Image Processing using

Xiaocun Sun
Research Computing Support (RCS)|OIT
xsun@utk.edu |865-974-7505|524-A Greve Hall

Research Computing Support

- Offer up to 10 free hours of research assistance each semester to students, faculty and staff.

- Assistance includes:
  - SPSS, SAS, MATLAB, Maple, R, …
  - Research planning, design, sample size, …
  - Data entry, management, file conversion
  - Statistics, mathematics, graphics, data mining, text analysis
  - Web survey design and deployment
  - Optical mark scanning &scoring of tests and surveys

How to Reach Us

OIT Helpdesk: 865-974-9900

5th Floor, Greve Hall

Resources for R

- How to access to R
  - Download R software and R packages http://cran.r-project.org
  - Use R on UT server: apps.utk.edu
- Where to learn find support
  - RCS R workshops oit.utk.edu/training
  - Contact Research computing support: 865-974-9900
  - UCLA R starter http://www.ats.ucla.edu/stat/r/sk
  - Quick R http://www.statmethods.net

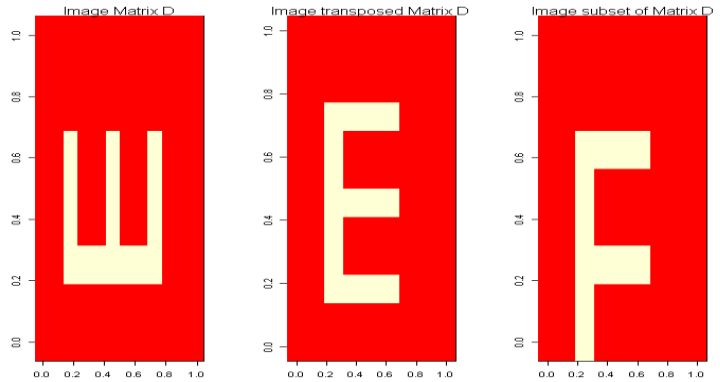
Image processing in R

- R is a comprehensive statistical environment and programming language for professional data analysis and graphical display. (CRAN Task View: Medical Image Analysis)

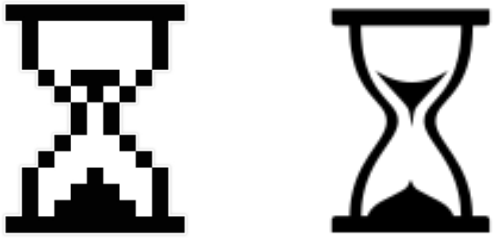
- Bioconductor project, which is based primarily on R, provides many additional R packages for statistical data analysis in different life science areas, such as tools for microarray, next generation sequence and genome analysis. http://www.bioconductor.org/

Fundamentals of Digit Image Processing

- A digital image is nothing more than data
- A 2D image can be defined as a bi-dimensional function f(x; y), where x and y are the spatial coordinates, and the amplitude at a coordinate (x; y) is called the intensity of the image



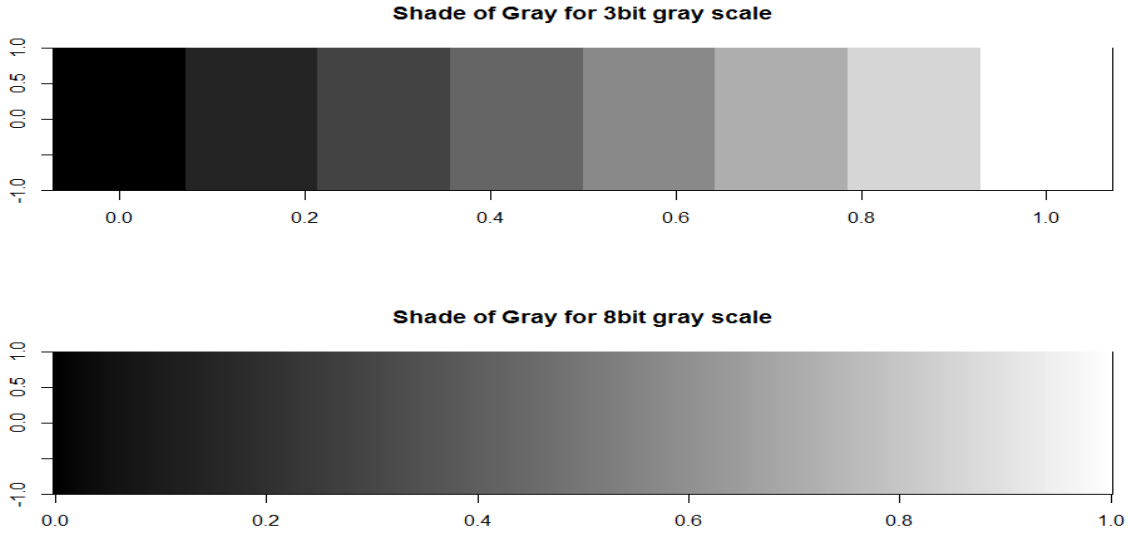
- **The number of pixels on a screen (dpi-dots per inch) define the resolution**



```
D<-matrix(rep(0, 108), 12, 9)
B<-matrix(rep(c(0,0,1),3), 3,3)
C<-t(B)
D[4:6, 1:3]=C
D[7:9, 1:3]=C
D[1:3, 4:6]=B
D[4:6, 4:6]=B
D[7:9, 4:6]=B
D[3,3]=1 #a matrix ready to display as an image
par(mfrow=c(1,3))
image(D)
mtext("Image Matrix D as is")
image(t(D))
mtext("Image transposed Matrix D")
F<-t(D)
image(F[1:9, 4:12])
mtext("Image subset of Matrix D")
```

Fundamentals of Digit Image Processing

- **Bit(binary digit) depth is the number of bits used to describe each pixel. The greater the number of bits per pixel, the better the image.**
- **An 8 bit image is 2x2x2x2x2x2x2x2= 256 shades of gray or colors.**

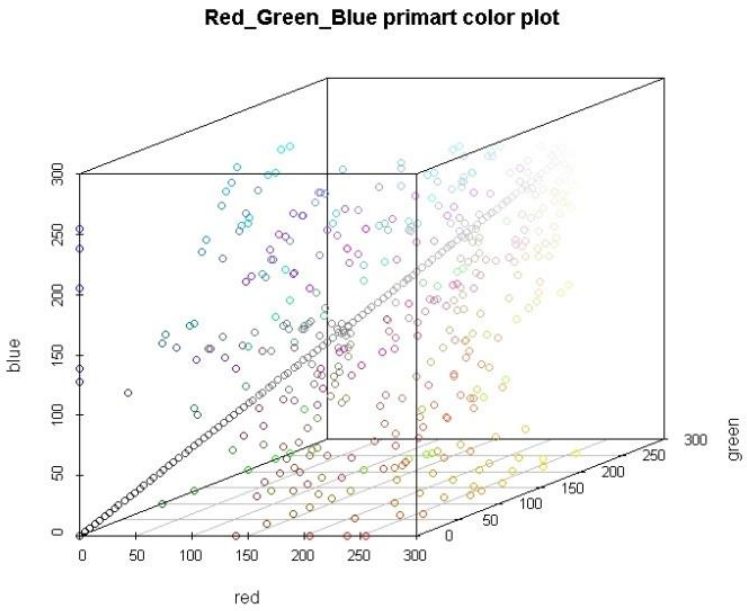



```
##How about details? Shade of Gray and pixel depth
X<-matrix(seq(from = 8, to = 1, by = -1), 1,8)
lab.palette<-colorRampPalette(c("white", "black"), space="Lab")
image(t(X),col=lab.palette(8), main="Shade of Gray for 3bit gray scale")
X<-matrix(seq(from = 255, to = 0, by = -1), 1,256)
image(t(X),col=lab.palette(256),  main="Shade of Gray for 8bit gray scale")
```

Fundamentals of Digit Image Processing

- The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.
- Information about Chart of r colors:

http://research.stowers-institute.org/efg/R/Color/Chart/




```
#color plot
library(scatterplot3d)
?scatterplot3d
##example 6 a) The named colors in R, i.e.
colors()
cc <- colors()
crgb <- t(col2rgb(cc))
par(xpd = TRUE)
rr <- scatterplot3d(crgb, color = cc, box = TRUE,
angle = 20, xlim = c(0, 256), ylim = c(0, 256), zlim
= c(0, 256), main="Red_Green_Blue primart
color plot")
```

Package EBImage(Bioconductor)

- Load EBImage
- General image processing

```
#load EBImage
source("http://bioconductor.org/biocLite.R")
biocLite("EBImage")
library(EBImage)
```

clown

Clown subset

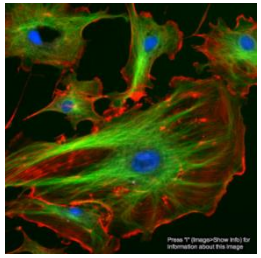
Clown>0.5

Clown flip

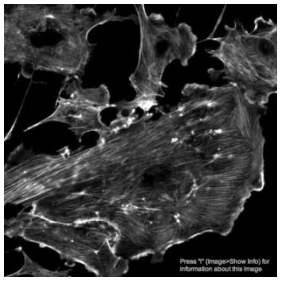
```
#load image files
myimage= readImage('angle.jpg')
#display mages
display(myimage)
#save  images
writeImage(m, 'm.jpeg', quality=100)

clown2 = clown[150:240, 55:110]
clown22 = resize(clown2, 320,200)#zooming
clown3 = clown>0.5
clown4= flip(clown)
clown5= flop(clown)
clown6= rotate(clown,180)
clowncomb1 = combine(clown, clown22,clown3,clown6)
display(clowncomb1)
```

Package EBImage(Bioconductor)

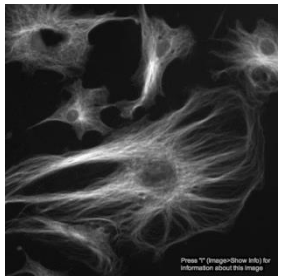
- General color management



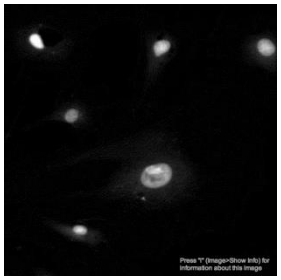
FluorescentCells



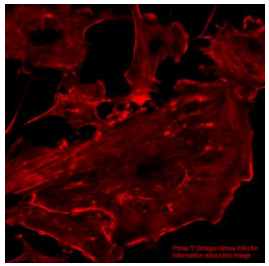
Red Channel

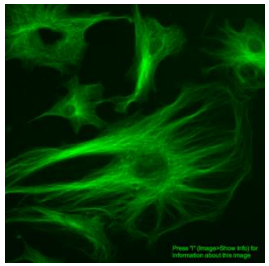


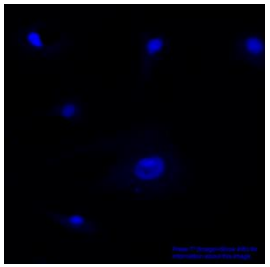
Green Channel



Blue Channel







```
#load image files
fc = readImage('FluorescentCells.png')
print(fc)
display(fc)

#general  color management
fcg = channel(fc, "asgreen")
fcr = channel(fc, "asred")
fcb = channel(fc, "asblue")
channel_c=combine(fcr, fcg, fcb)
display(channel_c)
cell_red=rgbImage(red=fcr )
cell_green=rgbImage(green=fcg )
cell_blue=rgbImage(blue=fcb )
cell_re=combine(cell_red, cell_green, cell_blue)
display(cell_re)
```

Package EBImage(Bioconductor)

- General image processing tools

clownc

```
clownc = readImage('clownc.png')
display(clownc)
```

High pass filter

```
fhi = matrix(1, nc=3, nr=3)
fhi[2,2] = -8
clown_sharp = filter2(clownc, fhi)
display(clown_sharp)
```

Blur

```
clown_blur=gblur(clownc, s=10)
display(clown_blur)
```

Contrast

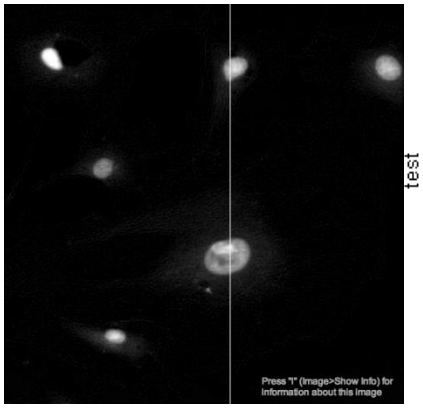
```
display(clownc*3)
```

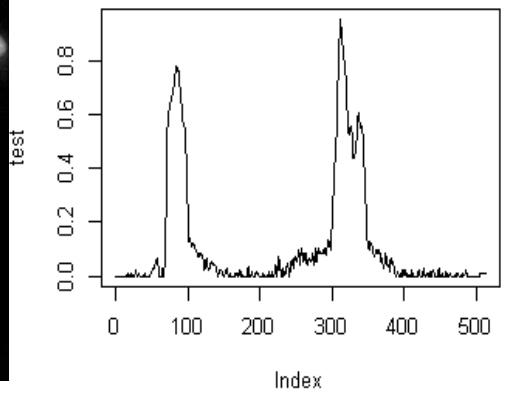
Brightness

```
display(clownc+0.3)
```

Package EBImage(Bioconductor)

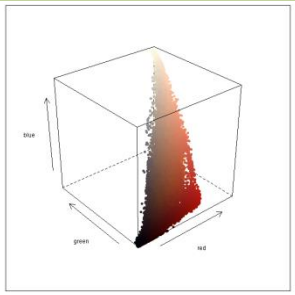
- Intensity measurement

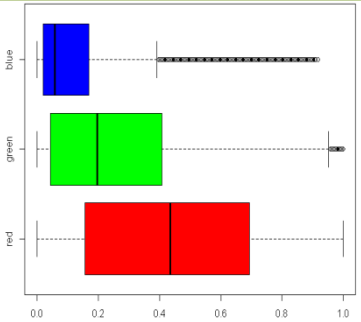


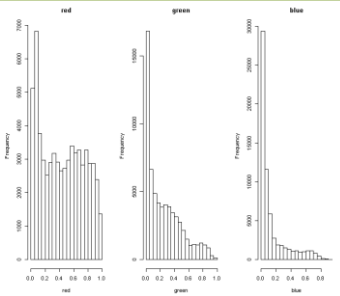



```
test=fcb[290, 1:512]#green channel of the fluroscence cell image
plot(test, type="l")#plot profile
fcb[290, 1:512]=1
display(fcb)
```

```
clownc_data=data.frame(red=as.vector(clownc[, ,1]),green=as.vector(clownc[, ,2]), blue=as.vector(clownc[, ,3]))
cloud(blue~red*green, col=rgb(red,green, blue), pch=19)
boxplot(clownc_data, horizontal=T, col=c("red", "green", "blue"))
hist(red, main="red")  hist(green, main="green")     hist(blue, main="blue")
```

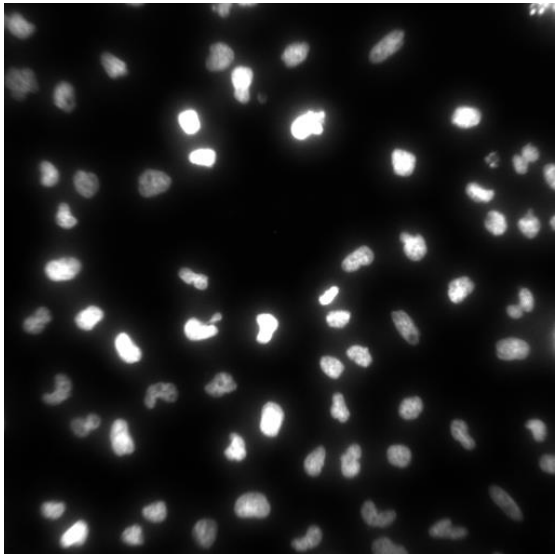
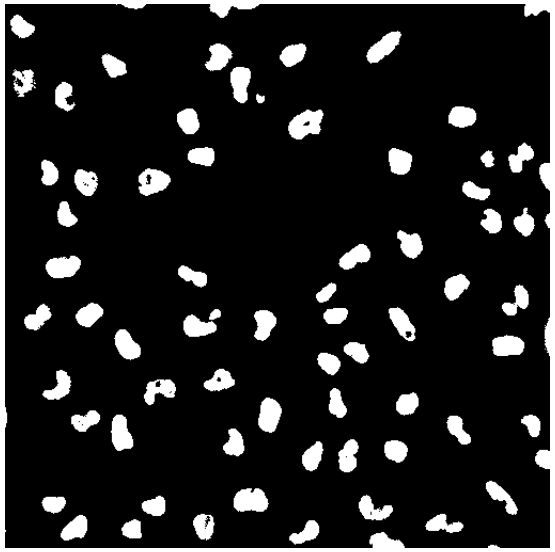
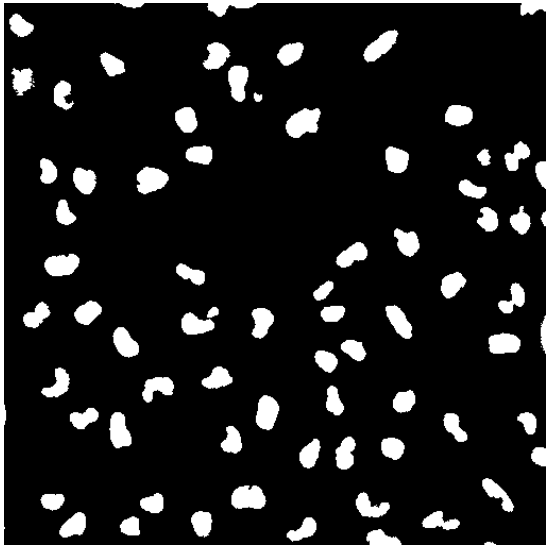






Package EBImage(Bioconductor)

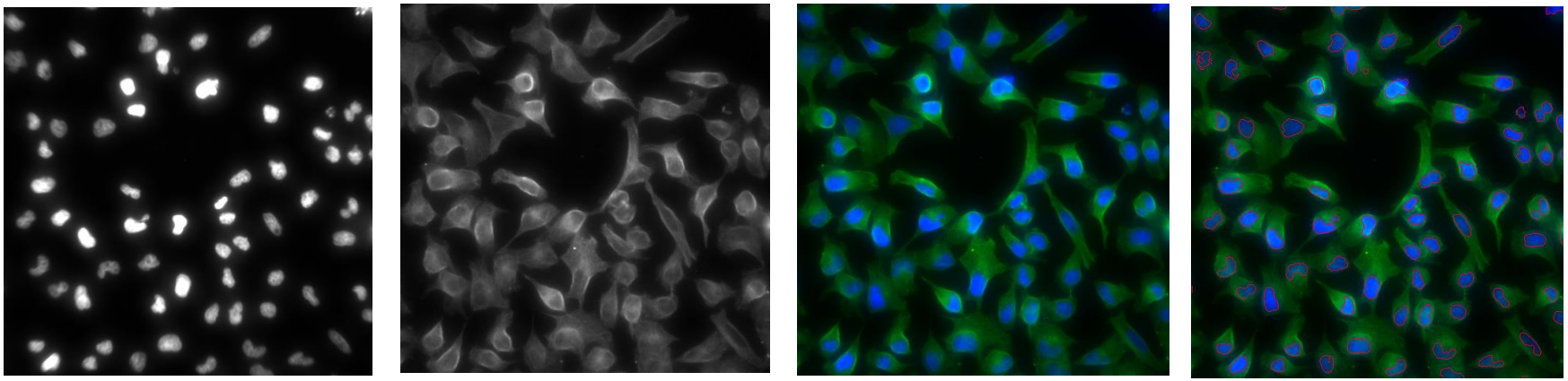
- Segmentation: thresh adjust and fill holes

```
nuc = readImage(system.file("images", "nuclei.tif", package="EBImage"))
display(nuc)
nmask = thresh(nuc, 40, 40, 0.05)
display(nmask)
nmask = fillHull(nmask)
```

Package EBImage(Bioconductor)

- Segmentation: highlight segmented objects



```
## using paintObjects to highlight objects
nuc = readImage(system.file("images", "nuclei.tif", package="EBImage"))
cel = readImage(system.file("images", "cells.tif", package="EBImage"))
img = rgbImage(green=cel, blue=nuc)
display(cel)
display(img)
res = paintObjects(nmask, img)
display(res)
```

Package EBImage(Bioconductor)

- Processing images as a group: How in R base (my way)

```
###### What R base can do for group processing
##1. identify files to read in
filesToProcess <- dir(pattern = "mristack.*\ \.png$")
filesToProcess
##2. Iterate over each of those file names with lapply
listOfFiles <- lapply(filesToProcess, function(x) readImage(x))
##3. apply manipulation to each file
listOfFiles <- lapply(listOfFiles, function(z) rotate(z, 10))#to rotate
lapply(listOfFiles, function(y) display(y))#to display
display(listOfFiles[[1]])# display one individual image of the group
##save the image files to folder as  png, jpeg, or tiff ect
#Step1, extract from the list
extract_list<-1:9
for (i in 1:9)
{
  extract_list[i] <- paste("imri10rot_",i, "<-listOfFiles[[",i, "]]",sep = "")

}
print.table <- function(m){write.table(format(m, justify="right"),
                        row.names=F, col.names=F, quote=F)
}
print.table(extract_list)
#step5 write to the folder
save_image<-1:9
for (i in 1:9)
{
  save_image[i] <- paste("writeImage(imri10rot_",i,",", "'imri10rot_",i,".png',  quality=100)",sep = "")

}
print.table(save_image)
```

Don't mean to let you read the codes on this slide

Package EBImage(Bioconductor)

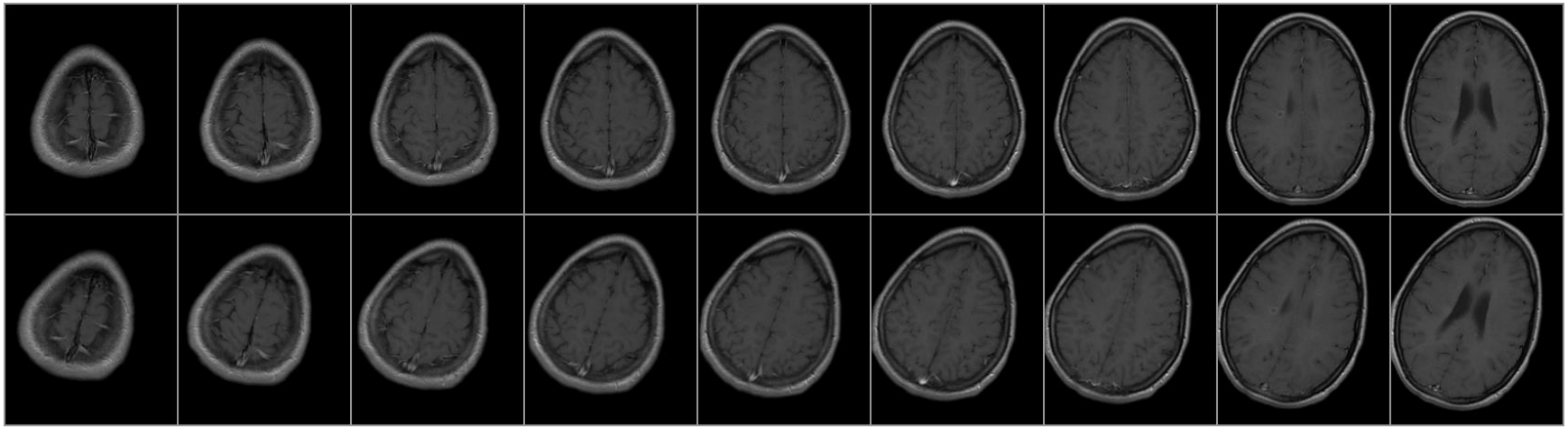
- Processing images as a group: How in EBImage

```
filesToProcess <- dir(pattern = "mristack.*\\.png$")
filesToProcess
a=readImage(filesToProcess)
a_20=rotate(a, 20)
writeImage(a_20, "a_20.png", quality=100)
```

Now you can read

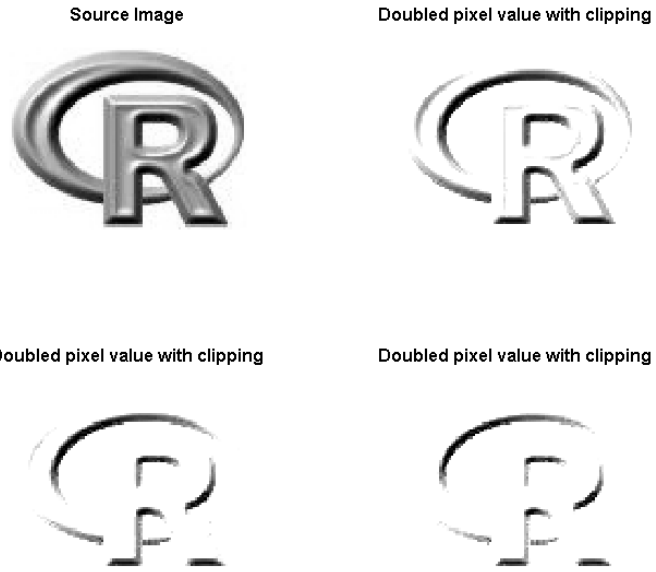
Package EBImage(Bioconductor)

- Tile and untile images



```
##tile and untile, Given a sequence of frames, tile generates a single image with frames tiled
mri=combine(a, a_20)
mri_tile = tile(mri, c(9,2))
display(mri_tile)
## untile
mri_untile=untile(mri_tile, c(9, 2))
display(mri_untile)
```

Package Ripa




```
install.packages("ripa")
library("ripa")

###adjust Brightness
data(logo)
par(mfrow=c(2,2))
plot(logo, main="Source Image")
# the next one is saturated as expected
plot(clipping(2*logo), main="Doubled pixel value with clipping")
plot(clipping(3*logo), main="Doubled pixel value with clipping")
plot(clipping(4*logo), main="Doubled pixel value with clipping")
```

Using R and imageJ in the same environment with Bio7

- The OpenSource application **Bio7** is a software for ecological simulation models, image analysis and statistical analysis.

- Bio7 contains a complete "Graphical User Interface" (GUI) for the statistical software R and the scientific image analysis tool ImageJ with special functions to send image data from ImageJ to R or R data to ImageJ

- Website: http://bio7.org

Questions and Answers