

Class 08: Unsupervised Learning Analysis of Human Breast Cells

Helen Le (PID:16300695)

1. Exploratory Data Analysis

Data input

The data is supplied on CSV format:

```
# 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					
842302		0.11890				
842517		0.08902				
84300903		0.08758				
84348301		0.17300				
84358402		0.07678				
843786		0.12440				

The `row.names` argument in the `read.csv()` function serves to make the ID numbers the row names so they're not included in the table. (think Duke incident)

Remove the diagnosis (column 1) from the data set & create a separate vector for the diagnosis for later use.

```
# Use -1 to remove the first column
wisc.data <- wisc.df[,-1]
# Create a diagnosis vector
diagnosis <- as.factor(wisc.df[,1])

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

```
head(diagnosis)
```

```
[1] M M M M M M
Levels: B M
```

Q1. How many observations are in this dataset?

```
nrow(wisc.df)
```

```
[1] 569
```

There are 569 patients in this data set.

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

```
table(diagnosis)
```

```
diagnosis
  B   M
357 212
```

212 of the observations have a malignant diagnosis.

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

```
# Use `colnames()` to access the column names in the data frame
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"

# Figure out which columns contain the suffix & assign it to a vector
suffix <- grep(pattern="_mean", x=colnames(wisc.df))

# Find the length of the vector for the total number
length((suffix))

[1] 10
```

10 of the variables are suffixed with “`_mean`”.

2. Principal Component Analysis

We need to scale our input data before PCA as some of the columns are measured in terms of very different units with different means and different variances. The upshot here is we set `scale=TRUE` argument to `prcomp()`.

The `scale` argument scales the variables by their standard deviations, which is important for variables on different scales, making their overall scales more comparable for PCA.

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

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	3.6444	2.3857	1.67867	1.40735	1.28403	1.09880	0.82172
Proportion of Variance	0.4427	0.1897	0.09393	0.06602	0.05496	0.04025	0.02251
Cumulative Proportion	0.4427	0.6324	0.72636	0.79239	0.84734	0.88759	0.91010
	PC8	PC9	PC10	PC11	PC12	PC13	PC14
Standard deviation	0.69037	0.6457	0.59219	0.5421	0.51104	0.49128	0.39624
Proportion of Variance	0.01589	0.0139	0.01169	0.0098	0.00871	0.00805	0.00523
Cumulative Proportion	0.92598	0.9399	0.95157	0.9614	0.97007	0.97812	0.98335
	PC15	PC16	PC17	PC18	PC19	PC20	PC21
Standard deviation	0.30681	0.28260	0.24372	0.22939	0.22244	0.17652	0.1731
Proportion of Variance	0.00314	0.00266	0.00198	0.00175	0.00165	0.00104	0.0010
Cumulative Proportion	0.98649	0.98915	0.99113	0.99288	0.99453	0.99557	0.9966
	PC22	PC23	PC24	PC25	PC26	PC27	PC28
Standard deviation	0.16565	0.15602	0.1344	0.12442	0.09043	0.08307	0.03987
Proportion of Variance	0.00091	0.00081	0.0006	0.00052	0.00027	0.00023	0.00005
Cumulative Proportion	0.99749	0.99830	0.9989	0.99942	0.99969	0.99992	0.99997
	PC29	PC30					
Standard deviation	0.02736	0.01153					
Proportion of Variance	0.00002	0.00000					
Cumulative Proportion	1.00000	1.00000					

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

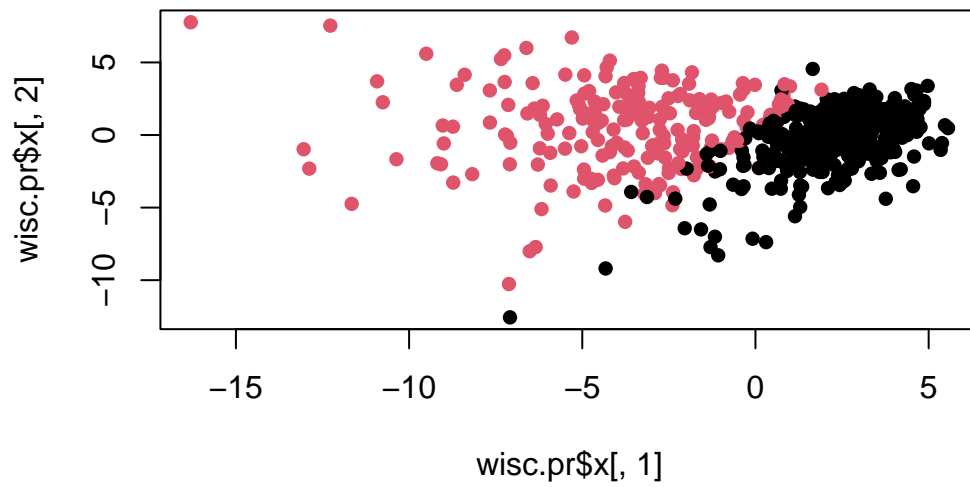
Q5. How many principal components (PCs) are required to describe at least 70% of the original variance in the data? The first 3 PCs are required to describe at least 70% of the original variance of the data.

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

8

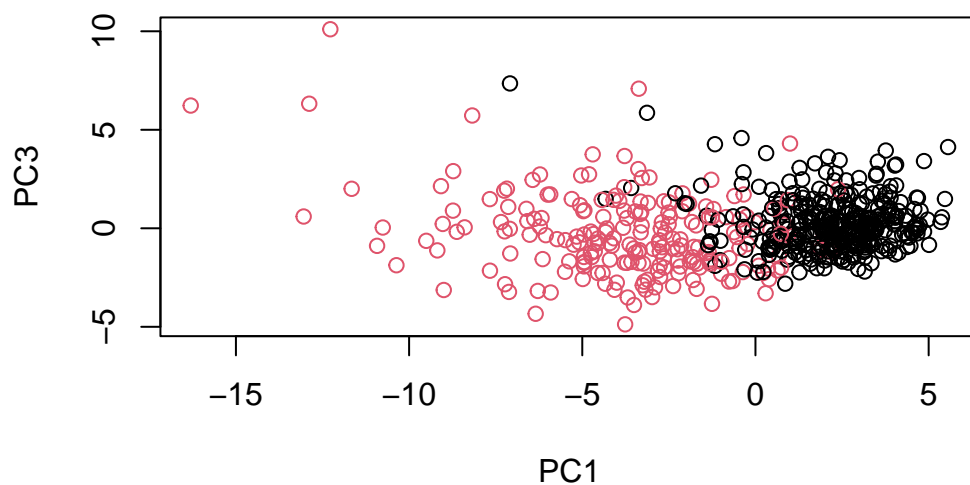
	PC13	PC14	PC15	PC16	PC17
842302	0.10329677	-0.690196797	0.601264078	0.74446075	-0.26523740
842517	-0.94269981	-0.652900844	-0.008966977	-0.64823831	-0.01719707
84300903	-0.41026561	0.016665095	-0.482994760	0.32482472	0.19075064
84348301	-0.93245070	-0.486988399	0.168699395	0.05132509	0.48220960
84358402	0.38760691	-0.538706543	-0.310046684	-0.15247165	0.13302526
843786	-0.02625135	0.003133944	-0.178447576	-0.01270566	0.19671335
	PC18	PC19	PC20	PC21	PC22
842302	-0.54907956	0.1336499	0.34526111	0.096430045	-0.06878939
842517	0.31801756	-0.2473470	-0.11403274	-0.077259494	0.09449530
84300903	-0.08789759	-0.3922812	-0.20435242	0.310793246	0.06025601
84348301	-0.03584323	-0.0267241	-0.46432511	0.433811661	0.20308706
84358402	-0.01869779	0.4610302	0.06543782	-0.116442469	0.01763433
843786	-0.29727706	-0.1297265	-0.07117453	-0.002400178	0.10108043
	PC23	PC24	PC25	PC26	PC27
842302	0.08444429	0.175102213	0.150887294	-0.201326305	-0.25236294
842517	-0.21752666	-0.011280193	0.170360355	-0.041092627	0.18111081
84300903	-0.07422581	-0.102671419	-0.171007656	0.004731249	0.04952586
84348301	-0.12399554	-0.153294780	-0.077427574	-0.274982822	0.18330078
84358402	0.13933105	0.005327110	-0.003059371	0.039219780	0.03213957
843786	0.03344819	-0.002837749	-0.122282765	-0.030272333	-0.08438081
	PC28	PC29	PC30		
842302	-0.0338846387	0.045607590	0.0471277407		
842517	0.0325955021	-0.005682424	0.0018662342		
84300903	0.0469844833	0.003143131	-0.0007498749		
84348301	0.0424469831	-0.069233868	0.0199198881		
84358402	-0.0347556386	0.005033481	-0.0211951203		
843786	0.0007296587	-0.019703996	-0.0034564331		

```
plot(wisc.pr$x[,1], wisc.pr$x[,2], col=diagnosis, pch=16)
```

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

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



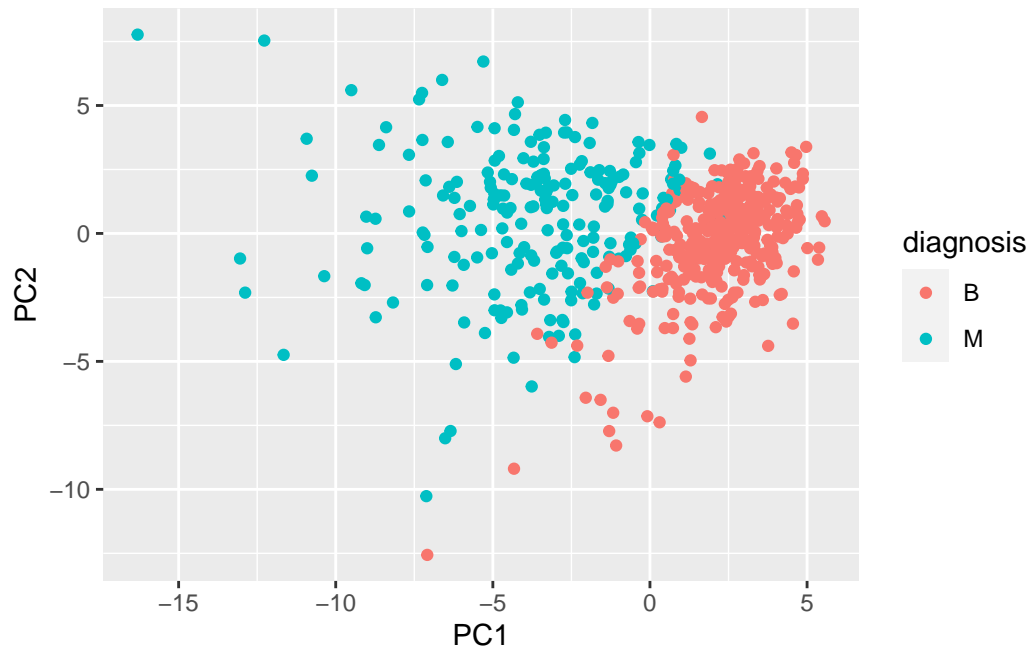
In this plot between PC1 and PC3, it's harder to determine a hard cut-off to separate the two subgroups.

Let's view this 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()
```



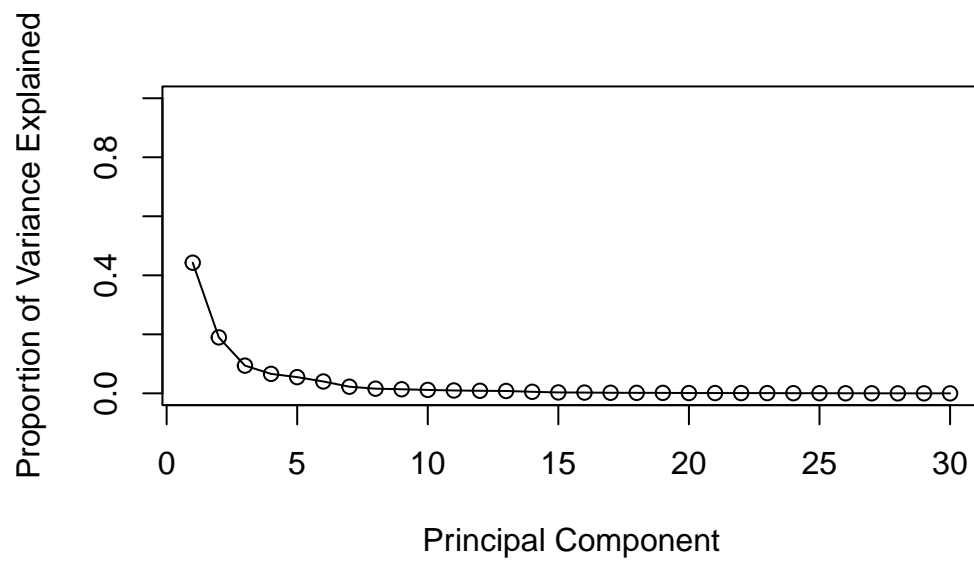
Variance Explained

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

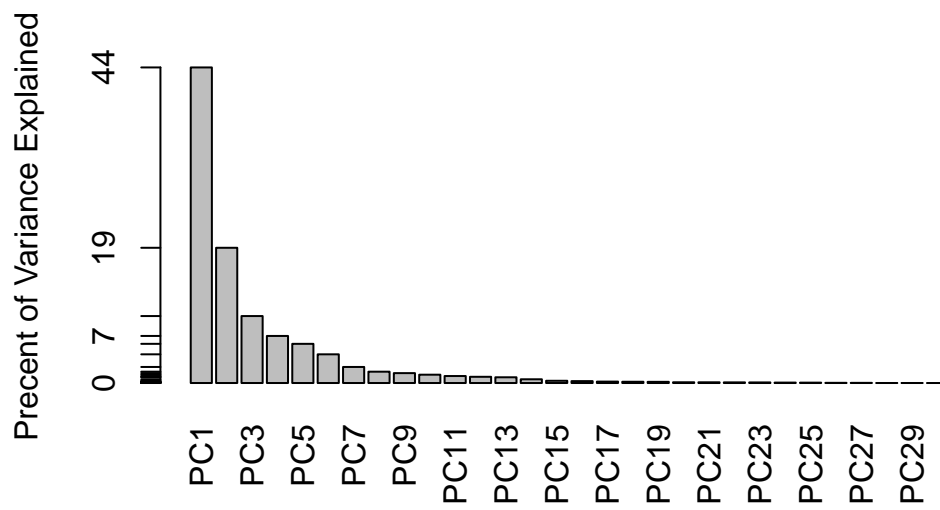
```
[1] 13.281608  5.691355  2.817949  1.980640  1.648731  1.207357
```

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

# 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 )
```

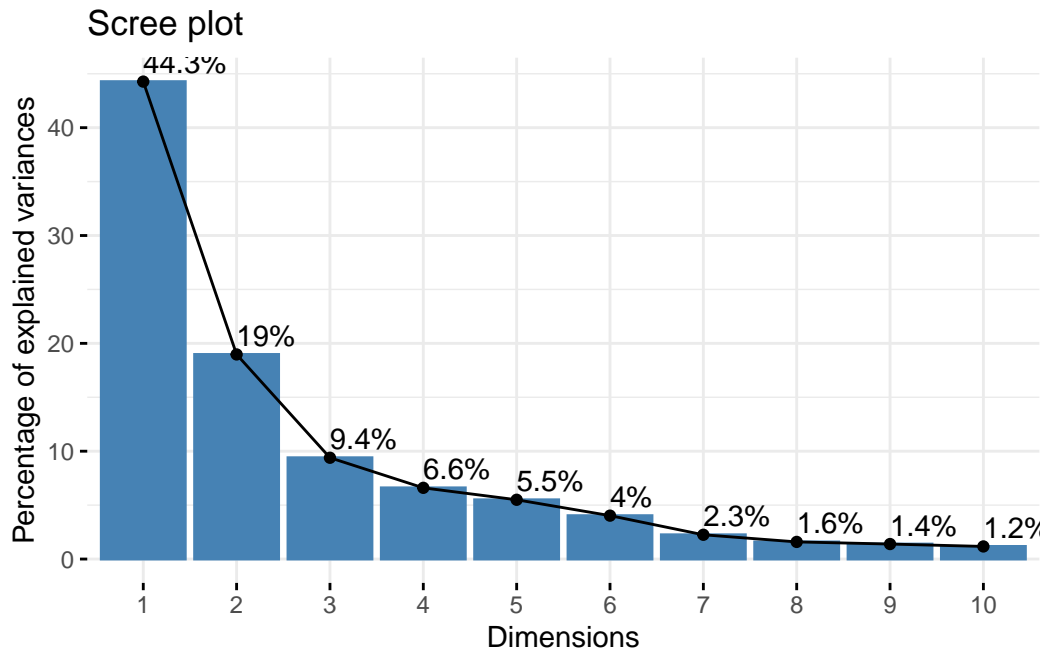


This is another way to visualize the amount of variance between the difference principal components.

```
## ggplot based graph
# 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)
```



Opened a lot of package but this is another way to visualize the variance.

Q9. For the first principal component, what is the component of the loading vector (i.e. `wisc.pr$rotation[,1]`) for the feature `concave.points_mean`?

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

radius_mean	texture_mean	perimeter_mean
-0.21890244	-0.10372458	-0.22753729
area_mean	smoothness_mean	compactness_mean
-0.22099499	-0.14258969	-0.23928535
concavity_mean	concave.points_mean	symmetry_mean
-0.25840048	-0.26085376	-0.13816696
fractal_dimension_mean	radius_se	texture_se
-0.06436335	-0.20597878	-0.01742803
perimeter_se	area_se	smoothness_se
-0.21132592	-0.20286964	-0.01453145
compactness_se	concavity_se	concave.points_se
-0.17039345	-0.15358979	-0.18341740
symmetry_se	fractal_dimension_se	radius_worst
-0.04249842	-0.10256832	-0.22799663
texture_worst	perimeter_worst	area_worst

-0.10446933	-0.23663968	-0.22487053
smoothness_worst	compactness_worst	concavity_worst
-0.12795256	-0.21009588	-0.22876753
concave.points_worst	symmetry_worst	fractal_dimension_worst
-0.25088597	-0.12290456	-0.13178394

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

```
[1] -0.2608538
```

The component of the loading vector for the feature concave.points_mean is -0.2608538.

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

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

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	3.6444	2.3857	1.67867	1.40735	1.28403	1.09880	0.82172
Proportion of Variance	0.4427	0.1897	0.09393	0.06602	0.05496	0.04025	0.02251
Cumulative Proportion	0.4427	0.6324	0.72636	0.79239	0.84734	0.88759	0.91010
	PC8	PC9	PC10	PC11	PC12	PC13	PC14
Standard deviation	0.69037	0.6457	0.59219	0.5421	0.51104	0.49128	0.39624
Proportion of Variance	0.01589	0.0139	0.01169	0.0098	0.00871	0.00805	0.00523
Cumulative Proportion	0.92598	0.9399	0.95157	0.9614	0.97007	0.97812	0.98335
	PC15	PC16	PC17	PC18	PC19	PC20	PC21
Standard deviation	0.30681	0.28260	0.24372	0.22939	0.22244	0.17652	0.1731
Proportion of Variance	0.00314	0.00266	0.00198	0.00175	0.00165	0.00104	0.0010
Cumulative Proportion	0.98649	0.98915	0.99113	0.99288	0.99453	0.99557	0.9966
	PC22	PC23	PC24	PC25	PC26	PC27	PC28
Standard deviation	0.16565	0.15602	0.1344	0.12442	0.09043	0.08307	0.03987
Proportion of Variance	0.00091	0.00081	0.0006	0.00052	0.00027	0.00023	0.00005
Cumulative Proportion	0.99749	0.99830	0.9989	0.99942	0.99969	0.99992	0.99997
	PC29	PC30					
Standard deviation	0.02736	0.01153					
Proportion of Variance	0.00002	0.00000					
Cumulative Proportion	1.00000	1.00000					

5 is the minimum number of PCs required to explain 80% of the variance of the data.

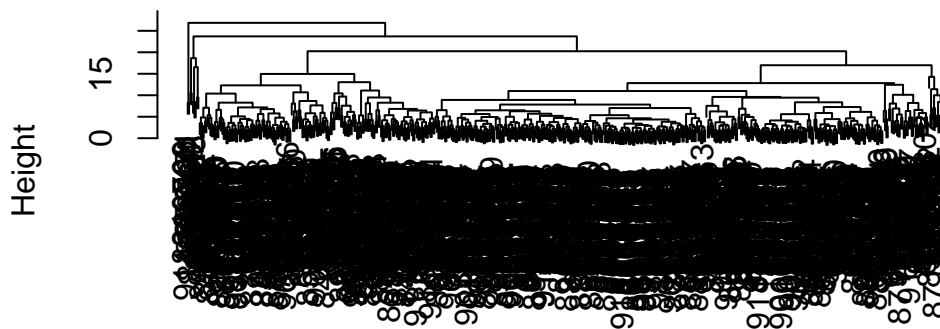
3. Hierarchical Clustering

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

# Calculate the distance between the pairs in the newly scaled dataset
data.dist <- dist(data.scaled)

# Create & plot a hierarchical clustering model using complete linkage
wisc.hclust <- hclust(data.dist)
plot(wisc.hclust)
```

Cluster Dendrogram

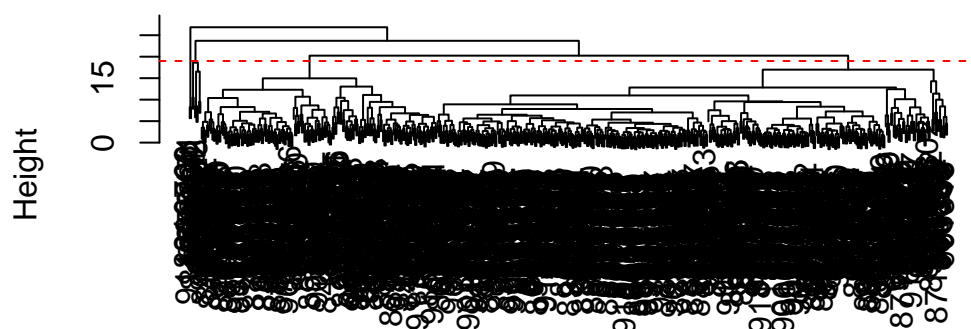


```
data.dist
hclust (*, "complete")
```

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

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


Cluster Dendrogram



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

The height at which the clustering model has 4 clusters is $h=19$.

Selecting number of clusters

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

# Compare 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

Here, we see that cluster 1 mainly consists of malignant cells while cluster 3 consists of benign cells.

Q12. Can you find a better cluster vs diagnoses match by cutting into a different number of clusters between 2 and 10? Yes, depending on the data, different number of

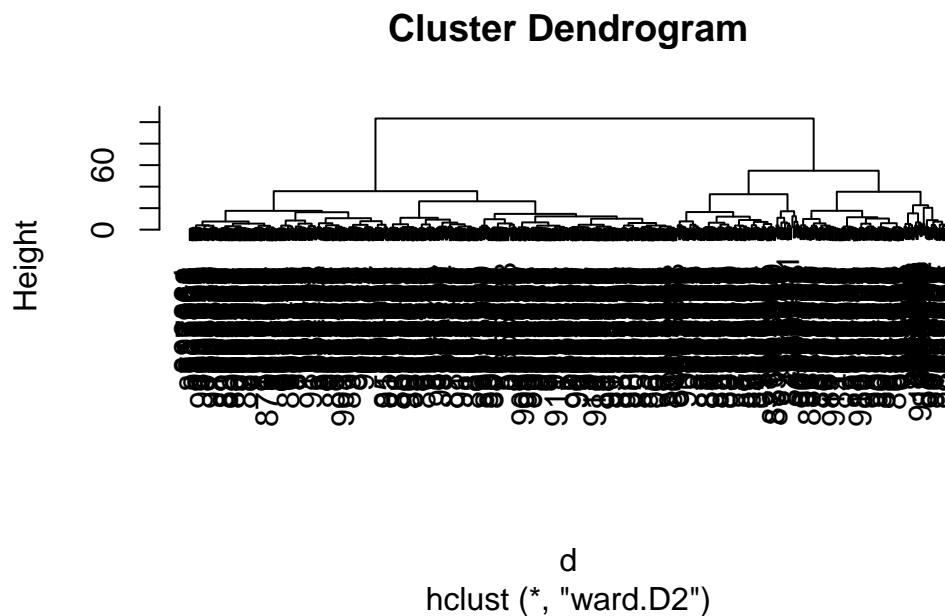
clusters may be suitable for matching the cluster vs. diagnoses. In this case, having a higher amount of clusters may be beneficial as it could produce results that are closer to the true spread of diagnoses in the patients. The incorporation of more clusters may help in making the clusters more homogeneous and better separate the subgroups.

Q13. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning. For the same data.dist dataset, the `ward.D2` method gives my favorite results since it accounts for the within cluster variance, and produces clusters of roughly equal sizes.

5. Combining Methods

This approach will take not original data but our PCA results and work with them.

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



Generate 2 cluster groups from this hclust object.

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

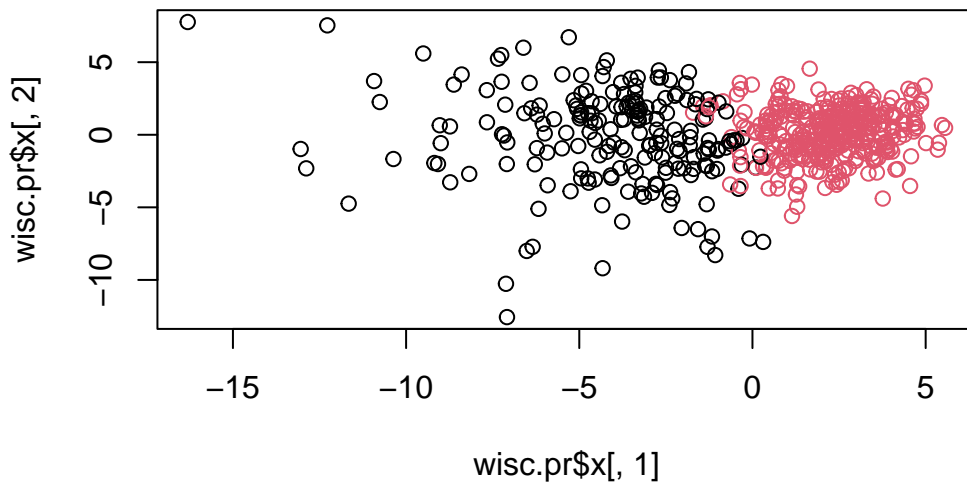
842302	842517	84300903	84348301	84358402	843786	844359	84458202
1	1	1	1	1	1	1	1
844981	84501001	845636	84610002	846226	846381	84667401	84799002
1	1	2	1	1	2	1	1
848406	84862001	849014	8510426	8510653	8510824	8511133	851509
2	1	1	2	2	2	1	1
852552	852631	852763	852781	852973	853201	853401	853612
1	1	1	1	1	2	1	1
85382601	854002	854039	854253	854268	854941	855133	855138
1	1	1	1	1	2	2	1
855167	855563	855625	856106	85638502	857010	85713702	85715
2	1	1	1	2	1	2	1
857155	857156	857343	857373	857374	857392	857438	85759902
2	2	2	2	2	1	2	2
857637	857793	857810	858477	858970	858981	858986	859196
1	1	2	2	2	2	1	2
85922302	859283	859464	859465	859471	859487	859575	859711
1	1	2	2	1	2	1	1
859717	859983	8610175	8610404	8610629	8610637	8610862	8610908
1	2	2	2	2	1	1	2
861103	8611161	8611555	8611792	8612080	8612399	86135501	86135502
2	1	1	1	2	1	2	1
861597	861598	861648	861799	861853	862009	862028	86208
2	1	2	2	2	2	1	1
86211	862261	862485	862548	862717	862722	862965	862980
2	2	2	1	2	2	2	2
862989	863030	863031	863270	86355	864018	864033	86408
2	1	2	2	1	2	2	2
86409	864292	864496	864685	864726	864729	864877	865128
1	2	2	2	2	1	1	2
865137	86517	865423	865432	865468	86561	866083	866203
2	1	1	2	2	2	2	1
866458	866674	866714	8670	86730502	867387	867739	868202
1	1	2	1	1	2	1	2
868223	868682	868826	868871	868999	869104	869218	869224
2	2	1	2	2	2	2	2
869254	869476	869691	86973701	86973702	869931	871001501	871001502
2	2	1	2	2	2	2	1
8710441	87106	8711002	8711003	8711202	8711216	871122	871149
1	2	2	2	1	2	2	2
8711561	8711803	871201	8712064	8712289	8712291	87127	8712729
2	1	1	2	1	2	2	2
8712766	8712853	87139402	87163	87164	871641	871642	872113

1	2	2	2	1	2	2	2
872608	87281702	873357	873586	873592	873593	873701	873843
1	1	2	2	1	1	1	2
873885	874158	874217	874373	874662	874839	874858	875093
2	2	2	2	2	2	1	2
875099	875263	87556202	875878	875938	877159	877486	877500
2	1	1	2	1	1	1	1
877501	877989	878796	87880	87930	879523	879804	879830
2	1	1	1	2	2	2	2
8810158	8810436	881046502	8810528	8810703	881094802	8810955	8810987
1	2	1	2	1	1	1	1
8811523	8811779	8811842	88119002	8812816	8812818	8812844	8812877
2	2	1	1	2	2	2	1
8813129	88143502	88147101	88147102	88147202	881861	881972	88199202
2	2	2	2	2	1	1	2
88203002	88206102	882488	88249602	88299702	883263	883270	88330202
2	1	2	2	1	1	2	1
88350402	883539	883852	88411702	884180	884437	884448	884626
2	2	1	2	1	2	2	1
88466802	884689	884948	88518501	885429	8860702	886226	886452
2	2	1	2	1	1	1	1
88649001	886776	887181	88725602	887549	888264	888570	889403
1	1	1	1	1	2	1	2
889719	88995002	8910251	8910499	8910506	8910720	8910721	8910748
1	1	2	2	2	2	2	2
8910988	8910996	8911163	8911164	8911230	8911670	8911800	8911834
1	2	2	2	2	2	2	2
8912049	8912055	89122	8912280	8912284	8912521	8912909	8913
1	2	1	1	2	2	2	2
8913049	89143601	89143602	8915	891670	891703	891716	891923
1	2	1	2	2	2	2	2
891936	892189	892214	892399	892438	892604	89263202	892657
2	2	2	2	1	2	1	2
89296	893061	89344	89346	893526	893548	893783	89382601
2	2	2	2	2	2	2	2
89382602	893988	894047	894089	894090	894326	894329	894335
2	2	2	2	2	1	1	2
894604	894618	894855	895100	89511501	89511502	89524	895299
2	1	2	1	2	2	2	2
8953902	895633	896839	896864	897132	897137	897374	89742801
1	1	1	2	2	2	2	1
897604	897630	897880	89812	89813	898143	89827	898431
2	1	2	1	1	2	2	1

89864002	898677	898678	89869	898690	899147	899187	899667
2	2	2	2	2	2	2	1
899987	9010018	901011	9010258	9010259	901028	9010333	901034301
1	1	2	2	2	2	2	2
901034302	901041	9010598	9010872	9010877	901088	9011494	9011495
2	2	2	2	2	1	1	2
9011971	9012000	9012315	9012568	9012795	901288	9013005	901303
1	1	1	2	1	1	2	2
901315	9013579	9013594	9013838	901549	901836	90250	90251
1	2	2	1	2	2	2	2
902727	90291	902975	902976	903011	90312	90317302	903483
2	2	2	2	2	1	2	2
903507	903516	903554	903811	90401601	90401602	904302	904357
1	1	2	2	2	2	2	2
90439701	904647	904689	9047	904969	904971	905189	905190
1	2	2	2	2	2	2	2
90524101	905501	905502	905520	905539	905557	905680	905686
1	2	2	2	2	2	2	2
905978	90602302	906024	906290	906539	906564	906616	906878
2	1	2	2	2	1	2	2
907145	907367	907409	90745	90769601	90769602	907914	907915
2	2	2	2	2	2	1	2
908194	908445	908469	908489	908916	909220	909231	909410
1	1	2	1	2	2	2	2
909411	909445	90944601	909777	9110127	9110720	9110732	9110944
2	1	2	2	1	2	1	2
911150	911157302	9111596	9111805	9111843	911201	911202	9112085
2	1	2	1	2	2	2	2
9112366	9112367	9112594	9112712	911296201	911296202	9113156	911320501
2	2	2	2	1	1	2	2
911320502	9113239	9113455	9113514	9113538	911366	9113778	9113816
2	1	2	2	1	2	2	2
911384	9113846	911391	911408	911654	911673	911685	911916
2	2	2	2	2	2	2	1
912193	91227	912519	912558	912600	913063	913102	913505
2	2	2	2	2	1	2	1
913512	913535	91376701	91376702	914062	914101	914102	914333
2	2	2	2	1	2	2	2
914366	914580	914769	91485	914862	91504	91505	915143
1	2	1	1	2	1	2	1
915186	915276	91544001	91544002	915452	915460	91550	915664
1	1	2	2	2	1	2	2
915691	915940	91594602	916221	916799	916838	917062	917080

1	2	2	2	1	1	2	2
917092	91762702	91789	917896	917897	91805	91813701	91813702
2	1	2	2	2	2	2	2
918192	918465	91858	91903901	91903902	91930402	919537	919555
2	2	2	2	2	1	2	1
91979701	919812	921092	921362	921385	921386	921644	922296
1	2	2	2	2	1	2	2
922297	922576	922577	922840	923169	923465	923748	923780
2	2	2	2	2	2	2	2
924084	924342	924632	924934	924964	925236	925277	925291
2	2	2	2	2	2	2	2
925292	925311	925622	926125	926424	926682	926954	927241
2	2	1	1	1	1	2	1
92751							
2							

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



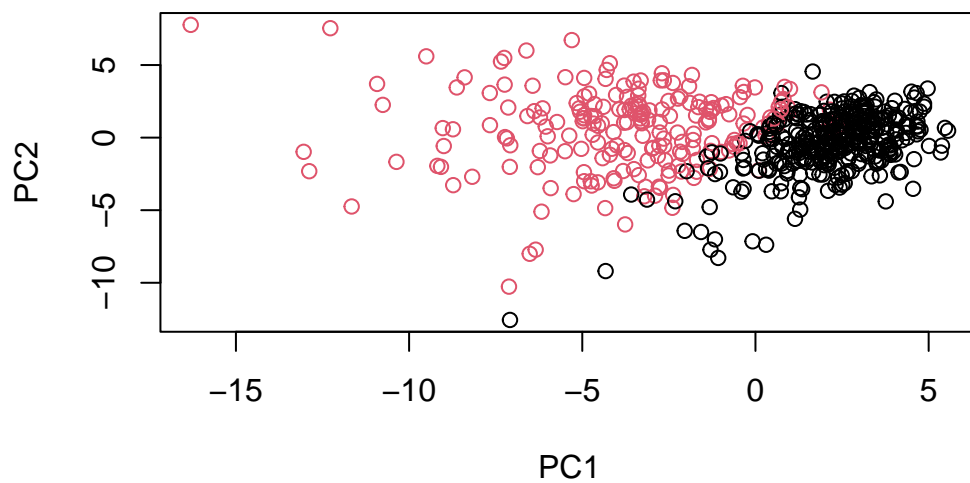
We see some overlap between the red and black data points. This is due to the 2D visualization of 3D data—there's another dimension we're overlooking.

```
table(diagnosis)
```

```
diagnosis  
  B    M  
357 212
```

To compare with the plot of diagnoses:

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



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

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

There's 24 false positives (benign in group 1) & 33 false negatives (malignant in group 2).

To make the plots match in color, the factors can be reordered:

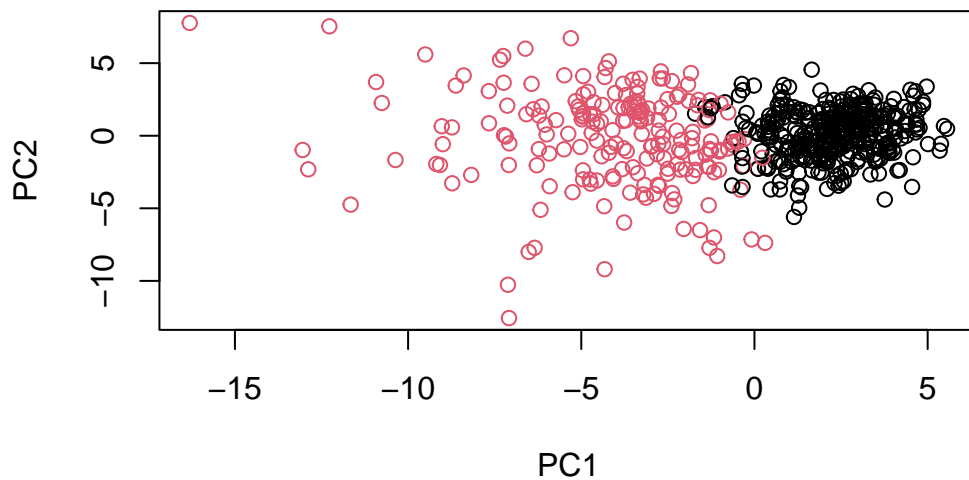
```
g <- as.factor(grps)
levels(g)
```

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

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

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

```
# Plot using our re-ordered factor
plot(wisc.pr$x[,1:2], col=g)
```



Cut hierarchical clustering model– wisc.pr.hclust.two– into 2 clusters:

```
## Use the distance along the first 7 PCs for clustering i.e. wisc.pr$x[, 1:7]
wisc.pr.hclust.two <- hclust(dist(wisc.pr$x[,1:7]), method="ward.D2")

wisc.pr.hclust.clusters <- cutree(wisc.pr.hclust.two, k=2)
```



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

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

Q15. How well does the newly created model with four clusters separate out the two diagnoses? The newly created model does similarly with the four clusters in separating out the two diagnoses. There are more true positives and less false positives. Unfortunately, there are less true negatives and more false negatives.