Carotenoid deposition simulation for common yellowthroat feather coloration.

Conor C. Taff

Purpose

The goal here is to demonstrate how different patterns of variation in i) the shape of structural white spectra, ii) the UV shift of the white spectra, iii) the overall reflectance of structural white, and iv) amount of carotenoid deposition generate different patterns of covariance in downstream metrics of coloration. A simulation is set up allowing me to vary each of these characteristics separately or together to demonstrate the effects.

Simulation

The simulation here is simple and involves four stages that represent the four possible sources of variation in ultimate color measurements. These four transformations are employed sequentially in the order presented to a set number of randomly drawn individuals. The amount of variation in each process can be set to examine the consequences of variation in each process alone or any combination of processes.

Shape of spectral curve

To generate the structural shape of the white feather spectra, I start with the average white reflectance curve in Shawkey & Hill (Shawkey & Hill, 2005, fig. 1A 'before treatment' and 'after treatment' can be chosen). I extracted these curves using WebPlotDigitizer and interpolated points for every 1 nm wavelength by fitting a smoothed regression. This results in the 'typical' white color that is used as the default when structural color does not vary (figure 1).

To generate individuals that differ in their underlying structural shape, I multiplied this entire spectra by a percentage chosen from a random normal distribution with mean of 100% and SD of 10%. Multiplying by a percentage has the effect of both raising or lowering the overall curve and also of changing the overall percentage of reflectance that is represented in

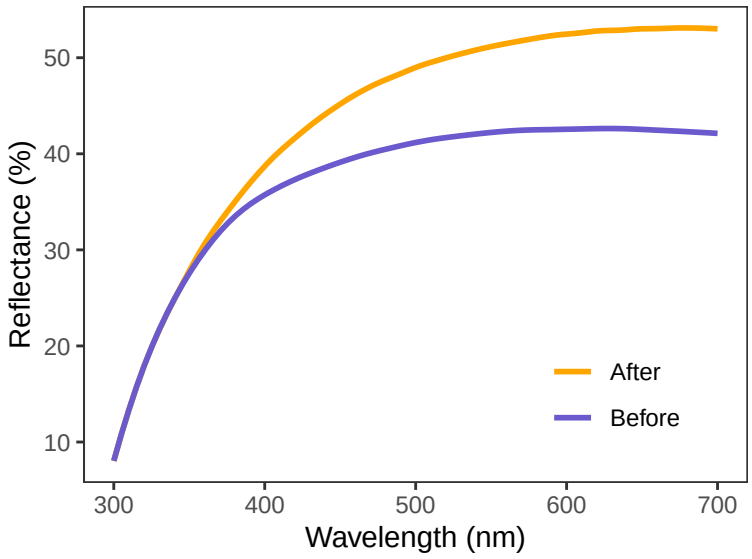


Figure 1: Structural white curves taken from Shawkey and Hill 2005 Figure 1A before and after feather bleaching treatments.

different wavelengths. The different spectral shapes generated by this sampling process can bee seen in Figure 2.

Amount of UV in spectral curve

I'm not sure this is really a distinct mechanism from the shape section above, but we also discussed the fact that the same curve could be shifted to have more UV reflectance. I've implemented this source of variation in the following way. First, for each wavelength across the spectra I calculate one minus the percentage of the maximum reflectance for that wavelength (so low reflectance areas in UV end up with high numbers). Second, I draw a number from a random normal distribution with mean of 0 (no distortion) and SD of 0.15. I multiply this fixed value by the vector calculated above and add it to the original spectra. This results in a shifted towards UV peak for positive draws and a shifted away from UV peak for negative draws (Figure 3).

Overall level of reflectance

Here the goal was to change the overall level of reflectance while maintaining the shape of the underlying spectral shape. To accomplish this, I added or subtracted a fixed percentage of reflectance to the entire curve. When generating between-individual differences in this component, I chose a fixed value to add or subtract from a random normal distribution with mean of 0 and SD of 10%. In some cases, this subtraction could lead to negative reflectance values, so after the transformation I set any reflectance values less than zero to zero. This can result in slightly different shapes to curves at the low end, but is necessary to create realistic curves. Clipping can be avoided by starting with a curve that has a higher initial reflectance

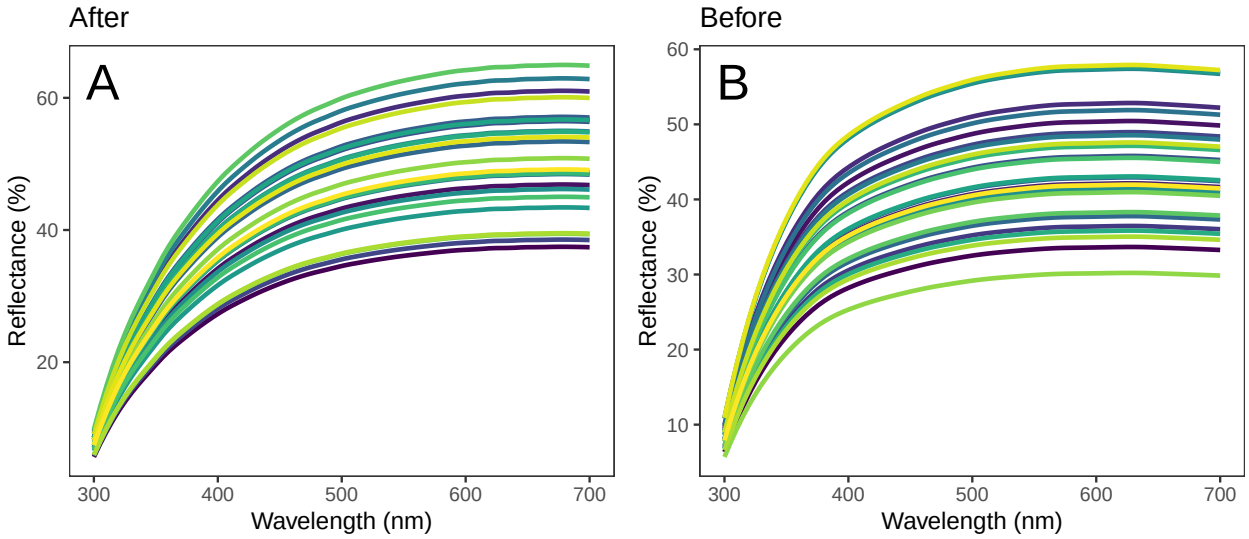


Figure 2: Sampled spectral shapes starting from the base curves from Shawkey and Hill 2005 and multiplying by a random normal sample of percentages with mean of 100% and SD of 15%. Panel A starts with the base curve for 'after bleaching' and panel B starts with the base curve for 'before bleaching.'

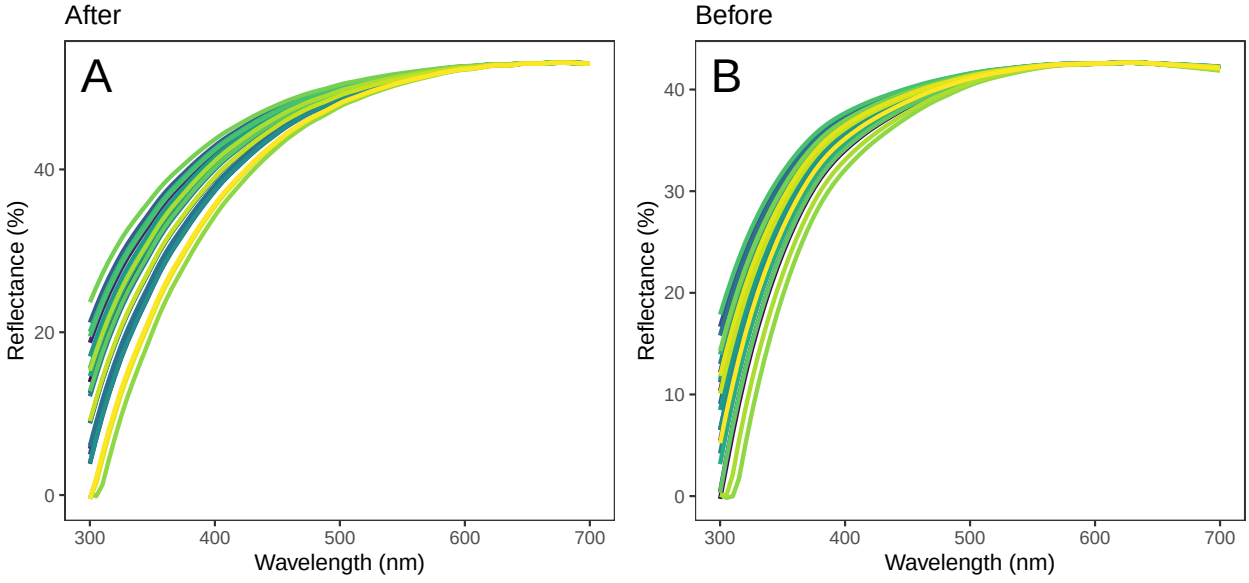


Figure 3: Sampled spectral shapes with shifts indicating more or less relative UV while max reflectance stays the same. Panels show the process applied to the 'after' (A) or 'before' (B) spectra from Shawkey and Hill 2005.

or changing the random addition numbers (e.g., use a mean of 10 and SD of 10%). Holding everything else constant, this process results in a population of curves that look like Figure 4.

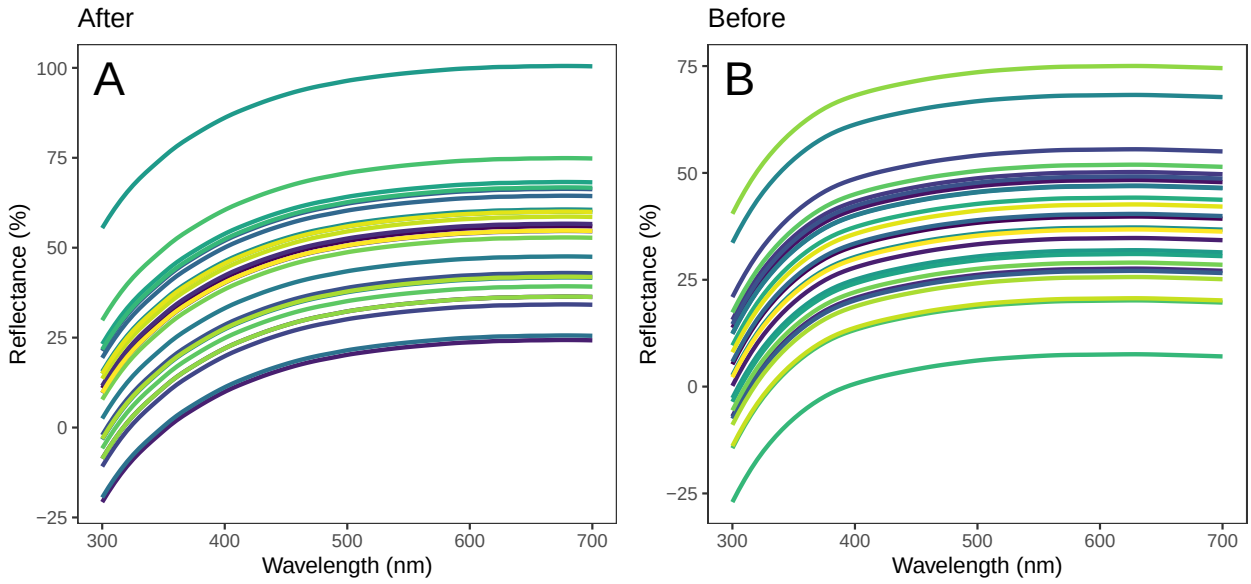


Figure 4: Sampled spectral shapes starting from the base curves from Shawkey and Hill 2005 and adding a random normal sample of percentages with mean of 0 and SD of 15%. Panel A starts with the base curve for 'after bleaching' and panel B starts with the base curve for 'before bleaching.'

Amount of carotenoid deposition

Last, we can take the curves generated by the two processes above and then layer carotenoid deposition on top of them. In common yellowthroats, yellow color is generated solely by lutein. I found a plot of the absorption spectrum of lutein online (there are many published spectra) and digitized it using WebPlotDigitizer as described above so that I ended up with an absorption value for everywavelength from 300 to 700 nm (Figure 5).

The absorbance spectrum that I used was calibrated such that the maximum absorbance at 452 nm was ~96%. I generated samples of different amounts of lutein deposition by multiplying this lutein spectrum by number chosen from a random normal distribution with mean of 0.8 and SD of 0.1. Spectral curves with carotenoid absorption were then calculated by the formula: white - white * lutein absorption. When underlying white coloration and reflectance is held constant, this process resulted in a population of spectral shapes as shown in Figure 6. Note that there is no assumption here about the actual amount of carotenoid deposition represented by these curves; rather, it is designed to create realistic spectral shapes and variation without considering the detailed physical properties. As above, I clipped any negative reflectance values to 0.

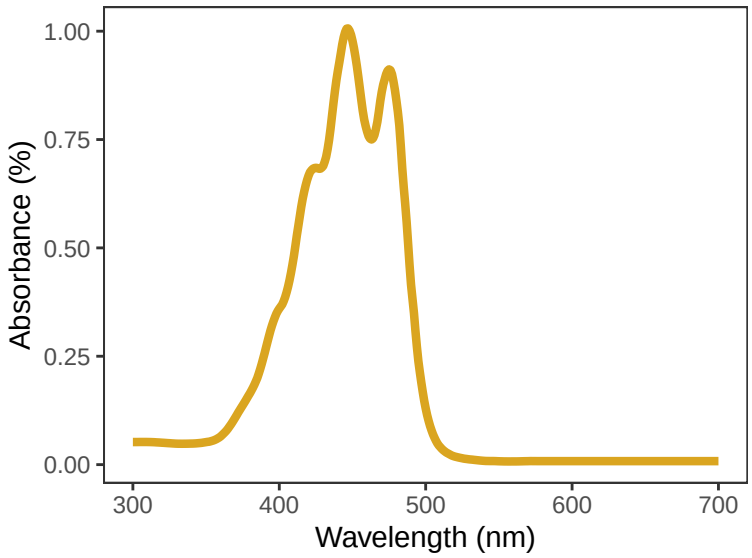


Figure 5: Absorbance spectrum of lutein.

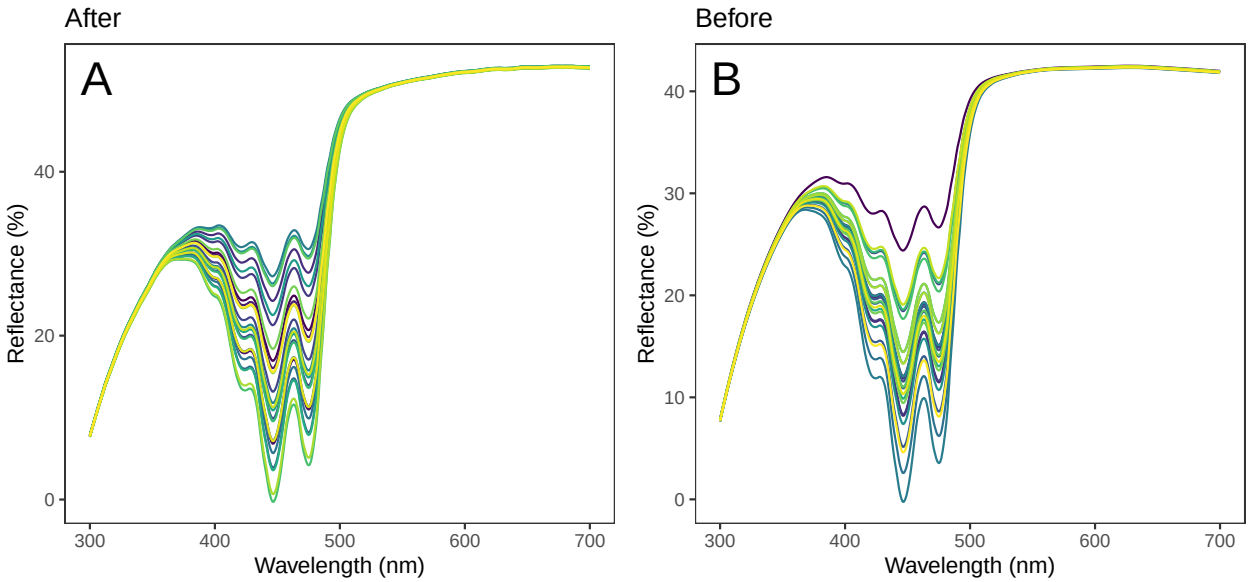


Figure 6: Variation in lutein deposition added to the basic structure for 'after' (A) or before (B) white spectra.

Calculating color metrics

The three step process described above can generate an arbitrary number of spectra that vary in any way desired (i.e., we can specify variation in any one, two, or all three of the contributing processes). After generating a sample of spectra as desired, I calculated a variety of downstream color metrics that summarize certain properties of the color spectra. For now I've calculated metrics exactly as we describe in Freeman-Gallant et al. ((2010)). This includes total brightness, uv brightness, yellow brightness, uv saturation, yellow saturation, and Ccar. The final result is a table of coloration metrics for a population of a defined sample size and set sources of variation in overall color that can be used to ask how each metric covaries.

Function

The description above describes how each step in the simulation works, but the entire set of steps is contained in a single function called `lutein_sim`. The function takes arguments for the sample size desired, which initial curve to start with, and the mean and standard deviation settings for each source of variation described above. The output includes i) the full spectra for each individual, ii) the summary table of color metrics, iii) a plot with the full spectra for each individual, and iv) a correlation plot of the key pair-wise color metrics. Each of these four outputs can be accessed from the list object returned by the function. The entire set of steps describe above is run in a single line of code. Because this is drawing from a random distribution, remember to set a seed if you want identical results to be produced each time. For now I just have the function running as a standalone script, but I can set it up to install via GitHub if we want. You'll need to have the following packages already installed in your R environment: `tidyverse`; `ggplot2`; `viridis`; `GGally`. Once those are installed, you can just run the single code block from the repository to load the function and then have access to the function to generate new variations.

```
lutein_sim <- function(n = 25, white_mu = 1, white_sd = 0,
                       uv_mu = 0, uv_sd = 0,
                       reflect_mu = 0, reflect_sd = 0,
                       lut_mu = 0.8, lut_sd = 0,
                       base_shape = "after"){
  # Convert uv and reflect sds to *100 since the percentages are on whole number scale
    uv_sd2 <- uv_sd * 100
    reflect_sd2 <- reflect_sd * 100

  # Set up for using before or after spectra
    if(base_shape == "after"){bs <- "after"}
    if(base_shape == "before"){bs <- "before"}

  # Read in basic shape data. This requires the file to be present. Should really be h
      # load shawkey figure data
```

```
      shaw <- read.delim(here::here("1_raw_data", "shawkey_2005_fig1.txt"))
      shaw <- subset(shaw, shaw$type == bs)

    # Fit and predict from loess smoothed line to get reflectance at each nm wavelen
        r <- data.frame(wavelength = seq(300, 700, 1))
        lo_fit <- loess(reflectance ~ wavelength, data = shaw, span = 0.1, surface = "
        lo_pred <- data.frame(wavelength = r$wavelength, reflectance = predict(lo_fit,

# Read in lutein shape data. This also requires file and should be hardcoded in.
      # load lutein data that I copied from an absorbance spectra posted online
          lut <- read.delim(here::here("1_raw_data", "lutein.txt"))

      # In order to make it flat outside of range I'm adding more points that extend
          lut_begin <- data.frame(molecule = "lutein",
                                  wavelength = seq(280, 310, 5),
                                  absorbance = 0.05209852)
          lut_end <- data.frame(molecule = "lutein",
                                wavelength = seq(565, 800, 5),
                                absorbance = 0.008027346)
          lut <- rbind(lut, lut_begin, lut_end)

      # Fit loess smooth and interpolate value at each 1 nm
          lu_lo <- loess(absorbance ~ wavelength, data = lut, span = 0.05, surface = "
          lu_pred <- data.frame(wavelength = r$wavelength, absorbance = predict(lu_lo,

# 1. Add in white variation
      # multiply by percentages
        pop_p <- rnorm(n, white_mu, white_sd)
        pop_r <- as.data.frame(matrix(ncol = length(pop_p), nrow = nrow(lo_pred)))
        for(i in 1:n){
          pop_r[, i] <- lo_pred$reflectance * rep(pop_p[i], nrow(lo_pred))
        }

# 2. Add in uv shift variation
     # multiply by uv shift
        for(i in 1:n){
          temp <- (1 - pop_r[, i] / rep(max(pop_r[, i]), nrow(lo_pred)))
          draw <- rnorm(1, uv_mu, uv_sd2)
          new <- pop_r[, i] + rep(draw, nrow(lo_pred)) * temp
          new <- pmax(new, 0)
          pop_r[, i] <- new
        }

# 3. Add in overall reflectance variation
```

```
    pop_p2 <- rnorm(n, reflect_mu, reflect_sd2)
    for(i in 1:n){
      pop_r[, i] <- pop_r[, i] + rep(pop_p2[i], nrow(lo_pred))
    }

# 4. Add in lutein deposition variation
    # multiply by lutein
        for(i in 1:n){
          draw <- rnorm(1, lut_mu, lut_sd)
          if(draw > 1){draw <- 1}
          temp <- draw * lu_pred$absorbance
          pop_r[, i] <- pop_r[, i] - (pop_r[, i] * temp)
        }
        pop_r$wavelength <- lo_pred$wavelength
        pop_r2 <- pivot_longer(pop_r, cols = starts_with("V"), names_to = "id", values

## Make figure
    p_spec <- ggplot(data = pop_r2, mapping = aes(x = wavelength, y = reflectance, col
        #geom_smooth(se = FALSE, span = 0.05) +
        geom_line() +
        scale_color_viridis(discrete = TRUE) +
        theme_bw() +
        theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
        guides(color = "none") +
        xlab("Wavelength (nm)") +
        ylab("Reflectance (%)")

## make dataframe of color metrics
    color_met <- data.frame(id = unique(pop_r2$id),
                            b_tot = rep(NA, n),
                            b_uv = rep(NA, n),
                            b_yel = rep(NA, n),
                            sat_uv = rep(NA, n),
                            sat_yel = rep(NA, n),
                            ccar = rep(NA, n))
    for(k in 1:nrow(color_met)){
      color_met$b_tot[k] <- mean(subset(pop_r2$reflectance, pop_r2$id == color_met$id[
                                        pop_r2$wavelength < 701 &
                                        pop_r2$wavelength > 319))
      color_met$b_uv[k] <- mean(subset(pop_r2$reflectance, pop_r2$id == color_met$id[k
                                        pop_r2$wavelength < 401 &
                                        pop_r2$wavelength > 319))
      color_met$b_yel[k] <- mean(subset(pop_r2$reflectance, pop_r2$id == color_met$id[
                                        pop_r2$wavelength < 626 &
```

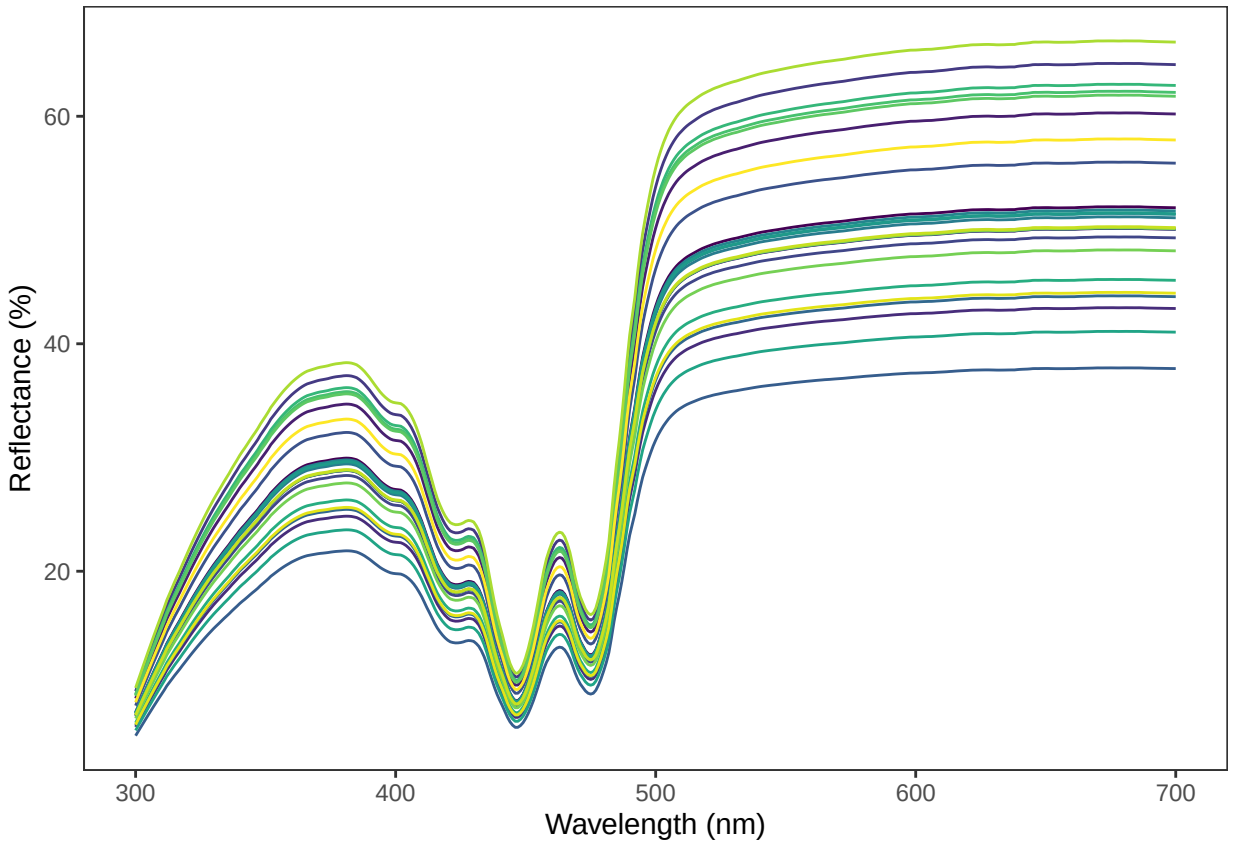
```
                                          pop_r2$wavelength > 549))
      color_met$sat_uv[k] <- sum(subset(pop_r2$reflectance, pop_r2$id == color_met$id[
                                        pop_r2$wavelength < 401 & pop_r2$wavelength > 3
                          sum(subset(pop_r2$reflectance, pop_r2$id == color_met$id[k]
                                        pop_r2$wavelength < 701 & pop_r2$wavelength > 3
      color_met$sat_yel[k] <- sum(subset(pop_r2$reflectance, pop_r2$id == color_met$id
                                        pop_r2$wavelength < 626 & pop_r2$wavelength > 5
                          sum(subset(pop_r2$reflectance, pop_r2$id == color_met$id[k]
                                        pop_r2$wavelength < 701 & pop_r2$wavelength > 3
      color_met$ccar[k] <- (subset(pop_r2$reflectance, pop_r2$id == color_met$id[k] &
                                     pop_r2$wavelength == 700) -
                            subset(pop_r2$reflectance, pop_r2$id == color_met$id[k] &
                                     pop_r2$wavelength == 450)) /
                            subset(pop_r2$reflectance, pop_r2$id == color_met$id[k] &
                                     pop_r2$wavelength == 700)
    }

  ## Add little function for smoothed lines
    my_fn <- function(data, mapping, method="loess", ...){
    p <- ggplot(data = data, mapping = mapping) +
    geom_point() +
    geom_smooth(color = "coral3", fill = "coral3", alpha = 0.3, method=method, ...)
    p
   }

  ## Make a pairwise plot of relationships
    p_cors <- GGally::ggpairs(color_met, columns = 2:7, lower = list(continuous = wrap
      theme_bw() +
      theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())

  ## Put the output together
    lut_out <- list(
      wide_specs <- pop_r,
      long_specs <- pop_r2,
      color_mets <- color_met,
      spec_plot <- p_spec,
      cor_plot <- p_cors
    )

    return(lut_out)

}
```

Using the simulation

This is the easy part now that the function is written. We can compare the covariation patterns for a variety of different scenarios. Here I'm just changing the parameters in the functions and then showing the default plots that are produced by the function with everything else held the same.

Variation only in structural shape

```
t <- lutein_sim(white_sd = 0.15)
t[[4]]
```

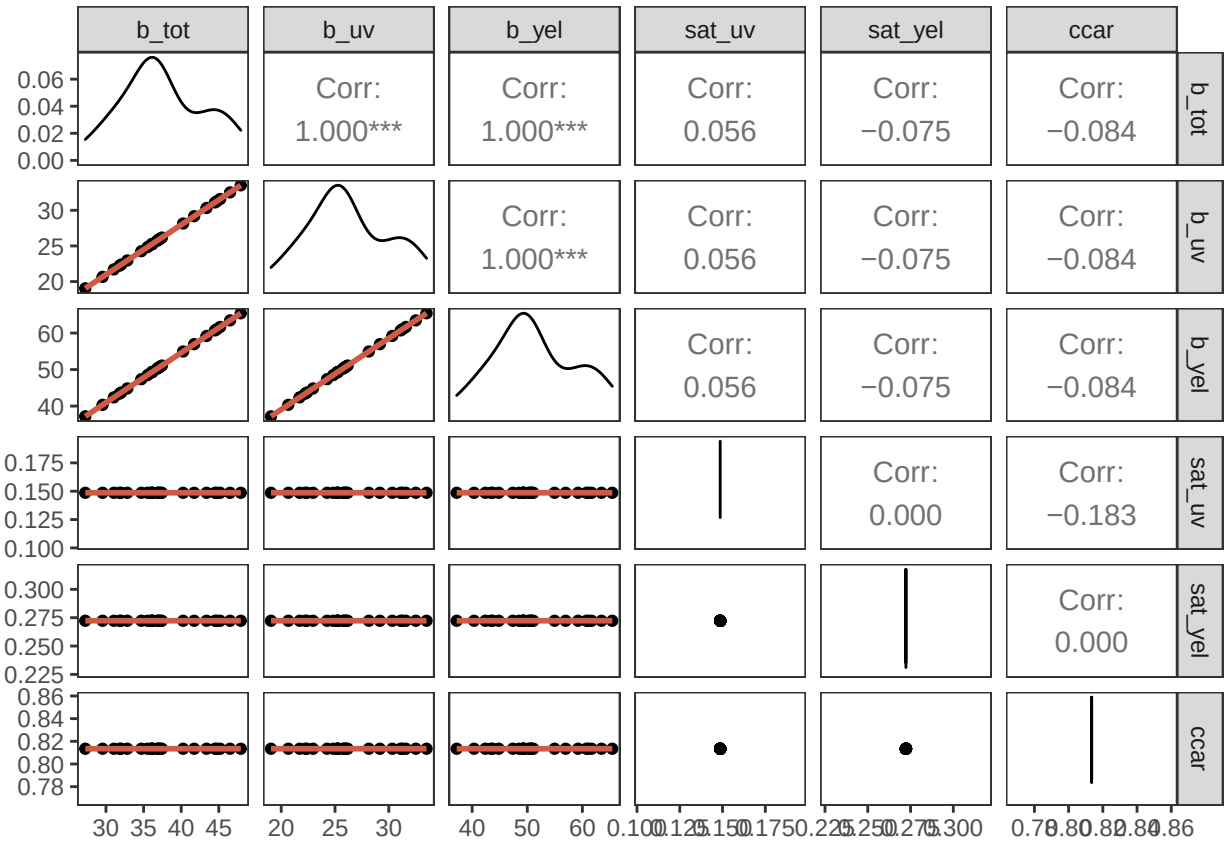



```
t[[5]]
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
```

```
## collapsing to unique 'x' values

## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'

## Warning in predict.lm(model, newdata = new_data_frame(list(x = xseq)), se.fit =
## se, : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = new_data_frame(list(x = xseq)), se.fit =
## se, : collapsing to unique 'x' values

## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'

## Warning in predict.lm(model, newdata = new_data_frame(list(x = xseq)), se.fit =
## se, : prediction from a rank-deficient fit may be misleading

## `geom_smooth()` using formula 'y ~ x'

## Warning in predict.lm(model, newdata = new_data_frame(list(x = xseq)), se.fit =
## se, : prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(model, newdata = new_data_frame(list(x = xseq)), se.fit =
## se, : collapsing to unique 'x' values
```



Variation only in UV shift

Variation only reflectance

Variation only in lutein deposition

Freeman-Gallant, C. R., Taff, C. C., Morin, D. F., Dunn, P. O., Whittingham, L. A., & Tsang, S. M. (2010). Sexual selection, multiple male ornaments, and age-and condition-dependent signaling in the common yellowthroat. *Evolution: International Journal of Organic Evolution*, *64*(4), 1007–1017.

Shawkey, M. D., & Hill, G. E. (2005). Carotenoids need structural colours to shine. *Biology Letters*, *1*(2), 121–124.