Code for: *Flower Power: What people aesthetically like in temperate agricultural grasslands*

Abstract

This document presents all codes used to develop the analays relative to Hypothesis 1.2: Standardized automatic image analysis can predict people's aesthetic appreciations via (a) the extraction of flower cover and colour richness and (b) the diversity in spectral characteristics of the respective images.

Contents

1.Getting ready

1.1 Materials

Images, labels, appreciation scores, and manual estimates of flower cover and flower colour richness are provided as supplementary files.

1.2 Working environment preparation

```r
setwd("path/to/your/working/directory/")
dir.create("imgs_class")
dir.create("imgs_class_ANDRGB")
dir.create("imgs_down10")
dir.create("lab_xy")
dir.create("colsummary")
dir.create("Mov_wind_rao")
dir.create("figures")
```

1.3. Package installation

```r
packages <- c("caret", "crfsuite", "data.table", "dplyr", "forecast", "ggplot2",
  "ggpubr", "rgdal", "randomForest", "raster", "rasterdiv", "sp", "terra",
  "vegan")
# Load or install missing packages
lapply(packages, function(package) {
  if (!require(package, character.only = TRUE)) {
    install.packages(package)
    library(package, character.only = TRUE)
  }
})
```

2. Image classification

2.1 Image labelling

We developed a labelled dataset by randomly selecting a 1000 pixels × 1000 pixels image patch in each image, plotting in RGB colours labelling around 50 pixels, and assigning to each pixel the class to which it belongs. Next we removed pixels too close to each other from the dataset.

```r
library(ggplot2)
library(raster)
library(rgdal)
setwd("path/to/your/working/directory/")
rm(list = ls())

# load the image list and create seeds list that you will use to select a
# random image patch
imgs <- list.files("imgs/", full.names = TRUE)


set.seed(1002)
seedsx <- sample(1:2^15, length(imgs))
set.seed(1056)
seedsy <- sample(1:2^15, length(imgs))
buffer <- 500

# PIXEL LABELLING PROCEDURE 1) Define the classes
classes <- c("Y", "W", "R", "G", "B", "V")
# 2) Select one class (e.g, 'Y' yellow flowers)
class <- classes[4]
# 3) Run the 'RUN HERE' line below. An image will appear in the plot pane.  4)
# Click on some pixels of the class selected in point 2, (e.g., 'Y'), then
# press esc on the keyboard.  A dataframe with coordinates, image name and
# label will be saved as a '.csv' file.  5) Press Ctrl+Alt+T to run the section
# again. A new image will appear in the plot pane.  Repeat point 4) and 5)
# until all the images have been labelled 6) Once labels have been placed on
# all images, go to step 2) and select the second class.  Then go on with point
# 3),4),5),6) until all classes will be labelled.  NB: LOCATOR WORKS PROPERLY
# ONLY WHEN ZOOM IN GLOBAL OPTIONS IS SET TO 100 %

#-------------------------------------------------------------------------------
# RUN HERE
```

```r
#-------------------------------------------------------------------------------

{
  if (!exists("i")) {
    i <- 1
  } else {
    i <- i + 1
  }
  print(paste0("i=", i, "class=", class))
  img <- zoiCx <- zoiCy <- NULL
  img <- brick(imgs[i])
  fig <- gsub("\\..*", "", basename(imgs[i]))
  set.seed(seedsx[i])
  zoiCx <- sample(buffer:(4000 - buffer), 1)
  set.seed(seedsy[i])
  zoiCy <- sample(buffer:(3000 - buffer), 1)
  imgc <- crop(img, extent(zoiCx - buffer, zoiCx + buffer, zoiCy - buffer,
    zoiCy + buffer))
  par(xaxt = "n", yaxt = "n", mar = c(0, 0, 0, 0), oma = c(0, 0, 0, 0))
  plotRGB(imgc, axes = F)
  labelled <- NULL
  labelled <- data.frame(locator(type = "p", pch = 16, col = "red"))
  if (nrow(labelled) > 0) {
    labelled$type <- class
    labelled$im <- fig
    labelled$zoiCx <- zoiCx
    labelled$zoiCy <- zoiCy
    write.csv(labelled, file = paste0("./lab_xy/", "lab_xy_pic_", fig, "class_",
      class, ".csv"))
  }
}

# combine labs and remove pixels which are too close one to each other.

cdf <- do.call(rbind, lapply(list.files(path = "lab_xy/", full.names = T),
  read.csv, header = TRUE, row.names = 1))

cdf$x10 <- floor(cdf$x/10)
cdf$y10 <- floor(cdf$y/10)
cdfundupli <- cdf[!duplicated(cdf[, c("x10", "y10", "im")]), ]
cdfundupli <- cdfundupli[, -c(7:8)]

write.csv(cdfundupli, "labelled.csv")
```

2.2 Image sampling, Classifier compilation, Accuracy assessment and Image classification

The reflectance values in the red, green and blue bands of the images in labelled pixels were extracted to obtain a dataset reporting image ID, classification class, xy coordinates, and reflectance values for each labelled pixel. Afterwards the dataset was split into training and validation by randomly assigning 66% of images for training, and 33% of images for validation. The training dataset was used to develop a random forest classifier, and classifier performance was assessed on the validation dataset. The compiled classifier was used to classify all images.

```r
library(dplyr)
library(caret)
library(randomForest)
library(crfsuite)
library(raster)
library(terra)
library(sp)
library(ggplot2)

# Image Sampling

rm(list = ls())
labs <- read.csv("labelled.csv", row.names = 1)
imgs <- unique(labs$im)

for (i in 1:length(imgs)) {
  imgname <- imgs[i]
  print(i)
  imgpath <- paste0("./imgs/", imgname, ".jpg")
  imgr <- brick(imgpath)

  labs_im_i <- labs[labs$im == imgname, ]
  pt <- SpatialPoints(coords = labs_im_i[, c("x", "y")])

  ptsa <- raster::extract(imgr, pt, sp = T)
  ptsa@data$x <- ptsa@coords[, 1]
  ptsa@data$y <- ptsa@coords[, 2]
  ptsa@data$type <- labs_im_i$type
  ptsa@data$imname <- imgname
  ptsa@data$zoiCx <- labs_im_i$zoiCx
  ptsa@data$zoiCy <- labs_im_i$zoiCy

  ptsa_data <- ptsa@data

  colnames(ptsa_data) <- c("R", "G", "B", "x", "y", "type", "imname", "zoiCx",
    "zoiCy")
  if (!exists("totsampled")) {
    totsampled <- ptsa_data
  } else {
    totsampled <- rbind(totsampled, ptsa_data)
  }
}
write.csv(totsampled, file = "./sampled_precheck.csv")


# Check RGB values of labelled pixels, and remove wrongly labelled pixels

rm(list = ls())
sam <- read.csv("sampled_precheck.csv", row.names = 1)
sam$cod <- rgb(sam$R/256, sam$G/256, sam$B/256)

ggplot(sam, aes(x = R, y = G, col = cod)) + geom_point() +
  scale_colour_identity() + facet_wrap(~type) + theme_bw()
```

```r
ggplot(sam, aes(x = G, y = B, col = cod)) + geom_point() +
  scale_colour_identity() + facet_wrap(~type) + theme_bw()

ggplot(sam, aes(x = R, y = B, col = cod)) + geom_point() +
  scale_colour_identity() + facet_wrap(~type) + theme_bw()

# based on the above plots I identified the following rules.

indextoberemoved <- unique(c(which(sam$type %in% c("Y") & sam$R < 150),
  which(sam$type %in% c("W") & sam$R < 200), which(sam$type %in% c("W") &
    sam$G < 200)))

sam <- sam[-indextoberemoved, ]
write.csv(sam, "./sampled.csv")

# RF classifier compilation and accuracy assessment

rm(list = ls())
sam <- read.csv("sampled.csv", row.names = 1)
imgs <- unique(sam$imname)

set.seed(123)
train_imgs <- sample(imgs, length(imgs) * 2/3)
sam <- sam %>%
  mutate(TV = ifelse(imname %in% train_imgs, "T", "V"))
train <- sam[sam$TV == "T", ]
test <- sam[sam$TV == "V", ]

rf <- randomForest(x = train[, c("R", "G", "B")], y = as.factor(train$type),
  ntree = 500, proximity = T)

saveRDS(rf, "./RF_1px_train.rds")

p1 <- predict(rf, train)
confusionMatrix(p1, as.factor(train$type))
p2 <- predict(rf, test)
caret::confusionMatrix(p2, as.factor(test$type))


metrics <- crf_evaluation(p2, as.factor(test$type))
metrics
mean(metrics[[1]][, "f1"])

# Classify all images, save statistics and false colour classified images

setwd("path/to/your/working/directory/")
rm(list = ls())
rf <- readRDS("./RF_1px_train.rds")
imgs <- list.files("./imgs/", full.names = T)

colnamesShort <- c("B", "G", "R", "V", "W", "Y")
colnames <- c("lightblue", "forestgreen", "red", "purple", "white", "yellow")
cods <- c(1, 2, 3, 4, 5, 6)
```

```r
# add a spatial filter
for (i in 1:length(imgs)) {
  print(i)
  # Compute image features
  img <- brick(imgs[i])
  fig <- gsub("\\..*", "", basename(imgs[i]))
  names(img) <- c("R", "G", "B")
  classifiedbeforefocal <- terra::predict(img,
    rf, na.rm = T)
  classified <- focal(classifiedbeforefocal,
    w = matrix(1, 5, 5), fun = modal)
  writeRaster(classified, filename = paste0("imgs_class/",
    fig, ".tiff"), overwrite = T)

  # Plot the classified raster
  pcla <- rasterVis::levelplot(classified, margin = F,
    colorkey = F, scales = list(draw = F),
    at = seq(0, 6), col.regions = colnames)
  # Plot the rgb raster
  r <- raster(img)
  cols <- factor(rgb(img[], maxColorValue = 255))
  r[] <- cols
  prgb <- rasterVis::levelplot(r, col.regions = as.character(levels(cols)),
    margin = F, scales = list(draw = F), colorkey = FALSE)

  # Arrange plots using grid.arrange
  gga <- ggarrange(pcla, prgb, ncol = 1) %>%
    annotate_figure(top = text_grob(paste0("Fig.",
      fig), size = 14), bottom = text_grob(colsummaryString,
      size = 14))

  ggsave(gga, filename = paste0("imgs_class_ANDRGB/",
    fig, ".png"), width = 20, height = 35,
    unit = "cm", bg = "white")
  # create perc stats
  summa <- summary(as.factor(getValues(classified)))
  summadf <- data.frame(cods = names(summa),
    perc = round(as.numeric(summa)/sum(summa),
      5))
  colnamesdf <- data.frame(colnames, colnamesShort,
    cods)
  colsummary <- merge(colnamesdf, summadf, by = "cods")
  colsummary$fig <- fig
  colsummaryString = paste(colsummary$colnamesShort,
    "=", colsummary$perc, sep = "", collapse = ", ")

  write.csv(colsummary, paste0("./colsummary/colsummary_",
    fig, ".csv"))
}
```

3. Colour classes diversity indices extraction

Since we hypothesised that people‘s appreciation could not only be affected by flower cover and flower colour richness, but also by the evenness of the distribution of the pixel classes, with higher diversity associated with higher appreciation, we applied two diversity metrics widely used in grassland ecology studies, Shannon and Simpson, to the frequency distribution of the pixel classes in each image. The six pixel classes were considered as species, and pixel counts as species abundances. Finally, since we hypothesised that images with more differences in the reflectance spectra between the pixels (e.g., less homogeneously green) to be associated with higher people’s aesthetic rating, we assessed the diversity in spectral characteristics of all 94 pictures.

```r
setwd("path/to/your/working/directory/")
library(vegan)
library(dplyr)
rm(list = ls())

res <- read.csv("Whole_DIV_IND_class_species_pre_indices.csv", row.names = 1)
res[is.na(res)] <- 0

res_with_DIV_INDICES <- res %>%
  rowwise() %>%
  mutate(values = list(c(C1, C2, C3, C4, C5, C6, CNA)), sha = diversity(values,
    index = "shannon"), sim = diversity(values, index = "simpson"), ) %>%
  ungroup() %>%
  dplyr::select(-values)

write.csv(res_with_DIV_INDICES, "Whole_DIV_IND_class_species.csv")
```

4. Spectral analysis

The rationale behind spectral diversity metrics is that the increased spectral variability reflects an increased variety of habitats, species, and organs, since different habitat, species and organs respond in their own way to incoming solar radiation according to their pigment, water, and biochemical content, as well as leaf and canopy structure. The diversity metrics we applied here were whole image standard deviation, moving window standard deviation, multidimensional Rao index. Since shadowed pixels may lead to noise rather than enhancing the information content at very-high spatial resolution, image resolution was reduced by applying a downscaling factor equal to 10 before computing the diversity metrics, as frequently applied in the optical diversity context.

Standard deviations were separately computed on each of the three bands (red, green, blue) and the resulting statistics were then averaged to obtain a standard deviation (SD) for each image. Computing standard deviations in square moving windows and not only in the whole image allowed us to measure only the spatial variability occurring at the scale of interest (i.e., flower scale in our study), and that can therefore be attributed to flowers. We first computed a diversity map by assigning to each pixel the SD of reflectance values in a square moving window of five pixels, and then obtained a summary metric for each image by averaging the diversity map. Lastly, we computed the Rao’s index, which quantifies the difference in reflectance values between two pixels drawn randomly with replacement from a set of pixels by considering their abundance and their pairwise distance

```r
setwd("path/to/your/working/directory/")
library(raster)
library(terra)
library(rasterdiv)
library(dplyr)
rm(list = ls())
```

```r
# Aggregate images with aggregation factor= 10

imgs <- list.files(path = "imgs/", full.names = T)
i <- 1
for (i in 1:length(imgs)) {
  print(i)
  imga <- raster::aggregate(stack(imgs[i]), fact = 10)
  writeRaster(imga, format = "GTiff", overwrite = T,
    filename = paste0("imgs_down10/", gsub("\\..*",
      "", basename(imgs[i])), ".tiff"))
}

# Compute the whole image standard deviation (SD) of each figure, separately
# for each band

rm(list = ls())
imgs <- list.files("./imgs_down10/", full.names = T)
res <- data.frame(imgs = gsub("\\..*", "", basename(imgs)), sdR = NA, sdG = NA,
  sdB = NA)

for (i in 1:length(imgs)) {
  img <- brick(imgs[i])
  res[i, "sdR"] <- sd(getValues(img[[1]]))
  res[i, "sdG"] <- sd(getValues(img[[2]]))
  res[i, "sdB"] <- sd(getValues(img[[3]]))
}
write.csv(res, "WHOLE_standard_dev.csv")

# Compute the standard deviation (SD) of each figure, separately for each band,
# in 5 pixels moving windows

rm(list = ls())
imgs <- list.files("./imgs_down10/", full.names = T)
res <- data.frame(path = imgs, imgs = gsub("\\..*", "", basename(imgs)),
  mo_wi_sd_R = NA, mo_wi_sd_G = NA, mo_wi_sd_B = NA)
for (i in 1:nrow(res)) {
  img <- brick(res$path[i])
  res[i, "mo_wi_sd_R"] <- mean(getValues(focal(img[[1]], w = matrix(1, 5, 5),
    fun = sd)), na.rm = T)
  res[i, "mo_wi_sd_G"] <- mean(getValues(focal(img[[2]], w = matrix(1, 5, 5),
    fun = sd)), na.rm = T)
  res[i, "mo_wi_sd_B"] <- mean(getValues(focal(img[[3]], w = matrix(1, 5, 5),
    fun = sd)), na.rm = T)
}
write.csv(res, "MOV_WIND_standard_dev.csv")

# Compute the multidimentional Rao's Index for each image

rm(list = ls())
imgs <- list.files("./imgs_down10/", full.names = T)
res <- data.frame(imgs = gsub("\\..*", "", basename(imgs)), rao = NA)
for (i in 1:nrow(res)) {
  img <- brick(imgs[i])
  mlist <- list(raster::as.matrix(img[[1]]), raster::as.matrix(img[[2]]),
    raster::as.matrix(img[[3]]))
  rao <- paRao(mlist, window = 5, na.tolerance = 0.9, alpha = 1,
    dist_m = "euclidean", np = 1, method = "multidimension")
  res[i, "rao"] <- mean(rao[[1]][[1]])
  resi <- res[i, ]
  write.csv(resi, paste0("MOV_WIND_RAO5_", i, "_.csv"))
}
write.csv(res, "MOV_WIND_RAO5.csv")
```

5. Analysis of the drivers

5.1 Final dataset compilation

Here, we compiled a dataset by combining all image analysis we carried out, and we assessed the relationship between manually and automatically extracted Flower cover and Flower colour richness. While deriving the flower cover from the classified image through image statistics was straightforward, defining the flower colour richness required the definition of the percentage threshold above which a flower class is to be considered as present in the picture. We tested 2000 possible thresholds from 0% to 1% and chose the threshold resulting in the highest correlation between manually and automatically estimated flower colour richness.

```
setwd("path/to/your/working/directory/")
library(ggplot2)
library(dplyr)
library(tidyr)
library(ggpmisc)
library(patchwork)
library(readxl)
rm(list = ls())

whole_div <- read.csv("Whole_DIV_IND_class_species.csv", row.names = 1)
whole_sd <- read.csv("WHOLE_standard_dev.csv", row.names = 1)
mo_wi_sd <- read.csv("MOV_WIND_standard_dev.csv", row.names = 1)
mo_wi_rao <- do.call(rbind, lapply(list.files(path = "Mov_wind_rao/",
  full.names = T), read.csv, header = TRUE, row.names = 1))
manual_flower_estimates <- readxl::read_excel("Flower appreciation and cover.xlsx",
  sheet = "Data") %>%
  rename(imgs = colnames(.["picture_no"]))


merged_df <- whole_div %>%
  merge(whole_sd, by = "imgs") %>%
  merge(mo_wi_sd, by = "imgs") %>%
  merge(mo_wi_rao, by = "imgs") %>%
  merge(manual_flower_estimates, by = "imgs") %>%  merge(manual_flower_estimates,
  merge(manual_flower_estimates, by = "imgs") %>%  by = "imgs") %>%
mutate(sdRGB = (sdR + sdG + sdB)/3) %>%
  mutate(mo_wi_sd_RGB = (mo_wi_sd_R + mo_wi_sd_G + mo_wi_sd_B)/3) %>%
  mutate(validpx = (C1 + C2 + C3 + C4 + C5 + C6)) %>%
  mutate(Flower_cover_automatic = (C1 + C3 + C4 + C5 + C6)/validpx *
    100) %>%
  mutate(C1p = C1/validpx) %>%
```

```r
  mutate(C3p = C3/validpx) %>%
  mutate(C4p = C4/validpx) %>%
  mutate(C5p = C5/validpx) %>%
  mutate(C6p = C6/validpx) %>%
  select(c("imgs", "Appreciation_mean", "Flower_cover_manual",
    "Flower_colours_manual", "Flower_cover_automatic", "sha",
    "sim", "sdRGB", "mo_wi_sd_RGB", "rao", "C1p", "C3p", "C4p",
    "C5p", "C6p"))


write.csv(merged_df, "merged_df_beforeautocolN.csv")

# Identification of the best threshold for Flower colour richness
# identification

rm(list = ls())
merged_df <- read.csv("merged_df_beforeautocolN.csv", row.names = 1)
tss <- data.frame(thresh = c(seq(from = 0, to = 0.01, by = 5e-06)),
  correlation = NA)

for (i in 1:nrow(tss)) {
  tsi <- tss[i, "thresh"]
  merged_df <- merged_df %>%
    mutate(Flower_colours_automatic = rowSums(across(c(C1p, C3p,
      C4p, C5p, C6p)) > tsi))
  tss[i, "correlation"] <- cor(merged_df$Flower_colours_manual,
    merged_df$Flower_colours_automatic)
}


plot(tss$thresh, tss$correlation, type = "b")
best_thresh <- tss[which.max(tss$correlation), "thresh"]
best_thresh

merged_df <- merged_df %>%
  mutate(Flower_colours_automatic = rowSums(across(c(C1p, C3p, C4p, C5p, C6p)) >
    best_thresh)) %>%
  select(!c("C1p", "C3p", "C4p", "C5p", "C6p"))

options(scipen = 999)
cor.test(merged_df$Flower_colours_manual, merged_df$Flower_colours_automatic)

write.csv(merged_df, "merged_df.csv")

# Compare manual and automatic flower cover and flower colour richness

rm(list = ls())
df <- read.csv("merged_df.csv", row.names = 1)

p_cover <- ggplot(df, aes(x = Flower_cover_manual,
  y = Flower_cover_automatic)) + geom_point(alpha = 0.3) +
  stat_poly_line(se = F) + stat_correlation(use_label(list("R",
  "p.value")), small.r = T, small.p = T) + xlab("Manual Flower Cover") +
  coord_cartesian(xlim = c(0, 25), ylim = c(0, 25)) +
  ylab("Automatic Flower Cover") + theme_bw()

p_rich <- ggplot(df, aes(x = Flower_colours_manual,
  y = Flower_colours_automatic)) + geom_jitter(width = 0.03,
  height = 0.03, alpha = 0.3) + stat_poly_line(se = F) +
  stat_correlation(use_label(list("R", "p.value")),
    small.r = T, small.p = T) + xlab("Manual Flower Colour Richness") +
  coord_cartesian(xlim = c(0, 5), ylim = c(0, 5)) +
  ylab("Automatic Flower Colour Richness") + theme_bw()
p11p12 <- p11 + p12
p11p12
ggsave(p11p12, filename = "figures/Flowers_manual_automatic.png", width = 16,
  height = 8.5, unit = "cm")
```

5.2 Relationship testing

We identified the relationship between image appreciation and Flower cover, Flower colour richness, Colour classes diversity indices, Spectral diversity metrics. The relationships between appreciation and flower cover and between appreciation and diversity metrics were assessed by fitting second-order polynomial regression models, since the relationships were not linear, whereas the relationship between appreciation and Spectral diversity metrics were assessed by computing r_Pearson.

```
rm(list=ls())
setwd("path/to/your/working/directory/")
df<-read.csv("merged_df.csv",row.names=1)

library(ggplot2)
library(dplyr)
library(tidyr)
library(ggpmisc)
library(patchwork)
library(readxl)
library(GGally)

ggpairs(df[,c("Appreciation_mean","Flower_cover_automatic","Flower_colours_automatic",
              "sha","sim","sdRGB","mo_wi_sd_RGB","rao")])
cs<-rev(c("#f94144","#f9c74f","#90be6d","#577590","grey"));
basopts<-list(coord_cartesian(ylim=c(1,5)),
                theme_bw()
                )
formula <- y ~ poly(x, 2, raw = TRUE)

flower_manual<-ggplot(df,aes(x=Flower_cover_manual,y=Appreciation_mean,group=NA,
                             col=factor(Flower_colours_manual,levels=c(1:5))))+
   geom_point()+
   xlab("Manual flower cover (%)")+
   ylab("Appreciation")+
   scale_colour_manual(name="Manual flower\ncolour richness",
                       values=cs,drop=F)+
   stat_poly_line(formula = formula,se=F,col="black") +
   guides(col=guide_legend(ncol=3))+
   stat_poly_eq(mapping = (use_label(list("R2","p"))),
                formula = formula,col="black",small.p=T)+
```

```r
  basopts+
  theme(legend.position = c(1, 0), # Position legend in top-right inside the plot
        legend.justification = c(1, 0) ,
        legend.background =  element_rect(fill="white",colour="black",size=.2,
                                          linetype="solid"))

flower_automatic<-ggplot(df,aes(x=Flower_cover_automatic,y=Appreciation_mean,group=NA,
                               col=factor(Flower_colours_automatic,levels=c(1:5))))+
  geom_point()+
  xlab("Automatic flower cover (%)")+
  ylab("Appreciation")+
  scale_colour_manual(name="Automatic flower\ncolour richness",
                      values=cs,drop=F)+
  stat_poly_line(formula = formula,se=F,col="black") +
  guides(col=guide_legend(ncol=3))+
  stat_poly_eq(mapping = (use_label(list("R2","p"))),
               formula = formula,col="black",small.p=T)+
  basopts+
  theme(legend.position = c(1, 0), # Position legend in top-right inside the plot
        legend.justification = c(1, 0) ,
        legend.background =  element_rect(fill="white",colour="black",size=.2,
                                          linetype="solid"))

Sha<-ggplot(df,aes(x=sha,y=Appreciation_mean,group=NA))+
  geom_point(alpha=2/10, shape=21, fill="blue") +
  xlab("Shannon")+
  ylab("Appreciation")+
  stat_poly_line( formula=formula,col="black",se=F)+
  stat_poly_eq(mapping = (use_label(list("R2","p"))),
              formula=formula,col="black",small.p=T)+
  basopts

Sim<-ggplot(df,aes(x=sim,y=Appreciation_mean,group=NA))+
  geom_point(alpha=2/10, shape=21, fill="blue") +
  xlab("Simpson")+
  ylab("Appreciation")+
  stat_poly_line( formula=formula,col="black",se=F)+
  stat_poly_eq(mapping = (use_label(list("R2","p"))),
               formula=formula,col="black",small.p=T)+
  basopts

whosd<-ggplot(df,aes(x=sdRGB,y=Appreciation_mean,group=NA))+
  geom_point(alpha=4/10, shape=21, fill="grey") +
  xlab("Whole image SD")+
  ylab("Appreciation")+
  stat_poly_eq(mapping = (use_label(list("p"))),
              col="black",small.p=T)+
  basopts

mosd<-ggplot(df,aes(x=mo_wi_sd_RGB,y=Appreciation_mean,group=NA))+
  geom_point(alpha=4/10, shape=21, fill="grey") +
  xlab("Moving window SD")+
  ylab("Appreciation")+
      stat_poly_eq(mapping = (use_label(list("p"))),col="black",small.p=T)+
   basopts


Rao<-ggplot(df,aes(x=rao,y=Appreciation_mean,group=NA))+
   geom_point(alpha=4/10, shape=21, fill="grey") +
   xlab("Multid. Rao Index")+
   ylab("Appreciation")+
   stat_poly_eq(mapping = (use_label(list("p"))),
             col="black",small.p=T)+
   basopts

patch<- (flower_manual+ggtitle("a Flower presence metrics")+flower_automatic+
             plot_layout(axis_titles = "collect"))/
        (Sha+ggtitle("b Colour classes diversity indices")+
             Sim+ plot_layout(axis_titles = "collect"))/
        (whosd+ggtitle("c Spectral diversity")+
             mosd+ggtitle("metrics")+
             theme(plot.title = element_text(hjust = -0.38))+Rao+
             plot_layout(axis_titles = "collect"))+
         plot_layout(heights=c(1,1,.7))

patch
ggsave(patch,filename="figures/summary.png",width=16,height=22,unit="cm")
```

Licence