implant: Functional Data Analysis Pipeline for High Throughput Plant Phenotyping
User's Guide

Ronghao Wang[1], **Yumou Qiu**[2], **and James C. Schnable**[3]

[1]Department of Statistics, University of Nebraska-Lincoln, Lincoln, 68503, USA
[2]Department of Statistics, Iowa State University, Ames, 50011, USA
[3]Center for Plant Science Innovation, Department of Agronomy and Horticulture, University of Nebraska-Lincoln, Lincoln, 68503, USA

ABSTRACT

Functional ANOVA has many benefits in estimating the growth curve dynamics for plants, comparing to the point-wise ANOVA. This package provides functions to help users extract the phenotypical parameters from RGB images of plants taken in green houses as the data and analyze them using Functional ANOVA model.

Contents

1 Introduction

This tutorial provides a guide of the *implant* package for processing RGB images of plants and using functional anova to analyze the growth dynamics for the plants. The whole procedure can be devided into two part, the data pre-processing part and functional analysis part. In the data pre-processing part, we first process the images of the plants of interest taken from the green house and using a series of steps to process these images to segment the bodies of these plants from their backgrounds. Then we can extract the phenotypical parameters such as size, height and width, etc., from the processed images. In the analysis part, we use the data extracted from the processed images in the first part, and use the functional anova approach to build models and have the corresponding analysis.

2 Pre-Processing

In this part, we will use functions within this package to pre-process the RGB image data we have. Our goal is to segment the object of interest (body of the plant) from its background, then we extract the phenotypical parameters from the segmented images. We will follow the following steps:
1. Defining the Region of Interest;
2. Segmenting the object of Interest;
3. Morphorlogical Operations.

Figure1(a) from the Appendix is an image of a plant in a green house, and Figure1(b) is another image with exactly the same condition as Figure 1(a) except that it has no plant in the pot. Instead, a green strip was inserted into the pot.

2.1 Defining the Region of Interest

Using the package *PNG*, we can transform these two images into matrices of pixel intensities. Let's name Figure 1(a) as *imageOriginal* and Figure 1(b) as *image*2. Since these two images all contain RGB channels, the figures should be read as three-dimensional arrays with the last dimensions is equal to 3 or 4. This depends on whether the images you use are RGB (3 channels) or RGBA (4 channels). In our case, the images we use has 4 channels, and we only want to consider the RGB channels. Therefore, we take the first three channels as our data matrices instead of 4. Then we can check the dimensions of these two matrices. For example, we'd like to find the dimensions for *imageOriginal* for each channel:

```
sizeOriginal = dim(imageOriginal)[c(1,2)]
>2056 2454
```

And we can see that in each channel, the matrix has 2056 rows and 2454 columns. Using these information, we can define our region of interest.

Using the function *ColorG*, we can find the green strip from *image*2 and change its color from green to the background color, white. Also we can find the vertical location of the top of the pot (the location is the same as the location is the bottom part of the plant body in Figure 1(a).)

```
tempG = ColorG(image, rowThreshold = 0.002)
lowerRowBound = tempG$lowb
```

We may also reduce the sizes of these two images using function *sample*, and name the reduced original image and the reduced empty pot image as iamge1 and imageX1, respectively.

```
image1 = sample(imageOriginal, RowSample = 2, ColSample = 2)
imageX1 = sample(image2, RowSample = 2, ColSample = 2)
```

And we can see that the sizes of these two images are both reduced from 2056×2454 to 1028×1227.

By applying the function *ColorB* to the original image(the one with plant), we can find the region of interest and change the color of the region that we are not interested in into black (Since we will segment the image in the next step and make the segmented image binary, which means coloring the background as black and the object of interest as white.).

```
resultColor = ColorB(imageOriginal)
ColBound = c(resultColor$lb, resultColor$rb)
RI = matrix(1, sizeOrignal[1], sizeOrignal[2])
```

```
RI[, c(1 : ColBound[1])] = 0
RI[, c(ColBound[2] : sizeOrignal[2])] = 0
RI[c(lowerRowBound : sizeOrignal[1]), ] = 0
imageOriginalR = imageOriginal[, , 1]
imageOriginalG = imageOriginal[, , 2]
imageOriginalB = imageOriginal[, , 3]
imageCutR = matrix(imageOriginalR[RI == 1],
lowerRowBound - 1, ColBound[2] - ColBound[1] - 1)
imageCutG = matrix(imageOriginalG[RI == 1], lowerRowBound - 1,
ColBound[2] - ColBound[1] - 1)
imageCutB = matrix(imageOriginalB[RI == 1], lowerRowBound - 1,
ColBound[2] - ColBound[1] - 1)
imageCut = array(c(imageCutR, imageCutG, imageCutB), dim = c(dim(imageCutR), 3))
```

2.2 Image Segmentation using Thresholding Method

Applying the function *ColorG* on imageX1, we can have a new matrix with the color of the strip being changed to white (imageX1G, see Figure 1(a) in the Appendix). Then we can have a contrasted image (imageContrast) using the absolute value of the difference between image1 and imageX1G. (see Figure 1(c) in the Appendix).

```
tempG1 = ColorG(imageX1, rowThreshold = 0.002)
imageX1G = tempG1$c
imageContrast = abs(image1 - imageX1G)[, , 1 : 3]
```

Function *imageBinary* can help us segment the object of interest from the background and transform the segmented image into black and white, where the object of interest is colored as white and the background is colored as black. By defining the parameters of this function, we can have the segmented images for image1 and imageContrast, respectively. Then we can have a well segmented result by finding the intersection between the segmented image1 and the segmented imageContrast. See Figure 1(d) in the Appendix.

```
imageB1 = imageBinary(image1,  weight = c(-1, 2, -1), threshold1 = 30 / 255,
threshold2 = 0.02)
imageB2 = imageBinary(imageContrast1,  weight = c(1, -2, 1), threshold1 = 0.7,
threshold2 = 0)
imageB = imageB1 * imageB2
```

Note: An alternative method for segmentation we provide in this package is the Hidden Markov Random Field-EM Method. For details, please check the help documentation for the function $HMRF - EM$.

2.3 Morphological Operations

The next step is the morphological operations. Since there are still some unwanted noise, we would like to use the *erosion* function to eliminate those noise. However, this action may cause the loose information of the segmented plant. Therefore, the function *dilation* is needed as well. We follow the procedure: "Dilation-Erosion-Erosion-Dilation" and get the following result. (See Figure1 (e)-(h) in the Appendix).

```
imageBD = dilation(imageB, mask = matrix(1, 5, 5))
imageBDE = erosion( imageBD, mask = matrix(1, 5, 5) )
imageBDEE = erosion( imageBDE, mask = matrix(1, 2, 2) )
imageBDEED = dilation( imageBDEE, mask = matrix(1, 2, 2) )
```

2.4 Obtaining the Final Segmented Image

From the "Defining the Region of Interest" step, we have a matrix *RI* which has the same size as the original image, but the region that we are not interested in has been colored as black. We first reduce the size of *RI* to the same size as *imageBDEED* we got from the previous step. Then we cut the processed image (imageBDEED) and keep the region of interest(in order to remove the noise around the plant body).

```
RI1 = sampleMatrix(RI, RowSample = 2, ColSample = 2)
RI1row = sum(rowSums(RI1) > 0)
RI1col = sum(colSums(RI1) > 0)
imageBF = matrix(imageBDEED[RI1 == 1], RI1row, RI1col)
```

imageBF is the final segmented image we want. See Figure 1(i) in the Appendix.

2.5 Extracting the Phenotypical Parameters

Using the obtained segmented image, we can extract the information we want from its matrix. Firstly, we need to know how many millimeters that each pixel represents horizontally and vertically. The information we have is:

```
Xsize (in mm): 1.54902601242065
Ysize (in mm): 1.55327594280243
```

which means each pixel from the image represents $1.549mm$ in horizontal and $1.553mm$ in vertical. Using this information, we can calcualte the size, height and weight from the pixel matrix of the segmented image.

```
pixelCount = sum(imageBF)
#We multiplied by 2*2 since we have reduced the size by picking every two pixels
#in each row and each column
plantSize = pixelCount * Xsize * Ysize * 2 * 2
RowCount = rowSums(imageBF); ColCount = colSums(imageBF)
NonzeroRow = which(RowCount > 0); NonzeroCol = which(ColCount > 0)
plantheight = Ysize * (quantile(NonzeroRow, 0.975)[[1]] -
quantile(NonzeroRow, 0.025)[[1]])
plantwidth = Xsize * (quantile(NonzeroCol, 0.975)[[1]] -
quantile(NonzeroCol, 0.025)[[1]])
if (is.na(plantheight))
plantheight = 0
if (is.na(plantwidth))
plantwidth = 0
```

3 Functional ANOVA

3.1 Preparing for the Model Fitting

In the previous part, we have extracted the phenotypical parameters we want from the processed images. In this part, we will use these parameters as our data to build Functional ANOVA models.

By using the function $fanova$, we can estimate the model parameters. And we are able to have the estimated treatment effect and obtain the corresponding confidence intervals according to these estimated parameters got from $fanova$. Before using this function, there are a few steps we need to finish:
1. Decide your model (including whether you want to have interactions or not.) This step is recommended using pointwise anova and check the significant factors before applying $fanova$.
2. Create a Y matrix for the response variable. Each row should represent an observation and each column should represent an observing time point.
3. Create an X matrix for the explanatory variables. Each column represents a factor you would like to involve in your model.
4. Build a Time matrix for the observing time points. Each row of this matrix should be used for one observation and each column of this matrix should be used for the observed time from the original time point of this observation(i.e. the ith row and the jth column represents the jth Day/Hour/Month/Year/Minute from the original observing time for the ith observation. Each observation can have its own original observing time. The observing time can be regular or irregular.
Note: if your observing time is regular, you can omit the Time matrix term.

In this example, we would like to study how would the size of a plant be affected by its genotype effect and its block effect. In our experiment, we have 406 plants with 140 different genotypes in three blocks. Each genotype has 3 replicates. Within each block, a replicate of a genotype appears only once. Let's treat the block and genotype are two fixed effects in this example. Since the observing time for each plant is irregular, we have an irregular Time matrix T with 406 rows and 22 columns. Also, the dimension of Y matrix is 406×22 and the dimension of X matrix is $140 * 2$. For the model,we decide to use a model with no interactions between these two factors. Both factors are categorical variables.

3.2 Model Fitting

After well preparation, we can use function $fanova$ to fit our model. We assume the functional ANOVA model for the plant growth to be:

$$y_i(t) = \mu(t) + G_i g_(t) + B_i b(t) + \varepsilon_i(t), \tag{1}$$

where $\mu(t)$ is the growth function of the plant which has the $1st$ type of genotype from the 1^{st} block, $g(t)$ is the genotype effect, $b(t)$ is the fixed block effect, and $\varepsilon_i(t)$ is the random error with mean 0. Since what we want is to study the relationship of the size, genotype effect and block effect, we will define the parameters of function $fanova$ as follows:

```
#This step is to factorize each factor
#The first factor, genotype
X1 = as.factor(unlist(X_geno))
#The second factor, block
X2 = as.factor(blk_X[,ncol(blk_X)])
X = data.frame(X1,X2)
formula = toString("~ X[,1]+X[,2]")
n = dim(Y)[1]
analysis = fanova(Y = Y, X = X,  Time = T,
                  regular = 0, tt = as.numeric(T[1,][!is.na(T[1,])]),
                  K.int = 6, n, order = 4, total.time = 60,
                  interact = 0, p = 2, d0 = 0, d1 = 2, d2 = 2, lower = -10,
                  upper = 15, n.eval = 100, formula = formula)
para = analysis$est_fun
lambda = analysis$lambda,
bhat = analysis$bhat,
hatmatrix = analysis$S
```

where *para* is the a matrix of estimated parameters. Each of its columns represents the estimated parameter for a factor. In this example, its first column represents the estimated $\mu(t)$s. λ is the estimated smooth parameter using GCV, *bhat* is a vector of all the estimated parameters, and *hatmatrix* is the hat matrix of the model.

3.3 Treatment Effects

Function *est_trt* can be used for estimating the treatment effect of interest, using the estimated parameter matrix *para* being obtained using $fanova$. For example, if we would like to know the estimated treatment effect of the $1st$ genotype and the $1st$ block, we can use:

```
trt_g1b1 = est_trt(1,1,0, est_fun = para, X)
```

Similarly, the estimated treatment effect for the $2nd$ genotype in block 1 is:

```
trt_g2b1 = est_trt(2,1,0, est_fun = para, X)
plot(x = t.eval, y = trt_g1b1, type = "l", col = "1")
lines(x = t.eval, y = trt_g2b1, type = "l", col = "2")
```

3.4 Confidence Intervals

In order to calculate the point-wise confidence interval for $\hat{Y}(t)$, we need to know the variance of $\hat{Y}(t)$.

$$Var(\hat{Y}(t)) = LS\Sigma_e S'L',$$

where L is the contrast vector, S is the hatmatrix of the model, and Σ_e is the covariance matrix of the residual term $\varepsilon(t)$. Using the function *Sigma_epsilon*, we can calculate the estimated sample covariance matrix of $\varepsilon(t)$:

```
Cov = Sigma_epsilon(Y = Y, X = X, Time = T)
```

Suppose we would like to know the confidence intervals for the $1st$ genotype in block 1, and the confidence interval for the $1st$ genotype in block 2. We construct our contrast vector L and calculate the corresponding variance of $\hat{Y}(t)$ using function *Var_yhat*:

```
L = c(rep(1,K),rep(0,139*K),rep(0,2*K))
Var_yhat = Var_yhat(data_X = X,Time = T,plant_id = 1,total_time = 60,
                    K = 10, order = 4,d0 = 0,L = L , S = hatmatrix, Cov = Cov)
```

Then we can calculate the confidence intervals using function *ci*:

```
ci_1 = ci(trt = trt_g1b1, Var_yhat = Var_yhat, alpha = 0.05)
ci_2 = ci(trt = trt_g1b2, Var_yhat = Var_yhat, alpha = 0.05)
plot(x = tt, y = trt_g1b1, type = "l", ylim = c(-10000, 350000))
lines(x = tt, y = ci_1$lb, type = "l", lty = 2, col = "red")
lines(x = tt, y = ci_1$up, type = "l", lty = 2, col = "red")
lines(x = tt, y = trt_g1b2, type = "l",lty = 3, col = "green")
lines(x = tt, y = ci_2$lb, type = "l", lty = 2, col = "red")
lines(x = tt, y = ci_2$up, type = "l", lty = 2, col = "red")
```

4 Appendix

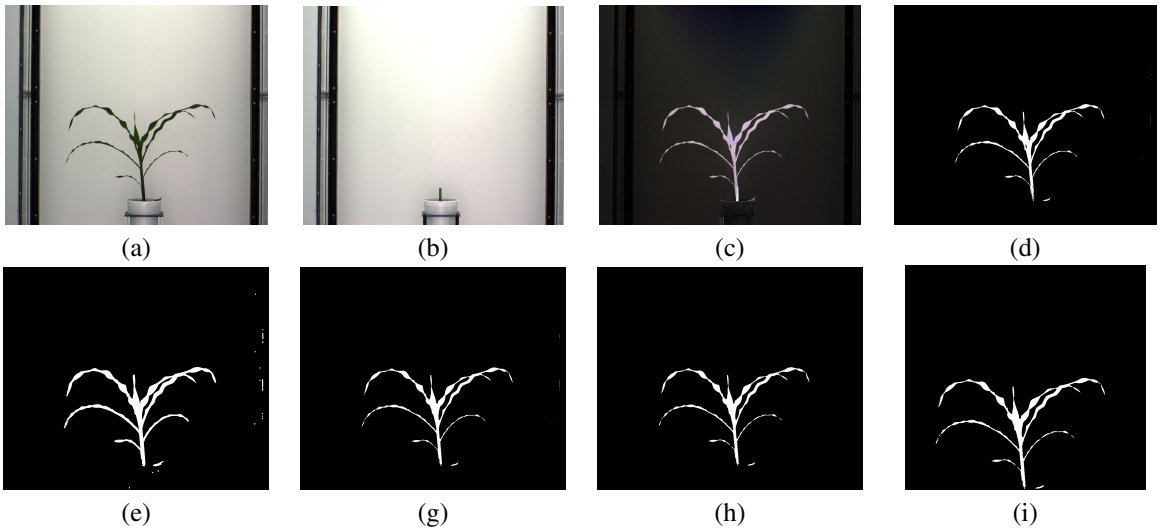

Figure 1. (a) Original Image with Plant. (b) Original Image without Plant. (c) Contrast of (a) and (b). (d) Segmented Image by Thresholding Method. (e) Dilated Image. (f) Dilated-Eroded Image. (g) Dilated-Eroded-Eroded-Dilated Image. (i) Final Segmented Image.

References

Acknowledgements