

# Package ‘ReadImages’

January 2, 2012

**Version** 0.1.3.1

**Date** 2009-07-18

**Title** Image Reading Module for R

**Author** Markus Loecher

**Maintainer** Markus Loecher <markus@sensenetworks.com>

**Depends** R (>= 1.6)

**Description** This package provides functions for reading JPEG and PNG files. This package requires libjpeg <<http://www.iijg.org>>. This package can be used on Unixes / MacOS X / Windows.

**License** BSD

**Repository** CRAN

**Date/Publication** 2009-07-30 12:51:52

## R topics documented:

ReadImages-package . . . . .	2
clipping . . . . .	2
imagematrix . . . . .	3
imageType . . . . .	4
logo . . . . .	5
normalize . . . . .	5
plot.imagematrix . . . . .	6
print.imagematrix . . . . .	7
read.jpeg . . . . .	7
rgb2grey . . . . .	8

<b>Index</b>	<b>9</b>
--------------	----------

---

ReadImages-package      *Utilities to read in various image formats into R.*

---

### Description

This package provides functions for reading JPEG (and PNG in a later version) files. This package requires libjpeg <<http://www.iijg.org>>.

### Details

Package:	ReadImages
Type:	Package
Version:	0.1.3.1
Date:	2009-07-18
License:	GPL
LazyLoad:	yes

This set of functions is really just a subset of the rimage package. We extracted only those functions essential to reading jpeg files. The dependency on the libjpeg library is still there, while the requirement to have fftw installed has vanished.

### Author(s)

Markus Loecher, Sense Networks <[markus@sensenetworks.com](mailto:markus@sensenetworks.com)>

### Examples

```
x <- read.jpeg(system.file("data", "Rlogo.jpg", package="ReadImages"))
plot(x)
```

---

clipping      *Clipping image*

---

### Description

This function returns the image which restricts pixel value from the specified lowest value to the specified highest value in the original image. This means that the pixels which have lower value than the given lowest (default: 0) are replaced to the lowest and the pixels have greater value than the given highest (default: 1) are replaced to the highest.

### Usage

```
clipping(img, low=0, high=1)
```

**Arguments**

img	target image
low	lowest value
high	highest value

**Value**

Data of the same mode as 'img'

**Examples**

```
data(logo)
op <- par(mfrow=c(2,2))
plot(logo, main="Source Image")

# the appearance of next one doesn't change because of normalization.
plot(normalize(2*logo), main="Doubled pixel value with normalization")

# the next one is saturated as expected
plot(clipping(2*logo), main="Doubled pixel value with clipping")
```

---

imagematrix	<i>Generate an imagematrix, i.e. primary data structure of rimage</i>
-------------	---

---

**Description**

This function makes an imagematrix object from a matrix. This data structure is primary data structure to represent image in rimage package.

**Usage**

```
imagematrix(mat, type=NULL,
            ncol=dim(mat)[1], nrow=dim(mat)[2], noclipping=FALSE)
```

**Arguments**

mat	array, matrix or vector
type	"rgb" or "grey"
ncol	width of image
nrow	height of image
noclipping	TRUE if you disable automatic clipping. See details.

**Details**

For grey scale image, matrix should be given in the form of 2 dimensional matrix. First dimension is row, and second dimension is column.

For rgb image, matrix should be given in the form of 3 dimensional array (row, column, channel). `mat[,1]`, `mat[,2]`, `mat[,3]` are red plane, green plane and blue plane, respectively.

You can omit 'type' specification if you give a proper array or matrix. Also, if you give a rgb image matrix and specify "grey" as type, the rgb image matrix is automatically converted to a grey scale image.

This function automatically clips the pixel values which are less than 0 or greater than 1. If you want to disable this behavior, give 'noclipping=TRUE'.

The major difference between `imagematrix` and `pixmap` is representation method. `pixmap` (>0.3) uses OOP class. On the other hand, `rimage` uses traditional S class. The advantage of traditional S class in representing image is that one can deal with the data structure as an ordinary matrix.

The minor difference between `imagematrix` and `pixmap` is automatic data conversion behavior. `pixmap` normalizes a given matrix automatically if any element of the matrix is out of range between 0 and 1. On the other hand, `imagematrix` clips the matrix, which means that the pixels which have lower value than 0 are replaced to 0 and the pixels have greater value than 1 are replaced to 1.

**Value**

return an `imagematrix` object

**See Also**

[plot.imagematrix](#), [print.imagematrix](#)

**Examples**

```
p <- q <- seq(-1, 1, length=20)
r <- 1 - outer(p^2, q^2, "+") / 2
plot(imagematrix(r))
```

---

imageType

*Get information on color type of imagematrix*

---

**Description**

This function returns color type ("rgb" or "grey") of a given `imagematrix`.

**Usage**

```
imageType(x)
```

**Arguments**

x                    target image

**Value**

"rgb" or "grey"

**Examples**

```
x <- read.jpeg(system.file("data", "Rlogo.jpg", package="ReadImages"))
cat("Image Type", imageType(x))
```

```
x.grey <- rgb2grey(x)
cat("Image Type", imageType(x.grey))
```

---

logo

*R logo imagematrix*

---

**Description**

The imagematrix object of R logo of the size 101x77.

**Usage**

```
data(logo)
```

**Format**

imagematrix

**Examples**

```
data(logo)
plot(logo)
```

---

normalize

*Normalization for vector and matrix*

---

**Description**

This function normalizes image so that the minimum value is 0 and the maximum value is 1.

**Usage**

```
normalize(img)
```

**Arguments**

img                    target image

**Value**

Data of the same mode as 'img', in which minimum value is 0 and maximum value is 1.

**Examples**

```
data(logo)
plot(normalize(logo))
```

---

plot.imagematrix      *Plotting an imagematrix object*

---

**Description**

This function outputs an imagematrix object as an image.

**Usage**

```
## S3 method for class 'imagematrix'
plot(x, ...)
```

**Arguments**

x	target image
...	plotting options

**See Also**

[imagematrix](#)

**Examples**

```
op <- par(mfrow=c(1,2))

data(logo)
plot(logo, main="plot(logo)")
plot(logo^2, main="plot(logo^2)")

par(op)
```

---

`print.imagematrix`      *Print information on a given imagematrix object*

---

### **Description**

This function outputs information on a given imagematrix object.

### **Usage**

```
## S3 method for class 'imagematrix'  
print(x, ...)
```

### **Arguments**

<code>x</code>	target image
<code>...</code>	ignored (dummy)

### **See Also**

[imagematrix](#)

### **Examples**

```
data(logo)  
print(logo)
```

---

`read.jpeg`      *Read JPEG file*

---

### **Description**

This function reads a jpeg image file and return an imagematrix object.

### **Usage**

```
read.jpeg(filename)
```

### **Arguments**

<code>filename</code>	filename of JPEG image
-----------------------	------------------------

### **Value**

return an imagematrix object

**See Also**[imagematrix](#)**Examples**

```
x <- read.jpeg(system.file("data", "Rlogo.jpg", package="ReadImages"))
plot(x)
```

---

**rgb2grey***Convert color imagematrix to grey imagematrix*

---

**Description**

This function convert color imagematrix to grey imagematrix.

**Usage**

```
rgb2grey(img, coefs=c(0.30, 0.59, 0.11))
```

**Arguments**

<code>img</code>	target image
<code>coefs</code>	coefficients for red plane, green plane, and blue plane.

**Value**

grey imagematrix

**Examples**

```
x <- read.jpeg(system.file("data", "Rlogo.jpg", package="ReadImages"))
plot(rgb2grey(x))
```

# Index

## \*Topic **datasets**

logo, [5](#)

## \*Topic **misc**

clipping, [2](#)

imagematrix, [3](#)

imageType, [4](#)

normalize, [5](#)

plot.imagematrix, [6](#)

print.imagematrix, [7](#)

read.jpeg, [7](#)

rgb2grey, [8](#)

## \*Topic **package**

ReadImages-package, [2](#)

clipping, [2](#)

imagematrix, [3](#), [6–8](#)

imageType, [4](#)

logo, [5](#)

normalize, [5](#)

plot.imagematrix, [4](#), [6](#)

print.imagematrix, [4](#), [7](#)

read.jpeg, [7](#)

ReadImages (ReadImages-package), [2](#)

ReadImages-package, [2](#)

rgb2grey, [8](#)