ML: Clustering

Jin Seo Jo

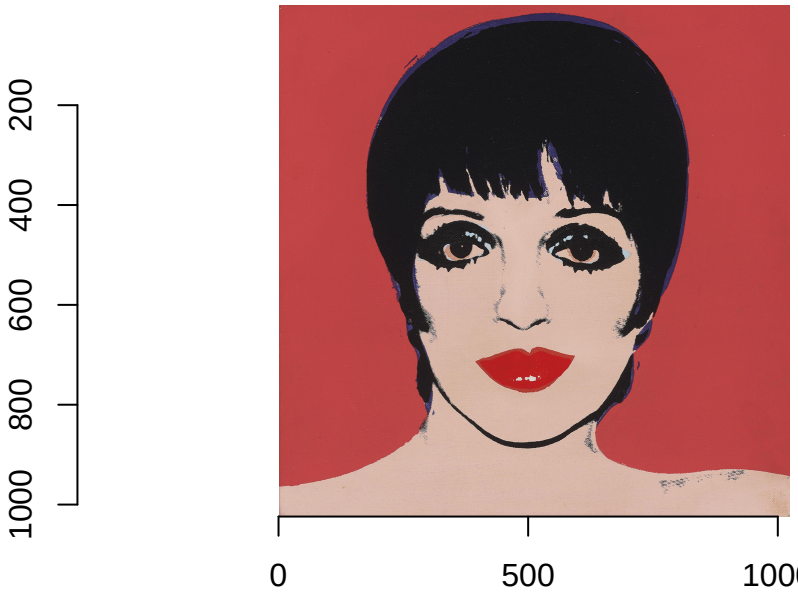
24/09/2020

Task: Data Analysis

For this task we will use Andy Warhol's iconic portrait: Liza Minnelli

```
library(imager)

im <- imager::load.image("liza_minnelli_andy_warhol_collection.jpg")
plot(im)
```



This picture is stored as a 'cimg' object, which is bascially a 4-dimensional array. The first index is the horizontal pixel, the second is the vertical, the third is the opacity, and the fourth is the colour (R,G,B)

And we can turn this into something useful for clustering by using the 'as.data.frame' method with the option wide = "c" (This option only works for a 'cimg' object.) We then rename the three colours to "R", "G", and "B" using 'rename' function from 'dplyr'.

```
library(dplyr)

tidy_data <- as.data.frame(im, wide = "c") %>%
  rename(R = c.1, G = c.2, B = c.3)

head(tidy_data, 5)
```

```
##   x y         R         G         B
## 1 1 1 0.7960784 0.5647059 0.5411765
## 2 2 1 0.5882353 0.3019608 0.2745098
## 3 3 1 0.6470588 0.2784314 0.2470588
## 4 4 1 0.6941176 0.2588235 0.2235294
## 5 5 1 0.7098039 0.2509804 0.2156863
```

Because 'class' has type 'cimg' (type 'class(im)' to confirm), when we call 'as.data.frame' R finds the version of 'as.data.frame' that works on on that type of object. In this case it finds the internal function 'imager:::as.data.frame.cimg()', which has the 'wide = "c" ' argument. (The three :s means that the function is internal to the package.)

We now have the data in the foremat required to the clustering. Explore various k-means clustering using the template laid out in the Learning K-Means with tidy data principles vignette.

First things first, let's make the scree plot.

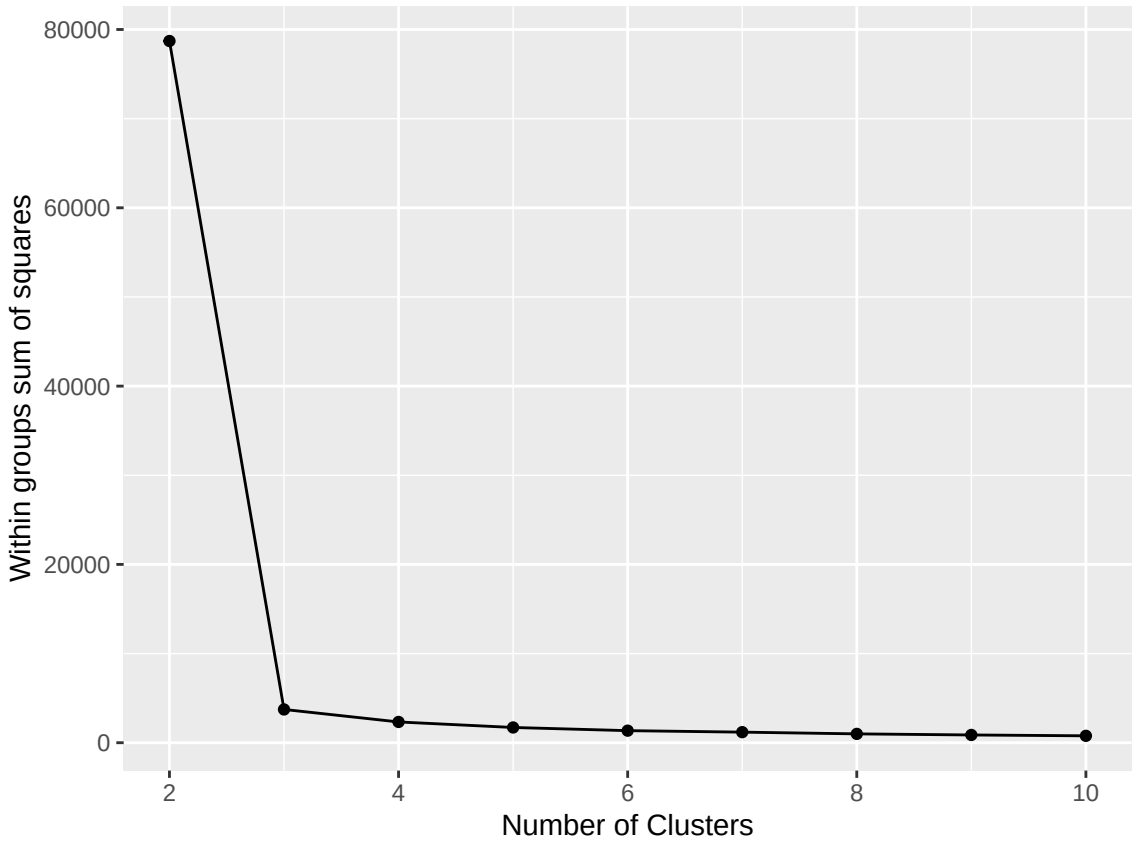
```
library(purrr)
library(tidymodels)

dat <- select(tidy_data, c(-x, -y))

kclusts <- tibble(k = c(2:10)) %>%
  mutate(kclust = map(k, ~kmeans(x = dat, centers = .x, nstart = 4)),
         glanced = map(kclust, glance))

clusterings <- kclusts %>%
  unnest(cols = c(glanced))

ggplot(clusterings, aes(k, tot.withinss)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of Clusters", y = "Within groups sum of squares")
```



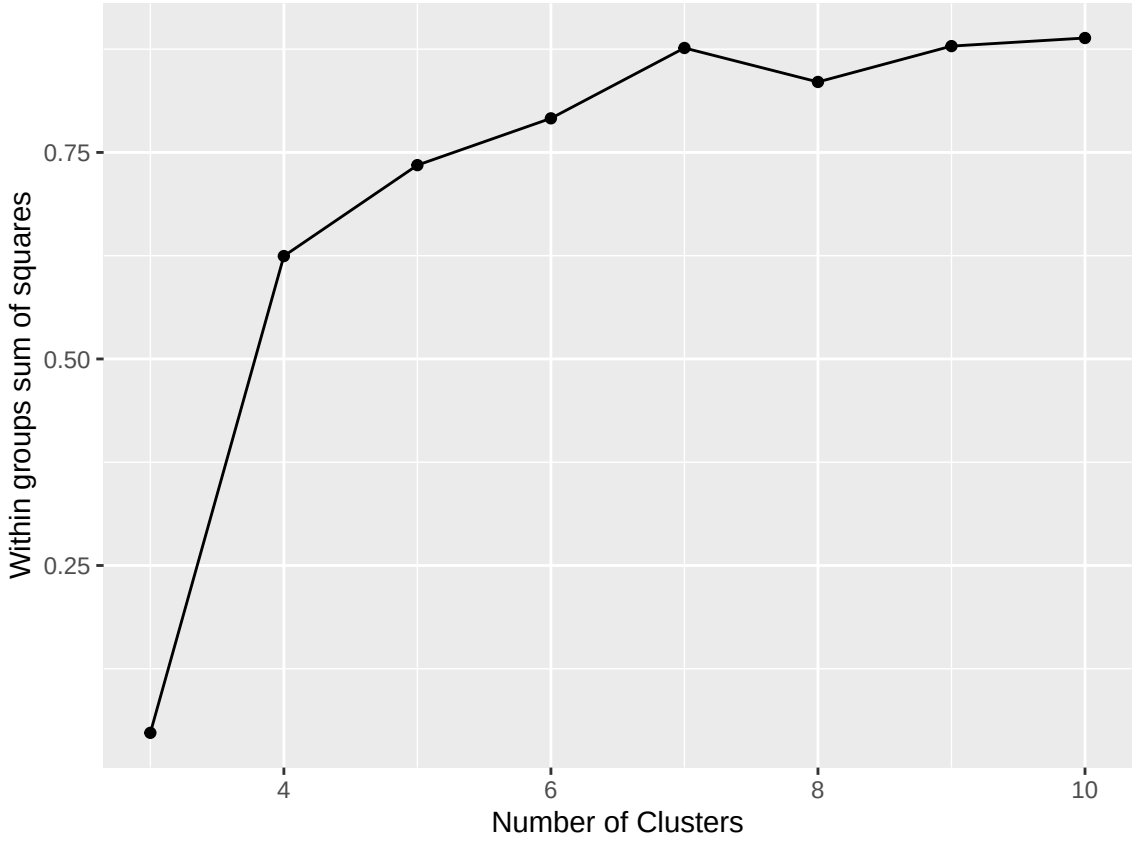
Maybe 6 is the right number of clusters? It's hard to tell. Hence, we will try the ratio version.

```
nclust = length(clusterings$k)
ratio = rep(NA, nclust-1)

for (kk in 2:nclust) {
  ratio[kk-1] = clusterings$tot.withinss[kk]/clusterings$tot.withinss[kk-1]
}

plot_data <- data.frame(k = clusterings$k[2:nclust], ratio)

ggplot(plot_data, aes(x = k, y = ratio)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of Clusters", y = "Within groups sum of squares")
```



From this the number of clusters seems to be six! So let's use that going forward.

First, let's re-do the clustering and save the centres.

```
k <- 7
kclust <- kmeans(select(tidy_data, -x, -y), centers = k, nstart = 20)
centres <- tidy(kclust)
```

We can also add a column to the tidied centres to add the colour in a way that we can use for plots. The 'rgb' function will do this and display the colour as a hex string.

```
centres <- centres %>%
  mutate(col = rgb(R, G, B))

centres
```

```
## # A tibble: 7 x 7
##       R     G     B   size withinss cluster col
##   <dbl> <dbl> <dbl>  <int>    <dbl> <fct>   <chr>
## 1 0.861 0.733 0.680 186232    158.  1       #DCBBAD
## 2 0.738 0.266 0.270 491573    249.  2       #BC4445
## 3 0.711 0.130 0.135   9767     24.9 3       #B52122
## 4 0.107 0.102 0.110 248942    196.  4       #1B1A1C
## 5 0.827 0.698 0.641  84197    134.  5       #D3B2A3
## 6 0.251 0.203 0.315  20310    234.  6       #403450
## 7 0.567 0.500 0.492   7555    142.  7       #90807E
```

It's probably worth seeing what the colours are. In this case, we will use 'show_col' from 'scales'.

```
library(scales)
show_col(centres$col)
```

Visually, we can see that two of these colours are skin tones. Let's see what happens if we choose 6 colours.

```
kclust6 <- kmeans(select(tidy_data, -x, -y), centers = 6, nstart = 20)

centres6 <- tidy(kclust6)

centres6 <- centres6 %>%
  mutate(col = rgb(R, G, B))

show_col(centres6$col)
```

It's slightly different but probably better. This is one of those cases where the scree plot can be misleading and using visualizations can help.

So now we have six clusters we need to put the do the cluster centre replacement. To do this, we first need to augment the initial data with the clusters. We can do this with 'broom::augment' function ('broom' is a package loaded by 'tidymodels'). The 'rename' command just makes the naming a little nicer.

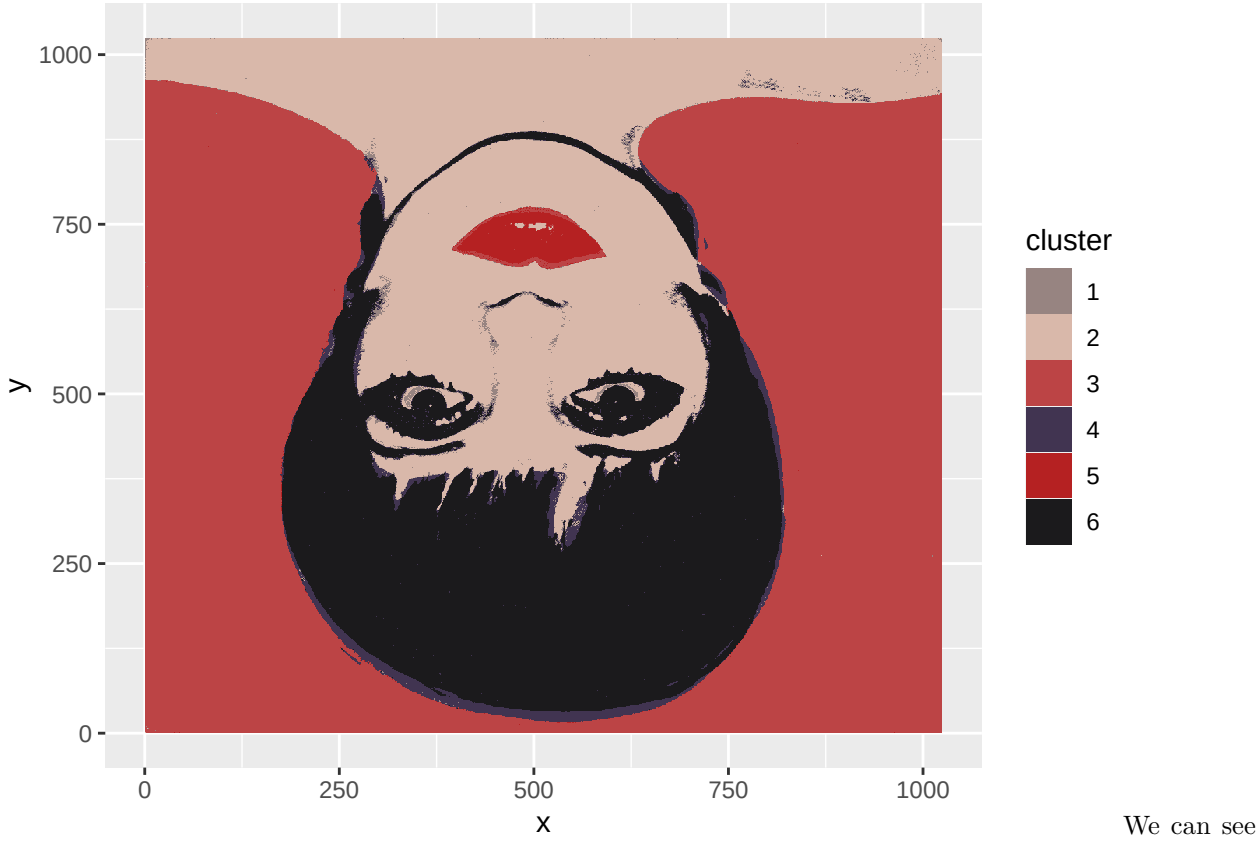
```
tidy_data <- augment(kclust6, tidy_data) %>%
  rename(cluster = .cluster)

glimpse(tidy_data)
```

```
## Rows: 1,048,576
## Columns: 6
## $ x       <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
## $ y       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
## $ R       <dbl> 0.7960784, 0.5882353, 0.6470588, 0.6941176, 0.7098039, 0.71...
## $ G       <dbl> 0.5647059, 0.3019608, 0.2784314, 0.2588235, 0.2509804, 0.25...
## $ B       <dbl> 0.5411765, 0.2745098, 0.2470588, 0.2235294, 0.2156863, 0.23...
## $ cluster <fct> 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,...
```

We can now plot the clustered picture!

```
ggplot(tidy_data, aes(x = x, y = y, fill = cluster)) +
  geom_tile() +
  scale_discrete_manual(aesthetics = "fill", values = centres6$col)
```



We can see that Liza is upside down.

```
ggplot(tidy_data, aes(x = x, y = y, fill = cluster)) +
  geom_tile() +
  scale_discrete_manual(aesthetics = "fill", values = centres6$col) +
  scale_y_reverse() +
  theme_void()
```


cluster
1
2
3
4
5
6