Package 'implant'

April 25, 2020

Type Package

Title A High-throughput Phenotyping Pipeline for Image Processing and Functional Growth Curve Analysis

Version 0.1.0

Author Ronghao Wang, Yumou Qiu, Yuzhen Zhou, Yuhang Xu, James Schnable

Maintainer Ronghao Wang <rhwang64@gmail.com>

Description The package ``implant'' is used for both image processing and functional data analysis, which is able to provide statistical inference for the plant traits directly from the input images. For image processing, the package provides methods including thresholding, hidden Markov random field model,etc. For the growth curve analysis, this package can produce nonparametric curve fitting with its confidence region for plant growth. A functional ANOVA model to test for the treatment and genotype effects of the growth curve dynamics is also provided.

Depends

Imports stats,
fda,
spatstat,
matrixcalc,
MASS

Suggests png,imager, R.rsp

VignetteBuilder R.rsp

License GPL (>= 2)

Encoding UTF-8

LazyData true

R topics documented:

`average_reducing` *Reducing size of an image using the method of averaging in blocks.*

Description

This function is used for reducing size of an image by averaging its pixels in blocks.

Usage

```
average_reducing(image, block_nrow = 2, block_ncol = 2)
```

Arguments

`image`	a pixel matrix or an array of the image for processing.
`block_nrow`	an integer number, needed to be divisible by number of rows of the pixel-arrary of the image.
`block_ncol`	an integer number, needed to be divisible by number of columns of the pixel-arrary of the image.

Details

This function is used to reduce the size of an image by dividing the original array into several blocks and calculate the average values within each block.

Value

pixel array of the reduced image.

Note

block_nrow and block_ncol must be divisible by number of rows and columns of the pixel-arrary of the image, respectively. Otherwise, Errors will be reported as: "block_nrow(block_ncol) must be divisible by number of rows(columns) of the pixel-arrary of the image."

Examples

```
average_reducing(image1,2,2)
```

CI.trt *Estimating a linear combination of treatment effects and obtaining its confidence regions*

Description

This function is used for estimating a linear combination of treatment effects and its confidence regions. Moreover, it can be used to test the coefficient of the difference between two treatments.

Usage

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

Arguments

fit	An object of output obtained by function "fanova_mean".
L	A numeric contrast vector which specifies the linear combination of the parameters of interest.
alpha	A positive small number, which is the probability of making type I error. e.g., if you want to calculate the 0.95 confidence band, take alpha = 0.05.

Value

trt	A t by 1 vector which includes the estimated mean of the treatment mean at different observation time points.
ub	A t by 1 vector which indicates the upper bound of the (1-alpha) confidence band at different time points.
lb	A t by 1 vector which indicates the lower bound of the (1-alpha) confidence band at different time points.

Examples

```
# NOT RUN {
fit = fanova_mean(Y.na.mat = Y, X = X,
                  tt = seq(from = 0, to = 44, by = 2),
                  formula = toString("~ X[,1]+X[,2]"),
                  K.int = 6, order = 4,
                  d0 = 0, d1 = 2, d2 = 2, lower = -10, upper = 15)
L1 = c(1,0,0,1/3,1/3)
ci1 = CI.trt(fit = fit, L = L1, alpha = 0.05)
plot(tt,ci1$trt,type = "l")
lines(tt,ci1$lb, col = "blue")
lines(tt,ci1$ub, col = "blue")
}
```

CI.trt.diff *Testing the significance of the treatment effects of interest by constructing the corresponding confidence regions.*

Description

This function can be used for testing the significance of the treatment effects of interest by constructing the corresponding confidence regions. In fact, it is a special case of the function "CI.trt" in this package. This function is more efficient and more easy to use when testing the difference between two treatments.

Usage

```
CI.trt.diff(fit, j1, j2, alpha)
```

Arguments

fit	An object of output obtained by function "fanova_mean".
j1	A positive integer, which indicates the columns of the design matrix corresponding to the treatment of interest.
j2	A positive integer, which indicates the columns of the design matrix corresponding to the treatment of interest.
alpha	A positive small number, which is the probability of making type I error. e.g., if you want to calculate the 0.95 confidence band, take alpha = 0.05.

Examples

```
#NOT RUN {
fit = fanova_mean(Y.na.mat = Y, X = X,
                  tt = seq(from = 0, to = 44, by = 2),
                  formula = toString("~ X[,1]+X[,2]"),
                  K.int = 6, order = 4,
                  d0 = 0, d1 = 2, d2 = 2, lower = -10, upper = 15)

ci_diff = CI.trt.diff(fit = fit,j1 = 5, j2 = 4, alpha = 0.05)
plot(tt,ci_diff$trt,type = "l")
lines(tt,ci_diff$lb, col = "blue")
lines(tt,ci_diff$ub, col = "blue")
}
```

Color2Gray *Converting RGB to Grayscale.*

Description

This function is used to convert an RGB image to Grayscale.

Usage

```
Color2Gray(image, weight = c(0.299,0.587,0.114))
```

Arguments

image	an array of an RGB image file for processing.
weight	a numeric vector, in which elements are weights for the Red channel of the image, Green channel of the image, and the Blue channel of the image, respectively.

Value

image	A matrix of pixels of the image converted from RGB to Grayscale.

Examples

```
#read an RGB image
library(png)
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,c(1:3)]
imageGray = Color2Gray(image)
```

ColorB *Identifying the region of interest*

Description

This function is used to identify the region of interest of an image. Specifically, this function is designed to identify the region of interest of the plant image taken from the UNL greenhouse system, by removing the parts of the image that contains the black bars, but keep the black part of the pot. In a word, to help users identify the region of interest.

Usage

```
ColorB(image, colThreshold = 0.5, colTol = c(5, 5), changefromub = rep(0.1, 3),
changeto = c(1, 1, 1))
```

Arguments

image	matrix of pixels of the image file for processing.
colThreshold	positive real number, which is the threshold level of the signals of the black bars appear in each column.
colTol	A 2 by 1 numeric vector, in which elements are some small numbers. Essentially, this argument is to add the tolarance when identifying the boundaries of the region of interest.
changefromub	A 3 by 1 numeric vector, in which elements refer to the threshold levels of the black bars of the image for the Red Channel, the Green Channel and the Blue Channel, respectively.
changeto	A 3 by 1 numeric vector, in which elements refer to the pixel intensities of the colors you want each channel of the eliminated part changed to, respectivley.

Value

lb	left bound of the part of the region of interest.
rb	right bound of the part of the region of the interest.
c	matrix of pixels of the obtained image by applying this function.

Examples

```
#read an RGB image
library(png)
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,c(1:3)]
resultColor = ColorB(image)
bound = c(resultColor$lb, resultColor$rb)
imageColor = resultColor$c
```

ColorG *Identifying the region of interest & changing color of particular part of an image*

Description

This function is used to identify the region of interest, and change the color of particular part of an image. Specifically, this function is designed for the plant images taken by the UNL greenhouse system. It helps identify the lower boundary of the region of interest and change the color of the green strip of an empty pot from green to white.

Usage

```
ColorG (imagefile, rowThreshold = 0.007, block = 5, Tol = 2, Bthreshold = 60 / 255,
EGThreshold = 0.1, weight = c(-1, 2, -1), changeto = c(1, 1, 1))
```

Arguments

imagefile	matrix of pixels of the image for processing.
rowThreshold	positive real number, which is the threshold value of each row to filter the black
block	postive integer, which is the moving average of average pixels of each row of the matrix of the matrix.
Tol	small number. Essentially, this argument to add the tolarance when identifying the lower boundary of the region of interest.
Bthreshold	Value between 0 and 1. It is applied to the sum of the RGB intensities.
EGThreshold	Value between 0 and 1. It is applied to the contrast intensity by the specified weight in the function.
weight	3 by 1 numeric vector. The three elements indicate the weight of the pixel intensities of R,G,B, respectively. In default, it takes the value of c(-1,2,-1), which helps to construct a relative green ratio.
changeto	numeric vector, in which elements refer to the pixel intensities of the colors you want each channel of the eliminated part changed to, respectivley.

Details

In the example part, this function helps identify the lower boundary of the region of interest, and eliminate the green strip of an empty pot by changing its color from green to white.

Value

uppb	upper bound of the region of interest.
lowb	lower bound of the region of interest.
rowmean	proportion of the signals of the green strip appear in each row of pixel matrix of the image.
c	matrix of pixels of the processed image.

Examples

```
#read an RGB image
library(png)
imageX = readPNG(system.file("extdata", "image_pot.png",
                             package = "implant",
                             mustWork = TRUE))[,,c(1:3)]
tempG = ColorG (imagefile = imageX, rowThreshold = 0.007, block = 5, Tol = 2,Bthreshold = 60 / 255,
EGThreshold = 0.1, weight = c(-1, 2, -1), changeto = c(1, 1, 1))
tempG$c
tempG$lowb
```

Color_Exchange *Exchanging the color of the background and the foreground.*

Description

This function exchanges the color of the background and the subject for a binary image.

Usage

```
color_exchange(image1)
```

Arguments

image1	A pixel matrix of a binary image

Details

The input matrix should be a matrix of 0 and 1 only.

Value

A pixel matrix with exchanging the color of the background and subject.

Examples

```
color_exchange(image1)
```

dilation *Morphological Dilation*

Description

This function is used to perform morphological dilation of an image.

Usage

```
dilation(imagefile, mask = matrix(1, 3, 3))
```

Arguments

imagefile	array or matrix of pixels of the original image.
mask	matrix constructed by structuring elements.

Value

matrix of pixels of the dilated image.

Examples

```
#read an RGB image
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,c(1:3)]

imageB = imageBinary(image, weight = c(-1, 2, -1),
                     threshold1 = 30 / 255,
                     threshold2 = 0.075)

imageBD = dilation(imageB, mask = matrix(1, 5, 5))
```

erosion *Morphological Erosion*

Description

This function is used to perform morphological erosion of an image.

Usage

```
erosion(imagefile, mask = matrix(1, 3, 3))
```

Arguments

imagefile	matrix of pixels of the original image.
mask	matrix constructed by structing elements.

Value

matrix of pixels of the eroded image.

Examples

```
#read an RGB image
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,c(1:3)]

imageB = imageBinary(image, weight = c(-1, 2, -1),
                     threshold1 = 30 / 255,
                     threshold2 = 0.075)

imageBE = erosion(imageB, mask = matrix(1, 5, 5))
```

`extract_pheno` *Extract phenotypical features from segmented images*

Description

This function extract phenotypical features from segmented images.

Usage

```
extract_pheno(processed_image, Xsize = 1, Ysize = 1, a = 1, b = 1)
```

Arguments

`processed_image`	a binary matrix contains only 0 and 1. i.e., the segmented image of a plant.
`Xsize, Ysize`	Xsize and Ysize are the actual horizontal and vertical lengths of one pixel, respectively. If users prefer to calculate in pixel level rather than acutual units, use the default set, i.e. Xsize = 1, Ysize = 1.
`a,b`	positive integers. To be more specific, a and b are respectively the same as the values of RowSample and ColSample in function: "sample". This is only used when you have used function "sample" to reduce the size of the image in image segmentation.

Value

`plantheight`	The height of the segmented plant by taking the 2.5th% quantile and 97.56th% qunatile of the segmented pixels of interest.
`plantwidth`	The width of the segmented plant by taking the 2.5th% quantile and 97.56th% qunatile of the segmented pixels of interest.
`plantSize`	The size of the segmented plant based on the total number of pixels of the segmented plant of interest.
`pixelCount`	The total number of pixels of the segmented plant of interest.

See Also

reducingsize for reducing size of an image.

Examples

```
#reduce the size of an image
image_reduced = sample(original_image, RowSample = 2, ColSample = 2)
#segment the reduced image
imageBD = imageBinary(image_reduced,  weight = c(-1, 2, -1), threshold1 = 30 / 255, threshold2 = 0.05)
#obtain the size of the segmented image
extract_pheno(processed_image = imageBD, Xsize = 1.5, Ysize = 1.5, a = 2, b = 2)$size
```

---

fanova_mean *Fitting Functional ANOVA Models*

Description

This function is used for fitting fanova models using B-Spline basis expansion. The model performed by this function is constructed by a general functional model plus a penalty term for the second derivative of the spline basis function. The estimated penalty parameter, labmda, can be obtained by GCV (generalized cross validation).

Usage

```
fanova_mean(Y.na.mat, X, tt, formula, K.int = 6,
            order = 4, lower = -10, upper = 15)
```

Arguments

Y.na.mat	A n by t matrix of response variable (the extracted features), where n is the number of observations (for example, in the study of plant growth curve, each plant id would be an observation), and t is the number of observation time points. In fact, the value of t should match the length of another argument shown below called "tt". If a measurement of an observed object (e.g., the height of a plant) is missing on a particular date shown in tt, the value should be filled by "NA" in the Y.na.mat matrix. In a word, the ith row of this matrix represents the values of the response varable of the ith observation at t different observation time points.
X	A n by r dataframe of explanatory variables, where n is the number of observations and r is the number of explanatory variables.
tt	A t by 1 vector, in which elements imply the observation days.
formula	An object of class "formula", which specifies the model to use.
K.int	A positive integer, which refers to the number of interior knots.
order	A positive integer, which refers to the order of the polynomial.
lower	Lower bound of the possible values of the smoothing parameter, lambda.
upper	Upper bound of the possible values of the smoothing parameter, lambda.

Details

This function can help users perform functional anova models, by assuming that each variable is a continuous function (in general, a function of time t). We use B-Spline as the basis function to expand each variable.

For example, the intercept term of a functional anova model, mu(t), will be expanded as

mu(t) = Sigma_{j = 1 to K}beta_{mu,j}B_{u,j}(t),

where beta_{mu,j} is a coefficient and B_{u,j}(t) is an order u B-spline basis function. u is defined by the argument "order" in this function. With another argument "K.int" (i.e., the number of interior knots) being defined, we can know the value of K, the rank of spline expansion for mu(t), according to the relationship K = K.int + order.

Value

`est_fun` A t by q matrix of the estimates of the functional parameters, where t is the number of observation time points, and q is the number of columns of the design matrix. The ijth element represents the estimated functional parameter of the jth variable of the design matrix at the ith observation time point.(i.e., bhat_j(t_i)).

For example, if your model includes two categorical variables without interaction terms, each categorical variable contains two levels. Your model will be: y(t) = b_0(t)+b_1(t)X1+b_2(t)X2+e(t). In this case, q = 3. This means your output "est_fun" will be constructed by 3 columns, and the columns repesent b_0hat(t), b_1hat(t) and b_2hat(t), respectively.

`bhat` A q by K numeric vector, in which contains all the estimated parameters (i.e., betahat), where q is the number of columns of the design matrix, and K is the rank of the spline expansion.

`lambda` The estimated penalty parameter, obatined using GCV.

Examples

```
#load the data
data_new = read.csv(system.file("extdata","data.csv",
                                package = "implant", mustWork = TRUE))
Y = data_new[,-c(1:3)]
#This step is to factorize each factor
X1 = as.factor(data_new$Genotype)
X2 = as.factor(data_new$Block)
X = data.frame(X1,X2)
formula = "~ X[,1]+X[,2]"
tt = seq(from = 0, to = 44, by = 2)
fit = fanova_mean(Y.na.mat = Y, X = X, tt = tt, formula,
                  K.int = 6, order = 4, lower = -10, upper = 15)
fit$est_fun
```

`HMRF_EM` *Image Segmentation using Hidden Markov Random Field and EM Algorithm Framework*

Description

This function can be used to obtain the segmented image using HMRF-EM.

Usage

```
HMRF_EM(X,Y,Z,mu,sigma,k,em_iter,map_iter, beta = 2,
        epsilon_em = 0.00001, epsilon_map = 0.00001)
```

Arguments

X	A matrix of the inital labelled image, which can be obtained using other segmentation methods, such as K-means.
Y	A m by n matrix of the response variable.
Z	A m by n matrix. This is to have a brief edge detection of the response variable. e.g., We can obtain Z using the Canny edge detector: Z = cannEdges( )[,,1,1] from the package: imager. See the example for details.
mu	A k by 1 matrix. The k rows represent the mean of the pixel intensities of the response variable under k different classes, respectively. This be obtained by another function from this package, called imkmeans( ). See the example for details.
sigma	A k by 1 matrix. The k rows represent the standard deviation of the pixel intensities of the response variable under k different classes, respectively. This can be obtained by another function from this package, called imkmeans( ). See the example for details.
k	A positive integer, which infers the number of classes that you want to classify the image into. In default, k = 2.
em_iter	A positive integer, which is the number of iteration steps of EM Algorithm.
map_iter	A positive integer, which is the number of iteration steps of MAP.
beta	The clique potential parameter, in default, beta = 2. See more in the supplementary file of HMRF Model.
epsilon_em	A small positive number, which is the convergence criterion of EM Alogrithm.
epsilon_map	A small positive number, which is the convergence criterion of MAP.

Details

The arguement Z can be obatined by CannyEdge detector using function cannEdges( ) from the package: imager. However, since this package needs to involve Rcpp and other dependent packages which may increase installation complexity of our package: implant. Considering not every user needs to use this function HMRF_EM( ), we recommend the users to install the package: imager by themselves if they need.

Value

image_matrix	A matrix of the segmented image.
mu	Updated mu.
sigma	Updated sigma.

Note

This function is modified based on the matlab code written by Quan Wang (see reference).

References

Wang, Quan (2012), "Hmrf-em-image: implementation of the hidden markov random field model and its expectation-maximization algorithm."arXivpreprintarXiv:1207.3510

Examples

```
library(imager)
orig1 = load.image(system.file("extdata", "image1.png",
                                    package = "implant", mustWork = TRUE))
orig = resize(orig1, size_x = 500, size_y = 500, size_z = 1, size_c = 3)
#Define the response as relative green.
I = orig[,,1,2]/(orig[,,1,1]+orig[,,1,2]+orig[,,1,3])
Y = t(I)
Z = cannyEdges(orig)
Z = Z[,,1,1]
Z = t(Z)
#Take the initial label of EM algorithm using K-means
X = imkmeans(Y,k = 2)$X
mu = imkmeans(Y,k = 2)$mu
sigma = imkmeans(Y,k = 2)$sigma
output = matrix(as.numeric(X),nrow = nrow(X), ncol = ncol(X))-1
#Run the HMRF Model
img = HMRF_EM(X,Y,Z,mu,sigma,k = 2,em_iter = 20,map_iter = 20,beta = 2,
              epsilon_em = 0.00001, epsilon_map = 0.00001)
#Obtain the matrix of the segmented image
image = img$image_matrix
#Morphological Operations
imageD = dilation(image)
imageDE = erosion(imageD)
imageDEE = erosion(imageDE)
imageDEED = dilation(imageDEE)
```

`imageBinary` *Segmentation and Binarization*

Description

This function uses the Double-Criterion Thresholding method (DCT) to segment the object of study from the background of an image, and tranform the image to a binary image, i.e., a black and white image.

Usage

```
imageBinary(image, weight = c(-1, 2, -1), threshold1 = 30 / 255,
threshold2 = 0.075)
```

Arguments

`image`	an array of pixels of the image for processing.
`weight`	a 3 by 1 vector. The three elements indicate the weight of the pixel intensities of R,G,B, respectively. In default, it takes the value of c(-1,2,-1), which helps to construct a relative green ratio.

`threshold1,threshold2`
: Values between 0 and 1. threshold1 is applied to the sum of the RGB intensities. threshold2 is applied on the contrast intensity by the specified weight in the function.

Details

In processing the green plants images taken in the greenhouse system, the two thresholds are to delete the black pixels and to segment the plant green pixels,respectively.

Value

A matrix of pixels of the processed image.

Examples

```
#read an RGB image
library(png)
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,c(1:3)]
imageB = imageBinary(image,  weight = c(-1, 2, -1),
threshold1 = 30 / 255, threshold2 = 0.05)
```

---

imkmeans *Obtain the Matrix of the Segmented Image using K-means Method.*

Description

This function is to obtain the Matrix of the Segmented Image using K-means Method, togehter with the means and variances of the pixel intensities within different classes.

Usage

```
imkmeans(Y,k)
```

Arguments

`Y`

`k`

Value

`X` A matrix of the segmented image, using the K-means method.

`mu` A k by 1 matrix. The k rows represent the mean of the pixel intensities of the response variable under k different classes, respectively.

`sigma` A k by 1 matrix. The k rows represent the standard deviation of the pixel intensities of the response variable under k different classes, respectively.

Examples

```
install.packages("imager")
library(imager)
orig1 = load.image(system.file("extdata", "image1.png",
                                        package = "implant", mustWork = TRUE))
orig = resize(orig1, size_x = 500, size_y = 500, size_z = 1, size_c = 3)
I = orig[,,1,2]/(orig[,,1,1]+orig[,,1,2]+orig[,,1,3])
Y = t(I)
k = 2
X = imkmeans(Y,k)$X
mu = imkmeans(Y,k)$mu
sigma = imkmeans(Y,k)$sigma
output = matrix(as.numeric(X),nrow = nrow(X), ncol = ncol(X))-1
```

`sample` *Reducing size of an RGB image*

Description

This function reduces the size of an image by picking sample pixels from each row and each column of the original image, and use the selected pixels to construct a reduced image.

Usage

```
sample(image, RowSample = 1, ColSample = 1)
```

Arguments

`image`	an array of pixel intensities of image to reduce.
`RowSample`	a positive integer. It is the increment of the index of the selected elements (pixel intensities). For example, RowSample = 2 means you select the 1st, 3rd, ... element from each row of the original image to construct the new image.
`ColSample`	a positive integer. It is the increment of the index of the selected elements (pixel intensities). For example, ColSample = 2 means you select the 1st, 3rd, ... element from each column of the original image to construct the new image.

Details

This function can be used to reduce the size by picking sample pixels of the original image as the pixels of the reduced image. For example, "RowSample = 2, and ColSample = 2"" means that we pick the 1st, the 3rd, the 5th,..., pixel of each row and each column of the original image as the pixels of the reduced image.

Value

array of pixels of the reduced image.

Note

This function is different from another function in this package called "sampleMatrix( )". That function is used to reduce the size of matrices while this function, sampleMatrix( ), is used to reduce the size of 3-D arrays.

Examples

```
#read an RGB image
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,c(1:3)]
sample(image, RowSample = 2, ColSample = 2)
```

`sampleMatrix` *Reducing the Size of a Matrix*

Description

This function reduces the size of a matrix by picking sample elements from each row and each column of the original matrix, and use the selected elements to construct a reduced matrix. This function can be used to reduce sizes of binary images in the study of image processing.

Usage

```
sampleMatrix(M, RowSample = 1, ColSample = 1)
```

Arguments

`M`	the original matrix to reduce.
`RowSample`	a positive integer. It is the increment of the index of the selected elements. For example, RowSample = 2 means you select the 1st, 3rd, ... element from each row of the original matrix to construct your new matrix.
`ColSample`	a positive integer. It is the increment of the index of the selected elements. For example, ColSample = 2 means you select the 1st, 3rd, ... element from each column of the original matrix to construct your new matrix.

Details

This function can be used to reduce the size of a matrix by picking sample elements of the original matrix as the elements of the reduced matrix. For example, "RowSample = 2, and ColSample = 2"" means that we pick the 1st, the 3rd, the 5th,..., element of each row and each column of the original matrix as the element of the reduced matrix.

Value

the reduced matrix.

Note

This function is different from another function in this package called "sample( )". That function is used to reduce the size of 3-D arraysm while this function, sampleMatrix( ), is used to reduce the size of matrices.

Examples

```
#read the red band of an RGB image
image = readPNG(system.file("extdata", "image1.png",
                            package = "implant",
                            mustWork = TRUE))[,,1]
sampleMatrix(M = image, RowSample = 2, ColSample = 2)
```

Index