lab08

Shreyas Sankaranarayanan

##About

##Data Import

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

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

Q1. How many observations are in this dataset?

According to the snippet below, there are 569 patients in this data set.

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

```
[1] 569
```

Q2. How many observations have a malignant diagnosis?

According to the code snippet below, there are 212 patients with a malignant diagnosis

```
sum(diagnosis == "M" )
```

```
[1] 212
```

```
table(diagnosis)
```

```
diagnosis
  B   M
357 212
```

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

There are 10 features with the suffix "_mean" according to the coding snippet below:

```
sum(grepl("_mean" , names(wisc.data), fixed = TRUE))
```

```
[1] 10
```

```
grep("_mean", names(wisc.data))
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

##Initial Analysis

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

Q5.How many principal components (PCs) 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?

##Clustering

We can try `kmeans()` clustering first:

```
km <- kmeans(wisc.data, centers = 2)
table(km$cluster)
```

```
  1   2
131 438
```

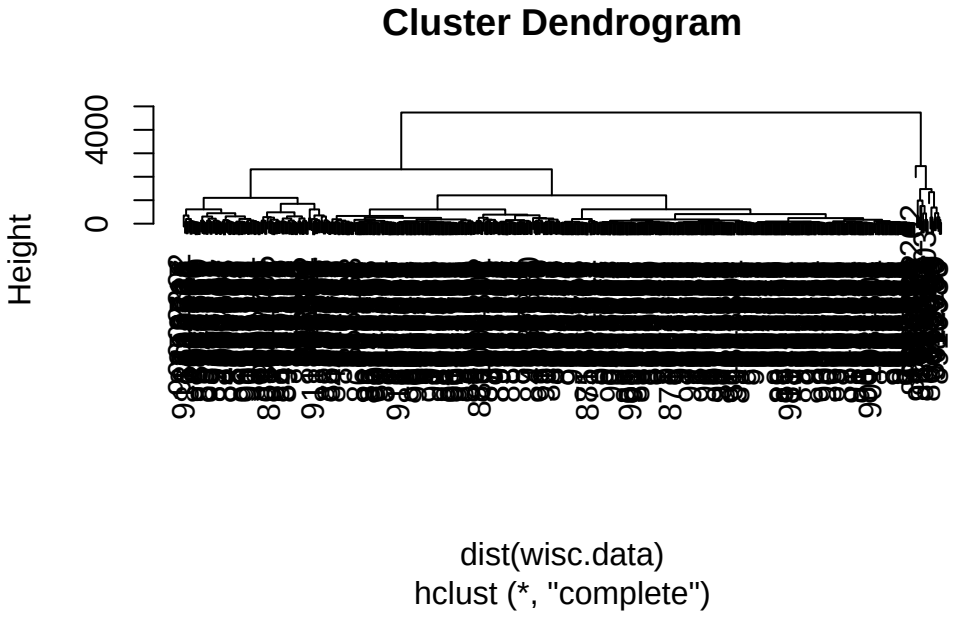
Cross-Table

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

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

Then try heirarchical clustering (`hclust()`):

```
hc <- hclust(dist(wisc.data))
plot(hc)
```



The question arises whether we need to scale so we must look at the standard deviations of our different data values

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

```
        radius_mean        texture_mean      perimeter_mean
                  4                   4                  24
          area_mean     smoothness_mean    compactness_mean
                352                   0                   0
     concavity_mean concave.points_mean       symmetry_mean
                  0                   0                   0
```

```
 fractal_dimension_mean               radius_se              texture_se
                      0                       0                       1
          perimeter_se                 area_se          smoothness_se
                      2                      45                       0
        compactness_se           concavity_se       concave.points_se
                      0                       0                       0
           symmetry_se    fractal_dimension_se            radius_worst
                      0                       0                       5
         texture_worst         perimeter_worst              area_worst
                      6                      34                     569
      smoothness_worst       compactness_worst         concavity_worst
                      0                       0                       0
   concave.points_worst         symmetry_worst fractal_dimension_worst
                      0                       0                       0
```

Yes, we need to scale and so we will run `prcomp()` with `scale=TRUE`

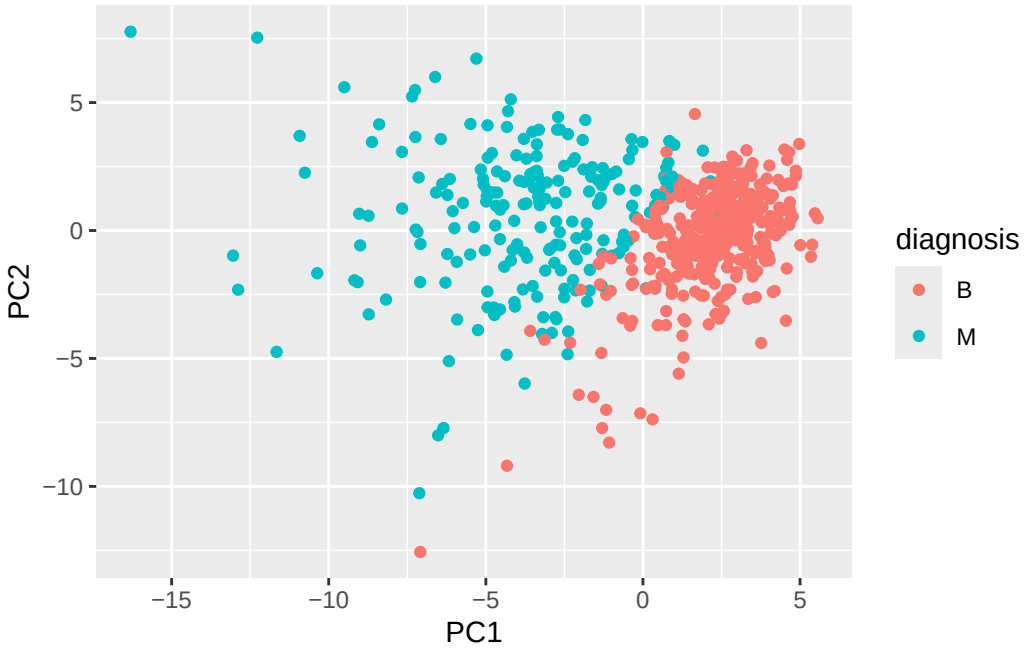
```
wisc.pr <- prcomp(wisc.data, scale = TRUE)
summary(wisc.pr)
```

```
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
```

Generate our main PCA plot (score plot, PC1 v. PC2 plot)...

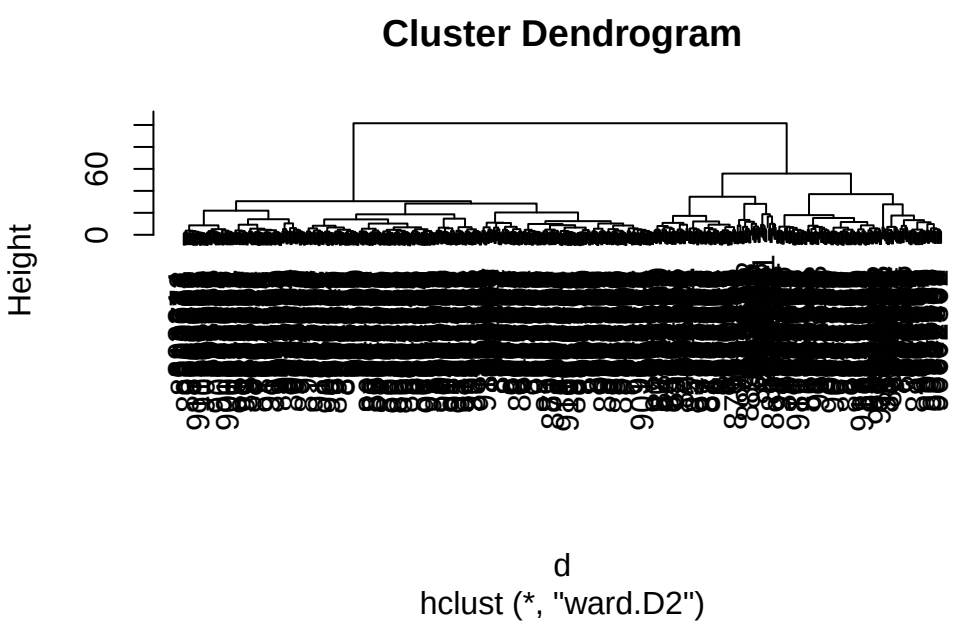
```
library(ggplot2)
res <-as.data.frame(wisc.pr$x)
```

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



Now we cluster on PCA results:

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




```
grps <- cutree(hcd, k=2)
table(grps)
```

```
grps
  1   2
216 353
```

To get clustering result/membership vector I need to cut the tree with `cutree()` function.

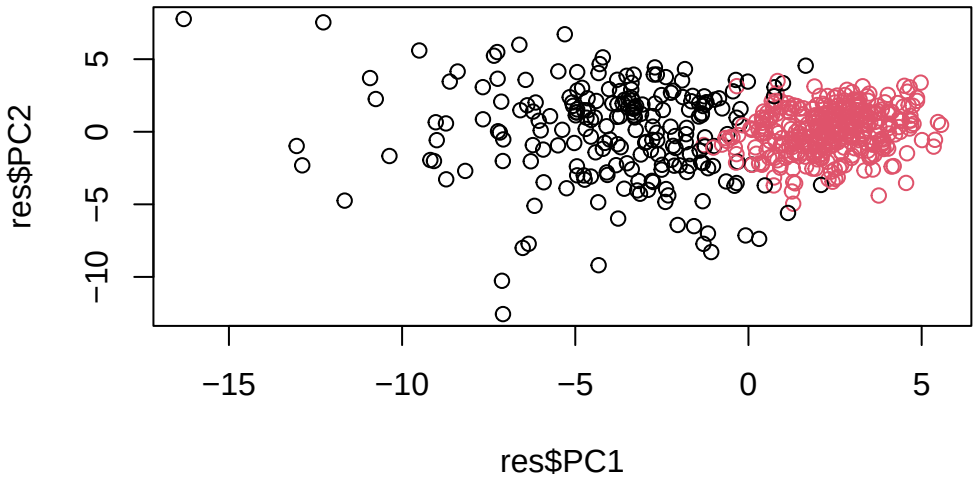
```
grps <- cutree(hcd, k=2)
table(grps)
```

```
grps
  1   2
216 353
```

Q. How many patients in each cluster group?

203 in Group 1 and 366 in group 2

```
plot(res$PC1, res$PC2, col = grps)
```



#Prediction

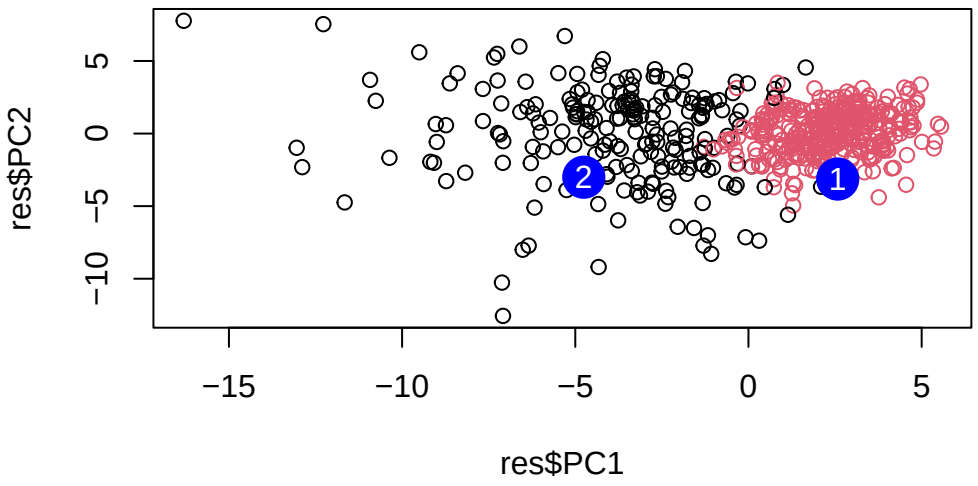
We can utilzie PCA results (our model) to make predictions. We can take unseen data

```
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
```

```r
plot(res$PC1, res$PC2, col = grps)
points(npc[,1],npc[,2], col = "blue", pch = 16, cex = 3)
text(npc[,1], npc[,2], labels = c(1,2), col = "white")
```



#Summary

Principal Component Analysis (PCA) is a super useful technique for analyzing large datasets. The algorithm finds new variables (PCs) that attempt to capture the maximum variance from original variables in the dataset.