

Unsupervised Machine Learning Project

Tusha Karnani

Table of contents

Background	1
Data import	2
Principal Component Analysis	5
Performing PCA	7
Visualizing PCA results	8
Hierarchical clustering	11
Combining methods	17
Clustering based on PCA results	17
Specificity and Sensitivity	18
Predictions	19

Background

The goal of this mini-project is to explore a complete analysis using unsupervised learning techniques. I look at combining PCA as a pre-processing step to clustering using data that consists of measurements of cell nuclei of human breast masses. This also expands on my experience with RNA-Seq analysis.

The data itself comes from the Wisconsin Breast Cancer Diagnostic Data Set first reported by K. P. Benne and O. L. Mangasarian: “Robust Linear Programming Discrimination of Two Linearly Inseparable Sets”.

Values in this data set describe characteristics of the cell nuclei present in digitized images of a fine needle aspiration (FNA) of a breast mass.

Data import

Data was downloaded as a csv file.

```
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
842517	0.1812		0.05667	0.5435	0.7339
84300903	0.2069		0.05999	0.7456	0.7869
84348301	0.2597		0.09744	0.4956	1.1560
84358402	0.1809		0.05883	0.7572	0.7813
843786	0.2087		0.07613	0.3345	0.8902
	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			

Removing the first “diagnosis” column since I want to use unsupervised methods to classify this data.

```
wisc.data <- 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				

Saving the diagnosis column so we can refer to it later.

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

How many observations are in this dataset?

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

```
[1] 569
```

How many of the observations have a malignant diagnosis?

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

```
[1] 212
```

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

```
length(grep("_mean", colnames(wisc.data))) # returns column numbers of the column names with
```

```
[1] 10
```

```
grep("_mean", colnames(wisc.data), value=TRUE) # returns column names with "_mean"
```

```
[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"
```

Principal Component Analysis

Need to do EDA to check if the data is scaled.

```
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

```
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

Performing PCA

Note: The `prcomp()` function has a `scale=FALSE` default. We nearly always want to set this as `TRUE` since we don't want our analysis to be dominated by columns/variables in our data set that have high standard deviation and mean when compared to others simply because the measurements are on different units/scales.

```
# scaling makes the sd=1
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					

About 44.27% of the original variance is captured by PC1.

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

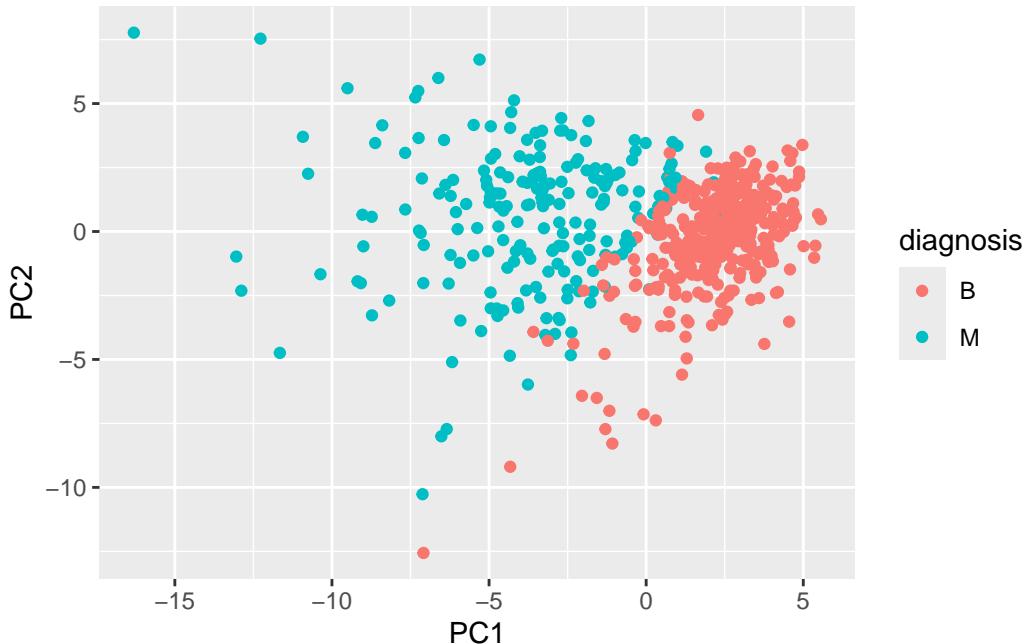
How many principal components (PCs) are required to describe at least 90% of the original variance in the data?

7

Visualizing PCA results

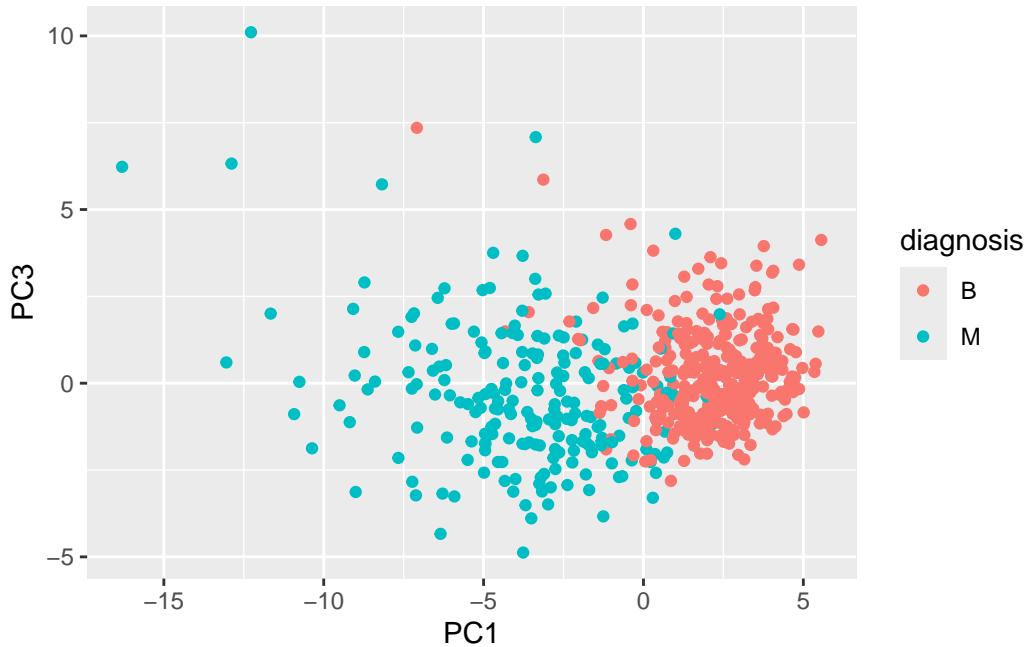
```
library(ggplot2)

ggplot(wisc.pr$x) +
  aes(PC1, PC2, col=diagnosis) +
  geom_point()
```



```
ggplot(wisc.pr$x) +
  aes(PC1, PC3, col=diagnosis) +
  geom_point()
```

8



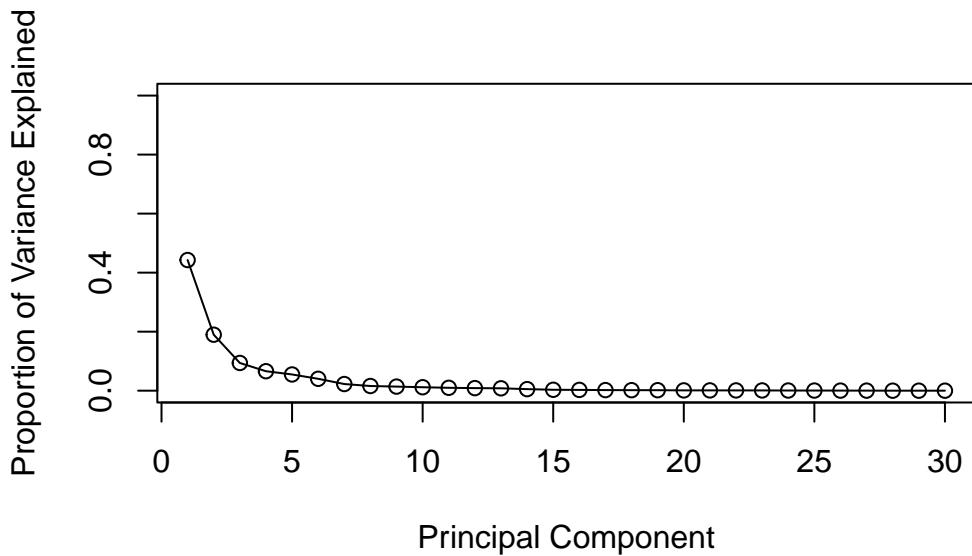
Since PC2 explains more variance than PC3, the groups are more separated in the first plot

```
pr.var <- wisc.pr$sdev^2
head(pr.var)
```

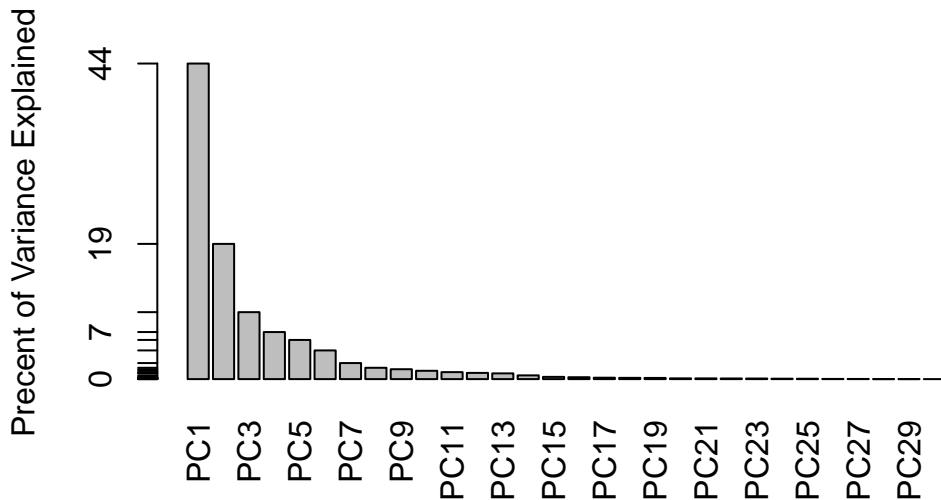
```
[1] 13.281608 5.691355 2.817949 1.980640 1.648731 1.207357
```

```
# Variance explained by each principal component: pve
pve <- wisc.pr$sdev^2 / sum(wisc.pr$sdev^2)

# 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 = "Percent of Variance Explained",
         names.arg=paste0("PC",1:length(pve)), las=2, axes = FALSE)
axis(2, at=pve, labels=round(pve,2)*100 )
```



```
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

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

data.dist <- dist(data.scaled)

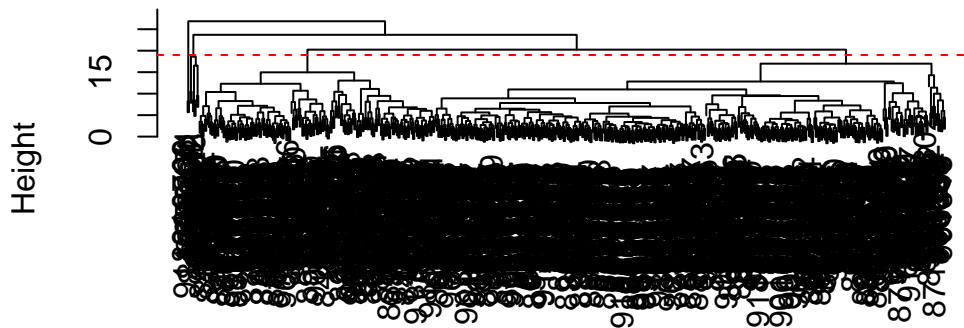
wisc.hclust <- hclust(data.dist, method="complete")
wisc.hclust
```

Call:
`hclust(d = data.dist, method = "complete")`

Cluster method : complete
Distance : euclidean
Number of objects: 569

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

Cluster Dendrogram



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

For 4 clusters, we need to cut it off at $h \sim 19$.

```
wisc.hclust.clusters <- cutree(wisc.hclust, h=19)  
table(wisc.hclust.clusters, diagnosis)
```

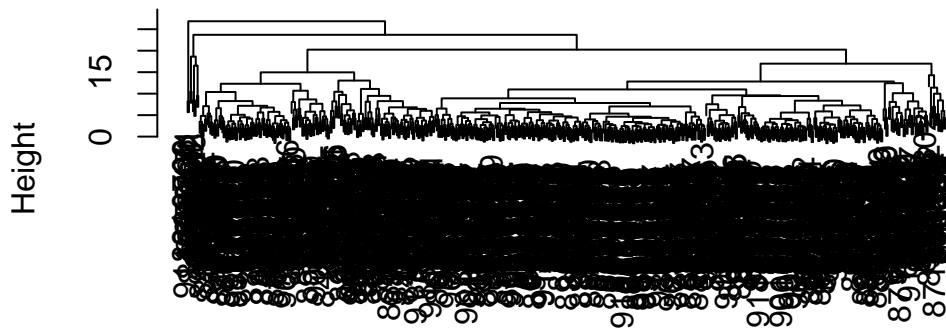
diagnosis		
wisc.hclust.clusters	B	M
1	12	165
2	2	5
3	343	40
4	0	2

For the above, Group 1 seems to be malignant patients and Group 3 seems to be benign patients.

I noted that creating either 4 or 8 groups seemed to be the best choices for cluster vs diagnoses matches.

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

Cluster Dendrogram



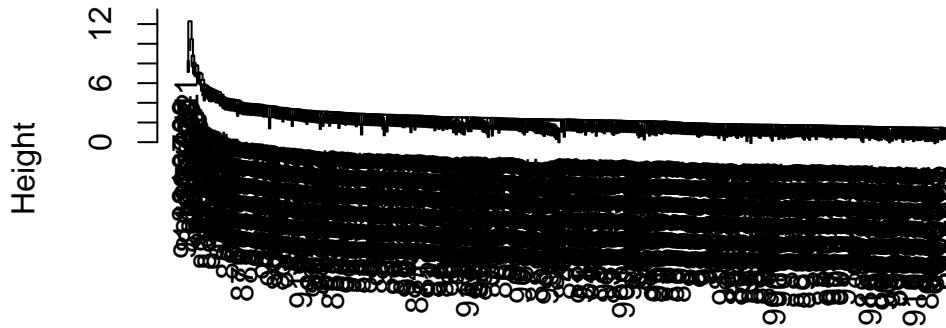
```
data.dist  
hclust (*, "complete")
```

```
wisc.hclust.clusters.sample <- cutree(wisc.hclust.sample, k=8)  
table(wisc.hclust.clusters.sample, diagnosis)
```

wisc.hclust.clusters.sample	B	M	diagnosis
1	12	86	
2	0	79	
3	0	3	
4	331	39	
5	2	0	
6	12	1	
7	0	2	
8	0	2	

```
wisc.hclust.sample1 <- hclust(data.dist, method="single")  
plot(wisc.hclust.sample1)
```

Cluster Dendrogram



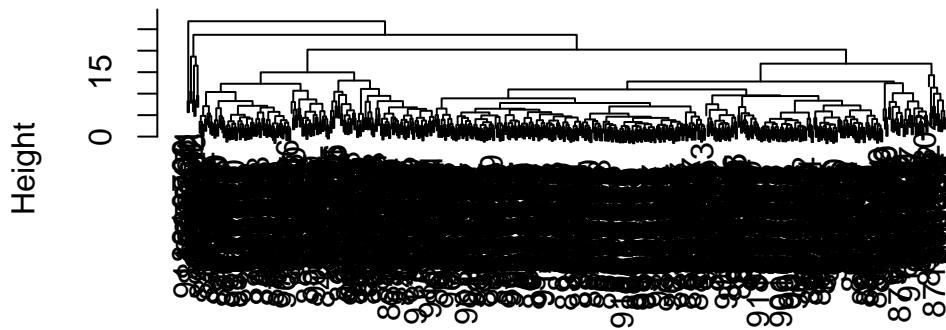
```
data.dist  
hclust (*, "single")
```

```
wisc.hclust.clusters.sample1 <- cutree(wisc.hclust.sample1, k=4)  
table(wisc.hclust.clusters.sample1, diagnosis)
```

diagnosis		
wisc.hclust.clusters.sample1	B	M
1	356	209
2	1	0
3	0	2
4	0	1

```
wisc.hclust.sample2 <- hclust(data.dist, method="complete")  
plot(wisc.hclust.sample2)
```

Cluster Dendrogram



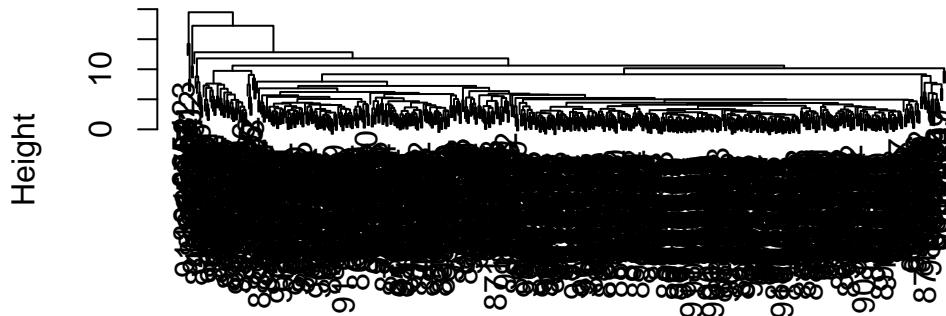
```
data.dist  
hclust (*, "complete")
```

```
wisc.hclust.clusters.sample2 <- cutree(wisc.hclust.sample2, k=4)  
table(wisc.hclust.clusters.sample2, diagnosis)
```

diagnosis		
wisc.hclust.clusters.sample2	B	M
1	12	165
2	2	5
3	343	40
4	0	2

```
wisc.hclust.sample3 <- hclust(data.dist, method="average")  
plot(wisc.hclust.sample3)
```

Cluster Dendrogram

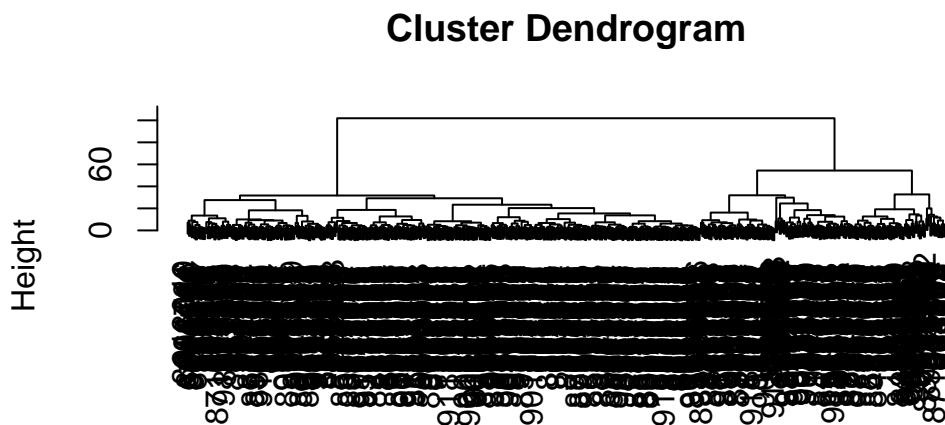


```
data.dist  
hclust (*, "average")
```

```
wisc.hclust.clusters.sample3 <- cutree(wisc.hclust.sample3, k=4)  
table(wisc.hclust.clusters.sample3, diagnosis)
```

diagnosis		
wisc.hclust.clusters.sample3	B	M
1	355	209
2	2	0
3	0	1
4	0	2

```
wisc.hclust.sample4 <- hclust(data.dist, method="ward.D2")  
plot(wisc.hclust.sample4)
```



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

```
wisc.hclust.clusters.sample4 <- cutree(wisc.hclust.sample4, k=4)
table(wisc.hclust.clusters.sample4, diagnosis)
```

```
wisc.hclust.clusters.sample4   diagnosis
                                B      M
1          0    115
2          6     48
3       337     48
4       14      1
```

Combining methods

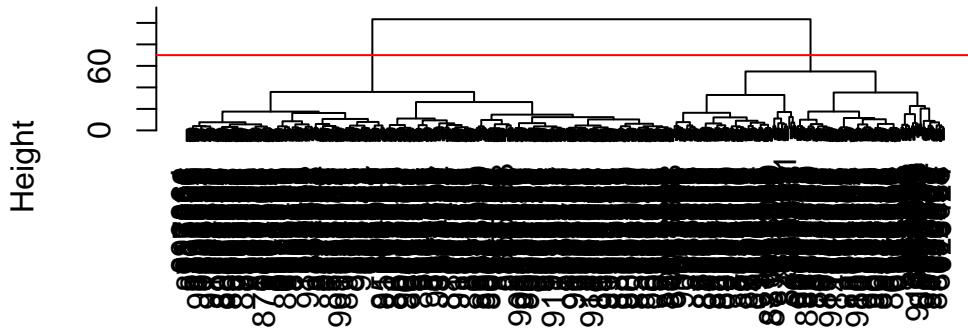
Clustering based on PCA results

Clustering the original data was not very productive. The PCA results looked promising. So clustering in PC space will help.

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

```
plot(wisc.pr.hclust)
abline(h=70, col="red")
```

Cluster Dendrogram



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

To get our clustering membership vector, we cut the tree.

```
grps <- cutree(wisc.pr.hclust, h=70)  
table(grps)
```

grps
1 2
203 366

How does this clustering compare to the actual diagnosis?

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

	diagnosis	
grps	B	M
1	24	179
2	333	33

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: $TN/(TN+FN)$.

Predictions

```
url <- "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			

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

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

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

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

```
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")
```

