class08

Ebru Robinson

Table of contents

Background

The goal of this mini-project is for us to explore a complete analysis using the unsupervised learning techniques covered in the class. We will extend what we learned by combining PCA as a preprocessing step to clustering using data that consist of measurements of cell nuclei of human breast masses.

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

Save your input data file into your Project directory

```
wisc.df <- read.csv("WisconsinCancer.csv", row.names=1)
```

Make sure we do not code include sample id or diagnosis columns in the data that we analyze below.

```
diagnosis <- as.factor(wisc.df$diagnosis)
wisc.data <- wisc.df[,-1]
dim(wisc.data)
```

```
[1] 569  30
```

```
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
```

Q1. How many observations are in this dataset?

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

```
[1] 569
```

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

```
table(wisc.df$diagnosis)
```

```
  B   M
357 212
```

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

```
#colnames(wisc.data)
length(grep("_mean",colnames(wisc.data)))
```

```
[1] 10
```

Principle Component Analysis

The main function in base R for PCA is called `prcomp()`. An optional argument `scale` should nearly always be switched to `scale=TRUE` for this function.

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

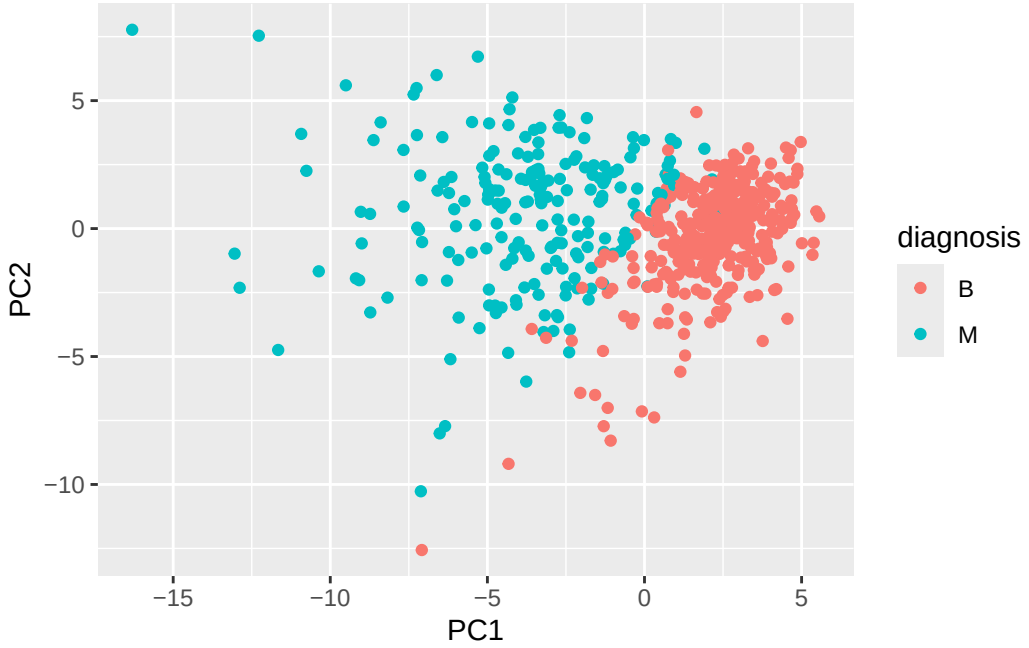
```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880 0.82172
Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025 0.02251
Cumulative Proportion  0.4427 0.6324 0.72636 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
```

Let's make our main result figure - the "PC plot" or "score plot", "ordienation plot"...

```
library(ggplot2)

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



Q4.What proportion of the original variance is captured by the first principal component (PC1)?

From the output: PC1 captures 44.27% (0.4427) of the total variance.

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

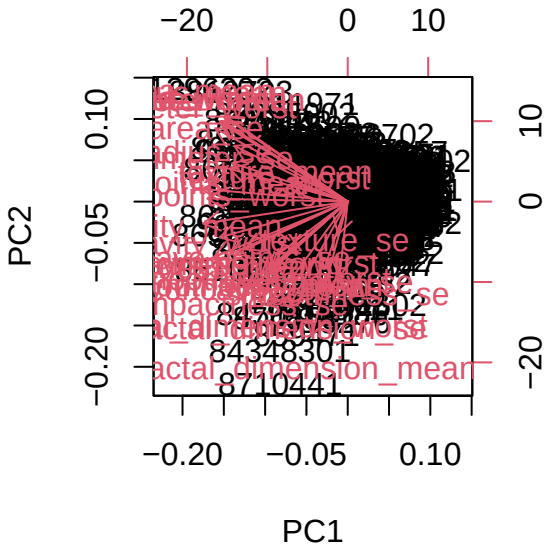
From the cumulative proportion: PC1 + PC2 = 0.6324 (63.24%) PC3 = 0.72636 (72.64%) At least 3 PCs are needed to reach 70% of the variance.

> Q6. How many PCs are required to describe at least 90% of the variance? From the cumulative proportion:

Up to PC6 = 0.88759 (88.76%) PC7 = 0.91010 (91.01%) At least 7 PCs are needed to reach 90% of the variance.

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

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



The plot contains hundreds of sample points (one per observation) and dozens of variable vectors (one per feature).Many arrows (feature loadings) overlap and point in different directions, making it hard to interpret which features correspond to which samples.The text labels for both samples and features overlap heavily, especially since the dataset has 30+ features and 500+ observations.It is very difficult.

Calculate variance of each component

```
pr.var <- wisc.pr$sdev^2
```

Inspect the first few values

```
head(pr.var)
```

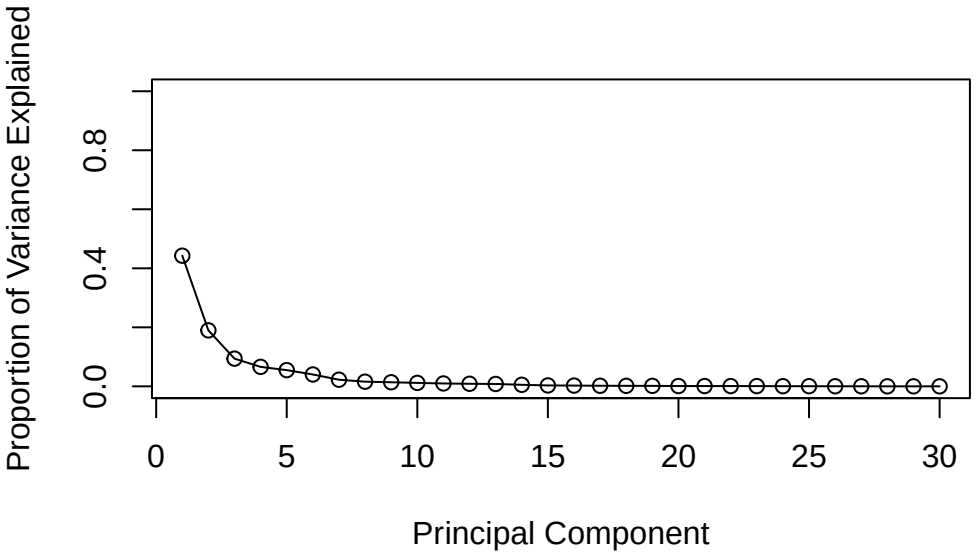
```
[1] 13.281608  5.691355  2.817949  1.980640  1.648731  1.207357
```

Variance explained by each principal component: pve

```
pve <- pr.var / sum(pr.var)
```

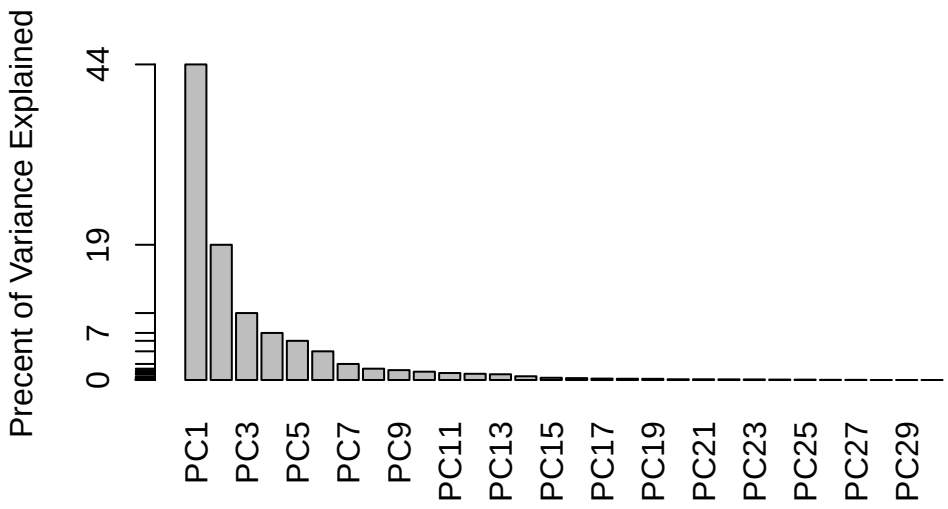
Plot variance explained for each principal component

```
plot(pve,xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0
```



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)), la
axis(2, at=pve, labels=round(pve,2)*100 )
```



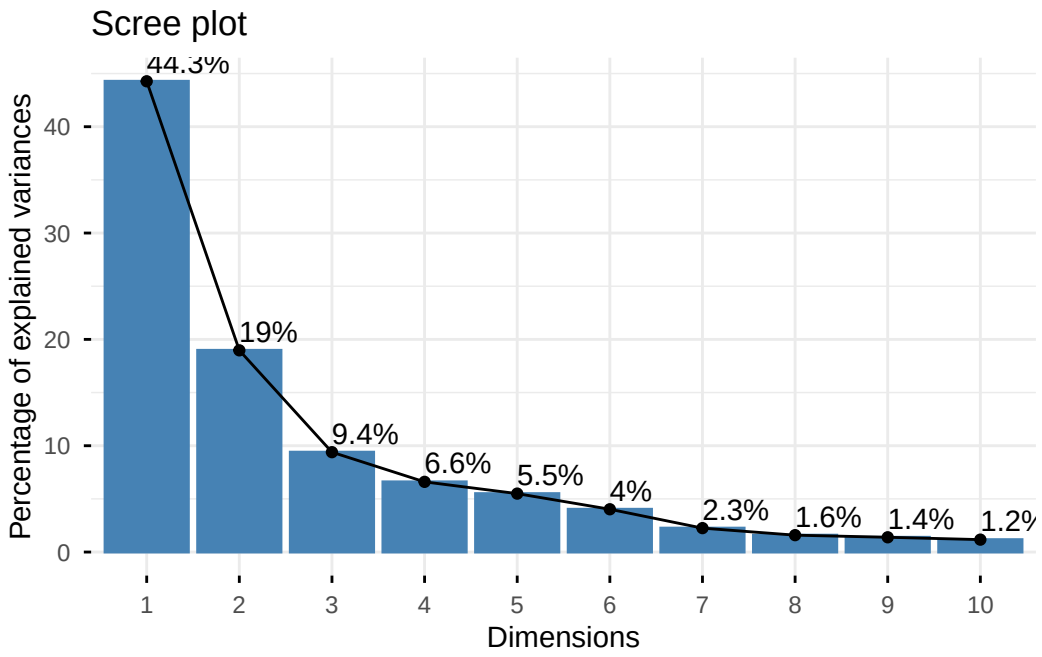
ggplot based graph

```
library(factoextra)
```

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

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

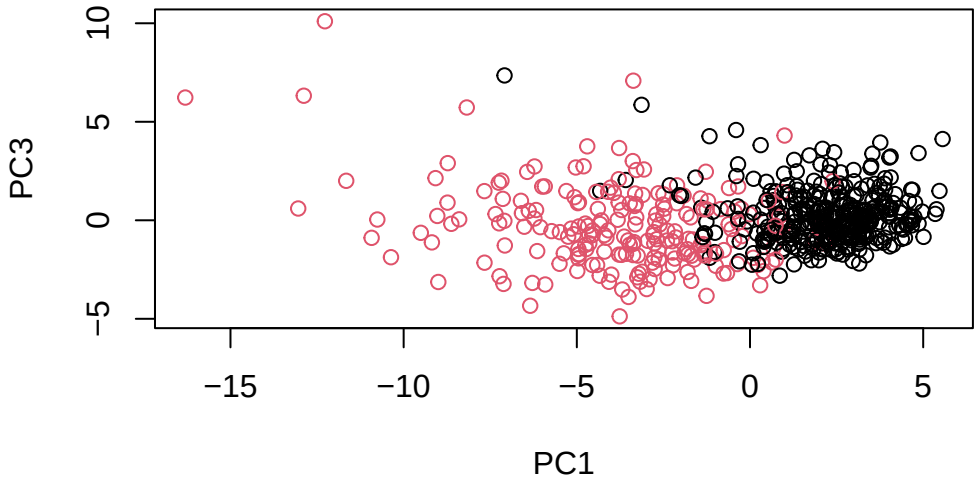
```
Warning in geom_bar(stat = "identity", fill = barfill, color = barcolor, :
Ignoring empty aesthetic: `width`.
```



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

In both plots, malignant and benign samples separate fairly well along PC1. However, PC3 provides less additional separation between the two groups compared to PC2.This indicates that most of the discrimination between malignant and benign samples is captured by PC1 (and partly by PC2), while PC3 mainly represents other sources of variance not directly related to diagnosis

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



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? This tells us how much this original feature contributes to the first PC.

The returned value (typically around 0.26) indicates how strongly the feature concave.points_mean contributes to PC1. A higher absolute value means a stronger contribution (positive or negative) to that principal component.In the Wisconsin Breast Cancer dataset, concave.points_mean has one of the largest loadings on PC1, suggesting it is a key driver of the variance separating malignant and benign samples.

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

```
[1] -0.2608538
```

3. 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")
```

Q10. Using the plot() and abline() functions, what is the height at which the clustering model has 4 clusters?

```
# I will determine the height that gives 4 clusters
cutree(wisc.hclust, k = 4)
```

```
   842302    842517  84300903  84348301  84358402    843786    844359  84458202
        1         1         1         2         1         1         1         1
   844981  84501001    845636  84610002    846226    846381  84667401  84799002
        1         2         3         1         1         3         1         1
   848406  84862001    849014   8510426   8510653   8510824   8511133    851509
        3         1         1         3         3         3         1         1
   852552    852631    852763    852781    852973    853201    853401    853612
        1         1         1         1         1         3         1         1
 85382601    854002    854039    854253    854268    854941    855133    855138
        1         1         1         1         1         3         3         1
   855167    855563    855625    856106  85638502    857010  85713702     85715
        3         1         1         1         1         1         3         1
   857155    857156    857343    857373    857374    857392    857438  85759902
        3         3         3         3         3         1         3         3
   857637    857793    857810    858477    858970    858981    858986    859196
        1         1         3         3         3         3         1         3
 85922302    859283    859464    859465    859471    859487    859575    859711
        1         1         3         3         2         3         1         3
   859717    859983  8610175   8610404   8610629   8610637   8610862   8610908
        1         1         3         3         3         1         2         3
   861103   8611161   8611555   8611792   8612080   8612399  86135501  86135502
        3         1         1         1         3         1         3         1
   861597    861598    861648    861799    861853    862009    862028     86208
        3         1         3         3         3         3         1         1
    86211    862261    862485    862548    862717    862722    862965    862980
        3         3         3         3         3         3         3         3
   862989    863030    863031    863270     86355    864018    864033     86408
        3         1         1         3         1         3         3         3
    86409    864292    864496    864685    864726    864729    864877    865128
        3         3         3         3         3         1         1         3
   865137     86517    865423    865432    865468     86561    866083    866203
```

3	1	2	3	3	3	1	3
866458	866674	866714	8670	86730502	867387	867739	868202
1	1	3	1	1	3	1	3
868223	868682	868826	868871	868999	869104	869218	869224
3	3	1	3	3	3	3	3
869254	869476	869691	86973701	86973702	869931	871001501	871001502
3	3	1	3	3	3	3	3
8710441	87106	8711002	8711003	8711202	8711216	871122	871149
2	3	3	3	1	3	3	3
8711561	8711803	871201	8712064	8712289	8712291	87127	8712729
3	1	1	3	1	3	3	3
8712766	8712853	87139402	87163	87164	871641	871642	872113
1	3	3	3	1	3	3	3
872608	87281702	873357	873586	873592	873593	873701	873843
3	1	3	3	1	1	1	3
873885	874158	874217	874373	874662	874839	874858	875093
1	3	3	3	3	3	2	3
875099	875263	87556202	875878	875938	877159	877486	877500
3	1	1	3	1	3	1	1
877501	877989	878796	87880	87930	879523	879804	879830
3	3	1	1	3	3	3	3
8810158	8810436	881046502	8810528	8810703	881094802	8810955	8810987
1	3	1	3	4	3	1	1
8811523	8811779	8811842	88119002	8812816	8812818	8812844	8812877
3	3	1	1	3	3	3	1
8813129	88143502	88147101	88147102	88147202	881861	881972	88199202
3	3	3	3	3	1	1	3
88203002	88206102	882488	88249602	88299702	883263	883270	88330202
3	1	3	3	1	1	3	1
88350402	883539	883852	88411702	884180	884437	884448	884626
3	3	3	3	1	3	3	3
88466802	884689	884948	88518501	885429	8860702	886226	886452
3	3	1	3	1	3	1	3
88649001	886776	887181	88725602	887549	888264	888570	889403
1	1	1	1	1	3	1	3
889719	88995002	8910251	8910499	8910506	8910720	8910721	8910748
1	1	3	3	3	3	3	3
8910988	8910996	8911163	8911164	8911230	8911670	8911800	8911834
1	3	3	3	3	3	3	3
8912049	8912055	89122	8912280	8912284	8912521	8912909	8913
1	3	1	1	3	3	3	3
8913049	89143601	89143602	8915	891670	891703	891716	891923
3	3	3	3	3	3	3	3

891936	892189	892214	892399	892438	892604	89263202	892657
3	3	3	3	1	3	1	3
89296	893061	89344	89346	893526	893548	893783	89382601
3	3	3	3	3	3	3	3
89382602	893988	894047	894089	894090	894326	894329	894335
3	3	3	3	3	1	3	3
894604	894618	894855	895100	89511501	89511502	89524	895299
3	3	3	1	3	3	3	3
8953902	895633	896839	896864	897132	897137	897374	89742801
1	1	1	1	3	3	3	1
897604	897630	897880	89812	89813	898143	89827	898431
3	1	3	1	3	3	3	1
89864002	898677	898678	89869	898690	899147	899187	899667
3	3	3	3	3	3	3	1
899987	9010018	901011	9010258	9010259	901028	9010333	901034301
1	1	3	3	3	3	3	3
901034302	901041	9010598	9010872	9010877	901088	9011494	9011495
3	3	3	3	3	1	1	3
9011971	9012000	9012315	9012568	9012795	901288	9013005	901303
1	1	1	3	1	1	3	3
901315	9013579	9013594	9013838	901549	901836	90250	90251
3	3	3	1	3	3	3	3
902727	90291	902975	902976	903011	90312	90317302	903483
3	3	3	3	3	1	3	3
903507	903516	903554	903811	90401601	90401602	904302	904357
1	1	3	3	3	3	3	3
90439701	904647	904689	9047	904969	904971	905189	905190
1	3	3	3	3	3	3	3
90524101	905501	905502	905520	905539	905557	905680	905686
1	3	3	3	3	3	3	3
905978	90602302	906024	906290	906539	906564	906616	906878
3	1	3	3	3	1	3	3
907145	907367	907409	90745	90769601	90769602	907914	907915
3	3	3	3	3	3	1	3
908194	908445	908469	908489	908916	909220	909231	909410
1	1	3	1	3	3	3	3
909411	909445	90944601	909777	9110127	9110720	9110732	9110944
3	3	3	3	3	3	1	3
911150	911157302	9111596	9111805	9111843	911201	911202	9112085
3	1	3	1	3	3	3	3
9112366	9112367	9112594	9112712	911296201	911296202	9113156	911320501
3	3	3	3	1	4	3	3
911320502	9113239	9113455	9113514	9113538	911366	9113778	9113816

```
      3         3         3         3         1         1         3         3
 911384   9113846    911391    911408    911654    911673    911685    911916
      3         3         3         3         3         3         3         1
 912193     91227    912519    912558    912600    913063    913102    913505
      3         3         3         3         3         3         3         1
 913512    913535  91376701  91376702    914062    914101    914102    914333
      3         3         3         3         1         3         3         3
 914366    914580    914769     91485    914862     91504     91505    915143
      1         3         1         1         3         1         3         1
 915186    915276  91544001  91544002    915452    915460     91550    915664
      3         3         3         3         3         1         3         3
 915691    915940  91594602    916221    916799    916838    917062    917080
      1         3         3         3         1         1         3         3
 917092  91762702     91789    917896    917897     91805  91813701  91813702
      3         1         3         3         3         3         1         3
 918192    918465     91858  91903901  91903902  91930402    919537    919555
      3         3         3         3         3         1         3         1
91979701    919812    921092    921362    921385    921386    921644    922296
      3         1         3         3         3         1         3         3
 922297    922576    922577    922840    923169    923465    923748    923780
      3         3         3         3         3         3         3         3
 924084    924342    924632    924934    924964    925236    925277    925291
      3         3         3         3         3         3         3         3
 925292    925311    925622    926125    926424    926682    926954    927241
      3         3         1         1         1         1         3         1
  92751
      3
```

```
# Let's check cluster heights
wisc.hclust$height
```

```
  [1]  1.005230  1.026711  1.096132  1.100008  1.119867  1.146701  1.147702
  [8]  1.209527  1.215177  1.216937  1.228931  1.268268  1.271313  1.298518
 [15]  1.320791  1.324071  1.331997  1.341092  1.350120  1.356226  1.358986
 [22]  1.367199  1.367246  1.367406  1.376013  1.382132  1.390225  1.392998
 [29]  1.406426  1.419157  1.429975  1.436004  1.443621  1.450628  1.477401
 [36]  1.485766  1.488450  1.530763  1.533372  1.534547  1.536138  1.547617
 [43]  1.565400  1.569854  1.574260  1.592749  1.595827  1.598243  1.598859
 [50]  1.599830  1.606257  1.609077  1.609248  1.629422  1.651875  1.652908
 [57]  1.654239  1.659267  1.665912  1.666256  1.668353  1.670058  1.687105
 [64]  1.695445  1.701228  1.702677  1.711957  1.723258  1.733826  1.737552
 [71]  1.739122  1.742630  1.743912  1.747820  1.749839  1.751217  1.766804
```

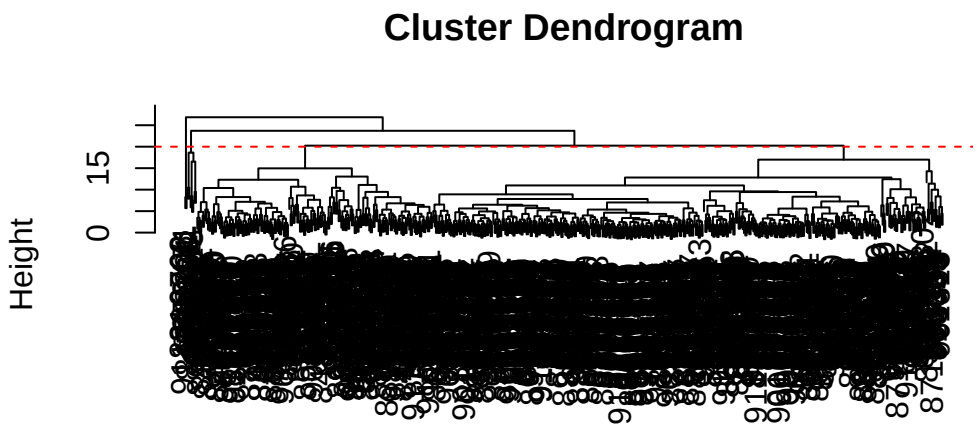
```
 [78]  1.767422  1.773438  1.773856  1.774662  1.779741  1.781768  1.787211
 [85]  1.788921  1.796730  1.799357  1.807758  1.814891  1.816495  1.817372
 [92]  1.821606  1.823033  1.823518  1.825508  1.826941  1.827691  1.844635
 [99]  1.849461  1.853010  1.858688  1.864548  1.869756  1.871654  1.871735
[106]  1.880079  1.887728  1.888468  1.890715  1.891311  1.891798  1.893887
[113]  1.904939  1.909234  1.909783  1.911427  1.924116  1.926464  1.929332
[120]  1.930438  1.934275  1.939516  1.944477  1.946280  1.974602  1.975521
[127]  1.994537  1.999286  2.011384  2.016180  2.022058  2.029413  2.032216
[134]  2.034585  2.034885  2.040434  2.044159  2.066417  2.067314  2.076174
[141]  2.083890  2.084520  2.084771  2.084820  2.086478  2.089702  2.095683
[148]  2.100539  2.109334  2.110946  2.113200  2.117617  2.118580  2.120169
[155]  2.124586  2.125511  2.127216  2.131799  2.136624  2.142092  2.149496
[162]  2.160120  2.161211  2.178885  2.179267  2.183077  2.197286  2.201076
[169]  2.218892  2.219137  2.221416  2.239867  2.260189  2.269696  2.275244
[176]  2.279286  2.291714  2.301881  2.301984  2.306024  2.317967  2.319906
[183]  2.331793  2.344151  2.346238  2.352106  2.359622  2.361766  2.370875
[190]  2.375271  2.375881  2.388371  2.391336  2.391725  2.415227  2.417728
[197]  2.435423  2.437448  2.440903  2.445407  2.448525  2.448843  2.449386
[204]  2.453962  2.461195  2.464110  2.464552  2.464566  2.466033  2.467459
[211]  2.473961  2.473991  2.475809  2.482193  2.488075  2.491304  2.491913
[218]  2.493406  2.498459  2.513145  2.515798  2.518534  2.522818  2.530981
[225]  2.540393  2.545612  2.557633  2.558287  2.559306  2.573835  2.576955
[232]  2.578323  2.604644  2.613944  2.625244  2.635215  2.650621  2.673002
[239]  2.673349  2.673644  2.675693  2.686567  2.689533  2.692775  2.694214
[246]  2.700272  2.700679  2.702083  2.705661  2.709335  2.715197  2.719637
[253]  2.722153  2.728442  2.738493  2.743977  2.744633  2.749743  2.750247
[260]  2.755635  2.756343  2.763187  2.765535  2.767643  2.778366  2.781427
[267]  2.781566  2.787540  2.816533  2.819703  2.822625  2.837623  2.838141
[274]  2.839211  2.844678  2.847168  2.847694  2.853021  2.854981  2.856284
[281]  2.867143  2.877951  2.887432  2.890137  2.891428  2.893594  2.900583
[288]  2.907303  2.910030  2.910749  2.933666  2.951909  2.953352  2.956051
[295]  2.956285  2.958726  2.960002  2.964908  2.968782  2.976127  2.988419
[302]  2.992190  2.997087  3.008613  3.020787  3.022405  3.033286  3.036721
[309]  3.040079  3.052697  3.056811  3.057908  3.072395  3.078508  3.081995
[316]  3.089342  3.090875  3.099869  3.106389  3.107299  3.109140  3.112594
[323]  3.144882  3.157014  3.159963  3.161172  3.165804  3.168631  3.188078
[330]  3.193575  3.197490  3.202124  3.213407  3.225338  3.242815  3.248166
[337]  3.251482  3.285321  3.290650  3.291670  3.307359  3.313863  3.318986
[344]  3.344213  3.352336  3.352402  3.354084  3.385203  3.387703  3.394852
[351]  3.399973  3.441553  3.458219  3.460710  3.464911  3.466990  3.468893
[358]  3.479058  3.488304  3.496882  3.562544  3.566301  3.566351  3.579221
[365]  3.596959  3.613420  3.624016  3.628544  3.646693  3.664820  3.673423
[372]  3.674342  3.686850  3.691358  3.708899  3.737276  3.744325  3.759048
```

```
[379]  3.759811  3.777131  3.790839  3.797925  3.813528  3.822259  3.825173
[386]  3.826515  3.829305  3.829965  3.830055  3.831017  3.865220  3.896544
[393]  3.908290  3.913497  3.913563  3.926535  3.929015  3.930831  3.945084
[400]  3.948747  3.996948  4.014483  4.044007  4.063300  4.067046  4.068461
[407]  4.089437  4.100528  4.115785  4.129474  4.133223  4.145537  4.169407
[414]  4.170211  4.189956  4.195116  4.197808  4.216784  4.236865  4.242305
[421]  4.242779  4.262300  4.273223  4.306003  4.315373  4.329950  4.359855
[428]  4.360679  4.361214  4.367910  4.376070  4.383003  4.401181  4.409695
[435]  4.438712  4.448294  4.454908  4.464907  4.498061  4.593197  4.600040
[442]  4.602226  4.629016  4.641532  4.670503  4.689338  4.717133  4.745250
[449]  4.769450  4.776486  4.776545  4.787026  4.814873  4.826310  4.834212
[456]  4.840763  4.841758  4.857763  4.869967  4.873945  4.906745  4.924646
[463]  4.932525  4.940799  4.975046  4.999032  5.007626  5.009415  5.066402
[470]  5.085591  5.086201  5.199286  5.218275  5.234202  5.252211  5.276128
[477]  5.312915  5.356423  5.360096  5.364566  5.390627  5.392103  5.448860
[484]  5.526307  5.598938  5.626338  5.626645  5.659072  5.676613  5.702048
[491]  5.771244  5.805293  5.850696  5.869093  5.926479  5.991276  6.005960
[498]  6.019395  6.127199  6.131595  6.242541  6.276511  6.277634  6.330833
[505]  6.355761  6.454382  6.489883  6.494109  6.522278  6.529232  6.580286
[512]  6.617581  6.703568  6.783072  6.817585  6.876998  6.925925  6.962872
[519]  6.980563  7.023077  7.029478  7.097307  7.168460  7.258934  7.406942
[526]  7.534671  7.566726  7.646292  7.827652  8.073957  8.077915  8.140482
[533]  8.209893  8.213918  8.261273  8.265040  8.277593  8.360934  8.440802
[540]  8.472046  8.661494  8.910477  9.446100  9.483452  9.512115  9.601947
[547]  9.707425  9.995552 10.110130 10.386553 10.440182 10.466865 10.686962
[554] 10.982812 11.100405 11.409358 12.306988 12.436039 12.828040 13.042946
[561] 14.290358 14.959469 16.543516 16.986102 18.636581 20.243301 23.705661
[568] 26.858388
```

I picked the height value (for example, around h 20) that corresponds to 4 clusters when I draw the red line.

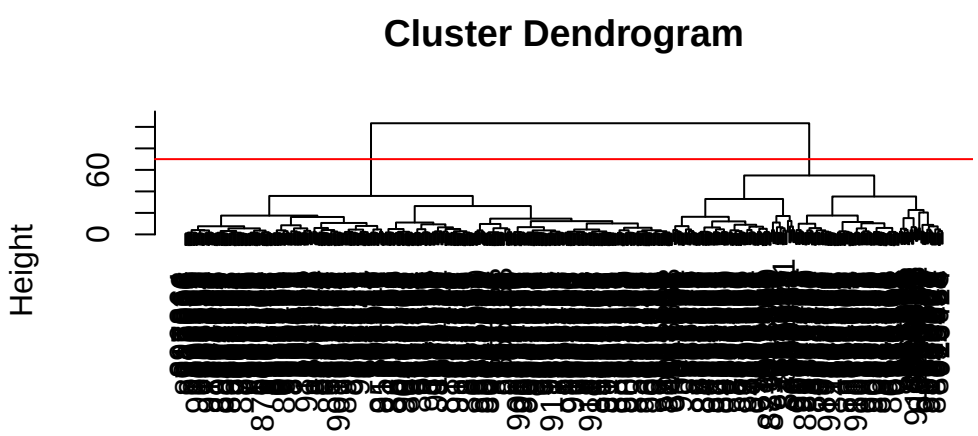
```
plot(wisc.hclust)

abline(h = 20, col = "red", lty = 2)
```



Combining clustering

```
d <- dist(wisc.pr$x[,1:3])
wisc.prhclust <- hclust(d,method="ward.D2")
plot(wisc.prhclust)
abline(h=70, col="red")
```



get my cluster membership vector

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

```
grps
  1   2
203 366
```

```
table(diagnosis)
```

```
diagnosis
  B   M
357 212
```

Make a wee "cross-table"

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

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

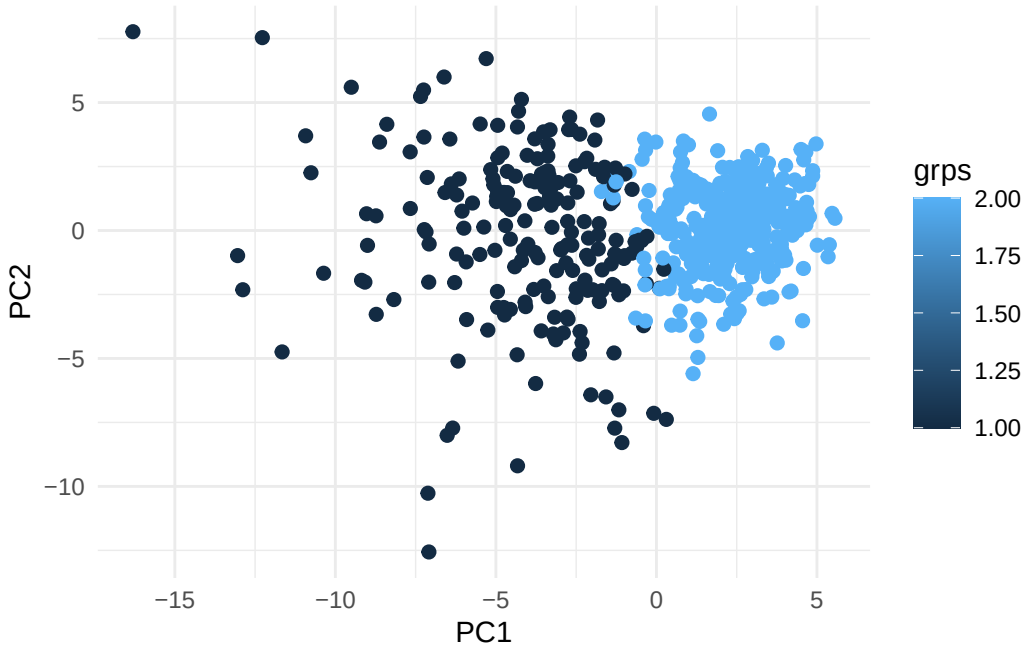
TP:179 FP:24 Sensitivity: TP/(TP+FN)

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

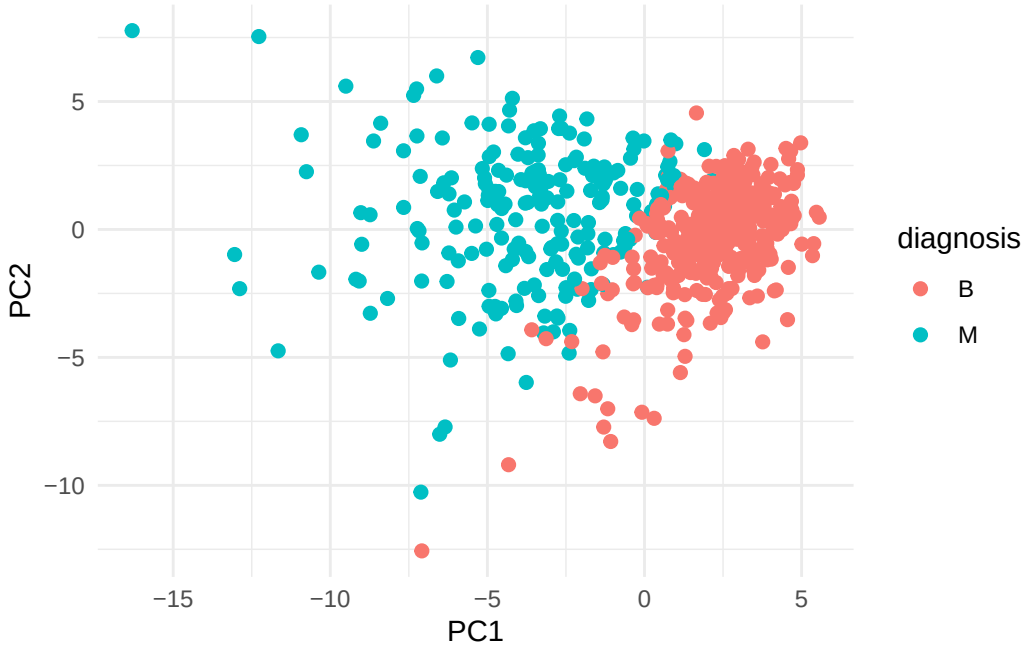
Q12. Which method gives your favorite results for the same data.dist dataset? Explain your reasoning.

My favorite is the ward.d2 method. I like the ward.d2 message because its easier to interpret.

```
ggplot(wisc.pr$x, aes(x = PC1, y = PC2, color = grps)) +
geom_point(size = 2) +
labs(x = "PC1", y = "PC2") +
theme_minimal()
```




```
ggplot(wisc.pr$x, aes(x = PC1, y = PC2, color = diagnosis)) +
geom_point(size = 2) +
labs(x = "PC1", y = "PC2") +
theme_minimal()
```



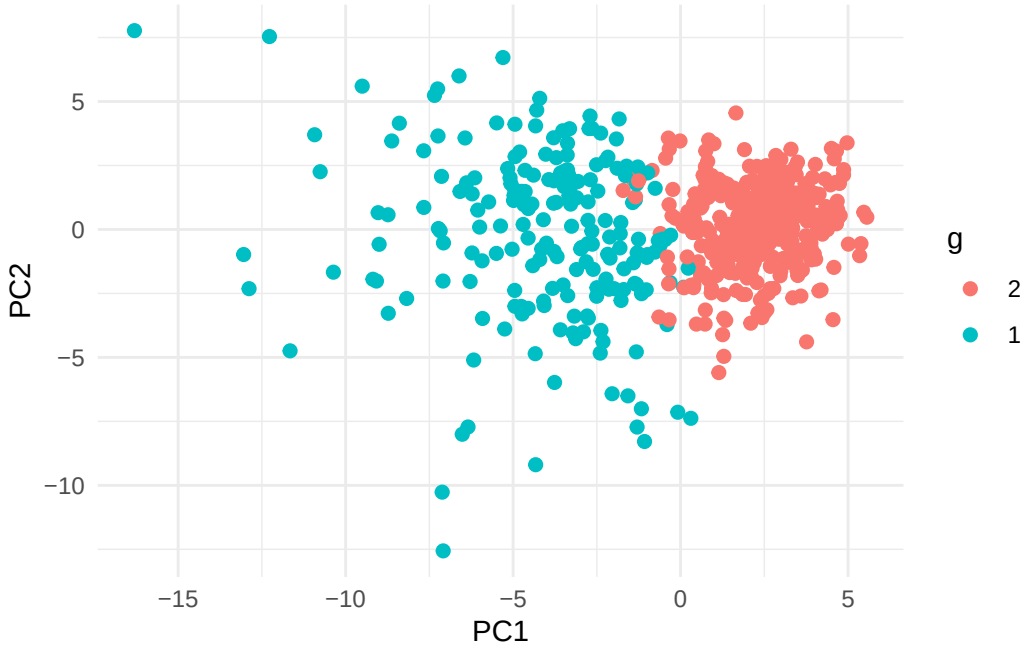

```
g <- as.factor(grps)
levels(g)
```

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

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

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

```
ggplot(wisc.pr$x, aes(x = PC1, y = PC2, color = g)) +
geom_point(size = 2) +
labs(x = "PC1", y = "PC2") +
theme_minimal()
```




```
wisc.pr.hclust <- hclust(dist(wisc.pr$x[, 1:7]), method = "ward.D2")
wisc.pr.hclust.clusters <- cutree(wisc.pr.hclust, k=2)
table(wisc.pr.hclust.clusters, diagnosis)
```

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

Q13. How well does the newly created model with four clusters separate out the two diagnoses? It does a really good job at separating the two diagnoses. I can easily tell that group 1 associates with mostly malignant whereas group 2 mostly associates with benign tumors.

```
table(wisc.hclust.clusters, wisc.df$diagnosis)
```

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

Q14. How well do the 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.

```
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 contains mostly malignant (M) cases → good separation.Cluster 3 contains mostly benign (B) cases → also good separation.Clusters 2 and 4 are small mixed clusters, possibly outliers or borderline samples.

The hierarchical clustering model partially separates malignant and benign samples, but not perfectly.Some malignant and benign cases are mixed in smaller clusters, indicating that unsupervised clustering before PCA does not perfectly capture the diagnostic separation. the main malignant/benign distinction is somewhat reflected in the clustering structure.

Prediction

```
#url <- "new_samples.csv"
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
```

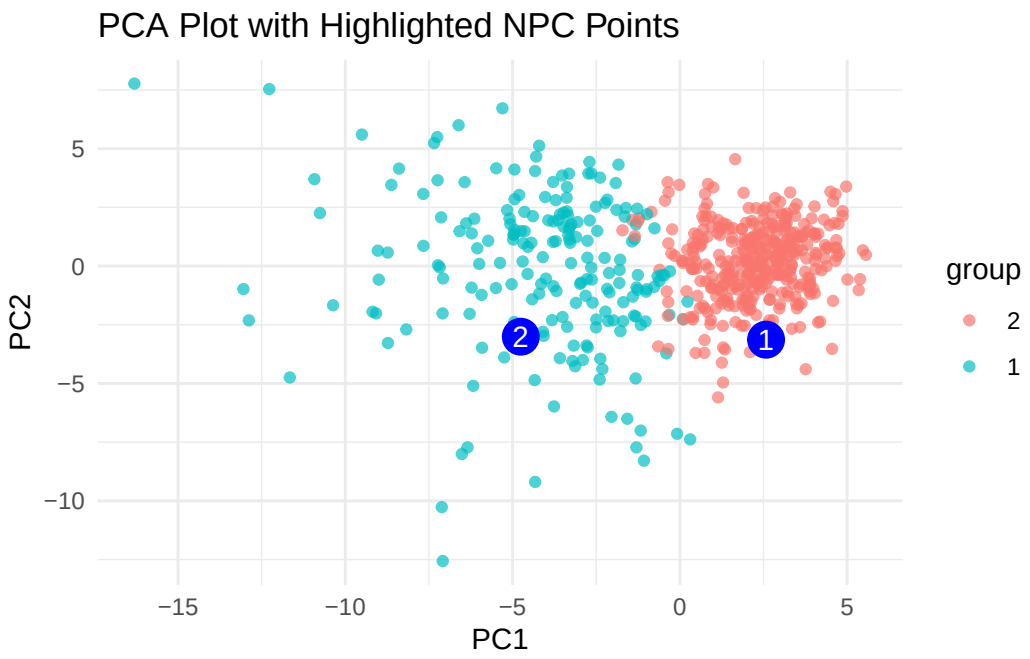
```
           PC21       PC22       PC23       PC24        PC25          PC26
[1,]  0.1228233 0.09358453 0.08347651  0.1223396  0.02124121  0.078884581
[2,] -0.1224776 0.01732146 0.06316631 -0.2338618 -0.20755948 -0.009833238
             PC27        PC28         PC29         PC30
[1,]  0.220199544 -0.02946023 -0.015620933  0.005269029
[2,] -0.001134152  0.09638361  0.002795349 -0.019015820
```

```r
library(ggplot2)
# Create a data frame for the PCA points
pca_df <- data.frame(PC1 = wisc.pr$x[, 1],
PC2 = wisc.pr$x[, 2],
group = as.factor(g))
# Create a data frame for npc points
npc_df <- data.frame(PC1 = npc[, 1],
PC2 = npc[, 2],
label = as.factor(c(1, 2)))

# Plot with ggplot2
ggplot(pca_df, aes(x = PC1, y = PC2, color = group)) +
geom_point(alpha = 0.7) +
geom_point(data = npc_df, aes(x = PC1, y = PC2),

color = "blue", size = 6) +
geom_text(data = npc_df, aes(label = label),
color = "white", size = 4) +

labs(x = "PC1", y = "PC2",
title = "PCA Plot with Highlighted NPC Points") +
theme_minimal()
```



Q16. Which of these new patients should we prioritize for follow up based on your results?

Patient 2 needs to be prioritized for a check up since their tumor most likely will be malignant (since they fall into group 1). For group 1: TP=188 and FP=28

```
sessionInfo()
```

```
R version 4.5.1 (2025-06-13)
Platform: aarch64-apple-darwin20
Running under: macOS Sequoia 15.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/Los_Angeles
tzcode source: internal

attached base packages:
```

```
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] factoextra_1.0.7 ggplot2_4.0.0

loaded via a namespace (and not attached):
 [1] gtable_0.3.6       jsonlite_2.0.0     dplyr_1.1.4        compiler_4.5.1
 [5] ggsignif_0.6.4     tidyselect_1.2.1   Rcpp_1.1.0         tidyr_1.3.1
 [9] scales_1.4.0       yaml_2.3.10        fastmap_1.2.0      R6_2.6.1
[13] ggpubr_0.6.2       labeling_0.4.3     generics_0.1.4     Formula_1.2-5
[17] knitr_1.50         backports_1.5.0    ggrepel_0.9.6      tibble_3.3.0
[21] car_3.1-3          pillar_1.11.1      RColorBrewer_1.1-3 rlang_1.1.6
[25] broom_1.0.10       xfun_0.53          S7_0.2.0           cli_3.6.5
[29] withr_3.0.2        magrittr_2.0.4     digest_0.6.37      grid_4.5.1
[33] rstudioapi_0.17.1  lifecycle_1.0.4    vctrs_0.6.5        rstatix_0.7.3
[37] evaluate_1.0.5     glue_1.8.0         farver_2.1.2       abind_1.4-8
[41] carData_3.0-5      rmarkdown_2.30     purrr_1.1.0        tools_4.5.1
[45] pkgconfig_2.0.3    htmltools_0.5.8.1
```