

# Class 8: Breast Cancer Mini Project

Julia Di Silvestri (A16950824)

Before we get stuck into project work, we will have a quick look at applying PCA to some example RNASeq data (end of lab 7)

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

	wt1	wt2	wt3	wt4	wt5	ko1	ko2	ko3	ko4	ko5
gene1	439	458	408	429	420	90	88	86	90	93
gene2	219	200	204	210	187	427	423	434	433	426
gene3	1006	989	1030	1017	973	252	237	238	226	210
gene4	783	792	829	856	760	849	856	835	885	894
gene5	181	249	204	244	225	277	305	272	270	279
gene6	460	502	491	491	493	612	594	577	618	638

Q. How many genes are in this data set?

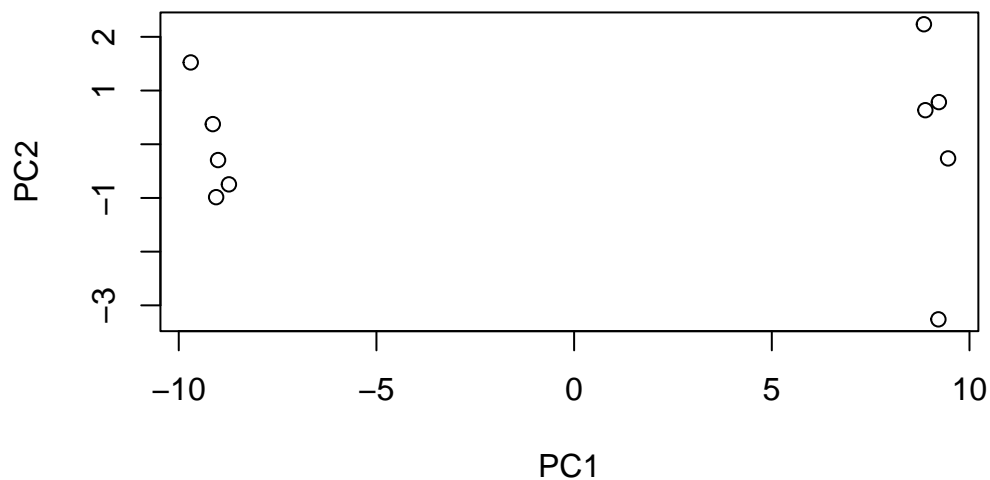
```
nrow(rna.data)
```

```
[1] 100
```

## Run PCA

```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



```
summary(pca)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
Standard deviation	9.6237	1.5198	1.05787	1.05203	0.88062	0.82545	0.80111
Proportion of Variance	0.9262	0.0231	0.01119	0.01107	0.00775	0.00681	0.00642
Cumulative Proportion	0.9262	0.9493	0.96045	0.97152	0.97928	0.98609	0.99251

	PC8	PC9	PC10
Standard deviation	0.62065	0.60342	3.345e-15
Proportion of Variance	0.00385	0.00364	0.000e+00
Cumulative Proportion	0.99636	1.00000	1.000e+00

^PC1 (only one axis) accounts for 92.62% of variance

```
pca$x
```

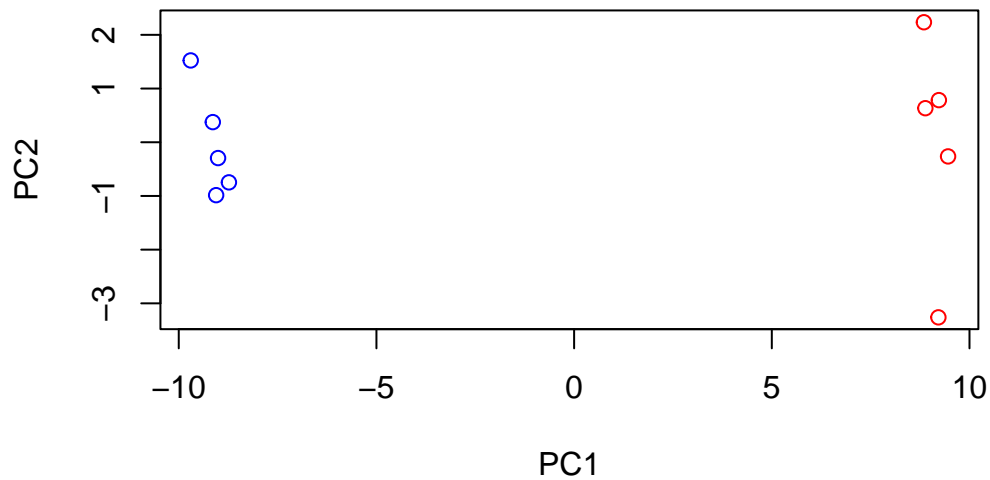
	PC1	PC2	PC3	PC4	PC5	PC6
wt1	-9.697374	1.5233313	-0.2753567	0.7322391	-0.6749398	1.1823860
wt2	-9.138950	0.3748504	1.0867958	-1.9461655	0.7571209	-0.4369228
wt3	-9.054263	-0.9855163	0.4152966	1.4166028	0.5835918	0.6937236

wt4	-8.731483	-0.7468371	0.5875748	0.2268129	-1.5404775	-1.2723618
wt5	-9.006312	-0.2945307	-1.8498101	-0.4303812	0.8666124	-0.2496025
ko1	8.846999	2.2345475	-0.1462750	-1.1544333	-0.6947862	0.7128021
ko2	9.213885	-3.2607503	0.2287292	-0.7658122	-0.4922849	0.9170241
ko3	9.458412	-0.2636283	-1.5778183	0.2433549	0.3654124	-0.5837724
ko4	8.883412	0.6339701	1.5205064	0.7760158	1.2158376	-0.1446094
ko5	9.225673	0.7845635	0.0103574	0.9017667	-0.3860869	-0.8186668
	PC7	PC8	PC9	PC10		
wt1	-0.24446614	1.03519396	0.07010231	3.388516e-15		
wt2	-0.03275370	0.26622249	0.72780448	2.996563e-15		
wt3	-0.03578383	-1.05851494	0.52979799	3.329630e-15		
wt4	-0.52795595	-0.20995085	-0.50325679	3.317526e-15		
wt5	0.83227047	-0.05891489	-0.81258430	2.712504e-15		
ko1	-0.07864392	-0.94652648	-0.24613776	2.768138e-15		
ko2	0.30945771	0.33231138	-0.08786782	3.317091e-15		
ko3	-1.43723425	0.14495188	0.56617746	3.299214e-15		
ko4	-0.35073859	0.30381920	-0.87353886	3.000948e-15		
ko5	1.56584821	0.19140827	0.62950330	2.785473e-15		

```
#we have 5 wt and 5 ko samples
mycols <- c(rep("blue",5), rep("red", 5))
mycols
```

```
[1] "blue" "blue" "blue" "blue" "blue" "red" "red" "red" "red" "red"
```

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", col = mycols)
```



I could examine which genes contribute most to this first PC

```
head(sort(abs(pca$rotation[,1]), decreasing = T))
```

gene100	gene66	gene45	gene68	gene98	gene60
0.1038708	0.1038455	0.1038402	0.1038395	0.1038372	0.1038055

## Analysis of Breast Cancer FNA data:

First we will call in and read the data

```
# Save your input data file into your Project directory
fna.data <- "https://bioboot.github.io/bimm143_S20/class-material/WisconsinCancer.csv"

# Complete the following code to input the data and store as wisc.df
wisc.df <- read.csv(fna.data, 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		

We do not want the first column because it essentially gives us the answer, so we will create a data set that omits the first column

```
# We can use -1 here to remove the first column
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			

No we will create a vector containing the data from the first column “diagnosis” to use later for plotting.

```
# Create diagnosis vector for later
diagnosis <- as.factor(wisc.df$diagnosis)
diagnosis
```

```
[1] M M M M M M M M M M M M M M M M M M B B B M M M M M M M M M M M M M M M M
```

```

[38] B M M M M M M M B M B B B B B M M B M M B B B B M B M M B B B B M B M M
[75] B M B M M B B B M M B M M M B B B M B B M M B B B M M B B B B M B B M B B
[112] B B B B B B M M M B M M B B B M M B M B M M B M M B B M B B B M B B M B
[149] B B B B B B B B M B B B B M M B M B B M M B B M M B B B B M B B M M M B M
[186] B M B B B M B B M M B M M M M B M M M B M B M B B M B M M M M B B M M B B
[223] B M B B B B B M M B B M B B M M B M B B B B M B B B B B M B M M M M M M M
[260] M M M M M M M B B B B B B M B M B B M B B M B M M B B B B B B B B B B B
[297] B M B B M B M B B B B B B B B B B B B B M B B B M B M B B B B M M M B B
[334] B B M B M B M B B B M B B B B B B B M M M B B B B B B B B B B M M B M M
[371] M B M M B B B B B M B B B B B M B B B M B B M M B B B B B B M B B B B B
[408] B M B B B B B M B B M B B B B B B B B B B B M B M M B M B B B B B M B B
[445] M B M B B M B M B B B B B B B B M M B B B B B B M B B B B B B B B B M B
[482] B B B B B B M B M B B M B B B B B M M B M B M B B B B B M B B M B M B M M
[519] B B B M B B B B B B B B B B B M B M M B B B B B B B B B B B B B B B B
[556] B B B B B B M M M M M M M B
Levels: B M

```

## Exploratory Data Analysis

Q1. How many observations are in this dataset?

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

```
[1] 569
```

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

```
table(diagnosis)
```

```
diagnosis
  B    M
357 212
```

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

```
colnames(wisc.data)
```

```

[1] "radius_mean"      "texture_mean"
[3] "perimeter_mean"   "area_mean"
[5] "smoothness_mean"  "compactness_mean"

```



```

[7] "concavity_mean"      "concave.points_mean"
[9] "symmetry_mean"       "fractal_dimension_mean"
[11] "radius_se"           "texture_se"
[13] "perimeter_se"        "area_se"
[15] "smoothness_se"       "compactness_se"
[17] "concavity_se"        "concave.points_se"
[19] "symmetry_se"         "fractal_dimension_se"
[21] "radius_worst"        "texture_worst"
[23] "perimeter_worst"     "area_worst"
[25] "smoothness_worst"    "compactness_worst"
[27] "concavity_worst"     "concave.points_worst"
[29] "symmetry_worst"      "fractal_dimension_worst"

```

```
length(grep("mean", colnames(wisc.data)))
```

```
[1] 10
```

## Performing PCA

Here we will use `prcomp()` on the `wisc.data` object – the one without the diagnosis column

First, we have to decide whether we need to scale the data or not by using `scale = T` when we run `prcomp()`

We can look at the means / standard deviations of each column. If they are similar, we are all good to go. If not, we should used `scale = T`

```
# 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

```
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

These are very different so we should scale them.

```
# Perform PCA on wisc.data by completing the following code
wisc.pr <- prcomp( wisc.data, scale = T )
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% (read from table)

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

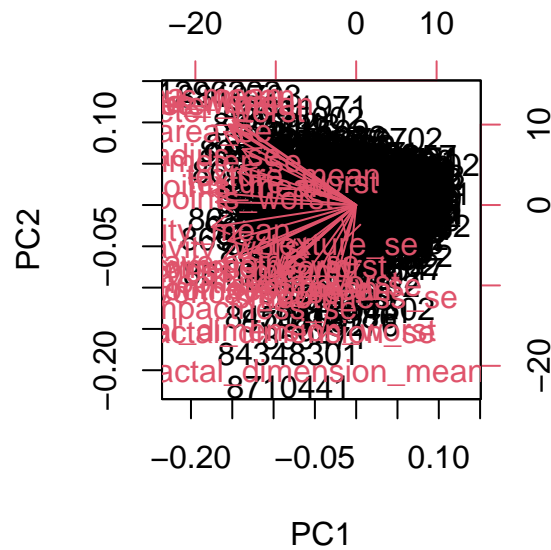
3 PCs (read from table)

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

7 PCs (read from table)

## Plotting the PCA results

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



This plot is confusing and useless. We need to build our own.

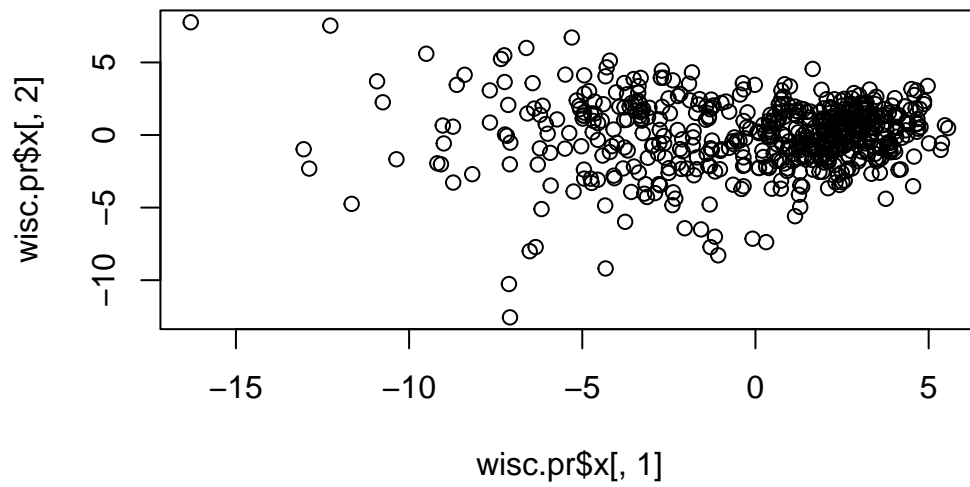
```
attributes(wisc.pr)
```

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

```
$class
[1] "prcomp"
```

```
#we want x
```

```
# Scatter plot observations by components 1 and 2
plot( wisc.pr$x[,1], wisc.pr$x[,2])
```

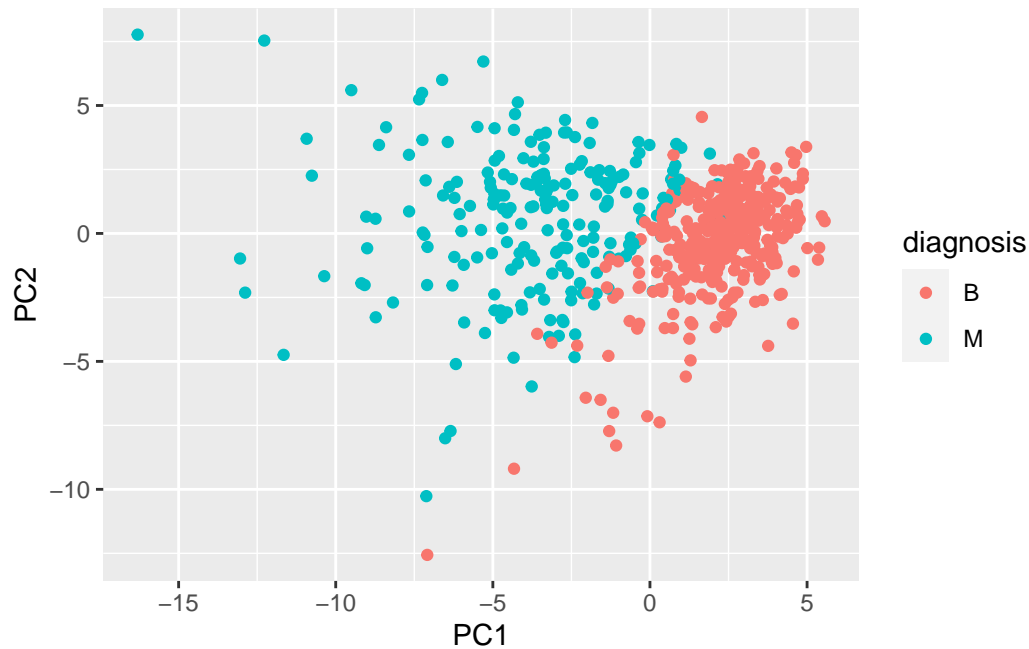


```
#can't get the col = diagnosis to work -- not sure why
```

```
library(ggplot2)
```

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

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



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["concave.points_mean",1]
```

```
[1] -0.2608538
```

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

```
tbl <- summary(wisc.pr)
which(tbl$importance[3,] > 0.8)[1]
```

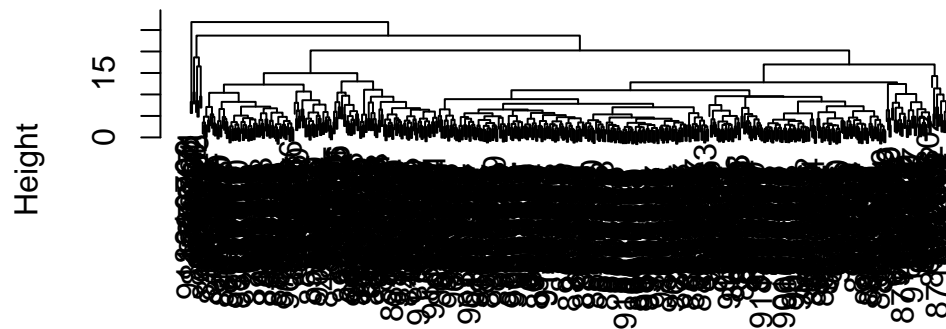
```
PC5
5
```

## Hierarchical Clustering

The main function for Hierarchical Clustering is called `hclust()` it takes a distance matrix as input

```
d <- dist(scale(wisc.data))
wisc.hclust <- hclust(d)
plot(wisc.hclust)
```

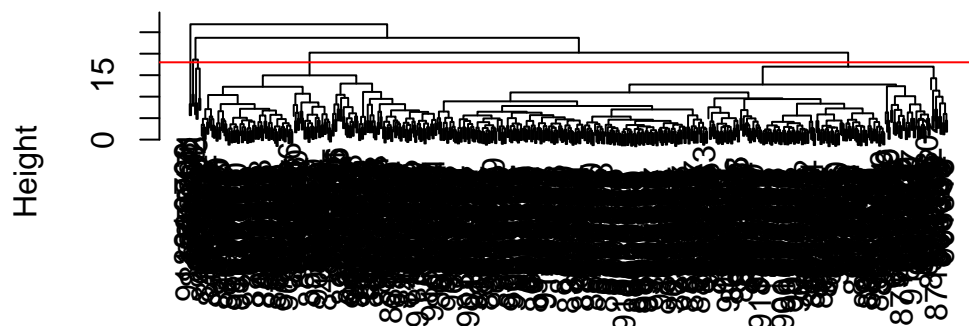
## Cluster Dendrogram



d  
hclust (\*, "complete")

```
plot(wisc.hclust)
abline(h=18, col = "red")
```

## Cluster Dendrogram



d  
hclust (\*, "complete")

```
grps <- cutree(wisc.hclust, h = 18)
table(grps)
```

grps

1	2	3	4	5
177	5	383	2	2

This method is not very helpful in giving clear groups. Instead we will use clustering PCA.

### Combining methods

Here we will perform clustering on our PCA results rather than our original data.

In other words, we will cluster using `wisc.pr$x` - our new better variables or PCs. We can choose as many or as few as we like. It is our call!

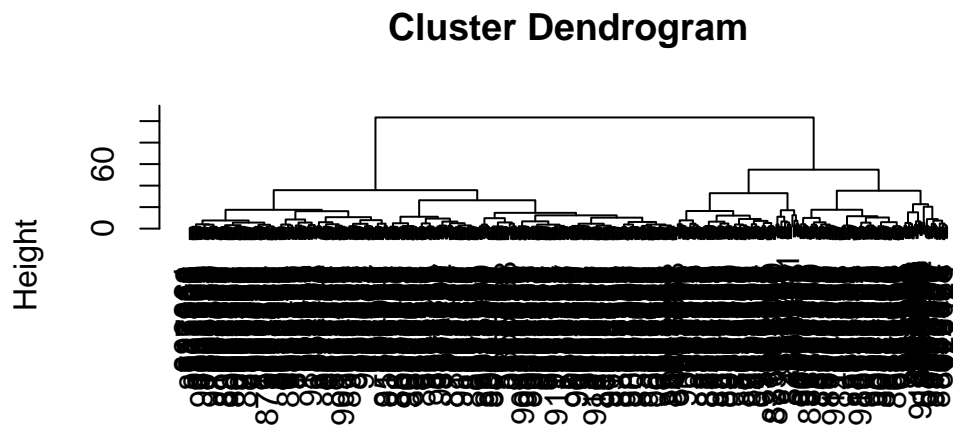
```
#start with the first 3 PCs
head(wisc.pr$x[, 1:3])
```



	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

```
d.pc <- dist(wisc.pr$x[, 1:3])

wisc.pr.hclust <- hclust( d.pc, method = "ward.D2")
plot(wisc.pr.hclust)
```



d.pc  
hclust (\*, "ward.D2")

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

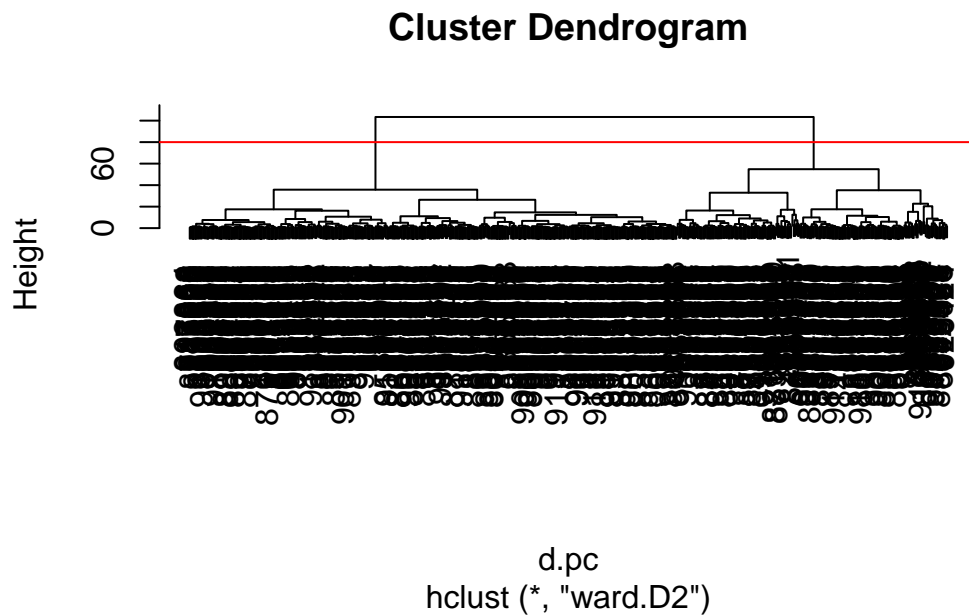
We did not do this in class

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

Professor only told us to use “ward.D2”.

We now have a much clearer separation into two groups. Let's try cutting it.

```
plot(wisc.pr.hclust)
abline(h= 80, col = "red")
```



Now we will cut at the line:

```
grps <- cutree(wisc.pr.hclust, h = 80)
table(grps)
```

```
grps
  1  2
203 366
```

We can use this `table()` function to make a cross-table as well as just a count table.

```
table(diagnosis)
```

```
diagnosis
  B  M
357 212
```

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

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

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

The new model is much better at separating.

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.

We did not do kmeans, so this code will not work since I did not set `wisc.km$cluster`

This table says that out of the 357 B professionally diagnosed results, our cluster correctly identified 333 of them. Out of the 212 M results, our cluster correctly identified 179 of them.

## Prediction

```
#url <- "new_samples.csv"
url <- "https://tinyurl.com/new-samples-CSV"
new <- read.csv(url)
#using our `wisc.pr` to predict where individuals would lie
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

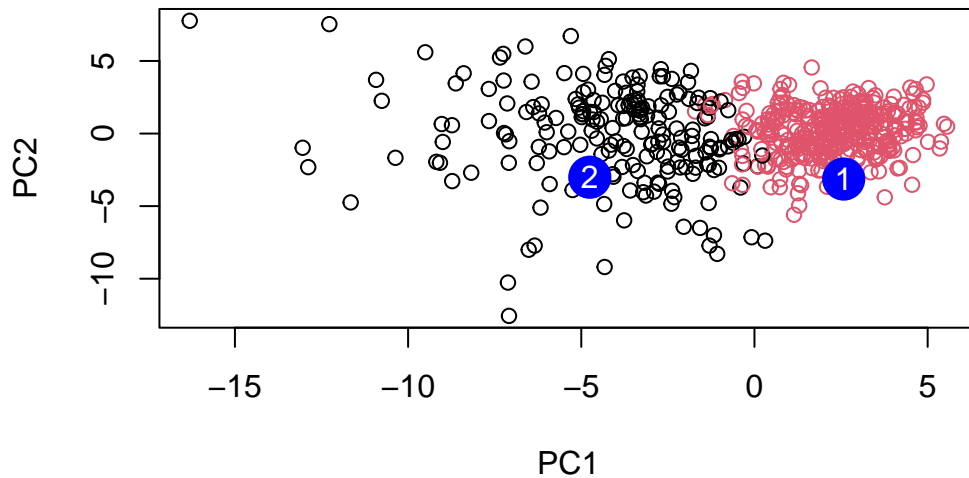
```

Plotting this information

```

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?

This very clearly shows that patient 2 is the one to be more concerned with.