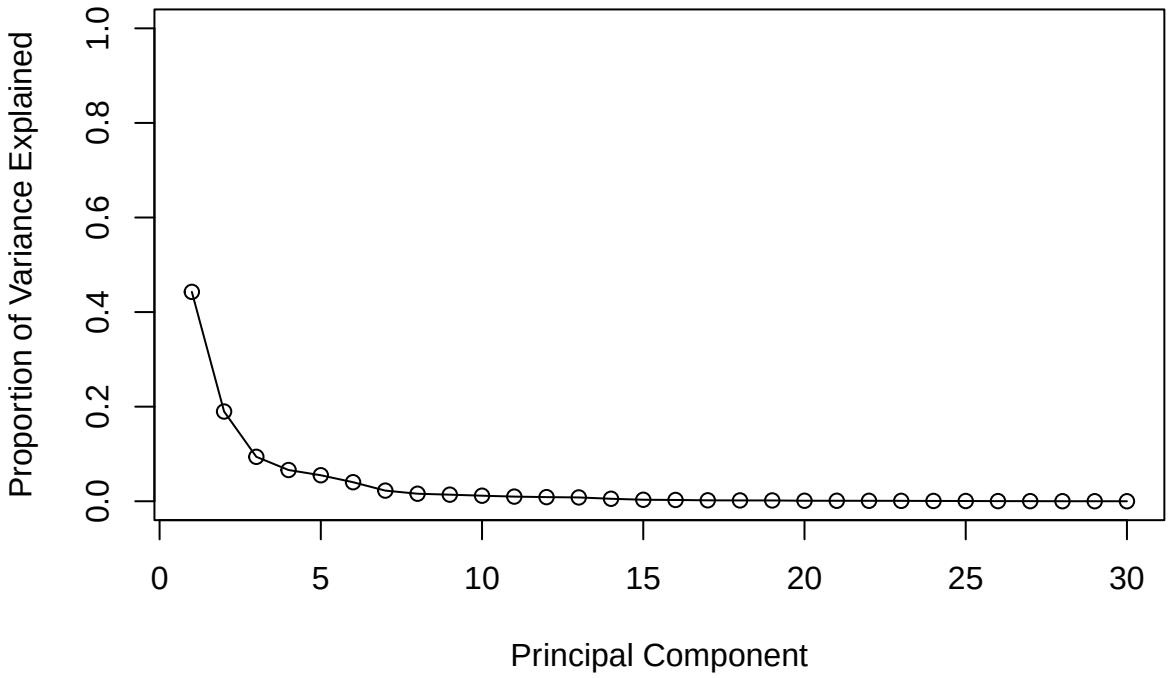
```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

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

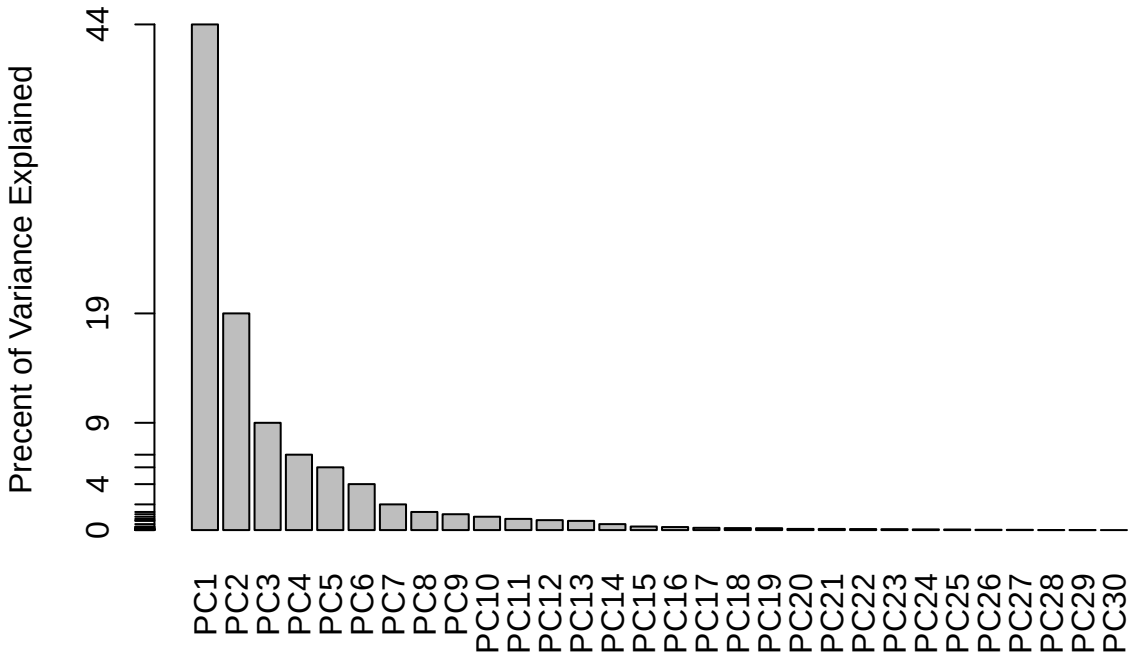
Now, calculate the variance explained by each principal component by dividing by the total variance explained of all principal components. Assign this to a variable called pve and create a plot of variance explained for each principal component.

```
# 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 )
```

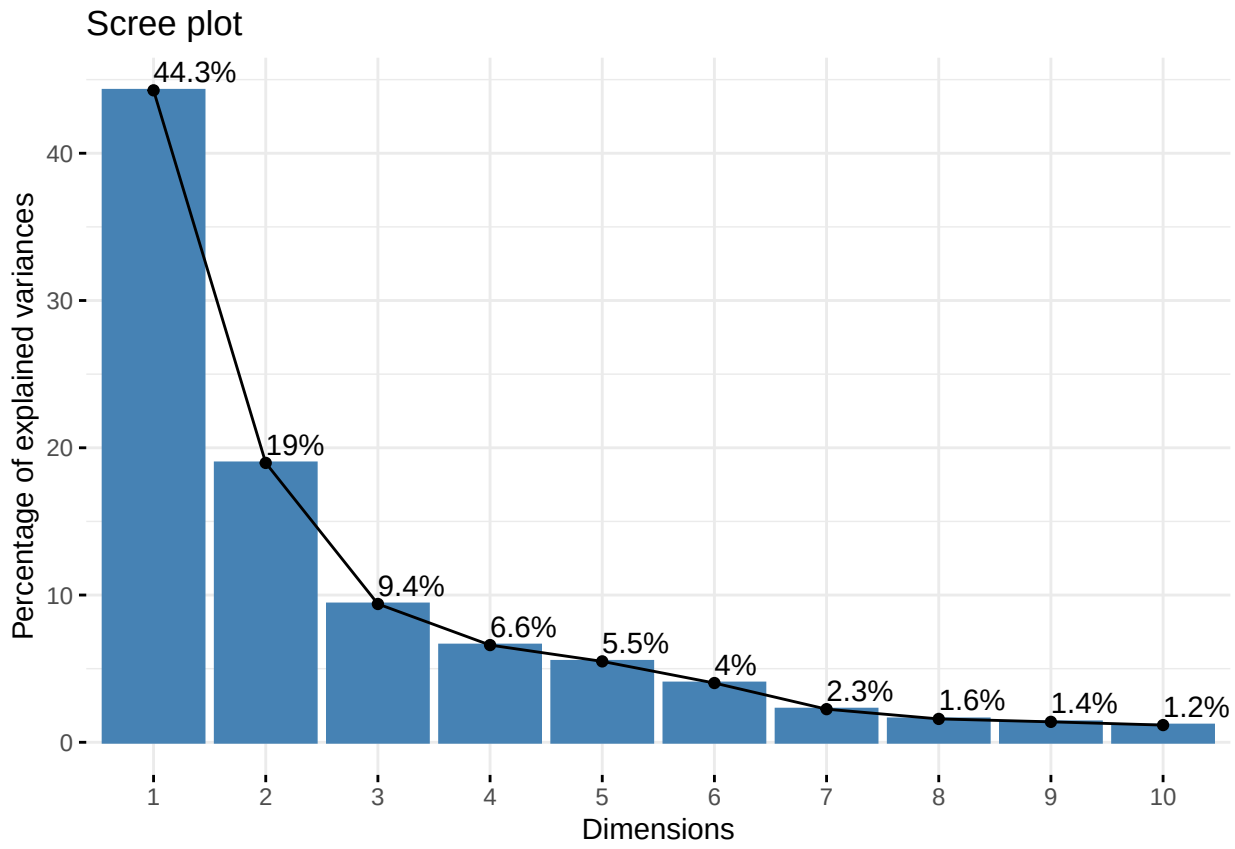


OPTIONAL: There are quite a few CRAN packages that are helpful for PCA. This includes the factoextra package. Feel free to explore this package. For example:

```
## ggplot based graph
#install.packages("factoextra")
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
fviz_eig(wisc.pr, addlabels = TRUE)
```



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?

Consider the influence of each of the original variables upon the principal components (typically known as loading scores:

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

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

- -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
```

```
var <- summary(wisc.pr)
sum(var$importance[3,] <= 0.8)
```

```
## [1] 4
```

- 4 principal components

#3. Hierarchical clustering

Recall: This type of clustering does not assume in advance the number of natural groups that exist in the data.

First prepare data by computing the distance between all pairs of observations. 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.

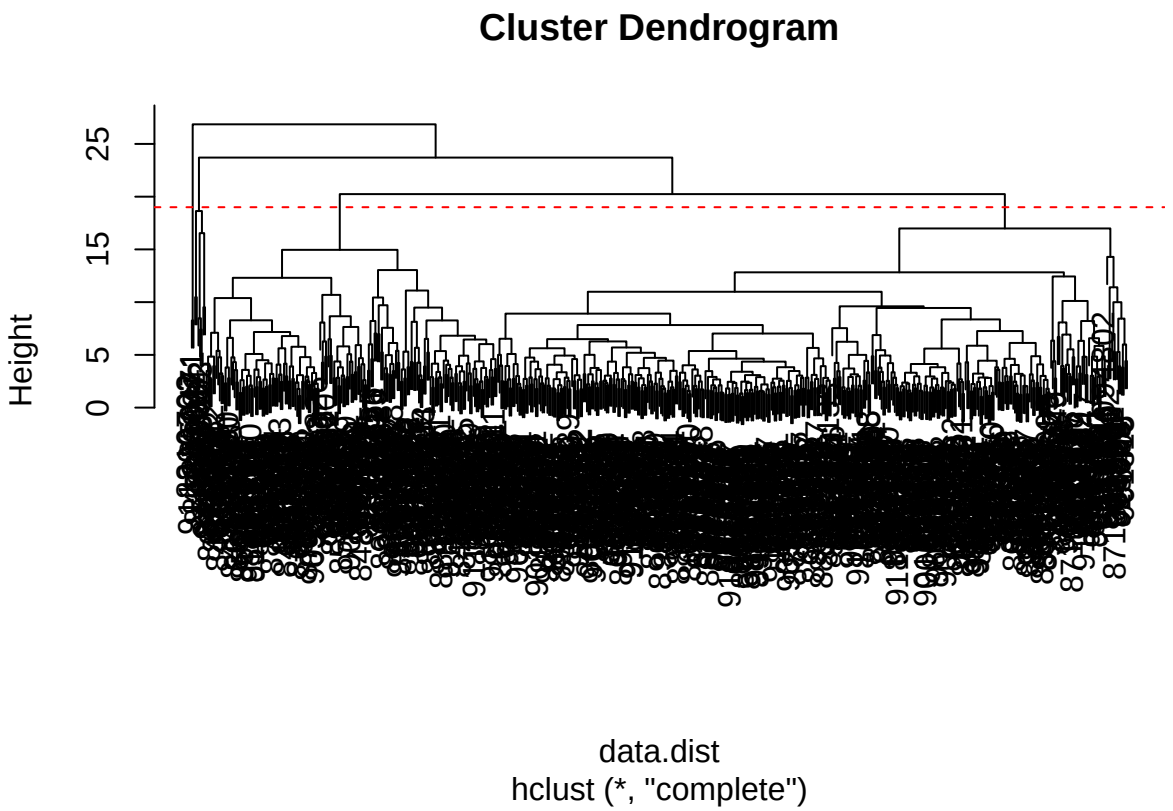
```
wisc.hclust <- hclust(data.dist, method="complete")
```

Results of hierarchical clustering

- Use the hierarchical clustering model you just created to determine a height (or distance between clusters) where a certain number of clusters exists:

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

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



The clustering model has 4 clusters at height = 19

Selecting number of clusters

- Comparing the outputs from your hierarchical clustering model to the actual diagnoses
- 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)
```

- Use the table() function to compare the cluster membership to the actual diagnose

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

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

- Note 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)
- How do different numbers of clusters affect the ability of the hierarchical clustering to separate the different diagnoses?

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

```
wisc.hclust.twoclusters <- cutree(wisc.hclust, k=5)
```

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

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

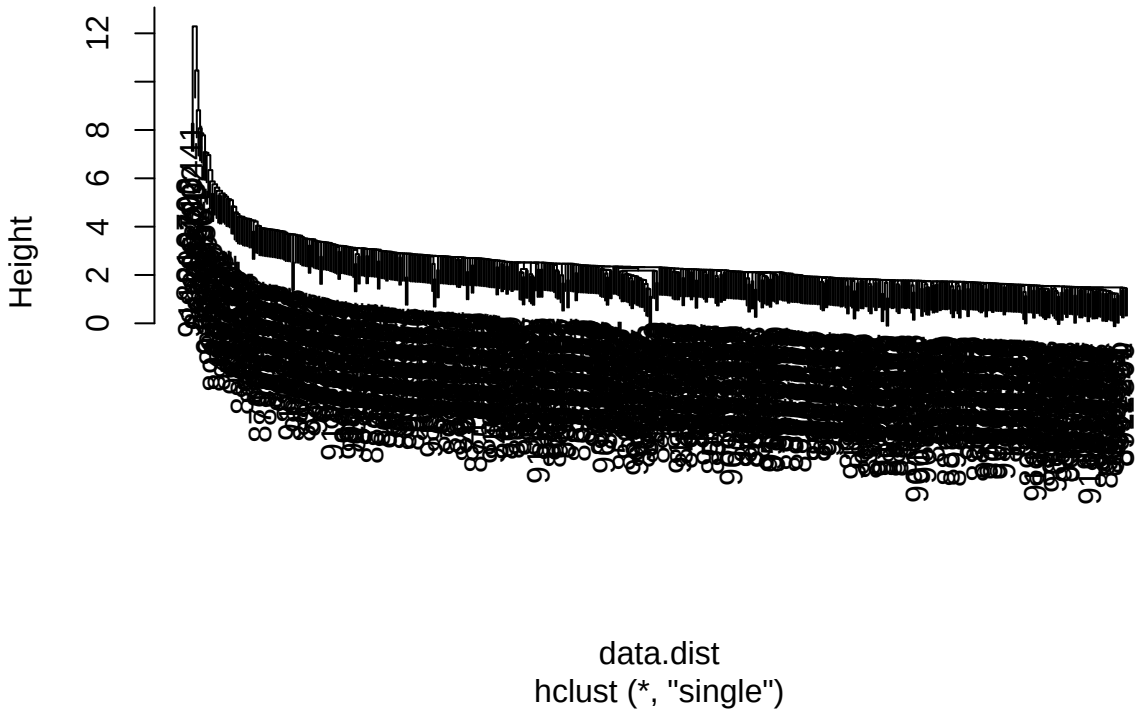
Yes. A slightly better cluster vs.diagnoses match is generated with 5 clusters (k = 5. It provides slightly better resolution, but 4 clusters was already nearing the maximum possibility of resolution via hierarchical clustering, so the improvement is minimal. With 5 clusters, at least the ambiguity in cluster 2 is removed.

Using different methods

- There are a 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"

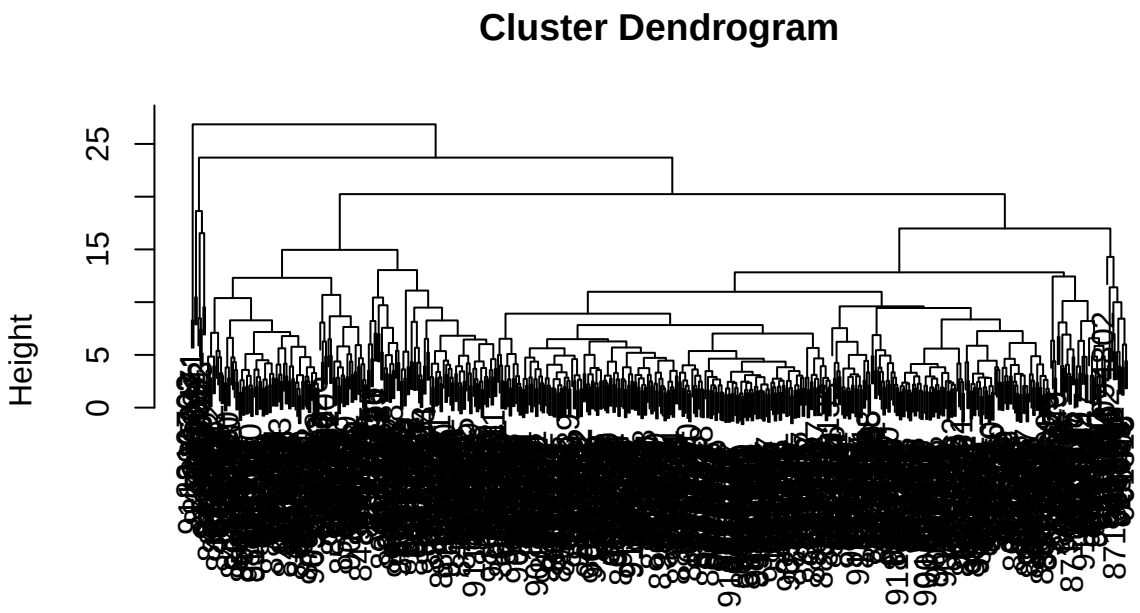
Q13. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning.

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



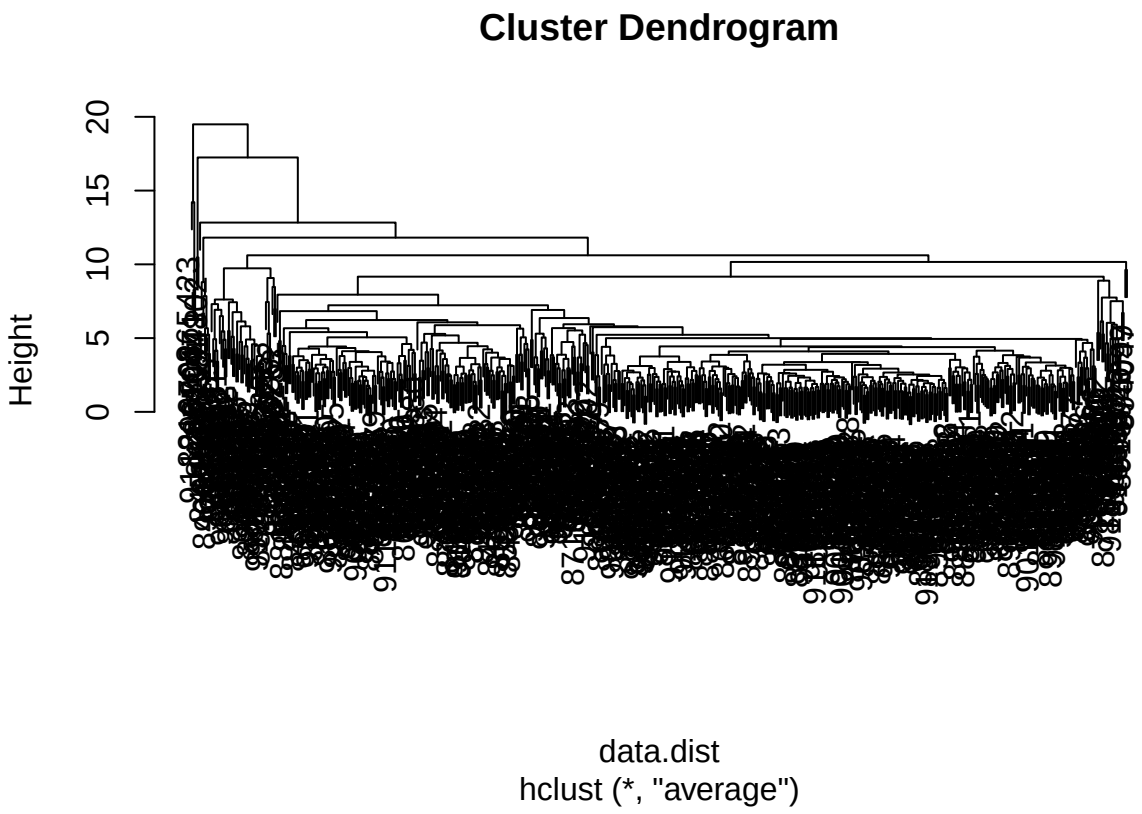

```
#Plot is curvy and stems from a single root!
```

```
#Using "complete" method
plot(wisc.hclust)
```



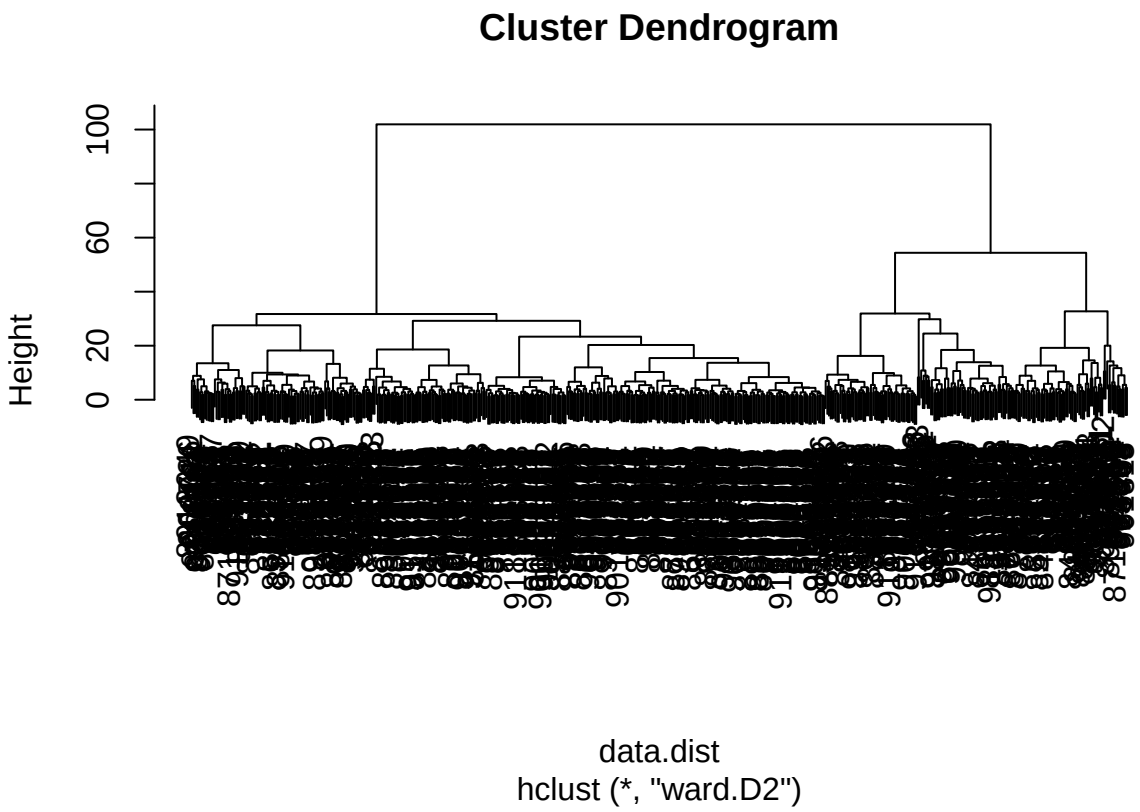

```
#Getting better, but the jumbled mess of data points on the bottom of the tree are still connected to t
```

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




```
#Branches are connected over long distances
```

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




```
#Looks the most interpretable and neatest!
```

The "ward.D2" method gives my favorite results for the same data.dist dataset. It yields the neatest plot with greatest clarity. The branches aren't spanning too long an area and aren't clustered in one central area. See code chunks for further rationals.

#4. OPTIONAL: K-means clustering

K-means clustering and comparing results

Let's see how each clustering model performs in terms of separating the two diagnoses and how the clustering models compare to each other.

Create a k-means model on wisc.data, assigning the result to wisc.km. Be sure to create 2 clusters, corresponding to the actual number of diagnosis. Also, remember to scale the data (with the scale() function and repeat the algorithm 20 times (by setting setting the value of the nstart argument appropriately). Running multiple times such as this will help to find a well performing model.

```
wisc.km <- kmeans(wisc.data, centers = 2, nstart = 20)
```

Use the table() function to compare the cluster membership of the k-means model (wisc.km$cluster) to the actual diagnoses contained in the diagnosis vector.

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

```
##    diagnosis
```

```
##       B   M
##   1 356  82
##   2   1 130
```

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

K-means does a very poor job at separating the two diagnoses. K-means yields worse results than hclust.

Use the table() function to compare the cluster membership of the k-means model (wisc.km$cluster) to your hierarchical clustering model from above (wisc.hclust.clusters). Recall the cluster membership of the hierarchical clustering model is contained in wisc.hclust.clusters object.

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

```
##
## wisc.hclust.clusters   1   2
##                    1  68 109
##                    2   5   2
##                    3 365  18
##                    4   0   2
```

Looking at this second table, it looks like clusters 1, 2, and 4 from the hierarchical clustering model can be interpreted as the cluster 1 equivalent from the k-means algorithm, and cluster 3 can be interpreted as the cluster 2 equivalent.

#5.Combining Methods

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

We take the results of our PCA analysis and cluster in this space `wisc.pr$x`

Clustering on PCA results

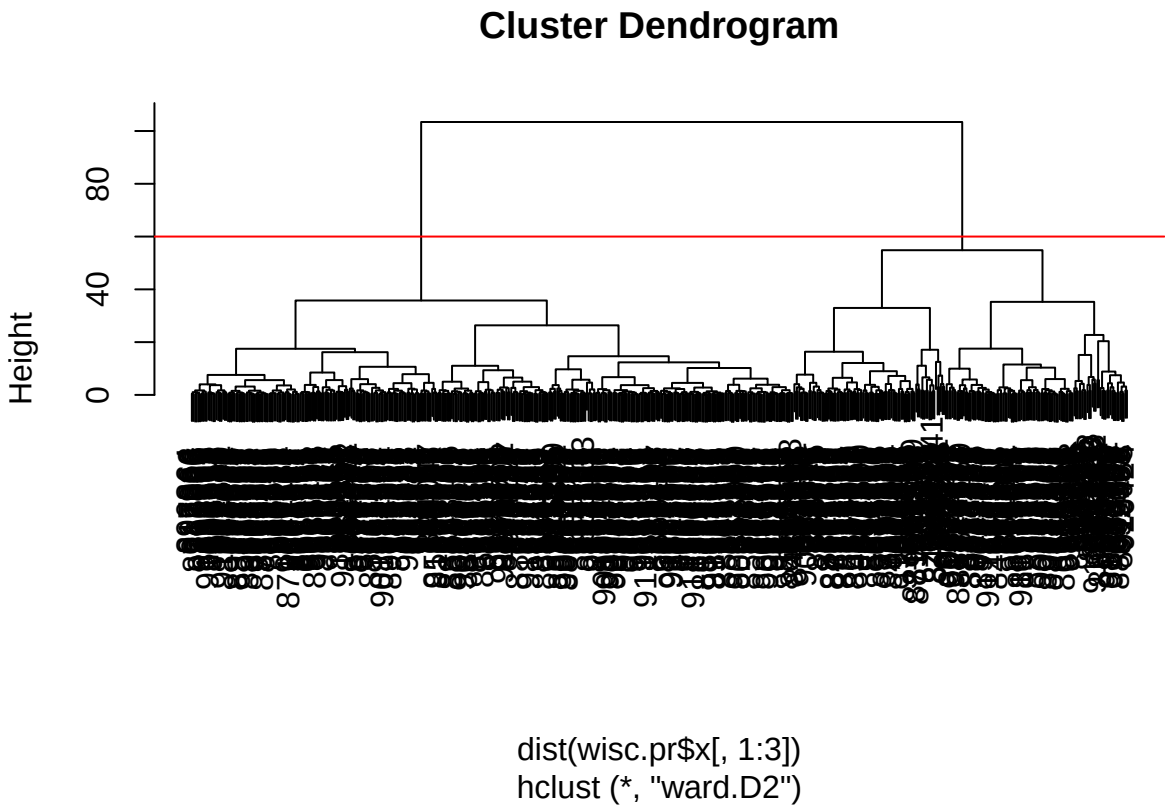
```
summary(wisc.pr)
```

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

Plot my dendrogram and pick height to cut into two groups

```
plot(wisc.pc.hclust)
abline(h=60, col="red")
```



Or use cutree() to cut into two groups (k=2)

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

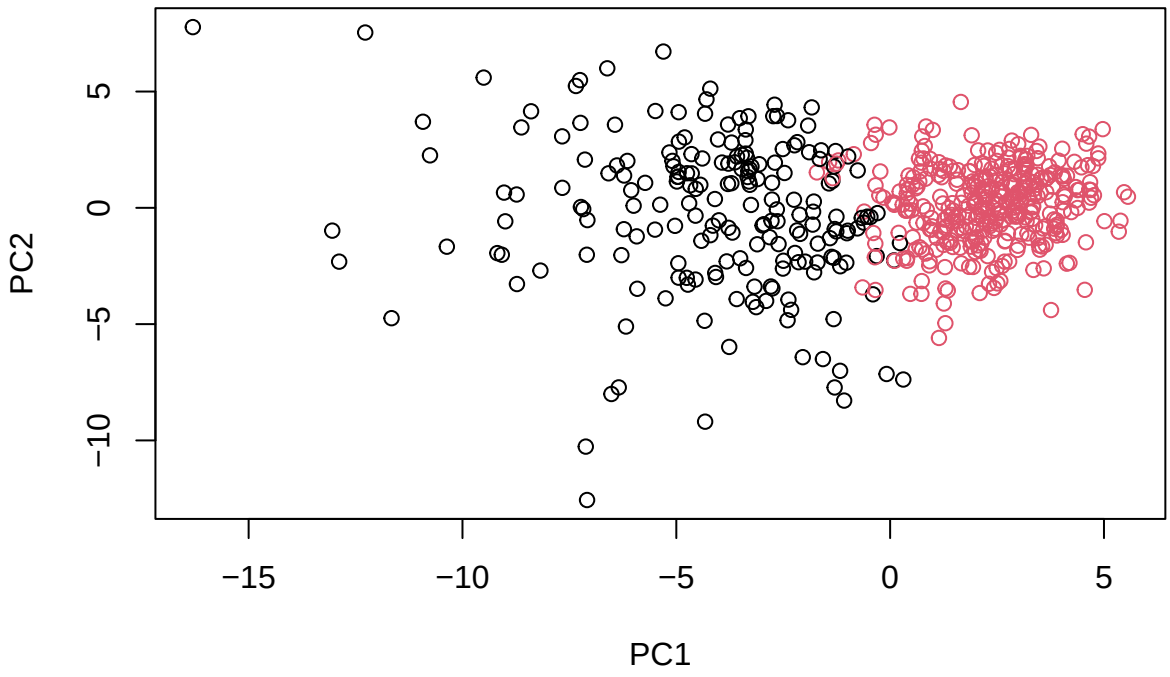
```
## grps
##   1   2
## 203 366
```

Cross table compare of diagnosis and my cluster groups

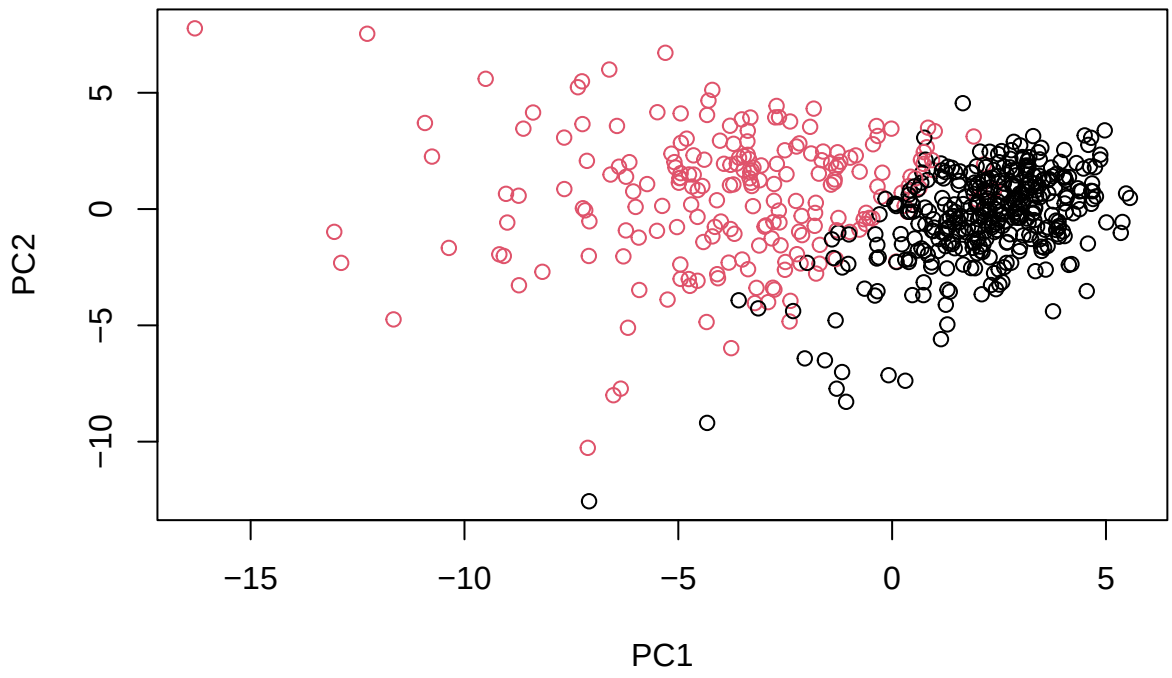
```
table(diagnosis, grps)
```

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

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




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




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

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

This newly created model does a much better hob at separating out the two diagnoses. Most malignant diagnoses have been successfully grouped into cluster 1 and most benign diagnoses have been successfully grouped into cluster 2.

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.

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

```
##    diagnosis
##       B   M
##   1 356  82
##   2   1 130
```

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

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

K-means has the most poor separation resolution out of the three. Many malignant diagnoses are wrongly grouped into cluster 2 with all the benign diagnoses. Hierarchical clustering does a better job at resolving the diagnoses, but there are extraneous groups that are unnecessary as well as ambiguity within clusters. K=4 is the minimum at which clusters begin separating benign and malignant diagnoses. However, the introduction of 4 clusters isn't ideal. Thus, the clustering model after PCA provides the best resolution for clustering based on diagnoses. It successfully reduces the minimum required number of clusters to 2 while still maintaining clear separation of malignant diagnoses into cluster 1 and benign diagnoses into cluster 2.

#6. Sensitivity/Specificity

Accuracy What proportion did we get correct if we call cluster 1 M and cluster 2 B

```
(333 + 179)/(nrow(wisc.data))
```

```
## [1] 0.8998243
```

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).

```
179/(179+33)
```

```
## [1] 0.8443396
```

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).

```
(333/(333+24))
```

```
## [1] 0.9327731
```

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

Clustering on PCA results resulted in a clustering model with the best specificity and sensitivity.

#7. Prediction

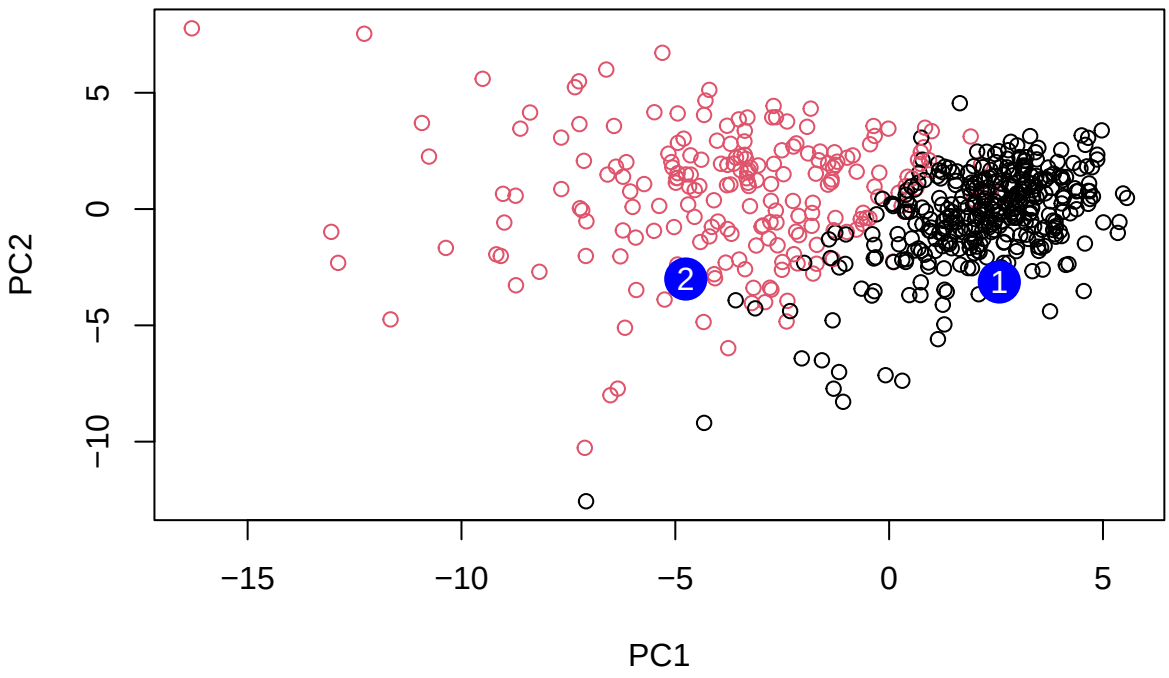
Use the predict() function that will take our PCA model from before and new cancer cell data and project that data onto our PCA space

```
#url <- "new_samples.csv"
url <- "https://tinyurl.com/new-samples-CSV"
new <- read.csv(url)
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
```

Plot our PCA model

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

Based on the results, it will be important to prioritize patient 2 for a follow up.