

H1-T1: zona H1 (hybrid) del transecto 1.

José Carlos Del Valle

11/11/2021

Índice

1. Introducción.	1
2. Importando y limpiando el dataset.	2
3. Obteniendo los promedios de las dos mediciones hechas en las partes basal y apical de la flor. . .	3
3.1. Primer test con un dataset reducido.	3
3.2. Opción 1: para hacerlo de la forma que lo he hecho anteriormente (si no comprendes algún paso, arriba se explica paso a paso para qué se hace cada cosa).	6
3.2. Opción 2: para hacerlo más fino, tendría que crear un loop como el que describo abajo, pero no me termina de correr bien No lo incluyo en los chunks porque no me interesa correrlo ahora. Lo dejo como texto normal por si en un futuro soy capaz de solucionarlo.	10
4. Cálculo de variables colorimétricas.	10
5. Modelos de visión.	12
5.1. Modelo de visión de las ABEJAS.	12
5.2. Modelo de visión de las MARIPOSAS.	15
5.3. Modelo de visión de las MOSCAS.	18
5.4. Modelo de visión de los ESCARABAJOS.	19
6. Calculando los contrastes cromáticos (distancia al centro)	23
7. Calculando el contraste acromático en el modelo de visión de las abejas.	24

1. Introducción.

En este script voy a analizar cómo ven las flores de *Erysimum merxmulleri* y *E. lagascae* los principales grupos de polinizadores de la zona: dípteros, coleópteros, lepidópteros e himenópteros.

Este script será un poco más largo porque es la primera zona con la que trabajo y he hecho varios test que no tengo que repetir en los siguientes scripts. Cada zona la analizaré en scripts separados porque si no va a ser demasiado largo y un poco caótico

2. Importando y limpiando el dataset.

Los datos de reflectancia de la zona T1-H1 están en la carpeta “T1_Hibrida_101-178”. En esa carpeta encontraremos dos mediciones: una hecha para la parte distal y otra para la parte apical. Nosotros utilizaremos el promedio de ambas mediciones, obteniendo tan solo una medición por flor.

Lo primero que haremos será cargar la librería *pavo*. Luego, cargaremos los datos (seleccionar el directorio apropiado en cada caso) y crearemos el objeto **plotH1**.

```
library(pavo)
H1 <- "/Users/delValle/Dropbox/UGR/T1_Hibrida_101-178"
plotH1<-getspec(H1)
```

Ahora hay que pulir un poco los espectros en bruto que se obtienen del espectrofotómetro. Para ello utilizaremos las funciones “procspec” contenidas en el paquete *pavo*.

Lo primero que voy a hacer es eliminar posibles valores negativos de los espectros añadiendo el valor absoluto del valor negativo más alto a todo el espectro. A continuación, suavizo un poco los espectros.

```
plotH1.fix <- procspec(plotH1, fixneg = "addmin")

## processing options applied:
## Negative value correction: added min to all reflectance

plotH1.sm <- procspec(plotH1.fix, opt = "smooth", span = 0.2)

## processing options applied:
## smoothing spectra with a span of 0.2

is.rspect(plotH1.sm)

## [1] TRUE
```

Ahora voy a plotear los dos espectros del individuo 101 (101a = parte apical del pétalo, 101b = parte basal del pétalo).

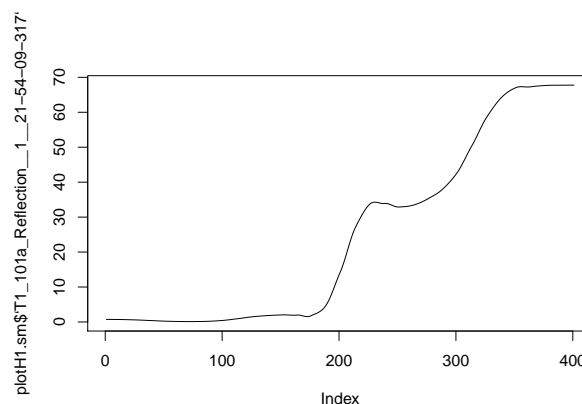


Figura 1: Parte apical de la flor del individuo 101.

Después de ver los espectros, no tengo demasiado claro si es buena idea mezclar las mediciones. Difieren demasiado entre ellos. De todas formas, voy a continuar así y calcularé el promedio de ambos espectros por individuo.

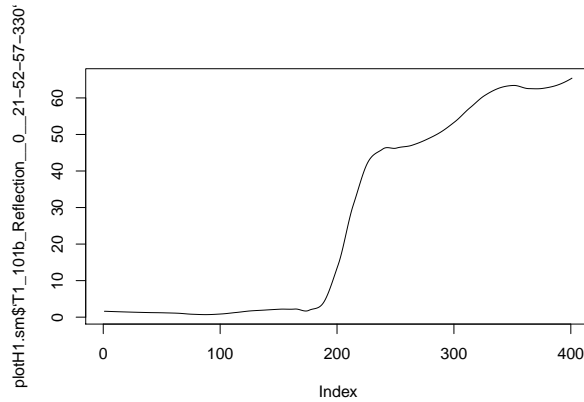


Figura 2: Parte basal de la flor del individuo 101.

3. Obteniendo los promedios de las dos mediciones hechas en las partes basal y apical de la flor.

3.1. Primer test con un dataset reducido.

Para promediar los espectros de las partes basal y apical de una flor, hay que calcular el promedio de dos columnas contiguas en *plotH1.sm* de cada muestra. Voy a hacer primero un pequeño test de forma manual utilizando varias columnas continuas, lo que me servirá como referencia más adelante para verificar que lo que estoy haciendo está bien

```
plotH1.sm_manual <- plotH1.sm$w1
plotH1.sm_manual <- as.data.frame(plotH1.sm_manual)
plotH1.sm_manual$T1_101 <- apply(plotH1.sm[,c(2,3)], 1, mean, na.rm = TRUE)
plotH1.sm_manual$T1_102 <- apply(plotH1.sm[,c(4,5)], 1, mean, na.rm = TRUE)
plotH1.sm_manual$T1_103 <- apply(plotH1.sm[,c(6,7)], 1, mean, na.rm = TRUE)
plotH1.sm_manual$T1_104 <- apply(plotH1.sm[,c(8,9)], 1, mean, na.rm = TRUE)
```

Lo primero que he hecho ha sido crear una primera columna con las longitudes de onda. He transformado el objeto *plotH1.sm_manual* a data frame y a continuación he ido añadiendo columnas (una por individuo) con los promedios de ambas mediciones. Con una simple comprobación, vemos que el objeto *plotH1.sm_manual* es reconocido correctamente como un objeto *rspec* (formato de *pavo*):

```
plotH1.sm_manual <- as.rspec(plotH1.sm_manual)
```

```
## wavelengths found in column 1
```

```
is.rspec(plotH1.sm_manual)
```

```
## [1] TRUE
```

Estos serían los cinco primeros valores que he obtenido para las muestras 101 a 104:

```
head(plotH1.sm_manual, n=5)
```

```
##      w1    T1_101    T1_102    T1_103    T1_104
## 1 300 1.177922 1.302502 1.022938 0.6922963
## 2 301 1.171390 1.296603 1.026394 0.7040755
## 3 302 1.164755 1.290602 1.029493 0.7153003
## 4 303 1.158008 1.284509 1.032242 0.7259692
## 5 304 1.151145 1.278331 1.034647 0.7360804
```

Como es inviable hacerlo manualmente, voy a crear un loop para que automáticamente me calcule el promedio cada dos columnas. Primero voy a correr un primer test con los promedios desde la columna 2 a la 11:

```
test0 <- plotH1.sm$w1
test0 <- as.data.frame(test0)
colnames(test0) <- c("w1")
head(test0)
```

```
##      w1
## 1 300
## 2 301
## 3 302
## 4 303
## 5 304
## 6 305
```

```
for(i in 2:11) {
  test0[, i] <- apply(plotH1.sm[, c(i, i+1)], 1, mean, na.rm=TRUE)
}
```

Al comparar los resultados con los obtenidos de forma manual (ver objeto *plotH1.sm_manual*), veo que me ha sumado las columnas contiguas (ej: columna 2 + columna 3, columna 3 + columna 4, columna 4 + columna 5):

```
test0[1:5, 1:8]
```

```
##      w1      V2      V3      V4      V5      V6      V7      V8
## 1 300 1.177922 1.606348 1.302502 1.052695 1.022938 0.8655964 0.6922963
## 2 301 1.171390 1.594218 1.296603 1.053359 1.026394 0.8749099 0.7040755
## 3 302 1.164755 1.582198 1.290602 1.053727 1.029493 0.8837075 0.7153003
## 4 303 1.158008 1.570291 1.284509 1.053809 1.032242 0.8920017 0.7259692
## 5 304 1.151145 1.558501 1.278331 1.053614 1.034647 0.8998051 0.7360804
```

```
plotH1.sm_manual[1:5, 1:5]
```

```
##      w1    T1_101    T1_102    T1_103    T1_104
## 1 300 1.177922 1.302502 1.022938 0.6922963
## 2 301 1.171390 1.296603 1.026394 0.7040755
## 3 302 1.164755 1.290602 1.029493 0.7153003
## 4 303 1.158008 1.284509 1.032242 0.7259692
## 5 304 1.151145 1.278331 1.034647 0.7360804
```

Al sumar las columnas contiguas, únicamente las columnas pares (V2, V4, V6, etc.) son las correctas. Es decir, V2 equivale a T1_101, V4 equivale a T1_102, V6 equivale a T1_103 y V8 equivale a T1_104.

Voy a probar a eliminar las columnas impares y a retener las pares, pero antes tengo que quitar la primera columna, que es la que tiene las longitudes de onda.

```
test1 <- test0[ , -c(1)]
col_odd <- seq_len(ncol(test1)) %% 2
test1 <- test1[ , col_odd == 1]
rm(col_odd)
```

Ahora que solo tengo las columnas pares (las que tienen los promedios que estoy buscando), vuelvo a añadir la columna con las longitudes de onda (columna *wl*) que había eliminado. Ahora sí que me coinciden los resultados:

```
test1$wl <- test0$wl
library(dplyr)
test1 <- test1 %>%
  select(wl, everything())
```

Ahora sí que me coinciden los resultados:

```
plotH1.sm_manual[1:5, 1:5]
```

```
##      wl    T1_101    T1_102    T1_103    T1_104
## 1 300 1.177922 1.302502 1.022938 0.6922963
## 2 301 1.171390 1.296603 1.026394 0.7040755
## 3 302 1.164755 1.290602 1.029493 0.7153003
## 4 303 1.158008 1.284509 1.032242 0.7259692
## 5 304 1.151145 1.278331 1.034647 0.7360804
```

```
test1[1:5, 1:5]
```

```
##      wl      V2      V4      V6      V8
## 1 300 1.177922 1.302502 1.022938 0.6922963
## 2 301 1.171390 1.296603 1.026394 0.7040755
## 3 302 1.164755 1.290602 1.029493 0.7153003
## 4 303 1.158008 1.284509 1.032242 0.7259692
## 5 304 1.151145 1.278331 1.034647 0.7360804
```

```
rm(test0)
```

Ahora solo me faltaría cambiar el nombre de las variables y ya estaría listo:

```
x0 <- c("wl")
x1 <- 101:125
x2 <- 127:150
x3 <- 152:158
x4 <- 160:178
```

Como faltan algunas muestras (ej: 126), las secuencias no van del 101 al 178 directamente, sino que he creado objetos *x* con una numeración acorde al nombre de las muestras. Solo tengo que unificar estos objetos y añadirle al nombre de cada individuo (i.e. el nombre de cada columna) el prefijo “T1_” correspondiente al transecto

```
x_sum <- c(x1, x2, x3, x4)
x_sum <- paste("T1", x_sum, sep="_")
head(x_sum)
```

```
## [1] "T1_101" "T1_102" "T1_103" "T1_104" "T1_105" "T1_106"
```

Todo esto que he hecho serviría para tener los nombres de todas las muestras, pero para este pequeño test solo necesito los nombres desde el T_101 hasta el T1_105:

```
x_sum_test1 <- c(x_sum[1:5])
names_test1 <- c(x0,x_sum_test1)
names_test1
```

```
## [1] "w1"      "T1_101" "T1_102" "T1_103" "T1_104" "T1_105"
```

Ya tengo listo el vector con los nombres de las columnas del dataframe *test1*

```
colnames(test1) <- names_test1
head(test1, n=5)
```

```
##      w1    T1_101    T1_102    T1_103    T1_104    T1_105
## 1 300 1.177922 1.302502 1.022938 0.6922963 1.303106
## 2 301 1.171390 1.296603 1.026394 0.7040755 1.315932
## 3 302 1.164755 1.290602 1.029493 0.7153003 1.328119
## 4 303 1.158008 1.284509 1.032242 0.7259692 1.339655
## 5 304 1.151145 1.278331 1.034647 0.7360804 1.350528
```

Así es como consigo obtener los valores promedios de cada muestra (T1_101, T1_102, etc.) con su correspondiente nomenclatura; con esto podría trabajar ya y extrapolarlo al conjunto completo de datos.

Elimino todo aquello que ya no necesito antes de crear un data frame con todos los datos del transecto 1 de la zona híbrida H1.

```
rm(x1)
rm(x2)
rm(x3)
rm(x4)
rm(x0)
rm(x_sum)
rm(x_sum_test1)
rm(test1)
rm(names_test1)
rm(i)
```

3.2. Opción 1: para hacerlo de la forma que lo he hecho anteriormente (si no comprendes algún paso, arriba se explica paso a paso para qué se hace cada cosa).

```

plotH1_average <- plotH1.sm$w1
plotH1_average <- as.data.frame(plotH1_average)
colnames(plotH1_average) <- c("w1")
head(plotH1_average)

```

```

##      w1
## 1 300
## 2 301
## 3 302
## 4 303
## 5 304
## 6 305

```

```

for(i in 2:150) {
  plotH1_average[, i] <- apply(plotH1.sm[, c(i, i+1)], 1, mean, na.rm=TRUE)
}
plotH1_average[1:5,1:5]

```

```

##      w1      V2      V3      V4      V5
## 1 300 1.177922 1.606348 1.302502 1.052695
## 2 301 1.171390 1.594218 1.296603 1.053359
## 3 302 1.164755 1.582198 1.290602 1.053727
## 4 303 1.158008 1.570291 1.284509 1.053809
## 5 304 1.151145 1.558501 1.278331 1.053614

```

```

plotH1_average[1:5,145:150]

```

```

##      V145      V146      V147      V148      V149      V150
## 1 1.567030 3.450058 3.926525 2.167462 1.556768 0.7105921
## 2 1.564825 3.435370 3.912730 2.169318 1.557202 0.7169350
## 3 1.562205 3.420320 3.898625 2.170780 1.557302 0.7227992
## 4 1.559153 3.404901 3.884223 2.171847 1.557066 0.7281846
## 5 1.555651 3.389105 3.869535 2.172519 1.556487 0.7330916

```

```

plotH1_average2 <- plotH1_average[, -c(1)]
col_odd <- seq_len(ncol(plotH1_average2)) %% 2
plotH1_average2 <- plotH1_average2[, col_odd == 1]
rm(col_odd)

```

```

plotH1_average2$w1 <- plotH1_average$w1
library(dplyr)
plotH1_average2 <- plotH1_average2 %>%
  select(w1, everything())
plotH1_average2[1:4,1:5]

```

```

##      w1      V2      V4      V6      V8
## 1 300 1.177922 1.302502 1.022938 0.6922963
## 2 301 1.171390 1.296603 1.026394 0.7040755
## 3 302 1.164755 1.290602 1.029493 0.7153003
## 4 303 1.158008 1.284509 1.032242 0.7259692

```

```
x0 <- c("w1")
x1 <- 101:125
x2 <- 127:150
x3 <- 152:158
x4 <- 160:178
x_sum <- c(x1, x2, x3, x4)
x_sum <- paste("T1", x_sum, sep="_")
head(x_sum)
```

```
## [1] "T1_101" "T1_102" "T1_103" "T1_104" "T1_105" "T1_106"
```

```
names_H1 <- c(x0,x_sum)
names_H1
```

```
## [1] "w1"      "T1_101" "T1_102" "T1_103" "T1_104" "T1_105" "T1_106" "T1_107"
## [9] "T1_108" "T1_109" "T1_110" "T1_111" "T1_112" "T1_113" "T1_114" "T1_115"
## [17] "T1_116" "T1_117" "T1_118" "T1_119" "T1_120" "T1_121" "T1_122" "T1_123"
## [25] "T1_124" "T1_125" "T1_127" "T1_128" "T1_129" "T1_130" "T1_131" "T1_132"
## [33] "T1_133" "T1_134" "T1_135" "T1_136" "T1_137" "T1_138" "T1_139" "T1_140"
## [41] "T1_141" "T1_142" "T1_143" "T1_144" "T1_145" "T1_146" "T1_147" "T1_148"
## [49] "T1_149" "T1_150" "T1_152" "T1_153" "T1_154" "T1_155" "T1_156" "T1_157"
## [57] "T1_158" "T1_160" "T1_161" "T1_162" "T1_163" "T1_164" "T1_165" "T1_166"
## [65] "T1_167" "T1_168" "T1_169" "T1_170" "T1_171" "T1_172" "T1_173" "T1_174"
## [73] "T1_175" "T1_176" "T1_177" "T1_178"
```

```
colnames(plotH1_average2) <- names_H1
dim(plotH1_average2)
```

```
## [1] 401 76
```

Las dimensiones del objeto *plotH1_average2* son correctas. Tiene 401 filas (el encabezado + 400nm) y 76 columnas (w1 + 75 plantas).

```
head(plotH1_average2, n=5)
```

```
##      w1      T1_101      T1_102      T1_103      T1_104      T1_105      T1_106      T1_107      T1_108
## 1 300 1.177922 1.302502 1.022938 0.6922963 1.303106 1.111165 0.6805735 1.049051
## 2 301 1.171390 1.296603 1.026394 0.7040755 1.315932 1.126826 0.6778195 1.059807
## 3 302 1.164755 1.290602 1.029493 0.7153003 1.328119 1.141892 0.6750776 1.069984
## 4 303 1.158008 1.284509 1.032242 0.7259692 1.339655 1.156353 0.6723518 1.079580
## 5 304 1.151145 1.278331 1.034647 0.7360804 1.350528 1.170199 0.6696463 1.088595
##      T1_109      T1_110      T1_111      T1_112      T1_113      T1_114      T1_115      T1_116
## 1 0.6951392 0.9393013 0.8598050 1.193591 1.815666 1.543634 3.211835 2.066130
## 2 0.7030502 0.9477575 0.8596325 1.198291 1.813789 1.538387 3.195573 2.065815
## 3 0.7105574 0.9557559 0.8592762 1.202577 1.811728 1.533140 3.179492 2.065081
## 4 0.7176594 0.9632901 0.8587369 1.206450 1.809499 1.527897 3.163605 2.063917
## 5 0.7243551 0.9703539 0.8580152 1.209911 1.807115 1.522662 3.147928 2.062312
##      T1_117      T1_118      T1_119      T1_120      T1_121      T1_122      T1_123      T1_124
## 1 0.6163769 1.170082 0.5742310 0.9751733 1.128549 1.135917 1.202241 1.265450
## 2 0.6237399 1.164253 0.5735592 0.9753116 1.134356 1.138705 1.213282 1.252913
## 3 0.6307063 1.158455 0.5727570 0.9752445 1.139811 1.141374 1.223758 1.240575
```



```
## 4 0.6372709 1.152685 0.5718290 0.9749603 1.144916 1.143922 1.233674 1.228439
## 5 0.6434286 1.146945 0.5707795 0.9744474 1.149673 1.146351 1.243037 1.216506
##      T1_125   T1_127   T1_128   T1_129   T1_130   T1_131   T1_132   T1_133
## 1 0.7419735 1.971802 0.9656343 0.8741431 1.281989 1.415564 0.9023117 1.398973
## 2 0.7528430 1.971278 0.9606595 0.8840403 1.291282 1.407531 0.9089544 1.399520
## 3 0.7632717 1.970364 0.9555544 0.8934233 1.300068 1.399517 0.9151778 1.399876
## 4 0.7732540 1.969051 0.9503283 0.9022970 1.308355 1.391522 0.9209711 1.400039
## 5 0.7827843 1.967334 0.9449905 0.9106664 1.316148 1.383549 0.9263230 1.400008
##      T1_134   T1_135   T1_136   T1_137   T1_138   T1_139   T1_140   T1_141
## 1 2.116579 0.6915084 0.5532602 0.6905636 2.604496 2.531151 1.009144 0.8764612
## 2 2.108159 0.6988810 0.5670744 0.7017840 2.598581 2.519168 1.008649 0.8815723
## 3 2.099781 0.7058610 0.5803072 0.7125795 2.592402 2.507134 1.008017 0.8863190
## 4 2.091449 0.7124579 0.5929559 0.7229494 2.585962 2.495057 1.007241 0.8906949
## 5 2.083170 0.7186807 0.6050180 0.7328928 2.579264 2.482942 1.006313 0.8946936
##      T1_142   T1_143   T1_144   T1_145   T1_146   T1_147   T1_148   T1_149
## 1 0.8960364 1.485039 1.691036 0.4846095 1.914893 1.195676 1.229139 1.229258
## 2 0.9040141 1.486715 1.690417 0.4957352 1.912647 1.199611 1.235324 1.240328
## 3 0.9115291 1.488039 1.689754 0.5064356 1.910202 1.203182 1.241168 1.250989
## 4 0.9185812 1.489007 1.689054 0.5167225 1.907566 1.206378 1.246672 1.261230
## 5 0.9251700 1.489617 1.688322 0.5266076 1.904747 1.209186 1.251839 1.271037
##      T1_150   T1_152   T1_153   T1_154   T1_155   T1_156   T1_157   T1_158
## 1 1.935734 2.336883 1.270379 0.7658079 0.5256959 0.6141223 1.220436 1.498040
## 2 1.923138 2.326032 1.264073 0.7603560 0.5357716 0.6219688 1.219630 1.485949
## 3 1.910829 2.315192 1.257733 0.7548916 0.5453662 0.6295018 1.218729 1.474162
## 4 1.898810 2.304374 1.251352 0.7494088 0.5544733 0.6367201 1.217736 1.462684
## 5 1.887081 2.293584 1.244923 0.7439017 0.5630864 0.6436221 1.216653 1.451519
##      T1_160   T1_161   T1_162   T1_163   T1_164   T1_165   T1_166   T1_167
## 1 2.030380 1.406037 0.5245388 1.538550 0.2780937 2.623936 2.380977 0.9350185
## 2 2.013655 1.399601 0.5316905 1.558147 0.2860468 2.611667 2.374888 0.9442331
## 3 1.997084 1.393240 0.5384727 1.577063 0.2937263 2.599264 2.368606 0.9529143
## 4 1.980657 1.386953 0.5448726 1.595289 0.3011160 2.586716 2.362130 0.9610562
## 5 1.964363 1.380739 0.5508773 1.612816 0.3081992 2.574014 2.355459 0.9686534
##      T1_168   T1_169   T1_170   T1_171   T1_172   T1_173   T1_174   T1_175
## 1 1.463836 0.7604865 0.6090474 0.5345297 1.402610 1.079505 1.332258 1.289677
## 2 1.463481 0.7590971 0.6098860 0.5484804 1.411189 1.073245 1.342016 1.283073
## 3 1.462813 0.7576207 0.6105071 0.5617958 1.419202 1.066917 1.351244 1.276452
## 4 1.461823 0.7560570 0.6109124 0.5744683 1.426644 1.060533 1.359920 1.269821
## 5 1.460500 0.7544058 0.6111034 0.5864904 1.433509 1.054107 1.368023 1.263185
##      T1_176   T1_177   T1_178
## 1 3.450058 2.167462 0.7105921
## 2 3.435370 2.169318 0.7169350
## 3 3.420320 2.170780 0.7227992
## 4 3.404901 2.171847 0.7281846
## 5 3.389105 2.172519 0.7330916
```

```
plotH1_average2 <- as.rspec(plotH1_average2)
is.rspec(plotH1_average2)
```

```
## [1] TRUE
```

Eliminamos todo aquello que ya no necesito. Importante: *names_H1* me hará falta mas adelante, por eso no lo elimino todavía.

```
rm(x0)
rm(x1)
rm(x2)
rm(x3)
rm(x4)
rm(x_sum)
rm(i)
rm(plotH1_average)
```

3.2. Opción 2: para hacerlo más fino, tendría que crear un loop como el que describo abajo, pero no me termina de correr bien. No lo incluyo en los chunks porque no me interesa correrlo ahora. Lo dejo como texto normal por si en un futuro soy capaz de solucionarlo.

```
plotH1_average <- plotH1.sm$wl
plotH1_average <- as.data.frame(plotH1_average)
colnames(plotH1_average) <- c("wl")
head(plotH1_average)

i_seq <- seq(from=2, to=151, by=2)

for(i in i_seq) {
  plotH1_average[, i] <- apply(plotH1.sm[, c(i, i+1)], 1, mean, na.rm=TRUE)
}

plotH1_average
plotH1_average[1:5,1:5]
plotH1.sm_manual[1:5,1:5]
```

4. Cálculo de variables colorimétricas.

Las variables colorimétricas que pueden incluirse en el paper, tales como hue, chroma, spectral purity, etc., se calculan con este sencillo comando:

```
colouremetric.variables.plotH1_average2 <- summary(plotH1_average2)
# write.csv2(colouremetric.variables.plotH1_average2,
#           file="plotH1_average2_colourimetric_variables.csv", row.names = TRUE)
```

Se podría guardar en un csv los resultados, pero por el momento dejo las almohadillas porque no me interesa crear el archivo. Ahora voy a plotear los espectros de las 75 plantas de la zona H1. Además, la función colorea cada línea según la forma del espectro, tal y cómo se vería al ojo humano. Por último, voy a crear un objeto *rgb* con los colores de cada espectro, que utilizaré más adelante para colorear los puntos en los distintos modelos de visión.

```
plot(plotH1_average2, type = "o", col = spec2rgb(plotH1_average2))
```

```
rgb <- spec2rgb(plotH1_average2)
```

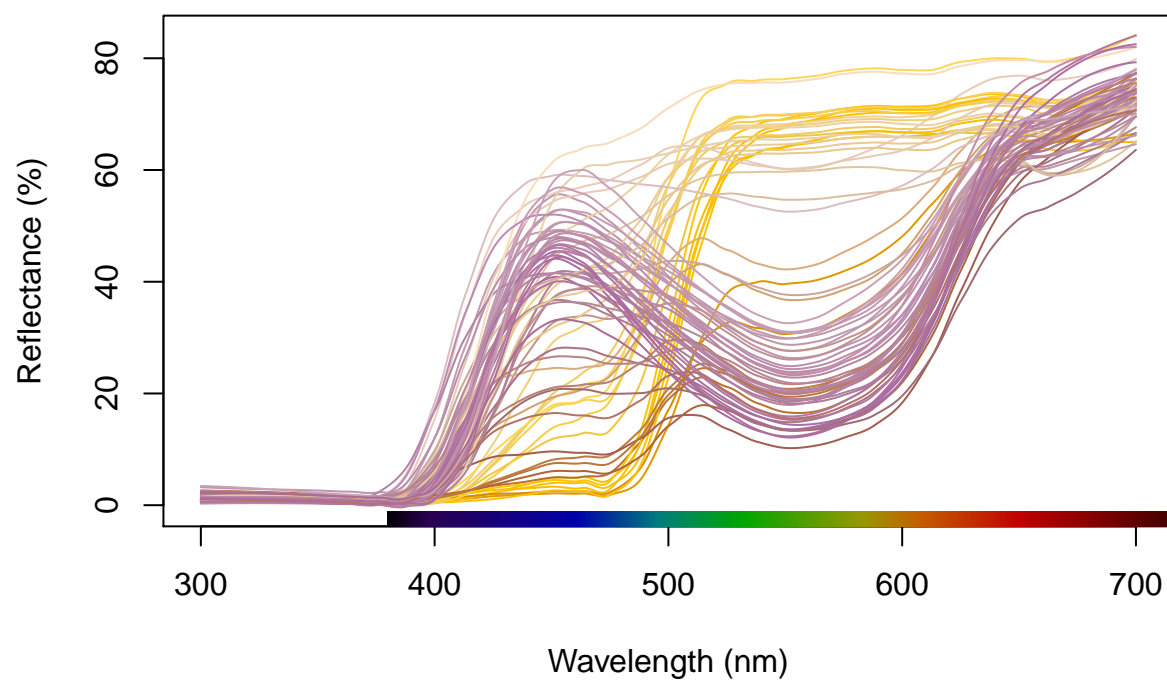


Figura 3: Espectros de todas las flores de la zona H1 del transecto 1.

5. Modelos de visión.

5.1. Modelo de visión de las ABEJAS.

Para prevenir errores, voy a corregir posibles valores “cero” en nuestro dataset con la función *procspec*.

```
plotH1_average2 <- procspec(plotH1_average2, fixneg = "addmin")
```

Para representar las 75 flores provenientes de la zona H1 del transecto 1 en el hexágono de color (himenópteros), primero tengo que crear el modelo visual con los siguientes parámetros. He utilizado el “green” estándar como background, la iluminación D65, el quantum catch = Ei, el modelo visual de *Apis mellifera* y la corrección de von kries.

```
vis.flowers.H1.bees <- vismodel(plotH1_average2,
                                visual = "apis",
                                qcatch = "Ei",
                                relative = FALSE,
                                vonkries = TRUE,
                                achromatic = "l",
                                bkg = "green",
                                illum="D65"
)
```

Una vez creado el modelo visual *vis.flowers.H1.bees*, ya puedo plotear los puntos en el hexágono.

```
hexagon.H1.bees <- colspace(vis.flowers.H1.bees, space = "hexagon")
hexplot(hexagon.H1.bees,
        sectors="coarse",
        labels=TRUE,
        main="Hybrid Transect 1",
        cex.main=1.5,
        col="black")
```

Si quisiera guardar en un csv los valores obtenidos del hexágono, como por ejemplo la distancia al centro de cada punto, tendría que ejecutar el siguiente scrip:

```
# write.csv2(hexagon.H1.bees, file="hexagon.H1.bees.csv", row.names = TRUE)
```

Ahora voy a depurar el gráfico para poder exportarlo para la publicación, incluyendo los colores de cada flor tal y como se perciben al ojo humano. El gráfico lo exportaría en pdf a 7.5 x 5.5 pulgadas.

```
1 par(mar=c(1, 1, 1, 1))
2 hexplot(hexagon.H1.bees,
3         sectors="coarse",
4         achro=TRUE,
5         labels=FALSE,
6         cex=1.1,
7         labels.cex = 1.2,
8         col=rgb)
```

Hybrid Transect 1

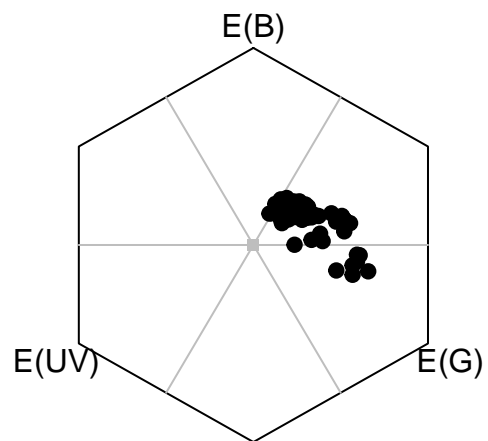


Figura 4: Hexágono de color con los datos en bruto.

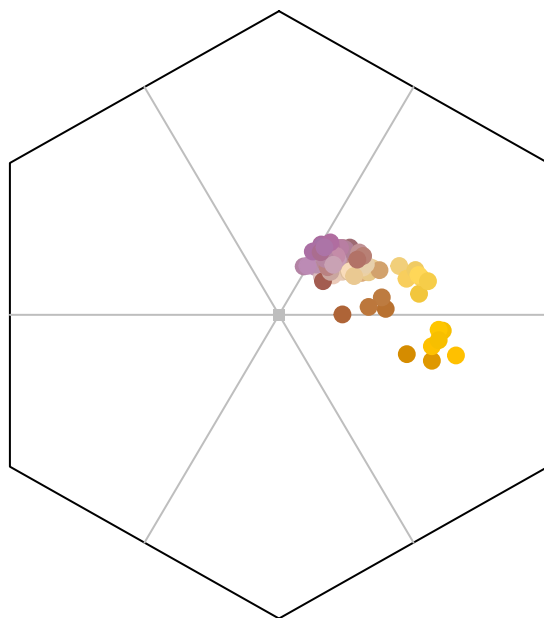


Figura 5: Hexágono de color con los color loci de cada flor coloreados tal y como son percibidos por el ojo humano.

5.2. Modelo de visión de las MARIPOSAS.

Voy a utilizar las sensibilidades de *Papilio xunthus* como representante de los Lepidópteros. Las sensibilidades (i.e., curvas de excitación de los cuatro fotorreceptores) las obtuve del paper de Koshitaka et al. 2008.

```
papilio <-read.table("sensibilidad_papilio_xunthus.txt", header=T)
papilio <- as.rspect(papilio)
is.rspect(papilio)
```

```
## [1] TRUE
```

```
head(papilio)
```

```
##      wl      u      s      m l
## 1 300 0.05100 0.038 0.0520 0
## 2 301 0.05912 0.040 0.0612 0
## 3 302 0.06724 0.042 0.0704 0
## 4 303 0.07536 0.044 0.0796 0
## 5 304 0.08348 0.046 0.0888 0
## 6 305 0.09160 0.048 0.0980 0
```

La dinámica a seguir es la misma que con las abejas. Primero se crea el modelo visual con los mismos parámetros y luego se representa en el tetraedro. He utilizado el “green” estándar como background, la iluminación D65, el modelo visual de *Papilio xuthus* que acabo de crear y la corrección de von kries.

```
vis.flowers.H1.butterflies <- vismodel(plotH1_average2,
                                       visual = papilio,
                                       vonkries = TRUE,
                                       illum= "D65",
                                       bkg = "green")
```

Una vez creado el modelo visual *vis.flowers.H1.butterflies*, ya puedo plotear los puntos en el tetraedro.

```
tetraedro.H1.butterflies <- colspace(vis.flowers.H1.butterflies, space = "tcs")
plot(tetraedro.H1.butterflies)
```

Si quisiera guardar en un csv los valores obtenidos del tetraedro, como por ejemplo la distancia al centro de cada punto, tendría que ejecutar el siguiente script:

```
# write.csv2(tetraedro.H1.butterflies, file="tetraedro.H1.butterflies.csv", row.names = TRUE)
```

Ahora voy a depurar el gráfico para poder exportarlo para la publicación, incluyendo los colores de cada flor tal y como se perciben al ojo humano. El gráfico lo exportaría en pdf a 7.5 x 5.5 pulgadas.

```
1 par(mar=c(1, 1, 1, 1))
2 plot(tetraedro.H1.butterflies,
3      achro=TRUE,
4      labels=FALSE,
5      cex.main=1.2,
6      achro.size = 0.5,
7      achro.col = "grey",
8      zoom = 1,
9      out.lwd = 1,
10     col=rgb)
```

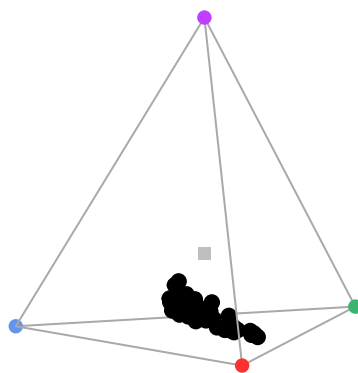


Figura 6: Hexágono de color con los datos en bruto.

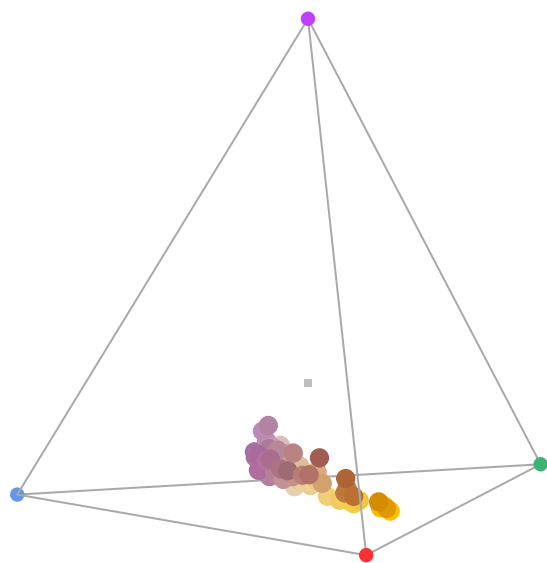


Figura 7: Tetraedro con los color loci de cada flor coloreados tal y como son percibidos por el ojo humano.

5.3. Modelo de visión de las MOSCAS.

La dinámica a seguir es la misma que con las abejas y mariposas. Primero se crea el modelo visual con los mismos parámetros y luego se representa en el espacio categórico. He utilizado el “green” estándar como background, la iluminación D65, el modelo visual de *Musca domestica* y la corrección de von Kries.

```
vis.flowers.H1.flies <- vismodel(plotH1_average2,  
                                bkg = "green",  
                                visual = "musca",  
                                achromatic = "md.r1",  
                                illum = "D65",  
                                vonkries = TRUE)
```

Una vez creado el modelo visual *vis.flowers.H1.flies*, ya puedo plotear los puntos en el espacio categórico.

```
categorical.H1.flies <- colspace(vis.flowers.H1.flies, space = "categorical")  
plot(categorical.H1.flies)
```

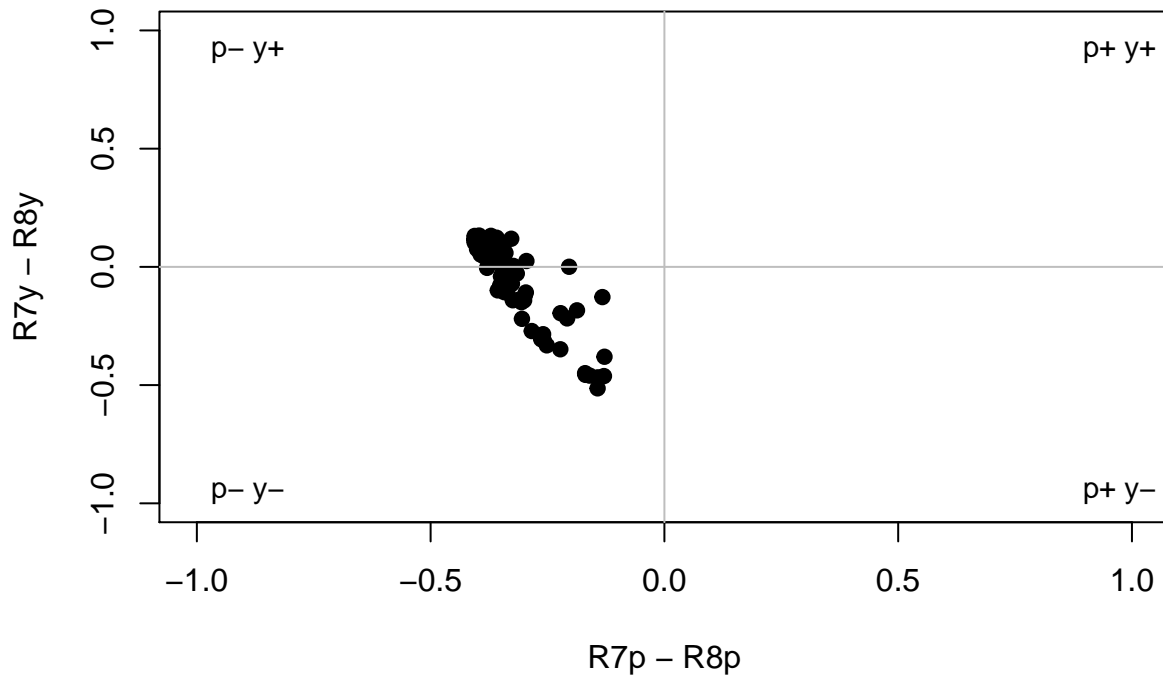


Figura 8: Hexágono de color con los datos en bruto.

Si quisiera guardar en un csv los valores obtenidos del espacio categórico, como por ejemplo la distancia al centro de cada punto, tendría que ejecutar el siguiente script:

```
# write.csv2(categorical.H1.flies, file="categorical.H1.flies.csv", row.names = TRUE)
```

Ahora voy a depurar el gráfico para poder exportarlo para la publicación, incluyendo los colores de cada flor tal y como se perciben al ojo humano. El gráfico lo exportaría en pdf a 7.5 x 5.5 pulgadas.

```
1 par(mar=c(2, 2, 2, 2))
2 plot(categorical.H1.flies,
3       labels=TRUE,
4       cex=1.1,
5       labels.cex = 1.1,
6       col=rgb)
```

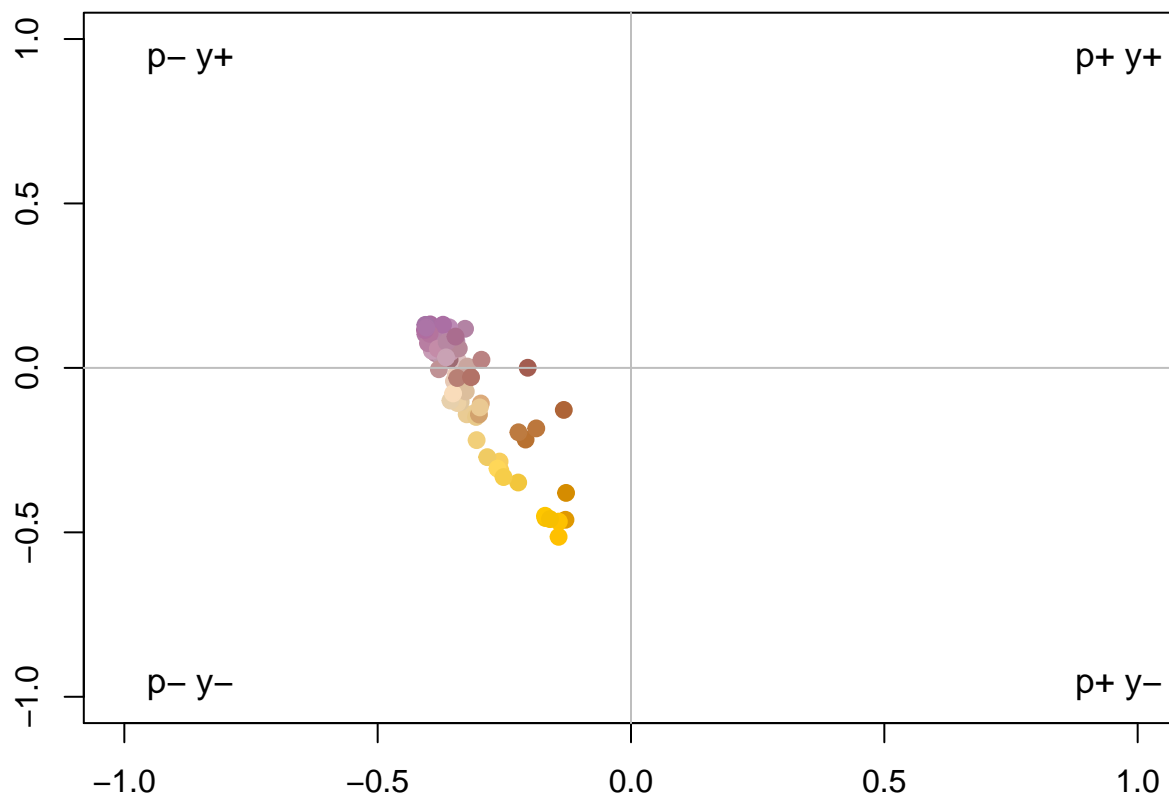


Figura 9: Espacio categórico con los color loci de cada flor coloreados tal y como son percibidos por el ojo humano.

5.4. Modelo de visión de los ESCARABAJOS.

Voy a utilizar las sensibilidades de *Pygopleurus israelitus* como representante de los Coleópteros. Las sensibilidades (i.e., curvas de excitación de los cuatro fotorreceptores) las obtuve del paper de Martinez-Harm et al 2020.

```
p.israelitus <-read.table("sensibilidad_pygopleurus_israelitus.txt", header=T)
p.israelitus <- as.rspec(p.israelitus)
is.rspec(p.israelitus)
```

```
## [1] TRUE
```

```
head(p.israelitus)
```

```
##      wl      s      m      l
## 1 300 0.144 0.063 0.224
## 2 301 0.154 0.065 0.229
## 3 302 0.164 0.066 0.234
## 4 303 0.172 0.068 0.244
## 5 304 0.179 0.070 0.251
## 6 305 0.187 0.072 0.257
```

La dinámica a seguir es la misma que con las abejas, mariposas y moscas. Primero se crea el modelo visual con los mismos parámetros y luego se representa en el espacio categórico. He utilizado el “green” estándar como background, la iluminación D65, el modelo visual de *Pygopleurus israelitus* que he creado y la corrección de von Kries.

```
vis.flowers.H1.beetles <- vismodel(plotH1_average2,
                                   visual = p.israelitus,
                                   vonkries = TRUE,
                                   illum= "D65",
                                   bkg = "green")
```

Una vez creado el modelo visual *vis.flowers.H1.beetles*, ya puedo plotear los puntos en el triángulo.

```
triangle.H1.beetles <- colspace(vis.flowers.H1.beetles, space = "tri")
plot(triangle.H1.beetles)
```

Si quisiera guardar en un csv los valores obtenidos del triángulo, como por ejemplo la distancia al centro de cada punto, tendría que ejecutar el siguiente script:

```
# write.csv2(triangle.H1.beetles, file="triangle.H1.beetles.csv", row.names = TRUE)
```

Ahora voy a depurar el gráfico para poder exportarlo para la publicación, incluyendo los colores de cada flor tal y como se perciben al ojo humano. El gráfico lo exportaría en pdf a 7.5 x 5.5 pulgadas.

```
1 par(mar=c(1, 1, 1, 1))
2 plot(triangle.H1.beetles,
3       achro=TRUE,
4       labels=FALSE,
5       cex.main=1.2,
6       achro.size = 0.5,
7       achro.col = "grey",
8       zoom = 1,
9       out.lwd = 1,
10      col=rgb)
```

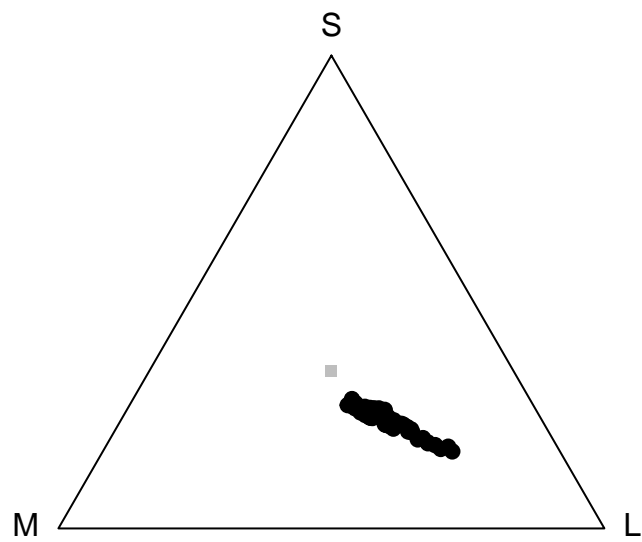


Figura 10: Hexágono de color con los datos en bruto.

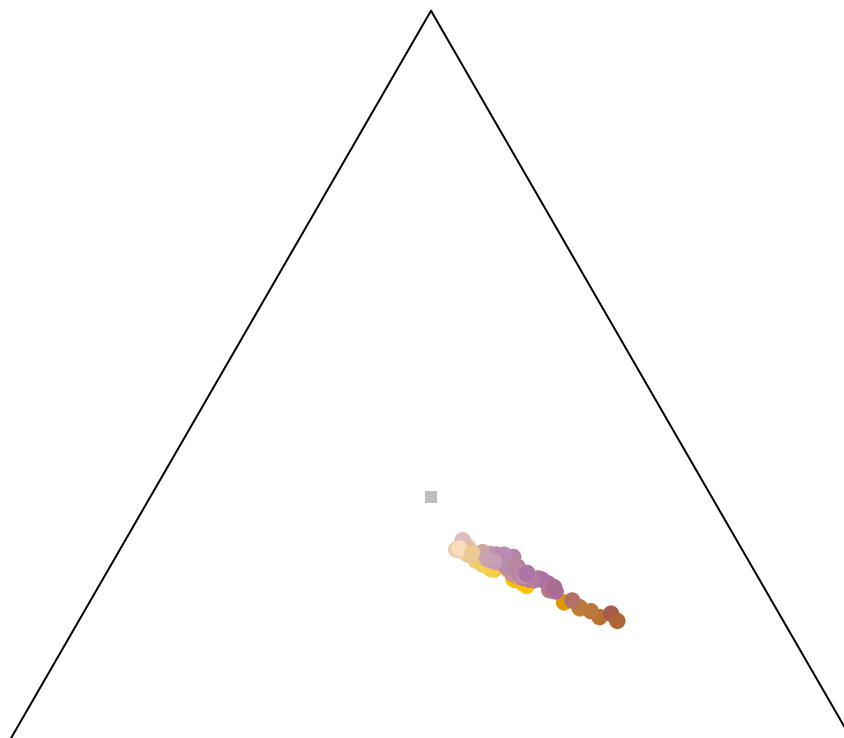


Figura 11: Triángulo con los color loci de cada flor coloreados tal y como son percibidos por el ojo humano.

6. Calculando los contrastes cromáticos (distancia al centro)

El contraste cromático o distancia al centro se obtiene de forma automática en *pavo*. El contraste cromático lo encontramos en la columna **r.vec** de cada modelo de visión. Vamos a extraer dicha columna de cada modelo de visión, además de crear un vector con los nombres de cada individuo, para construir un data frame con todos los contrastes cromáticos:

```
chrom_contrast_beets <- hexagon.H1.beets$r.vec
chrom_contrast_butterflies <- tetraedro.H1.butterflies$r.vec
chrom_contrast_flies <- categorical.H1.flies$r.vec
chrom_contrast_beetles <- triangle.H1.beetles$r.vec
names_H1 <- names_H1[-(1)]
chrom_contrast_table <- data.frame(names_H1,
                                   chrom_contrast_beets,
                                   chrom_contrast_butterflies,
                                   chrom_contrast_flies,
                                   chrom_contrast_beetles)
head(chrom_contrast_table, n=5)
```

```
##   names_H1 chrom_contrast_beets chrom_contrast_butterflies chrom_contrast_flies
## 1   T1_101          0.5256432              0.3556475          0.4795634
## 2   T1_102          0.4662946              0.2789381          0.4135562
## 3   T1_103          0.5418356              0.3285876          0.4868130
## 4   T1_104          0.4818189              0.2674756          0.4043440
## 5   T1_105          0.5138580              0.3344171          0.4884762
##   chrom_contrast_beetles
## 1          0.2856306
## 2          0.1620155
## 3          0.1981053
## 4          0.1564201
## 5          0.2102131
```

Ahora tan solo hay que calcular las medias, medianas, máximos, mínimos y errores estándar.

```
mean <- apply(subset(chrom_contrast_table, select = c(2:5)), 2, mean, na.rm=TRUE)
sd <- apply(subset(chrom_contrast_table, select = c(2:5)), 2, sd, na.rm=TRUE)
se <- sd/sqrt(75)
min <- apply(subset(chrom_contrast_table, select = c(2:5)), 2, min, na.rm=TRUE)
max <- apply(subset(chrom_contrast_table, select = c(2:5)), 2, max, na.rm=TRUE)
median <- apply(subset(chrom_contrast_table, select = c(2:5)), 2, median, na.rm=TRUE)
```

Ahora creo un data frame con los estadísticos descriptivos calculados:

```
chrom_contrast_statistics <- data.frame(mean,
                                         min,
                                         max,
                                         median,
                                         se)
chrom_contrast_statistics
```

```
##               mean      min      max    median      se
## chrom_contrast_beets 0.3125402 0.17894636 0.5978283 0.2826915 0.011483422
```

```
## chrom_contrast_butterflies 0.2099312 0.12659184 0.3556475 0.1935882 0.006126004
## chrom_contrast_flies      0.3745707 0.18416471 0.5331792 0.3743004 0.006510508
## chrom_contrast_beetles    0.1904982 0.09058221 0.3762839 0.1863247 0.007556061
```

Por último, elimino todo aquello que no volveré a necesitar:

```
rm(mean)
rm(sd)
rm(se)
rm(min)
rm(max)
rm(median)
rm(chrom_contrast_bees)
rm(chrom_contrast_butterflies)
rm(chrom_contrast_flies)
```

7. Calculando el contraste acromático en el modelo de visión de las abejas.

La última parte de este informe incluye el cálculo del contraste acromático, que solo se calcula para el modelo de visión de las abejas. Este contraste acromático se suele referenciar como **green contrast** y se calcula como el valor absoluto de la excitación del fotorreceptor verde (Eg) - 0.5. En el caso de *pavo*, los valores de excitación del fotorreceptor verde se encuentran en la columna “l” de los valores obtenidos en el hexágono de color.

Vamos a calcular los valores de *green contrast* a partir del objeto *hexagon.H1.bees* y lo transformamos en un data frame:

```
achrom_contrast_bees <- abs(hexagon.H1.bees$l-0.5)
achrom_contrast_table <- data.frame(names_H1,
                                     achrom_contrast_bees)
head(achrom_contrast_table, n=6)
```

```
##   names_H1 achrom_contrast_bees
## 1   T1_101          0.2608875
## 2   T1_102          0.3458817
## 3   T1_103          0.3336675
## 4   T1_104          0.3580826
## 5   T1_105          0.3390340
## 6   T1_106          0.2315003
```

Ahora tan solo tengo que calcular las medias, medianas, máximos, mínimos y errores estándar:

```
mean(achrom_contrast_table$achrom_contrast_bees)
```

```
## [1] 0.2753283
```

```
sd(achrom_contrast_table$achrom_contrast_bees)/sqrt(75)
```

```
## [1] 0.00825399
```



```
min(achrom_contrast_table$achrom_contrast_bees)
```

```
## [1] 0.07914547
```

```
max(achrom_contrast_table$achrom_contrast_bees)
```

```
## [1] 0.3843002
```

```
median(achrom_contrast_table$achrom_contrast_bees)
```

```
## [1] 0.2750608
```

Con esto acabarían todos los análisis para la zona híbrida H1 del transecto 1. El resto de zonas también se han analizado. Encontrarás los scripts en la carpeta *scripts* del repositorio de Github.