# PPG Paint Colors: Final Project

## Example: read data, save, and reload model object

Grishma Palkar

# Overview

This RMarkdown shows how to read in the final project data. It also shows how to calculate the logit-transformed response and setup the binary outcome for use with `caret` or `tidymodels`. It also demonstrates how to fit a simple model (with `lm()`), save that model, and load it back into the workspace. You may find these actions helpful as you work through the project.

**You must download the data from Canvas and save the data in the same directory as this RMarkdown file.**

# Load packages

This example uses the `tidyverse` suite of packages.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0     ✔ purrr   1.0.1
## ✔ tibble  3.2.1     ✔ dplyr   1.1.2
## ✔ tidyr   1.3.0     ✔ stringr 1.5.0
## ✔ readr   2.1.3     ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(coefplot)
```

# Read data

Please download the final project data from Canvas. If this Rmarkdown file is located in the same directory as the downloaded CSV file, it will be able to load in the data for you. It is **highly** recommended that you use an RStudio RProject to easily manage the working directory and file paths of the code and objects associated with the final project.

The code chunk below reads in the final project data.

```
df <- readr::read_csv("paint_project_train_data.csv", col_names = TRUE)
```

```
## Rows: 835 Columns: 8
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (2): Lightness, Saturation
## dbl (6): R, G, B, Hue, response, outcome
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

The `readr::read_csv()` function displays the data types and column names associated with the data. However, a glimpse is shown below that reveals the number of rows and also shows some of the representative values for the columns.

```
df %>% glimpse()
```

```
## Rows: 835
## Columns: 8
## $ R          <dbl> 172, 26, 172, 28, 170, 175, 90, 194, 171, 122, 0, 88, 144, …
## $ G          <dbl> 58, 88, 94, 87, 66, 89, 78, 106, 68, 151, 121, 140, 82, 163…
## $ B          <dbl> 62, 151, 58, 152, 58, 65, 136, 53, 107, 59, 88, 58, 132, 50…
## $ Lightness  <chr> "dark", "dark", "dark", "dark", "dark", "dark", "dark", "da…
## $ Saturation <chr> "bright", "bright", "bright", "bright", "bright", "bright",…
## $ Hue        <dbl> 4, 31, 8, 32, 5, 6, 34, 10, 1, 21, 24, 22, 36, 16, 26, 12, …
## $ response   <dbl> 12, 10, 16, 10, 11, 16, 10, 19, 14, 25, 14, 19, 14, 38, 15,…
## $ outcome    <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,…
```

The data consist of continuous and categorical inputs. The `glimpse()` shown above reveals the data type for each variable which state to you whether the input is continuous or categorical. The RGB color model inputs, `R`, `G`, and `B` are continuous (dbl) inputs. The HSL color model inputs consist of 2 categorical inputs, `Lightness` and `Saturation`, and a continuous input, `Hue`. Two outputs are provided. The continuous output, `response`, and the Binary output, `outcome`. However, the data type of the Binary outcome is numeric because the Binary `outcome` is **encoded** as `outcome = 1` for the EVENT and `outcome = 0` for the NON-EVENT.

# Regression task

#(iiA) Linear models

As stated in the project guidelines, you will **not** model the continuous output, `response`, directly. The `response` is a bounded variable between 0 and 100. The `response` must be transformed to an unbounded variable to appropriately be modeled by a Gaussian likelihood. We are making this transformation because we want the **uncertainty** in the predicted output to also satisfy output constraints. If we did not make this transformation the uncertainty could violate the bounds, which would mean the model is providing unphysical results! By logit-transforming `response`, we will fully respect the bounds of the output variable.

The code chunk below assembles the data for Part ii) of the project. You should use this data set for all regression modeling tasks. The logit-transformed output is named `y`. The `dfii` dataframe as the original `response` and Binary output, `outcome`, removed. This way you can focus on the variables specific to the regression task.

```
dfii <- df %>%
  mutate(y = boot::logit( (response - 0) / (100 - 0) ) ) %>%
  select(R, G, B,
         Lightness, Saturation, Hue,
         y)

dfii %>% glimpse()
```

```
## Rows: 835
## Columns: 7
## $ R          <dbl> 172, 26, 172, 28, 170, 175, 90, 194, 171, 122, 0, 88, 144, …
## $ G          <dbl> 58, 88, 94, 87, 66, 89, 78, 106, 68, 151, 121, 140, 82, 163…
## $ B          <dbl> 62, 151, 58, 152, 58, 65, 136, 53, 107, 59, 88, 58, 132, 50…
## $ Lightness  <chr> "dark", "dark", "dark", "dark", "dark", "dark", "dark", "da…
## $ Saturation <chr> "bright", "bright", "bright", "bright", "bright", "bright",…
## $ Hue        <dbl> 4, 31, 8, 32, 5, 6, 34, 10, 1, 21, 24, 22, 36, 16, 26, 12, …
## $ y          <dbl> -1.9924302, -2.1972246, -1.6582281, -2.1972246, -2.0907411,…
```

```
#Process data
dfii_train <- dfii %>%
  select(R,G,B,Hue,y) %>%
  scale() %>% as.data.frame() %>%
  bind_cols(dfii %>% select(Lightness,Saturation))

dfii_train %>% glimpse()
```

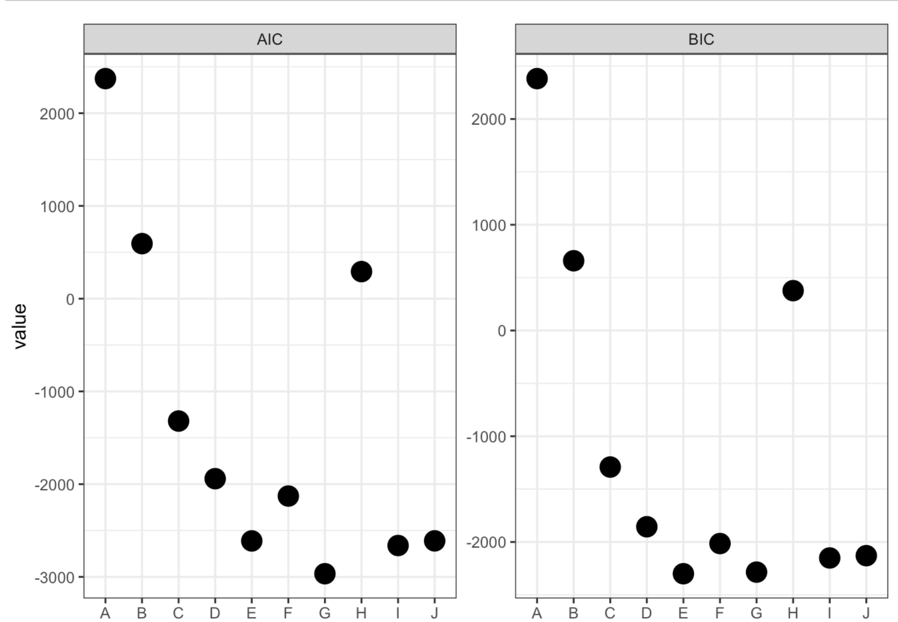
```
## Rows: 835
## Columns: 7
## $ R          <dbl> -0.19790120, -2.74419521, -0.19790120, -2.70931447, -0.2327…
## $ G          <dbl> -2.3619736, -1.7631189, -1.6433480, -1.7830807, -2.2022790,…
## $ B          <dbl> -1.7994266, -0.1706872, -1.8726283, -0.1523868, -1.8726283,…
## $ Hue        <dbl> -1.3548215, 1.3198239, -0.9585777, 1.4188848, -1.2557605, -…
## $ y          <dbl> -1.5899718, -1.7628901, -1.3077880, -1.7628901, -1.6729807,…
## $ Lightness  <chr> "dark", "dark", "dark", "dark", "dark", "dark", "dark", "da…
## $ Saturation <chr> "bright", "bright", "bright", "bright", "bright", "bright",…
```

**Important**: It is up to you as to whether further pre-processing of the inputs are required before fitting the models. ##(1) You must therefore train 10 different models!

```
# Model 1: Intercept-only model
mod01 <- lm(y ~ 1, data = dfii_train)
mod01 %>% readr::write_rds("model01.rds")

# Model 2:Categorical variables only
mod02 <- lm(y~(Lightness + Saturation ), data = dfii_train)
mod02 %>% readr::write_rds("model02.rds")

# Model 3:Continuous variables only
mod03 <- lm(y~(R+G+B+Hue), data = dfii_train)
mod03 %>% readr::write_rds("model03.rds")

# Model 4:All categorical and continuous variables – linear additive
mod04 <- lm(y~(R+G+B+Lightness + Saturation + Hue), data = dfii_train)
mod04 %>% readr::write_rds("model04.rds")

# Model 5:Interaction of the categorical inputs with all continuous inputs main effects
mod05 <- lm(y~(R+G+B+Hue)*(Lightness + Saturation), data = dfii_train)
mod05 %>% readr::write_rds("model05.rds")

# Model 6: Add categorical inputs to all main effect and all pairwise interactions of continuous inputs
mod06 <- lm(y ~ (R+G+B+Hue)*(R+G+B+Hue)+(Lightness + Saturation ), data = dfii_train)
mod06 %>% readr::write_rds("model06.rds")

# Model 7 : Interaction of the categorical inputs with all main effect and all pairwise interactions of co
ntinuous inputs
mod07 <- lm(y ~ (R+G+B+Hue)*(R+G+B+Hue)*(Lightness+Saturation) , data = dfii_train)
mod07 %>% readr::write_rds("model07.rds")

#  3 models with basis functions of your choice
# Model 8:Try non-linear basis functions based on your EDA.
mod08 <- lm(y ~ I(R^2) + I(G^2) + I(B^2) + I(Hue^2) + Lightness + Saturation, data = dfii_train)
mod08 %>% readr::write_rds("model08.rds")

# Model 9:Can consider interactions of basis functions with other basis functions!
mod09 <- lm(y ~ (R+G+B+Lightness + Saturation + Hue)*(R+G+B+Lightness + Saturation + Hue), data = dfii_tra
in)
mod09 %>% readr::write_rds("model09.rds")

# Model 10:Can consider interactions of basis functions with the categorical inputs!
mod10 <- lm(y ~ (R+G+B+Lightness + Saturation + Hue)*(Lightness+Saturation), data = dfii_train)
mod10 %>% readr::write_rds("model10.rds")
```

##(2) Which of the 10 models is the best? What performance metric did you use to make your selection?

```
extract_metrics <- function(mod_object, mod_name)
{
  broom::glance(mod_object) %>%
    mutate(model_name = mod_name)
}

glm_mle_results <- purrr::map2_dfr(list(mod01, mod02, mod03, mod04,
                                        mod05, mod06, mod07, mod08, mod09, mod10),
                                   LETTERS[1:10],extract_metrics)


glm_mle_results %>%
  select(model_name, AIC, BIC) %>%
  pivot_longer(c(AIC, BIC)) %>%
  ggplot(mapping = aes(x = model_name, y = value)) +
  geom_point(size = 5) +
  facet_wrap(~name, scales = 'free_y') +
  theme_bw()
```

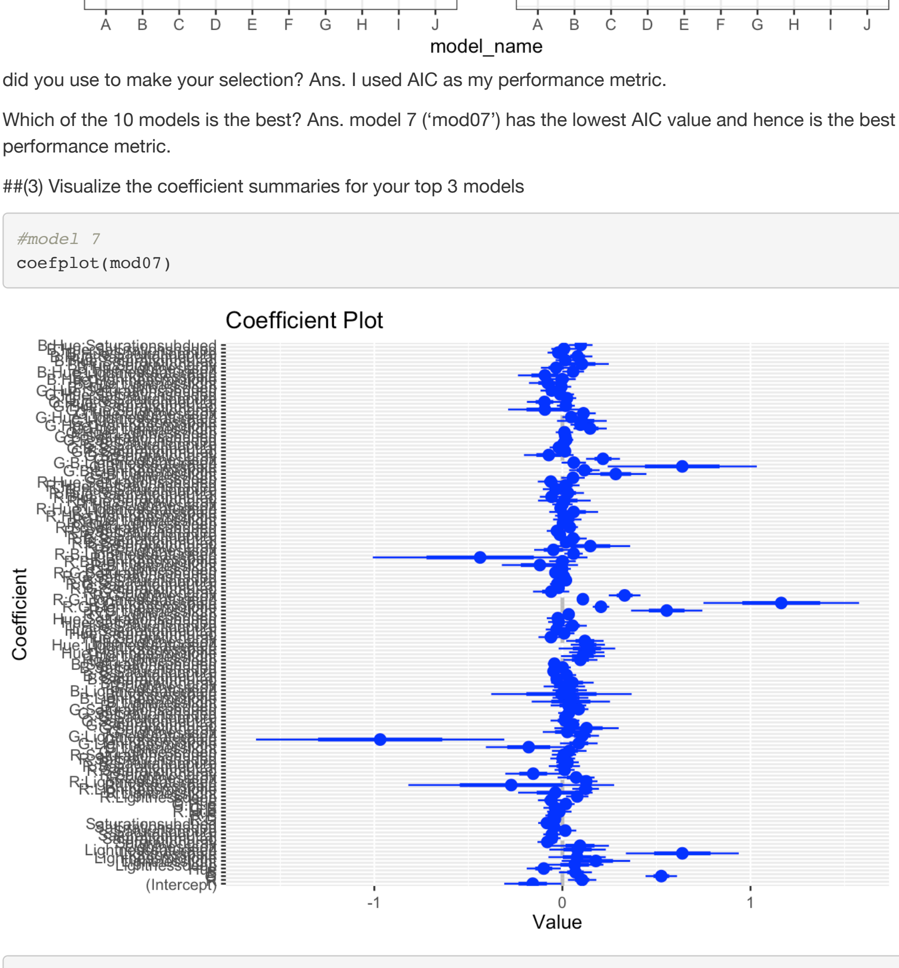



What performance metric did you use to make your selection? Ans. I used AIC as my performance metric.

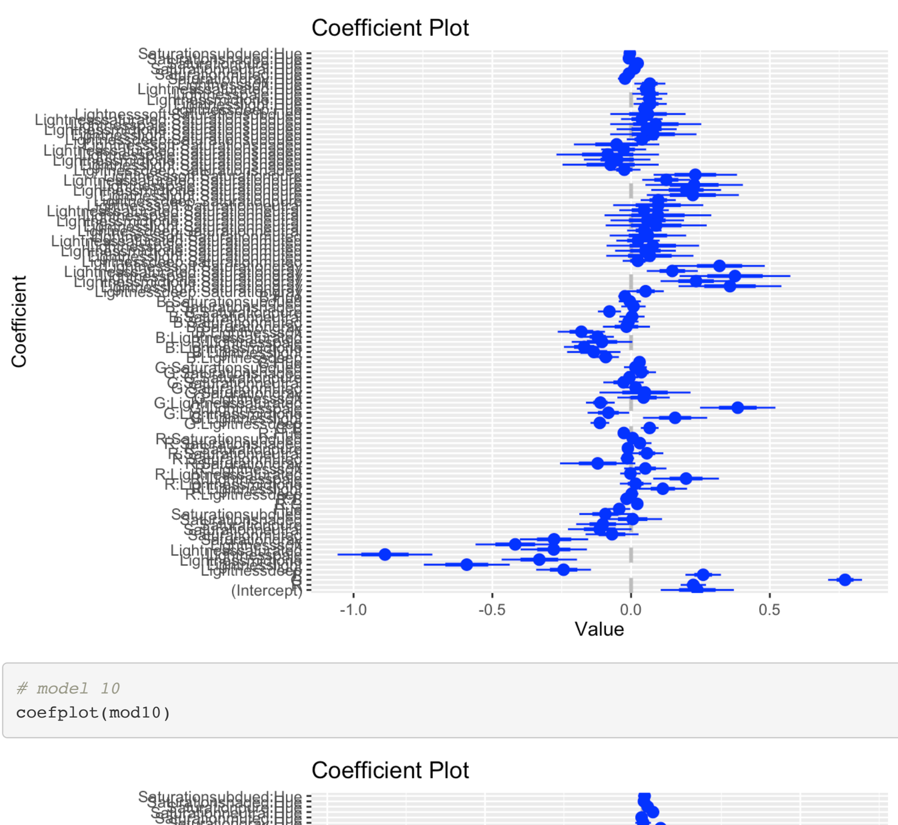
Which of the 10 models is the best? Ans. model 7 ('mod07') has the lowest AIC value and hence is the best model according to my performance metric.

##(3) Visualize the coefficient summaries for your top 3 models

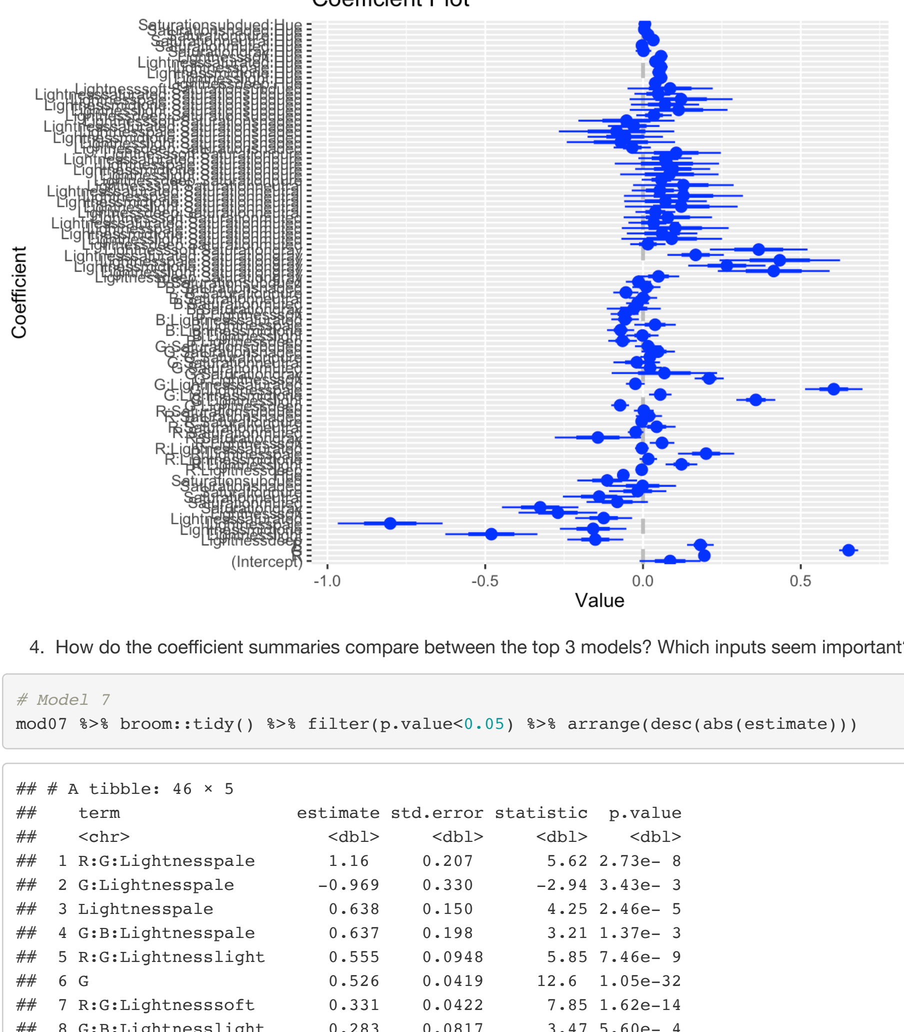
```
#model 7
coefplot(mod07)
```




```
#model 9
coefplot(mod09)
```




```
# model 10
coefplot(mod10)
```




4. How do the coefficient summaries compare between the top 3 models? Which inputs seem important?

```
# Model 7
mod07 %>% broom::tidy() %>% filter(p.value<0.05) %>% arrange(desc(abs(estimate)))
```

```
## # A tibble: 46 × 5
##    term                 estimate std.error statistic  p.value
##    <chr>                   <dbl>     <dbl>     <dbl>    <dbl>
##  1 R:G:Lightnesspale       1.16     0.207       5.62 2.73e- 8
##  2 G:Lightnesspale        -0.969    0.330      -2.94 3.43e- 3
##  3 Lightnesspale           0.638    0.150       4.25 2.46e- 5
##  4 G:B:Lightnesspale       0.637    0.198       3.21 1.37e- 3
##  5 R:G:Lightnesslight      0.555    0.0948      5.85 7.46e- 9
##  6 G                       0.526    0.0419     12.6  1.05e-32
##  7 R:G:Lightnesssoft       0.331    0.0422      7.85 1.62e-14
##  8 G:B:Lightnesslight      0.283    0.0817      3.47 5.60e- 4
##  9 G:B:Lightnesssoft       0.216    0.0451      4.78 2.14e- 6
## 10 R:G:Lightnessmidtone    0.205    0.0220      9.33 1.39e-19
## # ℹ 36 more rows
```

```
# Model 9
mod09 %>% broom::tidy() %>% filter(p.value<0.05) %>% arrange(desc(abs(estimate)))
```

```
## # A tibble: 50 × 5
##    term                              estimate std.error statistic   p.value
##    <chr>                                <dbl>     <dbl>     <dbl>     <dbl>
##  1 Lightnesspale                       -0.887    0.0854    -10.4  1.18e- 23
##  2 G                                    0.771    0.0306     25.2  1.83e-101
##  3 Lightnesslight                      -0.592    0.0774     -7.65 6.45e- 14
##  4 Lightnesssoft                       -0.418    0.0716     -5.84 8.08e-   9
##  5 G:Lightnesspale                      0.385    0.0679      5.67 2.12e-   8
##  6 Lightnesspale:Saturationgray         0.374    0.0999      3.75 1.91e-   4
##  7 Lightnesslight:Saturationgray        0.356    0.0927      3.84 1.31e-   4
##  8 Lightnessmidtone                    -0.331    0.0679     -4.87 1.36e-   6
##  9 Lightnesssoft:Saturationgray         0.319    0.0815      3.91 1.00e-   4
## 10 Lightnesssaturated                  -0.279    0.0598     -4.66 3.72e-   6
## # ℹ 40 more rows
```

```
# Model 10
mod10 %>% broom::tidy() %>% filter(p.value<0.05) %>% arrange(desc(abs(estimate)))
```

```
## # A tibble: 39 × 5
##    term                              estimate std.error statistic   p.value
##    <chr>                                <dbl>     <dbl>     <dbl>     <dbl>
##  1 Lightnesspale                       -0.801    0.0831     -9.64 8.43e- 21
##  2 G                                    0.652    0.0151     43.1  2.26e-203
##  3 G:Lightnesspale                      0.605    0.0455     13.3  3.21e- 36
##  4 Lightnesslight                      -0.481    0.0730     -6.59 8.60e- 11
##  5 Lightnesspale:Saturationgray         0.433    0.0956      4.54 6.72e-   6
##  6 Lightnesslight:Saturationgray        0.414    0.0886      4.68 3.47e-   6
##  7 Lightnesssoft:Saturationgray         0.367    0.0774      4.74 2.56e-   6
##  8 G:Lightnesslight                     0.358    0.0308     11.6  1.10e- 28
##  9 Saturationgray                      -0.326    0.0606     -5.38 1.02e-   7
## 10 Lightnesssoft                       -0.270    0.0624     -4.33 1.73e-   5
## # ℹ 29 more rows
```

Looking at the coefficient summaries we see that all the inputs continuous and categorical are important.