xstitch (Cross-stitch) Vignette

Dionysius Indraatmadja

11/3/2020

Preface

Ensure that this vignette R-markdown file, the R file `functions.R`, and the example image `warhol.png` are contained in the same folder. We set the working directory to this folder manually:

```
setwd(getwd())
source("functions.R")
#> Loading required package: magrittr
#>
#> Attaching package: 'imager'
#> The following object is masked from 'package:magrittr':
#>
#>     add
#> The following objects are masked from 'package:stats':
#>
#>     convolve, spectrum
#> The following object is masked from 'package:graphics':
#>
#>     frame
#> The following object is masked from 'package:base':
#>
#>     save.image
#> Warning: package 'tidyverse' was built under R version 4.0.3
#> -- Attaching packages ------------------------------------------- tidyverse 1.3.0 --
#> v ggplot2 3.3.2     v purrr   0.3.4
#> v tibble  3.0.4     v dplyr   1.0.2
#> v tidyr   1.1.2     v stringr 1.4.0
#> v readr   1.3.1     v forcats 0.5.0
#> Warning: package 'tibble' was built under R version 4.0.3
#> -- Conflicts ---------------------------------------------- tidyverse_conflicts() --
#> x imager::add()       masks magrittr::add()
#> x stringr::boundary() masks imager::boundary()
#> x tidyr::extract()    masks magrittr::extract()
#> x tidyr::fill()       masks imager::fill()
#> x dplyr::filter()     masks stats::filter()
#> x dplyr::lag()        masks stats::lag()
#> x purrr::set_names()  masks magrittr::set_names()
#> -- Attaching packages ------------------------------------------ tidymodels 0.1.1 --
#> v broom     0.7.1      v recipes   0.1.13
#> v dials     0.0.9      v rsample   0.0.8
#> v infer     0.5.3      v tune      0.1.1
#> v modeldata 0.0.2      v workflows 0.2.1
#> v parsnip   0.1.3      v yardstick 0.0.7
#> Warning: package 'workflows' was built under R version 4.0.3
#> -- Conflicts --------------------------------------------- tidymodels_conflicts() --
#> x imager::add()      masks magrittr::add()
#> x scales::discard()  masks purrr::discard()
#> x tidyr::extract()   masks magrittr::extract()
#> x tidyr::fill()      masks imager::fill()
#> x dplyr::filter()    masks stats::filter()
#> x recipes::fixed()   masks stringr::fixed()
#> x dplyr::lag()       masks stats::lag()
#> x purrr::set_names() masks magrittr::set_names()
#> x yardstick::spec()  masks readr::spec()
#> x recipes::step()    masks stats::step()
#> x dials::threshold() masks imager::threshold()
#>
#> Attaching package: 'sp'
#> The following object is masked from 'package:imager':
#>
#>     bbox
#> Warning: package 'cowplot' was built under R version 4.0.3
#>
#> Attaching package: 'cowplot'
#> The following object is masked from 'package:imager':
#>
#>     draw_text
```

Usage

Let's take a look at the image we will be using. The image is comprised of relatively few colours which are generally easily distinguishable and separable from one another. We also use an image that is not too high of a resolution to allow the program to run within a reasonable time.

```
image_file_name <- "warhol.png"
im<-imager::load.image(image_file_name)
plot(im, yaxt='n', axes=FALSE, ann=FALSE, main="original image")
```

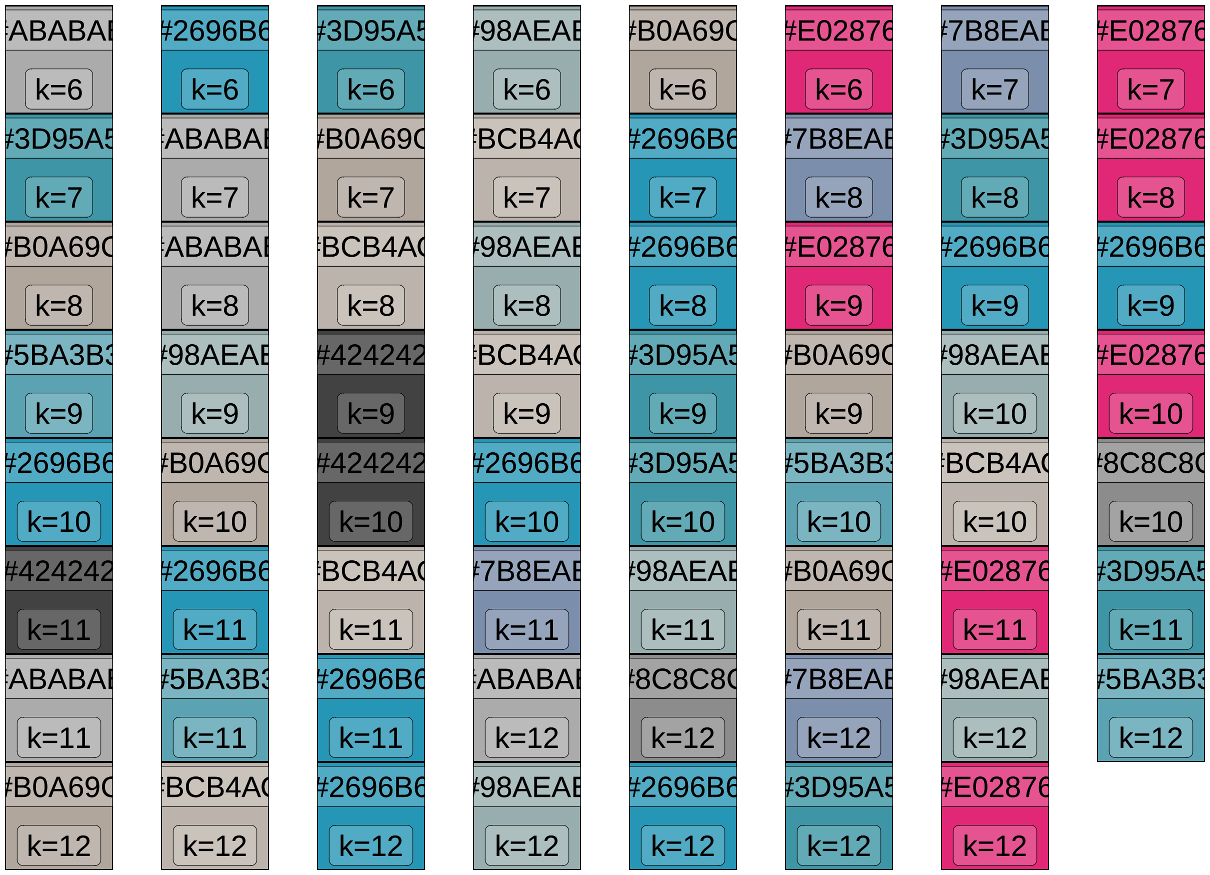
original image

We then perform k-means clustering on the image for the different levels specified in the list `k_list`. This process maps the original image RGB colours to a small set of DMC Floss colors. It outputs a list of useful summaries to use with the other functions. For this example, we'll compare clusterings with 6 to 12 centers.

```
k_list <- c(6:12)
cluster_info <- process_image(image_file_name, k_list)
#> Warning: Problem with `mutate()` input `kclust`.
#> i Quick-TRANSfer stage steps exceeded maximum (= 51257700)
#> i Input `kclust` is `map(k_list, ~kmeans(x = dat, centers = .x, nstart = 2))`.
#> Warning: Quick-TRANSfer stage steps exceeded maximum (= 51257700)
#> 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.
```

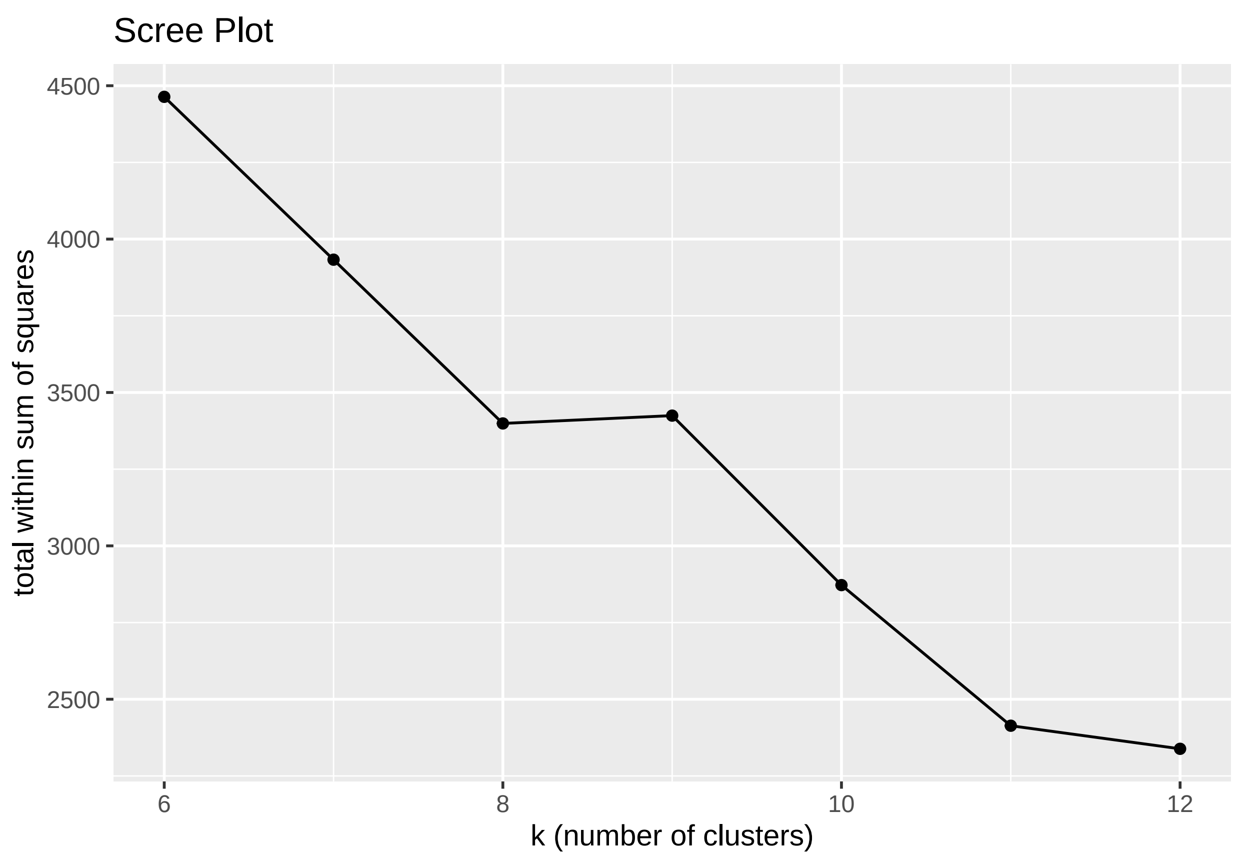
We can plot the colours used in all of the clusterings (or levels of k):

```
cluster_info %>% colour_strips
```



We can take a look at the scree plot to find the "elbow" of the graph, where the change in the total within sum of squares does not change significantly between values of k:

```
cluster_info %>% scree_plot
```

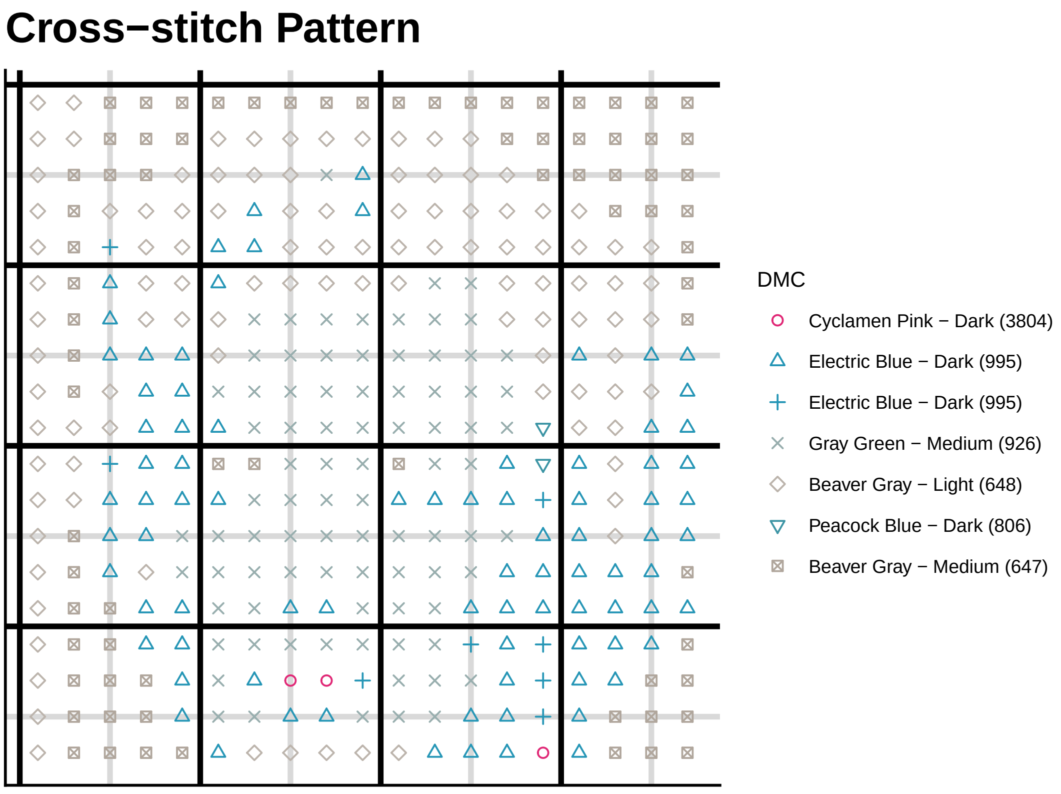


We see that the y-values decrease sharply at first but slow down after about `k=9` and again at `k=11`. After this point the values decrease at relatively similar rates. This `k` value where the y-values level off is one in which an increase in `k` yields not much new information compared to `k`. Our task now is to evaluate if the resulting pattern given by these cluster levels actually depict the design to our liking.

We are now ready to print the pattern. We must specify the number of clusters `k` and the approximate total number of stitches in the horizontal direction `x_size`. We also have optional arguments `black_white` and `background_colour`.

We first show the simplest usage: only `k` and `x_size` are specified. Note that the value of `k` is only valid for the values of `k_list`, with the smallest value permitted being 1.

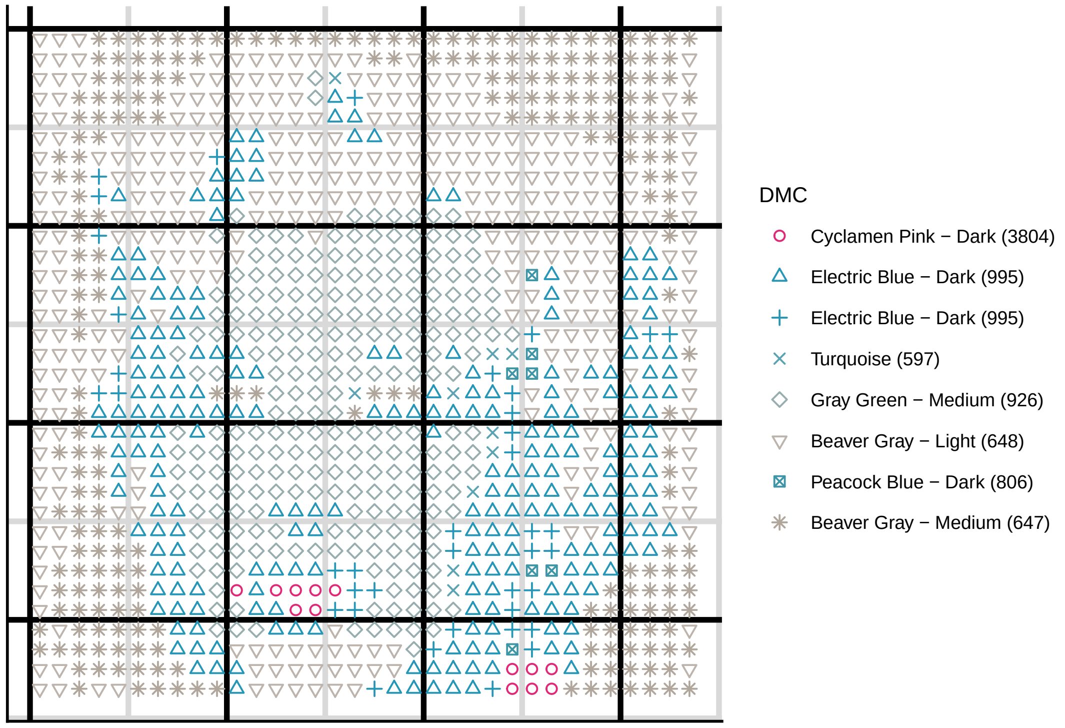
```
cluster_info %>% make_pattern(k=9, x_size=20)
```



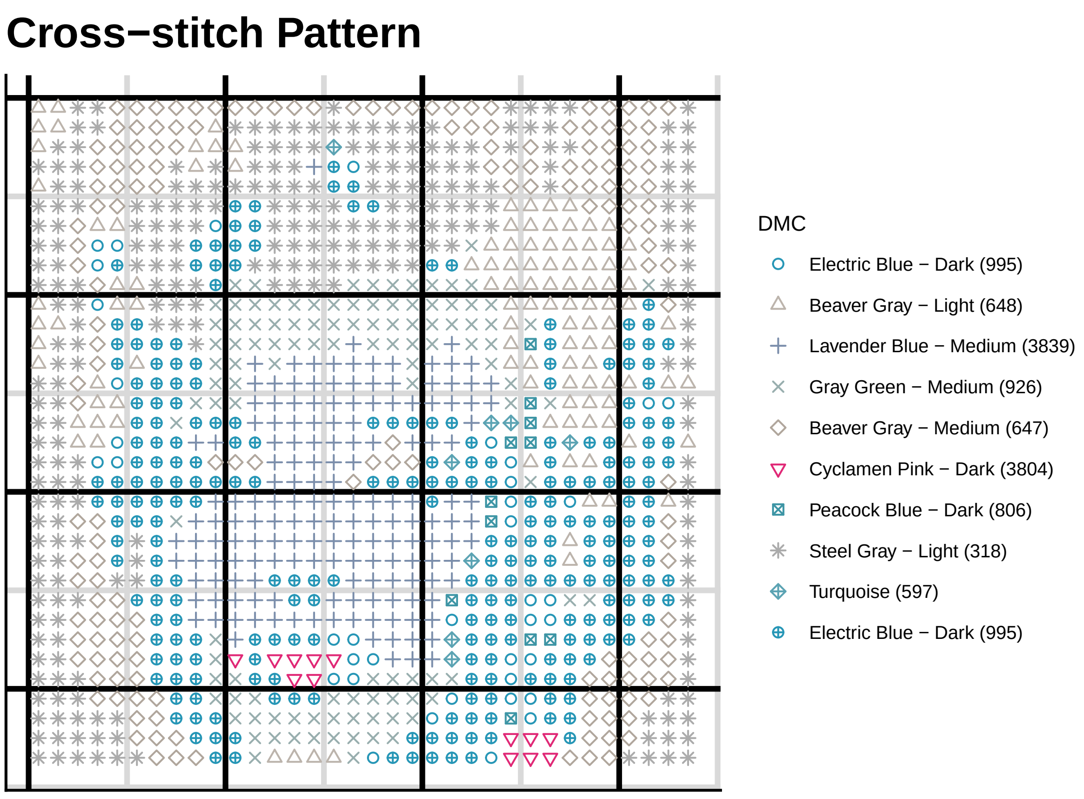
We see that `x_size=20` results in a pattern that does not show the design well. So, we just increase the value of `x_size`. We now compare the patterns for the different number of clusters:

```
cluster_info %>% make_pattern(k=9, x_size=35)
```

Cross−stitch Pattern

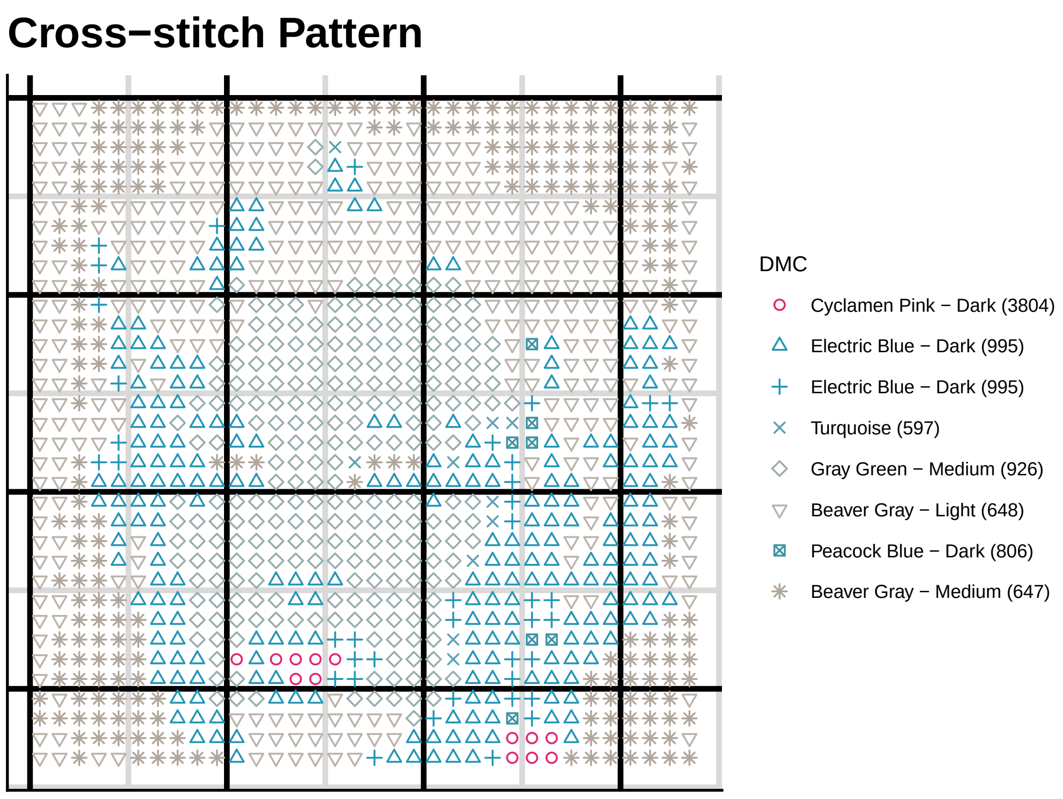

DMC
Cyclamen Pink − Dark (3804)
Electric Blue − Dark (995)
Electric Blue − Dark (995)
Turquoise (597)
Gray Green − Medium (926)
Beaver Gray − Light (648)
Peacock Blue − Dark (806)
Beaver Gray − Medium (647)

```r
cluster_info %>% make_pattern(k=11, x_size=35)
```



We find that `k=9` depicts the areas on the face around the nose and forehead well as well as the eyebrows on the face. The face in `k=11` appears to be too dark of a blue compared to the original image. Hence, we will choose `k=9` for the final cross-stitch pattern. Here it is again:

```
cluster_info %>% make_pattern(k=9, x_size=35)
```

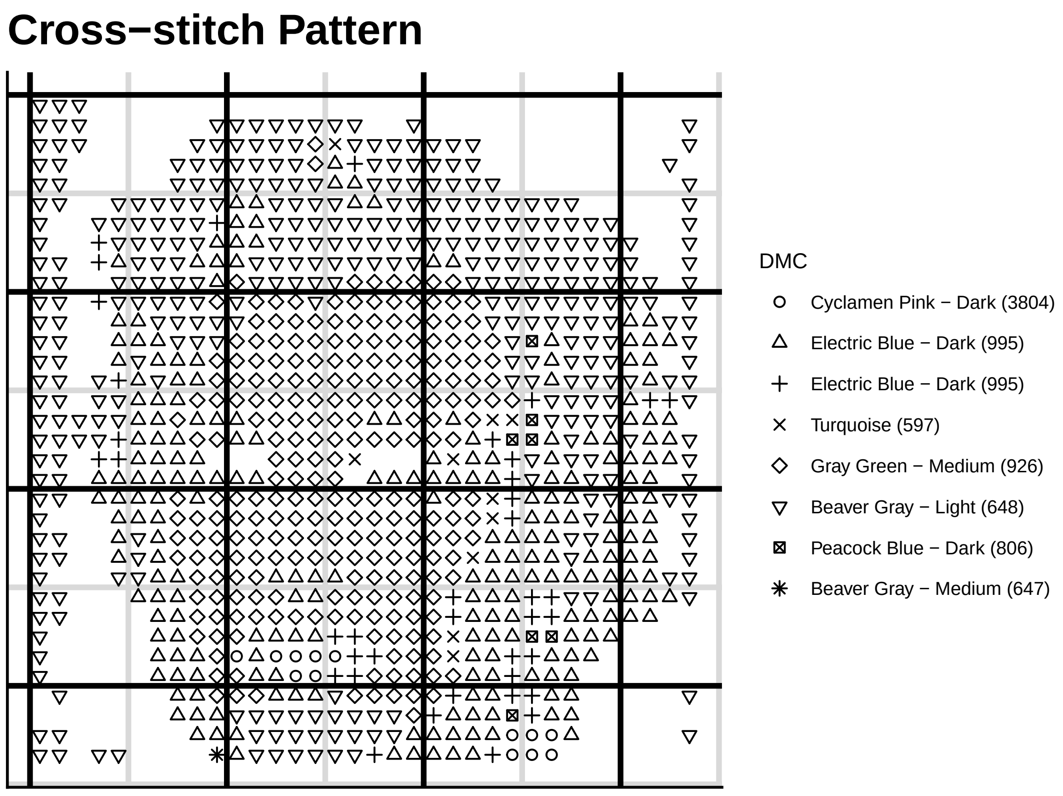


Optional features

We could stop here, but there are a couple of optional features available to us. In this example, we see that "Beaver Gray - Medium (647)" is a cluster corresponding to one of the background colours of the photo; it's not the face. We can exclude the background colour (as a DMC color) from the final pattern by specifying `background_colour="647"` to the function to omit all points with the DMC colour id "647".

Lastly, let's say we want to print the pattern and the brighter colors of the image would not show up well on the paper. By passing the argument `black_white=TRUE`, we can print the pattern in black and white instead of colour:

```
cluster_info %>% make_pattern(k=9, x_size=35, black_white=TRUE, background_colour = "647")
```



Conclusion

Starting with the filename of an image file in the same directory as `functions.R` and a list of numbers `k_list`, use `process_image()` to get its output `clustering_info`. Then pass it on to the functions `scree_plot()`, `colour_strips()` and `make_pattern()` to evaluate which clustering level works best for the image.