class09

Thisha Thiagarajan A15474979

10/26/2021

Preparing the Data

```
wisc.df <- read.csv("WisconsinCancer.csv", row.names=1)
#omit first column (diagnosis) since that is essentially the answer
wisc.data <- wisc.df[,-1]
#save the diagnosis data to check our work later
diagnosis <- factor(wisc.df[,1])
```

> **Q1. How many observations are in this data set?**

Using the dim() function, I determined that there are 569 observations in the data set.

```
dim(wisc.data)
```

```
## [1] 569  30
```

> **Q2. How many obervations have a malignant diagnosis?**

Using the length() function combined with the grep() function, I determined that there are 47.

```
#length totals the grep() to get the # obs with "M"
length(grep("M", diagnosis))
```

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

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

There are 10 features in the data that are suffixed with _mean. I used the colnames(), length(), and grep() functions to determine this.

```
#create vector with just the colnames
colnam <- colnames(wisc.data)
#total number of colnames with _mean
length(grep("_mean",colnam))
```

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

Principal Component Analysis

Determine if the data needs to be scaled. We do need to set scale = TRUE since the columns data are on different scales.

```
#determine col means
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
```

```
#determine standard deviation
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
```

Execute PCA

```r
#pca
wisc.pr <- prcomp(wisc.data, scale = TRUE)
#print summary info
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)?**

44.27% of the original variance is captures by the first principal component.

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

PC1, PC2, PC3 are needed to describe at least 70% (72.6% to be exact).

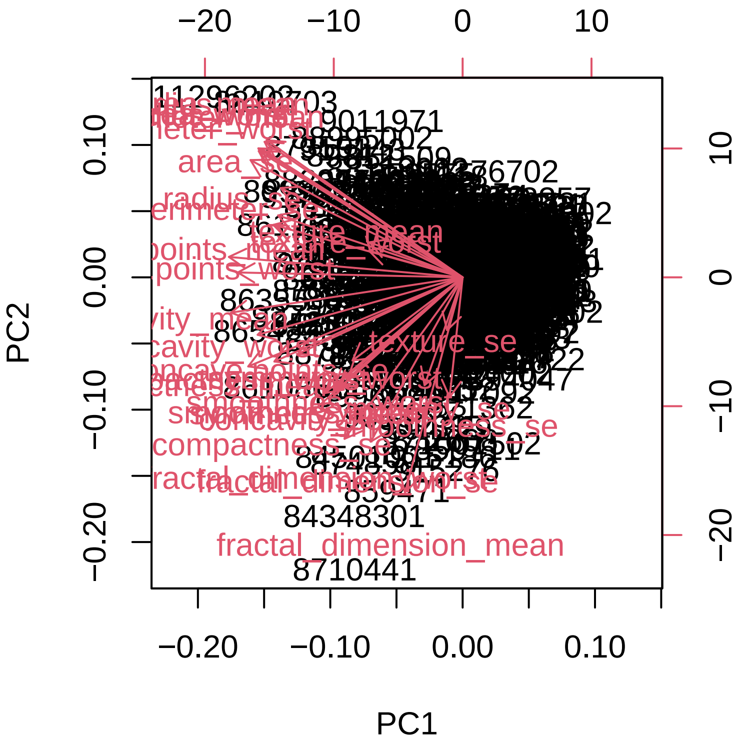
> **Q6. How many principal components (PCs) are required to describe at least 90% of the original variance in the data?**

The first 7 PC components are needed to describe at least 90% (91% to be exact).

Interpret PCA

Create a biplot

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

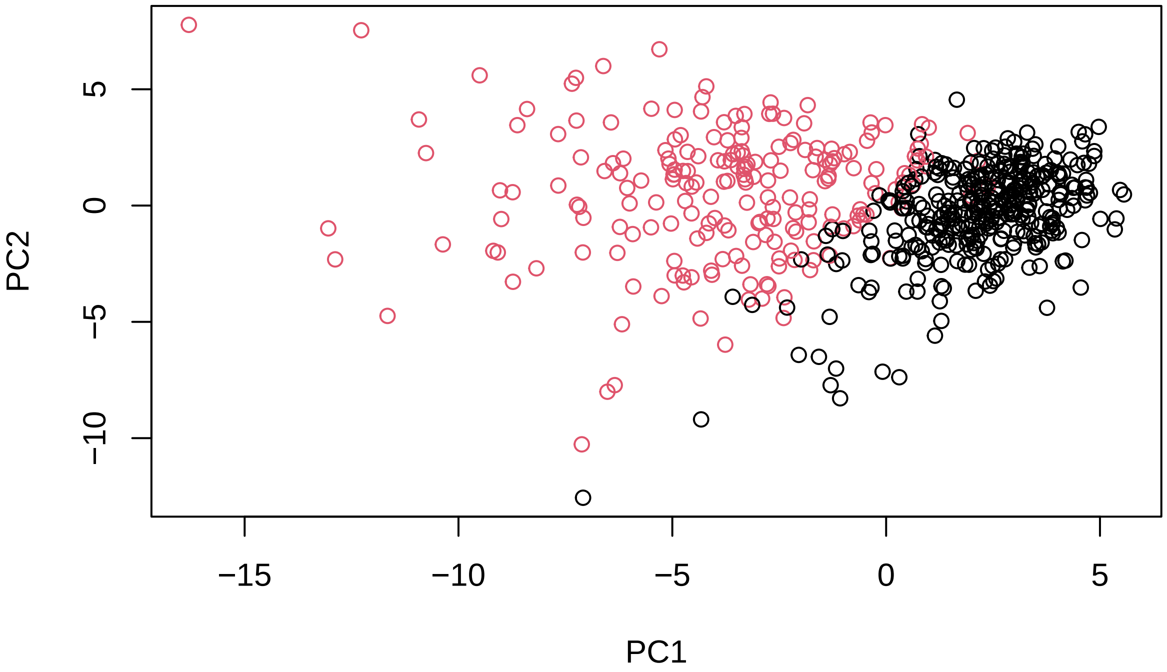


Q7. What stands out to you about this plot? Is it easy or difficult to understand? Why?

This plot is difficult for me to understand since there are many data points. The points overlap throughout the plot, making it difficult to even see labels to interpret the results. Overall, there are too many data points to effectively use this plot to interpret results. Thus, I think we will need to use other techniques.

Plot of PCA1 vs PCA2

```r
plot(x = wisc.pr$x[,1], y = wisc.pr$x[,2], col = diagnosis ,
     xlab = "PC1", ylab = "PC2")
```

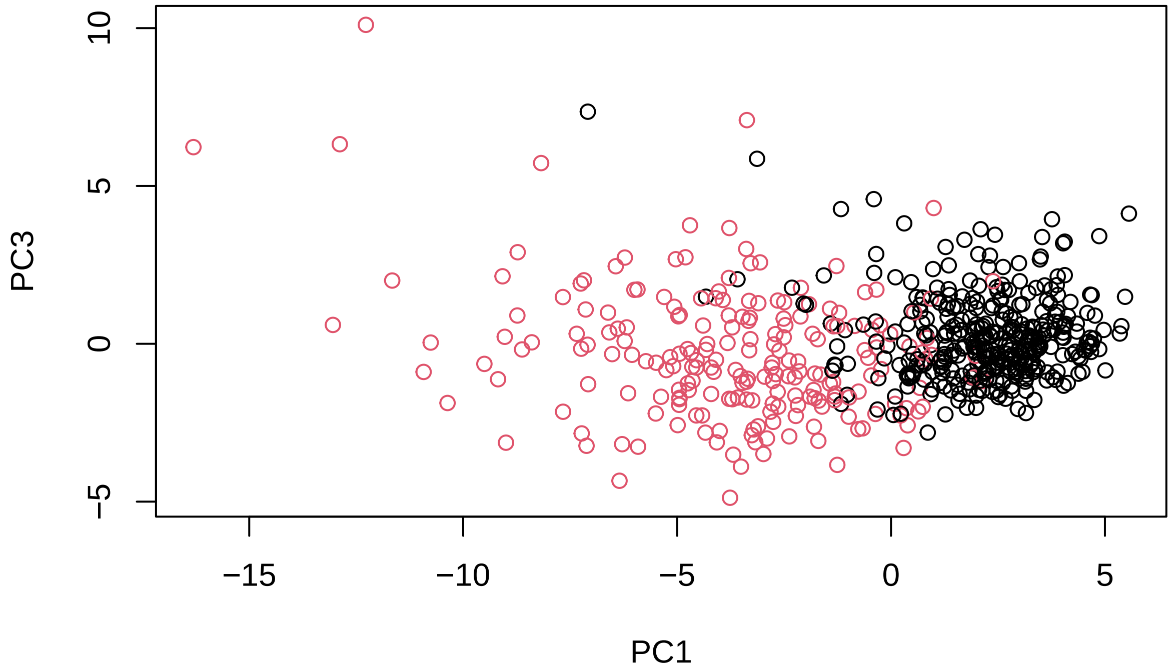


```
#generic color palette sets 0 = black, 1 = red
```

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

Both plots have similar subgroups (benign vs malignant!, this is the data we were hoping to get). The first plot (PC1 vs PC2) has a cleaner separation between the two subgroups and this is most likely due to how PC2 accounts for a larger proportion of variance in the data than PC3.

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

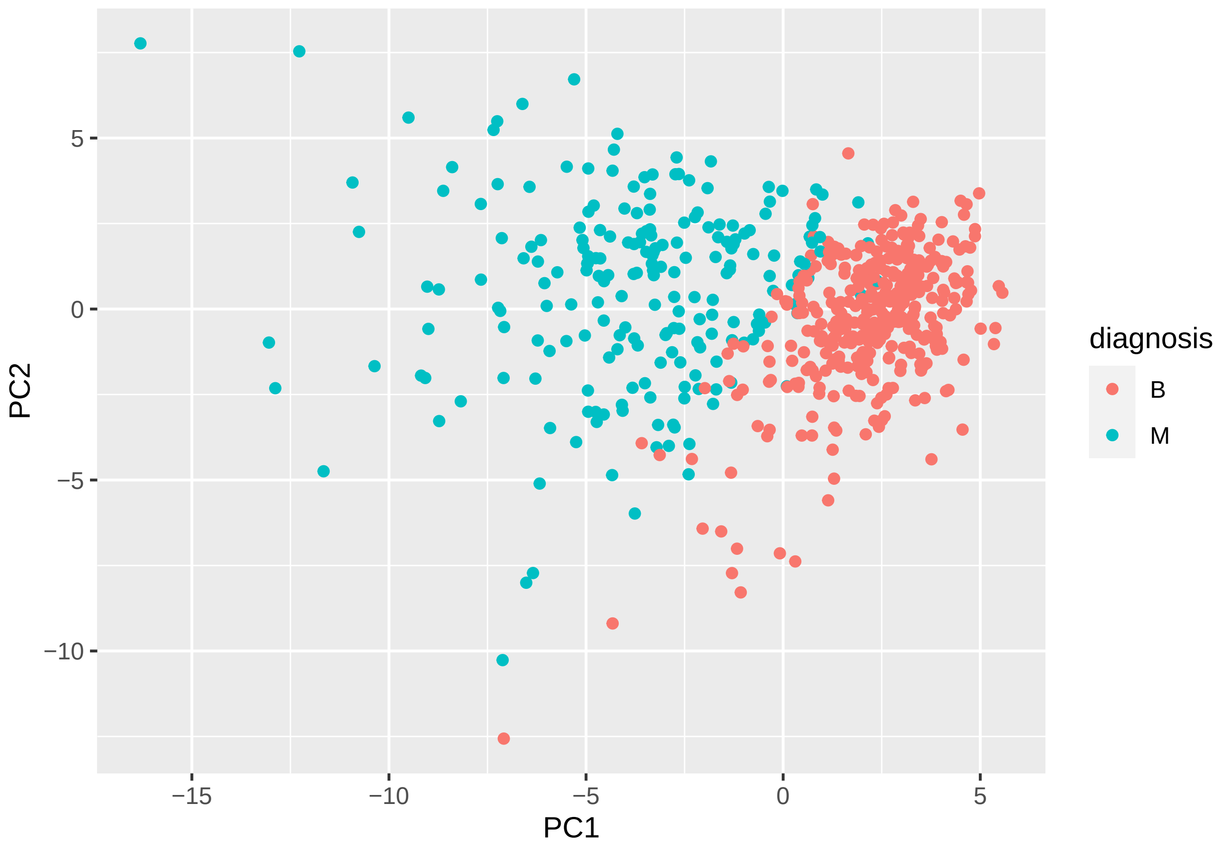


Create ggplot

```
# ggplot needs data frame
df<- as.data.frame(wisc.pr$x)

# load
library(ggplot2)

# plot
ggplot(df) +
  aes(x=PC1, y=PC2, col=diagnosis) +
  geom_point()
```



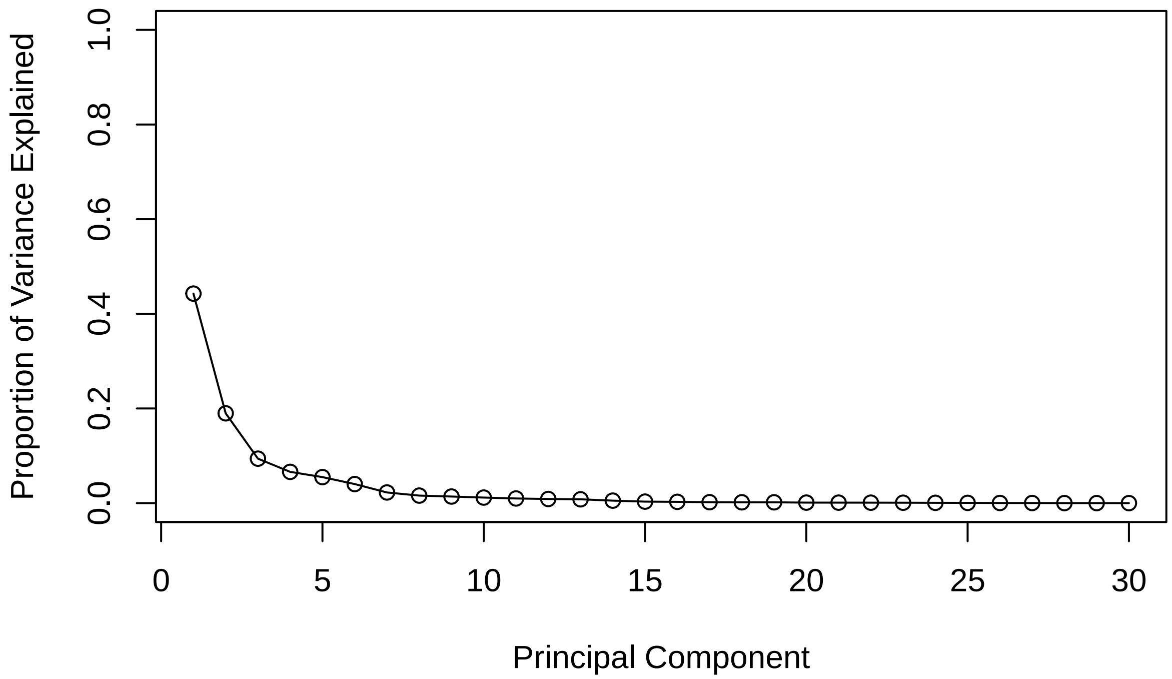
Understand Variance

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

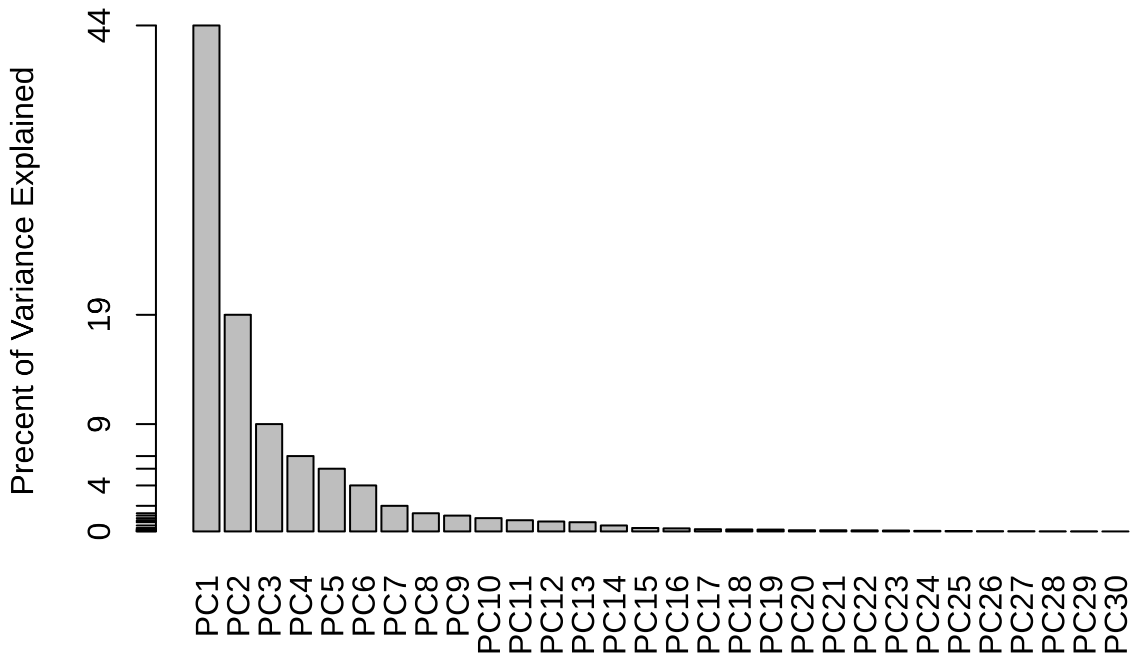
```
## [1] 13.281608  5.691355  2.817949  1.980640  1.648731  1.207357
```

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

# scree plot (scatterplot)
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "o")
```



```
#scree plot (barplot)
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 )
```

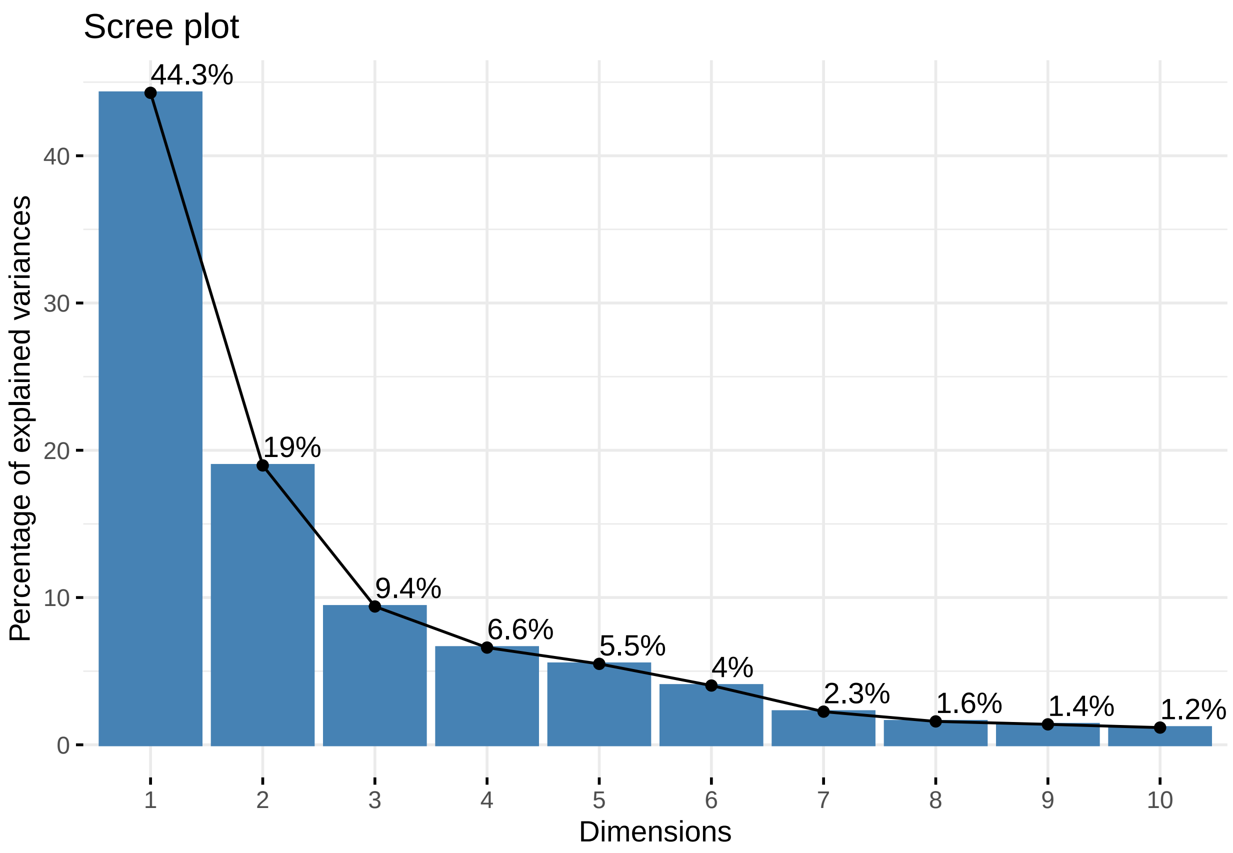


Use “factoextra” package to make a fancy scree plot.

```
#install.packages("factoextra")
library(factoextra)
```

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

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



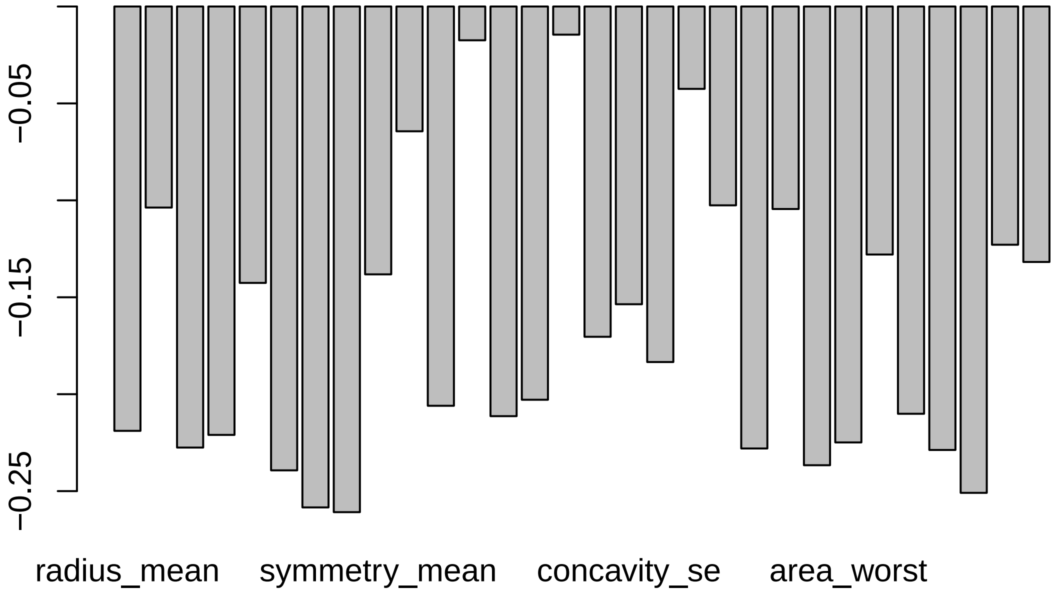
Q9. For the first principal component, what is the component of the loading vector (i.e. wisc.pr$rotation

, 1

) for the feature concave.points_mean?

For pca1, the component of the loading vector for the feature concave.points_mean is -0.2608538.

```
barplot(wisc.pr$rotation[,1])
```



```
pca1_loadingvector <- wisc.pr$rotation[,1]
pca1_loadingvector["concave.points_mean"]
```

```
## concave.points_mean
##          -0.2608538
```

```
##or you can also just use wisc.pr$rotation["concave.points_mean",1]
```

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

5 principal components are required to explain 80% of the variance of the data. I revisited the summary of wisc.pr to determine this.

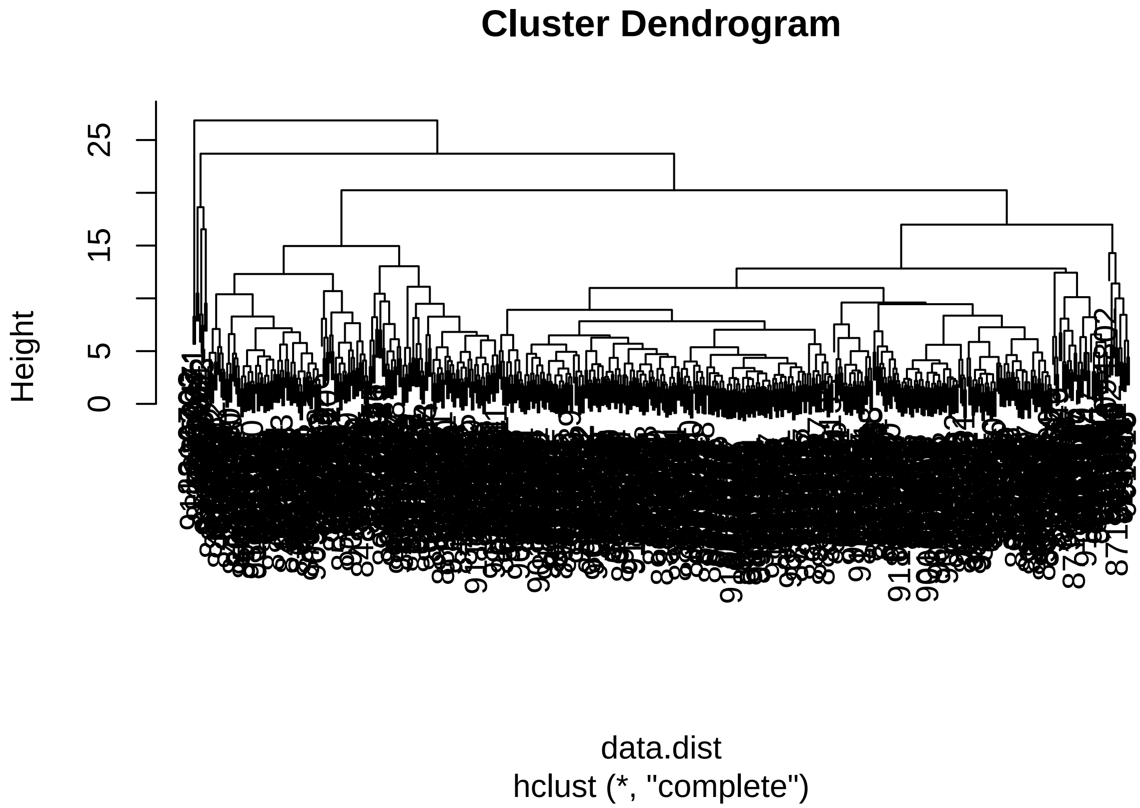
```
summary(wisc.pr)
```

```
## 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

Remember, with this type of clustering you need to specify the number of groups.

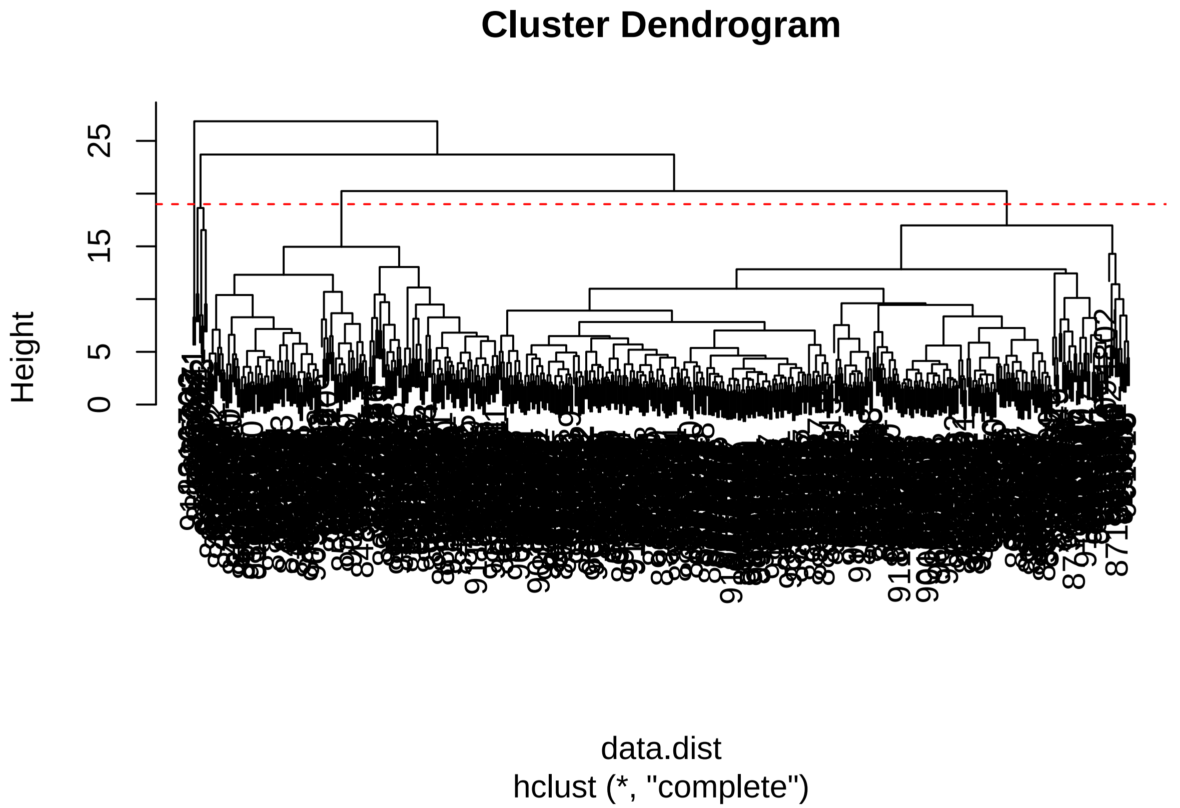
```
#scale data
data.scaled <- scale(wisc.data)
data.dist <- dist(data.scaled)
wisc.hclust <- hclust(data.dist, method = "complete")
#plot hclust
plot(wisc.hclust)
```



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

At a height of 19.

```
plot(wisc.hclust)
#abline() will draw the line
abline(h = 19, col="red", lty=2)
```



Selecting Number of Clusters

```
wisc.hclust.clusters <- cutree(wisc.hclust, 4)
table(wisc.hclust.clusters, diagnosis)
```

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

Cluster 1 and 3 are somewhat split into the benign and malignant clusters, but its not very helpful data.

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

I tried all the different number of clusters between 2 and 10 and I was not able to find a better cluster vs. diagnosis match.

```
wisc.hclust.clusters <- cutree(wisc.hclust, 5)
table(wisc.hclust.clusters, diagnosis)
```

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

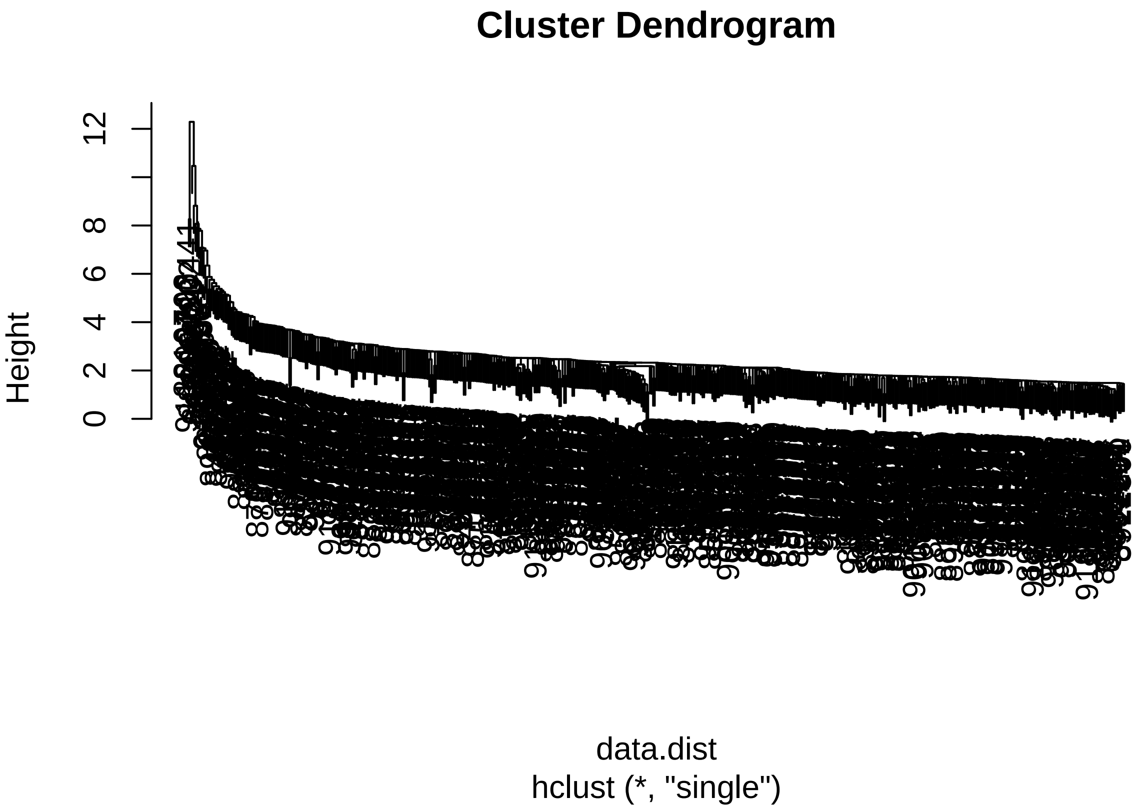
Using different methods

There are 4 different methods; these include "single", "complete", "average" and (my favorite) "ward.D2".

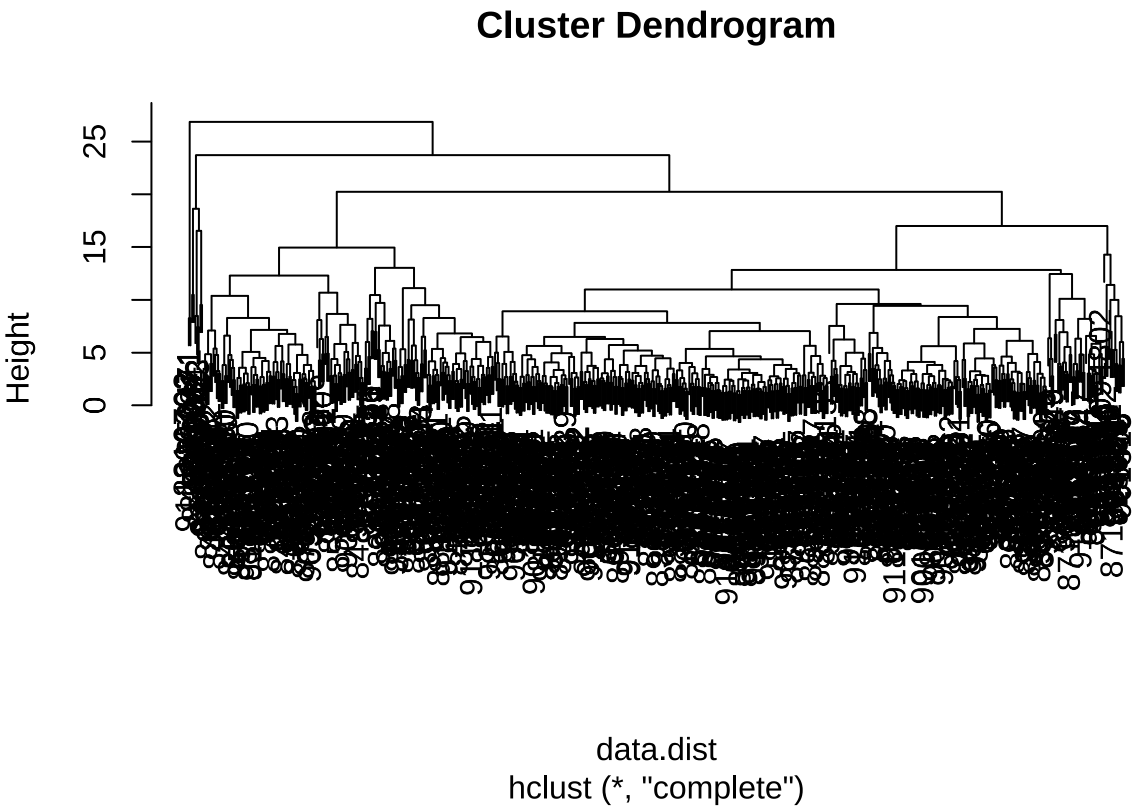
Q13. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning.

Ward.D2 has the the most balanced dendrogram. As explained, it creates groups such that the variance is minimized within clusters. All the other dendrograms are heavily skewed to one side.

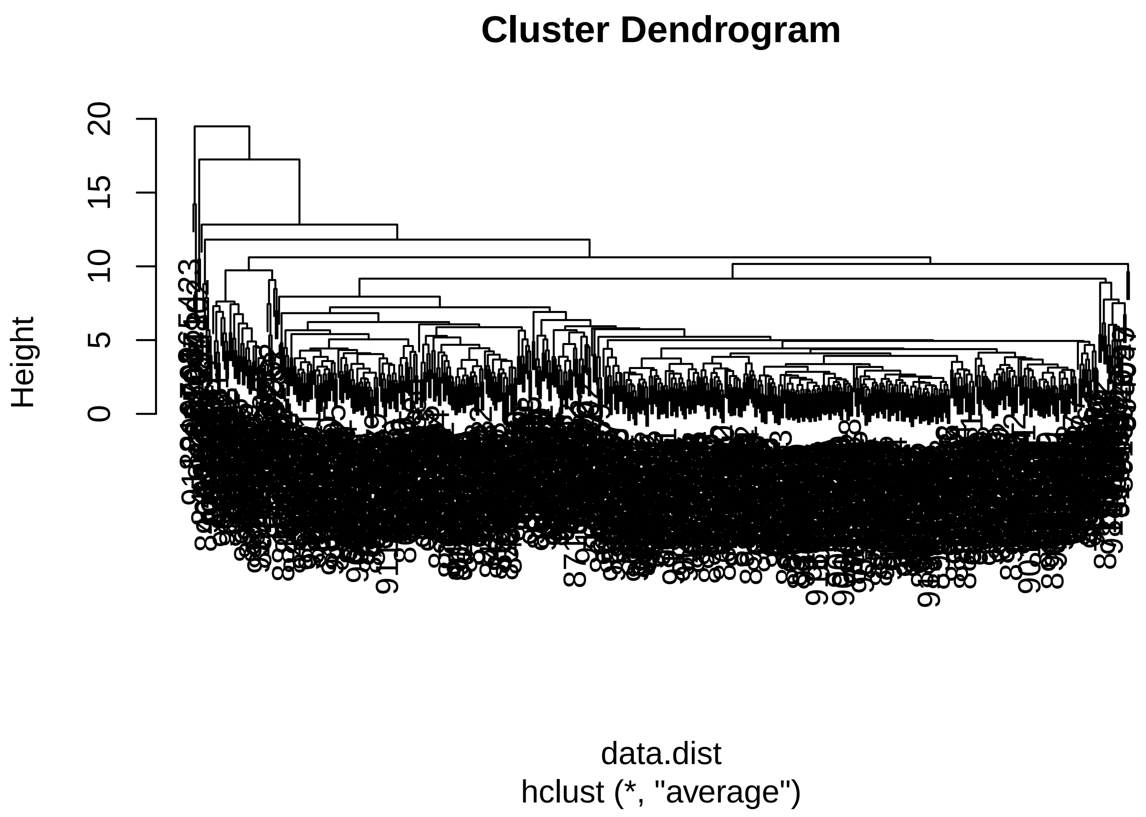
```
#method = single
plot(hclust(data.dist, method = "single"))
```



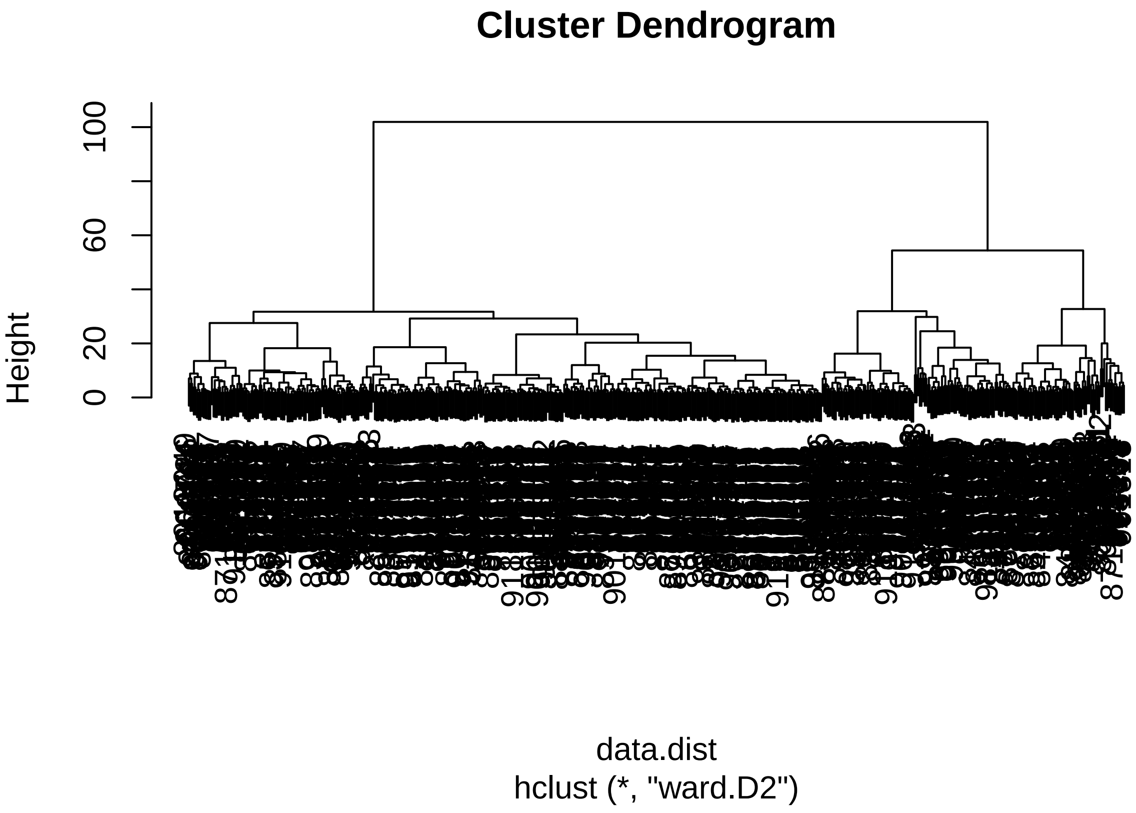
```
#method = complete
plot(wisc.hclust)
```



```
#method = average
plot(hclust(data.dist, method = "average"))
```



```
#method = ward.D2
plot(hclust(data.dist, method = "ward.D2"))
```



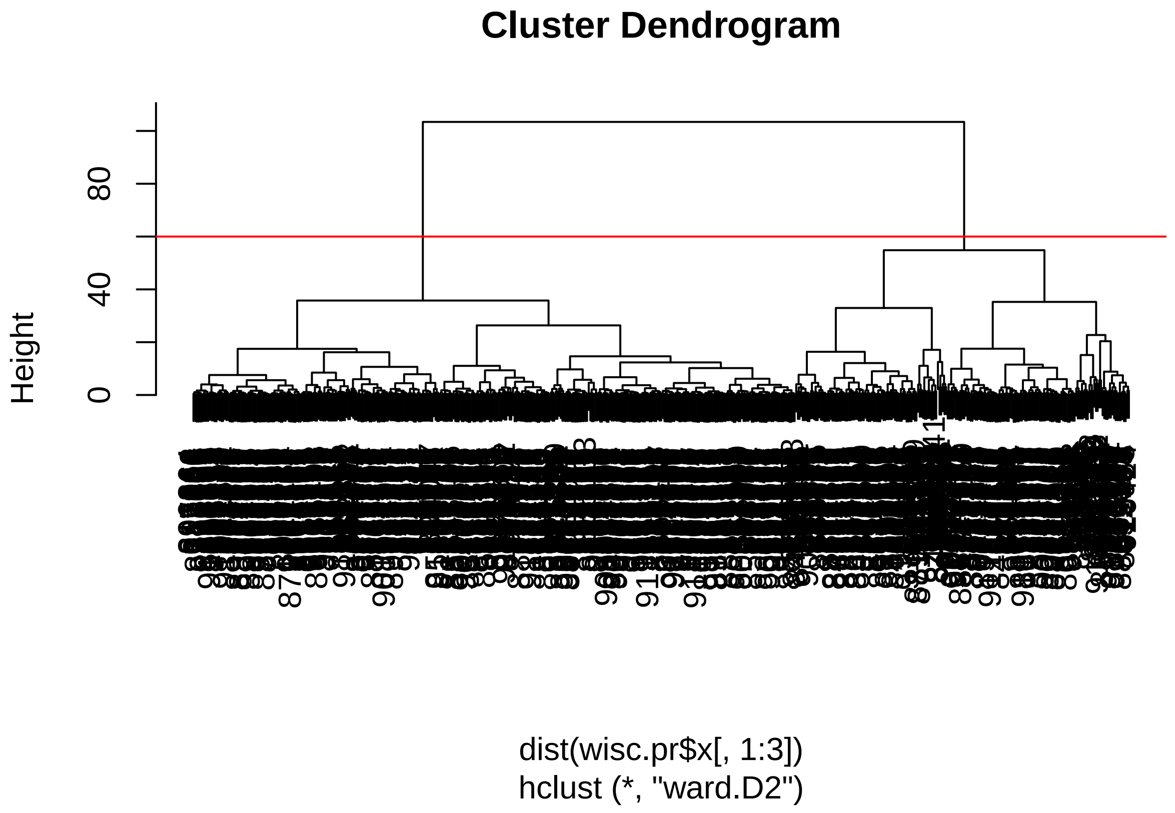
Combining Methods

We take the results of our PCA analysis and cluster in this space `wisc.pr$x`.

```
#summary(wisc.pr)
#hclust using PCA data of frist 3 PCs (70% variance reached)
wisc.pr.hclust <- hclust(dist(wisc.pr$x[,1:3]), method = "ward.D2")
head(wisc.pr$x[,1:3])
```

```
##                PC1        PC2        PC3
## 842302   -9.184755  -1.946870 -1.1221788
## 842517   -2.385703   3.764859 -0.5288274
## 84300903 -5.728855   1.074229 -0.5512625
## 84348301 -7.116691 -10.266556 -3.2299475
## 84358402 -3.931842   1.946359  1.3885450
## 843786   -2.378155  -3.946456 -2.9322967
```

```
#plot the dendrogram
plot(wisc.pr.hclust)
abline(h=60, col="red")
```



```
#cut the tree into k=2 groups
grps <- cutree(wisc.pr.hclust, k=2)
table(grps)
```

```
## grps
##   1   2
## 203 366
```

Cross table compare of diagnosis and my cluster groups

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

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

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

This method is a lot more effective. Using similar processes we can continue to determine the best method for the specific data set in real life to best analyze data (will never be perfect). These groups can be easily split into false positive, true positive, false negative, and true negative as well.

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.

The combined method is significantly better than just using hclust. We did not use kmeans so I did not compare it here.

```
#hclust + pca
table(grps, diagnosis)
```

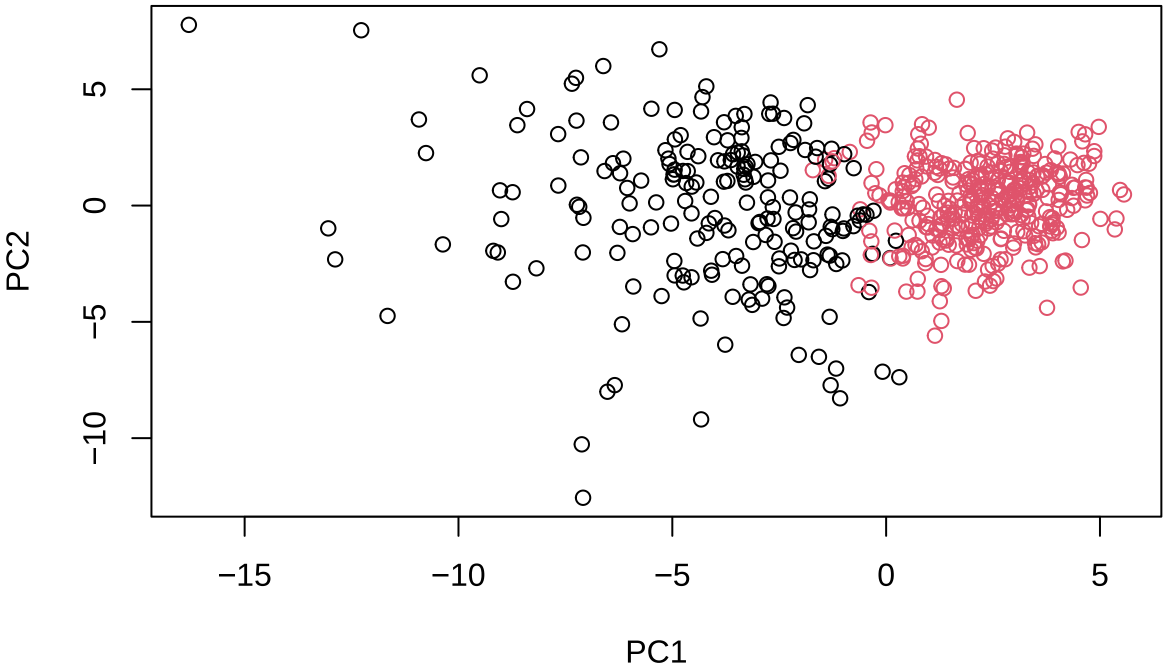
```
##     diagnosis
## grps   B   M
##    1  24 179
##    2 333  33
```

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

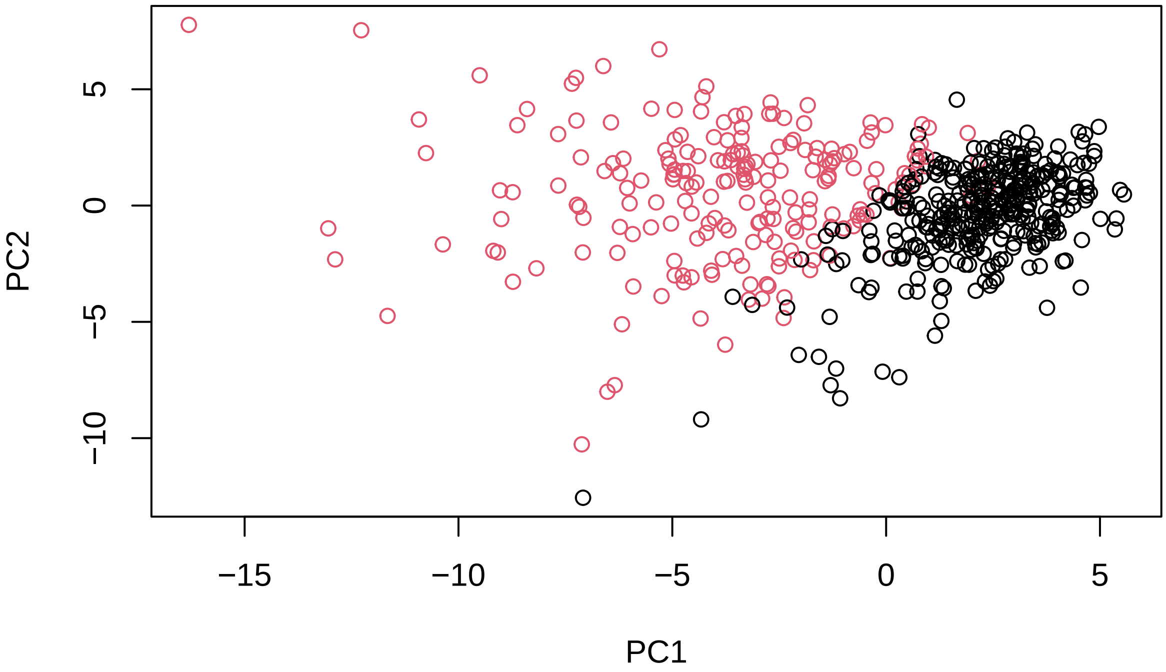
```
##                     diagnosis
## wisc.hclust.clusters   B   M
##                    1  12 165
##                    2   0   5
##                    3 343  40
##                    4   2   0
##                    5   0   2
```

Extra: Let's visualize the cluster groups/diagnosis groups.

```
plot(wisc.pr$x[,1:3], col=grps)
```



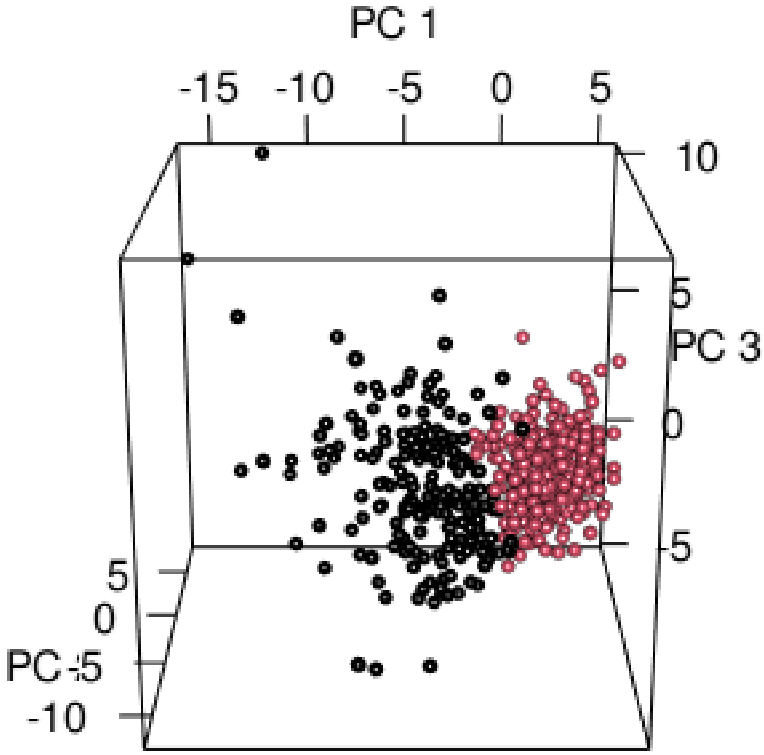
```
plot(wisc.pr$x[,1:3], col=diagnosis)
```



Try a 3D visual.

```
library(rgl)
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
```



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?***

The combined method of using hclust and pca resulted in the best sensitivity (as shown below). Just using hclust alone resulted in the best specificity. In terms of this example, I do think the sensitivity is a more important aspect.

Calculate sensitivity TP/(TP+FN):

```r
# hclust + pca = 0.844
179/(179 + 33)
```

```
## [1] 0.8443396
```

```r
# hclust = 0.804
165/(165 + 40)
```

```
## [1] 0.804878
```

Calculate specificity TN/(TN+FN):

```r
#hclust + pca = 0.932
333/(333+24)
```

```
## [1] 0.9327731
```

```r
#hclust = 0.966
343/(343 + 12)
```

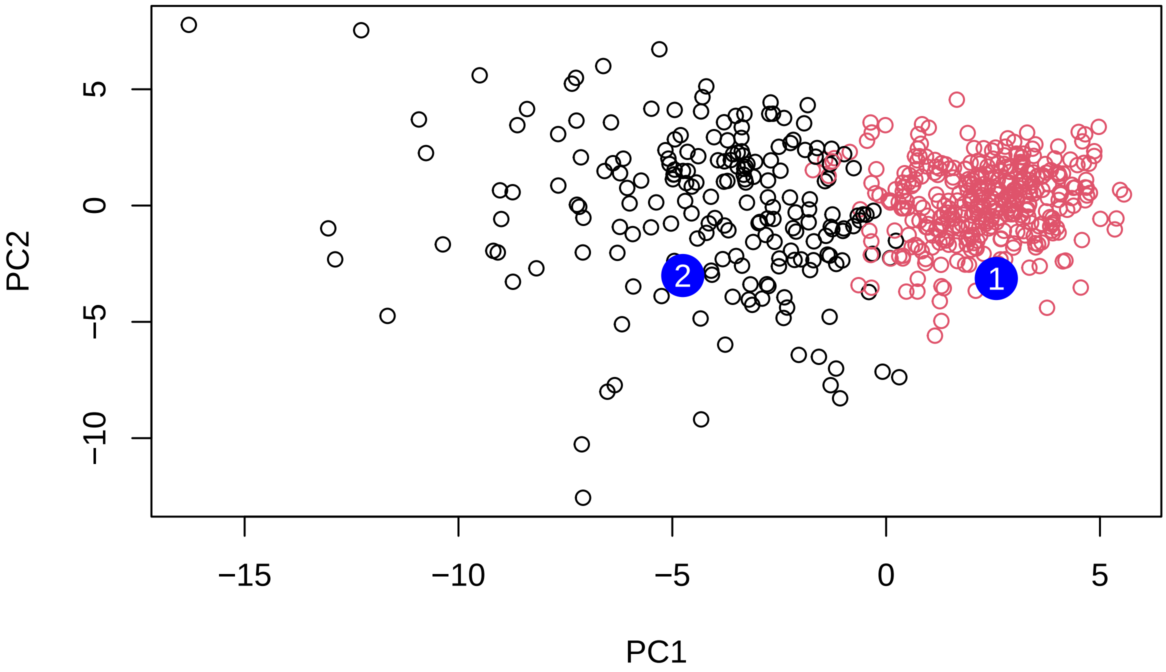
```
## [1] 0.9661972
```

Prediction

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

I would prioritize patient 2 since using the prediction, we can see that this patients results are most similar to the other results of malignant patients.