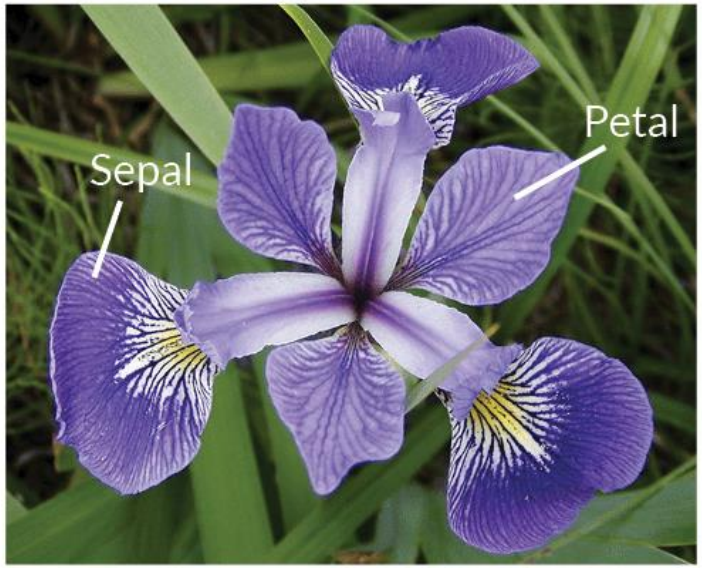
Cluster Analysis

- Belongs to the unsupervised machine learning subdiscipline
- The objective is to find “natural” grouping of observations in the data
- Applications in market analysis, genomics, housing, social network analysis, among many other applications
- The core concept behind all clustering methods is measuring “similarity” between observations

The iris dataset

```
> data("iris")
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

What can we do if we don't have the Species labels?



Iris Versicolor

Iris Setosa

Iris Virginica

The iris dataset

Use case A (with labels): Predictive model

x

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2

-->

y

Species
setosa

Use case B (without labels): Clustering

x

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

-->

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2
5.4	3.9	1.7	0.4

Distance between observations

- All clustering algorithms rely on determining how similar/dissimilar observations are
- The most straight forward measure of similarity is the Euclidian distance; in which each variable is treated as a coordinate
- The larger the distance between two observations, the less similar the observations are
- Clustering algorithms will “somehow” bundle observations that are similar into the same group (cluster)

Example of distance calculation in R

```
set.seed(1234)
iris_sample <- iris %>%
                select(-Species) %>%
                sample_n(10)
iris_sample
```

x_1 x_2 x_3 x_4

```
> iris_sample
   Sepal.Length Sepal.Width Petal.Length Petal.Width
1           5.2         3.5          1.5         0.2
2           5.7         2.6          3.5         1.0
3           6.3         3.3          6.0         2.5
4           6.5         3.2          5.1         2.0
5           6.3         3.4          5.6         2.4
6           6.4         2.8          5.6         2.2
7           6.8         3.2          5.9         2.3
8           7.9         3.8          6.4         2.0
9           6.2         2.9          4.3         1.3
10          7.1         3.0          5.9         2.1
```

Distance between obs. 2 & obs. 7 = $[(x_{21}-x_{71})^2+ (x_{22}-x_{72})^2+ (x_{23}-x_{73})^2+ (x_{24}-x_{74})^2]^{1/2}$

Implies that the variables (features) must be numeric

Example of distance calculation in R

```
dist_mat <- dist(iris_sample, method = "euclidean", diag = TRUE, upper = TRUE)
```

```
> dist_mat
          1         2         3         4         5         6         7         8         9        10
1  0.0000000 2.3874673 5.1759057 4.2402830 4.7822589 4.7686476 5.1400389 5.8847260 3.2264532 5.1797683
2  2.3874673 0.0000000 3.0577770 2.1354157 2.7147744 2.5258662 3.0033315 3.9610605 1.0344080 3.0149627
3  5.1759057 3.0577770 0.0000000 1.0535654 0.4242641 0.7141428 0.5567764 1.7944358 2.1213203 0.9486833
4  4.2402830 2.1354157 1.0535654 0.0000000 0.7000000 0.6782330 0.9055385 2.0024984 1.1445523 1.0246951
5  4.7822589 2.7147744 0.4242641 0.7000000 0.0000000 0.6403124 0.6244998 1.8761663 1.7776389 0.9899495
6  4.7686476 2.5258662 0.7141428 0.6782330 0.6403124 0.0000000 0.6480741 1.9824228 1.5968719 0.7937254
7  5.1400389 3.0033315 0.5567764 0.9055385 0.6244998 0.6480741 0.0000000 1.3820275 2.0024984 0.4123106
8  5.8847260 3.9610605 1.7944358 2.0024984 1.8761663 1.9824228 1.3820275 0.0000000 2.9325757 1.2409674
9  3.2264532 1.0344080 2.1213203 1.1445523 1.7776389 1.5968719 2.0024984 2.9325757 0.0000000 2.0049938
10 5.1797683 3.0149627 0.9486833 1.0246951 0.9899495 0.7937254 0.4123106 1.2409674 2.0049938 0.0000000
```

Notice that observations 7 and 10 have the closet distance; therefore, they are most similar

Likewise, observations 1 and 8 are farthest apart

A concise form of the distance matrix

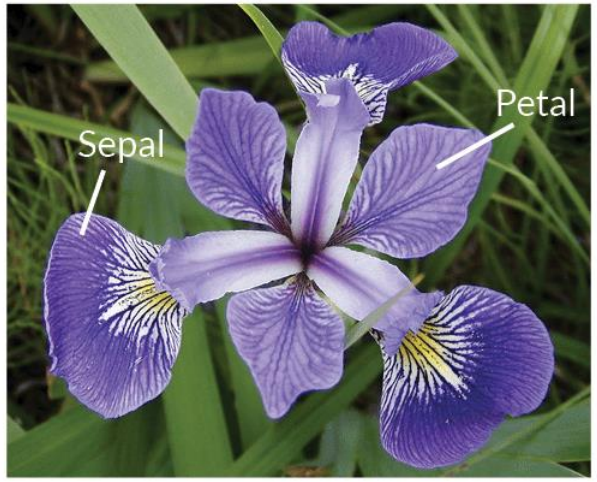
```
dist_mat <- dist(iris_sample, method = "euclidean")
```

```
> dist_mat
           1         2         3         4         5         6         7         8         9
2  2.6692696
3  5.7868385 3.4186986
4  4.7407805 2.3874673 1.1779219
5  5.3467280 3.0352100 0.4743416 0.7826238
6  5.3315101 2.8240042 0.7984360 0.7582875 0.7158911
7  5.7467382 3.3578267 0.6224950 1.0124228 0.6982120 0.7245688
8  6.5793237 4.4286002 2.0062403 2.2388613 2.0976177 2.2164160 1.5451537
9  3.6072843 1.1565034 2.3717082 1.2796484 1.9874607 1.7853571 2.2388613 3.2787193
10 5.7911571 3.3708308 1.0606602 1.1456439 1.1067972 0.8874120 0.4609772 1.3874437 2.2416512
```

Concise version of the distance matrix

The importance of variable normalization

```
> data("iris")
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```



Iris Versicolor

Iris Setosa

Iris Virginica

- Each variable relates to a different feature with a different scale
- Calculating the distances on the raw variables biases the result towards the variables with the larger scales

The importance of variable normalization

```
> house_data
# A tibble: 9 x 3
  bedroom bathroom  price
    <dbl>    <dbl>  <dbl>
1       2        1 100000
2       2        1 106000
3       3        1.5 160000
4       1        1  90000
5       2        2 130000
6       3        1 200000
7       3        2 100000
8       3        1.5 205000
9       4        2 150000
```

- The effect of different scales could be extreme as in this example
- For the house attributes dataset, the price in dollars is orders of magnitude larger than the number of bedrooms or bathrooms
- Calculating the distances on the raw data is as good as using a single variable (the price) for the calculation and cluster analysis that follows

Variable normalization in R

```
iris_sample_scaled <- scale(iris_sample)
```

```
> iris_sample_scaled
      Sepal.Length Sepal.Width Petal.Length Petal.Width
 [1,]   -1.6832752  0.92681355  -2.31736942  -2.1728579
 [2,]   -1.0045352 -1.60085976  -0.98554791  -1.0864290
 [3,]   -0.1900472  0.36510837   0.67922897   0.9506253
 [4,]    0.0814488  0.08425578   0.07990929   0.2716072
 [5,]   -0.1900472  0.64596096   0.41286467   0.8148217
 [6,]   -0.0542992 -1.03915458   0.41286467   0.5432145
 [7,]    0.4886928  0.08425578   0.61263789   0.6790181
 [8,]    1.9819207  1.76937131   0.94559327   0.2716072
 [9,]   -0.3257952 -0.75830199  -0.45281931  -0.6790181
[10,]    0.8959368 -0.47744940   0.61263789   0.4074109
```

```
> apply(iris_sample_scaled,2,mean)
 Sepal.Length   Sepal.Width  Petal.Length   Petal.Width
-4.919351e-16  1.249001e-16 -2.775558e-16 -6.103516e-17
> apply(iris_sample_scaled,2,sd)
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
           1            1            1            1
```

Scaling a variable involves subtracting the mean and dividing by the standard deviation

Distance calculation for categorical variables

Assume that the iris dataset has an additional variable—color:

```
> iris_sample
   Sepal.Length Sepal.Width Petal.Length Petal.Width  color
1           5.2         3.5          1.5         0.2 purple
2           5.7         2.6          3.5         1.0 yellow
3           6.3         3.3          6.0         2.5 purple
4           6.5         3.2          5.1         2.0 yellow
5           6.3         3.4          5.6         2.4 yellow
6           6.4         2.8          5.6         2.2   pink
7           6.8         3.2          5.9         2.3    red
8           7.9         3.8          6.4         2.0   pink
9           6.2         2.9          4.3         1.3    red
10          7.1         3.0          5.9         2.1    red
```

Calculating the distance requires that all variables are numeric

We must somehow convert “color” into a numeric variable(s)

Dummification of categorical variables

```
iris_sample_wide <- iris_sample %>%
                    mutate(dummy = 1) %>%
                    spread(color, dummy, fill = 0)
```

```
> iris_sample_wide
   Sepal.Length Sepal.Width Petal.Length Petal.Width pink purple red yellow
1           5.2         3.5          1.5         0.2    0      1   0      0
2           5.7         2.6          3.5         1.0    0      0   0      1
3           6.2         2.9          4.3         1.3    0      0   1      0
4           6.3         3.3          6.0         2.5    0      1   0      0
5           6.3         3.4          5.6         2.4    0      0   0      1
6           6.4         2.8          5.6         2.2    1      0   0      0
7           6.5         3.2          5.1         2.0    0      0   0      1
8           6.8         3.2          5.9         2.3    0      0   1      0
9           7.1         3.0          5.9         2.1    0      0   1      0
10          7.9         3.8          6.4         2.0    1      0   0      0
```

Recap

- Calculating distances between observations is a core operation to any clustering method
- In R, the `dist` function is used to calculate pairwise distances between observations in a data frame
- `dist` requires that all variable are numeric; categorical variables need to be converted to numeric by “dummification”
- Variables may need to be scaled prior to the distance calculation

How to cluster based on the distance matrix

```
> dist_mat
           1         2         3         4         5         6         7         8         9        10
1  0.0000000 2.3874673 5.1759057 4.2402830 4.7822589 4.7686476 5.1400389 5.8847260 3.2264532 5.1797683
2  2.3874673 0.0000000 3.0577770 2.1354157 2.7147744 2.5258662 3.0033315 3.9610605 1.0344080 3.0149627
3  5.1759057 3.0577770 0.0000000 1.0535654 0.4242641 0.7141428 0.5567764 1.7944358 2.1213203 0.9486833
4  4.2402830 2.1354157 1.0535654 0.0000000 0.7000000 0.6782330 0.9055385 2.0024984 1.1445523 1.0246951
5  4.7822589 2.7147744 0.4242641 0.7000000 0.0000000 0.6403124 0.6244998 1.8761663 1.7776389 0.9899495
6  4.7686476 2.5258662 0.7141428 0.6782330 0.6403124 0.0000000 0.6480741 1.9824228 1.5968719 0.7937254
7  5.1400389 3.0033315 0.5567764 0.9055385 0.6244998 0.6480741 0.0000000 1.3820275 2.0024984 0.4123106
8  5.8847260 3.9610605 1.7944358 2.0024984 1.8761663 1.9824228 1.3820275 0.0000000 2.9325757 1.2409674
9  3.2264532 1.0344080 2.1213203 1.1445523 1.7776389 1.5968719 2.0024984 2.9325757 0.0000000 2.0049938
10 5.1797683 3.0149627 0.9486833 1.0246951 0.9899495 0.7937254 0.4123106 1.2409674 2.0049938 0.0000000
```

Proposed approach

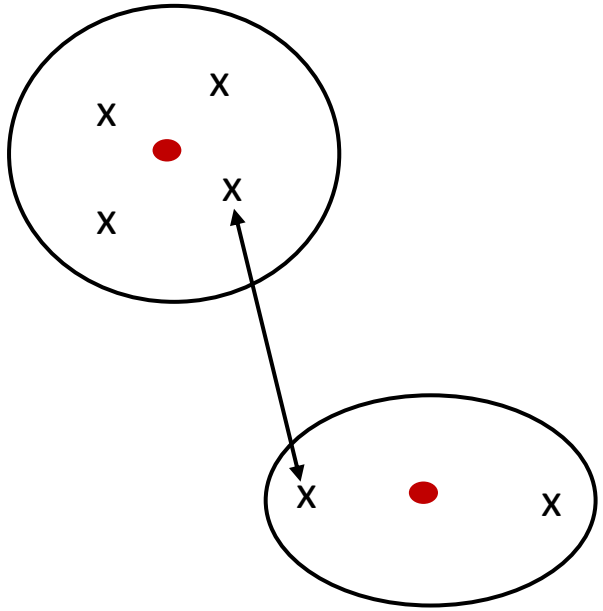
- Merge the closets pair of observations into one cluster (7 & 10 in the example above)
- We are left with 1 less observation
- Re-calculate the distance matrix
- Keep repeating this process

Issues to solve

- How to calculate the variables for the merged observations
- When to stop

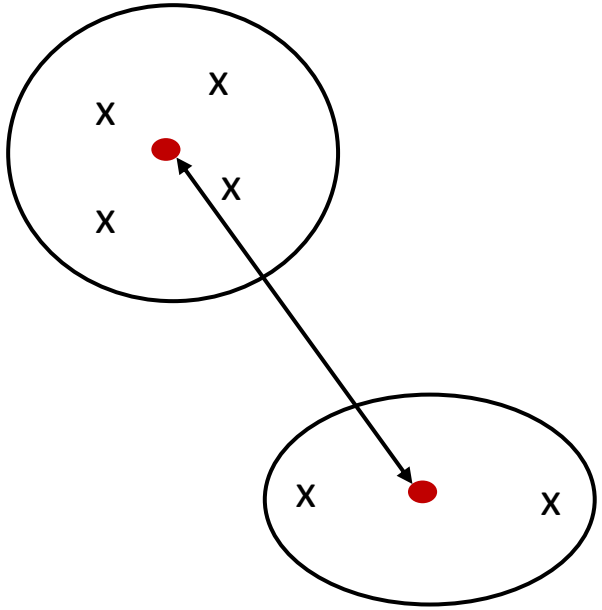
Distance between clusters

Simple linkage



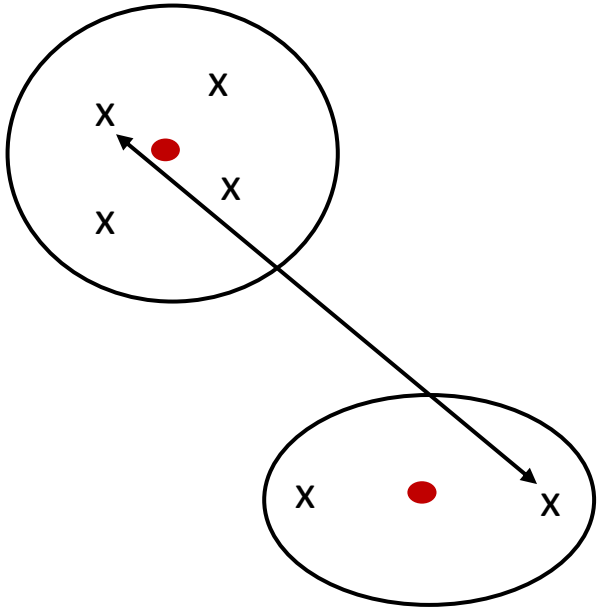
min

Average linkage



mean

Complete linkage



max

Hierarchical Clustering Algorithm

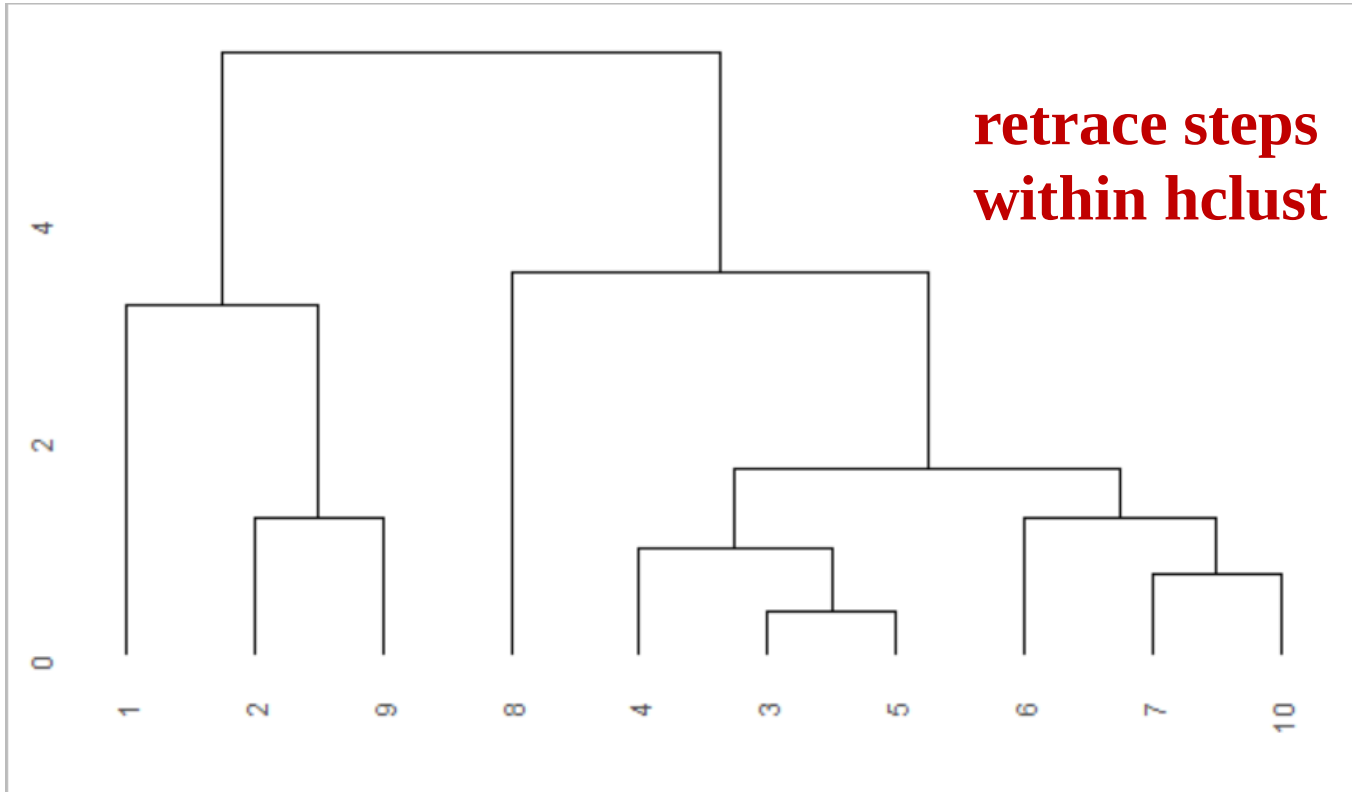
- Uses the distance matrix as input
- Starts by combining the closest pair of observations
- Subsequent distance calculations that involve a group of observations follow the specified linkage criteria
- The algorithm continues until all observations are contained within the same cluster!

Hierarchical Clustering in R

```
mod <- hclust(dist(iris_sample_scaled), method = "complete")

> mod

Call:
hclust(d = dist(iris_sample_scaled), method = "complete")

Cluster method   : complete
Distance         : euclidean
Number of objects: 10

> ggdendrogram(mod)
```

Specifying number of clusters

```
> cutree(mod, 4)
 [1] 1 2 3 3 3 3 3 4 2 3
```



retrace steps within hclust

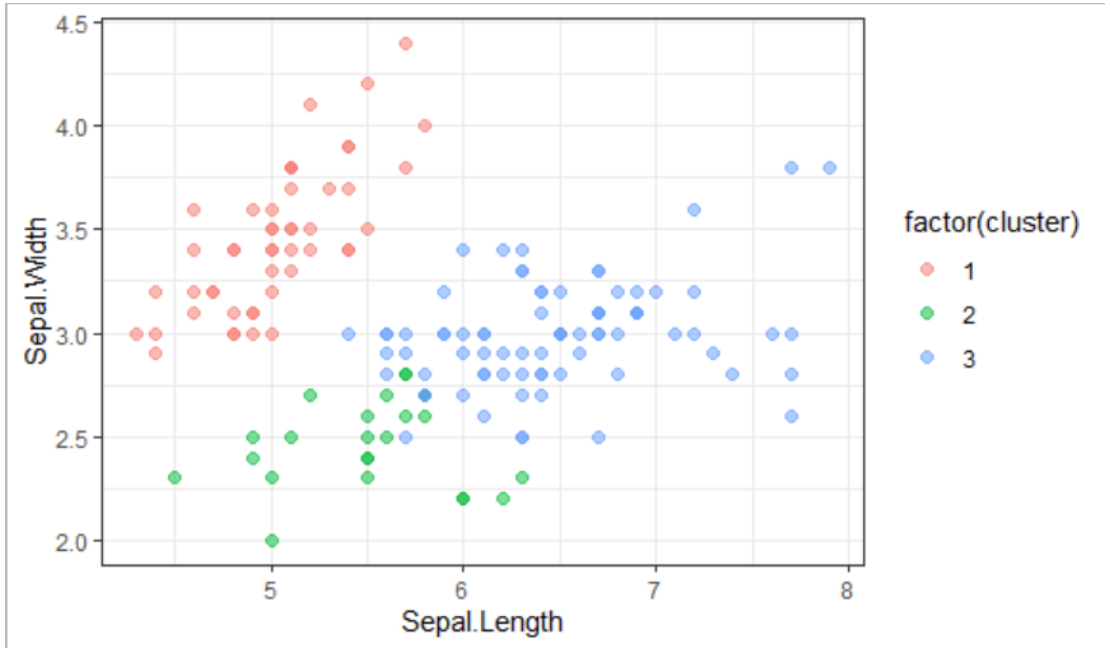
Adding a label to the original data

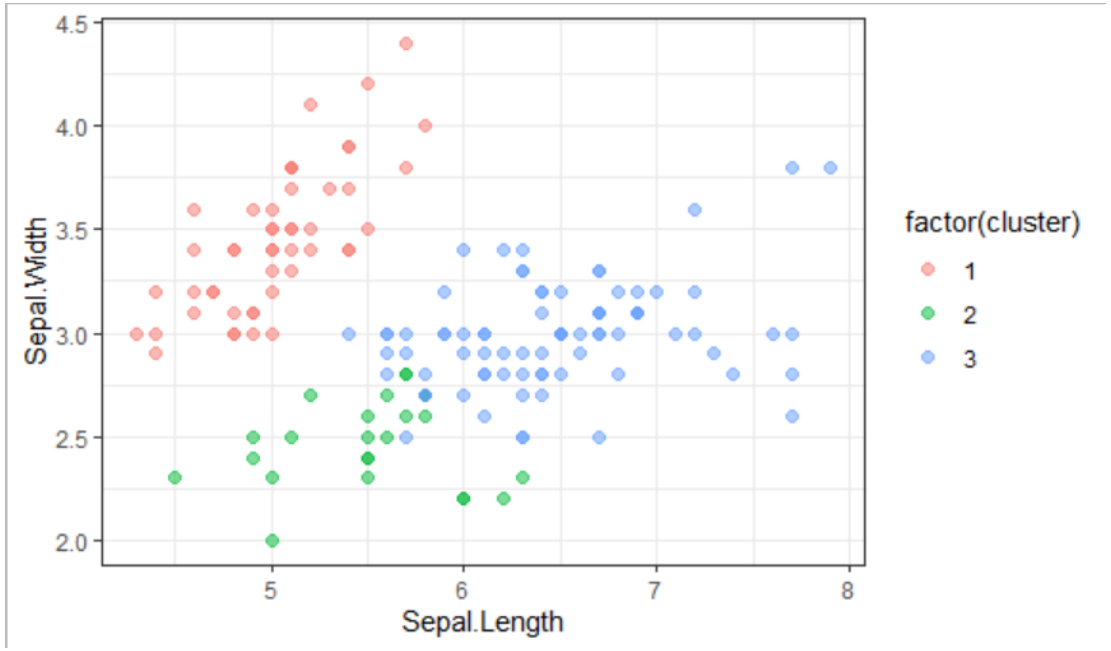
```
iris_sample <- iris_sample %>%
                mutate(cluster = cutree(mod, 4))
```

```
> iris_sample
   Sepal.Length Sepal.Width Petal.Length Petal.Width  color cluster
1           5.2         3.5          1.5         0.2 yellow       1
2           5.7         2.6          3.5         1.0 yellow       2
3           6.3         3.3          6.0         2.5 yellow       3
4           6.5         3.2          5.1         2.0   pink       3
5           6.3         3.4          5.6         2.4 yellow       3
6           6.4         2.8          5.6         2.2    red       3
7           6.8         3.2          5.9         2.3 yellow       3
8           7.9         3.8          6.4         2.0 yellow       4
9           6.2         2.9          4.3         1.3    red       2
10          7.1         3.0          5.9         2.1 yellow       3
```

Visualizing the clusters

```
iris_scaled <- scale(select(iris, -Species))
mod <- hclust(dist(iris_scaled), method = "complete")
iris_clustered <- iris %>%
                  mutate(cluster = cutree(mod, 3))

iris_clustered %>%
  ggplot(aes(x=Sepal.Length, y=Sepal.Width, col=factor(cluster))) +
        geom_point(size=2, alpha=0.5) +
        theme_bw()
```

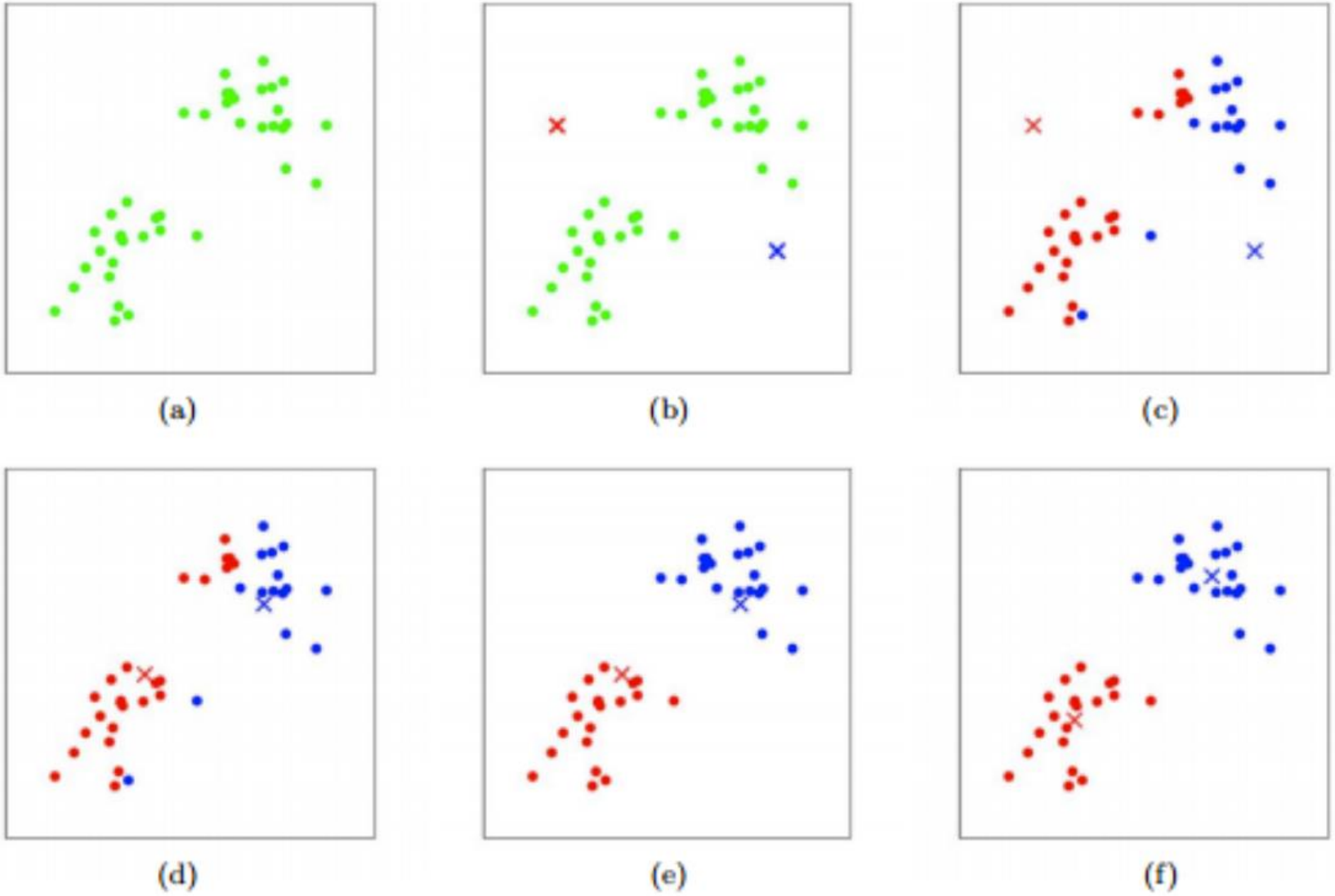




Exercise

- Apply what you learned so far to cluster cars in the `mtcars` dataset; report your findings to the class

K-means Clustering Algorithm



K-means Clustering in R

```
mod_km <- kmeans(dist(iris_sample_scaled), centers = 3)
```

```
> mod_km
K-means clustering with 3 clusters of sizes 2, 1, 7

Cluster means:
         1        2        3         4        5        6         7        8        9       10
1 1.565559 1.565559 3.998421 3.2941810 3.840967 3.517538 3.9915729 5.315542 2.248182 3.996689
2 5.546702 5.084382 2.687269 2.6834338 2.560933 3.520168 2.3121823 0.000000 3.817582 2.521283
3 4.330631 2.780098 1.062661 0.9201941 1.112061 1.194832 0.9409233 2.871836 1.650861 1.154996

Clustering vector:
 1  2  3  4  5  6  7  8  9 10
 1  1  3  3  3  3  3  2  3  3

Within cluster sum of squares by cluster:
[1] 18.90879  0.00000 26.44984
 (between_SS / total_SS =  73.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

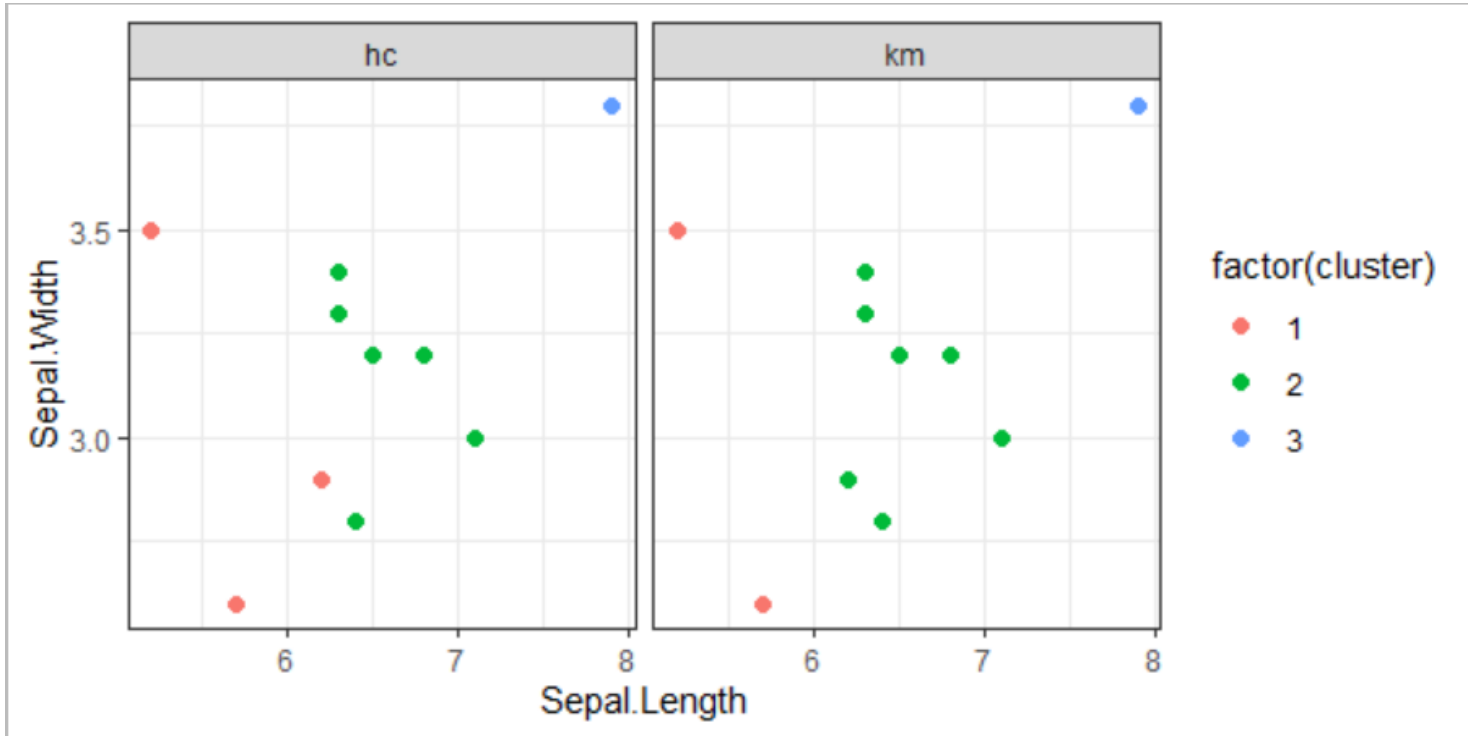
Comparison between hclust and kmeans

```
iris_sample <- iris_sample %>%
  mutate(hc = cutree(mod, 3)) %>%
  mutate(km = mod_km$cluster)
```

```
> iris_sample
   Sepal.Length Sepal.Width Petal.Length Petal.Width hc km
1           5.2         3.5          1.5         0.2  1  1
2           5.7         2.6          3.5         1.0  1  1
3           6.3         3.3          6.0         2.5  2  3
4           6.5         3.2          5.1         2.0  2  3
5           6.3         3.4          5.6         2.4  2  3
6           6.4         2.8          5.6         2.2  2  3
7           6.8         3.2          5.9         2.3  2  3
8           7.9         3.8          6.4         2.0  3  2
9           6.2         2.9          4.3         1.3  1  3
10          7.1         3.0          5.9         2.1  2  3
```

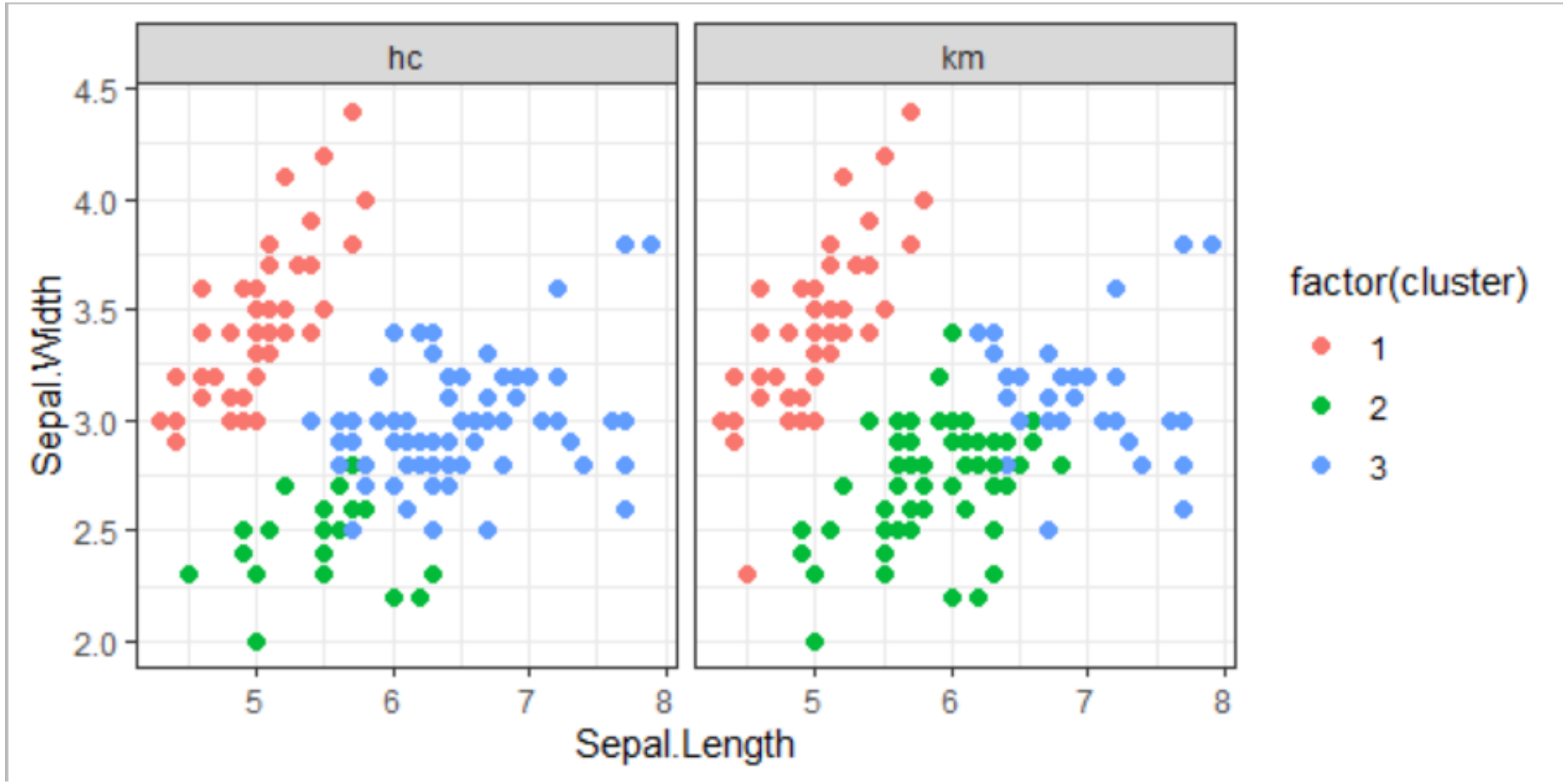
Comparison between hclust and kmeans

```
iris_sample %>%
    mutate(km = case_when(km==3 ~2, km==2 ~3, TRUE ~ 1)) %>%
    gather(model, cluster, hc:km) %>%
    ggplot(aes(x=Sepal.Length, y=Sepal.Width, col=factor(cluster))) +
    geom_point(size=2) +
    facet_wrap(~model, ncol=2)+
    theme_bw()
```



Comparison between hclust and kmeans

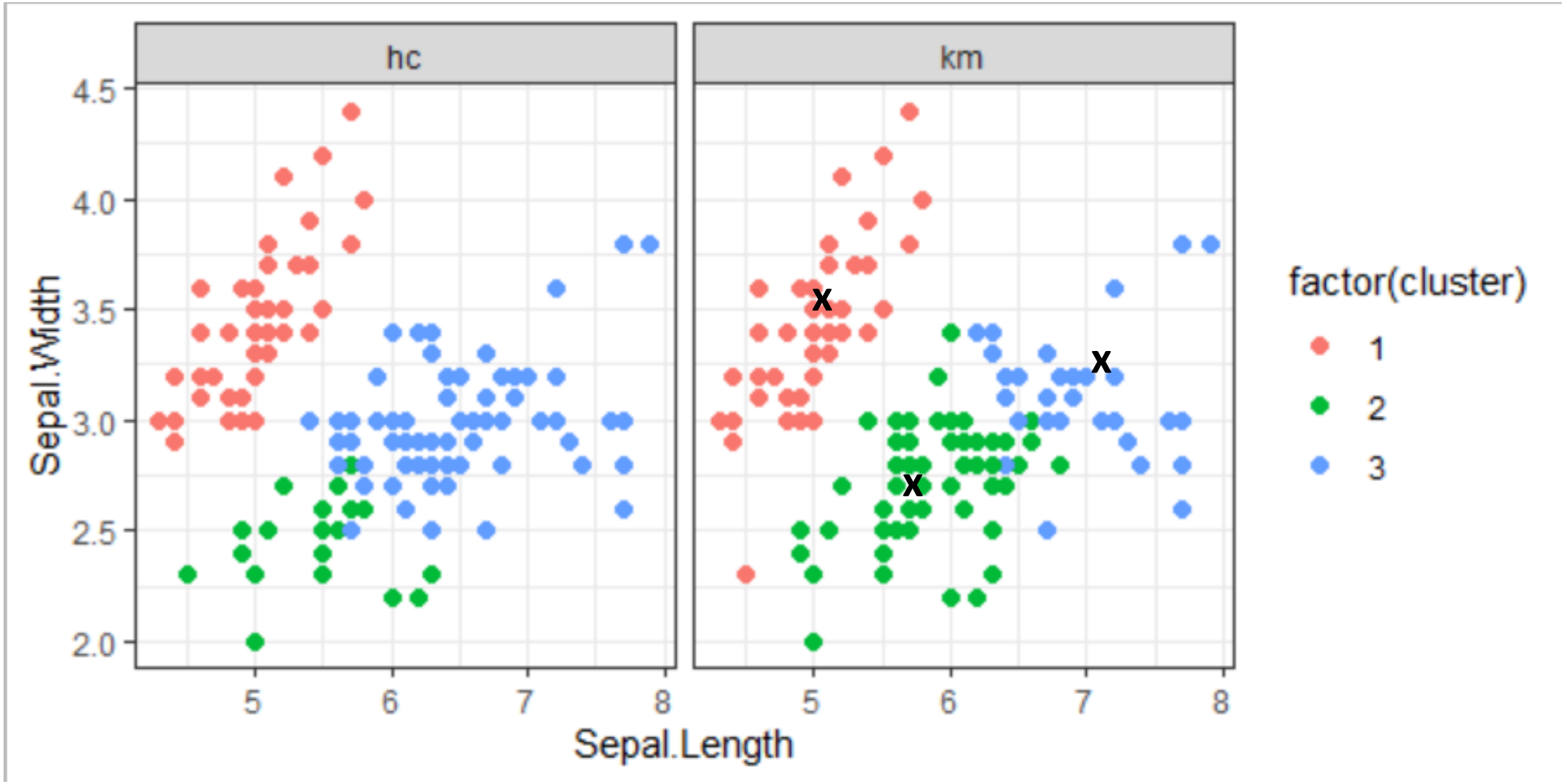
Full dataset



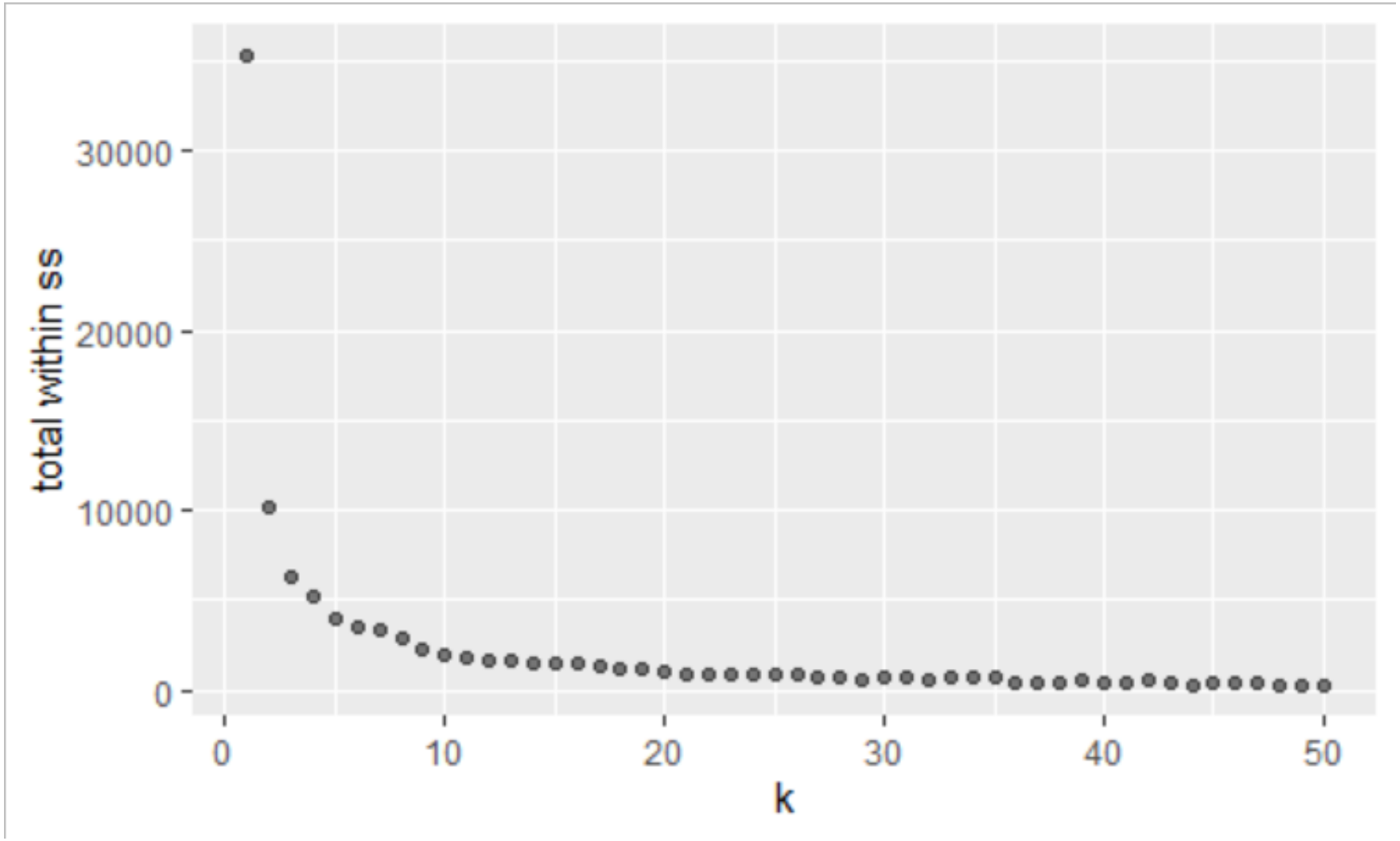
What is a good value for *k*?

- One can validate the clustering outcome visually; but visual inspection is not possible with more than 3 dimensions
- Another method is to track some total measure indicating how close observations are to the ‘center’ of their cluster
- One such measure is the total within sum of squares
- An elbow analysis shows the trade-off between reducing the total within ss and increasing the number of clusters

Total within SS



Elbow Analysis



Elbow Analysis (code)

```
tot_ss <- rep(0,50)
for (k in 1:50) {
  mod <- kmeans(dist(iris_scaled), centers = k)
  tot_ss[k] <- mod$tot.withinss
}
ggplot(tibble(x=1:50, y=tot_ss), aes(x,y)) +geom_point(alpha=0.5) +
  labs(x="k", y="total within ss")
```

Exercise

- Use k-means to cluster cars in the `mtcars` dataset; how does the result compare to h-clust? report your findings to the class