Class 8: Unsupervised Learning Mini-Project

Yiyu

Table of contents

Today we will practice applying our PCA and clustering methods from the last class on some breast cancer FNA data.

Let's get the data into R...

```
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
```

Q1. How many samples/patients do we have?

There are 569 samples in this dataset.

Q2. How many cancer/non-cancer diagnoses samples are there?

We can use the `table()` function. It is a super useful utility for counting up the number of observations of each type.

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

```
  B   M
357 212
```

Q3. How many columns/dimensions are there in this data set?

```
ncol(wisc.df)
```

```
[1] 31
```

```
dim(wisc.df)
```

```
[1] 569  31
```

Q4. How many columns are suffixed with "_mean"?

```
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"
```

```
x <- grep("_mean", colnames(wisc.df))
length(x)
```

```
[1] 10
```

Let's tidy our dataset to remove diagnosis.

We will not be using the diagnosis column for our unsupervised analysis as it tells us the answer. So let's save it as a vector first, then remove it from the data frame so the data can undergo clusering, PCA, etc…

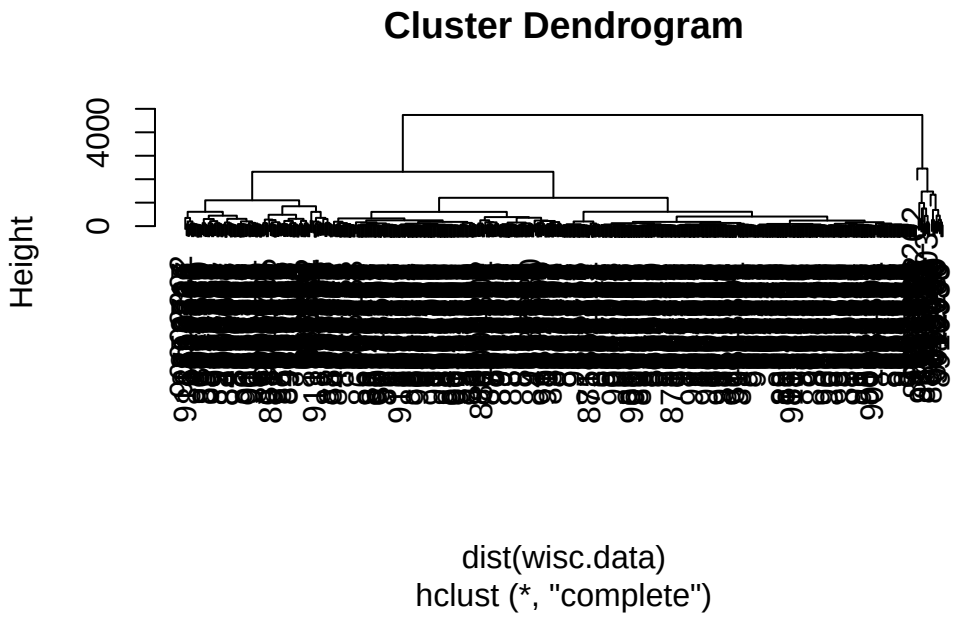
```
diagnosis <- wisc.df$diagnosis
```

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

Cluster the dataset

Let's try a `hclust()`.

```
hc.raw <- hclust(dist(wisc.data))
plot(hc.raw)
```



To get some clusters out of this, I can "cut" the tree at a given height:

```
grps <- cutree(hc.raw, h=4000)
table(grps)
```

```
grps
  1   2
549  20
```

To see the correspondance of our cluster `grps` with the expert `diagnosis`, I can use `table()` again.

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

```
    diagnosis
grps   B   M
   1 357 192
   2   0  20
```

That is not that useful of a clustering result...

Principal Component Analysis (PCA)

Scaling data before analysis is often critical.

Side-note: The default behavior for `prcomp()` is `scale=False`

The is a dataset in R called `mtcars()` which has loads of numbers about old cars.

```
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
colMeans(mtcars)
```

```
      mpg       cyl      disp        hp      drat        wt      qsec
20.090625  6.187500 230.721875 146.687500  3.596563  3.217250 17.848750
       vs        am      gear      carb
 0.437500  0.406250  3.687500  2.812500
```

```r
apply(mtcars, 2, sd)
```

```
       mpg        cyl       disp         hp       drat         wt
 6.0269481  1.7859216 123.9386938 68.5628685  0.5346787  0.9784574
      qsec         vs         am       gear       carb
 1.7869432  0.5040161  0.4989909  0.7378041  1.6152000
```

```r
pc.noscale <- prcomp(mtcars, scale = FALSE)
pc.scale <- prcomp(mtcars, scale = TRUE)
```

Let's look at the loadings first:

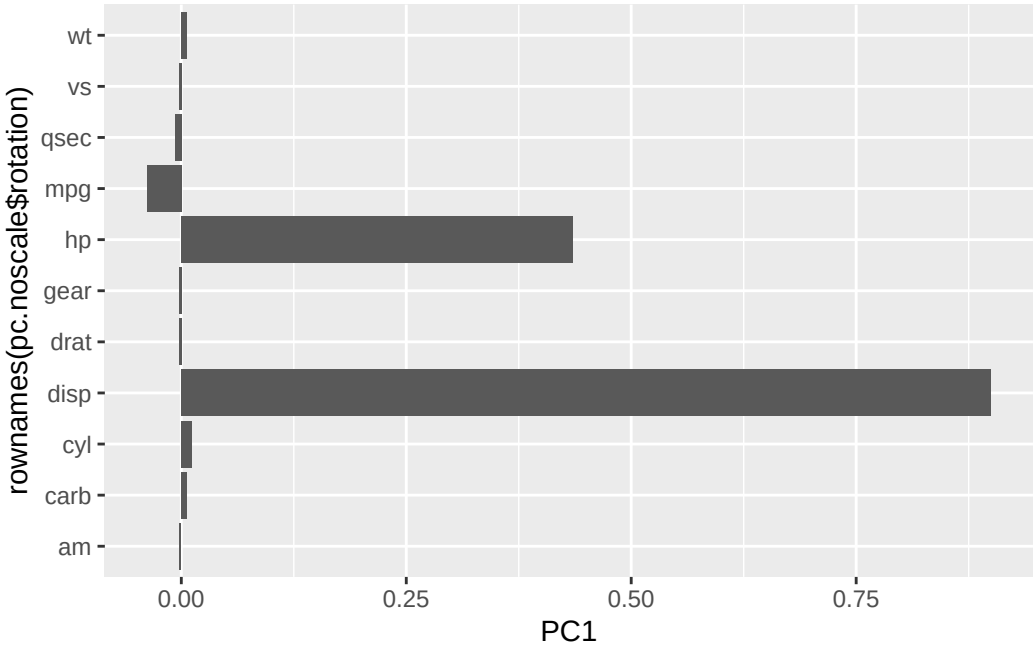
```r
pc.noscale$rotation
```

```
              PC1          PC2          PC3          PC4         PC5
mpg  -0.038118199  0.009184847  0.982070847  0.047634784 -0.08832843
cyl   0.012035150 -0.003372487 -0.063483942 -0.227991962  0.23872590
disp  0.899568146  0.435372320  0.031442656 -0.005086826 -0.01073597
hp    0.434784387 -0.899307303  0.025093049  0.035715638  0.01655194
drat -0.002660077 -0.003900205  0.039724928 -0.057129357 -0.13332765
wt    0.006239405  0.004861023 -0.084910258  0.127962867 -0.24354296
qsec -0.006671270  0.025011743 -0.071670457  0.886472188 -0.21416101
vs   -0.002729474  0.002198425  0.004203328  0.177123945 -0.01688851
am   -0.001962644 -0.005793760  0.054806391 -0.135658793 -0.06270200
gear -0.002604768 -0.011272462  0.048524372 -0.129913811 -0.27616440
carb  0.005766010 -0.027779208 -0.102897231 -0.268931427 -0.85520810
              PC6          PC7           PC8          PC9          PC10
mpg  -0.143790084 -0.039239174 -2.271040e-02 -0.002790139  0.030630361
cyl  -0.793818050  0.425011021  1.890403e-01  0.042677206  0.131718534
disp  0.007424138  0.000582398  5.841464e-04  0.003532713 -0.005399132
hp    0.001653685 -0.002212538 -4.748087e-06 -0.003734085  0.001862554
drat  0.227229260  0.034847411  9.385817e-01 -0.014131110  0.184102094
```

```
wt   -0.127142296 -0.186558915 -1.561907e-01 -0.390600261  0.829886844
qsec -0.189564973  0.254844548  1.028515e-01 -0.095914479 -0.204240658
vs    0.102619063 -0.080788938  2.132903e-03  0.684043835  0.303060724
am    0.205217266  0.200858874  2.273255e-02 -0.572372433 -0.162808201
gear  0.334971103  0.801625551 -2.174878e-01  0.156118559  0.203540645
carb -0.283788381 -0.165474186 -3.972219e-03  0.127583043 -0.239954748
              PC11
mpg   0.0158569365
cyl  -0.1454453628
disp -0.0009420262
hp    0.0021526102
drat  0.0973818815
wt    0.0198581635
qsec -0.0110677880
vs   -0.6256900918
am   -0.7331658036
gear  0.1909325849
carb -0.0557957968
```

Plot the noscale version

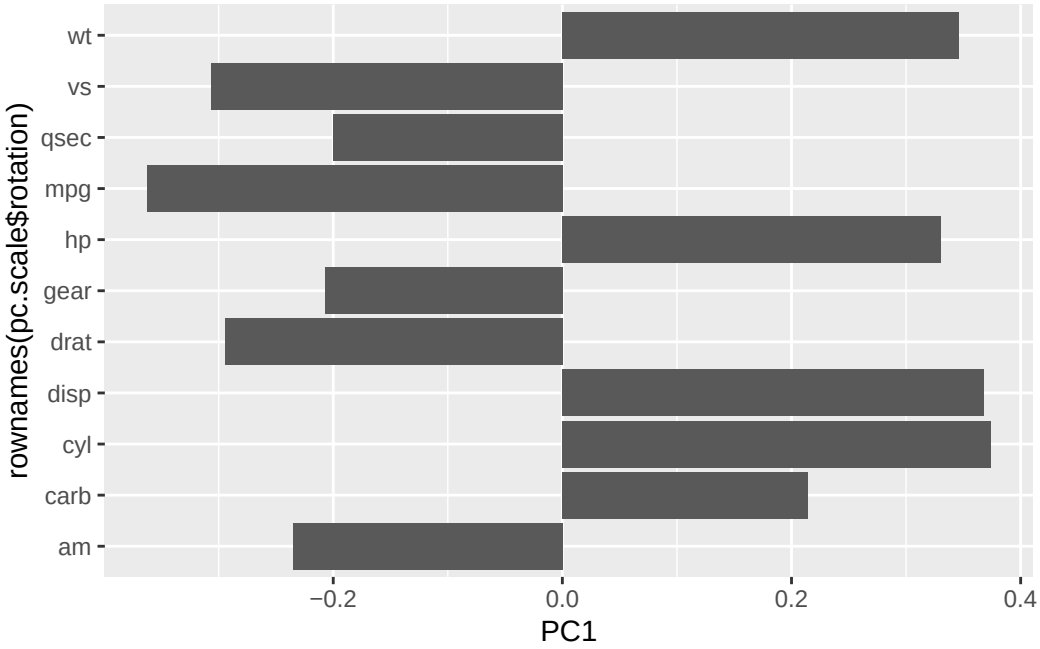
```
library(ggplot2)
ggplot(pc.noscale$rotation) +
  aes(PC1, rownames(pc.noscale$rotation)) +
  geom_col()
```



We see the representation is heavily dominated by hp and disp because it naturally has larger numbers.

Now plot the scaled version

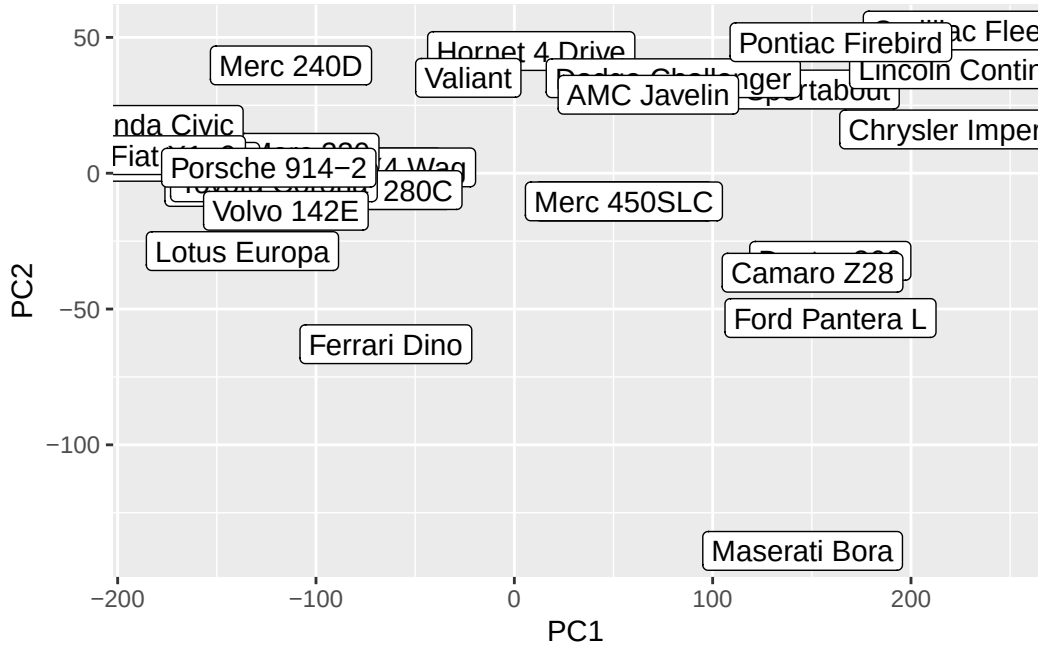
```
library(ggplot2)
ggplot(pc.scale$rotation) +
  aes(PC1, rownames(pc.scale$rotation)) +
  geom_col()
```



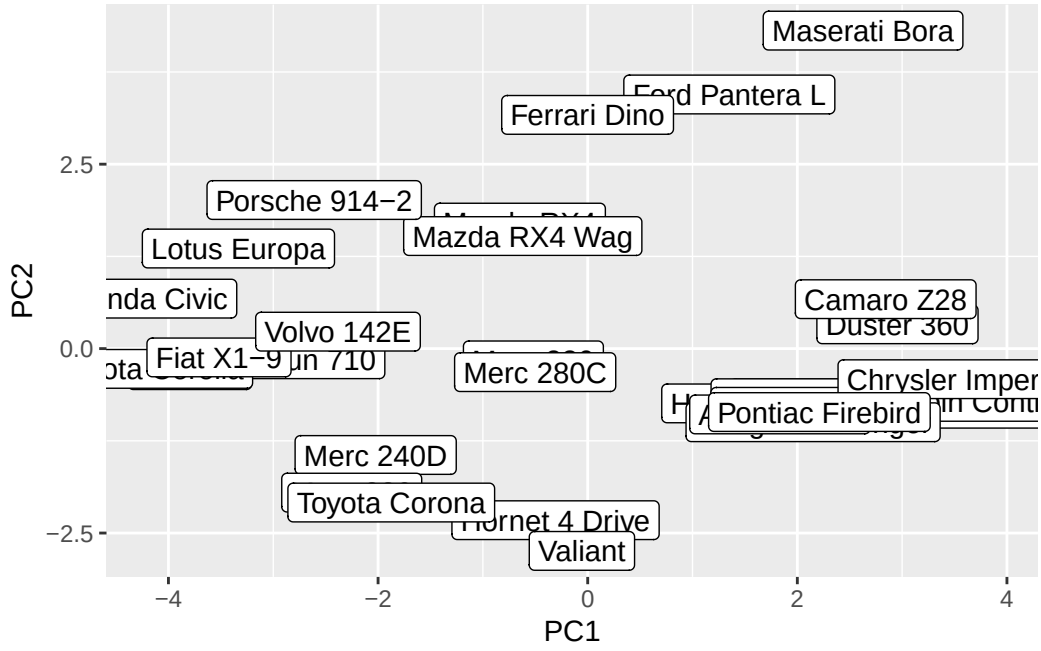
We see the representation is much more evenly distributed.

The main PC result figure is often alled a “score plot” or “PC plot” or “PC1 vs PC2 plot”

```
ggplot(pc.noscale$x) +
  aes(PC1, PC2, label = rownames(pc.noscale$x)) +
  geom_point() +
  geom_label()
```




```
ggplot(pc.scale$x) +
  aes(PC1, PC2, label = rownames(pc.scale$x)) +
  geom_point() +
  geom_label()
```




```
x <- scale(mtcars)
round(colMeans(x))
```

```
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
   0    0    0    0    0    0    0    0    0    0    0
```

```
round(apply(x, 2, sd))
```

```
 mpg  cyl disp   hp drat   wt qsec   vs   am gear carb
   1    1    1    1    1    1    1    1    1    1    1
```

> **Key-point**: Generally we want to "scale" our data before analysis to avoid being mis-lead due to your data having different measurement units.

Breast Cancer PCA

We will scale our data.

```
pca <-  prcomp(wisc.data, scale = T)
```

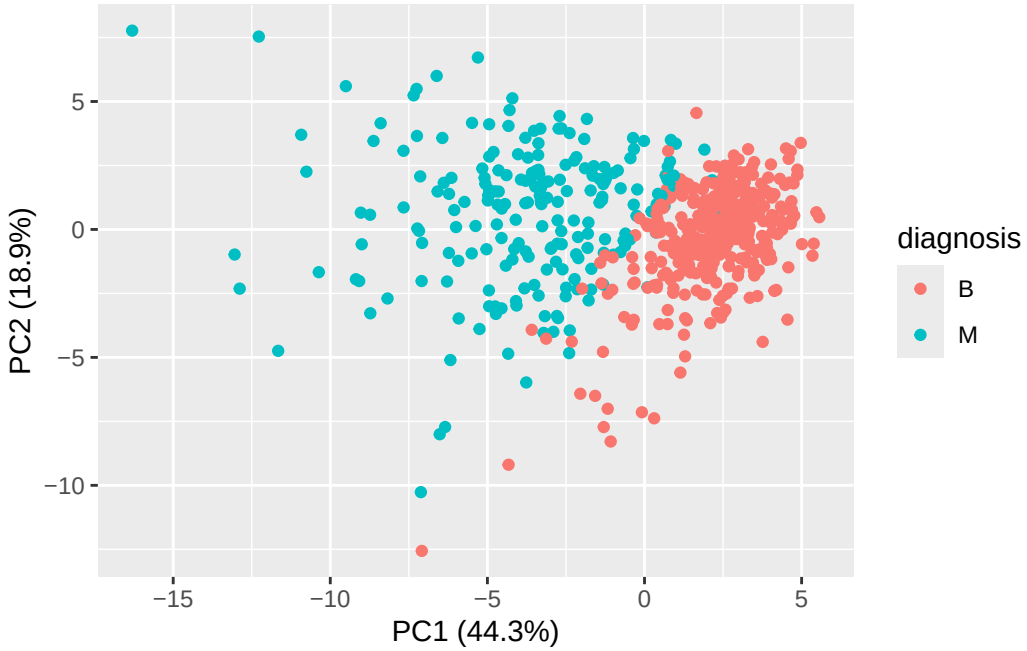
See how well we are doing:

```
summary(pca)
```

```
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
```

Our PC plot

```
ggplot(pca$x) +
  aes(PC1, PC2, col = diagnosis) +
  geom_point() +
  xlab("PC1 (44.3%)") +
  ylab("PC2 (18.9%)")
```



Q4. How many PCs capture 80% of the original variance in the dataset?

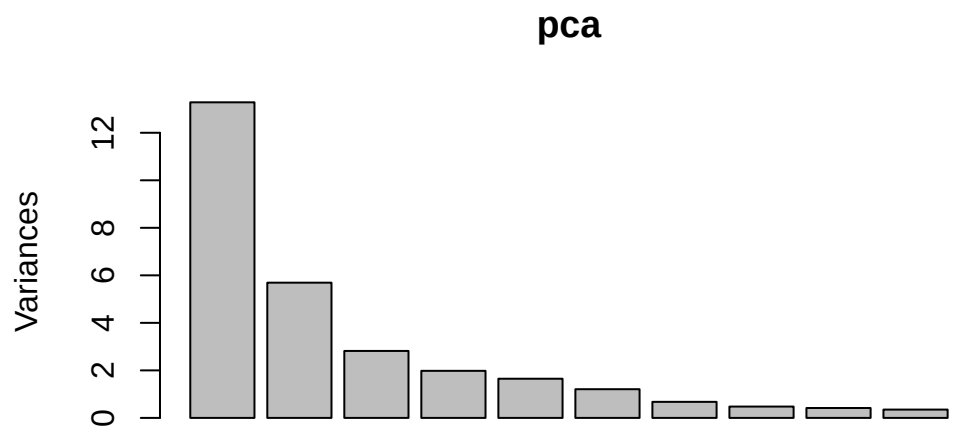
```
summary(pca)
```

```
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
```

Need 5 PCs to capture at least 80%.

```
plot(pca)
```



Q5. Use ggplot to plot a "screen-plot: of the variance per PC.

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

We can extract the sdev ad figure out the variance…

```r
v <- pca$sdev^2
round(v)
```

```
 [1] 13  6  3  2  2  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
[26]  0  0  0  0  0
```

```r
sum(v)
```

```
[1] 30
```

The portion of variance captured in each PC. See how it is the same as the proportion variance row of the summary pca table.

```r
round(v/sum(v), 4)
```

```
 [1] 0.4427 0.1897 0.0939 0.0660 0.0550 0.0402 0.0225 0.0159 0.0139 0.0117
[11] 0.0098 0.0087 0.0080 0.0052 0.0031 0.0027 0.0020 0.0018 0.0016 0.0010
[21] 0.0010 0.0009 0.0008 0.0006 0.0005 0.0003 0.0002 0.0001 0.0000 0.0000
```

See how it is the same as the cumulative sum row of the summary pca table.

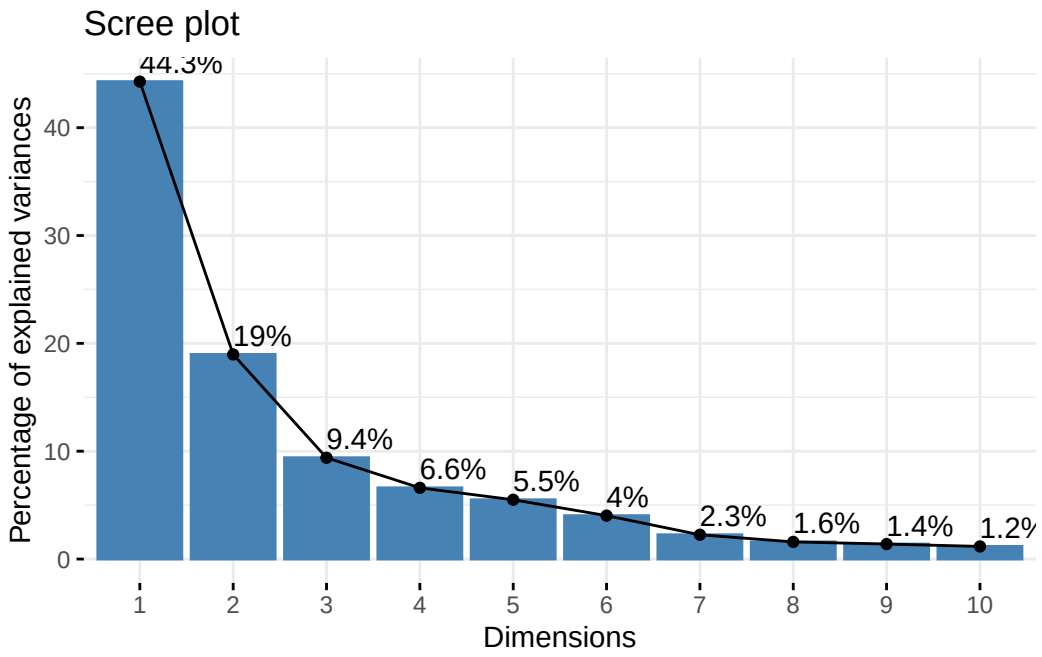
```r
which(cumsum(v/sum(v)) > 0.8)
```

```
 [1]  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
[26] 30
```

```r
library(factoextra)
```

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

```r
fviz_eig(pca, addlabels = TRUE)
```

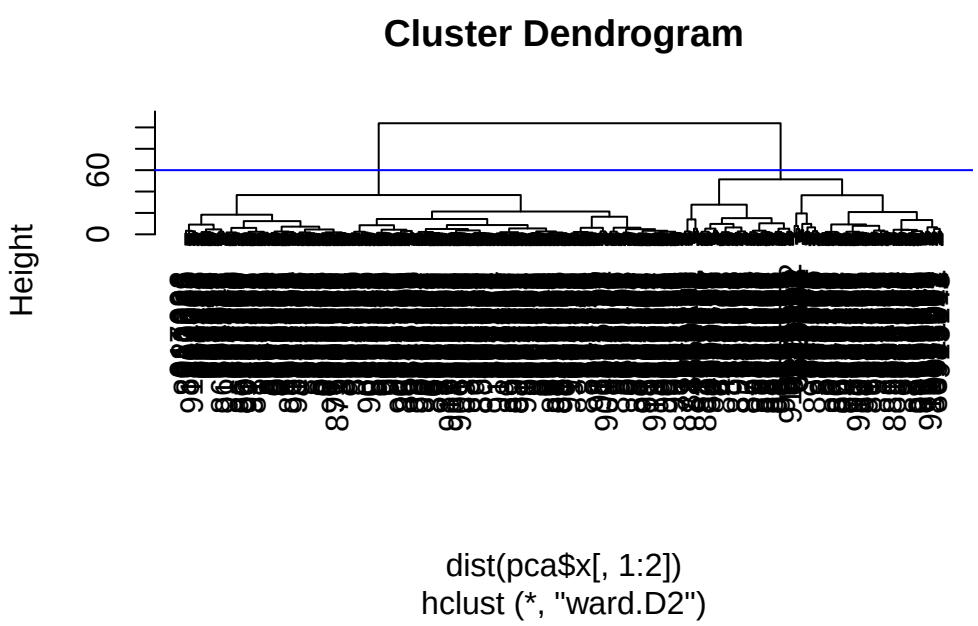


Combine PCA and clustering

We saw earlier that clustering the raw data alone did not provide useful results.

We can use our new PC variables (our PCs) as a basis for clustering. Use our `$x` PC scores. Don't include all PCs, it would be the same as previously done `hc.raw`. Let's cluster in PC1-2 subspace.

```
hc.pca <- hclust(dist(pca$x[,1:2]), method = "ward.D2")
plot(hc.pca)
abline(h=60, col="blue")
```



Q6. Does your clustering help separate cancer from non-cancer diagnoses (i.e. M vs B.)?

```
grps2 <- cutree (hc.pca, h= 60)
crosstable <- table(grps2, diagnosis)
```

Positive cancer samples “M” Negative non-cancer samples “B”

True is our cluster/group 1 False is our cluster/group 2

Q7. How many true positives (TP) do we have?

Q8. How many false positives (FP) do we have?

Sensitivity: TP/(TP+FN)

Specificity: TN/(TN+FN)

Prediction with our PCA model

We can take new data (in this case from UofM) and project it onto our new variables (PCs)

Read the UofM data

```
url <- "https://tinyurl.com/new-samples-CSV"
new <- read.csv(url)
```

Projection

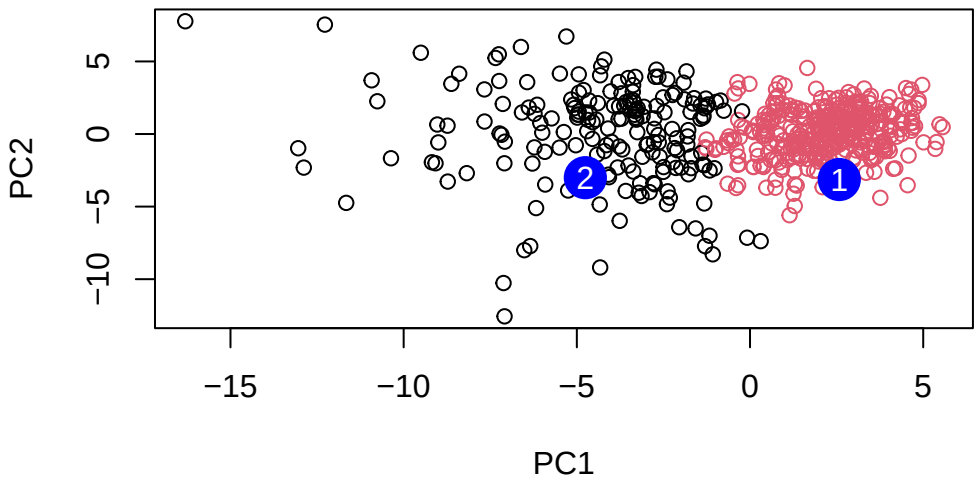
```
npc <- predict(pca, newdata=new)
npc
```

```
          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
```

Base R plot

```
plot(pca$x[,1:2], col=grps2)

#Now I will add the new points from UofM data
points(npc[,1], npc[,2], col="blue", pch=16, cex=3)
text(npc[,1], npc[,2], c(1,2), col="white")
```



We should follow up on patient 2.