Unsupervised Learning Analysis of Human Breast Cancer Cells

Yash Garodia

10/02/2022

First we use the read.csv function to import the data.

```
#importing the data

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

#head(wisc.df)
#View(wisc.df)
```

But we don't want the first column as they contain answers!

```
#to get rid of the first column
wisc.data <- wisc.df[,-1]
```

Now we setup a new vector called diagnosis that contains the data from the diagnosis column of the original dataset!

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

1.Exploratory Data Analysis

Q1) How many observations are in this dataset?

```
#Number of rows are the number of observations
nrow(wisc.data)
```

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

There are 569 observations in this dataset

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

```
#grouping the malignant diagnosis results in one variable
malignant <- grep("M", diagnosis)
length(malignant)
```

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

There are 212 malignant observations.

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

```
#First we group the headers containing the string "_mean"
var.mean <- grep("_mean", colnames(wisc.data))
#Then we count the number of values in the group
length(var.mean)
```

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

There are 10 variables suffixed with "_mean".

2. Principal Component Analysis

#Performing PCA

First we need to calculate the mean and standard deviations of wisc.data to determine if the data should be scaled.

```
#checking column means and sd values
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
```

```
#executing pca
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
```

Scaling clearly shows that other PCs apart from PC1 also capture significant levels of the original variance.

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

As seen in the proportion of variance section for PC1 of the summary, 0.4427 or 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?

As seen in the cumulative proportion section in the summary, 3 principal components are required to describe at least 70% of the original variance in the data.

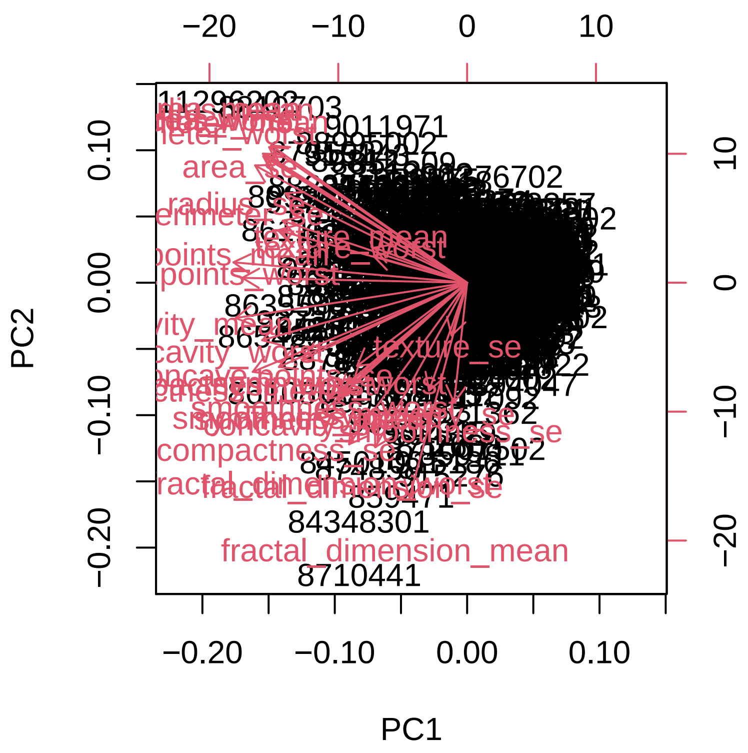
> Q6) How many principal components (PCs) are required to describe at least 90% of the original variance in the data?

As seen in the cumulative proportion section in the summary, 7 principal components are required to describe at least 90% of the original variance in the data.

#Interpreting PCA results

First, we create a biplot:

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

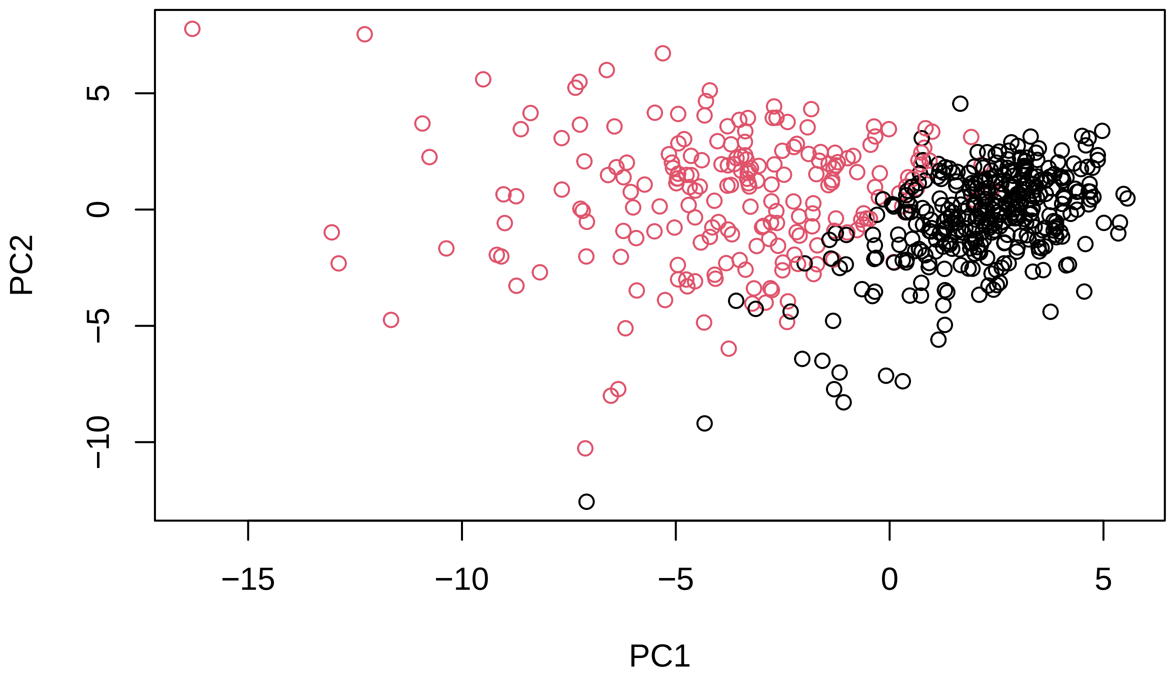


> Q7) What stands out to you about this plot? Is it easy or difficult to understand? Why?

This plot is very difficult to understand as everything in the plot is cluttered, making it hard to read any of the data. What stands out to me about this plot is that this scatter plot doesn't have any points clearly shown, but rather the label for each value represents the point, making it harder to read the values.

We can generate a more comprehensible standard scatter plot using this function:

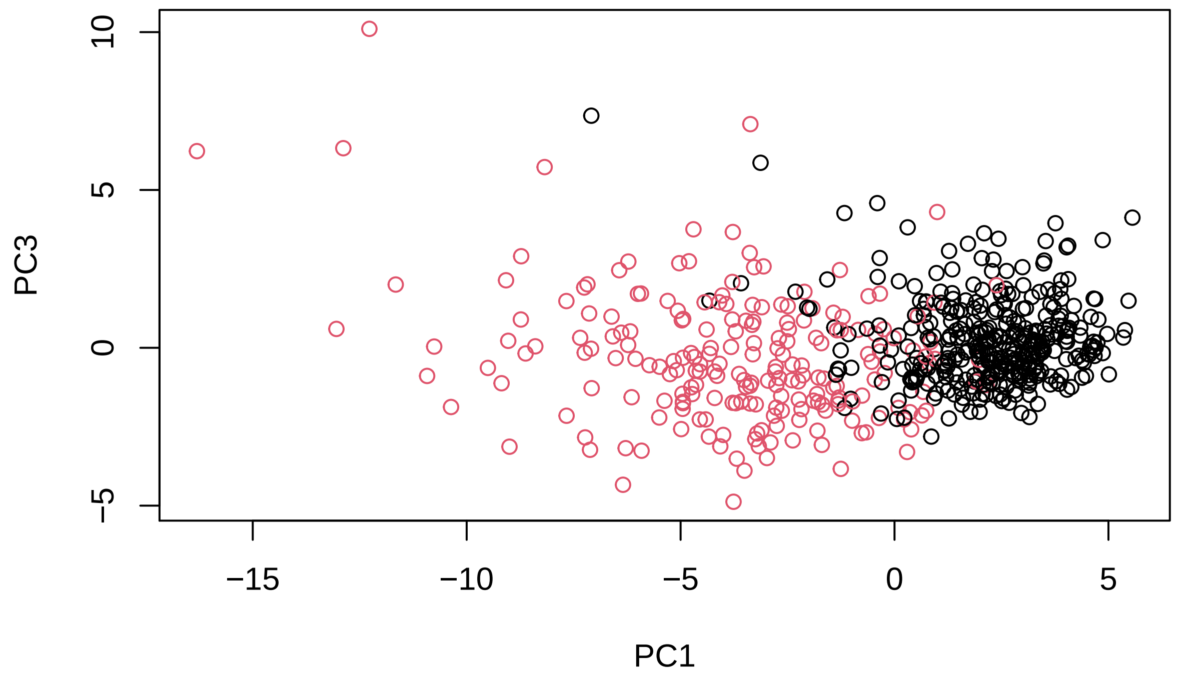
```
# Scatter plot observations by components 1 and 2
plot(wisc.pr$x[,1], wisc.pr$x[,2], col = diagnosis, xlab = "PC1", ylab = "PC2")
```



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

To do this we repeat the function used above, replacing the code for plotting PC2 with PC3.

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



One thing I notice about both these plots is that they both show that there is a separation between the results for benign and maligned samples. Malignant samples are more spread out than the benign samples, suggesting that they have greater variation in results.

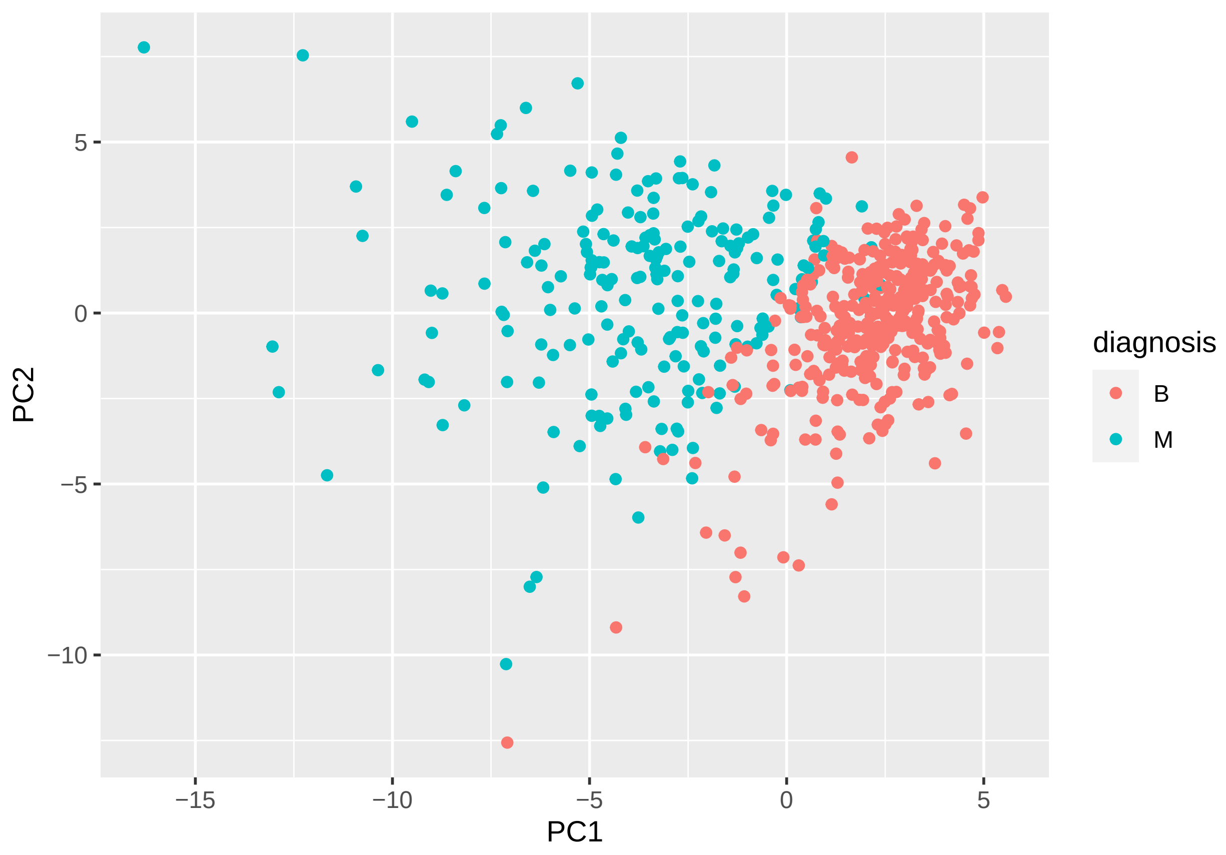
Lets see if we can use ggplot2 to make fancy figures of the same:

```
# Create a data.frame for ggplot
df <- as.data.frame(wisc.pr$x)
df$diagnosis <- diagnosis

#load the ggplot2 package
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
#make a scatter plot colored by diagnosis
ggplot(df) + aes(PC1, PC2, col = diagnosis) + geom_point()
```



#Variance Explained

First, we calculate variance of each principal component.

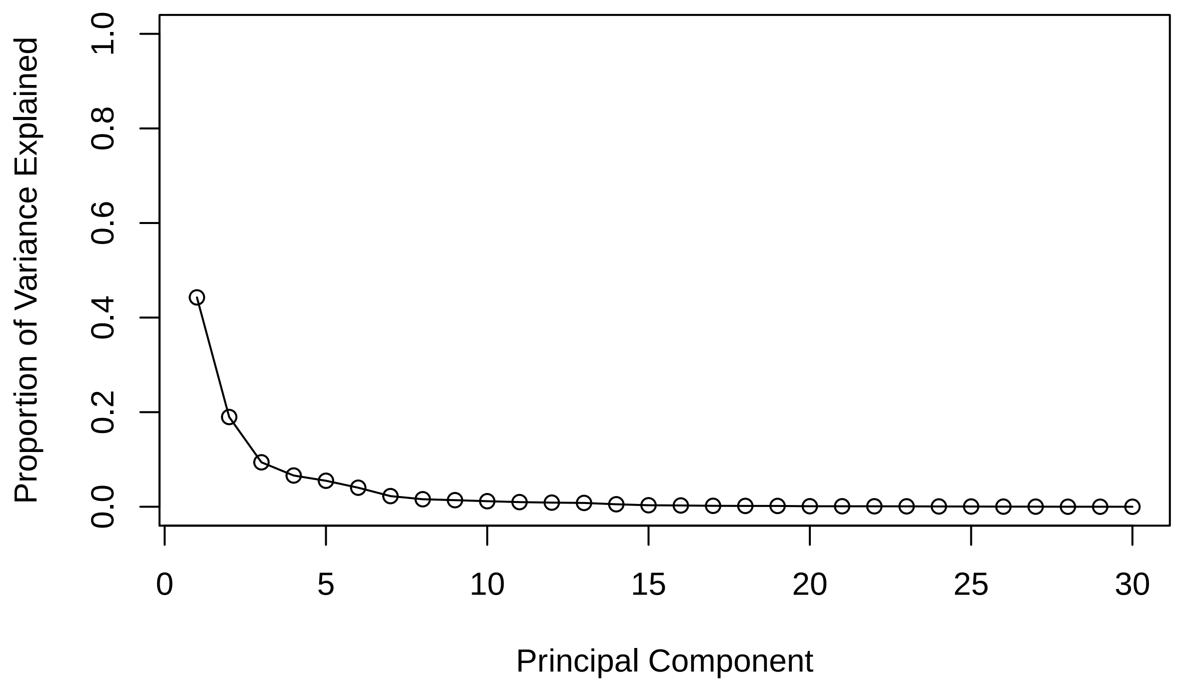
```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

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

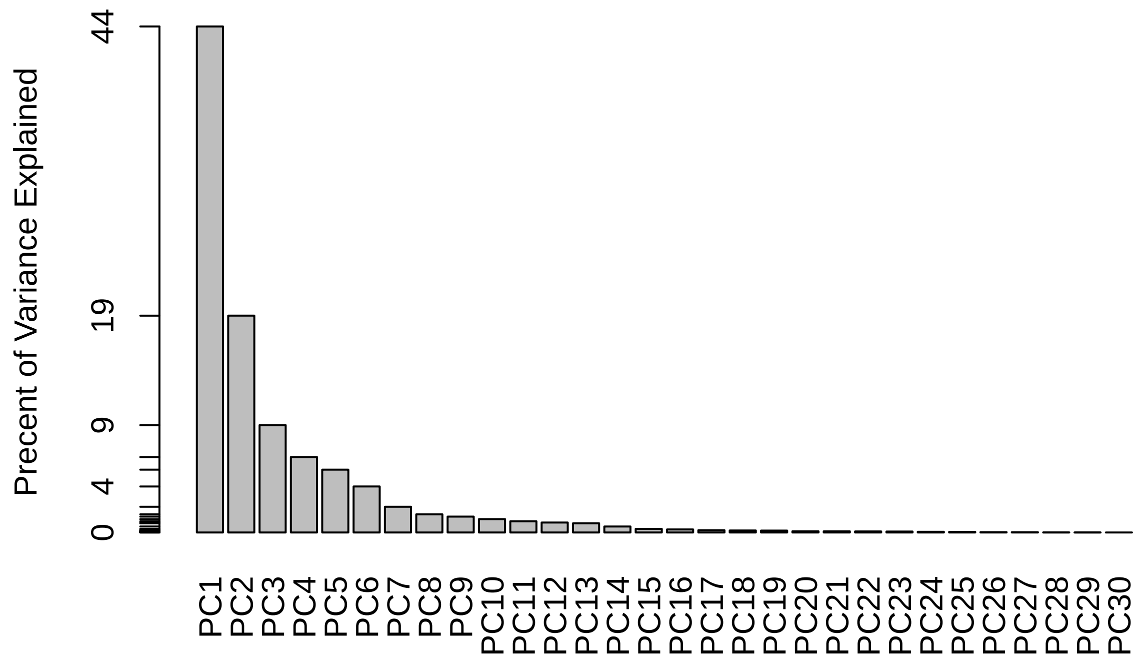
Then, to calculate the total variance explained by each principal component, we divide the variance of each component by the total variance.

```
# 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 )
```



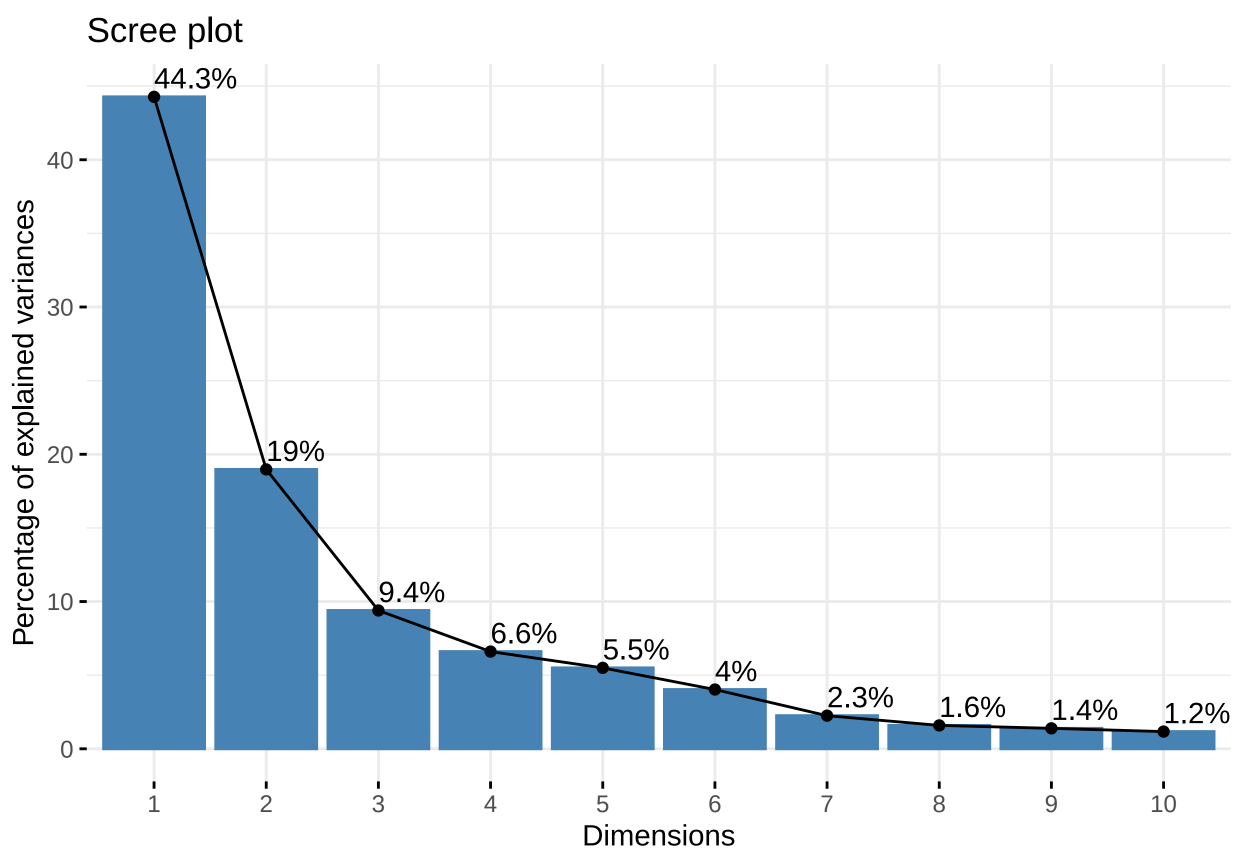
There are quite a few CRAN packages that are helpful for PCA. This includes the factoextra package. Feel free to explore this package. For example:

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

```
## Warning: package 'factoextra' was built under R version 4.0.2
```

```
## 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
```

The value for the component of the loading vector for PC1 for concave.points_mean is -0.26085376.

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

Looking at the barplot, a minimum of 5 principal components are required to explain 80% of the variance of the data.

3. Hierarchical Clustering

First we scale the wisc.data and assign it to data.scaled.

```
data.scaled <- scale(wisc.data)
```

Then we calculate the euclidean distances between all pairs of observations and assign it to data.dist

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

Finally, we create a hierarchical clustering model using a complete linkage.

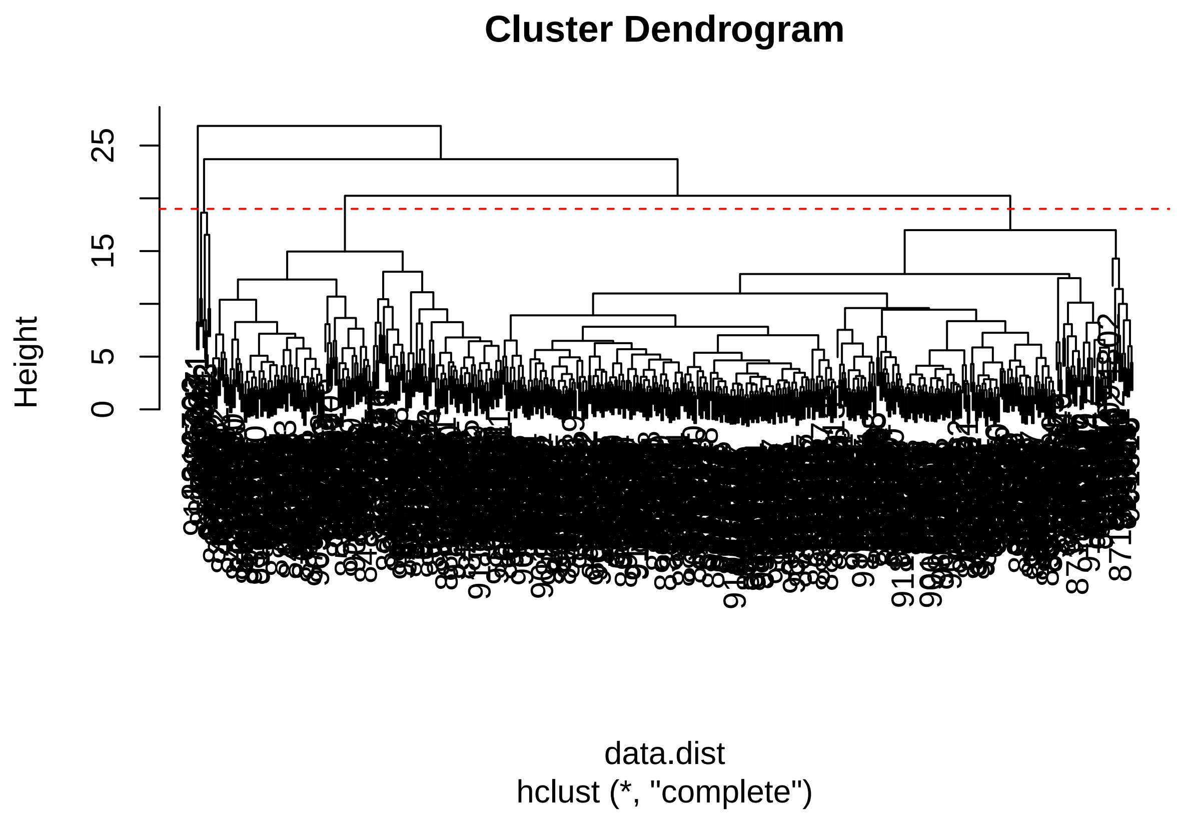
```
wisc.hclust <- hclust(data.dist, method = "complete")
```

Results of Hierarchical Clustering

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

The height at which the clustering model has 4 clusters is 20.

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



#Selecting number of clusters

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

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

Then using the table() function we can 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
```

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

Lets try having the number of clusters to equal 2:

```
wisc.hclust.clusters1 <- cutree(wisc.hclust, k = 2)
table(wisc.hclust.clusters1, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters1   B   M
##                     1 357 210
##                     2   0   2
```

Here we see that cluster 1 has more benign cases than malignant, but cluster 1 only has malignant cases.

On the other hand, if we have the number of cases equal to 6:

```
wisc.hclust.clusters2 <- cutree(wisc.hclust, k = 5)
table(wisc.hclust.clusters2, diagnosis)
```

```
##                        diagnosis
## wisc.hclust.clusters2   B   M
##                     1  12 165
##                     2   0   5
##                     3 343  40
##                     4   2   0
##                     5   0   2
```

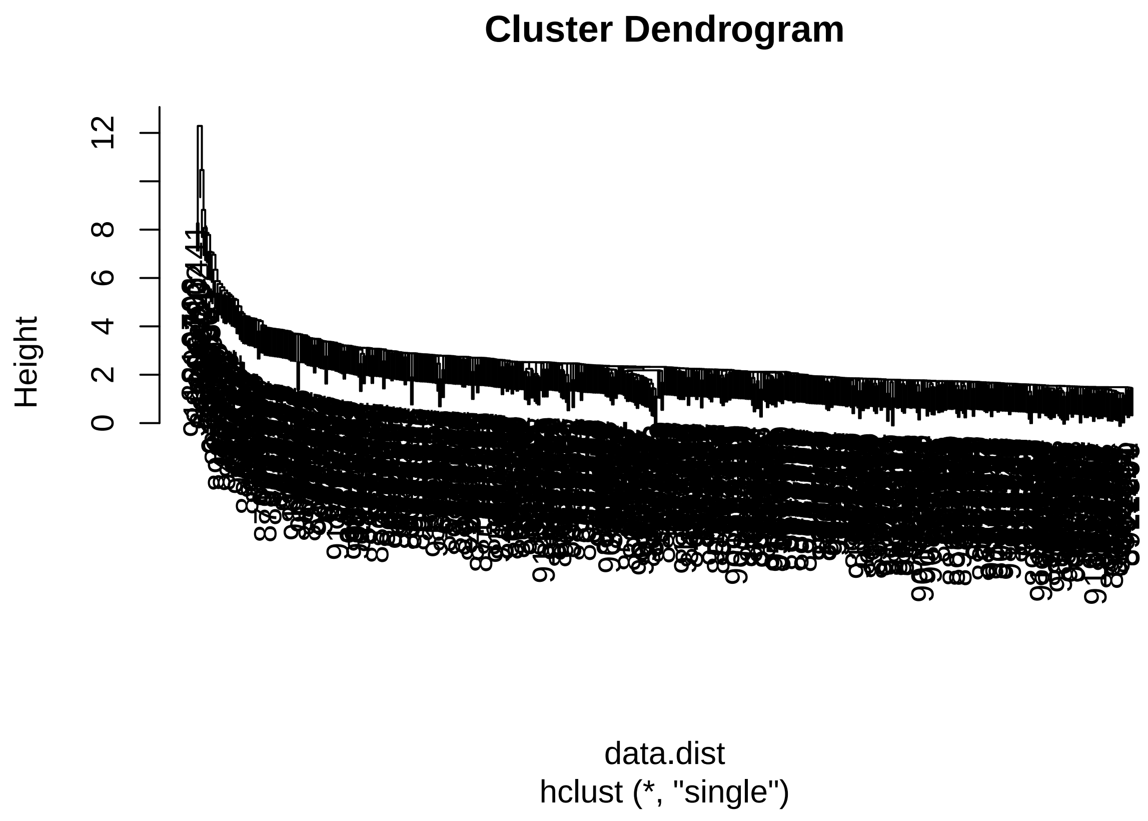
The cluster vs diagnosis match is clearer now. The system works in such a way that clusters with higher benign cases will have very few malignant cases and vice versa. This strongly suggests that there are significant differences between benign and malignant cases.

#Using different methods

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

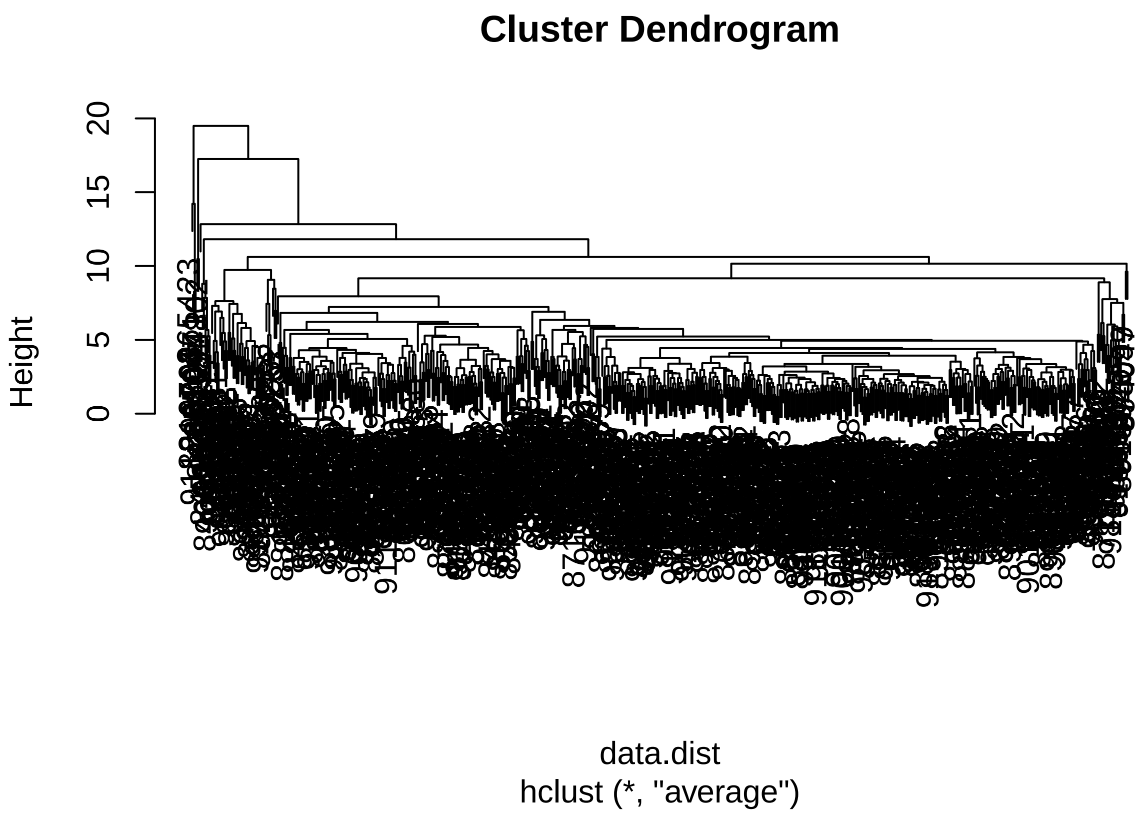
Since ward.D2 is professor's favorite, lets try this one out first:

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



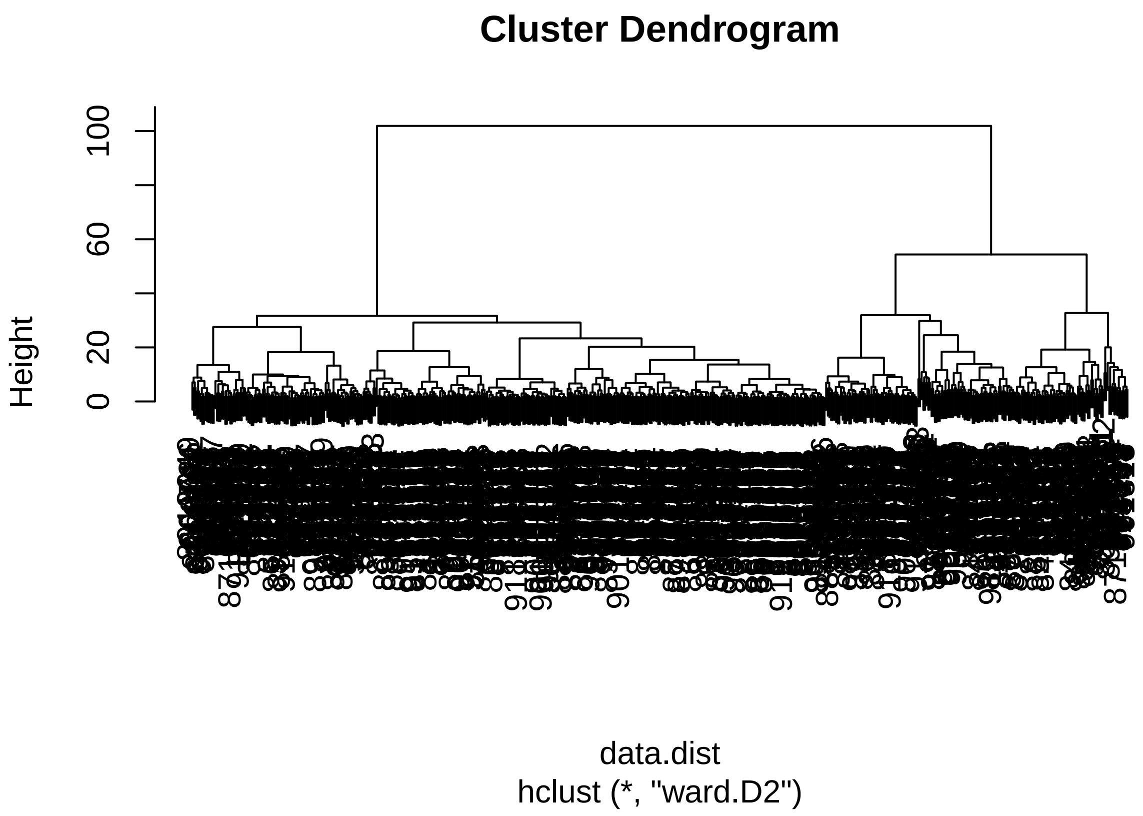
This is pretty cool, looks like a decay curve. Lets try average.

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



EEK this looks kinda disturbing. Lets try professors favorite, ward.d2

```r
wisc.hclust.ward.d2<- hclust(data.dist, method = "ward.D2")
plot(wisc.hclust.ward.d2)
```



Hm. its a tough choice between ward.D2 and single. I think I like single more because it is unique and the fact that the clusters are in a curved type layout make it easier to identify the common ancestors. It is easier to gauge the level of similarity between observations this way.

K-means clustering

K-means clustering and comparing results

Lets create a k-means model on wisc.data, scaling and creating two clusters corresponding to the 2 diagnoses, with the algorithm repeated 20 times:

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

Then we can 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  14 175
##   2 343  37
```

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

Clearly we can see that kmeans clustering is giving a clearer separation versus hclust clustering. In cluster 1 for hclust the difference between malignant and benign observations isn't as much, whereas there is a stark contrast in cases for kmer clustering. In cluster 1 there are much more maligned cases for kmer clustering and in cluster 2 there are much more benign cases.

Lets bring the result for hclust here so we can compare:

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

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

As we can see here, 3 clusters of hclust (1,2,4) are equivalent to one cluster of kmeans and the other cluster is equivalent to the other kmeans cluster. Kmeans was very good at separating the diagnoses, and was better.

5.Combining methods

Clustering PCA results

> Using the minimum number of principal components required to describe at least 90% of the variability in the data, lets create a hierarchical clustering model with the linkage method="ward.D2" and assign it to wisc.pr.hclust.

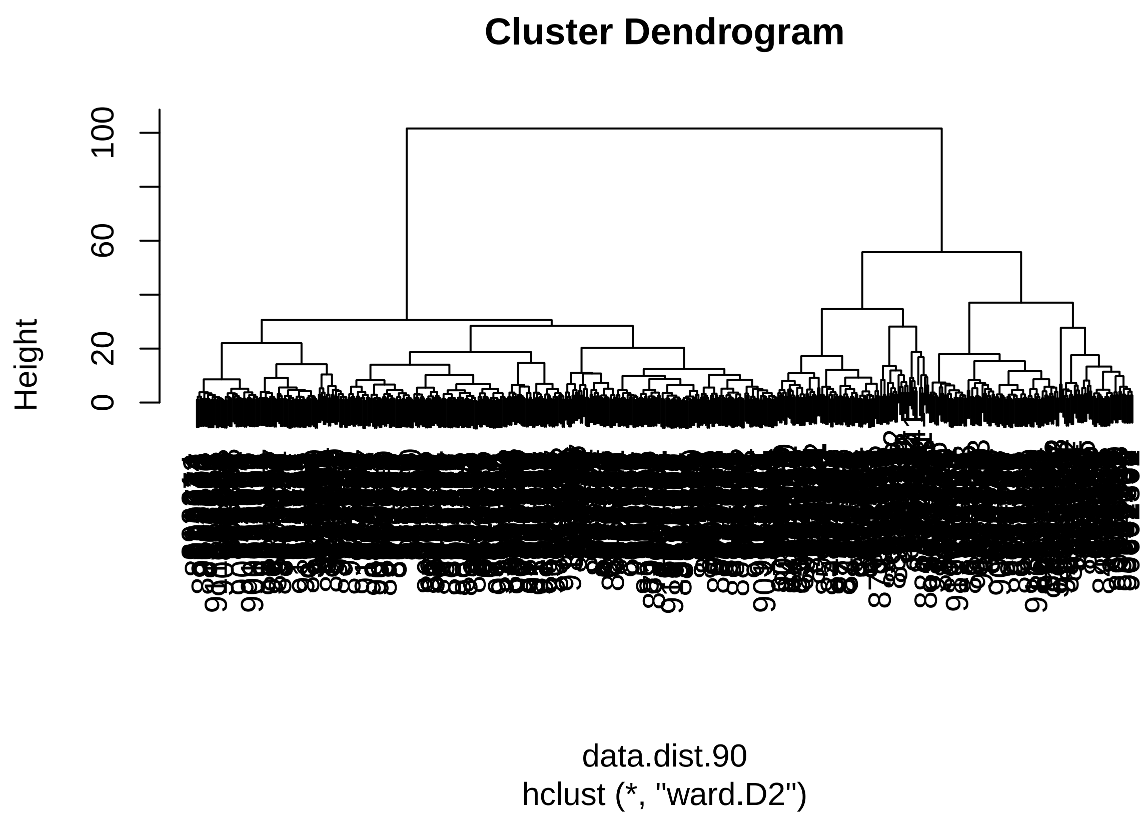
In order to answer this question, we must look at the proportion of variability that each PC covers, and the cumulative variability. In order to do this we need to take a look at the summary of running principal components analysis on wisc.data.

```
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
```

Through this, we see that the 7 PCs are needed to cover a cumulative proportion of 90%. Using this, we can now create the hierarchical clustering model:

```
data.dist.90 <- dist(wisc.pr$x[,1:7])
wisc.pr.hclust <- hclust(data.dist.90, method = "ward.D2" )
plot(wisc.pr.hclust)
```



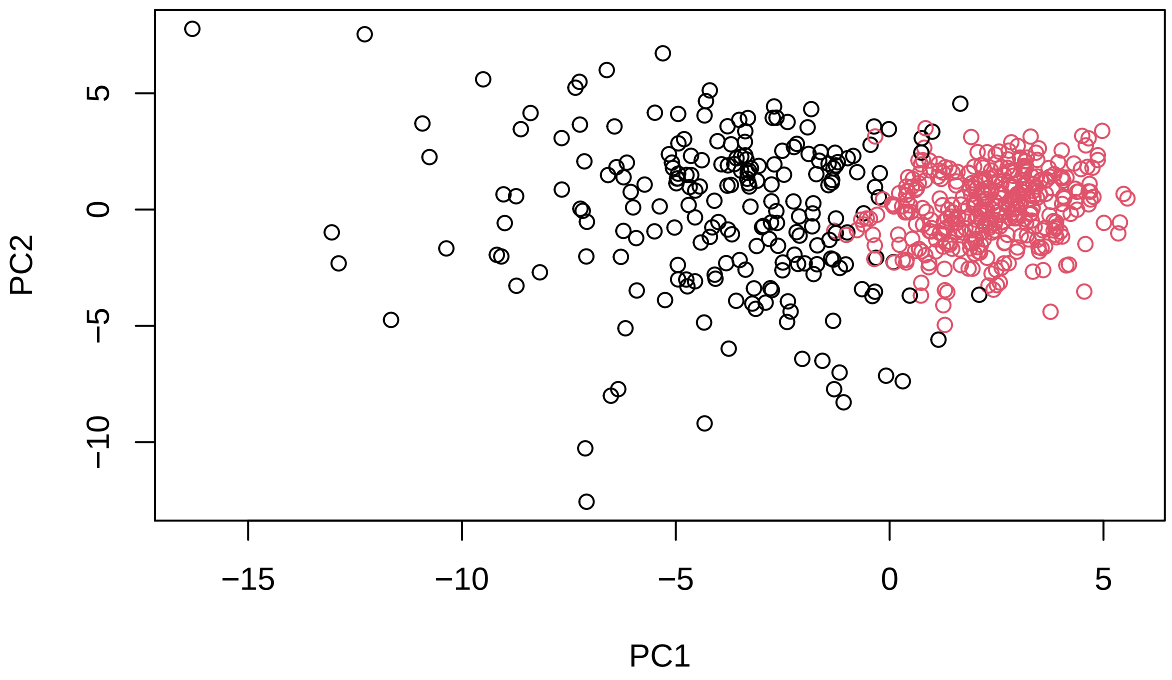
```
grps <- cutree(wisc.pr.hclust, k=2)
table(grps)
```

```
## grps
##   1   2
## 216 353
```

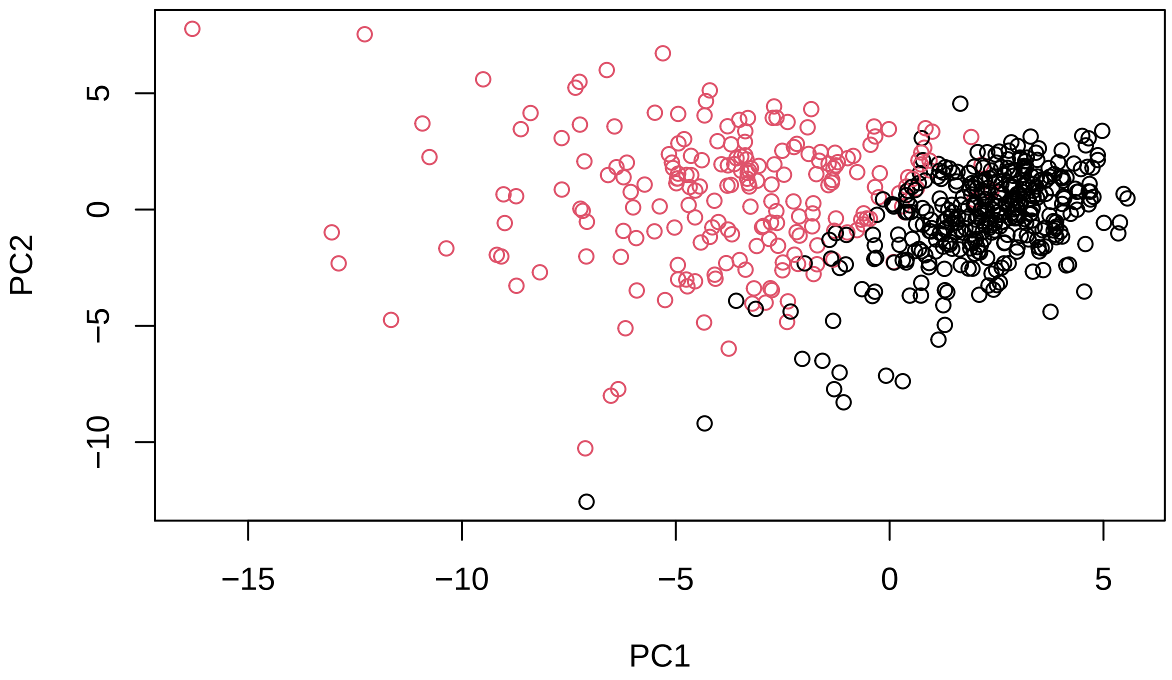
```
table(grps, diagnosis)
```

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

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



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



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).

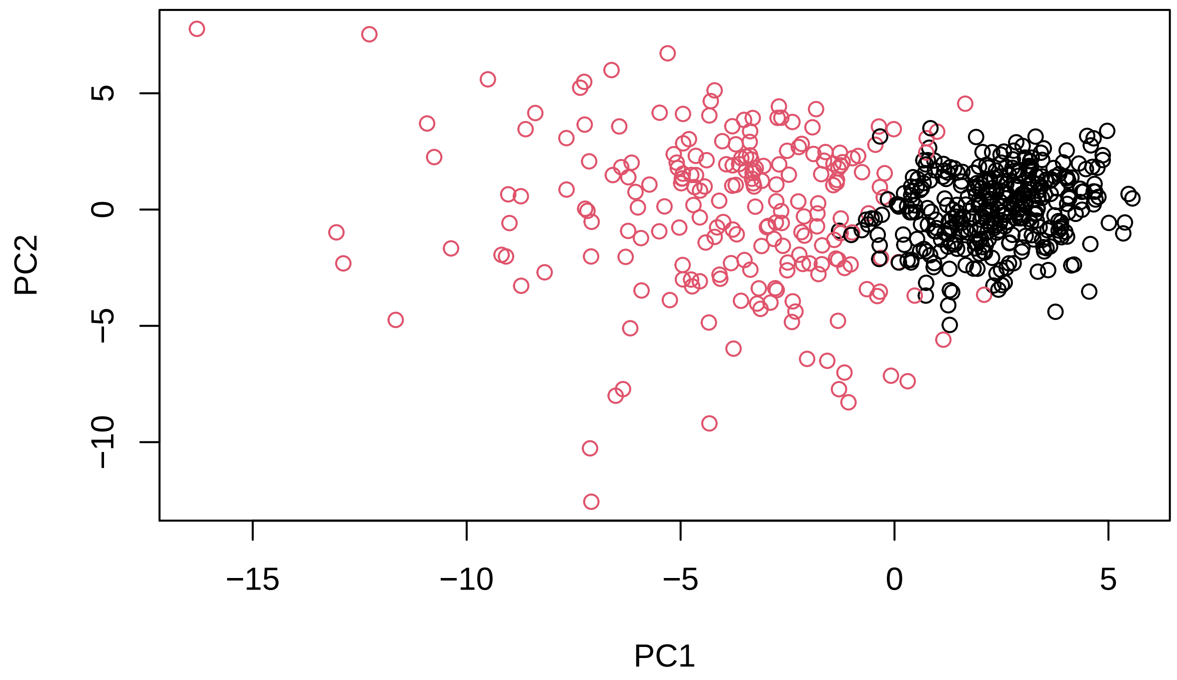
```
g <- as.factor(grps)
levels(g)
```

```
## [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)
```



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.

```
library(rgl)
```

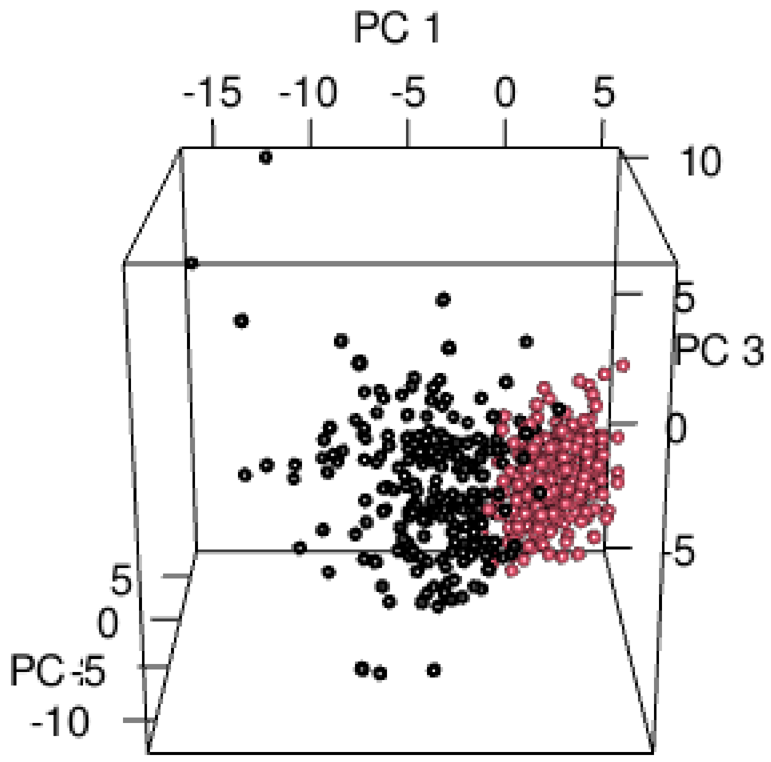
```
## Warning: package 'rgl' was built under R version 4.0.2
```

```
plot3d(wisc.pr$x[,1:3], xlab="PC 1", ylab="PC 2", zlab="PC 3", cex=1.5, size=1, type="s", col=grps)
```

To include the interactive rgl plot in your HTML renderd lab report (not PDF) you can add the R code rglwidget(width = 400, height = 400) after you call the plot3d() function. It will look just like the plot above. Try rotating and zooming on this 3D plot.

```
plot3d(wisc.pr$x[,1:3], xlab="PC 1", ylab="PC 2", zlab="PC 3", cex=1.5, size=1, type="s", col=grps)
rglwidget(width = 400, height = 400)
```

```
## Warning in snapshot3d(scene = x, width = width, height = height): webshot = TRUE
## requires the webshot2 package; using rgl.snapshot() instead
```



```
## Use the distance along the first 7 PCs for clustering i.e. wisc.pr$x[, 1:7]
wisc.pr.hclust <- hclust(data.dist.90, 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)
```

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

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

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

It separates the diagnoses pretty well and in a way that one cluster has more maligned results (cluster 1) while the other has more benign results (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.

```
#kmeans clustering
table(wisc.km$cluster, diagnosis)
```

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

```
#hclust from previous section
table(wisc.hclust.clusters, diagnosis)
```

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

The kmeans clustering does a good job as it separates the diagnoses in a way that one cluster has more maligned results (cluster 1) while the other has more benign results (cluster 2).

The hclust clustering shows that cluster 1,2 and 4 have more maligned cases while 3 has significantly more benign cases. I think kmeans clustering does a better job though.

6. Sensitivity/Specificity

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).

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

There are a total of 212 malignant and 357 benign cancer results. Based on the results, its clear that kmer clustering and the hclustering on the original data were the most specific as they identified the highest proportion of benign samples, with 343 out of 357 being identified in one cluster. The most sensitive analytical model was the newly created hclust model with 2 clusters as it identified the most maligned cases in the maligned cluster, that is 188 cases.

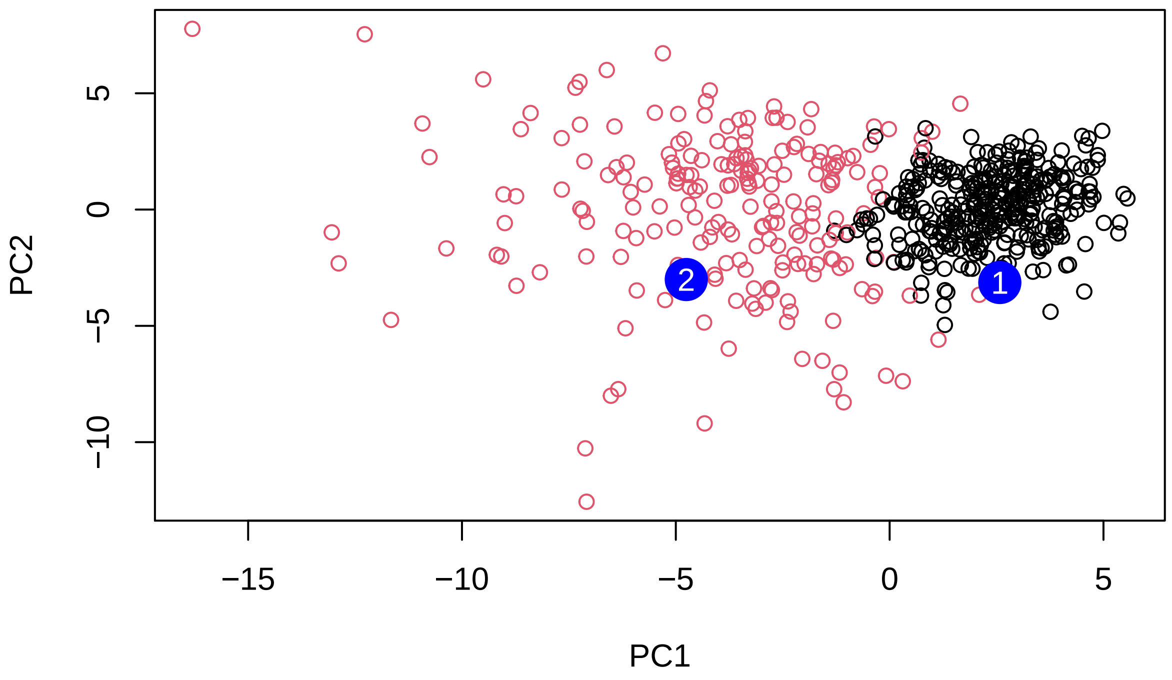
7. Prediction

We will 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(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")
```



The greater range and magnitude of PC values taken by patients belonging to section 2 rather than 1 suggests that it has a greater influence on PC1 and 2, and hence is most likely to include conditions and factors that scientists are yet to research on. Therefore, patient 2 should be prioritized for follow up.

```
sessionInfo()
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS  10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] rgl_0.108.3      factoextra_1.0.7 ggplot2_3.3.5
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_1.1.1  xfun_0.29         purrr_0.3.4       carData_3.0-5
##  [5] colorspace_2.0-2  vctrs_0.3.8       generics_0.1.0    htmltools_0.5.2
##  [9] yaml_2.2.1        utf8_1.2.2        rlang_0.4.11      pillar_1.6.3
## [13] ggpubr_0.4.0      glue_1.4.2        withr_2.4.2       DBI_1.1.1
## [17] lifecycle_1.0.1   stringr_1.4.0     munsell_0.5.0     ggsignif_0.6.3
## [21] gtable_0.3.0      htmlwidgets_1.5.4 evaluate_0.14     labeling_0.4.2
## [25] knitr_1.36        fastmap_1.1.0     fansi_0.5.0       highr_0.9
## [29] broom_0.7.12      Rcpp_1.0.7        scales_1.1.1      backports_1.2.1
## [33] jsonlite_1.7.2    abind_1.4-5       farver_2.1.0      png_0.1-7
## [37] digest_0.6.28     stringi_1.7.5     rstatix_0.7.0     dplyr_1.0.7
## [41] ggrepel_0.9.1     grid_4.0.0        tools_4.0.0       magrittr_2.0.1
## [45] tibble_3.1.5      crayon_1.4.1      tidyr_1.1.4       car_3.0-12
## [49] pkgconfig_2.0.3   ellipsis_0.3.2    assertthat_0.2.1  rmarkdown_2.11
## [53] R6_2.5.1          compiler_4.0.0
```