Creating Cross-Stitch Patterns of Images Using K-Means Clustering

STA314H1 - Fall 2020

Ziyue Yang

November 3rd, 2020

Introduction

This article demonstrates how to make a cross-stitch pattern of images, based on the k-means clustering algorithm.

Setup

It is required for you to use the following libraries. Note that if you don't have any of these, undo the commenting before you run:

```r
# Uncomment the following if need to install libraries
# install.packages("imager")
# install.packages("tidyverse")
# install.packages("tidymodels")
# install.packages("sp")
# install.packages("scales")
# install.packages("cowplot")
# devtools::install_github("sharlagelfand/dmc")

library(imager)
library(tidyverse)
library(tidymodels)
library(sp)
library(scales)
library(cowplot)
library(dmc)
```

In addition, we load the script `functions.R`, which will provide us the functions we are going to demonstrate:

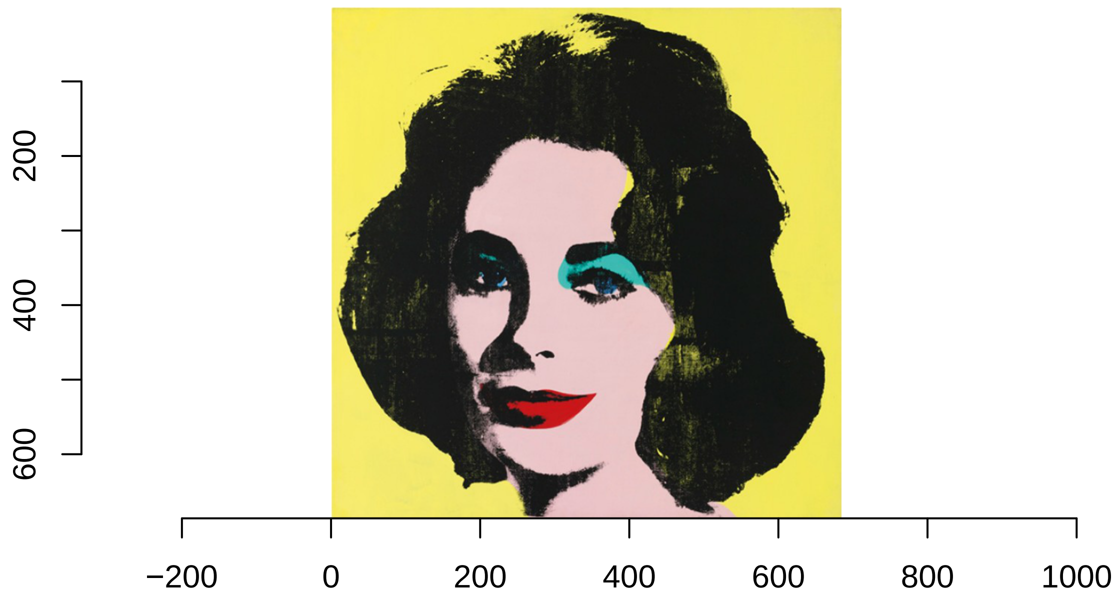
```r
source('functions.R')
```

The script `functions.R` contains the following functions:

a. `process_image()`
b. `scree_plot()`
c. `colour_strips()`
d. `make_pattern()`

For demonstration, let's make a cross-stitch pattern of a Warhol screenprint of Elizabeth Taylor:

```r
plot(imager::load.image('warhol.jpeg'))
```



Workflow

Our work flow can be briefly described as:

$$\text{Get Cluster Data for Multiple } K's \rightarrow \text{Choose Ideal } K \rightarrow \text{Make Cross-Stitch Pattern with Chosen } K.$$

Generating Cross-Stitch Patterns

Before we begin, let's go over the functions we need to make a cross-stitch pattern.

`process_image()`

Function `process_image()` allows us to retrieve the cluster information based on a list of k's.

This function takes two inputs:

- `image_file_name`: the path of the image you want to cluster;
- `k_list`: a list of numbers of centers in the clustering. For instance, if we want to cluster the image with 2, 3, 7 cluster centers respectively, then `k_list` takes `c(2, 3, 7)` as the input.

Calling the function as shown below allows us to retrieve the clustering information of the Warhol picture, from 1 to 10 cluster centers. We want to choose the best number of cluster centers later:

```
cluster_info <- process_image('warhol.jpeg', c(1:10))
```

The output of `process_image()` stores information in variable `cluster_info`, for every k in `k_list`.

For each k, the output contains

- `kclust`: the output of calling `kmeans(x = select(image_dat, c(-x, -y)), centers = .x, nstart = 5)`;
- `tidied`: the tidied data of `kclust`, i.e. `tidy(kclust)`;
- `glanced`: the `glance` of `kclust`.

Let's check the output of each of the above, for `k = 2`:

```
cluster_info$kclust[[2]]
```

```
## # A tibble: 469,224 x 6
##        x     y     R     G     B .cluster
##    <int> <int> <dbl> <dbl> <dbl> <fct>
##  1     1     1 1     0.984 0.733 1
##  2     2     1 0.969 0.922 0.624 1
##  3     3     1 0.922 0.886 0.514 1
##  4     4     1 0.922 0.898 0.467 1
##  5     5     1 0.929 0.898 0.463 1
##  6     6     1 0.937 0.906 0.486 1
##  7     7     1 0.953 0.906 0.506 1
##  8     8     1 0.949 0.902 0.502 1
##  9     9     1 0.941 0.898 0.482 1
## 10    10     1 0.949 0.910 0.475 1
## # ... with 469,214 more rows
```

```
cluster_info$tidied[[2]]
```

```
## # A tibble: 2 x 8
##       R     G     B   size withinss cluster RGB     DMC
##   <dbl> <dbl> <dbl>  <int>    <dbl> <fct>   <chr>   <chr>
## 1 0.926 0.854 0.517 236676   12092. 1       #ECDA84 #F3CE75
## 2 0.127 0.125 0.105 232548    3736. 2       #20201B #1E1108
```

```
cluster_info$glanced[[2]]
```

```
## # A tibble: 1 x 4
##     totss tot.withinss betweenss  iter
##     <dbl>        <dbl>     <dbl> <int>
## 1 172812.       15827.   156985.     1
```

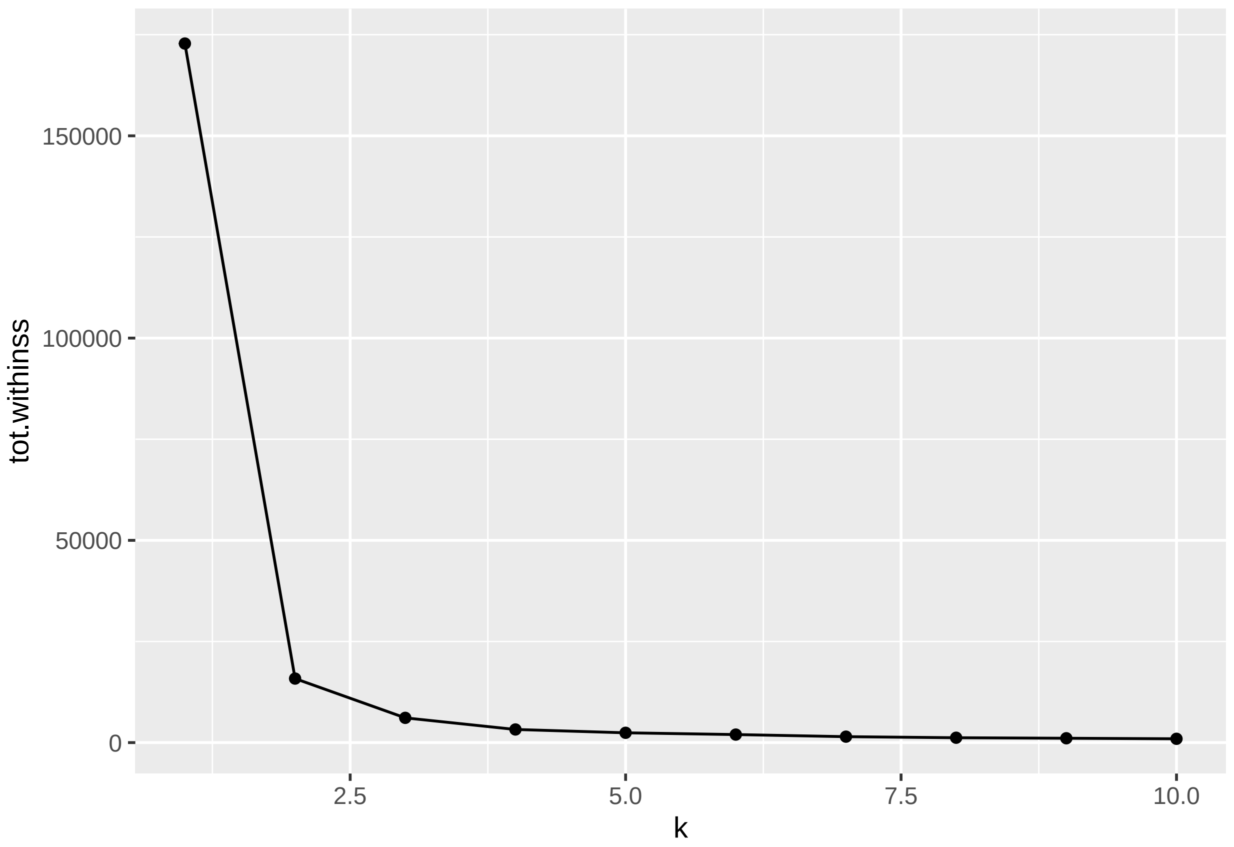
Note: From now on, `cluster_info` will store all the clustering information of the Warhol image, which will be used as the input for the following functions.

scree_plot

This function takes the input `cluster_info`, as produces a scree plot based on `glanced`:

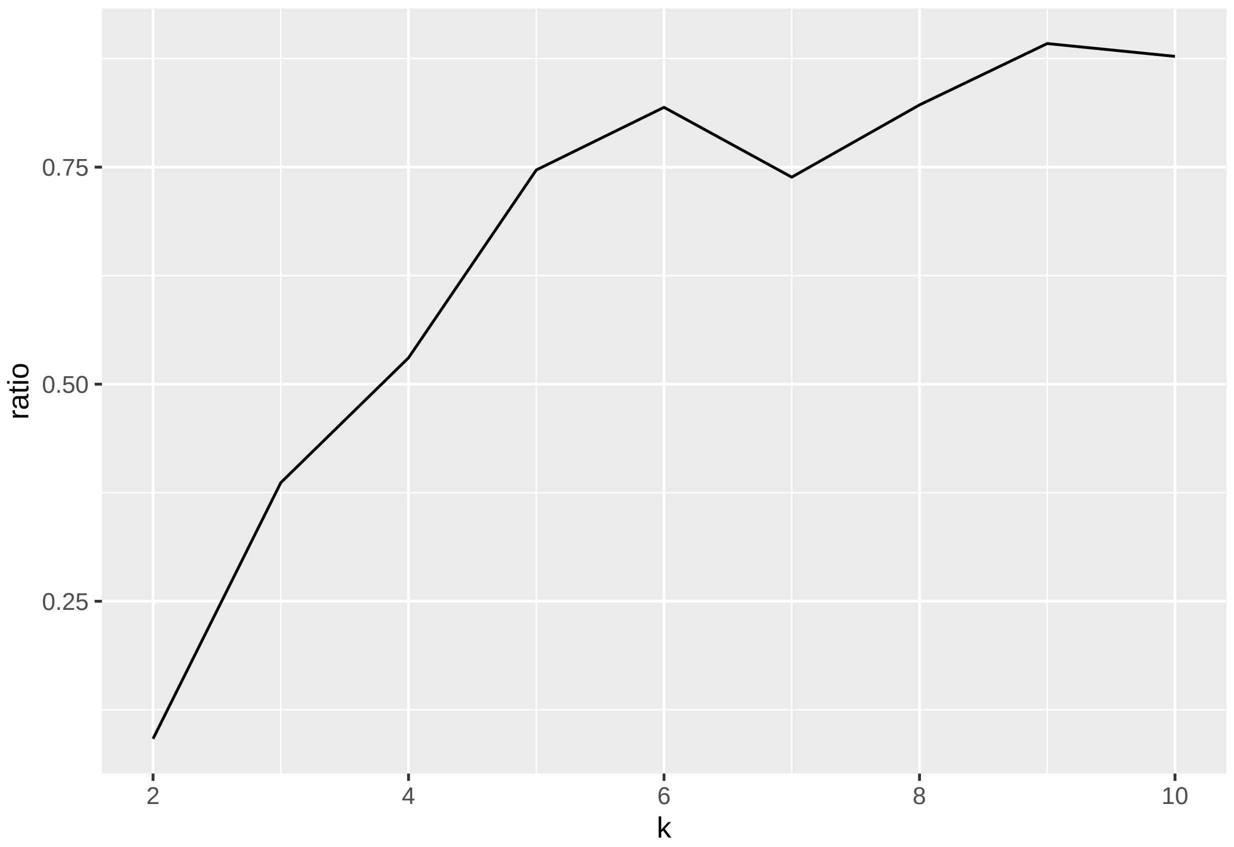
and returns a scree plot, with respect to the maximum numbers of cluster centers in `k_list` we inputted for `process_image()` (10 in this case):

```
scree_plot(cluster_info)
```



Note that it's hard to tell what number of centers to choose based on scree plot. Let's try the ratio version:

```r
clusterings <- cluster_info %>% unnest(cols = c(glanced))
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()
```

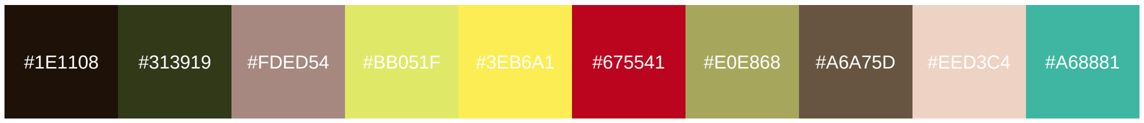


From which we can tell that the number of clusters seems to be 5.

`colour_strips()`

This function takes the input `cluster_info`, and produces the DMC colour strips that are closest to the RGB colours of cluster centers, for each k in `k_list`.

```
colour_strips(cluster_info)
```



Looks like 5 is a good option. Let's use 5 centers to make our cross-stitch pattern!

`make_pattern()`

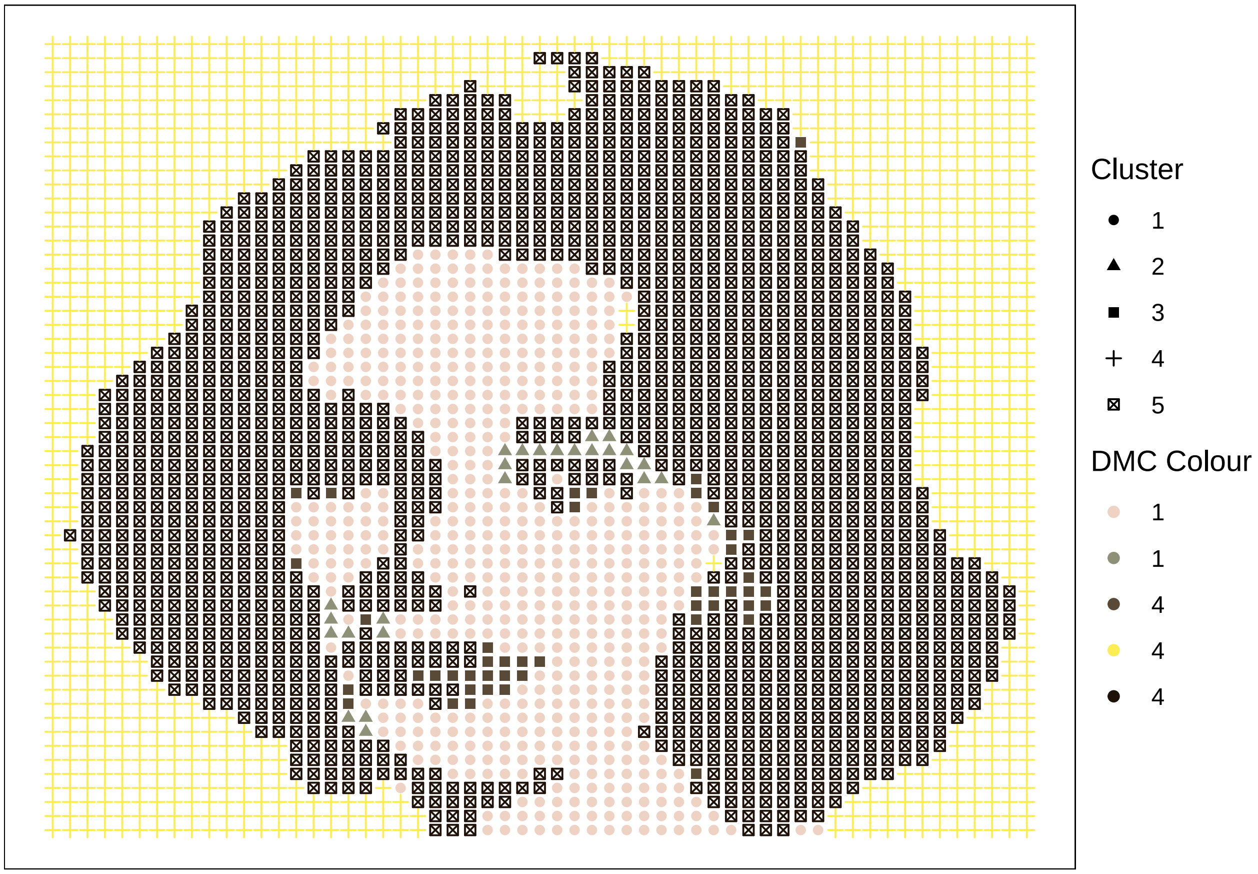
Finally, this function allows us to plot the cross-stitch of our image. This function takes several inputs:

- `cluster_info`: The output of `process_image`.
- `k`: The number of cluster centers.
- `x_size`: The total number of possible stitches in the horizontal direction.
- `black_white`: The logical value indicating whether the cross-sitch will be plotted in black and white. Default is `FALSE`, such that we have a cross-stitch where the iˆth cluster has the DMC colour that is closest to the RGB colour of the i$^{\text{th}}$ cluster center.

- `background_colour`: The colour of the background. Default is `NULL`, such that we have a transparent background.

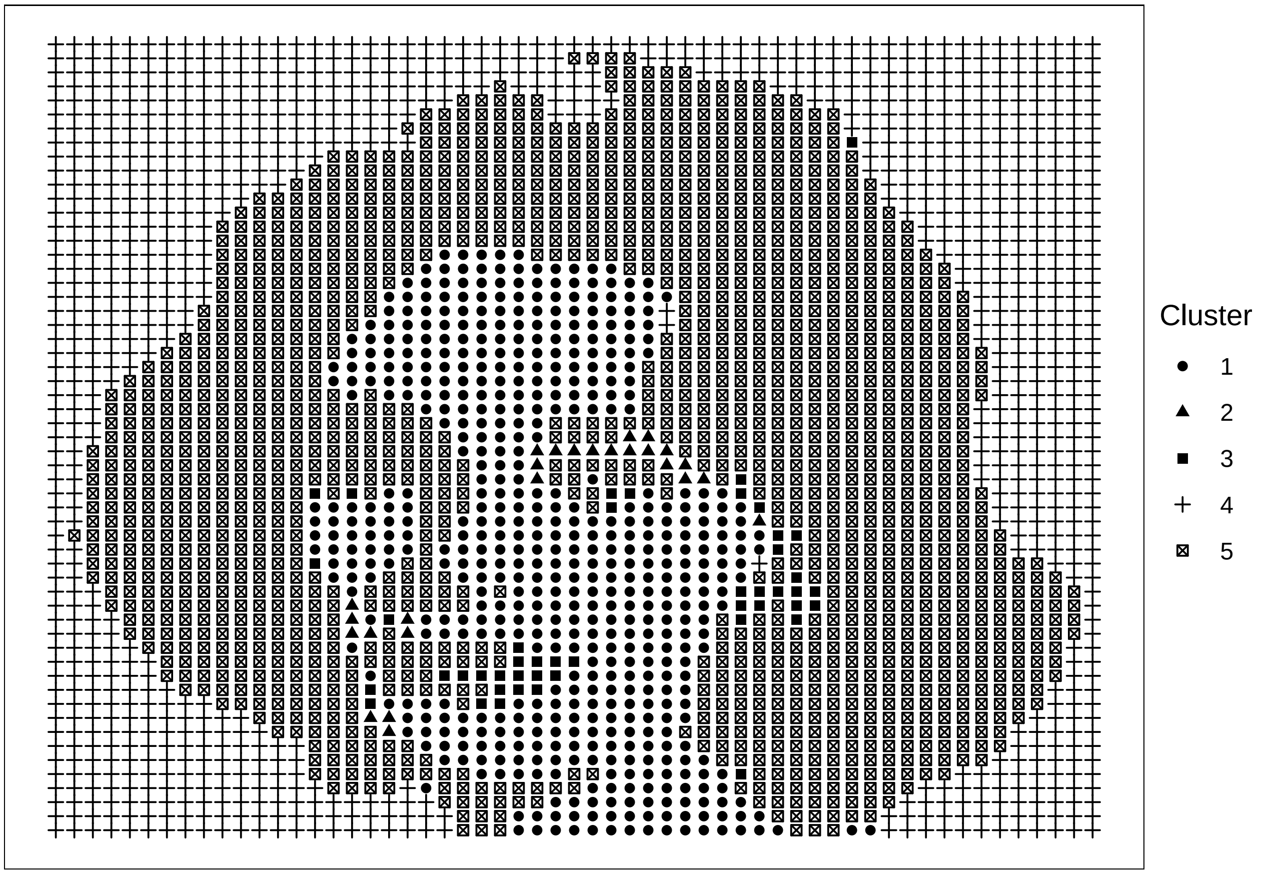
Here is where things are getting exciting. Let's make a 60 × 60 colourful cross-stitch of Elizabeth Taylor's picture, with 4 cluster centers:

```r
make_pattern(cluster_info, 5, 60)
```



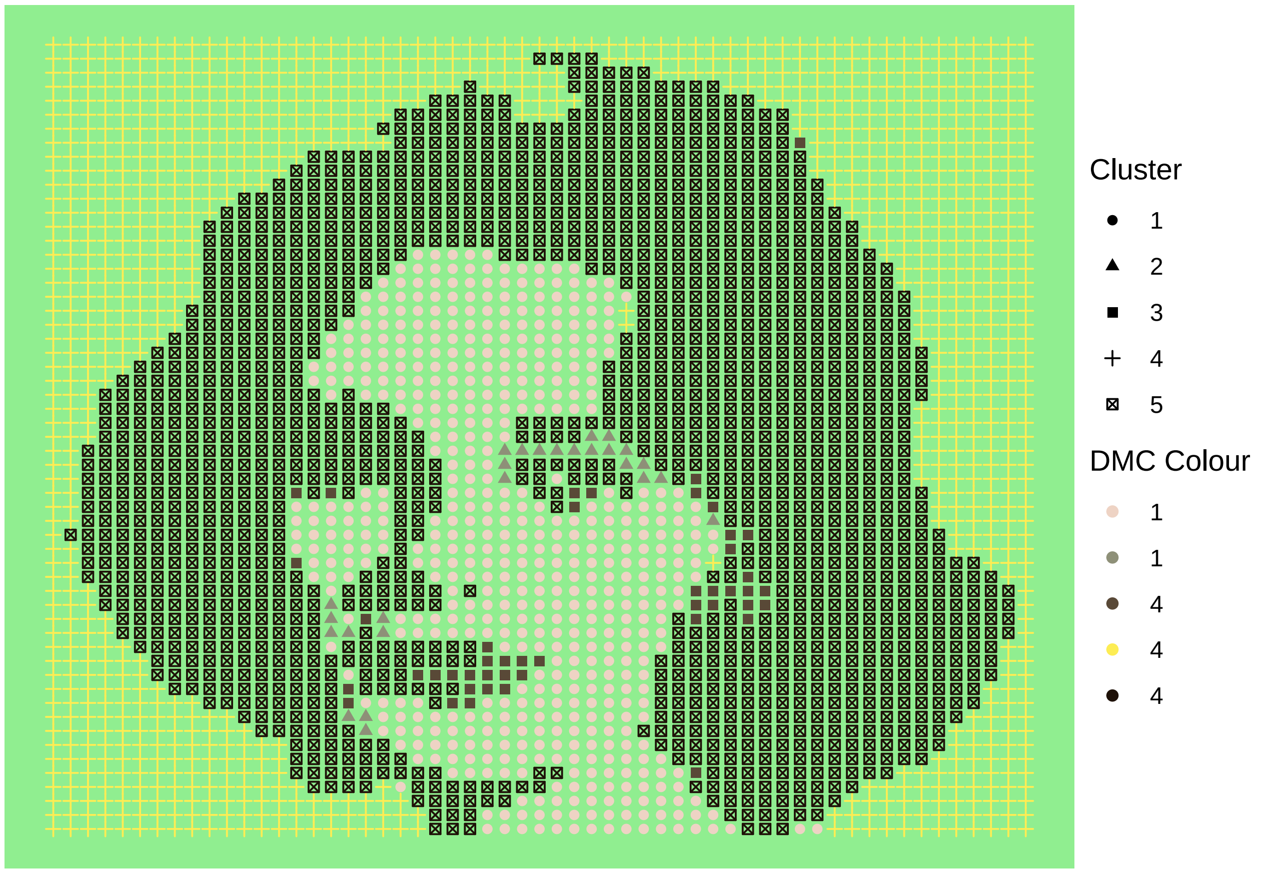
or, if we prefer a black-and-white version:

```r
make_pattern(cluster_info, 5, 60, black_white = TRUE)
```



Furthermore, we can make a colourful one with light green blackground:

```r
make_pattern(cluster_info, 5, 60, background_colour = "light green")
```



Session Information

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_CA.UTF-8/en_CA.UTF-8/en_CA.UTF-8/C/en_CA.UTF-8/en_CA.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] dmc_0.0.0.9001   cowplot_1.1.0    sp_1.4-2         yardstick_0.0.7
##  [5] workflows_0.2.1  tune_0.1.1       rsample_0.0.8    recipes_0.1.14
##  [9] parsnip_0.1.3    modeldata_0.1.0  infer_0.5.3      dials_0.0.9
## [13] scales_1.1.1     broom_0.7.2      tidymodels_0.1.1 forcats_0.5.0
## [17] stringr_1.4.0    dplyr_1.0.2      purrr_0.3.4      readr_1.3.1
## [21] tidyr_1.1.2      tibble_3.0.4     ggplot2_3.3.2    tidyverse_1.3.0
## [25] imager_0.42.3    magrittr_1.5
##
```

```
## loaded via a namespace (and not attached):
##  [1] colorspace_1.4-1   ellipsis_0.3.1     class_7.3-17       fs_1.4.2
##  [5] rstudioapi_0.11    listenv_0.8.0      furrr_0.2.1        farver_2.0.3
##  [9] prodlim_2019.11.13 fansi_0.4.1        lubridate_1.7.9    xml2_1.3.2
## [13] codetools_0.2-16   splines_4.0.2      knitr_1.29         readbitmap_0.1.5
## [17] jsonlite_1.7.1     pROC_1.16.2        dbplyr_1.4.4       png_0.1-7
## [21] compiler_4.0.2     httr_1.4.2         backports_1.1.8    assertthat_0.2.1
## [25] bmp_0.3            Matrix_1.2-18      cli_2.1.0          htmltools_0.5.0
## [29] tools_4.0.2        igraph_1.2.6       gtable_0.3.0       glue_1.4.2
## [33] Rcpp_1.0.5         cellranger_1.1.0   DiceDesign_1.8-1   vctrs_0.3.4
## [37] iterators_1.0.13   timeDate_3043.102  gower_0.2.2        xfun_0.16
## [41] globals_0.13.1     rvest_0.3.6        lifecycle_0.2.0    future_1.19.1
## [45] MASS_7.3-51.6      ipred_0.9-9        hms_0.5.3          parallel_4.0.2
## [49] yaml_2.2.1         rpart_4.1-15       stringi_1.5.3      foreach_1.5.1
## [53] tiff_0.1-5         lhs_1.1.1          lava_1.6.8         rlang_0.4.8
## [57] pkgconfig_2.0.3    evaluate_0.14      lattice_0.20-41    labeling_0.3
## [61] tidyselect_1.1.0   plyr_1.8.6         R6_2.5.0           magick_2.5.0
## [65] generics_0.1.0     DBI_1.1.0          pillar_1.4.6       haven_2.3.1
## [69] withr_2.2.0        survival_3.1-12    nnet_7.3-14        modelr_0.1.8
## [73] crayon_1.3.4       utf8_1.1.4         rmarkdown_2.3      jpeg_0.1-8.1
## [77] grid_4.0.2         readxl_1.3.1       blob_1.2.1         reprex_0.3.0
## [81] digest_0.6.27      GPfit_1.0-8        munsell_0.5.0
```