

FEATURE FUNCTIONS

```
linear <- function(pal, space) {  
  colors <- convert_colour(t(col2rgb(pal)), "rgb", space) %>%  
    as.data.frame() %>%  
    mutate(x = row_number())  
  
  out <- lm(x ~ ., colors) %>%  
    summary() %>%  
    .$adj.r.squared  
  
  if(is.nan(out))  
    return(0)  
  
  out  
}
```

convert_colour() and compare_colour() from **farver** package

```

data_num <- data_list %>%
  mutate(# contains all a certain color
    contains_yellow = map_dbl(colors, ~ color_contains(.x, "yellow", "hsv")),
    contains_orange = map_dbl(colors, ~ color_contains(.x, "orange", "hsv")),
    contains_red = map_dbl(colors, ~ color_contains(.x, "red", "hsv")),
    contains_purple = map_dbl(colors, ~ color_contains(.x, "purple", "hsv")),
    contains_blue = map_dbl(colors, ~ color_contains(.x, "blue", "hsv")),
    contains_green = map_dbl(colors, ~ color_contains(.x, "green", "hsv")),
    contains_brown = map_dbl(colors, ~ color_contains(.x, "brown", "hsv")),
    contains_white = map_dbl(colors, ~ color_contains(.x, "white", "hsv")),
    contains_black = map_dbl(colors, ~ color_contains(.x, "black", "hsv")),
    # All a certain color
    all_contains_yellow = map_dbl(colors, ~ color_all_contains(.x, "yellow", "hsv")),
    all_contains_orange = map_dbl(colors, ~ color_all_contains(.x, "orange", "hsv")),
    all_contains_red = map_dbl(colors, ~ color_all_contains(.x, "red", "hsv")),
    all_contains_purple = map_dbl(colors, ~ color_all_contains(.x, "purple", "hsv")),
    all_contains_blue = map_dbl(colors, ~ color_all_contains(.x, "blue", "hsv")),
    all_contains_green = map_dbl(colors, ~ color_all_contains(.x, "green", "hsv")),
    all_contains_brown = map_dbl(colors, ~ color_all_contains(.x, "brown", "hsv")),
    all_contains_white = map_dbl(colors, ~ color_all_contains(.x, "white", "hsv")),
    all_contains_black = map_dbl(colors, ~ color_all_contains(.x, "black", "hsv")),
    # Is it linear in a perceptually uniform space
    linear = map_dbl(colors, ~ linear(.x, "hunterlab")),
    linear_deutan = map_dbl(colors, ~ linear(deutan(.x), "hunterlab")),
    linear_protan = map_dbl(colors, ~ linear(protan(.x), "hunterlab")),
    linear_tritan = map_dbl(colors, ~ linear(tritan(.x), "hunterlab")),
    # Twice linear
    twice_linear = map_dbl(colors, ~ linear_split(.x, "hunterlab")),
    twice_linear_deutan = map_dbl(colors, ~ linear_split(deutan(.x), "hunterlab")),
    twice_linear_protan = map_dbl(colors, ~ linear_split(protan(.x), "hunterlab")),
    twice_linear_tritan = map_dbl(colors, ~ linear_split(tritan(.x), "hunterlab")),
  )

```