Assignment 1: Making a cross-stitch pattern

Chengshuo Zhang

10/31/2020

Task of the assignment

Taking an image and creating a cross-stitch pattern including thread color

Introduction

This vignette covers the following four functions to complete the task stated above:

`cluster_info <- process_image(image_file_name, k_list)`: we take a PNG or JPEG image and the list of the number of centers in the clustering as an input and the output is a tibble of information derived from the k_means that will be the input for the following three functions.

`scree_plot(cluster_info)`: it plots us a scree plot.

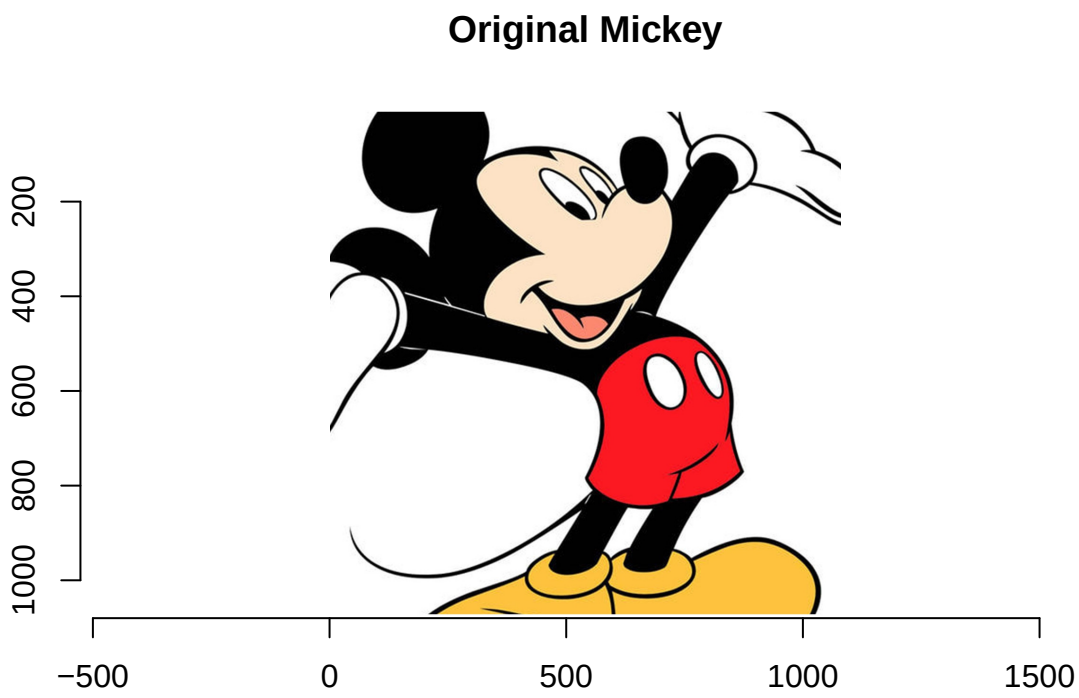
`color_strips(cluster_info)`: it produces color strips with DMC color closest to the cluster center color and returns a tibble including number of clusters k, DMC color mapping for each k, and gg plot for each k.

`make_pattern(cluster_info, k, x_size, black_white = FALSE, background_colour = NULL)`: it plots the cross-stitch pattern with a legend that has thread colors and a guide rid. It is the only function that use change_resolution(image_df, x_size). The detailed input info will be stated in the following section called basic usage.

Basic Usage

we will use the following `mickey.jpg` image with 1080*1080 resolutions as the example to go through the four functions and produce the cross-stitch pattern in the end.

```
set.seed(890)
im <- imager::load.image('mickey.jpg')
plot(im, main = 'Original Mickey', xlab = 'x', ylab = 'y')
```



First, we need to use `process_image(iamge_file_name, k_list)` to make a tibble of information from the image and the list of the number of centers in the clustering

1. `cluster_info <- process_image(image_file_name, k_list)`:

   Input:

   - image_file_name - a PNG or JPEG image.
   - k_list - the list of numbers of centres in the clustering

   Output:

   - cluster_info: A list or tibble of information derived from the k_means for k in k_list.

```
cluster_info= process_image('mickey.jpg',c(2:12))
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

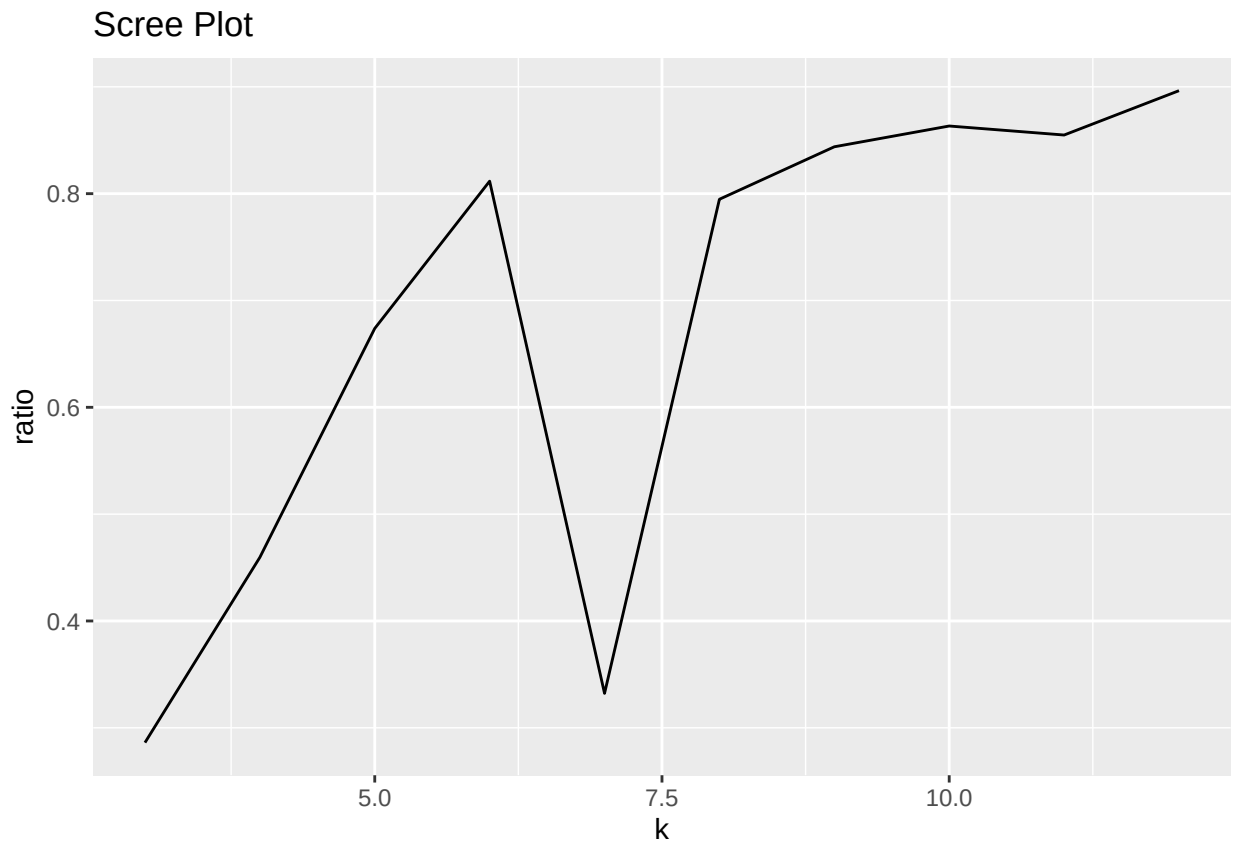
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: purrr

## Loading required package: tidymodels

## -- Attaching packages ------------------------------------- tidymodels 0.1.1 --

## v broom     0.7.2      v rsample   0.0.8
## v dials     0.0.9      v tibble    3.0.4
## v ggplot2   3.3.2      v tidyr     1.1.2
## v infer     0.5.3      v tune      0.1.1
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7
## v recipes   0.1.14

## -- Conflicts ---------------------------------------- tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x recipes::step()   masks stats::step()
```

the above code takes 8 different k values. we generate 8 k-means clustering models. we use them to cluster the image. AS a result, we have 8 set of data including the info about the color and image after clustering. we named the output as `cluster_info`.

Second, we use `scree_plot(cluster_info)` to plot a scree plot.

2. `scree_plot(cluster_info)`:

```
scree_plot(cluster_info)+
ggtitle('Scree Plot')
```



We uses the ratio of the current total within-cluster sum of squares and the previuous total within-cluster sum of squares From this ratio version plot, the number of clusters seems to be any number greater or equal to eight. The above plot looks more stable around 10. we do not want a too large number of clusters because we want to keep the pattern simple and avoid over fitting.

Then, we use `color_strips(cluster_info)`

3. `color_strips(cluster_info)`:

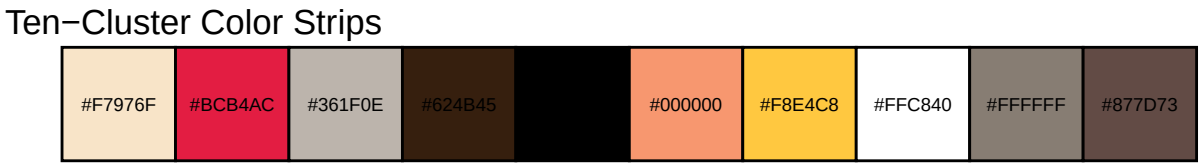
```
strips = color_strips(cluster_info)
```

```
## Loading required package: dmc
```

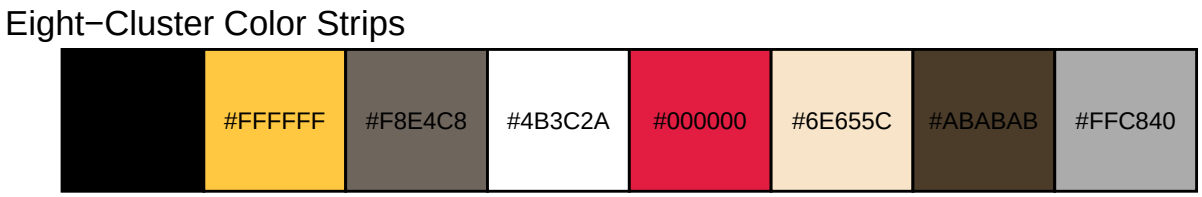
```
## Warning: Problem with `mutate()` input `..2`.
## i The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is omitte
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
## i Input `..2` is `dmc::dmc(col, visualize = FALSE)`.
## i The error occurred in row 1.
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

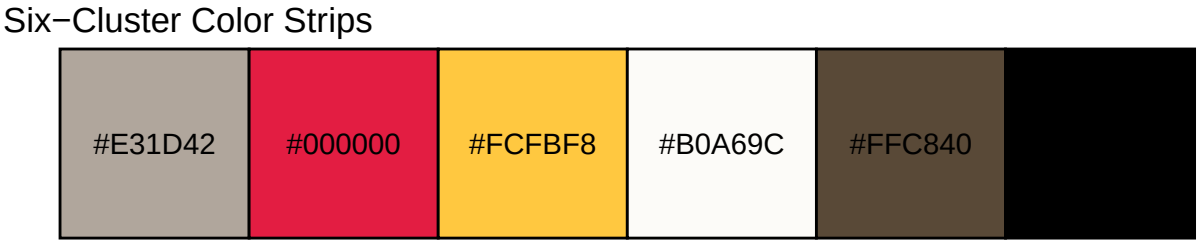
```
strips$plot[[which(10==strips$k)]]+ggtitle('Ten-Cluster Color Strips')
```



```
strips$plot[[which(8==strips$k)]]+ggtitle('Eight-Cluster Color Strips')
```




```
strips$plot[[which(6==strips$k)]]+ggtitle('Six-Cluster Color Strips')
```



from the above three color strips, we can see that the ten-cluster color strip matches more with our image's required colors comparing to the eight-cluster color strip and the six-cluster color strip, altouhg the image looks like having six color strips when we count colors by human eyes.

In the end, we will use `make_pattern(cluster_info, k, x_size, black_white = FALSE, background_colour = NULL)` to give us the final cross-stitch pattern with a legend that has thread color and a guide grid.

4. `make_pattern(cluster_info, k, x_size, black_white = FALSE, background_colour = NULL)`

   Input:

   - cluster_info - The output of process_image i.e. cluster info
   - k - The chosen cluster size
   - x_size - The (approximate) total number of possible stitches in the horizontal direction
   - black_white - (logical) Print the pattern in black and white (TRUE) or colour (FALSE, default)
   - background_colour - The colour of the background, which should not be stitched in the pattern.(Default is to not have a colour)

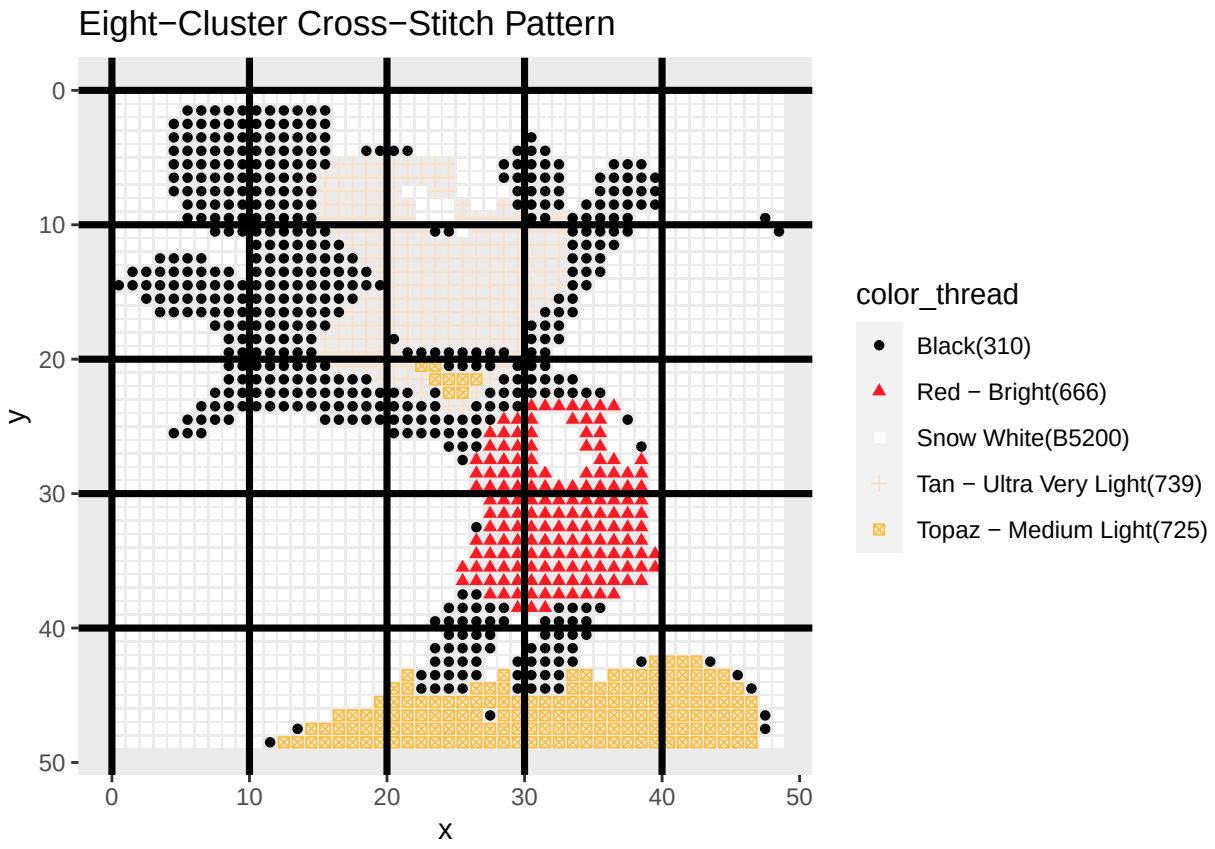
   Output:

   - It produces a cross-stitch pattern that can be followed, complete with a legend that has thread colour, and a guide grid.

```
make_pattern(cluster_info, 8, 50)+ggtitle('Eight-Cluster Cross-Stitch Pattern')
```

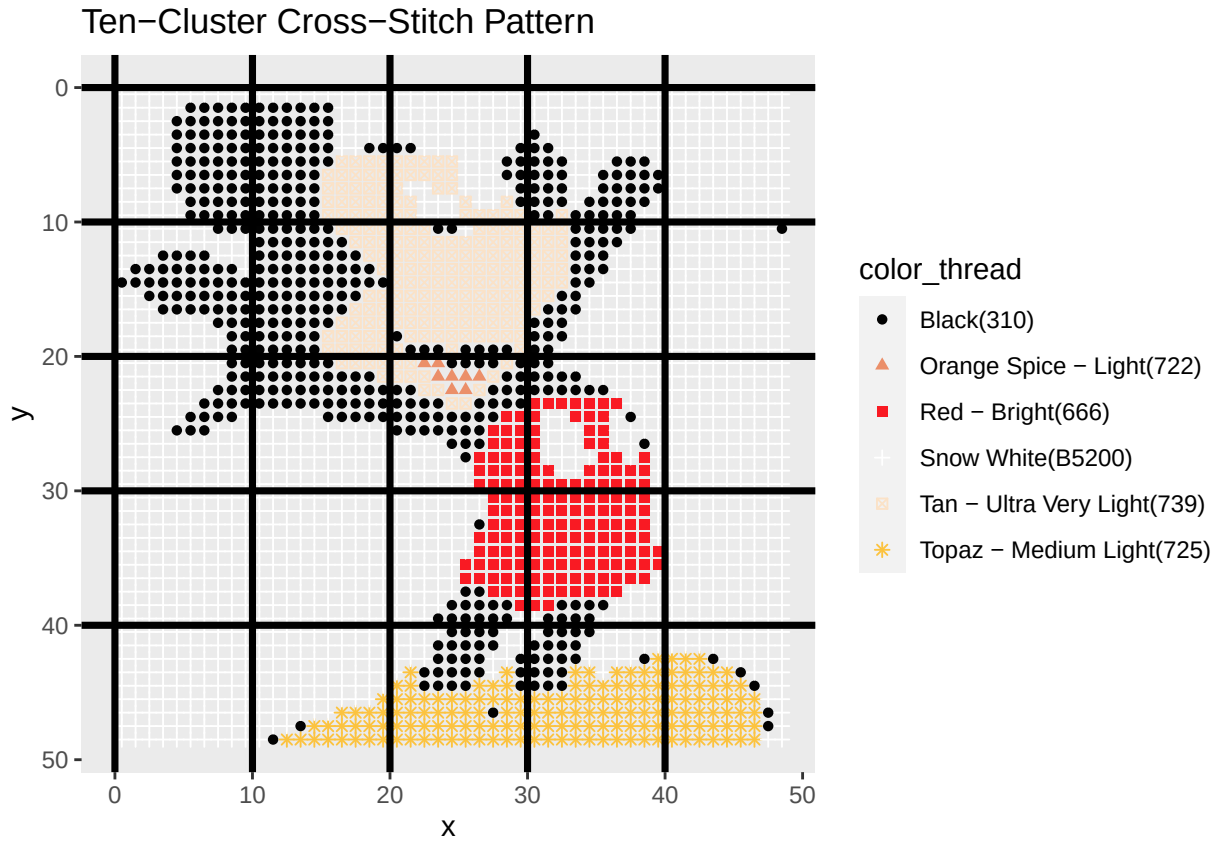
```
## Loading required package: sp
```

```
## Joining, by = "cluster"
```



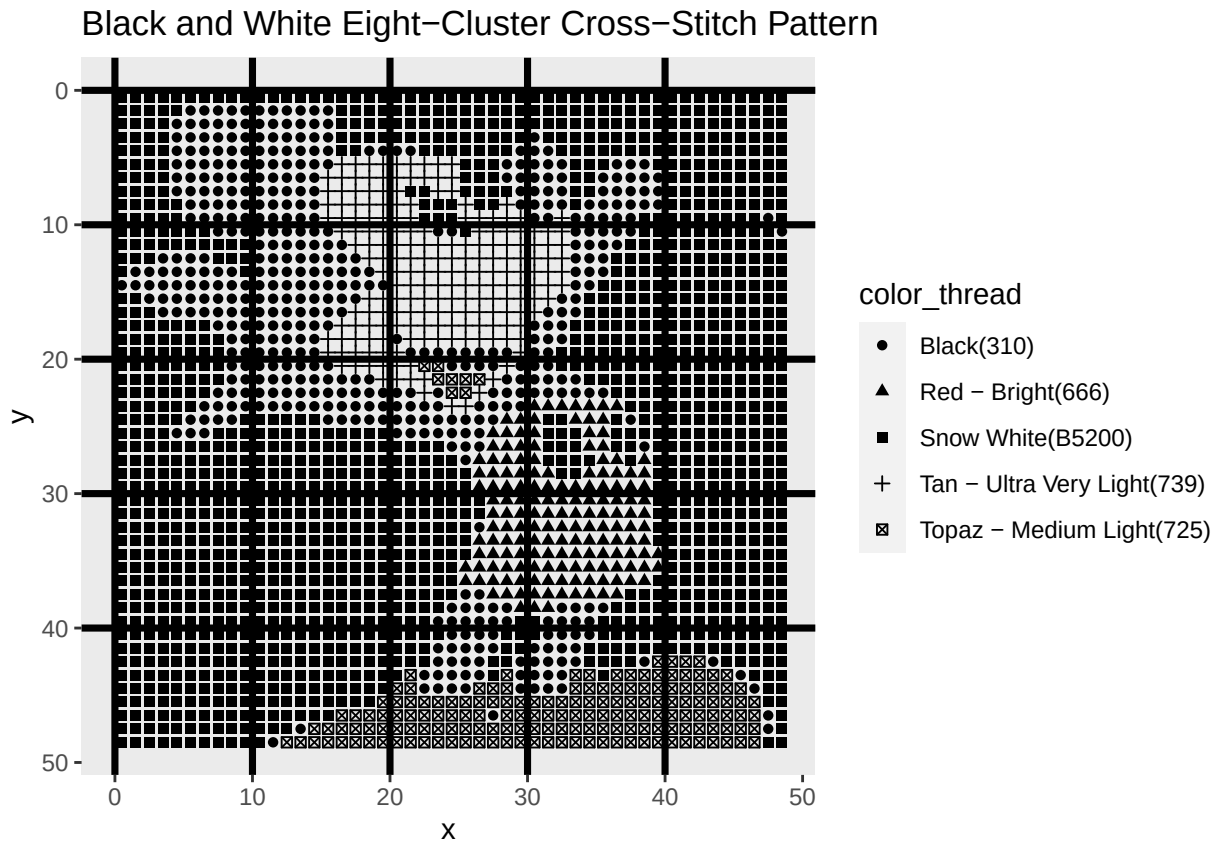
```
make_pattern(cluster_info, 10, 50)+ggtitle('Ten-Cluster Cross-Stitch Pattern')
```

```
## Joining, by = "cluster"
```



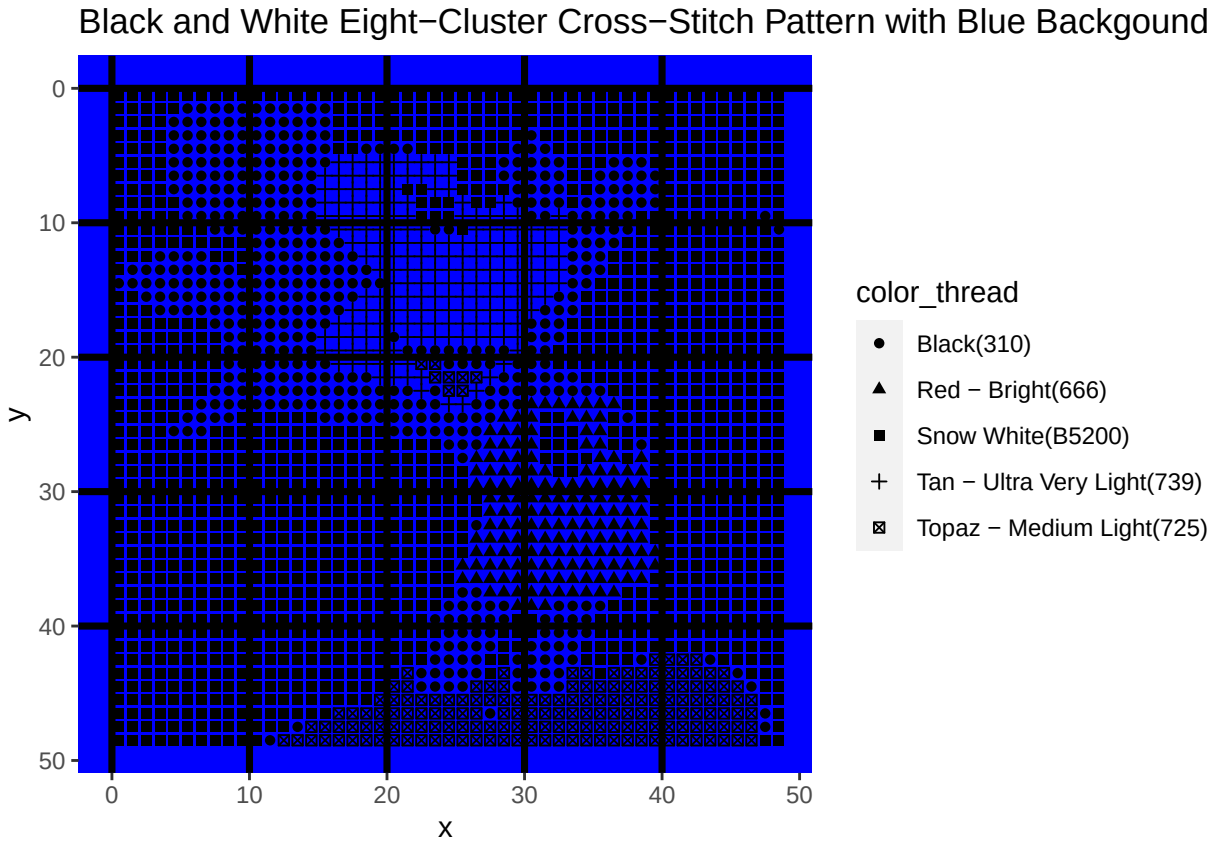

```
make_pattern(cluster_info, 8, 50, black_white = TRUE, background_colour = NULL)+ggtitle('Black and Whi
```

```
## Joining, by = "cluster"
```




```
make_pattern(cluster_info, 8, 50, black_white = TRUE, background_colour = 'blue')+ggtitle('Black and W
```

```
## Joining, by = "cluster"
```



The first image is the cross stitch pattern generated from the eight-cluster color strip. When we comparing it with the seond image which is the cross stitch pattern generated from the ten-cluster color strip, it is clearly that the second image has the same color strip division and the same number of colors with the original image. However, the first image has one less color for the tongue of the Mickey. The third image illistrates printing the pattern in black and white and the forth image shows that we can change the backgound color to any color we want it to be. Here, I choose the `blue` color.

Thus, from the outputs above, the Ten-Cluster Cross-Stitch Pattern is the one we want to use.