Class 8: Unsupervised Learning Mini-Project

Wenxi Tang

In today's class we will explore a complete analysis using the unsupervised learning techniqyes covered in the last class.

Complete the following code to input the data and store as wisc.df

```
wisc.df <- read.csv("WisconsinCancer.csv", 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
```

Q1. How many observations are in this dataset?

```
dim(wisc.df)
```

```
[1] 569  32
```

There are 569 rows (observations in this dataset).

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

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

```
[1] 212
```

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

```
[1] 357
```

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

```
  B   M
357 212
```

There are 212 malignant and 357 benign observations.

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

```
#get the column names in the dataset.
colnames(wisc.df)
```

```
 [1] "diagnosis"               "radius_mean"
 [3] "texture_mean"            "perimeter_mean"
 [5] "area_mean"               "smoothness_mean"
 [7] "compactness_mean"        "concavity_mean"
 [9] "concave.points_mean"     "symmetry_mean"
[11] "fractal_dimension_mean"  "radius_se"
[13] "texture_se"              "perimeter_se"
[15] "area_se"                 "smoothness_se"
[17] "compactness_se"          "concavity_se"
[19] "concave.points_se"       "symmetry_se"
[21] "fractal_dimension_se"    "radius_worst"
[23] "texture_worst"           "perimeter_worst"
[25] "area_worst"              "smoothness_worst"
[27] "compactness_worst"       "concavity_worst"
[29] "concave.points_worst"    "symmetry_worst"
[31] "fractal_dimension_worst" "X"
```

The `grep()` function can perhaps help us here.

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

```
[1] 10
```

Q. What features are mean values?

```
grep("_mean", colnames(wisc.df), value = TRUE)
```

```
 [1] "radius_mean"            "texture_mean"           "perimeter_mean"
 [4] "area_mean"              "smoothness_mean"        "compactness_mean"
 [7] "concavity_mean"         "concave.points_mean"    "symmetry_mean"
[10] "fractal_dimension_mean"
```

```
# First diagnosis column removed before doing analysis.
wisc.data <- wisc.df[,-1]
```

```
# Delete the last column "X".
wisc.data <- wisc.data[,-31]
```

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

2. Principal Component Analysis

The main PCA fucntion in base R is called `prcom()`

Before doing anything like PCA, it is important to check if the data need to be scaled. Recall two common reasons for scaling data include: - The input variables use different units of measurement. - The input variables have significantly different variances.

```
# Check column means and standard deviations
colMeans(wisc.data)
```

```
     radius_mean     texture_mean     perimeter_mean
    1.412729e+01     1.928965e+01     9.196903e+01
       area_mean  smoothness_mean  compactness_mean
```

```
          6.548891e+02           9.636028e-02           1.043410e-01
        concavity_mean     concave.points_mean          symmetry_mean
          8.879932e-02           4.891915e-02           1.811619e-01
fractal_dimension_mean              radius_se             texture_se
          6.279761e-02           4.051721e-01           1.216853e+00
          perimeter_se                area_se         smoothness_se
          2.866059e+00           4.033708e+01           7.040979e-03
        compactness_se          concavity_se      concave.points_se
          2.547814e-02           3.189372e-02           1.179614e-02
           symmetry_se    fractal_dimension_se           radius_worst
          2.054230e-02           3.794904e-03           1.626919e+01
         texture_worst        perimeter_worst             area_worst
          2.567722e+01           1.072612e+02           8.805831e+02
     smoothness_worst       compactness_worst         concavity_worst
          1.323686e-01           2.542650e-01           2.721885e-01
  concave.points_worst         symmetry_worst fractal_dimension_worst
          1.146062e-01           2.900756e-01           8.394582e-02
```

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

```
          radius_mean           texture_mean         perimeter_mean
                 3.52                   4.30                   24.30
            area_mean        smoothness_mean        compactness_mean
               351.91                   0.01                   0.05
       concavity_mean     concave.points_mean          symmetry_mean
                 0.08                   0.04                   0.03
fractal_dimension_mean              radius_se             texture_se
                 0.01                   0.28                   0.55
         perimeter_se                area_se         smoothness_se
                 2.02                  45.49                   0.00
       compactness_se          concavity_se      concave.points_se
                 0.02                   0.03                   0.01
          symmetry_se    fractal_dimension_se           radius_worst
                 0.01                   0.00                   4.83
        texture_worst        perimeter_worst             area_worst
                 6.15                  33.60                 569.36
     smoothness_worst       compactness_worst         concavity_worst
                 0.02                   0.16                   0.21
  concave.points_worst         symmetry_worst fractal_dimension_worst
                 0.07                   0.06                   0.02
```

The difference in variance and sd is very big, so we want to scale the data by setting `scale = TRUE` in our `prcomp()` function call.

```
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 of variance is captured by PC1.

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

3 PCs are required to describe at least 70% of the original variance.

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

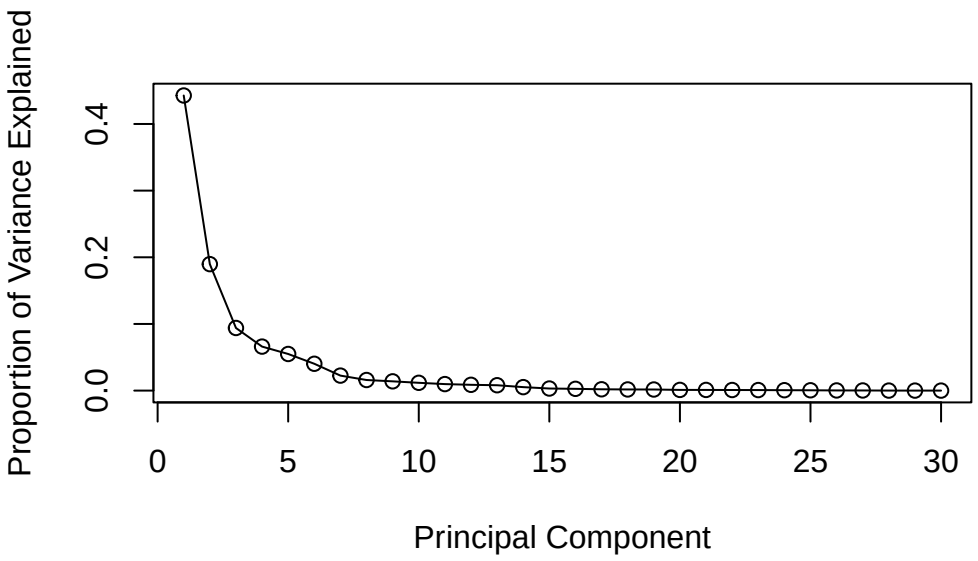
7 PCs are required to describe at lease 90% of the original variance.

```
attributes(wisc.pr)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

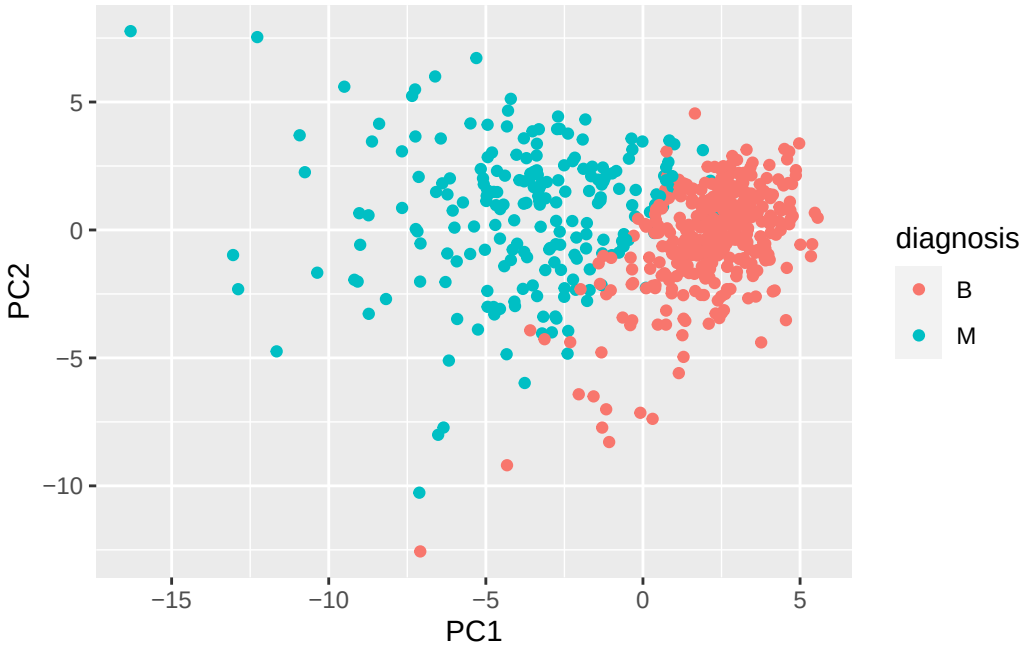
Make a little SCREE plot:

```
pr.var <- wisc.pr$sdev^2

# Proportion of variance
pve <- pr.var/sum(pr.var)

plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained", typ = "o")
```

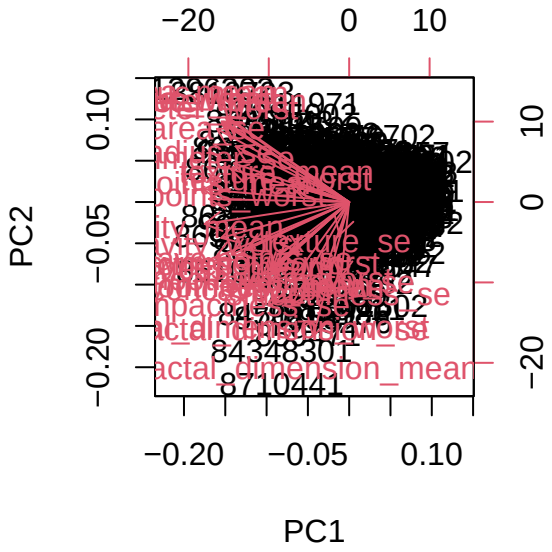


Elbow point is PC3, after that is only the marginal effect.

```
library(ggplot2)

pc <- as.data.frame(wisc.pr$x)

ggplot(pc) +
  aes(PC1, PC2, col= diagnosis) +
  geom_point()
```



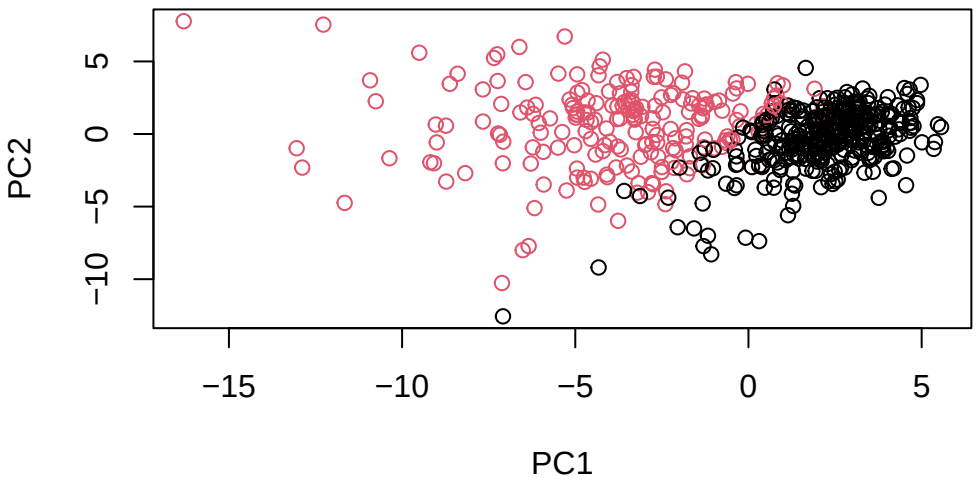

```
biplot(wisc.pr)
```



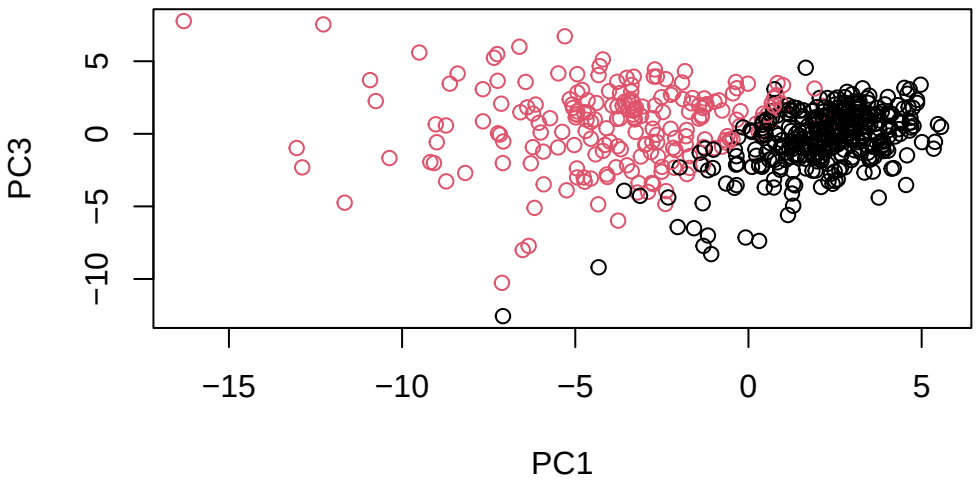
Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?

This plot above is a little bit hard to understand because it's messed up with all labels and I couldn't really tell how the two conditions separates.

```
# Scatter plot observations by components 1 and 2
plot(pc$PC1, pc$PC2, col = diagnosis ,
     xlab = "PC1", ylab = "PC2")
```



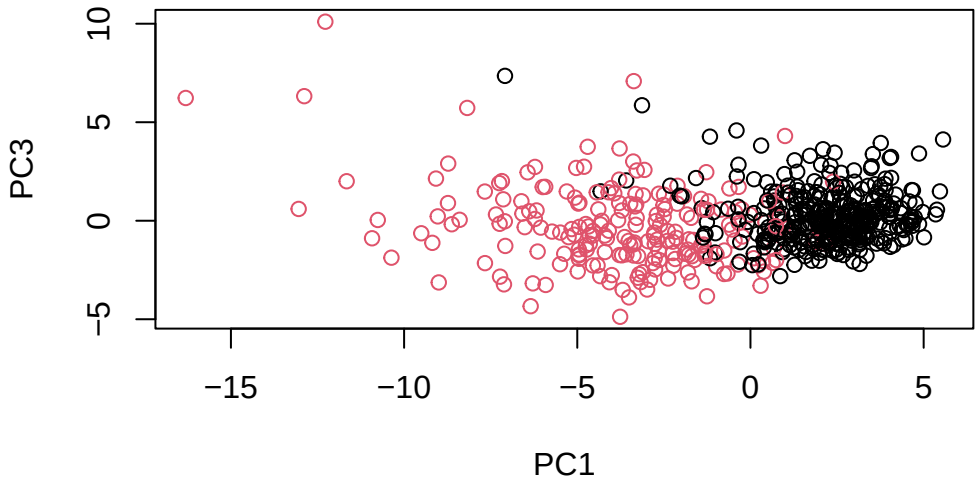

```
plot(wisc.pr$x[, c(1,2)], col = diagnosis,
     xlab = "PC1", ylab = "PC3")
```



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

It still separates the observations but it has more overlapping compared to the plot for PC1 and PC2, because PC2 explains more variance in this dataset than PC3 does.

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



Q9. For the first principal component, what is the component of the loading vector (i.e. wisc.pr$rotation[,1]) for the feature concave.points_mean? This tells us how much this original feature contributes to the first PC.

concave.points_mean -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
```

Hierarchical clustering

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

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

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

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

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

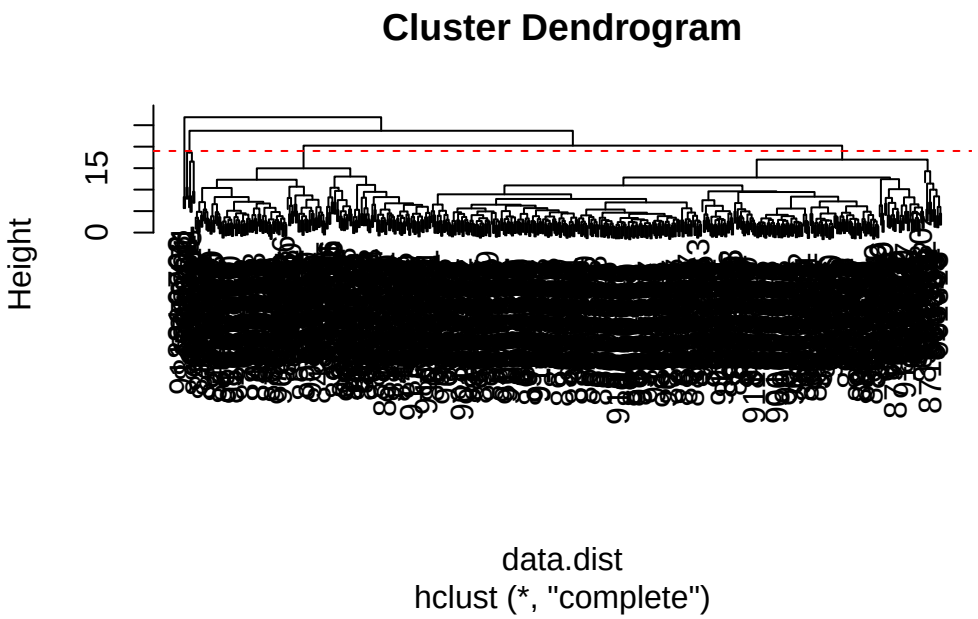
Create a hierarchical clustering model using complete linkage. Manually specify the method argument to hclust() and assign the results to wisc.hclust.

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

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

Plot it out. At height = 19, there are 4 clusters.

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



To get a cluster membership vector I will use the `cutree` function and cut into 4 groups or clusters.

```
grps <- cutree(wisc.hclust, h=19) #or can use `cutree(k=4)`
table(grps)
```

```
grps
  1   2   3   4
177   7 383   2
```

I can also use the `table()` to cross tabulate.

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

```
    diagnosis
```

```
grps   B   M
   1  12 165
   2   2   5
   3 343  40
   4   0   2
```

Calculate the accuracy of this clustering result.

```
(165+343)/nrow(wisc.data)
```

```
[1] 0.8927944
```

Try cluster N of 2 instead of 4.

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

```
grps
  1   2
567   2
```

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

```
    diagnosis
grps   B   M
   1 357 210
   2   0   2
```

Try culster N of 5.

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

```
grps
  1   2   3   4   5
177   5 383   2   2
```

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

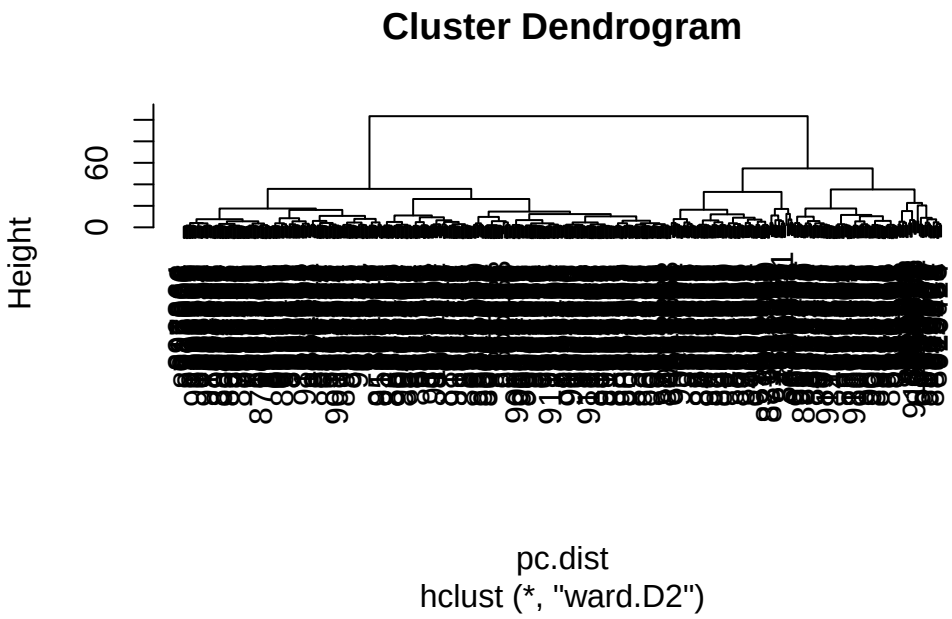
```
    diagnosis
grps   B   M
   1  12 165
   2   0   5
   3 343  40
   4   2   0
   5   0   2
```

Combining methods - Clustering on PCA result

I can cluster in PC-space and use as many or as few PCs as I want.

To start with I will use 3 PCs, that is I will cluster along PC1, PC2 and PC3.

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



This looks much nicer than our previous clustering result. Let's find the two major clusters with the `cutree()` functions.

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

```
grps
  1   2
203 366
```

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

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

We could calculate accuracy - the proportion of samples we got correct if we take cluster 1 to represent all M and cluster 2 to represent all B.

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

```
[1] 0.8998243
```

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

We can techniquely try out using different number of clusters. And we can use the `table()` function to see how much "M" and "B" are in each group. We can calculate the accuracy of our model by calculating how many cases are predicted/clustered correctly in this dataset like what's shown above.

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

The ward.D2 method gives me favorite results because it minimizes the variance across the clusters.

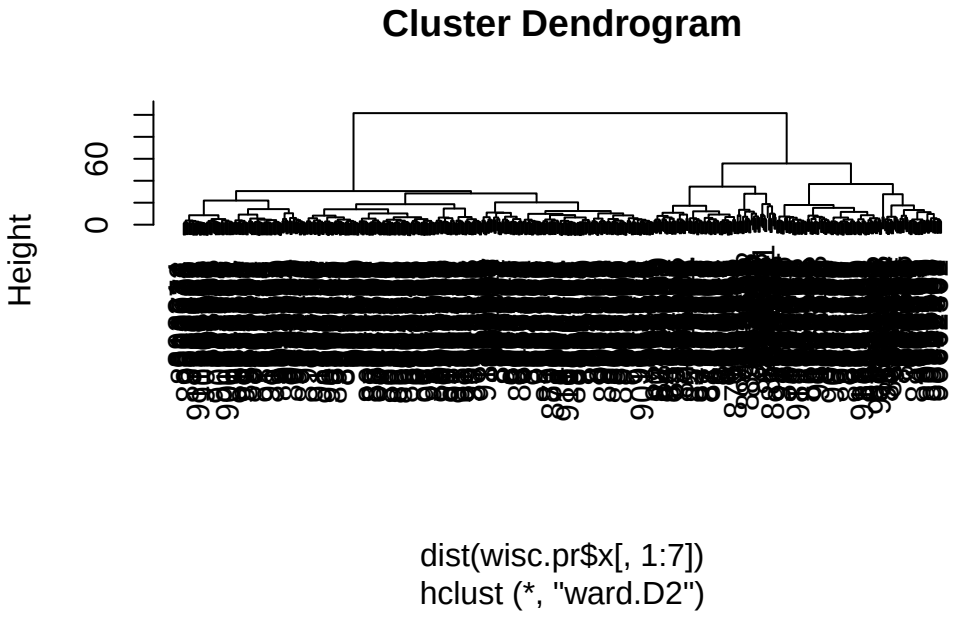
Specificity and Sensitivity

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).

Use the distance along the first 7 PCs for clustering i.e. wisc.pr$x[, 1:7]

```
wisc.pr.hclust2 <- hclust(dist(wisc.pr$x[, 1:7]), method="ward.D2")
plot(wisc.pr.hclust2)
```



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

```
wisc.pr.hclust2.clusters <- cutree(wisc.pr.hclust2, k=2)
table(wisc.pr.hclust2.clusters, diagnosis)
```

```
                        diagnosis
```

```
wisc.pr.hclust2.clusters   B   M
                       1  28 188
                       2 329  24
```

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

It separates the two diagnoses pretty well if we look at the accuracy of what proportion of diagnosis is predicted correctly, calculation shown below.

Let's calculate the accuracy.

```
(188+329)/nrow(wisc.data)
```

```
[1] 0.9086116
```

> Q14. How well do the hierarchical clustering models you created in previous sections (i.e. before PCA) do in terms of separating the diagnoses? Again, use the table() function to compare the output of each model (wisc.km$cluster and wisc.hclust.clusters) with the vector containing the actual diagnoses.

Calculation below. Overall, this method separates the M and B groups okay because we can see that group 1 covers 165 true M patients and group 3 covers 343 true B patients. If we look at the sensitivity and specificity of this method, it has a sensitivity of 0.804878 and specificity of 0.9661972. The sensitivity is not very high for correctly predicting malignant cancers compared to other methods, but the specificity is pretty good. The accuracy is above 0.89, so it's still a good prediction.

table() results copied from above: diagnosis grps B M 1 12 165 2 2 5 3 343 40 4 0 2

```
#Sensitivity
165/(165+40)
```

```
[1] 0.804878
```

```
#Specificity
343/(343+12)
```

```
[1] 0.9661972
```

```
#Accuracy
(165+343)/nrow(wisc.data)
```

```
[1] 0.8927944
```

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

Clustering with PCA using 1:7PCs gives highest sensitivity of 0.8867925, the hierarchical clustering gives highest specificity of 0.9661972. Working shown below.

wisc.pr.hclust: diagnosis grps B M 1 24 179 2 333 33

```
#Sensitivity
179/(179+33)
```

```
[1] 0.8443396
```

```
#Specificity
333/(333+24)
```

```
[1] 0.9327731
```

diagnosis wisc.pr.hclust2.clusters B M 1 28 188 2 329 24

```
#Sensitivity
188/(188+24)
```

```
[1] 0.8867925
```

```
#Specificity
329/(329+28)
```

```
[1] 0.9215686
```

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

```
#Sensitivity
165/(165+40)
```

```
[1] 0.804878
```

```
#Specificity
343/(343+12)
```

```
[1] 0.9661972
```

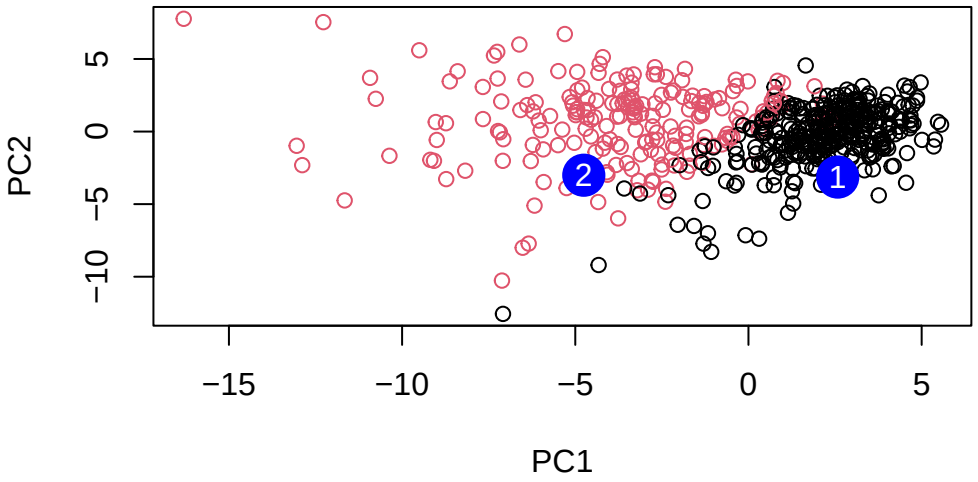
Prediction

```
#url <- "new_samples.csv"
url <- "https://tinyurl.com/new-samples-CSV"
new <- read.csv(url)
#predict the PC values for new dataset.
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=diagnosis)
points(npc[,1], npc[,2], col="blue", pch=16, cex=3)
```

```
text(npc[,1], npc[,2], c(1,2), col="white")
```



Q16. Which of these new patients should we prioritize for follow up based on your results?

We should prioritize patient 2 for follow up because this patient falls into the Malignant cluster while patient 1 falls into the Benign cluster.