

class09

Stefanie Hodapp (PID: A53300084)

10/27/2021

```
fna.data <- "WisconsinCancer.csv"
```

Read in data and store as wisc.df

```
wisc.df <- read.csv(fna.data, row.names=1)  
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 X
## 842302      0.11890 NA
## 842517      0.08902 NA
## 84300903      0.08758 NA
## 84348301      0.17300 NA
## 84358402      0.07678 NA
## 843786      0.12440 NA
```

Remove first column

```
wisc.data <- wisc.df[,-1]
```

Create a diagnosis vector for later

```
diagnosis <- wisc.df[,1]
head(diagnosis)
```

```
## [1] "M" "M" "M" "M" "M" "M"
```

Q1. How many observations are in the wisc.data and diagnosis datasets?

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

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

```
length(diagnosis)
```

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

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

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

```
## diagnosis
##   B   M
## 357 212
```

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

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

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

Performing PCA

Check the means and standard deviations of the columns

```
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
##          X
##          NA
```

```
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
##          X
##          NA
```

```
ncol(wisc.data)
```

```
## [1] 31
```

```
wisc.data <- wisc.data[,-31]
head(wisc.data)
```

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

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

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

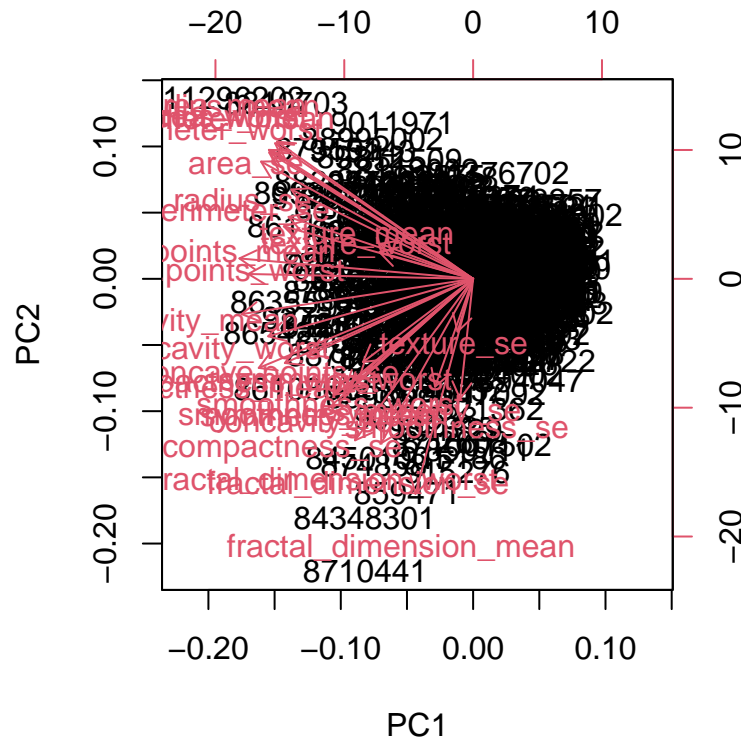
```
## Importance of components:
##      PC1      PC2      PC3      PC4      PC5      PC6      PC7
## Standard deviation 3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
## Cumulative Proportion 0.4427 0.6324 0.72636 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
```

Q4. From your results, what proportion of the original variance is captured by the first principal components (PC1)? 0.4427 or 44.27%

Q5. How many principal components (PCs) are required to describe at least 70% of the original variance in the data? 3

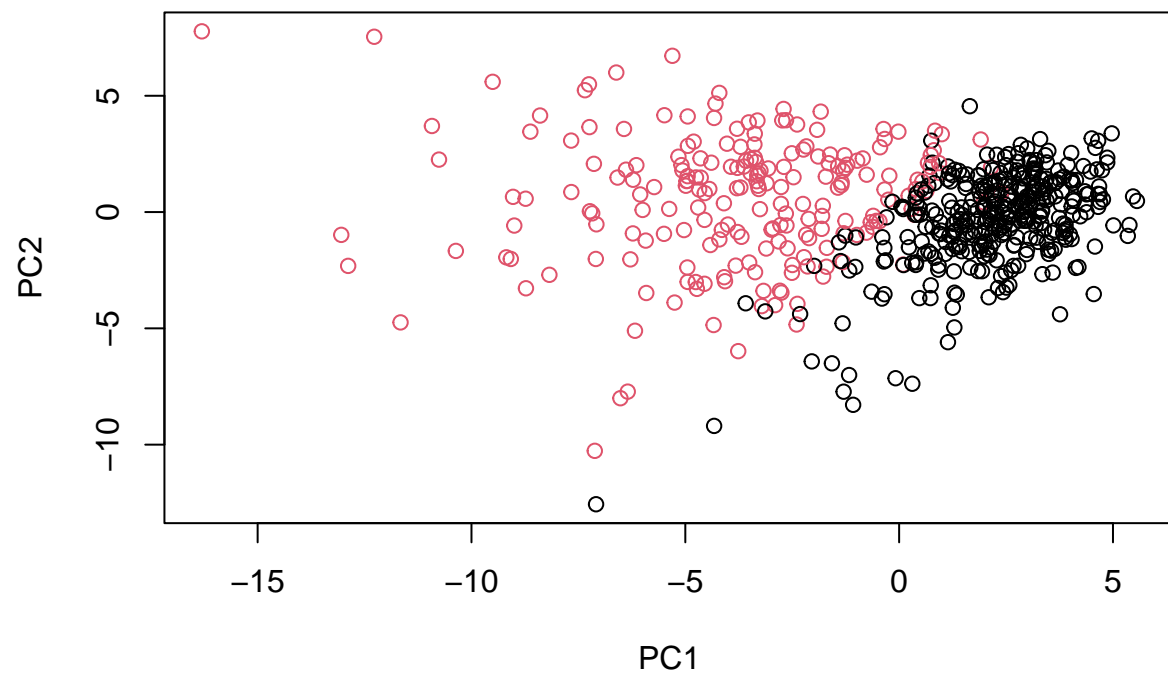
Q6. How many principal components (PCs) are required to describe at least 90% of the original variance in the data? 7

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

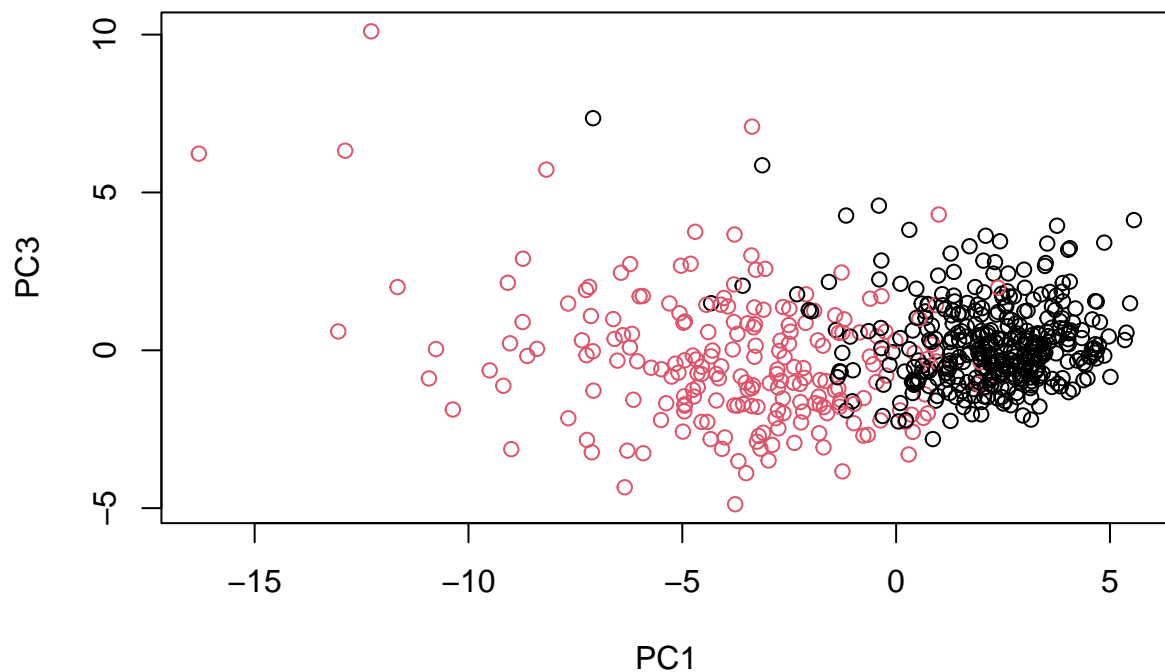


Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why? The quality of this plot is very poor and uninformative. The plot uses row names as the datapoints making it impossible to identify any trends or clusters in the data.

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



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



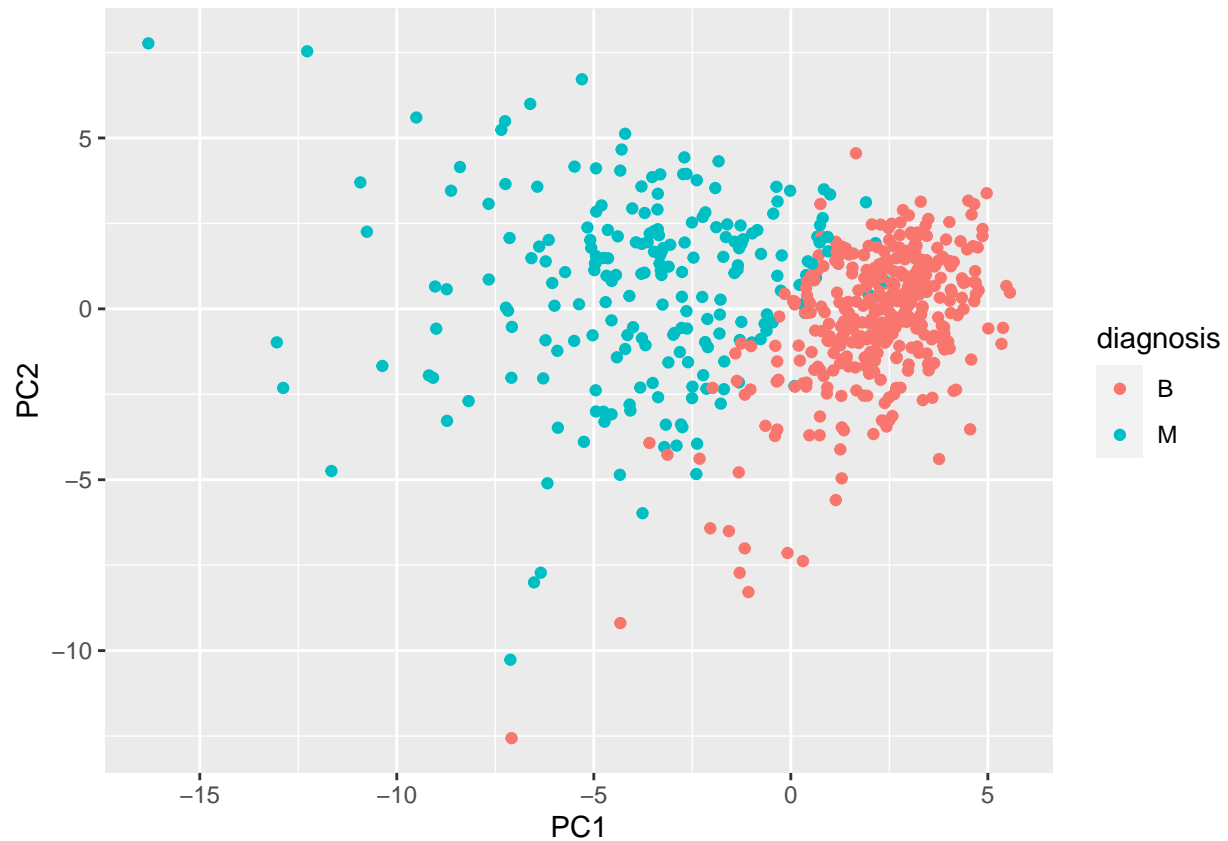
Q8. Generate a similar plot for principal components 1 and 3. What do you notice about these plots? There is better separation of the data in the first plot. This is because PC 2 explains for of the variance in the data than PC 3.

Make a plot using ggplot

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

# Load the ggplot2 package
library(ggplot2)

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

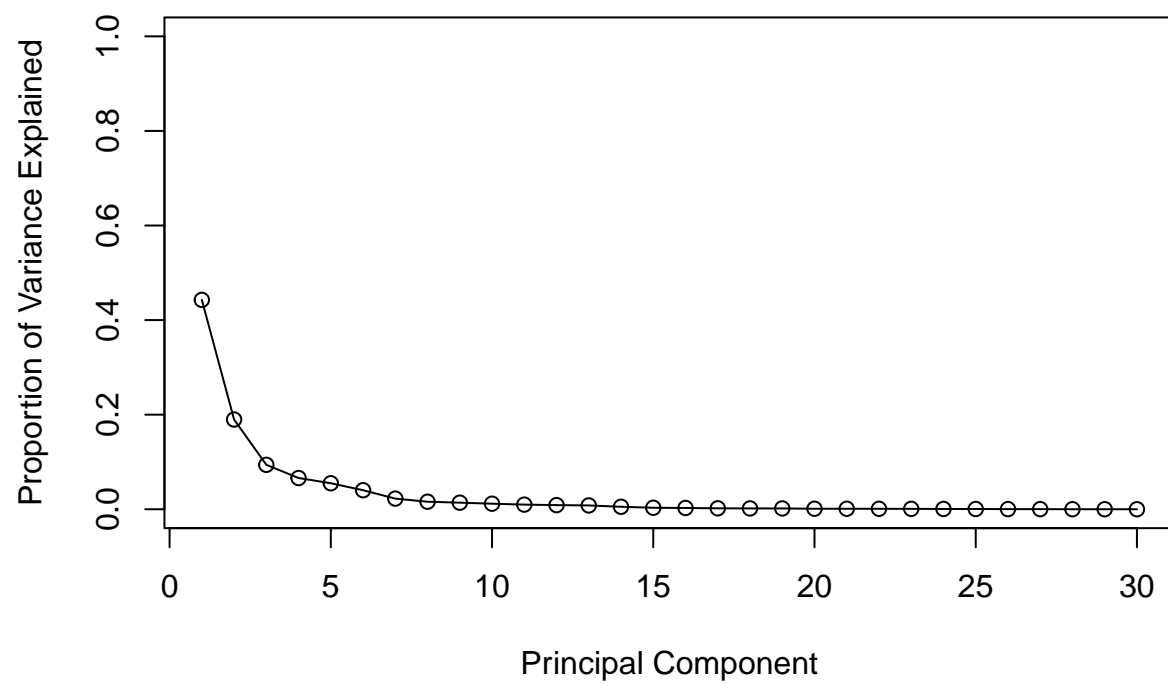
```
# Calculate variance of each component
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

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

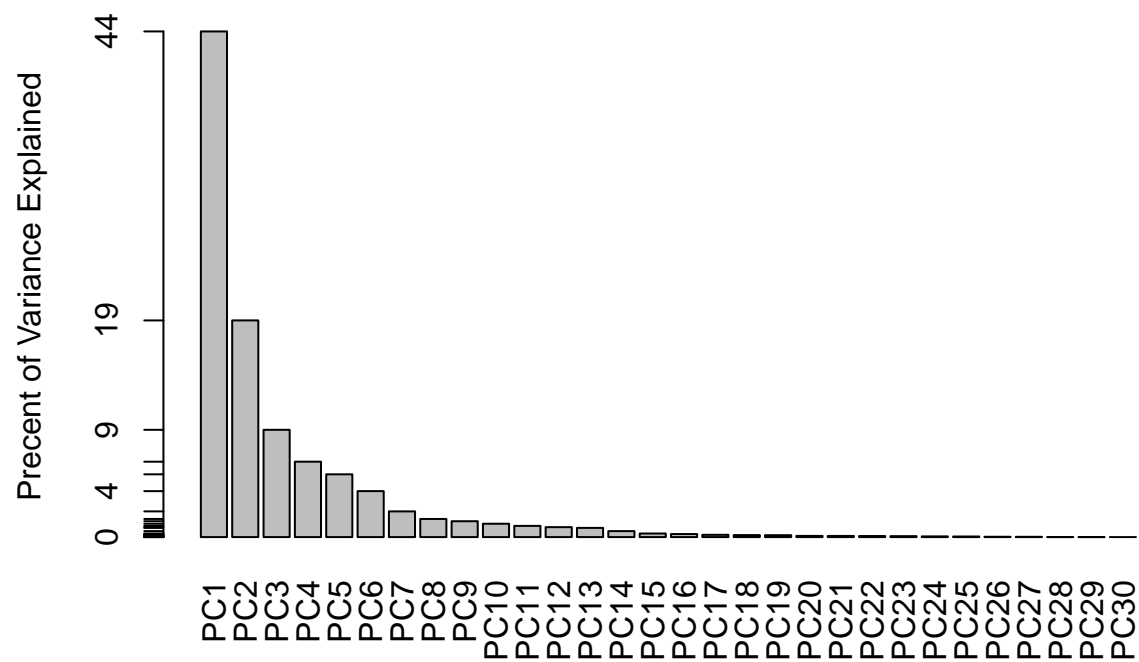
```
# Variance explained by each principal component: pve
pve <- pr.var / sum(pr.var)
print(pve)
```

```
## [1] 4.427203e-01 1.897118e-01 9.393163e-02 6.602135e-02 5.495768e-02
## [6] 4.024522e-02 2.250734e-02 1.588724e-02 1.389649e-02 1.168978e-02
## [11] 9.797190e-03 8.705379e-03 8.045250e-03 5.233657e-03 3.137832e-03
## [16] 2.662093e-03 1.979968e-03 1.753959e-03 1.649253e-03 1.038647e-03
## [21] 9.990965e-04 9.146468e-04 8.113613e-04 6.018336e-04 5.160424e-04
## [26] 2.725880e-04 2.300155e-04 5.297793e-05 2.496010e-05 4.434827e-06
```

```
# Plot variance explained for each principal component
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "o")
```



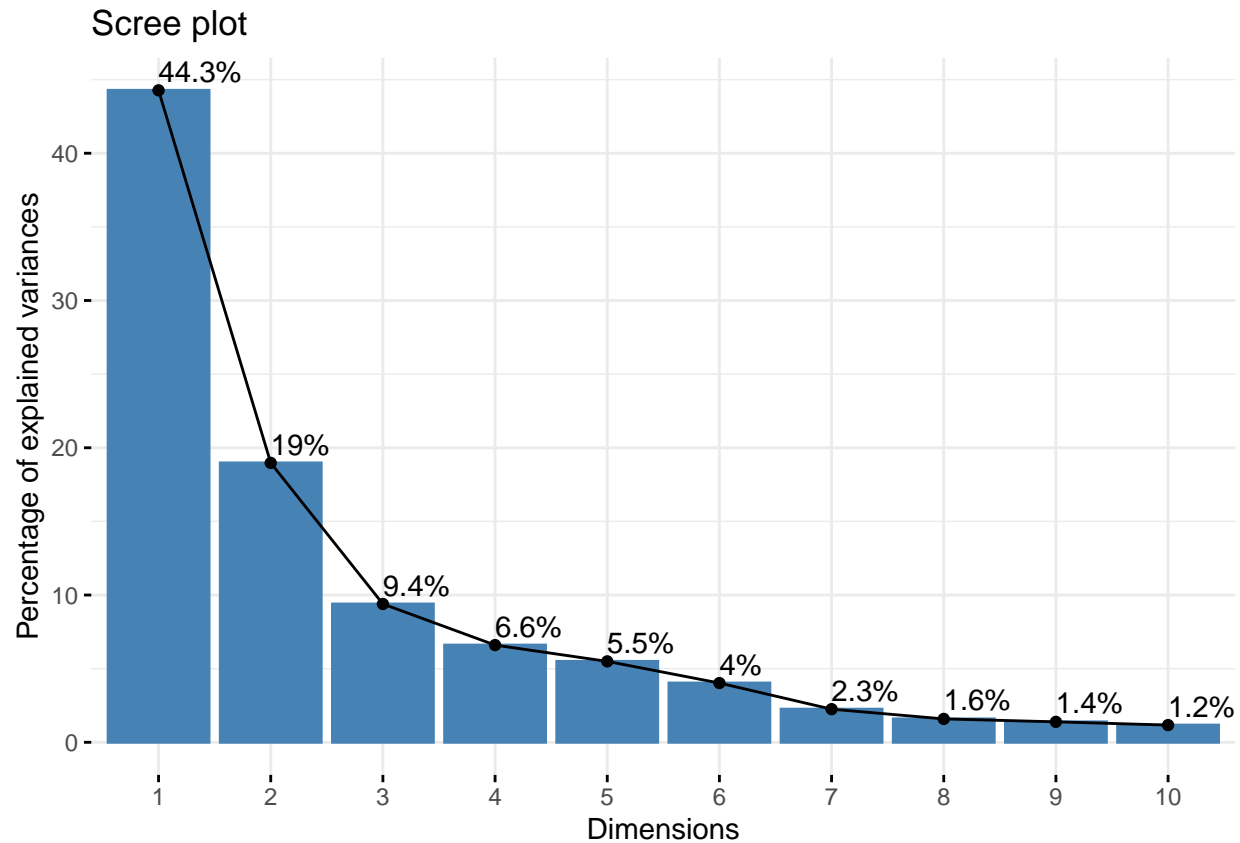
```
# Alternative scree plot of the same data, note data driven y-axis
barplot(pve, ylab = "Precent of Variance Explained",
        names.arg=paste0("PC",1:length(pve)), las=2, axes = FALSE)
axis(2, at=pve, labels=round(pve,2)*100 )
```



```
## ggplot based graph
library(factoextra)
```

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

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



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`? Answer:-0.26085376

```
wisc.pr$rotation[,1]
```

```
##          radius_mean          texture_mean          perimeter_mean
##          -0.21890244          -0.10372458          -0.22753729
##          area_mean          smoothness_mean          compactness_mean
##          -0.22099499          -0.14258969          -0.23928535
##          concavity_mean          concave.points_mean          symmetry_mean
##          -0.25840048          -0.26085376          -0.13816696
## fractal_dimension_mean          radius_se          texture_se
##          -0.06436335          -0.20597878          -0.01742803
##          perimeter_se          area_se          smoothness_se
##          -0.21132592          -0.20286964          -0.01453145
##          compactness_se          concavity_se          concave.points_se
##          -0.17039345          -0.15358979          -0.18341740
##          symmetry_se          fractal_dimension_se          radius_worst
##          -0.04249842          -0.10256832          -0.22799663
##          texture_worst          perimeter_worst          area_worst
##          -0.10446933          -0.23663968          -0.22487053
##          smoothness_worst          compactness_worst          concavity_worst
##          -0.12795256          -0.21009588          -0.22876753
##          concave.points_worst          symmetry_worst          fractal_dimension_worst
##          -0.25088597          -0.12290456          -0.13178394
```

Q10. What is the minimum number of principal components required to explain 80% of the variance of the data? 5 PCs

```
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
```

Hierarchical Clustering

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

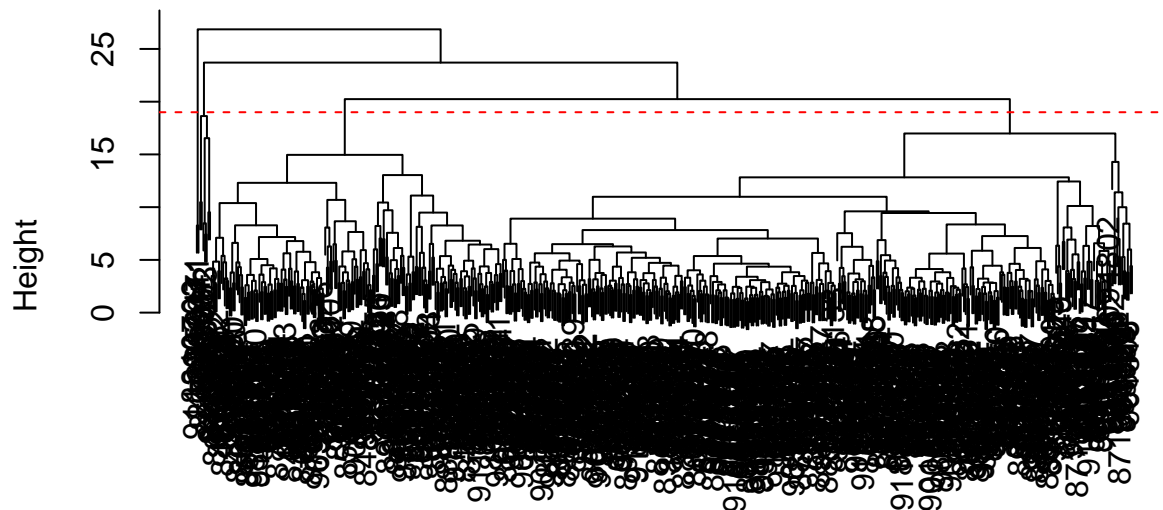
```
# Calculate the (Euclidean) distances between all pairs of observations in the new scaled dataset
data.dist <- dist(data.scaled)
```

```
# Create a hierarchical clustering model using complete linkage.
wisc.hclust <- hclust(data.dist, method = "complete")
```

Results of Hierarchical Clustering

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

Cluster Dendrogram



```
data.dist
hclust(*, "complete")
```

> Q11. Using the `plot()` and `abline()` functions, what is the height at which the clustering model has 4 clusters? `> height = 19`

Selecting number of clusters

```
# Use cutree() to cut the tree so that it has 4 clusters
wisc.hclust.clusters <- cutree(wisc.hclust, k=4)
```

```
# We can use the table() function to compare the cluster membership to the actual diagnoses.
table(wisc.hclust.clusters, diagnosis)
```

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

Q12. Can you find a better cluster vs diagnoses match by cutting into a different number of clusters between 2 and 10? A smaller number of clusters (i.e. 1-3) do not separate out the data well; both the benign and malignant datapoints fall within the same cluster. Increasing the number of clusters helps to separate the data more, but cutting into too many clusters (i.e. >7) doesn't provide additional useful information.

```

for (i in 2:10) {
  table <- table(cutree(wisc.hclust, k=i), diagnosis)
  print(table)
}

```

```

##      diagnosis
##      B  M
##  1 357 210
##  2   0   2
##      diagnosis
##      B  M
##  1 355 205
##  2   2   5
##  3   0   2
##      diagnosis
##      B  M
##  1  12 165
##  2   2   5
##  3 343  40
##  4   0   2
##      diagnosis
##      B  M
##  1  12 165
##  2   0   5
##  3 343  40
##  4   2   0
##  5   0   2
##      diagnosis
##      B  M
##  1  12 165
##  2   0   5
##  3 331  39
##  4   2   0
##  5  12   1
##  6   0   2
##      diagnosis
##      B  M
##  1  12 165
##  2   0   3
##  3 331  39
##  4   2   0
##  5  12   1
##  6   0   2
##  7   0   2
##      diagnosis
##      B  M
##  1  12  86
##  2   0  79
##  3   0   3
##  4 331  39
##  5   2   0
##  6  12   1
##  7   0   2

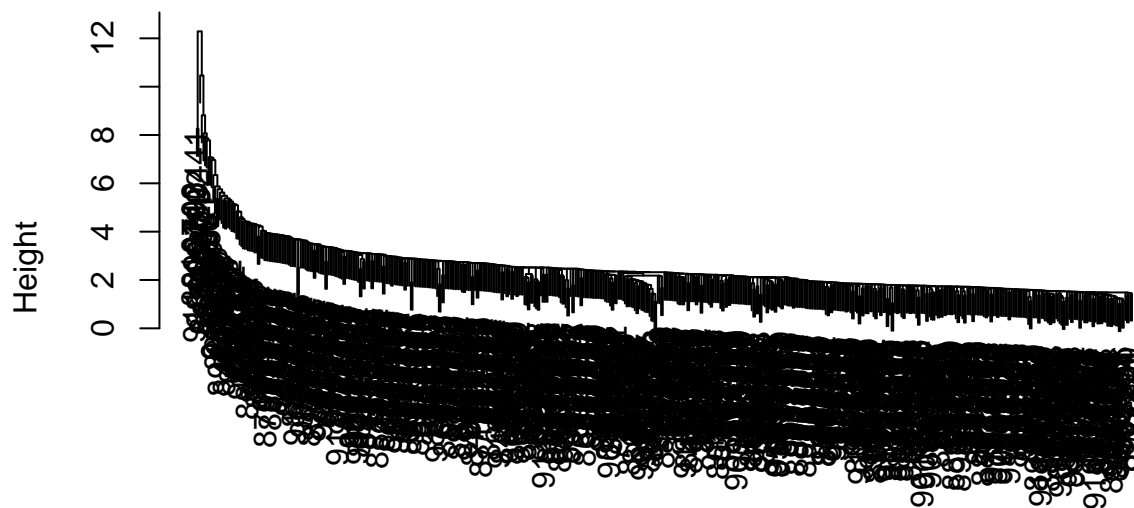
```

```
##      8    0    2
##      diagnosis
##           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
##      diagnosis
##           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
```

Q13. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning. My favorite result uses the ward.D2 method. I believe that this plot provides the most information, is the easiest to understand, and is the most visually pleasing. This is in comparison to the “single” method which doesn’t provided any clustering, the “average” method in which the clusters are hard to identify, and the “complete” method where the data are messier/squished together.

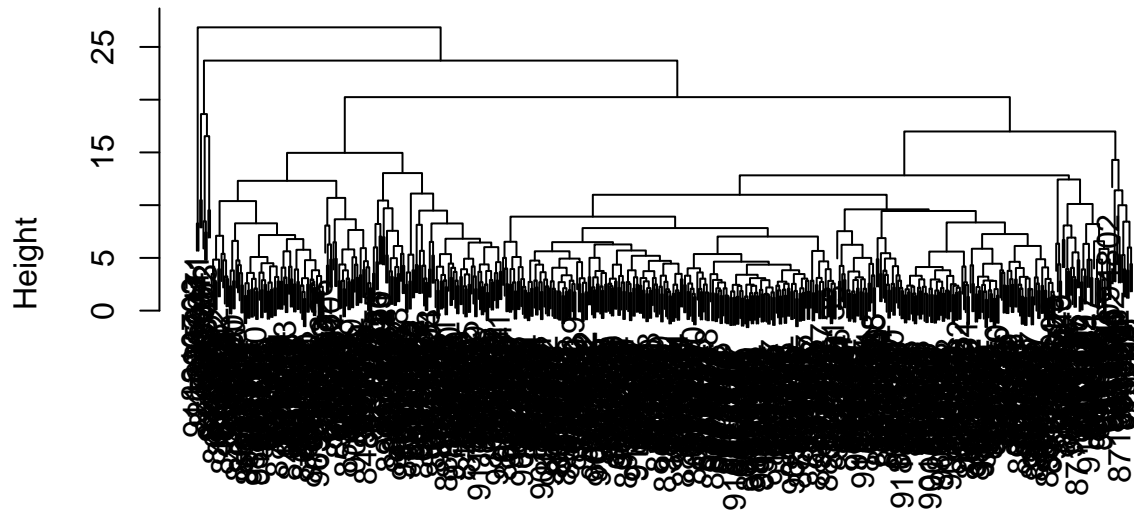
```
for (i in c("single", "complete", "average", "ward.D2")) {
  wisc.hclust <- hclust(data.dist, method = i)
  plot(wisc.hclust)
}
```


Cluster Dendrogram



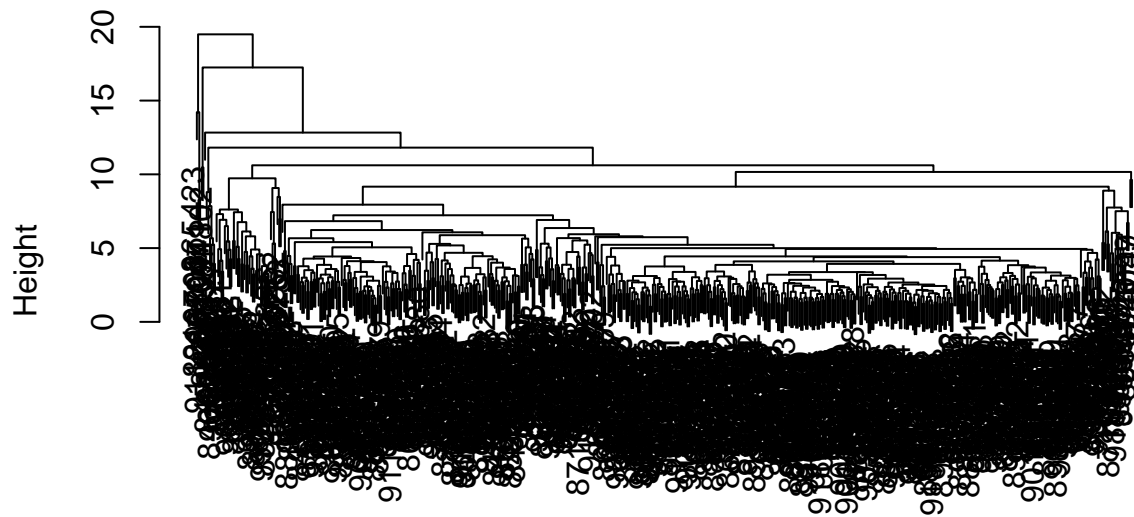
```
data.dist  
hclust (*, "single")
```

Cluster Dendrogram



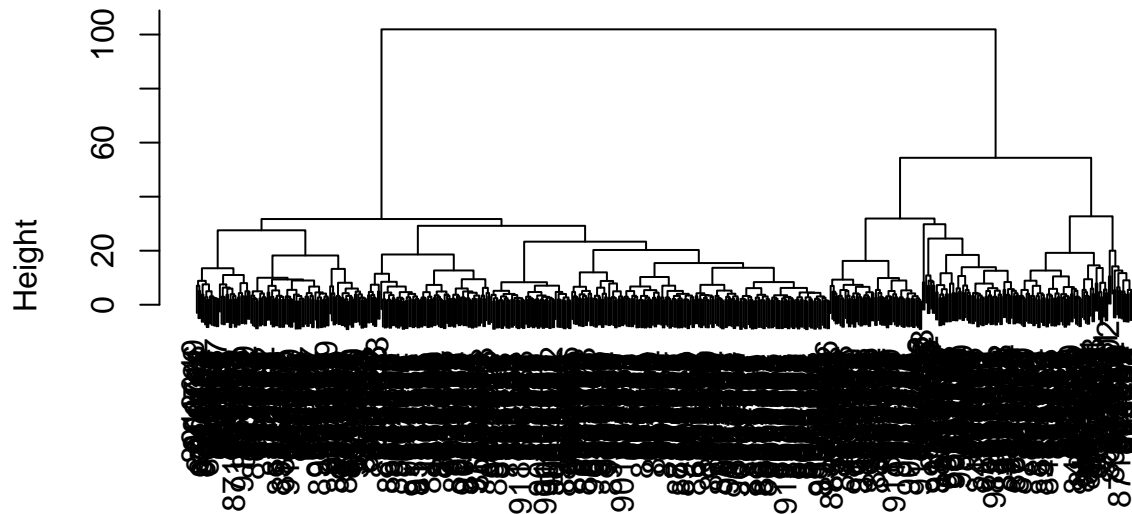
```
data.dist  
hclust (*, "complete")
```

Cluster Dendrogram



```
data.dist  
hclust (*, "average")
```

Cluster Dendrogram



```
data.dist
hclust (*, "ward.D2")
```

K-means clustering and comparing results

Create a k-means model on `wisc.data` and assign the result to `wisc.km`. Create 2 clusters corresponding to the actual number of diagnosis. Scale the data using the `scale()` function and repeat the algorithm 20 times

```
wisc.km <- kmeans(scale(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
##    1 343  37
##    2  14 175
```

Q14. How well does k-means separate the two diagnoses? How does it compare to your `hclust` results? k-means separates the diagnoses into 2 distinct clusters whereas `wisc.hclust.clusters` separated the diagnoses into 4 clusters where clusters 1 and 3 contain the majority of the datapoints. Clusters 1 and 3 in the `wisc.hclust.clusters` dataset and clusters 1 and 2 in the `wisc.km` dataset are very similar. Both separate the diagnoses well.

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`).

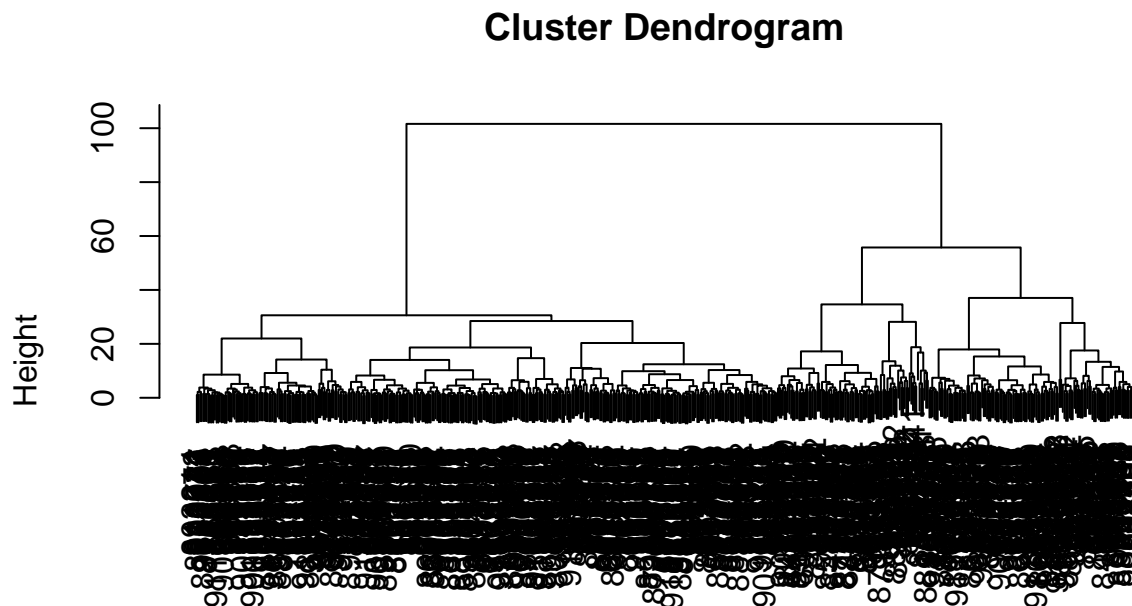
```
table(wisc.hclust.clusters, wisc.km$cluster)
```

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

Combining Methods

Using the minimum number of PCs required to describe at least 90% of the variability in the data, create a hierarchical clustering model with the linkage method="ward.D2"

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



```
dist(wisc.pr$x[, 1:7])
hclust (*, "ward.D2")
```

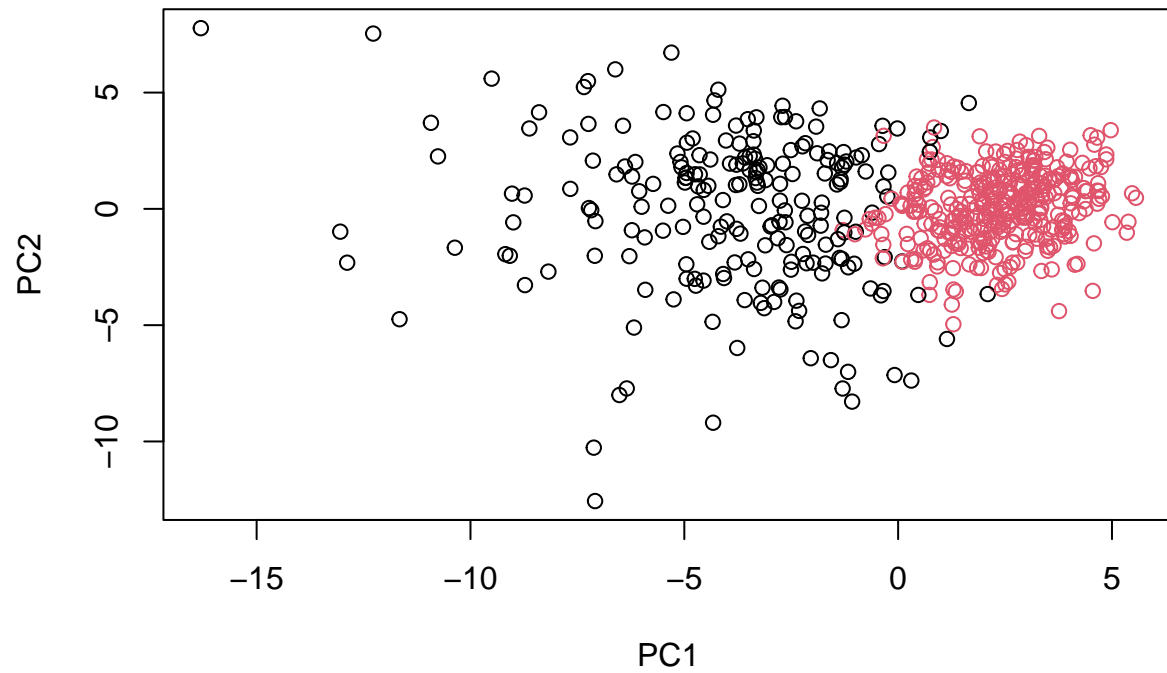
```
grps <- cutree(wisc.pr.hclust, k=2)
table(grps)
```

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

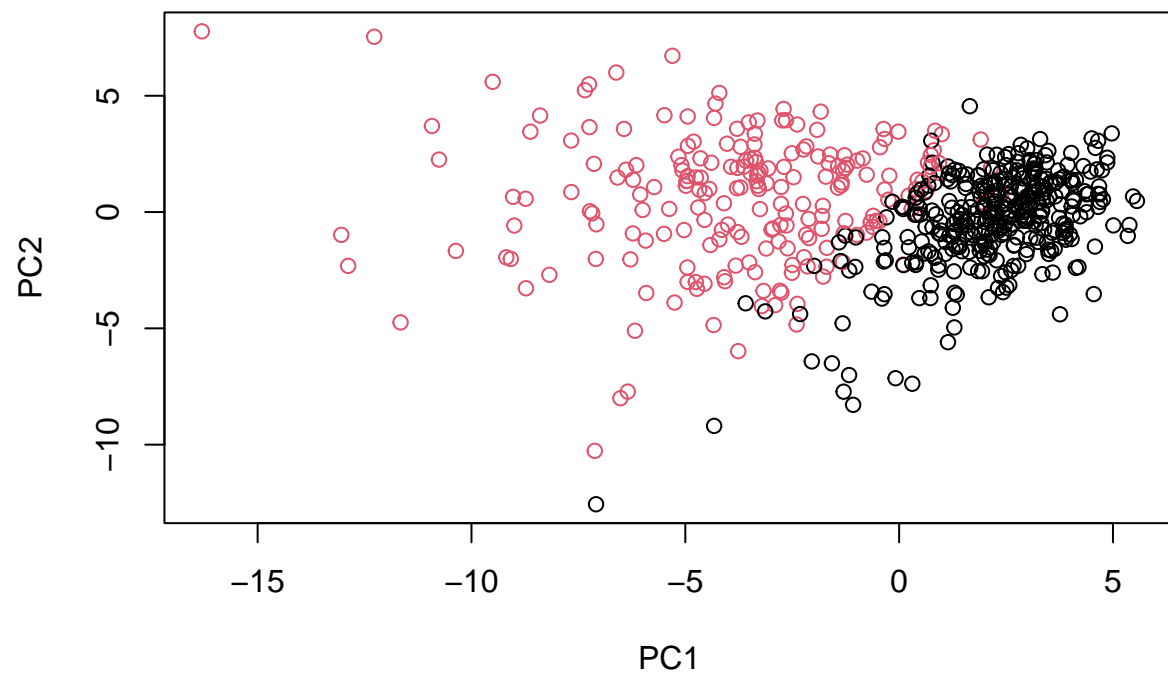
```
table(grps, diagnosis)
```

```
##      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)
```



Turn groups into a factor and reorder the levels

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

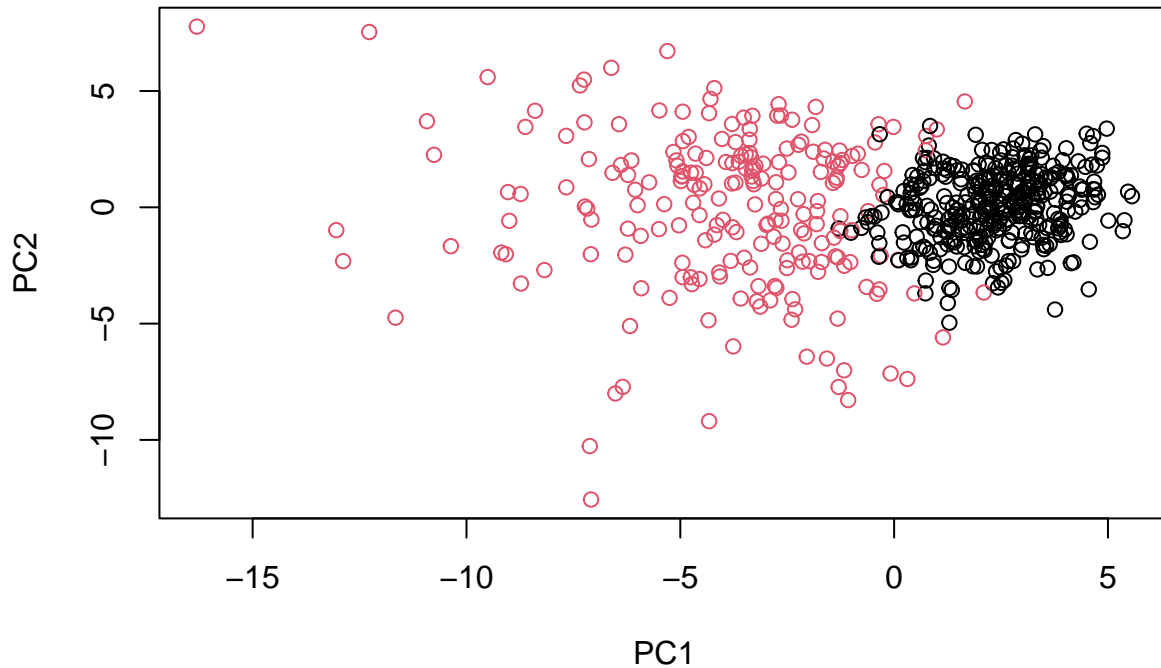
```
## [1] "1" "2"
```

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

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

Plot data using the re-ordered factor

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



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 4 clusters and assign the results to `wisc.pr.hclust.clusters`

```
wisc.pr.hclust.clusters_2 <- cutree(wisc.pr.hclust, k=2)
# clustering on PCA results with 2 clusters
```

```
wisc.pr.hclust.clusters_4 <- cutree(wisc.pr.hclust, k=4) # clustering on PCA results with 4 clusters
```

Q15. How well does the newly created model with four clusters separate out the two diagnoses? The newly created model generates 4 clusters where clusters 1, 2, and 3 are derived from cluster 1 when the model was cut into 2 clusters. Cluster 4 is equivalent to the cluster 2 in the 2 cluster model.

Using `table()`, compare the results from your new hierarchical clustering model with the actual diagnoses.

```
table(wisc.pr.hclust.clusters_2, diagnosis)
```

```
##              diagnosis
## wisc.pr.hclust.clusters_2  B  M
##              1  28 188
##              2  329  24
```



```
table(wisc.pr.hclust.clusters_4, diagnosis)
```

```
##              diagnosis
## wisc.pr.hclust.clusters_4  B  M
##              1  0  45
##              2  2  77
##              3 26  66
##              4 329  24
```

```
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
```

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? Both clustering models do a good job separating the diagnoses. `wisc.hclust.clusters` generates 4 clusters, whereas clusters 1, 2, and 4 are nearly equivalent to cluster 1 generated by k-means. However, clusters 2 and 4 generated by hierarchical clustering contain very few datapoints.

Sensitivity/Specificity

Q17. Which of your analysis procedures resulted in a clustering model with the best specificity? How about sensitivity? The analyses that resulted in the best specificity were hierarchical clustering and k-means clustering. Both of these analyses were correctly able to identify 96% of ill patients. The analysis with the best sensitivity was hierarchical clustering combined with PCA. This analysis correctly rejected 87% of healthy patients.

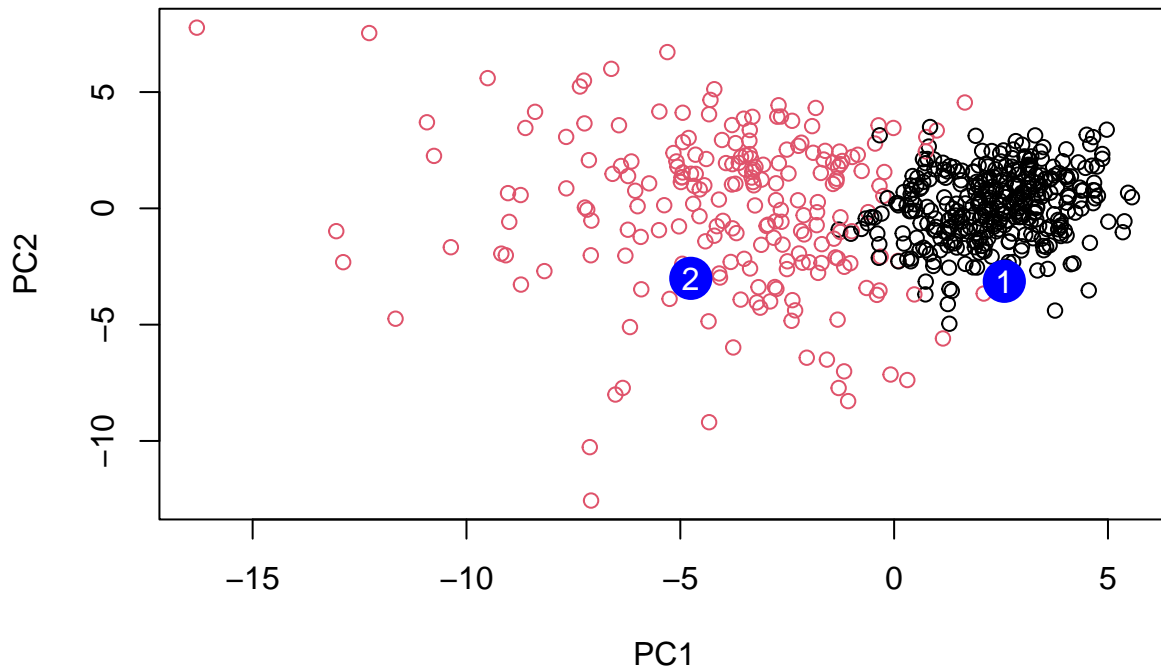
Prediction

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

```
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")
```



Q18. Which of these new patients should we prioritize for follow up based on your results?
 Patient 2 should be prioritized for a follow-up since they cluster with the malignant cluster

Here we use the `sessionInfo()` function to report on our R systems setup at the time of document execution.

```
sessionInfo()
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/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] factoextra_1.0.7 ggplot2_3.3.5
##
## loaded via a namespace (and not attached):
## [1] tidyselect_1.1.1 xfun_0.26      purrr_0.3.4    haven_2.4.3
## [5] carData_3.0-4     colorspace_2.0-2 vctrs_0.3.8    generics_0.1.0
## [9] htmltools_0.5.2  yaml_2.2.1     utf8_1.2.2     rlang_0.4.11
## [13] pillar_1.6.3      ggpubr_0.4.0    foreign_0.8-81 glue_1.4.2
## [17] withr_2.4.2       readxl_1.3.1    lifecycle_1.0.1 stringr_1.4.0
## [21] cellranger_1.1.0  munsell_0.5.0   ggsignif_0.6.3 gtable_0.3.0
## [25] zip_2.2.0         evaluate_0.14    labeling_0.4.2 knitr_1.36
## [29] rio_0.5.27        forcats_0.5.1   fastmap_1.1.0  curl_4.3.2
## [33] fansi_0.5.0       highr_0.9        broom_0.7.9     Rcpp_1.0.7
## [37] scales_1.1.1      backports_1.2.1 abind_1.4-5     farver_2.1.0
## [41] hms_1.1.1         digest_0.6.28    stringi_1.7.5   openxlsx_4.2.4
## [45] rstatix_0.7.0     dplyr_1.0.7      ggrepel_0.9.1   grid_4.1.1
## [49] tools_4.1.1       magrittr_2.0.1    tibble_3.1.5    crayon_1.4.1
## [53] tidyr_1.1.4       car_3.0-11        pkgconfig_2.0.3 ellipsis_0.3.2
## [57] data.table_1.14.2 rmarkdown_2.11    R6_2.5.1        compiler_4.1.1
```