User Guide for "implant" Package

Ronghao Wang[1], Yumou Qiu[2], Yuzhen Zhou[1], Yuhang Xu[3], and James C. Schnable[4]

[1]Department of Statistics, University of Nebraska-Lincoln, Lincoln, 68503, USA
[2]Department of Statistics, Iowa State University, Ames, 50011, USA
[3]Department of Applied Statistics and Operations Research, Bowling Green State University, Bowling Green, 43403, USA
[4]Center for Plant Science Innovation, Department of Agronomy and Horticulture, University of Nebraska-Lincoln, Lincoln, 68503, USA

ABSTRACT

This user guide provides examples for the R package *"implant"*. Those examples cover RGB image processing, plant feature extraction and functional data analysis for plant growth dynamics. The main results and methodology are presented in the paper "A High-throughput Phenotyping Pipeline for Image Processing and Functional Growth Curve Analysis".

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 plants. The whole procedure can be divided into two parts, the image data pre-processing part and functional analysis part. We first process the images of plants from greenhouse high-throughput phenotyping systems and use a series of steps to segment the bodies of plants from the backgrounds. Then, we extract the phenotypical traits, such as size, height and width, from the segmented images. In the functional analysis part, based on the extracted features from the images, we apply the functional Anova model to study the treatment and genotype effects on plant growth.

2 Downloading the Package

The package *"implant"* is uploaded on Github. Users can download the package using:

```
devtools::install_github("rwang14/implant")
library(implant)
```

Notice that users have to load the package *"devtools"* before using the above code. Please see the Github page https://github.com/rwang14/implant for detail instruction on installing *"Rtools"* and *"devtools"*.

3 Image Processing

In this part, we use functions in this package to pre-process the RGB image data. Our goal is to segment the foreground (body of the plant) from its background, then we extract the phenotypical traits from the segmented images. Figure1(a) from the Appendix is an image of a plant in a green house, and Figure1(b) is an image with exactly the same condition as Figure 1(a) except that it has no plant in the pot. There is also a green strip in the pot to help users identify the lower boundary of the region of interest.

3.1 Loading Example Images from the Package

We stored an example png file of plant called *"image1.png"*, and an image of empty pot called *"image_pot.png"* for the image processing part. In the Image Processing part, we first load a plant image and the pot image, and use them to illustrate how this part works.

```
library(png)
#load the image of plant
imageOriginal = readPNG(system.file("extdata", "image1.png",
                                    package = "implant", mustWork = TRUE))[ , ,1:3]
#load the image of empty pot
imageX = readPNG(system.file("extdata", "image_pot.png",
                                 package = "implant", mustWork = TRUE))[ , ,1:3]
```

3.2 Defining the Region of Interest

Using the package "*png*", we can transform these two images into arrays of pixel intensities. Name Figure 1(a) as *imageOriginal* and Figure 1(b) as *imageX*. Since both images contain RGB channels, the figures should be read as three-dimensional arrays with their last dimensions being 3 or 4. This depends on whether the images are RGB (3 channels) or RGBA (4 channels). In our case, each image 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 using all the four channels. Hence, for each image, we obtain an array of three same-size matrices, and then we can check the dimension of each matrix. For example:

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

We can see that in each channel, the matrix has 2056 rows and 2454 columns.

Functions *ColorG* and *ColorB* can help us identify the region of interest. Using the function *ColorG*, we can detect the lower boundary of the plant, that is, the bottom of the plant (equivalently, the border-top of the pot). See Figure 1(a).

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

"*ColorB*" can help us identify the left boundary and right boundary of the plant (the black bars in the image). Incorporated with the lower boundary (the border-top of the pot) we obtained by "*ColorG*", we can obtain the region of interest for the plants. See the red rectangle in Figure 1 panel (b). Then we change the color of the unwanted region into black. This is because in the next step, we conduct image segmentation and make the segmented image binary, i.e., coloring the background as black and the foreground as white. We then let the pixel intensities of the uninterested region be equal to 0 (i.e., change the color to black).

```
#Identify the region of interest
resultColor = ColorB(imageOriginal)
ColBound = c(resultColor$lb, resultColor$rb)
RI = matrix(1, sizeOriginal[1], sizeOriginal[2])
RI[, c(1 : ColBound[1])] = 0
RI[, c(ColBound[2] : sizeOriginal[2])] = 0
RI[c(lowerRowBound : sizeOriginal[1]), ] = 0
```

Note that this identification strategy is specifically designed for processing the plant images taken in the University of Nebraska-Lincoln (UNL) greenhouse Innovation Center. Different greenhouse systems need different strategies for locating the region of interest.

3.3 Image Segmentation using Thresholding Method

In order to make the process more efficient, we can reduce the size of *imageOriginal* and the size of *imageX* using function *downsize*, and name the corresponding reduced images as *image*1_*reduced* and imageX1, respectively.

```
image1_reduced = downsize(imageOriginal, RowSample = 2, ColSample = 2)
imageX1 = downsize(imageX, RowSample = 2, ColSample = 2)
```

We can see that the sizes of the images are both reduced from 2056×2454 to 1028×1227.

We then apply "*ColorG*" to imageX1 (Figure 1 (b)) to change the color of the strip from green to white (the color of the background). Then we can have a contrasted image (imageContrast) by calculating the absolute value of the difference between *image*1_*reduced* and *imageX*1 (see 1 (c) in the Appendix).

```
tempG1 = ColorG(imageX1, rowThreshold = 0.002)
imageX1G = tempG1$c
imageContrast = abs(image1_reduced - imageX1G)[ , , 1 : 3]
```

Function *imageBinary* can help us segment the foreground from the background and transform the segmented image into black and white, where the foreground is colored as white and the background is colored as black. Using this function, we can obtain the segmented images for *image*1_*reduced* and *imageContrast*, respectively. By taking the intersection between the segmented image for *image*1_*reduced* and the segmented image for *imageContrast*, we obtain a better segmented result. See Figure 1 (d) - (f) in the Appendix.

```
imageB1 = imageBinary(image1_reduced,  weight = c(-1, 2, -1), threshold1 = 30 / 255,
threshold2 = 0.02)
imageB2 = imageBinary(imageContrast,  weight = c(1, -2, 1), threshold1 = 0.7,
threshold2 = 0)
imageB = imageB1 * imageB2
```

Note: An alternative method for plant segmentation we provide in this package is to use the hidden Markov random field model. For details, please check the help documentation for the function *"HMRF"*.

3.4 Morphological Operations

In order to obtain a descent segmentation result, we add morphological operations to our process. The *erosion* function can help us perform the "erosion operation" and eliminate the unwanted noise. However, this action may cause the loss of information and therefore, the "dilation operation" is needed as well. The function *dilation* is used to perform the "dilation operation". We follow the procedure: "Dilation-Erosion-Erosion-Dilation" and get the following result (see Figure 1 (g) 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, 3, 3) )
imageBDEED = dilation( imageBDEE, mask = matrix(1, 3, 3) )
```

3.5 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, with the uninterested region being colored as black. Since *imageBDEED* is obtained based on the size-reduced original image, we need to reduce the size of *RI* to the same size as that of *imageBDEED*. Then we cut the processed image (imageBDEED) and keep the part that is within the region of interest.

```
RI1 = downsize_matrix(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.

3.6 Extracting the Phenotypical traits

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 1.549*mm* in horizontal and 1.553*mm* in vertical. Using this information, we can calculate the size, height and weight from the pixel matrix of the segmented image.

```
#a = 2, b = 2 since we have reduced the size by picking every two pixels
#in each row and each column
extract_pheno(processed_image = imageBF,
        Xsize = 1.54902601242065, Ysize = 1.55327594280243, a = 2, b = 2)$height
extract_pheno(processed_image = imageBF,
        Xsize = 1.54902601242065, Ysize = 1.55327594280243, a = 2, b = 2)$width
extract_pheno(processed_image = imageBF,
        Xsize = 1.54902601242065, Ysize = 1.55327594280243, a = 2, b = 2)$size
```

Details can be found in the help documentation for this function. All the above R code can be found in the file *"demo_DCT.R"* from the *"data-raw"* folder in the package.

4 Functional ANOVA

4.1 Preparing for Model Fitting

In the previous part, we have extracted the phenotypical traits from the processed images. In this part, we use these traits to build Functional Anova models. The example data we use in this tutorial is reduced from the data being described in the paper. The reduced data includes the size of 9 plants with three genotypes (1, and 2 and 3) within 3 different blocks. These plants were observed from the 0^{th} day to the 44^{th} day. We have attached this example data called *data.csv* in this package. We read the attached data and define the input terms as follows:

```
data_new = read.csv(system.file("extdata", "data.csv",package = "implant",
                    mustWork = TRUE))
#The first column is the positions of the observations from the original dataset,
#which can be ignored.
Y.na.mat = data_new[, -c(1:3)]
#This step is to factorize each factor
Genotype = as.factor(data_new$Genotype)
Block = as.factor(data_new$Block)
X = data.frame(Genotype, Block)
formula = "~ Genotype + Block"
tt = seq(from = 0, to = 44, by = 2)
```

where X is a data frame of explanatory variables, $Y.na.mat$ is the matrix of the extracted features, and $formula$ specifies the model we use, that is, model (2) in this example. For both the explanatory data frame X and the response matrix Y, each row gives the values for different plants. The columns of X and $Y.na.mat$ correspond to explanatory variables and observation dates, respectively. If a measurement of a plant is missing on a particular date, the value is filled by "NA" in the matrix.

4.2 Model Fitting

Based on the phenotypical traits we obtained by processing images, as an example, we study the relationship between the plant sizes and treatments using a functional Anova model. Let $y_i(t)$ be the size of the i^{th} plant measured at time t, where $i = 1, ..., 9$. We treat genotype and block as fixed effects (there are 3 different genotypes and 3 blocks). We can use $\mathbf{G}_i = (G_{ik})_{k=2}^3$ as the categorical indicator of the i^{th} plant genotype. Specifically, G_{ik} is set to one if the plant has the k^{th} genotype; otherwise, $G_{ik} = 0$. Genotype 1 is treated as the baseline. Similarly, $\mathbf{P_i} = (P_{ik})_{k=2}^3$ is defined to indicate the block that the i^{th} plant belongs to, and the first block is set as the baseline. Our functional Anova model for analyzing the relationship between plant size and genotype, block is:

$$y_i(t) = \mu(t) + \sum_{k=2}^{3} G_{ik} g_k(t) + \sum_{k=2}^{3} P_{ik} p_k(t) + \varepsilon_i(t), \tag{1}$$

where $\mu(t)$ is the growth function of the plant with Genotype 1 from the 1^{st} block, $g_k(t)$s are the genotype effect functions, $p_k(t)$s are the fixed block effect functions, and the residuals $\varepsilon_i(t)$s are modeled by independent random processes with mean zeros. Let m_i be the number of days that the i^{th} plant was imaged, which could vary from plant to plant, and let t_{ij} be the j^{th} observation time for the i^{th} plant, where $j = 1, ..., m_i$. Then, our model can be described as:

$$y_i(t_{ij}) = \mu(t_{ij}) + \sum_{k=2}^{3} G_{ik} g_k(t_{ij}) + \sum_{k=2}^{3} P_{ik} p_k(t_{ij}) + \varepsilon_i(t_{ij}). \tag{2}$$

We add a penalty term on the second derivative of the estimated functions for the smoothness of the curves. The model is fit using function *fanova*:

```
fit = fanova(Y.na.mat = Y.na.mat, X = X, tt = tt,
                  formula = "~ Genotype + Block",
                  K.int = 6, order = 4)
```

More details can be found in the help documentation.

4.3 Statistical Inference

Based on the fitted model, we can draw statistical inference on the curves of genotype and block effects. For example, parameter estimation can be obtained by

```
para = fit$est_fun
```

where *para* is a t by u matrix. Here t is the number of observation time points, and u is the number of columns of the design matrix. In this case, $t = 23$, $u = 5$. The j^{th} column represents the estimated parameters of the j^{th} variable in the design matrix. For example, $\hat{\mu}(t)$ can be obtained by:

```
mu.t.hat = para[,1]
```

Afterwards, we can test the significance of the treatment effects of interest by constructing the corresponding confidence regions using the function "*CI_contrast*". For example, the confidence region of the block effect function $p_2(t)$ infers whether there is a significant difference in plant size between block 1 and block 2, given the same plant genotype. To test whether block 2 and block 1 are significantly different or not, we can use:

```
blk2_1 = CI_contrast(fit = fit, j1 = 4, j2 = 1, alpha = 0.05)
```

where *fit* is the output from "*fanova*", $j1$ and $j2$ specify the columns of the design matrix corresponds to the treatment of interest. Figure 2 (a) shows the 95% confidence region of $p_2(t)$ from day 1 to day 44. We find that the average plant size in block 2 is significantly different from that in block 1 from the 5th day to the 20th day of the experiment.

In general, we can estimate a linear combination of treatment effects and its confidence region by the following function:

```
CI(fit, L, alpha)
```

where L is a contrast vector under model (2) specifying the linear combination of the parameters of interest. This includes estimating the average growth curve of a particular genotype over all the blocks. For example, we use the following code to calculate the 95% confidence bands of the estimated mean of Genotype 1 over three blocks.

```
CI(fit = fit, L = c(1,0,0,1/3,1/3), alpha = 0.05)
```

Users can use *fit$design_mat* from the output of the function *fanova* to obtain the design matrix under this model, which will be helpful in identifying the vector *L*. The design matrix of this example is shown below:

```
(Intercept) Genotype2  Genotype3   Block2  Block3
    1           0           1         0       0
    1           0           0         0       0
    1           1           0         0       0
    1           1           0         1       0
    1           0           1         1       0
    1           0           0         1       0
    1           0           1         0       1
    1           1           0         0       1
    1           0           0         0       1
```

5 Functional ANOVA with Interaction (Example 2)

This is another example of Functional Anova model with an interaction term. In this example, we use the data presented in Xu et al. (2018). Data can be obtained by

```
data_Xu = read.csv(system.file("extdata", "data_Xu.txt",package = "implant",
                  mustWork = TRUE), sep = "")
```

The object of this example is to study the growth curve of maize (plant size) over time, and its relationship with genotypes and water treatment. There are two genotypes, FFMM-A maize and B73 maize, and two levels for the water treatment, well watered and water stressed. We also consider the interaction between genotype and water treatment. In this experiment, 60 plants were recorded daily for a total of 21 days, excluding day 16 as a result of a technical failure.

We denote $G_i = 1$ and 0 if the genotype of the ith maize is B73 and FFMM-A maize, respectively. Similarly, $W_i = 1$ means the ith maize is well watered, and 0 if the ith maize is water stressed. The model can be written as

$$y_i(t) = \mu(t) + G_i g(t) + W_i w(t) + G_i W_i \gamma(t) + \varepsilon_i(t), \tag{3}$$

where $y_i(t)$ is the area of the ith maize, $\varepsilon_i(t)$ is the random error process, $\mu(t)$ is the growth function of water stressed FFMM-A maize (baseline), $g(t)$ and $w(t)$ are genotype effect and water effects, respectively, and $\gamma(t)$ is the interaction effect. We add a penalty term on the second derivative of the estimated functions for the smoothness of the curves. The function *fanova* can perform the model:

```
water = as.factor(data_Xu$water)
genotype = as.factor(data_Xu$genotype)
X = data.frame(water,genotype)
Y = data_Xu[,c(1:20)]
fit = fanova(Y.na.mat = Y, X = X,
                  tt = c(0:15,17:20),
                  formula = "~water + genotype + water:genotype",
                  K.int = 6, order = 4)
```

We obtain the estimated functional curves for the regression coefficients using the output:

```
fit$est_fun
```

where the first, second, third and fourth columns of the *est_fun* matrix represent $\hat{\mu}(t)$, $\hat{g}(t)$, $\hat{w}(t)$, and $\hat{\gamma}(t)$, respectively. We can use "*CI*" to test for the significance of each variable. For example, we can test whether the interaction term is significant by constructing its 95% confidence interval:

```
diff = CI(fit, L = c(0, 0, 0, 1), alpha = 0.05)
trt = diff$trt
upper_bound = diff$ub
lower_bound = diff$lb
```

The argument *L* in "*CI*" can be set based on the design matrix from the output of "fanova" by

```
fit$design_mat.
```

The first six rows of the design matrix under the above model are listed below:

```
(Intercept)  water1      genotype1    water1:genotype1
    1           1           0               0
    1           0           0               0
    1           1           0               0
    1           0           0               0
    1           1           0               0
    1           0           0               0
```

It is clear that the 4^{th} column in the design matrix corresponds to the interaction term, and the argument $L = (0,0,0,1)$ gives the confidence regions for the interaction effect.

The interaction effect is the difference of the excess in plant size of B73 to FFMM-A between the well water condition and the water stress condition. The plot of the confidence region is presented in Figure 2 (c). From this graph, we are 95% confident that the true coefficient $\gamma(t)$ is within this confidence region. We also find that the confidence region contains 0, which means the interaction term is not significant over the whole time period. This implies that the treatment (watered/stressed) has the same effect on two genotypes. Therefore, we could fit the model without the interaction term and obtain the growth curve for each combination of genotype and treatment using the following code.

```
plot(tt,fit$est_fun[,1], type = "l", ylim = c(0,100000), lty = 2,col = "blue",
 xlab="Day",ylab="Plant Area (pixel)",cex.axis=1,cex.lab=1)
lines(tt,fit$est_fun[,1]+fit$est_fun[,2], lty = 2, col = "red")
lines(tt,fit$est_fun[,1]+fit$est_fun[,3], lty = 1,col = "blue")
lines(tt,fit$est_fun[,1]+fit$est_fun[,2]+fit$est_fun[,3], col = "red")
legend("topleft",col=c("red","red","blue","blue"),
lty=c(1,2,1,2),
legend =c("Watered,B73","Watered,FFMMA","Stressed,B73","Stressed,FFMMA"),
lwd=0.5,cex=0.8)
```

Those growth curves are presented in Figure 3, together with the results from Xu et al. (2018). We can see the results obtained by our package are identical to that from Xu et al. (2018).

R Code for both examples can be found in the file *"demo.R"* in the *"data-raw"* folder of the package website from the link https://github.com/rwang14/implant.

References

Xu, Yuhang, Yumou Qiu, and James C Schnable (2018), "Functional modeling of plant growth dynamics." *The Plant Phenome Journal*, 1.

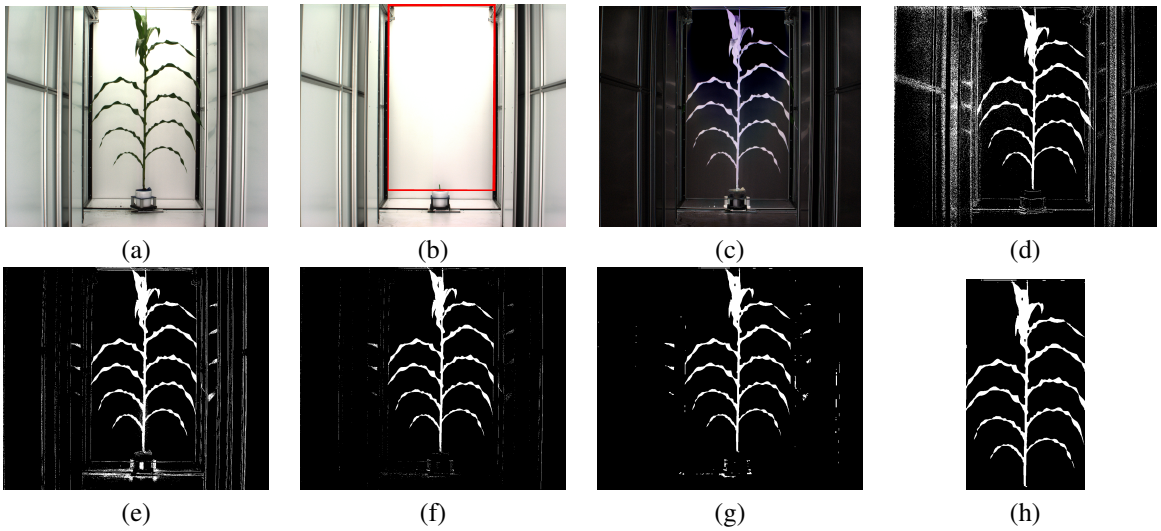


Figure 1. (a) Original plant image. (b) Original empty pot image; the red square is the identified region of interest by the functions *"ColorB"* and *"ColorG"*. (c) Contrast of (a) and (b). (d) Segmented image of (a) using DCT (e) Segmented image of (c) using DCT. (f) Intersection of (d) and (e). (g) Dilated-Eroded-Eroded-Dilated image of (f). (h) Final segmented image by identifying the region of interest.

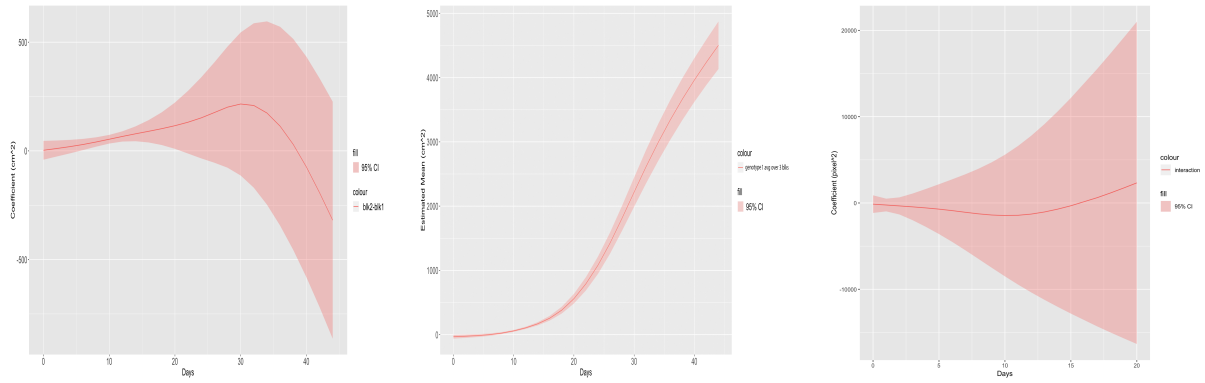
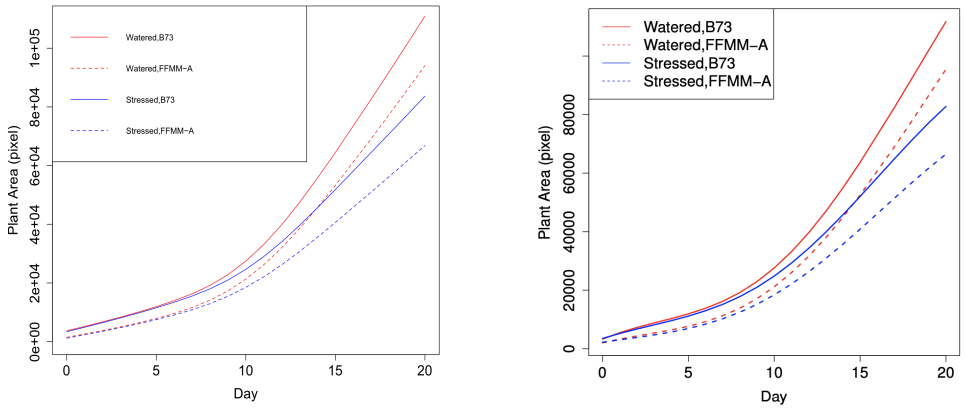


Figure 2. (a) 95% confidence bands for the coefficient of the difference between block 2 and block 1 in Example 1. (b) 95% confidence bands for the estimated mean of genotype 1 averaging over three blocks in Example 1. (c) 95% confidence bands for the interaction term of the Example 2.



(a) (b)

Figure 3. (a) Estimated growth curves for each genotype × treatment combination obtained by the package "implant". (b) Estimated growth curves for each genotype × treatment combination from Xu et al. (2018).