Unsupervised Learning Analysis of Human Breast Cancer Cells

Joshua Cheung

02/08/2022

1. Exploratory data analysis

Preparing the data.

We first use the read.csv() function to reach the CSV file containing the data.

```
# Here we save the input data file into our Project directory.
fna.data <- "WisconsinCancer.csv"
# We assign the result of the above code to an object called wisc.df.
wisc.df <- read.csv(fna.data, row.names=1)
```

We now examine the input data to ensure column names are set correctly. We use the head() function to preview the first 6 rows.

```
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
```

We now note that the first column here (wisc.df$diagnosis) is a pathologist's expert diagnosis. We wil not be using this for the unsupervised analysis in this project. Thus, to ensure we don't accidently include this column we define a new data.frame which omits this first column.

```r
# Note we use -1 index here to remove the first column.
wisc.data <- wisc.df[,-1]
```

Now, we also define a separate vector called diagnosis that contains the data from the diagnosis the column of the original data set.

```r
# We create diagnosis vector for later.
diagnosis <- as.factor(wisc.df$diagnosis)
```

Exploratory data analysis

Q1. How many observations are in this dataset?

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

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

There are 569 observations in this data set.

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

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

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

There are 212 observations with a malignant diagnosis.

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

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

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

There are 10 variables/features in the data suffixed with _mean.

Principal Component Analysis

Performing PCA

We first check the standard deviation of the features of the wisc.data to determine if the data should be scaled.

```
# We check the 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
```

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

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

We then execute the PCA with the prcomp() function on the wisc.data and scale if appropriate. We also assign the output model to wisc.pr.

```r
# We perform PCA on the wisc.data.
wisc.pr <- prcomp(wisc.data, scale=TRUE)
```

Now we look at a summary of the results of wisc.pr.

```r
# We inspect the summary.
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)?**

The summary print out indicate PC1 account for 44.3% of the original variance.

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

Three principal components (PC1, PC2, and PC3) are required to describe at least 70% of the original variance in the data.

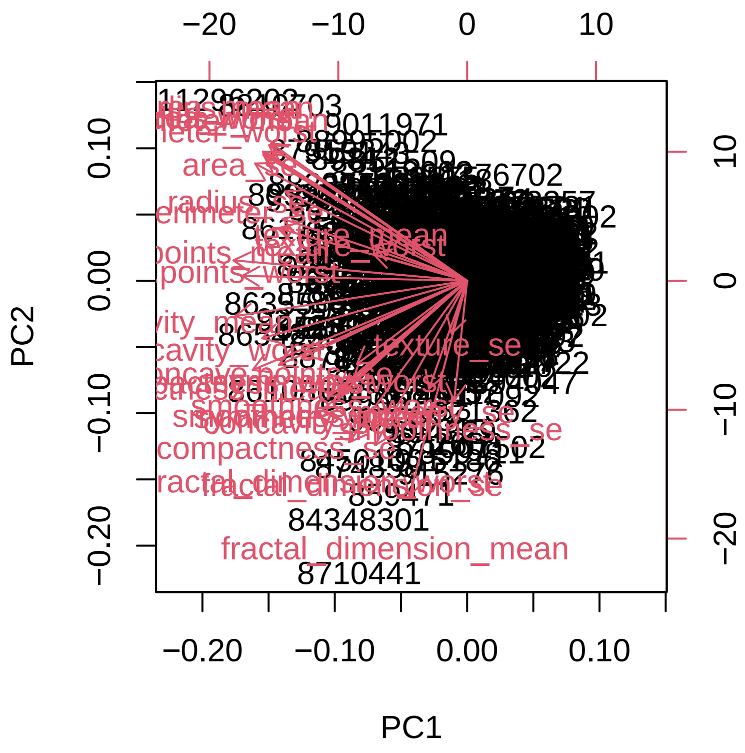
> **Q6. How many principal components (PCs) are required to describe at least 90% of the original variance in the data?**

Seven principal components (PC1, PC2, PC3, PC4, PC5, PC6, and PC7) are required to describe at least 90% of the original variance in the data.

Interpreting PCA results

We create a biplot of the wisc.pr using the biplot() function.

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

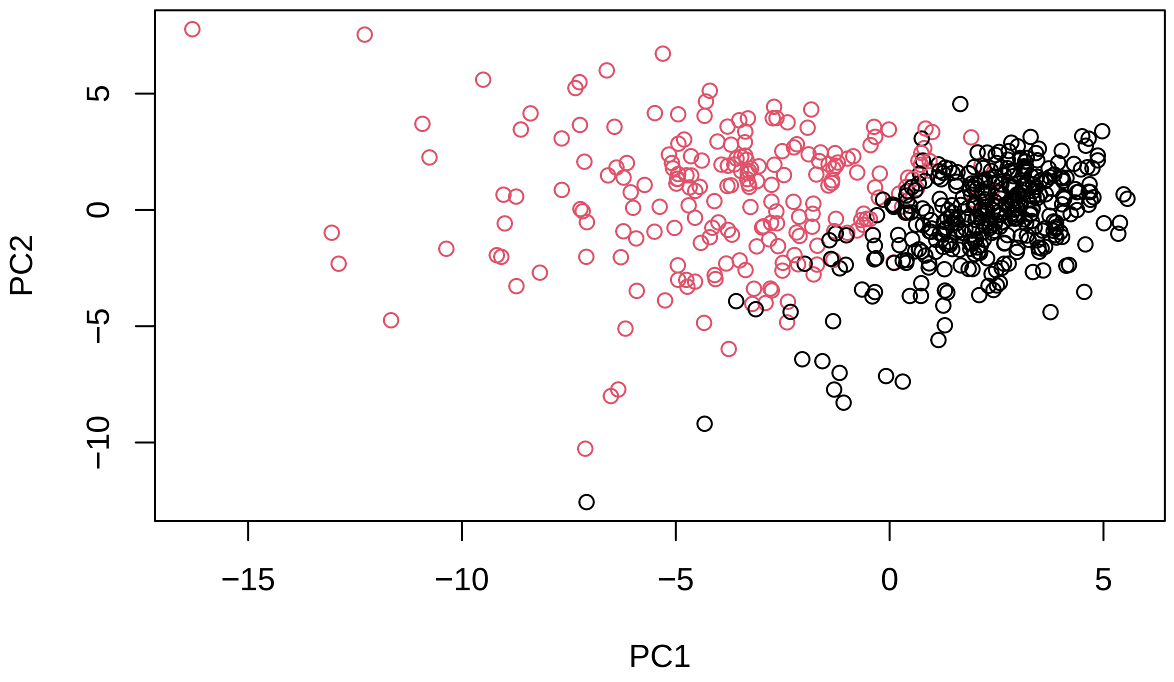


Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?

We see that there is not a central group of variable around the the middle of each principal component. Instead we see that the variable are clustered all around the periphery, and there are not discernible groups. As a results this plot is difficult to understand.

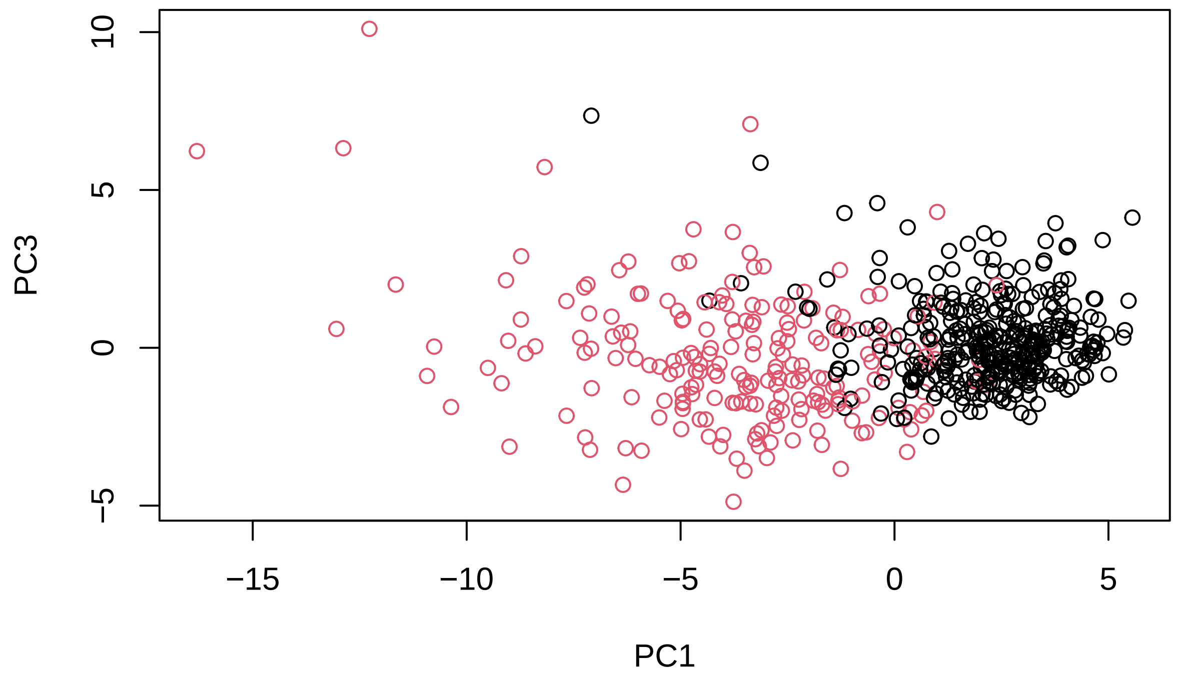
We generate a more standard scatter plot of each observation along PC1 and PC2 and color the points by the diagnosis.

```r
# We create scatter plot observations by components 1 and 2.
plot(wisc.pr$x[,1], wisc.pr$x[,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?

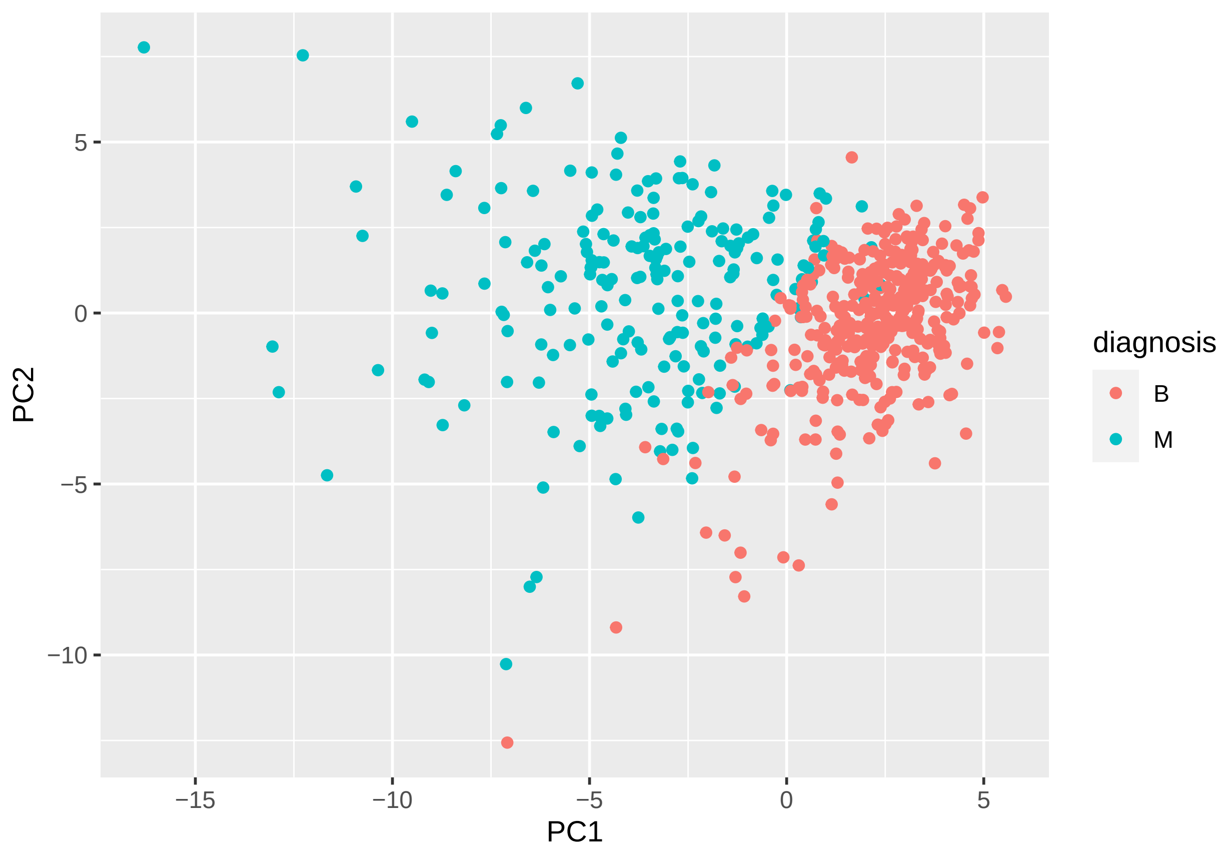
```r
# We create scatter plot observations by components 1 and 3.
plot(wisc.pr$x[,1], wisc.pr$x[,3], col = diagnosis ,
     xlab = "PC1", ylab = "PC3")
```



To answer question 8, we see that PC2 accounts for more of the variance in the original data than principal component 3. As a result, the plot of PC1 vs PC2 has greater separation of the malignant (red) and benign (black) subgroups than the plot of PC1 and PC3. Also, both plots generally indicate the PC1 is capturing a separation of malignant and benign samples.

Now we turn to using ggplot2 to make the figure more attractive. Recall that ggplot2 will require a data.frame input. Additionally, we will also require our diagnosis vector as a column if we want to use it for mapping to the plot color aesthetic.

```r
# We first create a data.frame for ggplot2.
df <- as.data.frame(wisc.pr$x)
df$diagnosis <- diagnosis
# Be sure to load the ggplot2 package.
library(ggplot2)
# Now we make a scatter plot colored by the diagnosis vector.
ggplot(df) +
  aes(PC1, PC2, col=diagnosis) +
  geom_point()
```



Variance explained

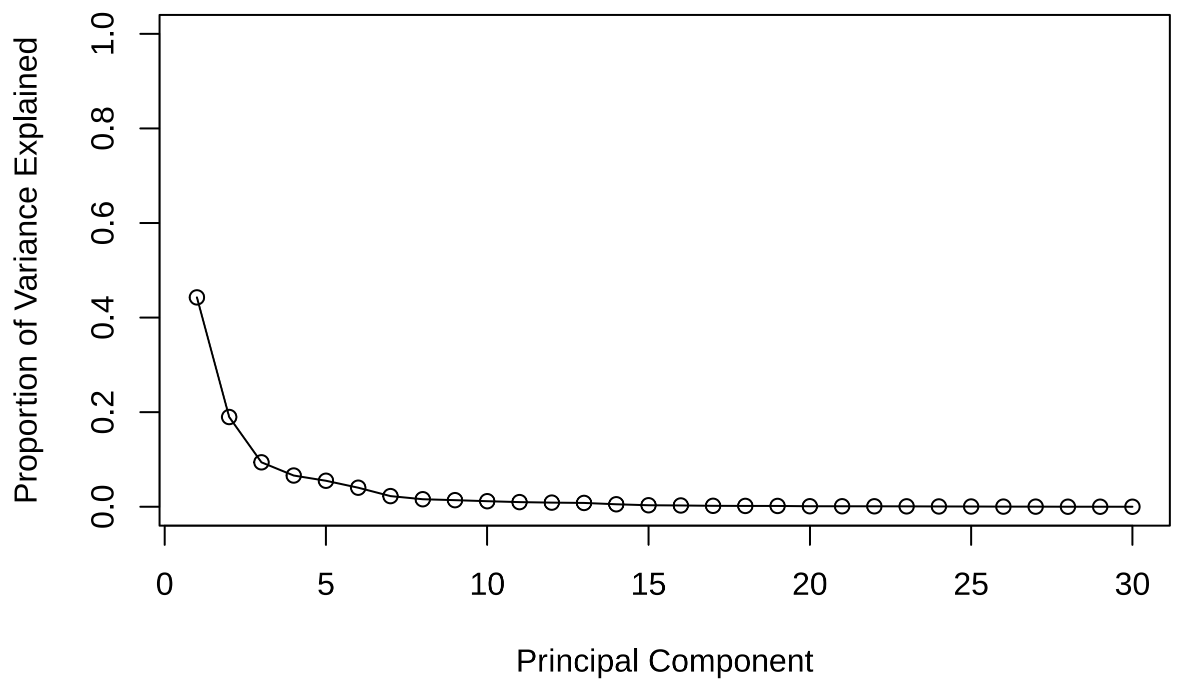
We calculate the variance of each principal component by squaring the sdev component of wisc.pr and saving the result to an object called pr.var.

```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
# We preview the first six results of pr.vr
head(pr.var)
```

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

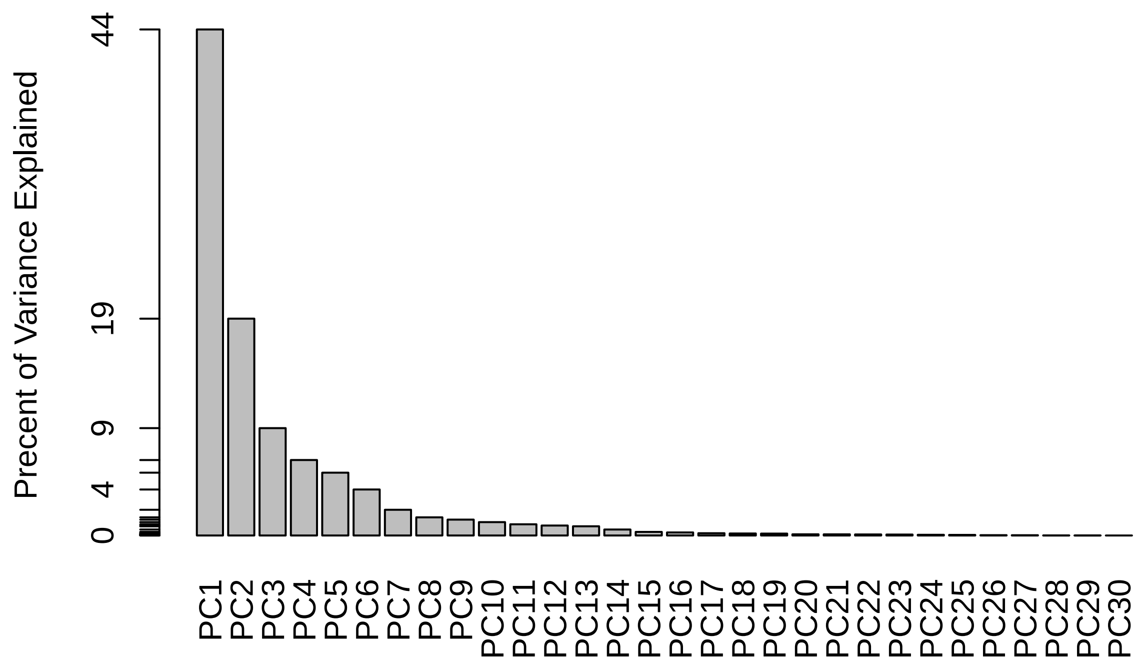
Now we compute the variance explained by each principal component by dividing the total variance explained of all principal components. We assign this to a variable called pve and creat a plot of variance explained for each principal component.

```
# We define pve as the variance explained by each principal component.
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")
```



We now consider an alternative scree plot of the same data. We create a bar plot as follows.

```
# Note the 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 )
```

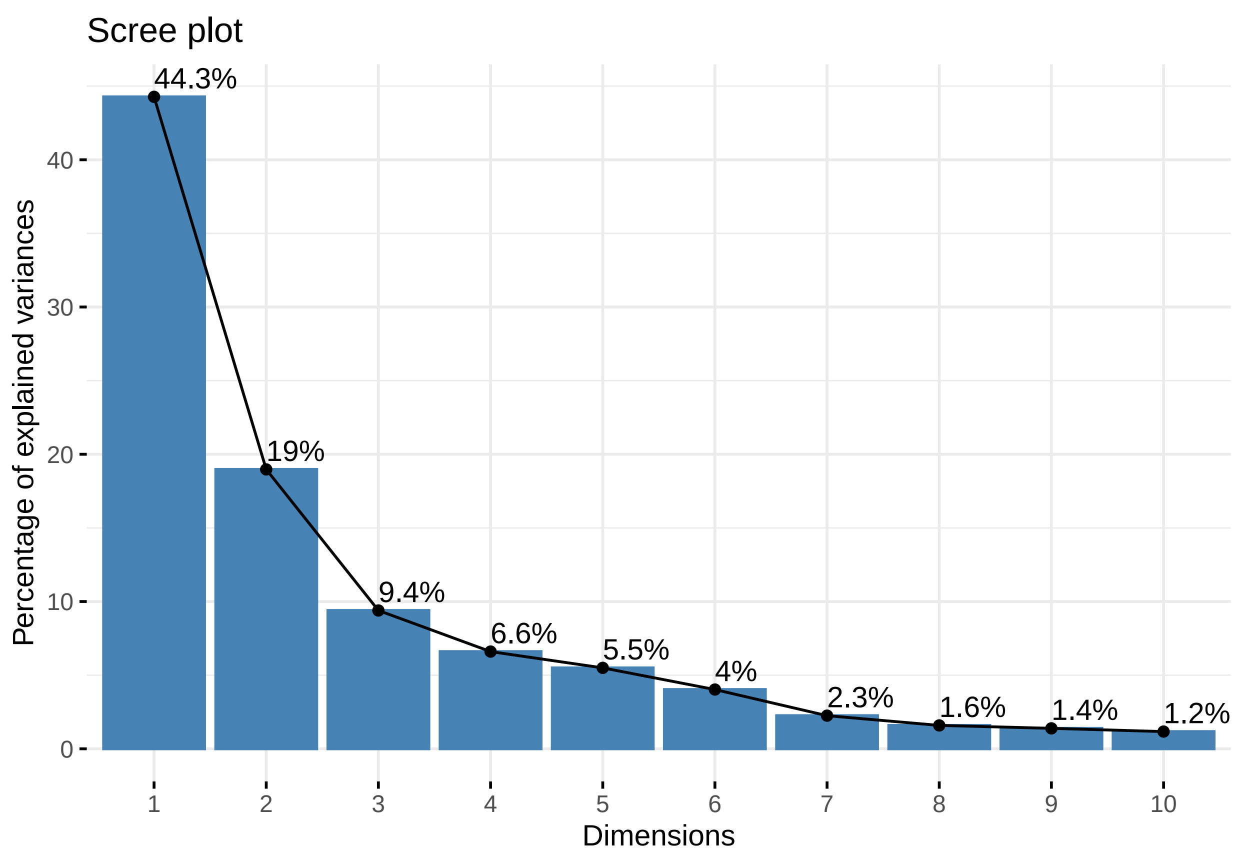


We now note that the CRAN package factoextra is helpful for PCA, We use this package as follows:

```
# ggplot based graph
# install.packages("factoextra") ## Un-comment to install if necessary
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?**

```
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
```

Answering question 9, we see that the component of the loading vector for the feature concave.points_mean is -0.26. This value is relatively small compared to the other feature's values. This indicates that this variable's influence upon the principal components is slightly smaller relative to the other variables' contributions.

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

The minimum number of principal component required to explain 80% of the variance of the data is five principle components (PC1, PC2, PC3, PC4, and PC5).

3. Hierarchial clustering

We first scale the wisc.data and assign the result to data.scaled.

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

Now, we compute the Euclidean distances between all pairs of obeservations in the new scaled dataset and assign this result to data.dist.

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

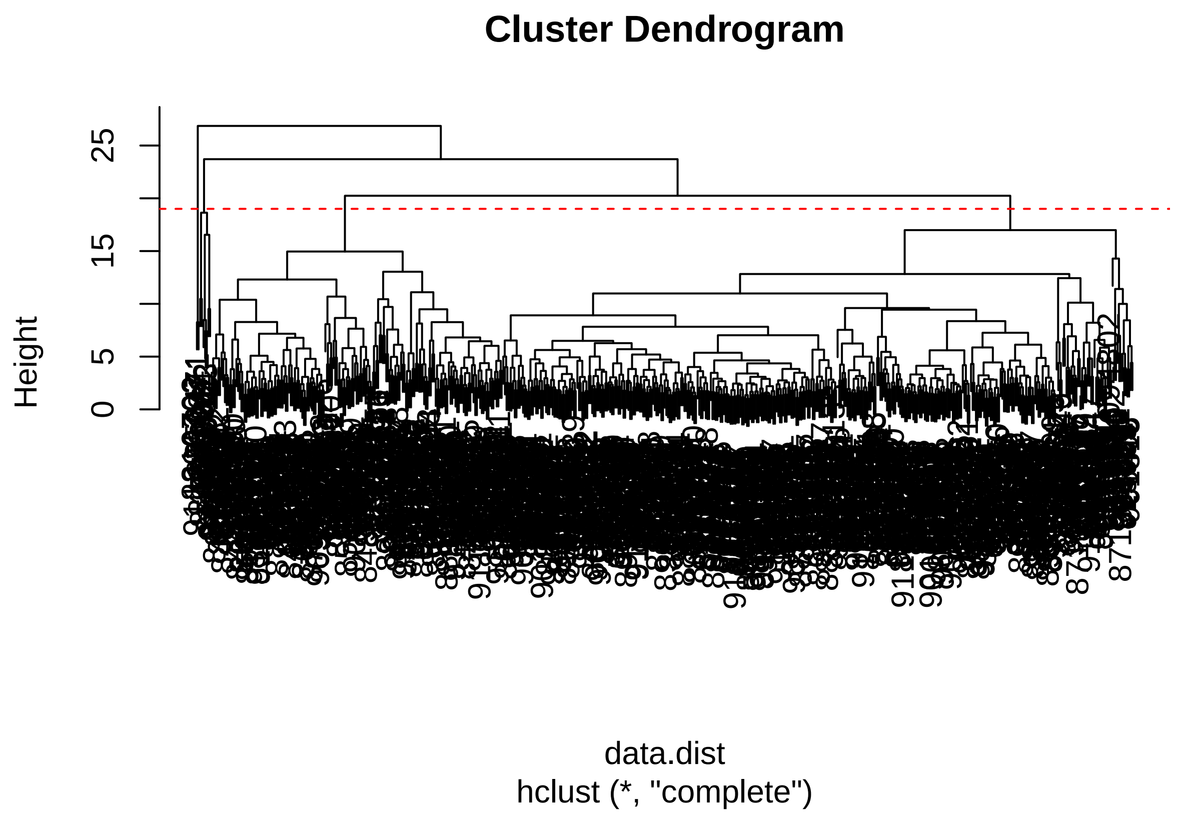
Now we can create the hierarchical clustering model using complete linkage. We manually specify the argument method to hclust() and assign the result to wisc.hclust.

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

Results of hierarchial clustering

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

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



To answer question 11, the clustering model has 4 clusters at a height of 19.

Selecting number of clusters

We use cutree() to cut the tree so that it has 4 clusters.

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

We can now 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
```

Thus we have picked 4 clusters. We see that cluster 1 largely corresponds to malignant cells (with a diagnosis value of 1), and cluster 3 largely corresponds to benign cells (with diagnosis values of 0).

We now compare the results we just got, with a the results from different number of clusters.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

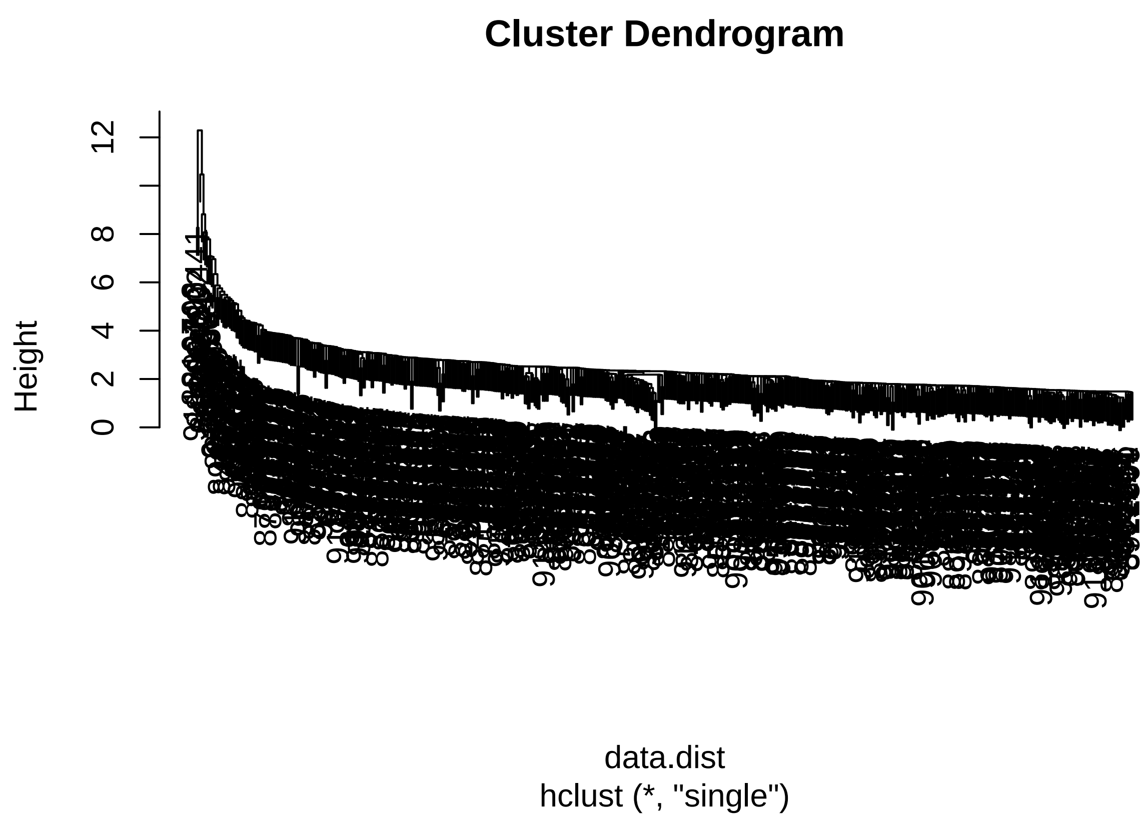
```
##                          diagnosis
## wisc.hclust.clusters10   B   M
##                      1  12  86
##                      2   0  59
##                      3   0   3
##                      4 331  39
##                      5   0  20
##                      6   2   0
##                  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

To answer the question 12, after checking all cluster vs diagnoses matches for different numbers of cluster between 2 and 10, we were unable to find a better cluster vs diagnoses match than the one generated for 4 clusters.

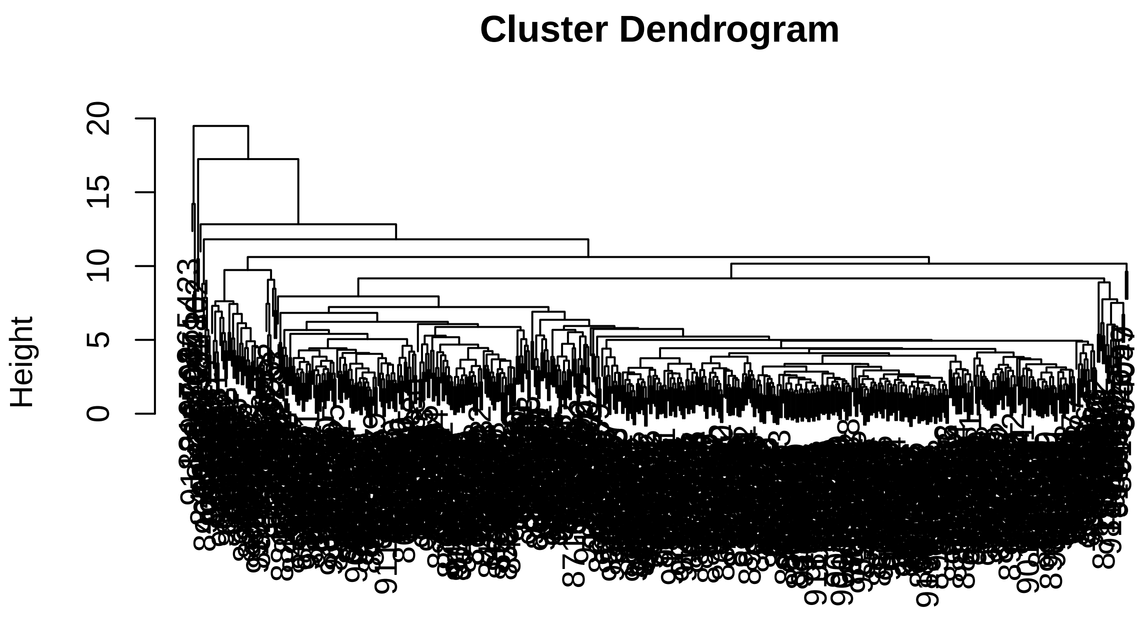
Using different methods

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

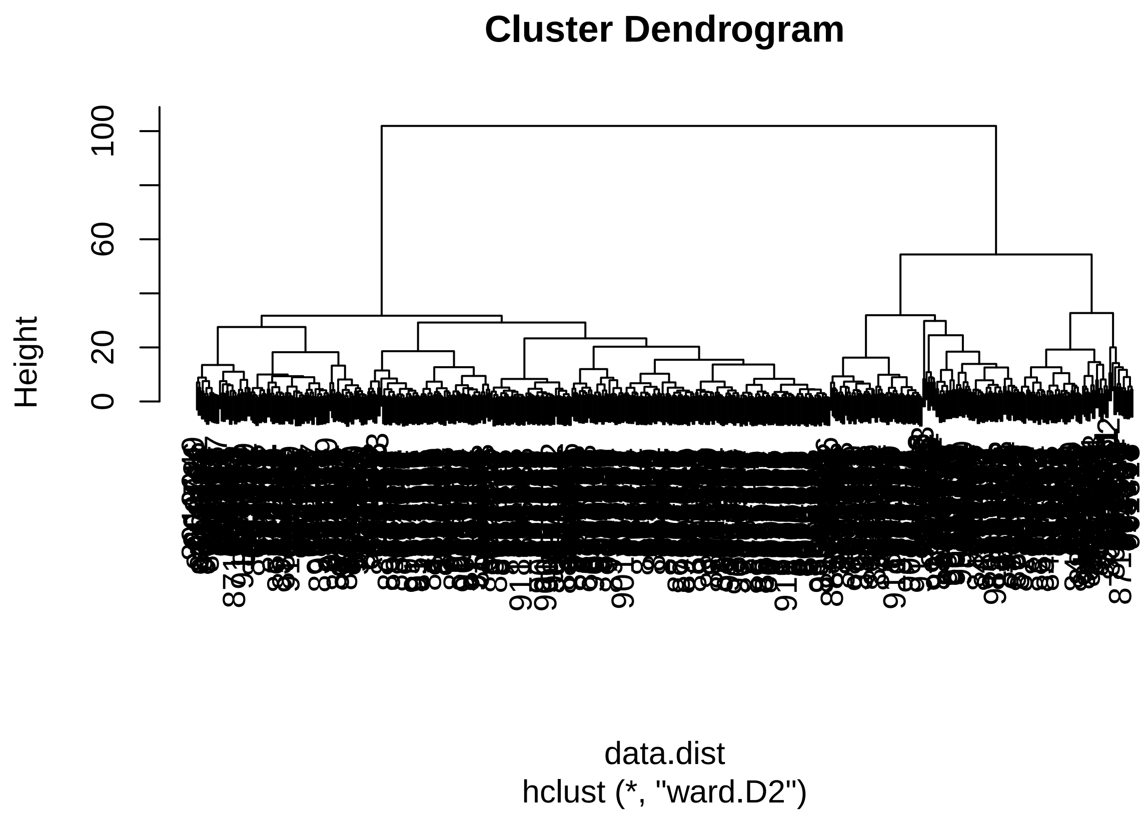
```r
# We try the single linkage method:
wisc.hclust.single <- hclust(data.dist, method="single")
plot(wisc.hclust.single)
```



```r
# We try the average linkage method:
wisc.hclust.average <- hclust(data.dist, method="average")
plot(wisc.hclust.average)
```



```
# We try the ward.D2 linkage method:
wisc.hclust.wardd2 <- hclust(data.dist, method="ward.D2")
plot(wisc.hclust.wardd2)
```



Answering question 13, our favorite method to use is the ward.D2 method as this yields the dendogram that appears to be the neatest and most easy to view out of all the methods. There are two clear main branches of the dendogram indicating two main clusters. The dendogram from the single linkage method is rather messy and difficult to view, and the dendogram from the average linakge method is slightly cleaner, but very similar to the complete linkage method's dendogram.

4. OPTIONAL: K-means clustering

We create a k-means model on wisc.data and assign the result to wisc.km.

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

We use the table function to compare the cluster membership of the k-means model to the actual diagnoses contained in the diagnosis vector.

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

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

We also compare the k-means model to the hierarchical clustering model.

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

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

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

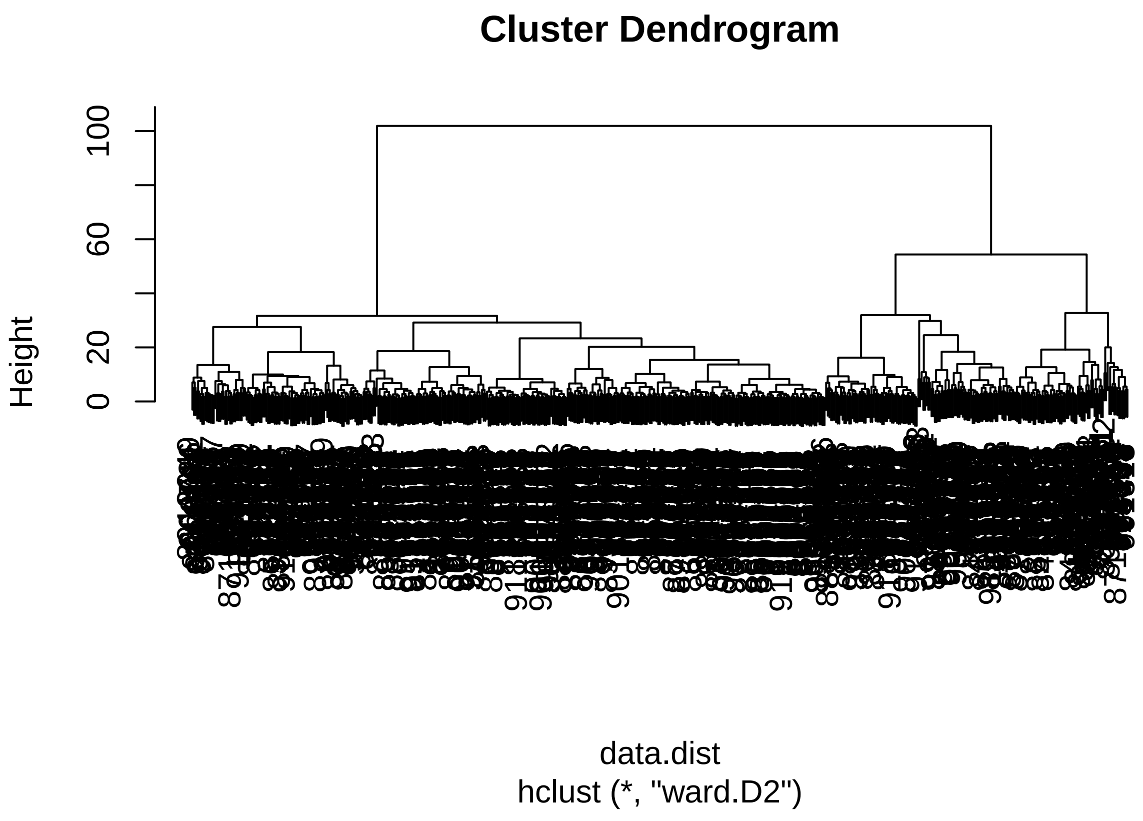
K-means separates the two diagnoses fairly well as we see a clear distinction between the two groups. When comparing it with the hclust, we see that clusters 1, 2, and 4 from the hclust results of the second table are roughly equivalent to cluster 1 of the k-means results from the first table. Similarly,clusters 3 from the hclust results of the second table are roughly equivalent to cluster 2 of the k-means results from the first table.

5. Combining methods

Clustering on PCA results

We create a hierarchical clustering model with the linkage method="ward.D2" and assign the results to wisc.pr.hclust.

```
wisc.pr.hclust <- hclust(data.dist, method="ward.D2")
plot(wisc.hclust.wardd2)
```



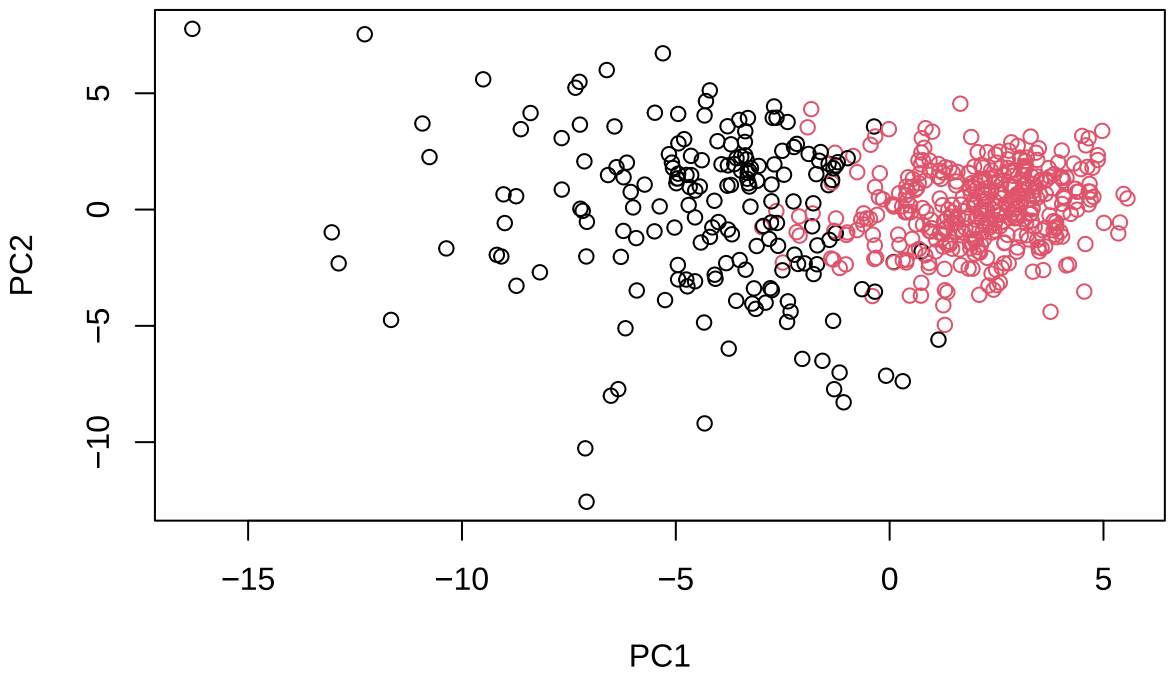
```
grps <- cutree(wisc.pr.hclust, k=2)
table(grps)
```

```
## grps
##   1   2
## 184 385
```

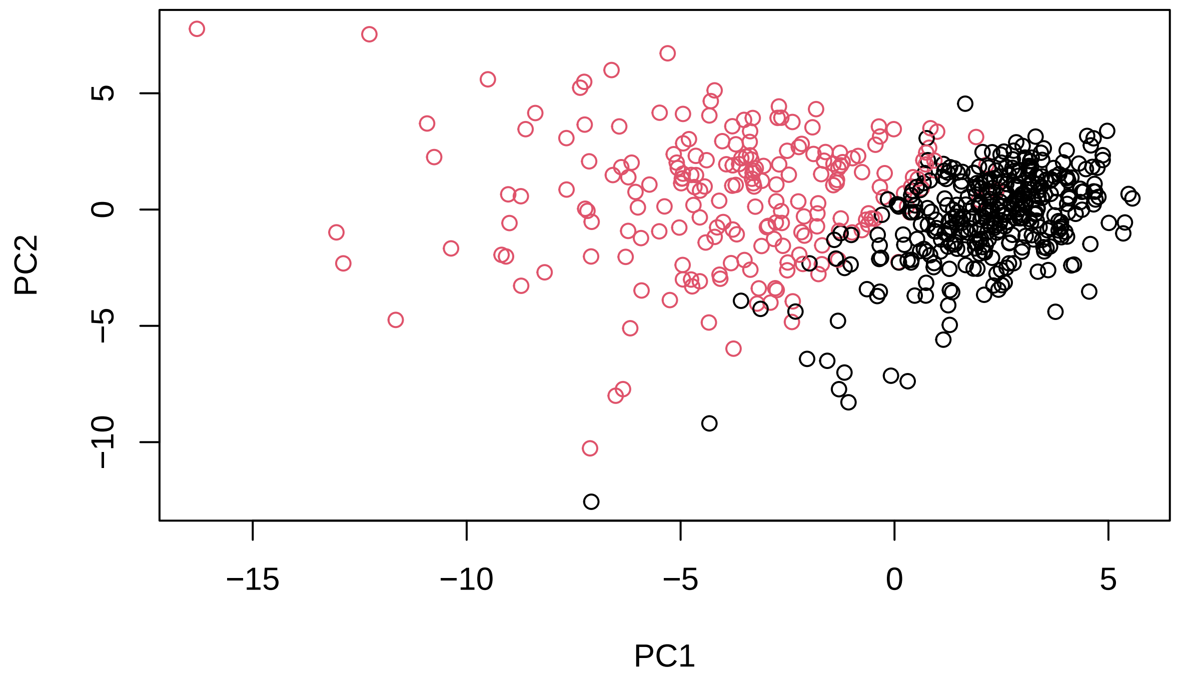
```
table(grps, diagnosis)
```

```
##     diagnosis
## grps   B   M
##    1  20 164
##    2 337  48
```

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



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



Note that there is color swap here. We fix this as follows.

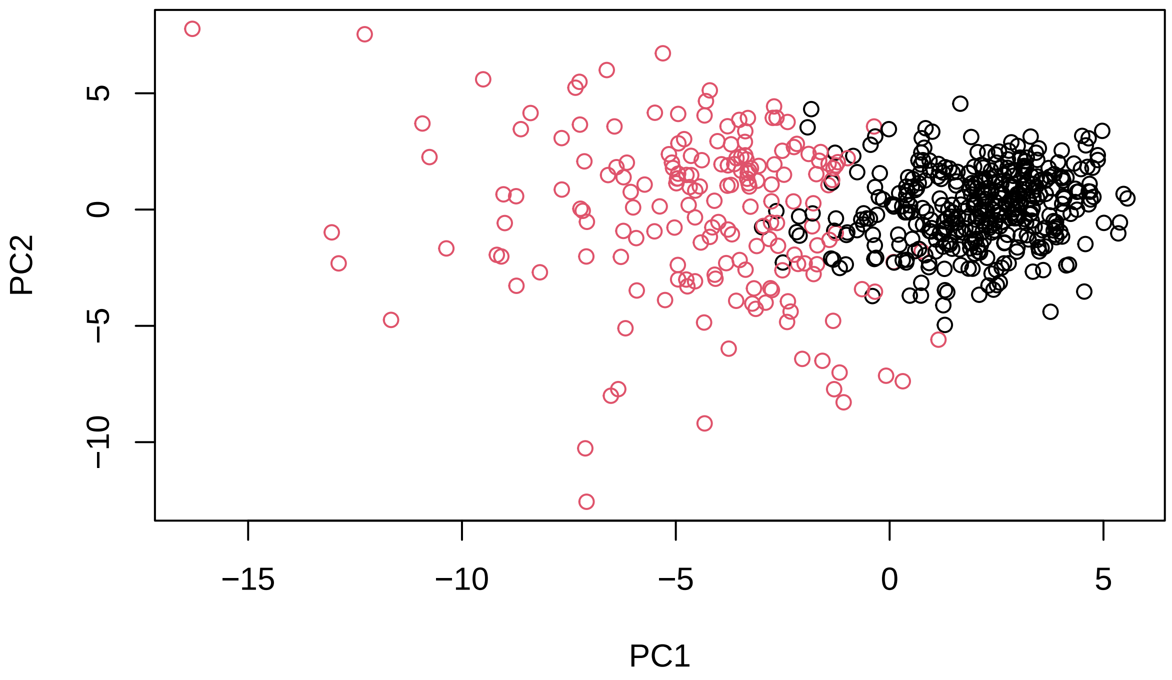
```
g <- as.factor(grps)
levels(g)
```

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

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

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

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



```r
# We 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")
```

```r
# We now cut the model into 2 clusters:
wisc.pr.hclust.clusters <- cutree(wisc.pr.hclust, k=2)
```

```r
# Now we compare the results 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?

The newly created model sorts out the two diagnoses relatively well, We can clearly see that cluster one is mostly “M” and cluster 2 is mostly “B”.

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
##   1 343  37
##   2  14 175
```

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

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

Answering question 16, k-means separates the diagnoses relatively well. We see that for k-means, cluster one is clearly mostly "B" and cluster 2 is mostly "M". We see that in the hierarchical cluster table, clusters 1, 2, and 4 are equivalent to cluster 2 in the k-means table. Additionally cluster 3 in the appear of the hierarchical cluster results appear to be equivalent.

6. Sensitivty/Specificty

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

We analyze the specificity and sensitivity of the k-means method.

```
km.table <- table(wisc.km$cluster, diagnosis)
# We see that cluster that is predominantly malignant is cluster 2.
# So the sensitivity is:
round(km.table[1, "M"]/(km.table[1, "M"] + km.table[2, "M"]), digits=3)
```

```
## [1] 0.175
```

```
# We see that cluster that is predominantly benign is cluster 1.
# So the specificity is:
round(km.table[2, "B"]/(km.table[1, "B"] + km.table[2, "B"]), digits =3)
```

```
## [1] 0.039
```

We analyze the specificity and sensitivity of the hierarchical clustering method with complete linkages.

```
hclust.table <- table(wisc.hclust.clusters, diagnosis)
# We see that cluster that is predominantly malignant is cluster 1.
# So the sensitivity is:
round(hclust.table[1, "M"]/(hclust.table[1, "M"] + hclust.table[2, "M"] +
                          hclust.table[3, "M"] + hclust.table[4, "M"]),
      digits=3)
```

```
## [1] 0.778
```

```
# We see that cluster that is predominantly benign is cluster 3.
# So the specificity is:
round(hclust.table[3, "B"]/(hclust.table[1, "B"] + hclust.table[2, "B"] +
                           hclust.table[3, "B"] + hclust.table[4, "B"]),
      digits=3)
```

```
## [1] 0.961
```

We analyze the specificity and sensitivity of the hierarchical clustering method with ward.d2 linkages.

```
hclust2.table <- table(grps, diagnosis)
# We see that cluster that is predominantly malignant is cluster 1.
# So the sensitivity is:
round(hclust2.table[1, "M"]/(hclust2.table[1, "M"] + hclust2.table[2, "M"]),
      digits=3)
```

```
## [1] 0.774
```

```
# We see that cluster that is predominantly benign is cluster 2.
# So the specificity is:
round(hclust2.table[2, "B"]/(hclust2.table[1, "B"] + hclust2.table[2, "B"]),
      digits=3)
```

```
## [1] 0.944
```

To answer question 17, we see that the both the Hierarchical clustering with complete linkage and and k-means models have equal highest specificity which is slightly higher than those of the model for Hierarchical clustering with ward.d2 linkage. Additionally, the k-means model, has the greatest sensitivity, followed by the model of Hierarchical clustering with complete linkage, followed lastly by the model for Hierarchical clustering with ward.d2 linkage.

7. Prediction

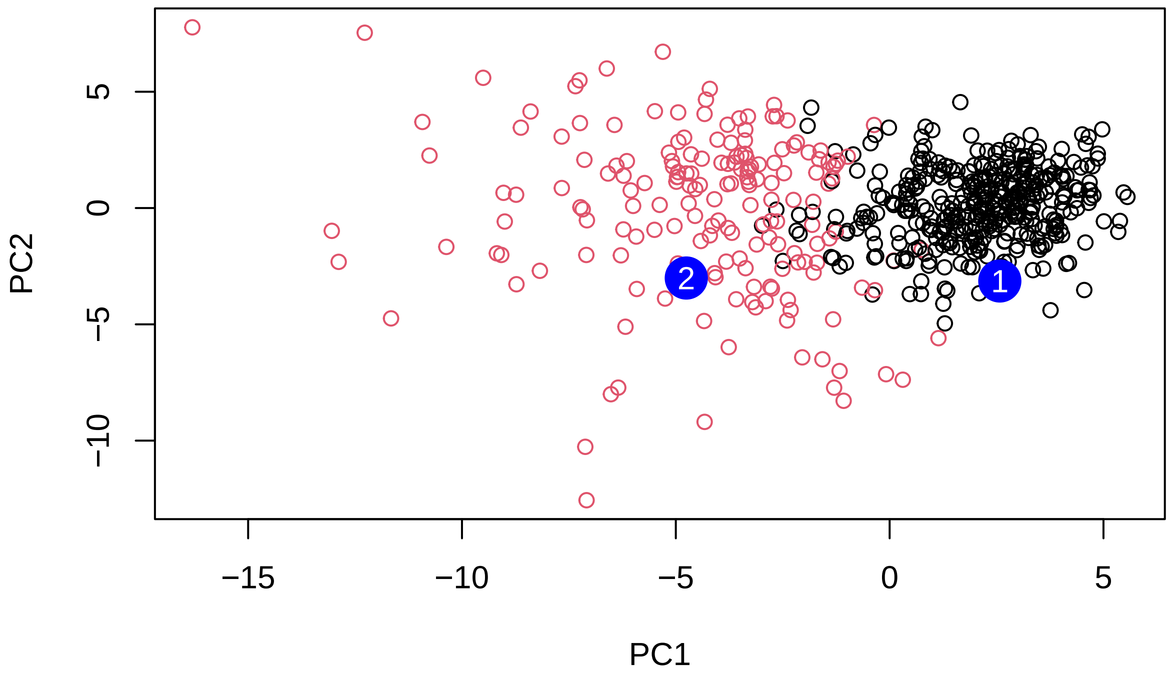
We will now use the predict() function that will take our PCA model from before and new cancer cell data and project that data onto our PCA space.

```
# 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
```

Now we plot this information.

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



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

We should prioritize the second patient for followup as their samples fall well within the red cluster which contain the more malignant cancerous samples.