Mini Project: Unsupervised Learning Analysis of Cancer Cells

Background

The goal of this hands-on session is for you to explore a complete analysis using the unsupervised learning techniques covered in the last class. You'll extend what you've learned by combining PCA as a preprocessing step to clustering using data that consist of measurements of cell nuclei of human breast masses. This expands on our RNA-Seq analysis from last day.

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. For example `radius` (i.e. mean of distances from center to points on the perimeter), `texture` (i.e. standard deviation of gray-scale values), and `smoothness` (local variation in radius lengths). Summary information is also provided for each group of cells including `diagnosis` (i.e. benign (not cancerous) and and malignant (cancerous)).

Section 1.

Preparing the data

Before we can begin our analysis we first have to download and import our data correctly into our R session.

For this we can use the `read.csv()` function to read the CSV (comma-separated values) file containing the data from the URL: https://bioboot.github.io/bimm143_W18/class-material/WisconsinCancer.csv

Assign the result to an object called `wisc.df`.

```
url <- "https://bioboot.github.io/bimm143_W18/class-material/WisconsinCancer.csv"

# Complete the following code to input the data and store as wisc.df
wisc.df <- ___
```

Examine your input data to ensure column names are set correctly. The `id` and `diagnosis` columns will not be used for most of the following steps. Use `as.matrix()` to convert the other features (i.e. columns) of the data (in columns 3 through 32) to a matrix. Store this in a variable called `wisc.data`.

```
# Convert the features of the data: wisc.data
wisc.data <- as.matrix( ___ )
```

Assign the row names of `wisc.data` the values currently contained in the `id` column of `wisc.df`. While not strictly required, this will help you keep track of the different observations throughout the modeling process.

```
# Set the row names of wisc.data
row.names(wisc.data) <- wisc.df$id
#head(wisc.data)
```

Finally, setup a separate new vector called `diagnosis` to be `1` if a diagnosis is malignant (`"M"`) and `0` otherwise. Note that R coerces `TRUE` to `1` and `FALSE` to `0`.

```
# Create diagnosis vector by completing the missing code
diagnosis <- as.numeric( ___ )
```

Exploratory data analysis

The first step of any data analysis, unsupervised or supervised, is to familiarize yourself with the data.

Explore the data you created before (`wisc.data` and `diagnosis`) to answer the following questions:

- **Q1**. How many observations are in this dataset?
- **Q2**. How many variables/features in the data are suffixed with `_mean`?
- **Q3**. How many of the observations have a malignant diagnosis?

  The functions `dim()`, `length()`, `grep()` and `sum()` may be useful for answering the first 3 questions above.

Section 2.

Performing PCA

The next step in your analysis is to perform principal component analysis (PCA) on `wisc.data`.

It is important to check if the data need to be scaled before performing PCA. Recall two common reasons for scaling data include:

- The input variables use different units of measurement.
- The input variables have significantly different variances.

Check the mean and standard deviation of the features (i.e. columns) of the `wisc.data` to determine if the data should be scaled. Use the `colMeans()` and `apply()` functions like you've done before.

```
# Check column means and standard deviations
colMeans(wisc.data)

apply(wisc.data,2,sd)
```

Execute PCA with the `prcomp()` function on the `wisc.data`, scaling if appropriate, and assign the output model to `wisc.pr`.

```
# Perform PCA on wisc.data by completing the following code
wisc.pr <- prcomp( ___ )
```

Inspect a summary of the results with the `summary()` function.

```
# Look at summary of results
summary(wisc.pr)
```

```
## Importance of components%s:
##                            PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     3.6444 2.3857 1.67867 1.40735 1.28403 1.09880
## Proportion of Variance 0.4427 0.1897 0.09393 0.06602 0.05496 0.04025
## Cumulative Proportion  0.4427 0.6324 0.72636 0.79239 0.84734 0.88759
##                            PC7     PC8    PC9    PC10   PC11    PC12
## Standard deviation     0.82172 0.69037 0.6457 0.59219 0.5421 0.51104
## Proportion of Variance 0.02251 0.01589 0.0139 0.01169 0.0098 0.00871
## Cumulative Proportion  0.91010 0.92598 0.9399 0.95157 0.9614 0.97007
##                           PC13    PC14    PC15    PC16    PC17    PC18
## Standard deviation     0.49128 0.39624 0.30681 0.28260 0.24372 0.22939
## Proportion of Variance 0.00805 0.00523 0.00314 0.00266 0.00198 0.00175
## Cumulative Proportion  0.97812 0.98335 0.98649 0.98915 0.99113 0.99288
##                           PC19    PC20   PC21    PC22    PC23   PC24
```

```
## Standard deviation     0.22244 0.17652 0.1731 0.16565 0.15602 0.1344
## Proportion of Variance 0.00165 0.00104 0.0010 0.00091 0.00081 0.0006
## Cumulative Proportion  0.99453 0.99557 0.9966 0.99749 0.99830 0.9989
##                            PC25    PC26    PC27    PC28    PC29    PC30
## Standard deviation     0.12442 0.09043 0.08307 0.03987 0.02736 0.01153
## Proportion of Variance 0.00052 0.00027 0.00023 0.00005 0.00002 0.00000
## Cumulative Proportion  0.99942 0.99969 0.99992 0.99997 1.00000 1.00000
```

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

- **Q5**. How many principal components (PCs) are required to describe at least 70% of the original variance in the data?
- **Q6**. How many principal components (PCs) are required to describe at least 90% of the original variance in the data?

Interpreting PCA results

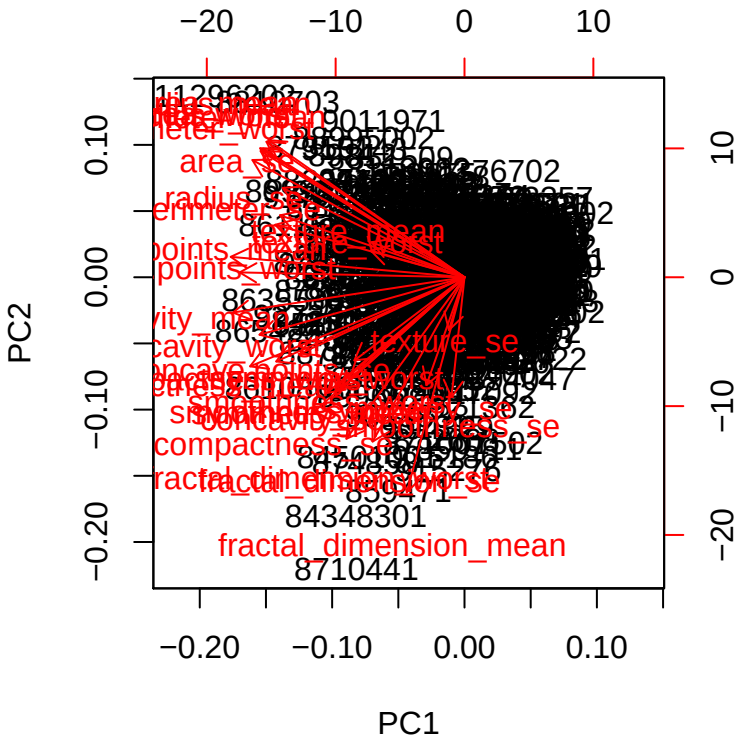
Now you will use some visualizations to better understand your PCA model. You were introduced to one of these visualizations, the biplot, last day.

You will often run into some common challenges with using biplots on real-world data containing a non-trivial number of observations and variables, then you'll look at some alternative visualizations. You are encouraged to experiment with additional visualizations before moving on to the next section

Create a biplot of the `wisc.pr` using the `biplot()` function.

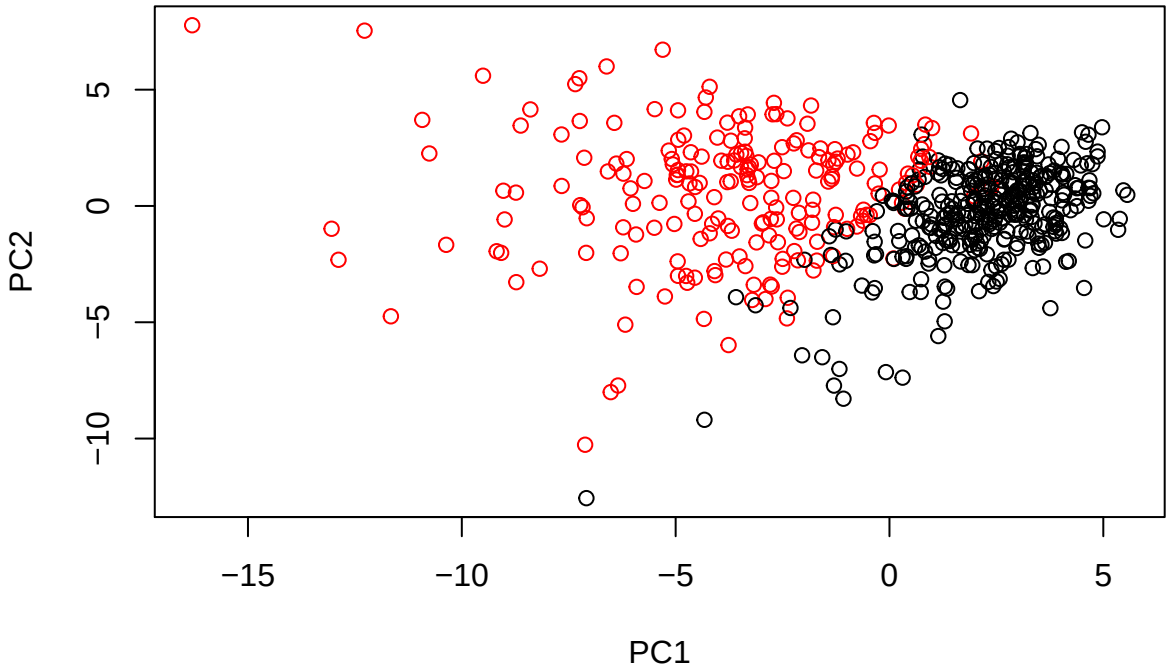
- **Q7.** What stands out to you about this plot? Is it easy or difficult to understand? Why?

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



Rownames are used as the plotting character for biplots like this one which can make trends rather hard

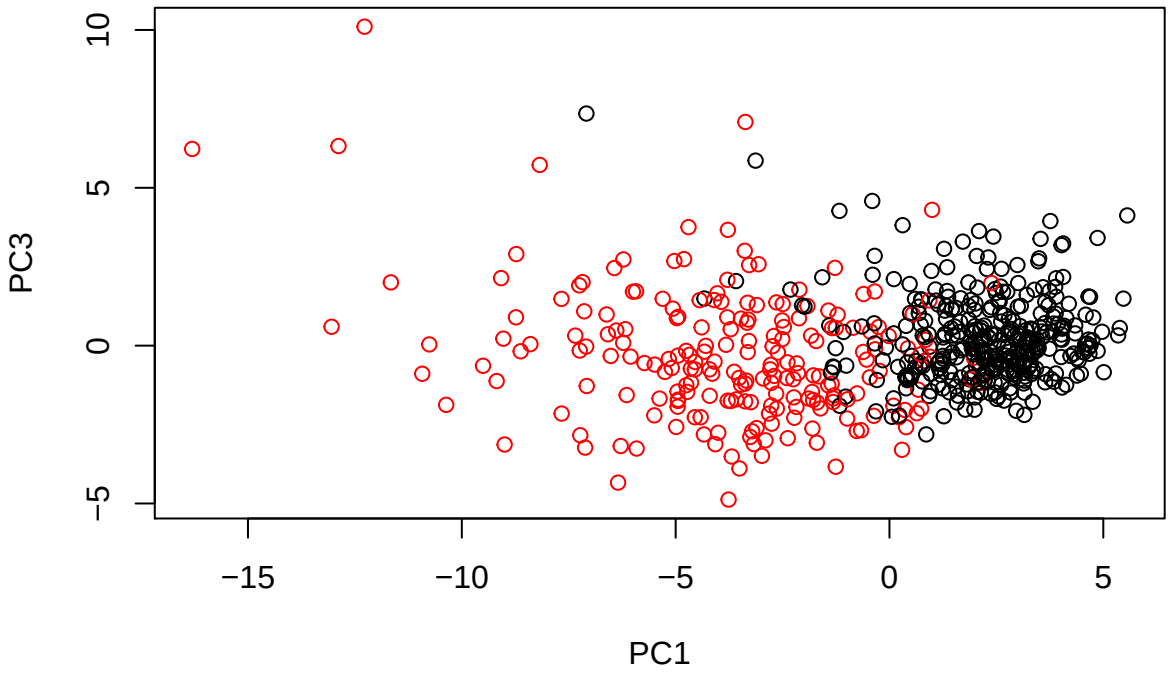
to see. Lets generate a more standard scatter plot of each observation along principal components 1 and 2 (i.e. a plot of PC1 vs PC2 available as the first two columns of `wisc.pr$x`) and color the points by the diagnosis (available in the `diagnosis` vector you created earlier).




```
# Scatter plot observations by components 1 and 2
plot( ___ , col = ___ ,
     xlab = "PC1", ylab = "PC2")
```

- **Q8.** Repeat the same for principal components 1 and 3. What do you notice about these plots?

```
# Repeat for components 1 and 3
plot(wisc.pr$x[, c(1, 3)], col = (diagnosis + 1),
     xlab = "PC1", ylab = "PC3")
```




```
# Do additional data exploration of your choosing below (optional)
```

Because principal component 2 explains more variance in the original data than principal component 3, you can see that the first plot has a cleaner cut separating the two subgroups.

> Overall, the plots indicate that principal component 1 is capturing a separation of malignant from benign samples. This is an important and interesting result worthy of further exploration - as we will do in the next sections!

Variance explained

In this exercise, you will produce scree plots showing the proportion of variance explained as the number of principal components increases. The data from PCA must be prepared for these plots, as there is not a built-in function in base R to create them directly from the PCA model.

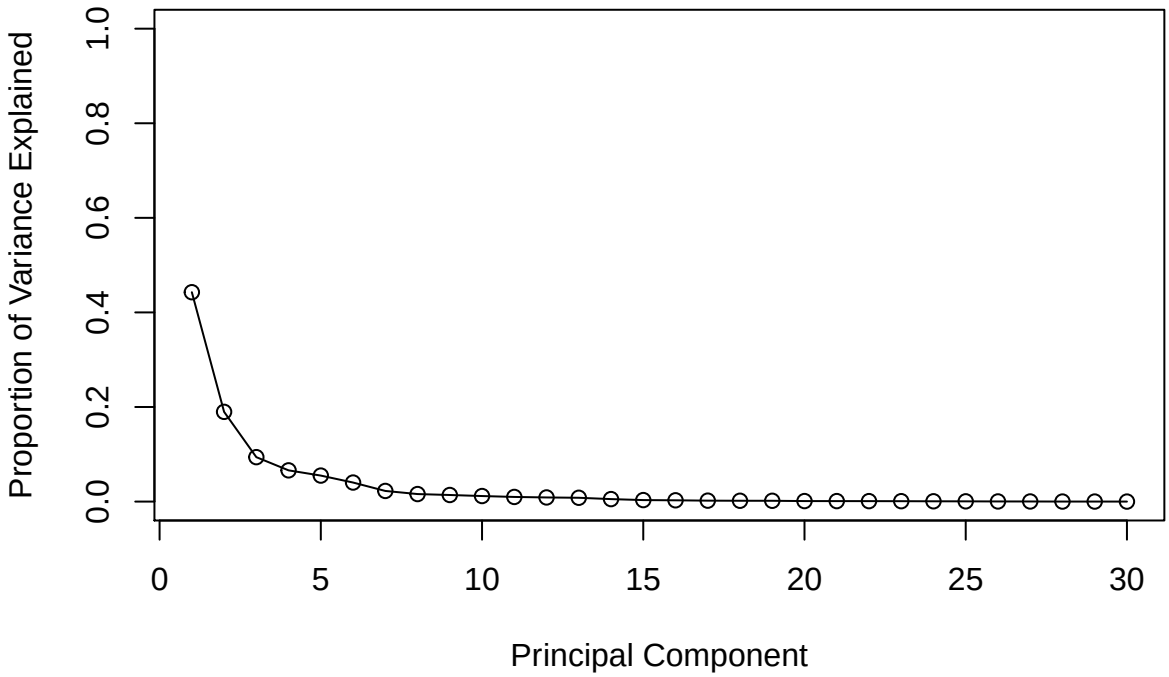
As you look at these plots, ask yourself if there's an 'elbow' in the amount of variance explained that might lead you to pick a natural number of principal components. If an obvious elbow does not exist, as is typical in some real-world datasets, consider how else you might determine the number of principal components to retain based on the scree plot.

Calculate the variance of each principal component by squaring the sdev component of `wisc.pr` (i.e. `wisc.pr$sdev^2`). Save the result as an object called `pr.var`.

Calculate the variance explained by each principal component by dividing by the total variance explained of all principal components. Assign this to a variable called `pve` and create a plot of variance explained for each principal component.

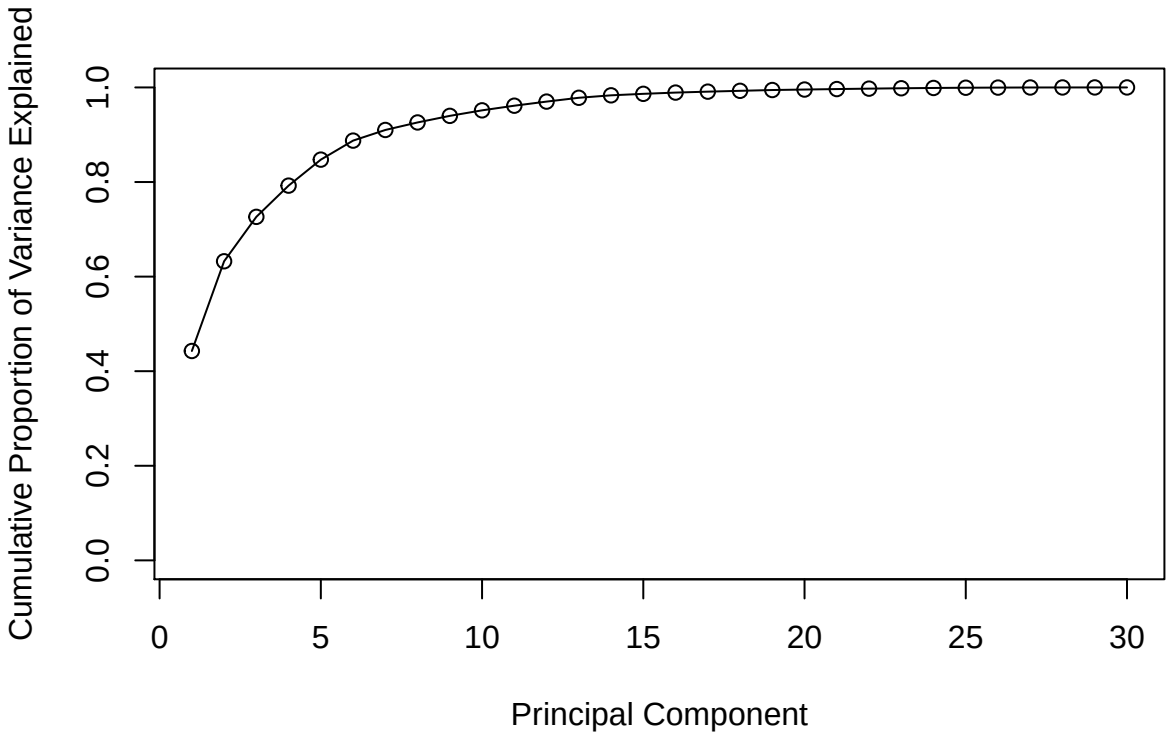
```
# Variance explained by each principal component: pve
pve <- ___ / ___

# Plot variance explained for each principal component
plot(pve, xlab = "Principal Component",
     ylab = "Proportion of Variance Explained",
     ylim = c(0, 1), type = "o")
```



Using the `cumsum()` function, create a plot of cumulative proportion of variance explained.

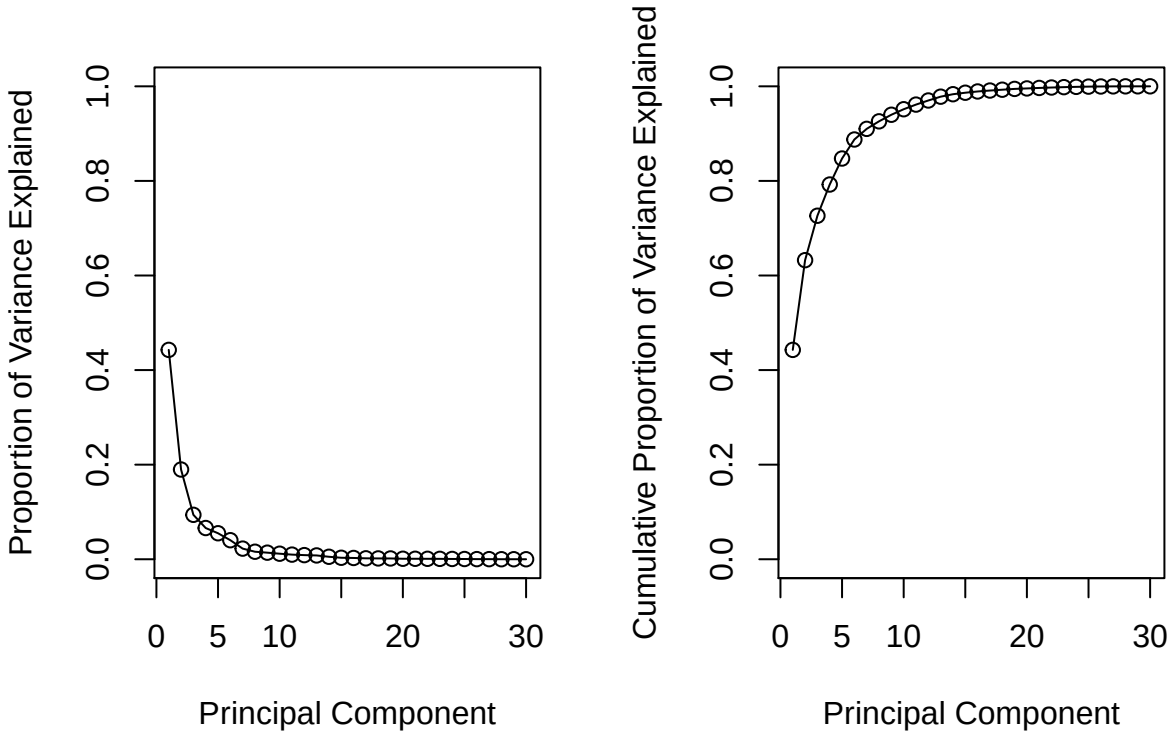
```
# Plot cumulative proportion of variance explained
plot(cumsum(pve), xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
     ylim = c(0, 1), type = "o")
```




```
# Plot cumulative proportion of variance explained
plot( ___ , xlab = "Principal Component",
     ylab = "Cumulative Proportion of Variance Explained",
```

```
     ylim = c(0, 1), type = "o")
```

Use the `par()` function to create a side by side plot (i.e. 1 row 2 column arrangement) of these two graphs.



Communicating PCA results

In this section we will check your understanding of the PCA results, in particular the loadings and variance explained. The loadings, represented as vectors, explain the mapping from the original features to the principal components. The principal components are naturally ordered from the most variance explained to the least variance explained.

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

Section 3.

Hierarchical clustering of case data

The goal of this section is to do hierarchical clustering of the observations. Recall from our last class that this type of clustering does not assume in advance the number of natural groups that exist in the data.

As part of the preparation for hierarchical clustering, the distance between all pairs of observations are computed. Furthermore, there are different ways to link clusters together, with single, complete, and average being the most common linkage methods.

Scale the `wisc.data` data and assign the result to `data.scaled`.

```
# Scale the wisc.data data: data.scaled
data.scaled <- ___(wisc.data)
```

Calculate the (Euclidean) distances between all pairs of observations in the new scaled dataset and assign the result to `data.dist`.

```
data.dist <- ___(data.scaled)
```

Create a hierarchical clustering model using complete linkage. Manually specify the method argument to hclust() and assign the results to `wisc.hclust`.

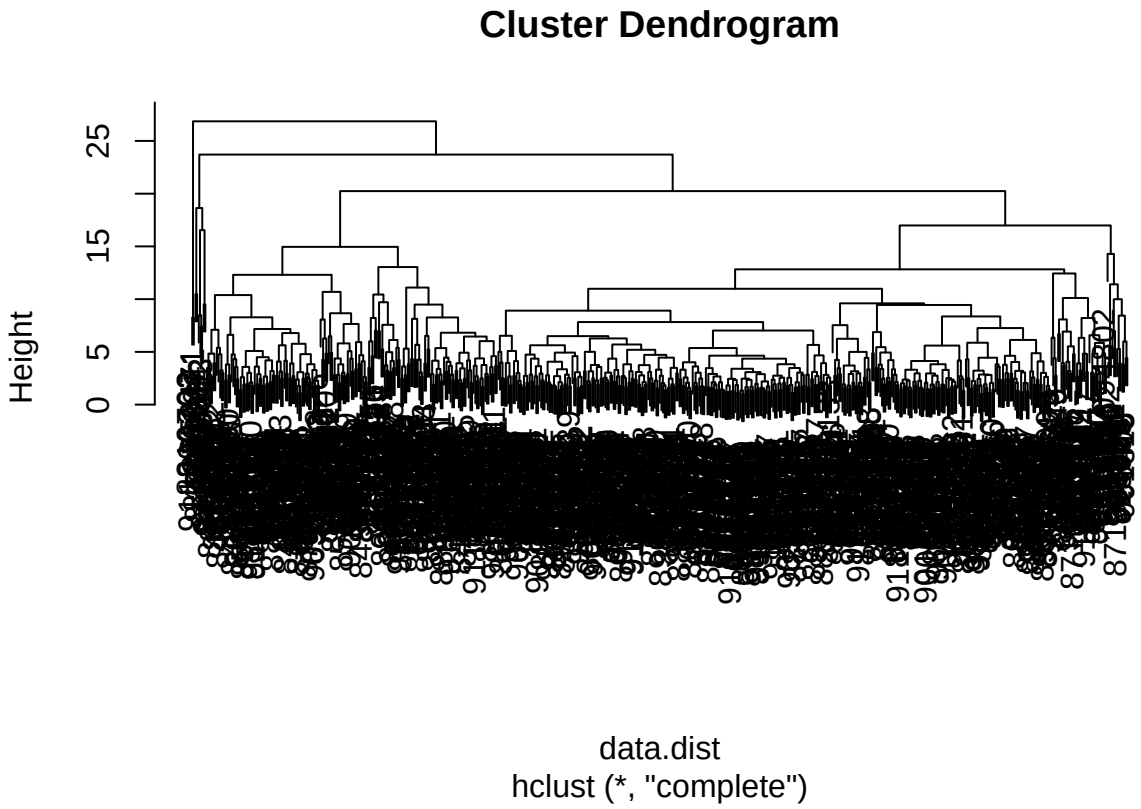
```
wisc.hclust <- ___(data.dist, ___)
```

Results of hierarchical clustering

Let's use the hierarchical clustering model you just created to determine a height (or distance between clusters) where a certain number of clusters exists.

- **Q11.** Using the `plot()` function, what is the height at which the clustering model has 4 clusters?

```
plot(wisc.hclust)
```



Selecting number of clusters

In this section, you will compare the outputs from your hierarchical clustering model to the actual diagnoses. Normally when performing unsupervised learning like this, a target variable (i.e. known answer or labels) isn't available. We do have it with this dataset, however, so it can be used to check the performance of the clustering model.

When performing supervised learning - that is, when you're trying to predict some target variable of interest and that target variable is available in the original data - using clustering to create new features may or may not improve the performance of the final model.

This exercise will help you determine if, in this case, hierarchical clustering provides a promising new feature.

Use `cutree()` to cut the tree so that it has 4 clusters. Assign the output to the variable `wisc.hclust.clusters`.

```
wisc.hclust.clusters <- ___
```

We can use the `table()` function to compare the cluster membership to the actual diagnoses.

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

```
##                     diagnosis
## wisc.hclust.clusters   0   1
##                    1  12 165
##                    2   2   5
##                    3 343  40
##                    4   0   2
```

Here we picked four clusters and see that cluster 1 largely corresponds to malignant cells (with `diagnosis` values of 1) whilst cluster 3 largely corresponds to benign cells (with `diagnosis` values of 0).

Before moving on, explore how different numbers of clusters affect the ability of the hierarchical clustering to separate the different diagnoses.

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

Section 4.

K-means clustering and comparing results

As you now know, there are two main types of clustering: hierarchical and k-means.

In this section, you will create a k-means clustering model on the Wisconsin breast cancer data and compare the results to the actual diagnoses and the results of your hierarchical clustering model. Take some time to see how each clustering model performs in terms of separating the two diagnoses and how the clustering models compare to each other.

Create a k-means model on `wisc.data`, assigning the result to `wisc.km`. Be sure to create 2 clusters, corresponding to the actual number of diagnosis. Also, remember to scale the data (with the `scale()` function and repeat the algorithm 20 times (by setting setting the value of the `nstart` argument appropriately). Running multiple times such as this will help to find a well performing model.

```
wisc.km <- kmeans(___, centers= ___, nstart= ___)
```

Use the `table()` function to compare the cluster membership of the k-means model (`wisc.km$cluster`) to the actual diagnoses contained in the `diagnosis` vector.

```
table(___, ___)
```

- **Q13**. How well does k-means separate the two diagnoses? How does it compare to your hclust results?

Use the table() function to compare the cluster membership of the k-means model (`wisc.km$cluster`) to your hierarchical clustering model from above (`wisc.hclust.clusters`). Recall the cluster membership of the hierarchical clustering model is contained in `wisc.hclust.clusters` object.

```
table(___, ___)
```

```
##
## wisc.hclust.clusters   1   2
##                    1 160  17
##                    2   7   0
##                    3  20 363
##                    4   2   0
```

Looking at the second table you generated, it looks like clusters 1, 2, and 4 from the hierarchical clustering model can be interpreted as the cluster 1 equivalent from the k-means algorithm, and cluster 3 can be interpreted as the cluster 2 equivalent.

Section 5.

Clustering on PCA results

In this final section, you will put together several steps you used earlier and, in doing so, you will experience some of the creativity and open endedness that is typical in unsupervised learning.

Recall from earlier sections that the PCA model required significantly fewer features to describe 70%, 80% and 95% of the variability of the data. In addition to normalizing data and potentially avoiding over-fitting, PCA also uncorrelates the variables, sometimes improving the performance of other modeling techniques.

Let's see if PCA improves or degrades the performance of hierarchical clustering.

Using the minimum number of principal components required to describe at least 90% of the variability in the data, create a hierarchical clustering model with complete linkage. Assign the results to `wisc.pr.hclust`.

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

Cut this hierarchical clustering model into 4 clusters and assign the results to `wisc.pr.hclust.clusters`.

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

Using `table()`, compare the results from your new hierarchical clustering model with the actual diagnoses.

- **Q14**. How well does the newly created model with four clusters separate out the two diagnoses?

```
# Compare to actual diagnoses
table(___, diagnosis)
```

```
##                        diagnosis
## wisc.pr.hclust.clusters   0   1
##                       1   5 113
##                       2 350  97
##                       3   2   0
##                       4   0   2
```

- **Q15**. 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.

```
table(___, diagnosis)
table(___, diagnosis)
```

```
##    diagnosis
##       0   1
##   1  14 175
##   2 343  37
```

```
##                      diagnosis
## wisc.hclust.clusters   0   1
##                    1  12 165
##                    2   2   5
##                    3 343  40
##                    4   0   2
```

Section 6.

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.

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.

- **Q16.** Which of your analysis procedures resulted in a clustering model with the best specificity? How about sensitivity?

Section 7.

PCA of protein structure data

Visit the Bio3D-web PCA app < http://bio3d.ucsd.edu > and explore how PCA of large protein structure sets can provide considerable insight into major features and trends with clear biological mechanistic insight. Note that the final report generated from this app contains all the R code required to run the analysis yourself - including PCA and clustering. We will delve more into this type of analysis in the next class.

About this document

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing any code chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Here we use the `sessionInfo()` function to report on our R systems setup at the time of document execution.

```r
sessionInfo()
```

```
## R version 3.4.1 (2017-06-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Sierra 10.12.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/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
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.4.1  backports_1.1.2 magrittr_1.5    rprojroot_1.3-2
##  [5] tools_3.4.1     htmltools_0.3.6 yaml_2.1.16     Rcpp_0.12.14
##  [9] stringi_1.1.6   rmarkdown_1.8   knitr_1.18      stringr_1.2.0
## [13] digest_0.6.14   evaluate_0.10.1
```