Cross-Stitch Vignette

Vinky Wang

19/10/2020

Introduction

This vignette covers the following functions:

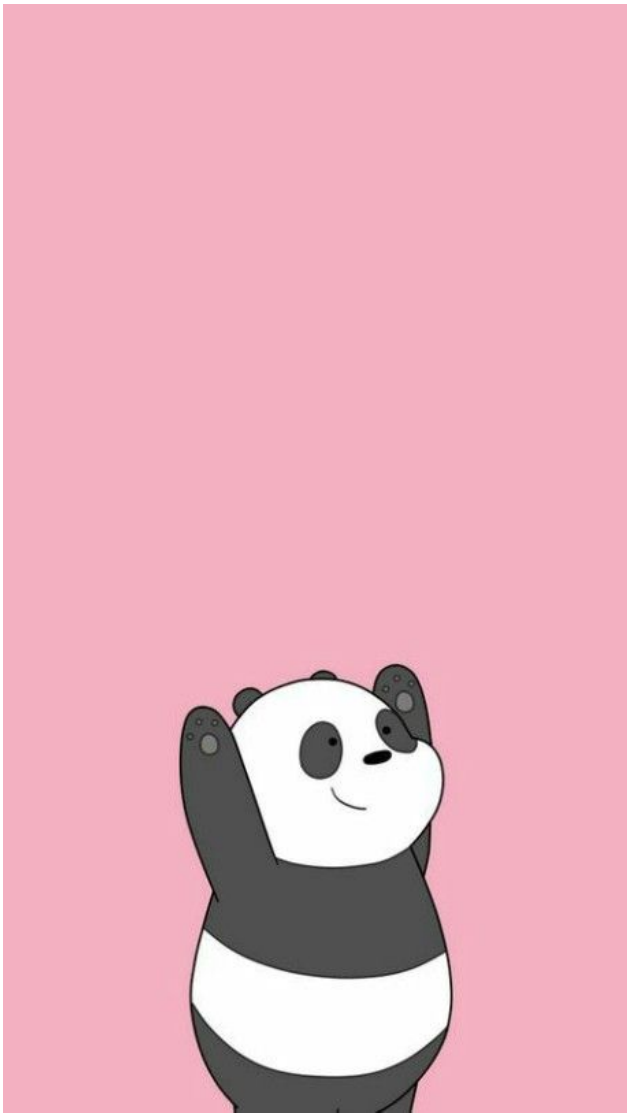
- `process_image()`
- `scree_plot()`
- `colour_strips()`
- `make_pattern()`

These functions can be used to create a cross-stitch pattern of an image using k-means clustering and matching colours to the nearest embroidery thread colour using the `dmc()` function in the DMC package.

K-means clustering

Let's start by taking an image and computing the k-means clustering for the pixels by RGB values and matching it to the nearest DMC thread colour.

I'll use this image:



We can load the image by copying the file path on our computer then storing it in a variable. Since we are not certain on exactly how many clusters we should use, we can experiment with k=2,6,14 clusters. The `process_image()` function will take the image and a list of possible k number of clusters to return a tibble where for each k cluster, the corresponding k-means clustering output, a tidied summary on a per-cluster level, a single row summary, the augmented classifications added to the original data set, and the colour containing the RGB with its matched DMC thread colour are computed.

```
im <- "image a1.jpg"
k = c(2,6,14)
my_cluster = process_image(im, k)
my_cluster
```

```
## # A tibble: 3 x 6
## # Groups:   k_list [3]
##   k_list kclust   tidied          glanced        augmented          colour
##    <dbl> <list>   <list>          <list>         <list>             <list>
## 1      2 <kmeans> <tibble [2 x ~ <tibble [1 x ~ <tibble [369,360 ~ <tibble [2 x~
## 2      6 <kmeans> <tibble [6 x ~ <tibble [1 x ~ <tibble [369,360 ~ <tibble [6 x~
## 3     14 <kmeans> <tibble [14 x~ <tibble [1 x ~ <tibble [369,360 ~ <tibble [14 ~
```

The following will demonstrate how you can access the information for k=6:

K-means clustering output The cluster centers

```
head(my_cluster$kclust[[2]][[2]])
```

```
##           R         G         B
## 1 0.9526403 0.6904116 0.7568851
## 2 0.7503155 0.6731419 0.6884301
## 3 0.3105791 0.3095123 0.3083052
## 4 0.9915167 0.9921373 0.9916431
## 5 0.5257909 0.4764626 0.4829344
## 6 0.1568893 0.1222141 0.1251979
```

All other available parameters

```
str(my_cluster$kclust[2])
```

```
## List of 1
##  $ :List of 9
##   ..$ cluster     : int [1:369360] 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ centers     : num [1:6, 1:3] 0.953 0.75 0.311 0.992 0.526 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:6] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:3] "R" "G" "B"
##   ..$ totss       : num 25864
##   ..$ withinss    : num [1:6] 18.3 11 30.9 15.4 10.5 ...
##   ..$ tot.withinss: num 118
##   ..$ betweenss   : num 25746
##   ..$ size        : int [1:6] 313934 926 25412 25601 978 2509
##   ..$ iter        : int 3
##   ..$ ifault      : int 0
##   ..- attr(*, "class")= chr "kmeans"
```

Tidied output of k-means clustering

```
my_cluster$tidied[2]
```

```
## [[1]]
## # A tibble: 6 x 6
##       R     G     B   size withinss cluster
##   <dbl> <dbl> <dbl>  <int>    <dbl> <fct>
## 1 0.953 0.690 0.757 313934     18.3 1
## 2 0.750 0.673 0.688    926     11.0 2
## 3 0.311 0.310 0.308  25412     30.9 3
## 4 0.992 0.992 0.992  25601     15.4 4
## 5 0.526 0.476 0.483    978     10.5 5
## 6 0.157 0.122 0.125   2509     31.9 6
```

The tidied output summarizes the k-means clustering for each RGB value with its corresponding cluster size, the total within cluster sum of squares, and its labelled cluster number.

The clustered RGB colours and its matched DMC thread colour

```
my_cluster$colour[2]
```

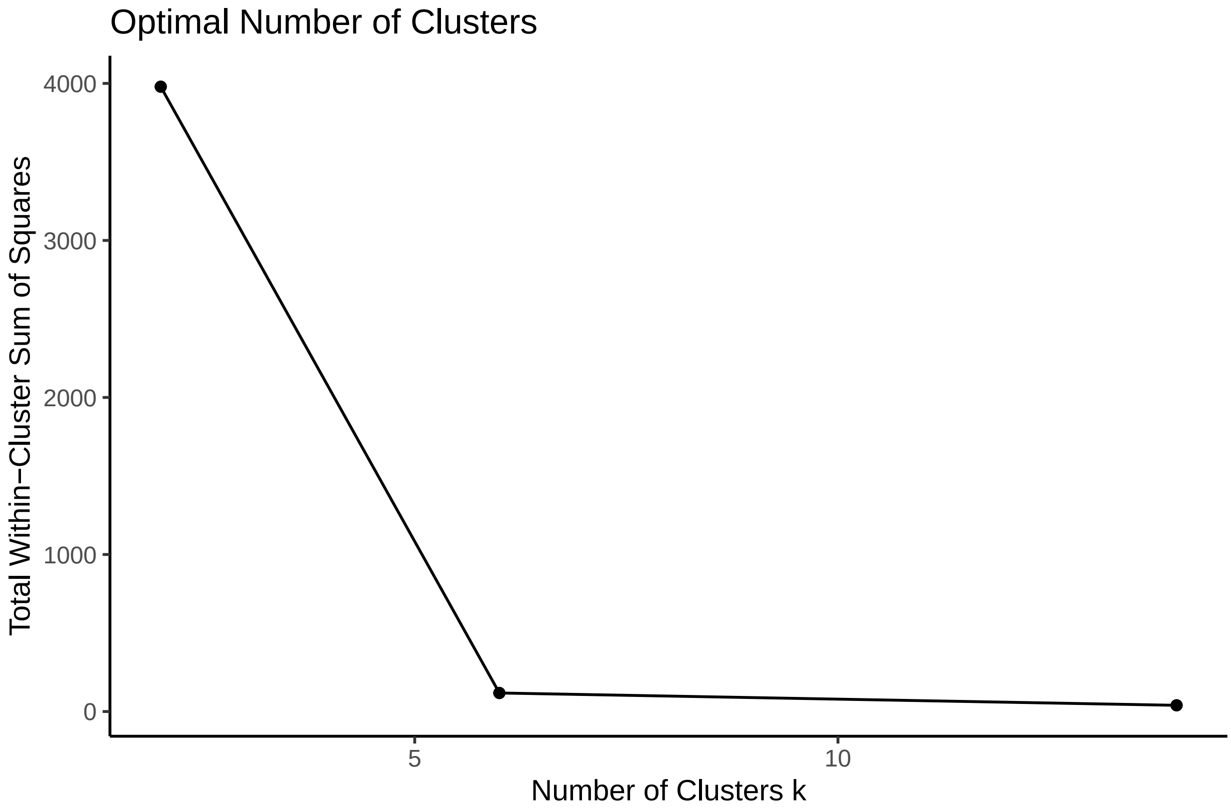
```
## [[1]]
## # A tibble: 6 x 7
##   hex_colour dmc   name                      hex       red green  blue
##   <chr>      <chr> <chr>                     <chr>   <dbl> <dbl> <dbl>
## 1 #F3B0C1    776   Pink - Medium             #FCB0B9   252   176   185
## 2 #BFACB0    452   Shell Gray - Medium       #C0B3AE   192   179   174
## 3 #4F4F4F    844   Beaver Gray - Ultra Dark  #484848    72    72    72
## 4 #FDFDFD    B5200 Snow White                #FFFFFF   255   255   255
## 5 #86797B    646   Beaver Gray - Dark        #877D73   135   125   115
## 6 #281F20    938   Coffee Brown - Ultra Dark #361F0E    54    31    14
```

The hex_colour corresponds to the hex code of the image computed by its RGB values. The dmc, name, and hex columns correspond to information for the matched DMC thread colours.

Scree Plot

One of the major challenges in using k-means clustering is determining the optimal number of clusters to use. A scree plot of the k number of clusters versus the total within sum of squares may be helpful. The function `scree_plot()` will take the cluster information computed above and plot its corresponding scree plot.

```
scree_plot(my_cluster)
```



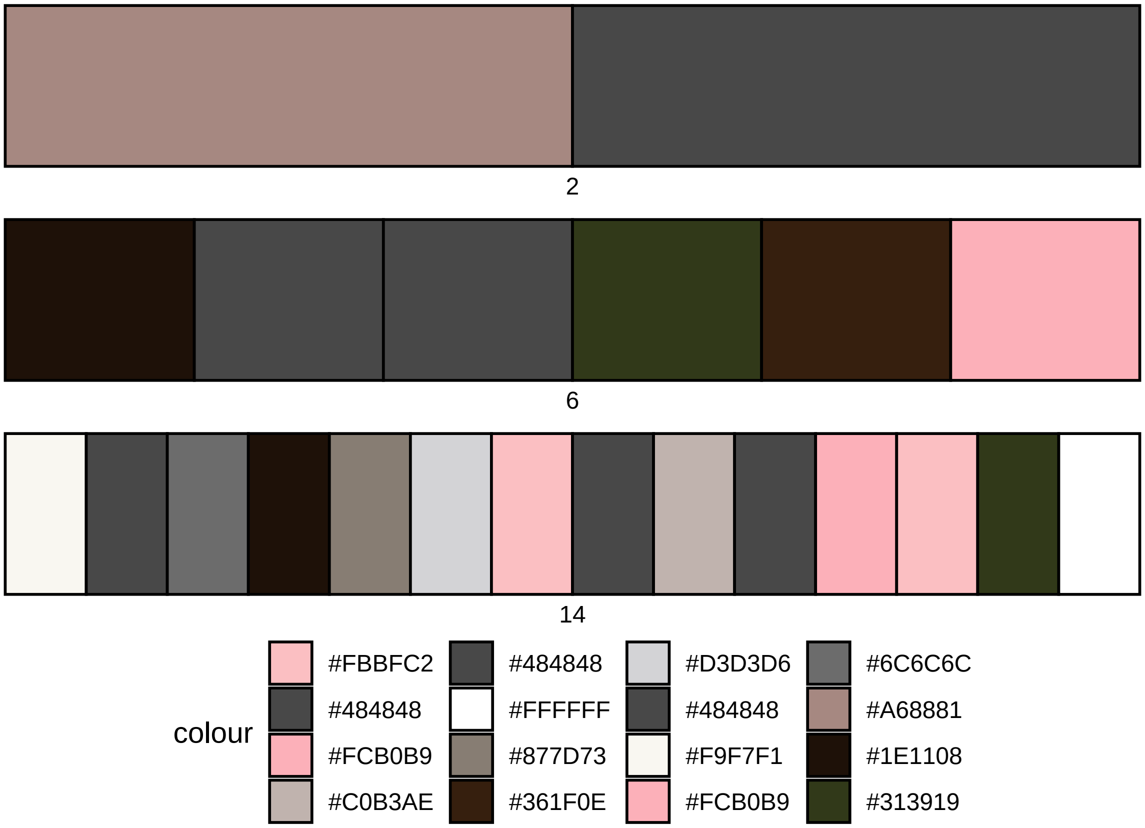
Scree Plot for Different Cluster Sizes

From the scree plot, we see a bend ('elbow') at k=6 which indicates that any additional clusters beyond 6 clusters have little value. Perhaps k=6 is a good choice for k-means clustering of our image. Although identifying the elbow is subjective, we can also plot the k number of clusters against the ratio of k and (k-1) and identify the kth cluster where the relative change becomes constant or less steep to confirm our choice.

Colour Strips

We can see which DMC colours were clustered from our image by calling the `colour_strips()` function. The colour strips can be helpful for the `make_pattern()` function below, where we can identify the background colour of the image and set the background_colour parameter to the HEX code of the DMC thread colour to create a cross stitch pattern without the background.

```r
my_colours = colour_strips(my_cluster)
my_colours
```



Colour Strips of Matched DMC Thread

We see that when we increase the number of clusters, we obtain more information and our number of colours subsequently increases. However, notice that when k=14, the colour strip shows a few repeated and similar colours. In contrast, k=6 uses fewer colours and are distinct which confirms our belief that k=6 seems like a good choice here. Of course, you can always use a very large number of clusters to obtain more information and gather more colours of the image, however this defeats the objective of clustering in which we are trying to reduce the amount of data into a lower but representative form.

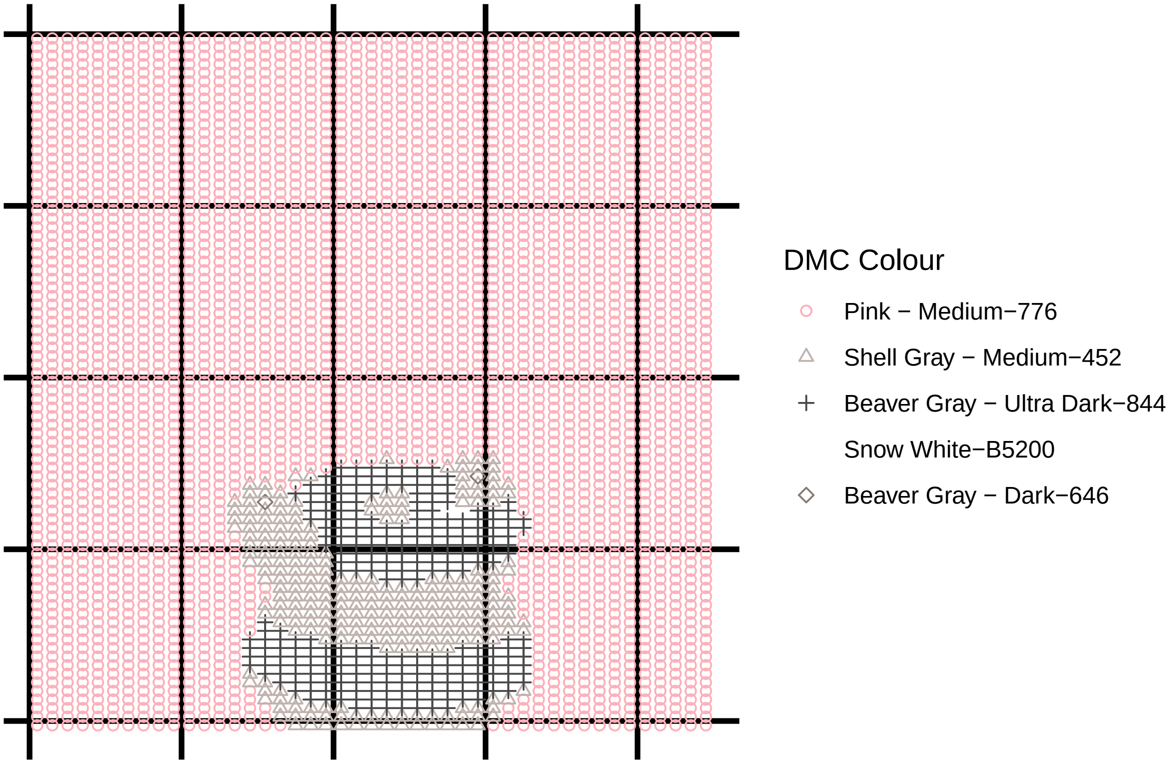
Cross-Stitching

Finally, we can plot the cross-stitched pattern of our image. Note that the cross-stitched pattern is performed on a lowered resolution of the original image such that the most common colour in the pixels will be combined to get the aggregate image and a small cluster that is not the most common colour may be dropped.

We have a few options on how we want our cross stitch pattern to look.

If we wish to obtain a coloured cross stitch pattern, then we just set the parameters black_white = FALSE and background_colour = NULL like so

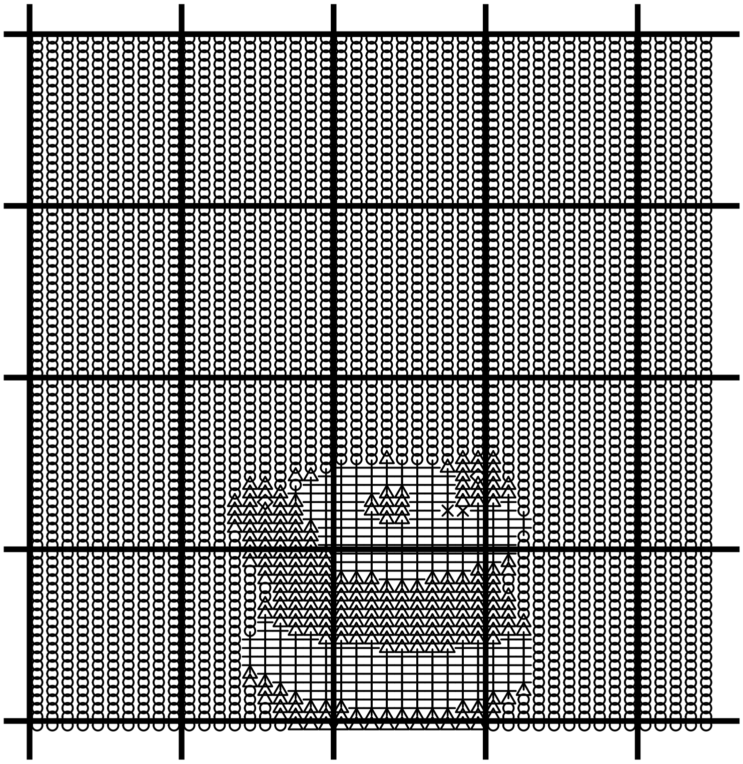
```
my_pattern = make_pattern(my_cluster, k=6, x_size=50, black_white = FALSE, background_colour = NULL)
my_pattern
```



Cross−Stitch with DMC Thread Colour

If we wish to obtain a cross-stitch only in black and white, then we just need to set black_white = TRUE.

```
my_pattern_bw = make_pattern(my_cluster, k=6, x_size=50, black_white = TRUE, background_colour = NULL)
my_pattern_bw
```



DMC Colour

○	Pink − Medium−776
△	Shell Gray − Medium−452
+	Beaver Gray − Ultra Dark−844
×	Snow White−B5200
◇	Beaver Gray − Dark−646

Cross−Stitch with DMC Thread Colour