Clustering: k-means

2024-09-05

- **M1 MIDS/MFA**
- **Université Paris Cité**
- Année 2024-2025
- Course Homepage

- Moodle

! Objectives

Setup

```r
stopifnot(
  require(DT),
  require(skimr),
  require(GGally),
  require(patchwork),
  require(ggforce),
  require(glue),
  require(ggfortify),
  require(ggvoronoi),
  require(magrittr),
  require(broom),
  require(tidyverse)
)

tidymodels::tidymodels_prefer(quiet = TRUE)

old_theme <-theme_set(
  theme_minimal(base_size=9,
                base_family = "Helvetica")
)
```

```r
knitr::opts_chunk$set(
  message = FALSE,
  warning = FALSE,
  comment=NA,
  prompt=FALSE,
  cache=FALSE,
  echo=TRUE,
  results='asis'
)
```

```r
gc <- options(ggplot2.discrete.colour="viridis")
gc <- options(ggplot2.discrete.fill="viridis")
gc <- options(ggplot2.continuous.fill="viridis")
gc <- options(ggplot2.continuous.colour="viridis")
```

Foreword

This lab is dedicated to the k*-means* clustering method. In words, k*-means* takes as input a collection of points in $\mathbb{R}^d$ (a numerical dataset) and a positive integer k. It returns a collection of k points (the *centers*) from $\mathbb{R}^d$. The centers define a Voronoï *tesselation/partition/diagran* of $\mathbb{R}^d$. The Voronoï *cells* define a clustering of the original dataset.

Voronoi tesselation/partition/diagram

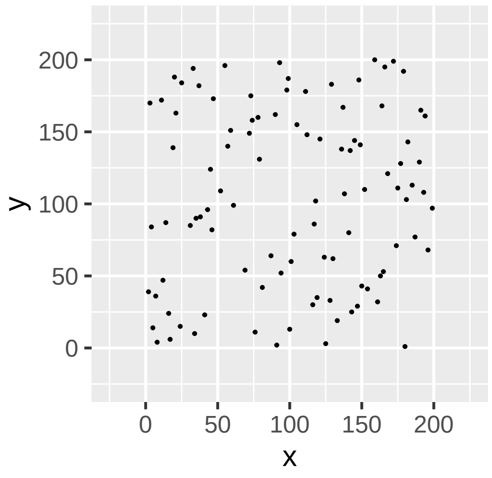
Wikipedia on Voronoï diagrams

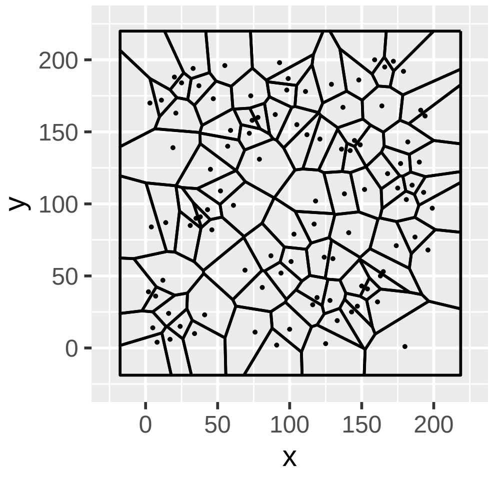
In the next chunk, we generate a Voronoï diagram on $\mathbb{R}^2$ with 100 cells defined from 100 random points drawn from the uniform distribution on a square. Function `stat_voronoi()` comes from ggvoronoi

```r
set.seed(45056)
x <- sample(1:200,100)
y <- sample(1:200,100)
points <- tibble(x,
                 y,
                 distance = sqrt((x-100)^2 + (y-100)^2))

p <- ggplot(points) +
    aes(x=x, y=y) +
    geom_point(size=.2) +
    coord_fixed() +
    xlim(c(-25, 225)) +
    ylim(c(-25, 225))

p + (p + stat_voronoi(geom="path")) +
  patchwork::plot_annotation(
    title="Voronoi tesselation",
    subtitle = "Left: 100 random points\nRight: Voronoï diagram")
```

Voronoi tesselation

Left: 100 random points
Right: Voronoï diagram





- Two adjacent Voronoï cells are separated by a (possibly semi-infinite) line segment
- Let the so-called *centers* be denoted by $c_1, \ldots, c_n$. They form the *codebook* $\mathcal{C}$.
- The Voronoï cell with *center* c_i is defined by

$$\left\{ x : x \in \mathbb{R}^d, \qquad \|x - c_i\|_2 = \min_{j \leq n} \|x - c_j\|_2 \right\}$$

- The center of a Voronoï cell is usually not its barycenter

***k*-means objective function**

The k-means algorithm aims at building a *codebook* $\mathcal{C}$ that minimizes

$$\mathcal{C} \mapsto \sum_{i=1}^{n} \min_{c \in \mathcal{C}} \|X_i - c\|_2^2$$

over all codebooks with given cardinality
If $c \in \mathcal{C}$ is the closest centroid to $X \in \mathbb{R}^p$,

$$\|c - X\|^2$$

is the *quantization/reconstruction error* suffered when using codebook $\mathcal{C}$ to approximate X
⚠ If there are no restrictions on the dimension of the input space, on the number of centroids, or on sample size, computing an optimal codebook is a NP -hard problem

`kmeans()` is a wrapper for a collection of Algorithms that look like the Lloyd algorithm
Initialize by sampling from the data `k-Mean++` try to take them as separated as possible.
Iterate the two phases until ?
👉 No guarantee to converge to a global optimum!
Proceeed by trial and error.
Repeat and keep the best result.

Iris data

Run `kmeans()` on the projection of the Iris dataset on the Sepal plane. We look for a partition into three cells.

```
data(iris)

kms <- iris %>%
  select(starts_with("Petal")) %>%
  kmeans(3)
```

The result is an object of class `kmeans`. The class is equiped with `broom` methods.

Summarizing a clustering

Question

Check the structure of objects of class `kmeans` and use `broom::tidy()` to get a summary. Compare with `summary()`

```
df_centers <- select(iris, starts_with("Petal")) |>
  kmeans(centers = 3) |>
  broom::tidy()

df_centers |>
  mutate(across(where(is.numeric), ~ signif(.x, digits=3) ))
```

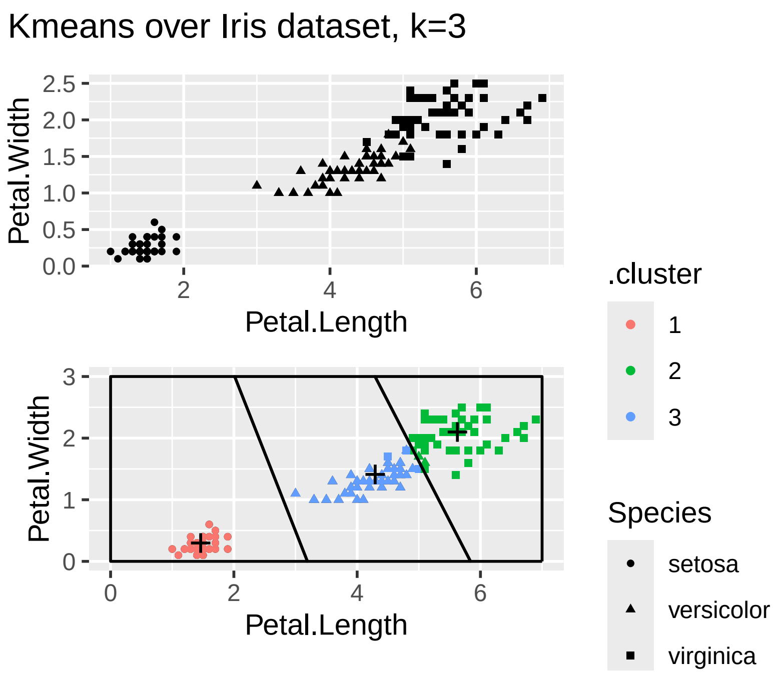
```
# A tibble: 3 x 5
  Petal.Length Petal.Width  size withinss cluster
         <dbl>       <dbl> <dbl>    <dbl> <fct>
1         4.29       1.36     54    14.2  1
2         1.46       0.246    50     2.02 2
3         5.63       2.05     46    15.2  3
```

Visualizing a clustering

Question

Use `broom::augment()` and `broom::tidy()` to prepare two dataframes that will allow you to overlay a scatterplot of the dataset and a Voronoï diagram defined by the centers outpu by `kmeans()`.
Compare the result with `plot()`

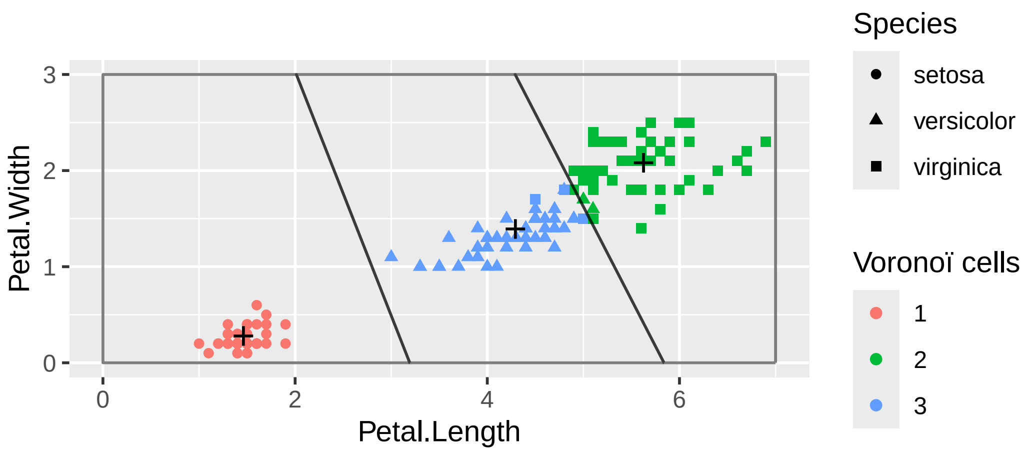
```
q <- kms %>%
  augment(iris) %>%
  ggplot() +
  aes(x=Petal.Length,
      y=Petal.Width
      ) +
  geom_point(aes(shape=Species), size=1, show.legend = F) +
  coord_fixed()

qq <-  (q + geom_point(aes(shape=Species,
                           colour=.cluster),
                       size=1))+
  stat_voronoi(data = df_centers,    #<<
               geom="path",
               outline=data.frame(x=c(0, 7, 7, 0),
                                  y=c(0, 0, 3, 3))
               ) +
  geom_point(data = df_centers,    #<<
             colour = "black",
             shape="+",
             size=5)

q / qq +
  plot_annotation(title = "Kmeans over Iris dataset, k=3")
```



```
geom_sugar <- list(
    stat_voronoi(data = df_centers,
                 geom="path",
                 alpha=.5,
                 outline = tribble(~x, ~y,
                                   0., 0.,
                                   7., 0.,
                                   7., 3,
                                   0., 3)
```

```r
                 ),
    geom_point(data = df_centers,
               colour = "black",
               shape="+",
               size=5),
    coord_fixed(),
    labs(col="Voronoï cells")
)
```

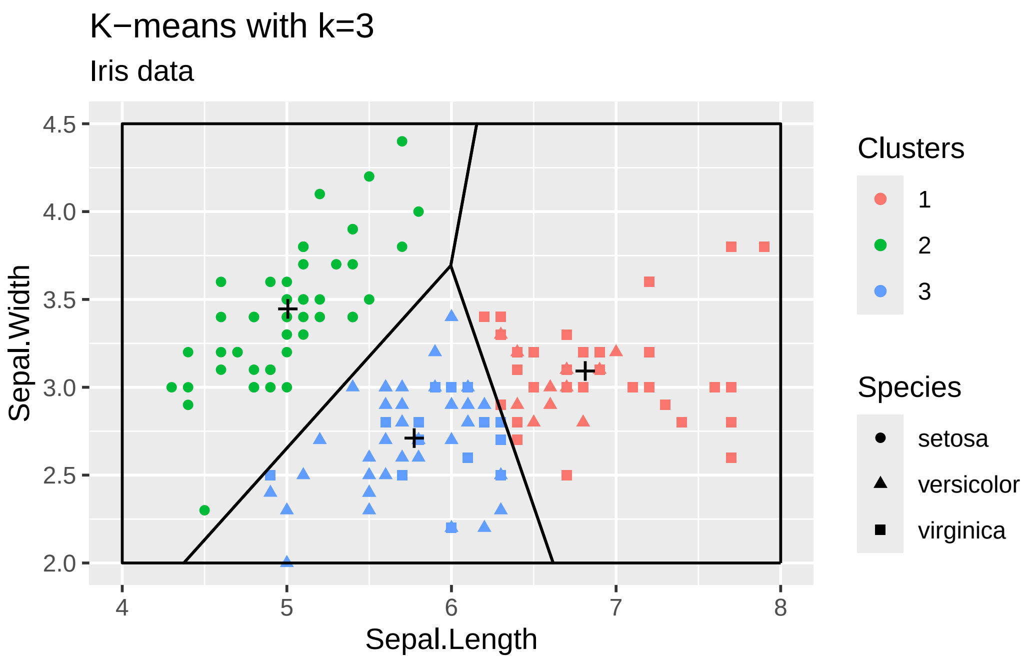
```r
broom::augment(kms, iris) %>%
  ggplot(aes(x=Petal.Length, y=Petal.Width)) +
  geom_point(aes(shape=Species, color=.cluster)) +
  geom_sugar
```



Question

Redo the same operations but choose the `Sepal.xxx` dimension.
Design a function to avoid repetitive coding.

```r
kms <- kmeans(select(iris, Sepal.Length, Sepal.Width), 3)

broom::augment(kms, iris) %>%
 ggplot() +
 geom_point(aes(x=Sepal.Length, y=Sepal.Width,
             shape=Species, col=.cluster)) +
 geom_point(data=data.frame(kms$centers),         #<<
             aes(x=Sepal.Length, y=Sepal.Width),
             shape='+',
             size=5) +
 stat_voronoi(data = as.data.frame(kms$centers),  #<<
               aes(x=Sepal.Length,y=Sepal.Width),
               geom="path",
               outline=data.frame(x=c(4, 8, 8, 4),
                                  y=c(2, 2, 4.5, 4.5))) -> p

p +
  ggtitle("K-means with k=3", "Iris data") +
  labs(col="Clusters")
```



Playing with k

The number of cells/clusters may not be given a priori.

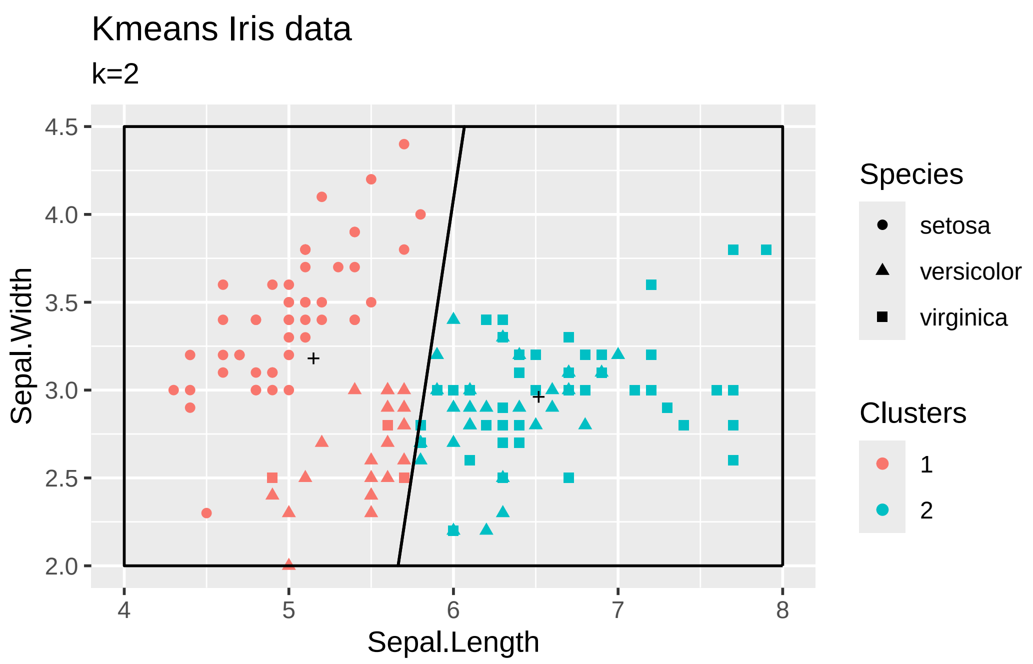
Question

Perform kmeans clustering with $k = 2$. Use `glance`, `tidy`, `augment` to discuss the result.

```r
df <- select(iris,
             starts_with("Sepal"))

kms <- kmeans(df, 2)

df_centers2 <- tidy(kms)

df_centers2
```

```
# A tibble: 2 x 5
  Sepal.Length Sepal.Width  size withinss cluster
         <dbl>       <dbl> <int>    <dbl> <fct>
1         5.15        3.17    74     30.0 1
2         6.52        2.95    76     28.4 2
```

```r
augment(kms, iris) %>%
 ggplot() +
 aes(x=Sepal.Length, y=Sepal.Width) +
 geom_point(aes(shape=Species, col=.cluster)) +
 geom_point(data=df_centers2,    #<<
            shape='+',
            size=3) +
 stat_voronoi(data = df_centers2,    #<<
               geom="path",
               outline=data.frame(x=c(4, 8, 8, 4),
                                  y=c(2, 2, 4.5, 4.5)))  +
  ggtitle(label="Kmeans Iris data",
          subtitle="k=2") +
  labs(col="Clusters")
```



We can compare the spread between inner and outer sum of squares for clusterings with $k \in 2, 3$.

```
bind_rows(glance(kms),
          glance(kmeans(df,
              centers=3,
              nstart = 32L))) |>
  mutate(k=c(2, 3))
```

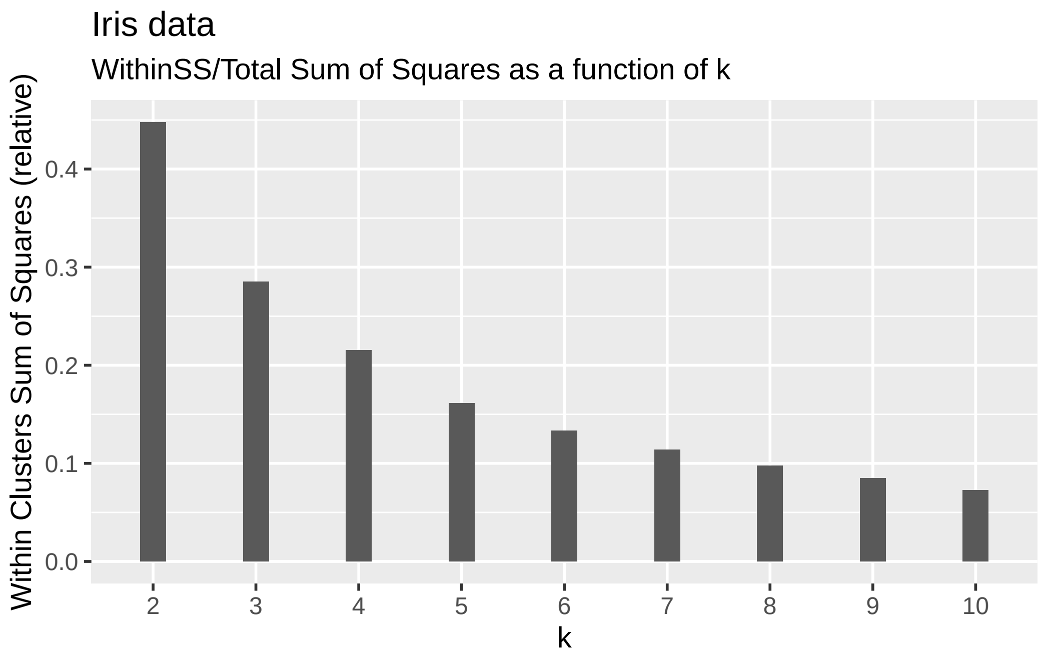
```
# A tibble: 2 x 5
  totss tot.withinss betweenss  iter     k
  <dbl>        <dbl>     <dbl> <int> <dbl>
1  130.         58.4      72.0     1     2
2  130.         37.1      93.4     3     3
```

Question

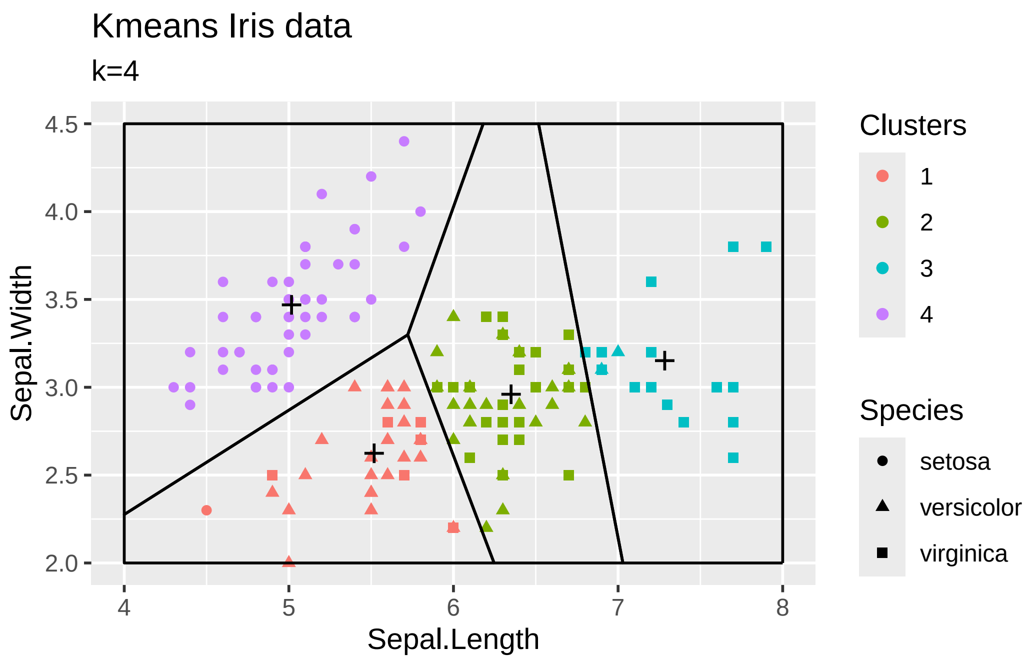
Perform k-means for $k = 2, ...10$, plot within sum of squares as function of k. Comment.

```
tmp <-map_dfr(2:10, ~ glance(kmeans(df,
                                    centers=.,
                                    nstart = 32L))) %>%
  rowid_to_column(var="k") %>%
  mutate(k=k+1, across(where(is.numeric), ~ signif(.x, 3)))
```

```
tmp %>%
  ggplot(aes(x=forcats::as_factor(k), y=tot.withinss/totss)) +
  geom_col(width=.25) +
  ggtitle("Iris data", "WithinSS/Total Sum of Squares as a function of k") +
  xlab("k") +
  ylab("Within Clusters Sum of Squares (relative)") +
  scale_x_discrete(breaks=as.character(2:10), labels=as.character(2:10))
```



```
kms <- kmeans(df, 4)
iris4 <- broom::augment(kms, iris)

ggplot(iris4) +
geom_point(aes(x=Sepal.Length, y=Sepal.Width,
           shape=Species, col=.cluster)) +
geom_point(data=data.frame(kms$centers),          #<<
           aes(x=Sepal.Length, y=Sepal.Width),
           shape='+',
           size=5) +
stat_voronoi(data = as.data.frame(kms$centers),   #<<
              aes(x=Sepal.Length,y=Sepal.Width),
              geom="path",
              outline=data.frame(x=c(4, 8, 8, 4), y=c(2, 2, 4.5, 4.5)))  +
 ggtitle(label="Kmeans Iris data",
         subtitle="k=4") +
 labs(col="Clusters")
```



```r
broom::tidy(kmeans(df, 4)) %>%
  knitr::kable(format = "markdown", digits = 2)
```

Sepal.Length	Sepal.Width	size	withinss	cluster
5.02	3.45	49	11.57	1
5.52	2.61	33	5.97	2
7.10	3.11	27	6.02	3
6.27	2.91	41	4.85	4

Lloyd's iterations

Initialize Choose k centroids

Iterations: Two phases

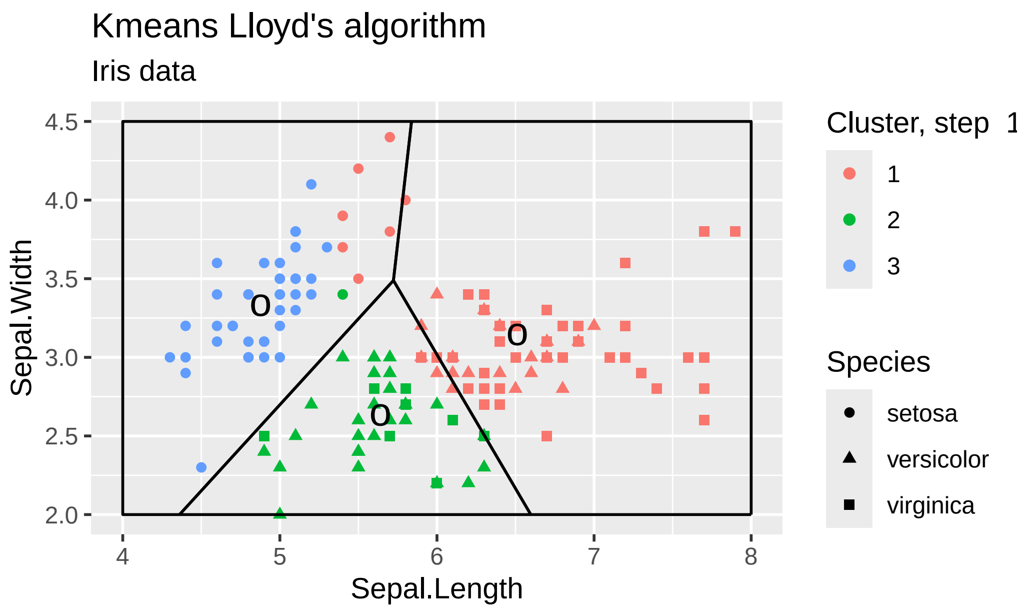
Movement Assign each sample point to the closest centroid Assign each sample point to a class in the Voronoi partition defined by the centroids

Update For each class in the current Voronoi partition, update teh centroid so as to minimize the Within Cluster Sum of Squared distances.

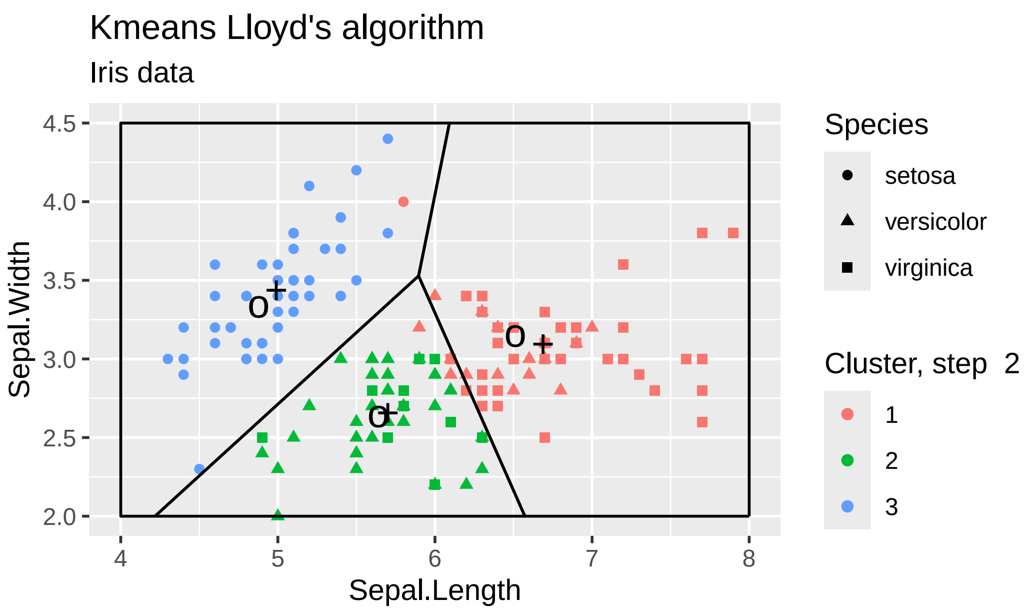
```r
km <- list(centers=df[1:3, ]) # stupid initialization

sequence <- list()

for (i in 1:20) {
  km <- kmeans(df,
               km$centers,
               algorithm = "Lloyd",
               iter.max = 1)
  sequence[[length(sequence)+1]] <- force(km)
}
```

```r
add_voronoi <- function(p, kmscenters, marker){
  p +
    geom_point(data=data.frame(kmscenters),          #<<
               mapping=aes(x=Sepal.Length, y=Sepal.Width),
               shape=marker,
               col="black",
               size=5) +
    stat_voronoi(data = as.data.frame(kmscenters),  #<<
                 aes(x=Sepal.Length,y=Sepal.Width),
                 geom="path",
                 outline=data.frame(x=c(4, 8, 8, 4),
                                    y=c(2, 2, 4.5, 4.5)))
}
```

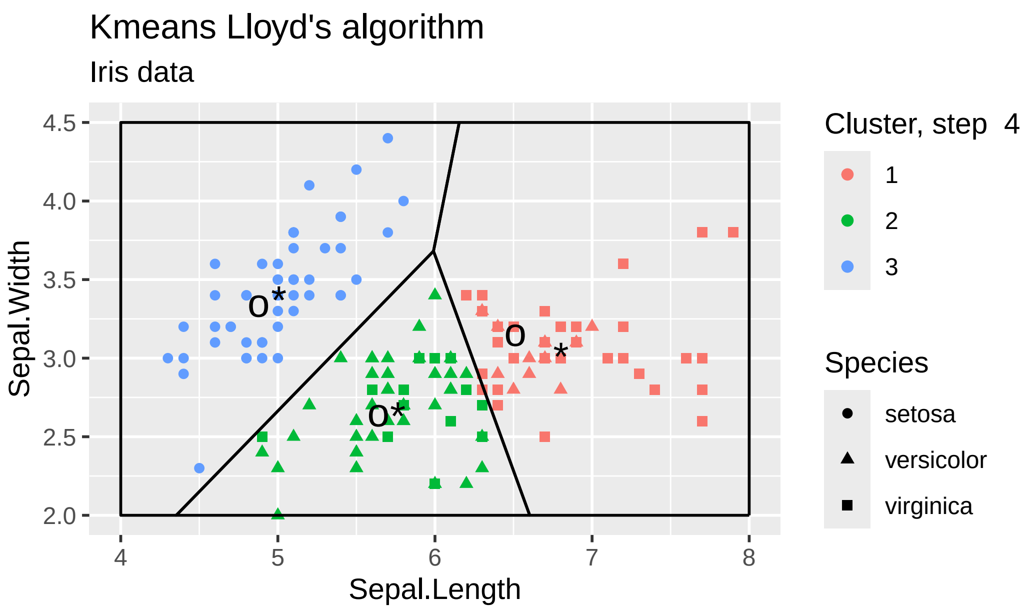
```
i <- 2

p <- broom::augment(sequence[[i]], iris) %>%
  ggplot() +
  coord_fixed(ratio=1) +
  geom_point(aes(x=Sepal.Length, y=Sepal.Width, shape=Species, col=.cluster)) +
  ggtitle("Kmeans Lloyd's algorithm", "Iris data")

p %>%
  add_voronoi(sequence[[i]]$centers, marker="o") +   #<<
  labs(colour=paste("Cluster, step ", i- 1))
```



```
i <- 3

(p %+%
  broom::augment(sequence[[i]], iris)) %>%
  add_voronoi(sequence[[i]]$centers, marker='+') +   #<<
  geom_point(data=data.frame(sequence[[2]]$centers),   #<<
             mapping=aes(x=Sepal.Length, y=Sepal.Width),
             shape="o", col="black", size=5) +
  labs(colour=paste("Cluster, step ", i- 1))
```



```r
i <- 5

(p %+%
  broom::augment(sequence[[i]], iris)) %>%
  add_voronoi(sequence[[i]]$centers, marker='*') +   #<<
  geom_point(data=data.frame(sequence[[2]]$centers),   #<<
             mapping=aes(x=Sepal.Length, y=Sepal.Width),
             shape="o", col="black",size=5) +
  labs(colour=paste("Cluster, step ", i- 1))
```



References

Vignette `k_means` from `tidyclust`