DSM3004
Spatial Statistics for Remotely Sensed Images

Mohammad Wasiq

Table of Contents

1 Practical - 1

1.1 Objective :

Using **Imager** package in R, working with image data :

i. Display an image in R window

ii. Compute the basic summary and by using the functions find spatial information regarding the image.

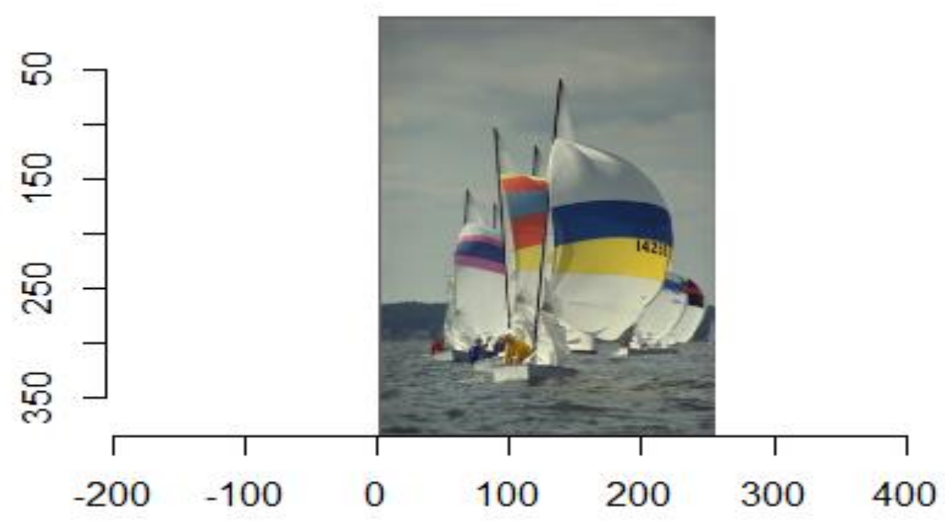
iii. Also make the arithmetic transformation on that image.

1.1.1 About `imager` package

imager contains a large array of functions for working with image data, with most of these functions coming from the CImg library by David Tschumperlé. This vignette is just a short tutorial, you'll find more information and examples on the website. Each function in the package is documented and comes with examples, so have a look at package documentation as well.

1.1.2 Getting Start with imager

```
# load the library
library(imager)

# plot the image boat
plot(boats)
```



```
class(boats)

## [1] "cimg"         "imager_array" "numeric"
```

The class of boat image is **cimg**.

```
boats

## Image. Width: 256 pix Height: 384 pix Depth: 1 Colour channels: 3
```

This image contain Width of 256 pix and Height of 384 pix with depth of 1 and this image boats has three channels named **GRB** (Green, Red, Blue).

```
grayscale(boats)

## Image. Width: 256 pix Height: 384 pix Depth: 1 Colour channels: 1
```

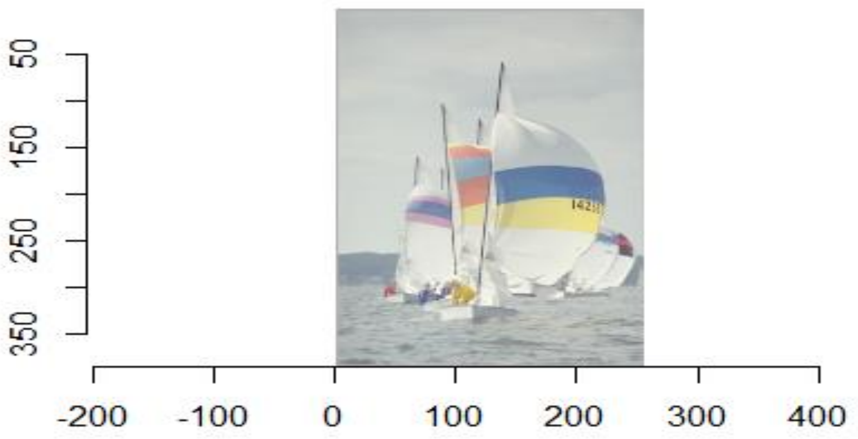
A grayscale image has only 1 channel.

```
dim(boats)

## [1] 256 384   1   3
```

`boats` image has four dimensions.

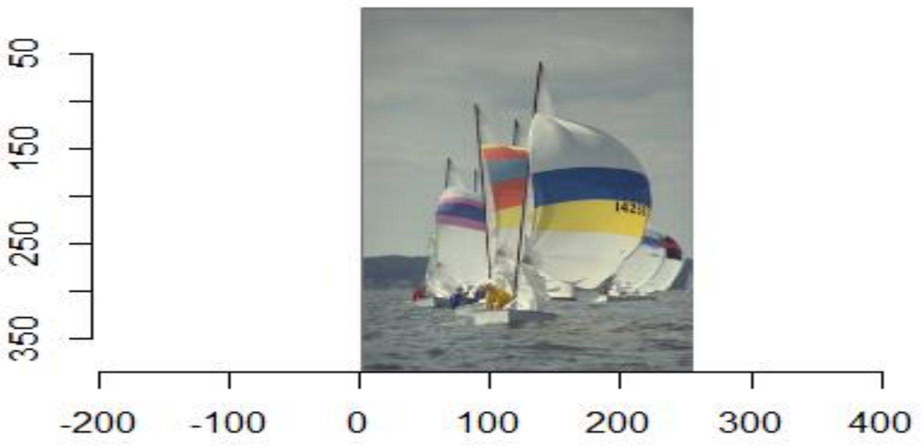
1.1.3 Transformation of Image

```
z1<- log(boats)+3*sqrt(boats)
plot(z1)
```



After image logorithm transformation, the image is more clear.

```
z2<- exp(boats)+3*sqrt(boats)
plot(z2)
```



```
mean(boats)

## [1] 0.5089061
```

The mean of `boats` image is 0.5089.

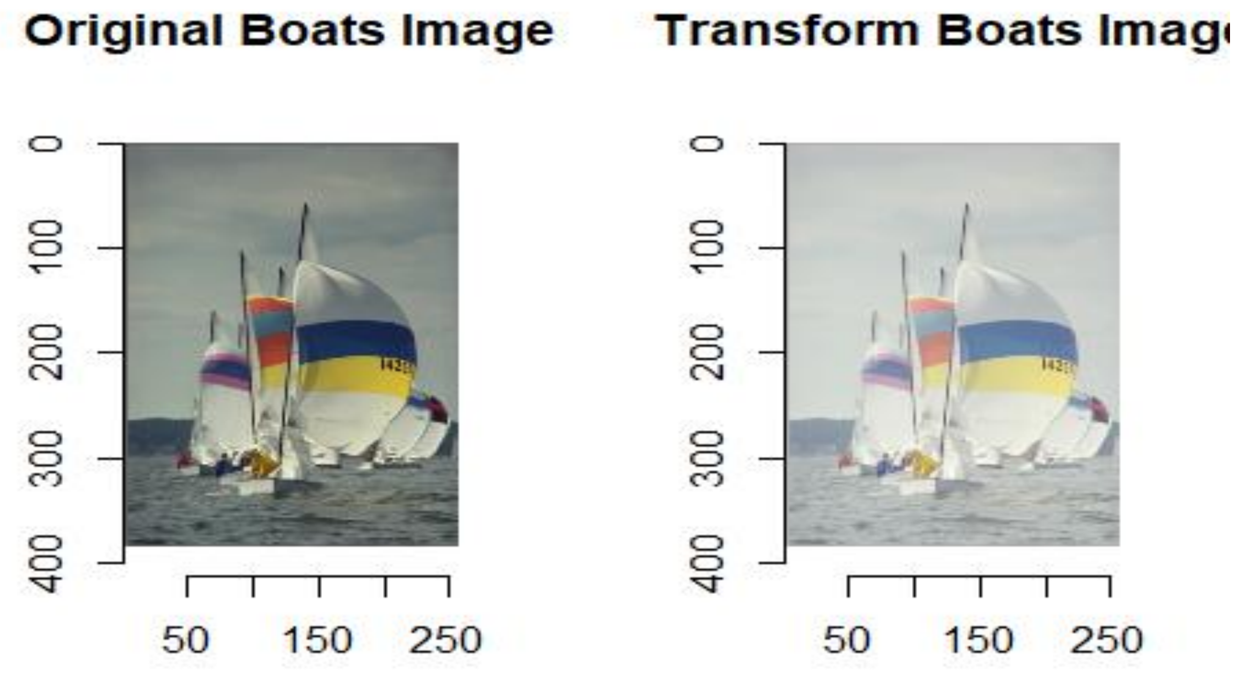
```
sd(boats)

## [1] 0.144797
```

The standard deviation of `boats` image is 0.1445

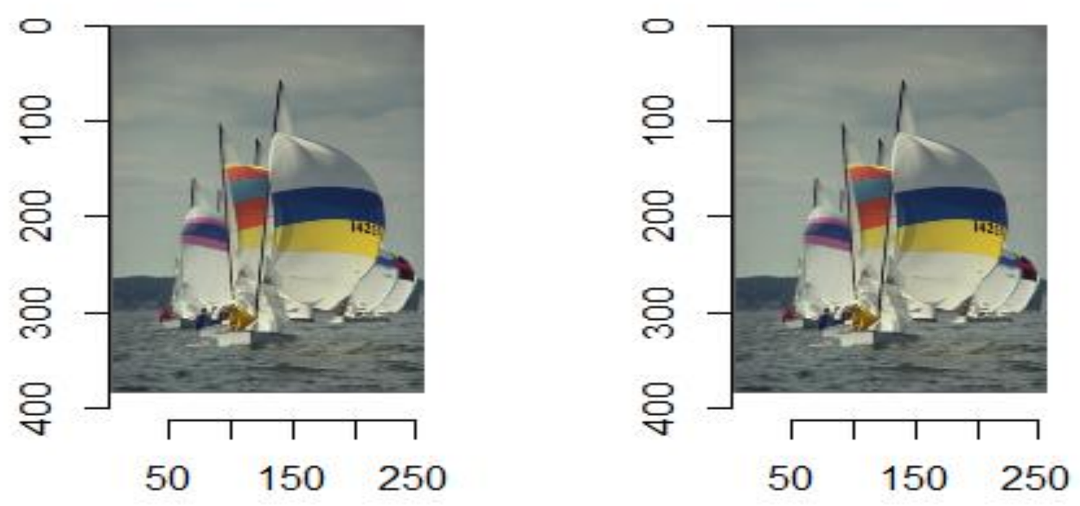
```
summary(boats)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01608 0.43386 0.52646 0.50891 0.59151 1.00000
```

- The minimum observation of `boats` image is 0.01608 and maximum observation is 1.0.
- 1^{st} quartile of this image is 0.4338 and the 3^{rd} quartile of this image is 0.5915.
- Median of the image is 0.6264 and mean is 0.508.

```
layout(t(1:2))

plot(boats, main= "Original Boats Image")
plot(z1, main= "Transform Boats Image")
```



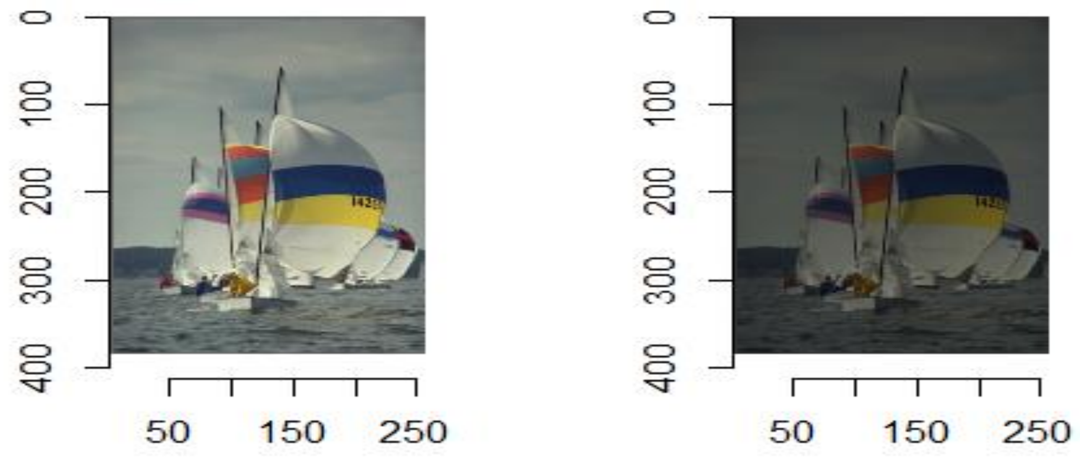
```
layout(t(1:2))

plot(boats)
plot(boats/2)
```



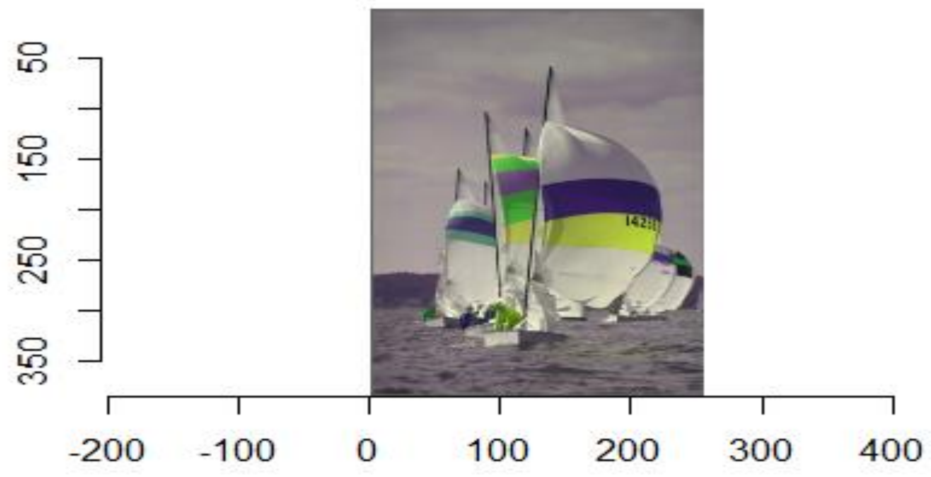
- There's no agreed-upon standard for how RGB values should be scaled. Some software, like CImg, uses a range of $0 - 255$ (dark to light), other, like R's rgb function, uses a 0-1 range.
- Often it's just more convenient to work with a zero-mean image, which means having negative values.

If we don't want imager to rescale the colours automatically, set rescale to FALSE, but now imager will want values that are in the [0,1][0,1] range.

```
layout(t(1:2))

plot(boats, rescale=FALSE)
plot(boats/2, rescale=FALSE)
```

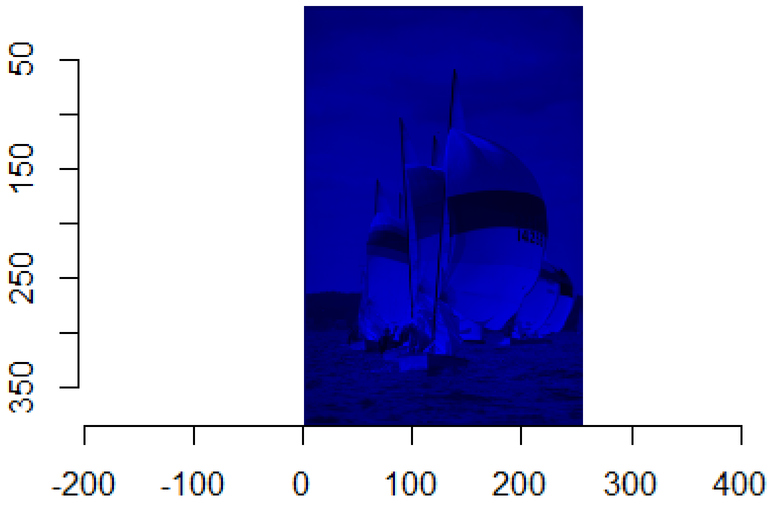


```
cscale<- function(r, g, b) rgb(g, r, b)
plot(boats, colourscale = cscale, rescale = F)
```



In grayscale images pixels have only one value, so that the colour map is simpler: it takes a single value and returns a colour. In the next example we convert the image to grayscale.

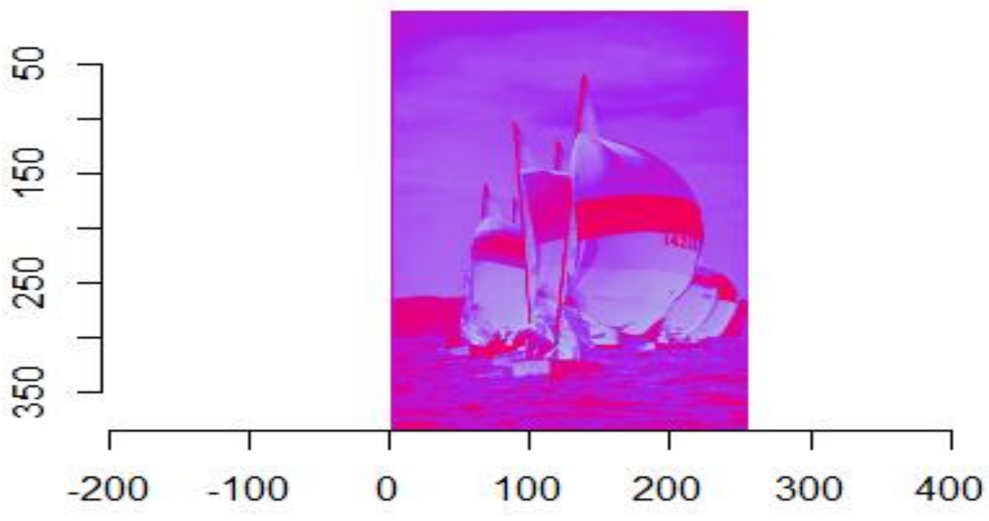
```
library(dplyr)

cscale<- function(v) rgb(0,0,v)
grayscale(boats) %>%
  plot(colourscale= cscale, rescale=F)
```



Now this image is covered by *blue* color.

```
cscale<- scales::gradient_n_pal(c("red","purple","lightblue"), c(0, 0.5, 1))

# cscale is now a function returning a colour codes
cscale(0)

## [1] "#FF0000"
```

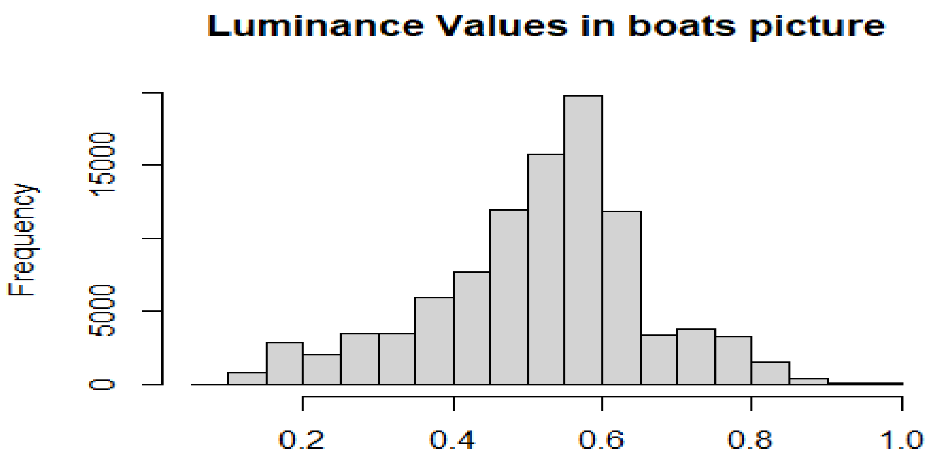
```r
grayscale(boats) %>%
  plot(colourscale= cscale, rescale=F)
```



Now this image contains three colors `red`,`purple`,`lightblue`.

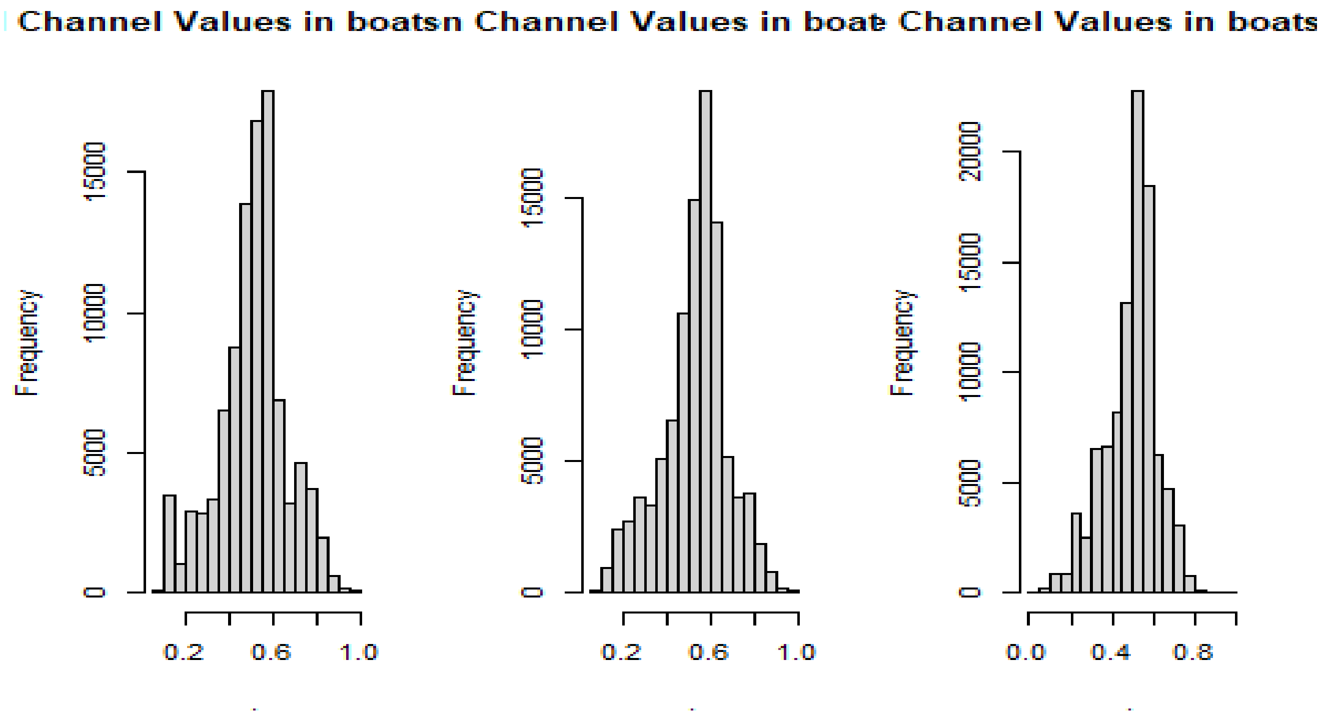
1.1.4 Histogram Equalisation

```r
grayscale(boats) %>%
  hist(main= "Luminance Values in boats picture")
```



From the above *histogram* we can easily see that the image has `left skewed` distribution.

```
layout(t(1:3))

R(boats) %>%
  hist(main= "Red Channel Values in boats picture")

G(boats) %>%
  hist(main= "Green Channel Values in boats picture")

B(boats) %>%
  hist(main= "Blue Channel Values in boats picture")
```



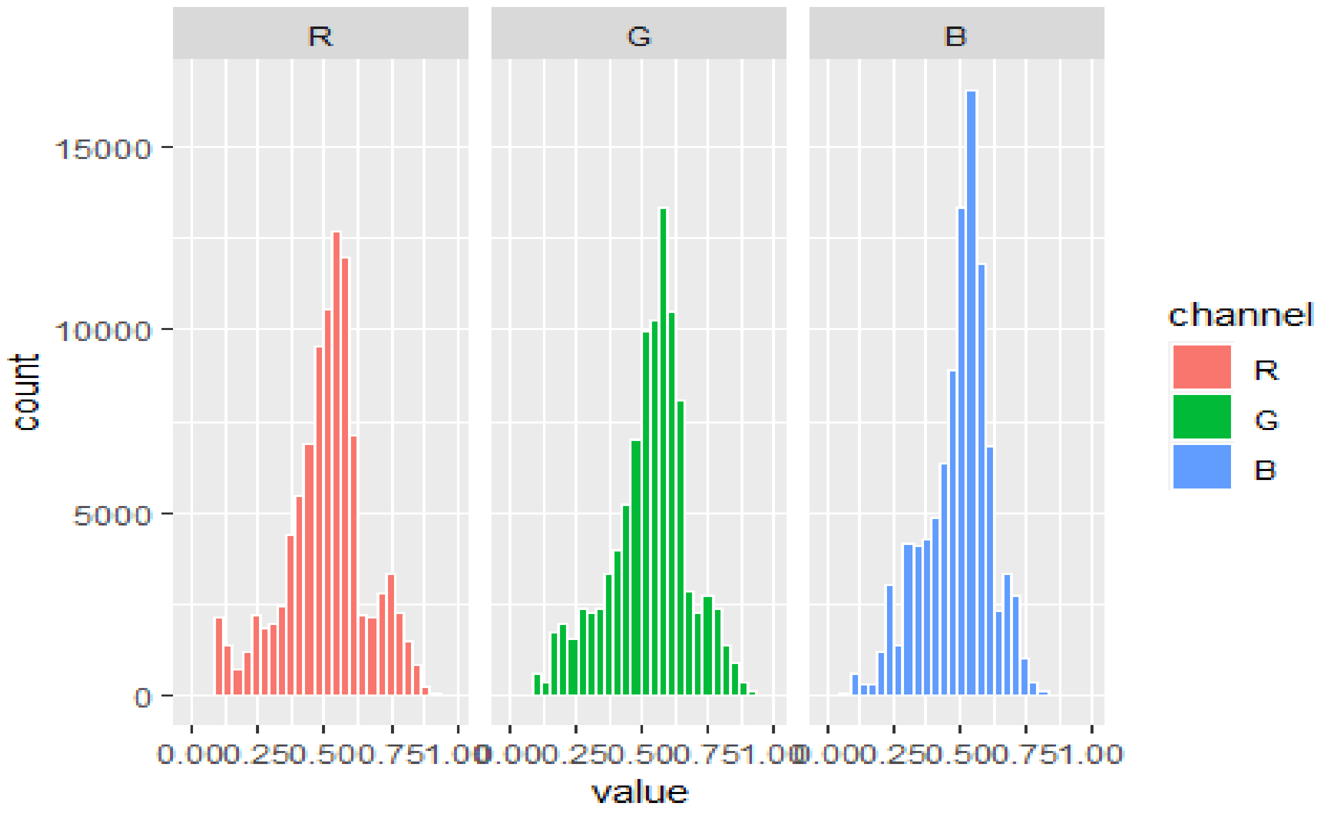
Convert the boats image into data frame

```
df<- boats %>%
  as.data.frame()

df %>% head()

##   x y cc     value
## 1 1 1  1 0.3882353
## 2 2 1  1 0.3858633
## 3 3 1  1 0.3849406
## 4 4 1  1 0.3852481
## 5 5 1  1 0.3860388
## 6 6 1  1 0.3855557
```

```
library(ggplot2)
df<- df %>%
  mutate(channel= factor(cc, labels = c("R", "G", "B")))
df %>% head()
```

```
##   x y cc     value channel
## 1 1 1  1 0.3882353       R
## 2 2 1  1 0.3858633       R
## 3 3 1  1 0.3849406       R
## 4 4 1  1 0.3852481       R
## 5 5 1  1 0.3860388       R
## 6 6 1  1 0.3855557       R
```

```
df %>%
  ggplot(aes(value, fill= channel))+
  geom_histogram(bins= 30, color= "white")+
  facet_wrap(~channel)
```



2 Practical - 2

2.1 Objective :

(i) Generate a set of 10 points without attributes on the unit square [0, 1] × [0, 1] from uniform distribution and convert them into spatial points, make a scattered plot of them in R .

(ii) Also, create a Spatial Points Data frame object by building it from a Spatial Points object and a data.frame containing the attributes, and plot these spatial points in coordinate system.

2.1.1 *Spatial Points*

The **sp** package provides classes and methods for dealing with spatial data in R. The spatial data structures implemented include points, lines, polygons and grids; each of them with or without attribute data. Although we mainly aim at using spatial data in the geographical (two-dimensional) domain, the data structures that have a straightforward implementation in higher dimensions (points, grids) do allow this.

```
# Load the require Library
library(sp)
```

We can generate a set of 10 points on the unit square [0,1] × [0,1].

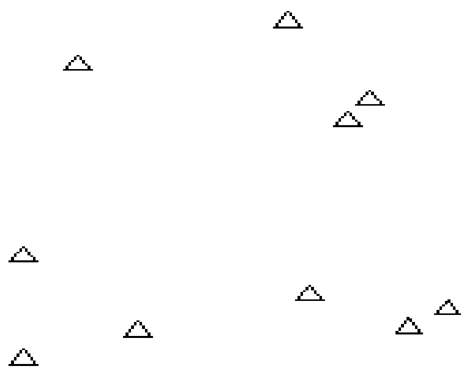
```
xc<- round(runif(10), 2)
yc<- round(runif(10), 2)
xy<- cbind(xc, yc)
xy

##         xc   yc
##  [1,] 0.22 0.02
##  [2,] 0.70 0.97
##  [3,] 0.43 0.10
##  [4,] 0.99 0.16
##  [5,] 0.74 0.20
##  [6,] 0.92 0.11
##  [7,] 0.85 0.75
##  [8,] 0.22 0.31
##  [9,] 0.32 0.85
## [10,] 0.81 0.69
```

Convert the above data into spatial points using `SpatialPoints()` function

```
xy.sp<- SpatialPoints(xy)
xy.sp

## SpatialPoints:
##         xc   yc
##  [1,] 0.22 0.02
##  [2,] 0.70 0.97
##  [3,] 0.43 0.10
##  [4,] 0.99 0.16
##  [5,] 0.74 0.20
##  [6,] 0.92 0.11
##  [7,] 0.85 0.75
##  [8,] 0.22 0.31
##  [9,] 0.32 0.85
## [10,] 0.81 0.69
## Coordinate Reference System (CRS) arguments: NA
```

Plot the spatial points

```
plot(xy.sp, pch= 2)
```



We can rewrite the co-ordinates from xy.sp

```
xy.cc<- coordinates(xy.sp)
xy.cc
```

```
##         xc   yc
##  [1,] 0.22 0.02
##  [2,] 0.70 0.97
##  [3,] 0.43 0.10
##  [4,] 0.99 0.16
##  [5,] 0.74 0.20
##  [6,] 0.92 0.11
##  [7,] 0.85 0.75
##  [8,] 0.22 0.31
##  [9,] 0.32 0.85
## [10,] 0.81 0.69
```

```
class(xy.cc)

## [1] "matrix" "array"

dim(xy.cc)

## [1] 10  2
```

and other methods retrieve the bounding box, the dimensions, select points (not dimensions or columns), coerce to a `data.frame`, or print a `summary` :

```
bbox(xy.cc)

##    min  max
## x 0.22 0.99
## y 0.02 0.97

dimensions(xy.sp)

## [1] 2

xy.sp[1:2]

## SpatialPoints:
##        xc   yc
## [1,] 0.22 0.02
## [2,] 0.70 0.97
## Coordinate Reference System (CRS) arguments: NA

xy.df<- as.data.frame(xy.sp)
xy.df

##      xc   yc
## 1  0.22 0.02
## 2  0.70 0.97
## 3  0.43 0.10
## 4  0.99 0.16
## 5  0.74 0.20
## 6  0.92 0.11
## 7  0.85 0.75
## 8  0.22 0.31
## 9  0.32 0.85
## 10 0.81 0.69

class(xy.df)

## [1] "data.frame"

dim(xy.df)

## [1] 10  2
```

```
summary(xy.df)

##        xc               yc        
##  Min.   :0.2200   Min.   :0.0200  
##  1st Qu.:0.3475   1st Qu.:0.1225  
##  Median :0.7200   Median :0.2550  
##  Mean   :0.6200   Mean   :0.4160  
##  3rd Qu.:0.8400   3rd Qu.:0.7350  
##  Max.   :0.9900   Max.   :0.9700
```

2.1.2 Points with attributes

One way of creating a Spatial Points data frame object is by building it from a Spatial Points object and a data.frame containing the attributes :

```
df<- data.frame(z1= round(5+ rnorm(10), 2),
                z2= 20:29)
df

##      z1 z2
## 1  4.72 20
## 2  5.76 21
## 3  5.18 22
## 4  3.52 23
## 5  5.25 24
## 6  3.10 25
## 7  5.62 26
## 8  3.95 27
## 9  6.03 28
## 10 4.88 29
```

Convert the above data frame `df` into spatial points

```
xy.spdf<- SpatialPointsDataFrame(xy.sp, df)
xy.spdf

##     coordinates   z1 z2
## 1  (0.22, 0.02) 4.72 20
## 2   (0.7, 0.97) 5.76 21
## 3   (0.43, 0.1) 5.18 22
## 4  (0.99, 0.16) 3.52 23
## 5   (0.74, 0.2) 5.25 24
## 6  (0.92, 0.11) 3.10 25
## 7  (0.85, 0.75) 5.62 26
## 8  (0.22, 0.31) 3.95 27
## 9  (0.32, 0.85) 6.03 28
## 10 (0.81, 0.69) 4.88 29
```

```
summary(xy.spdf)

## Object of class SpatialPointsDataFrame
## Coordinates:
##     min  max
## xc 0.22 0.99
## yc 0.02 0.97
## Is projected: NA
## proj4string : [NA]
## Number of points: 10
## Data attributes:
##        z1              z2
##  Min.   :3.100   Min.   :20.00
##  1st Qu.:4.143   1st Qu.:22.25
##  Median :5.030   Median :24.50
##  Mean   :4.801   Mean   :24.50
##  3rd Qu.:5.527   3rd Qu.:26.75
##  Max.   :6.030   Max.   :29.00

dimensions(xy.spdf)

## [1] 2

xy.spdf[1:2, ] # selects row 1 and 2

##    coordinates   z1 z2
## 1 (0.22, 0.02) 4.72 20
## 2  (0.7, 0.97) 5.76 21

xy.spdf[1] # selects attribute column 1, along with the coordinates

##     coordinates   z1
## 1  (0.22, 0.02) 4.72
## 2   (0.7, 0.97) 5.76
## 3   (0.43, 0.1) 5.18
## 4  (0.99, 0.16) 3.52
## 5   (0.74, 0.2) 5.25
## 6  (0.92, 0.11) 3.10
## 7  (0.85, 0.75) 5.62
## 8  (0.22, 0.31) 3.95
## 9  (0.32, 0.85) 6.03
## 10 (0.81, 0.69) 4.88

xy.spdf[1:2, "z2"] # select row 1,2 and attribute "z2"

##    coordinates z2
## 1 (0.22, 0.02) 20
## 2  (0.7, 0.97) 21

xy.df = as.data.frame(xy.spdf)
xy.df[1:2,]

##     z1 z2   xc   yc
## 1 4.72 20 0.22 0.02
## 2 5.76 21 0.70 0.97

xy.cc<- coordinates(xy.spdf)

class(xy.cc)

## [1] "matrix" "array"

dim(xy.cc)

## [1] 10  2

df1<- data.frame(xy, df)
coordinates(df1)<- c("xc", "yc")
df1

##     coordinates   z1 z2
## 1  (0.22, 0.02) 4.72 20
## 2   (0.7, 0.97) 5.76 21
## 3   (0.43, 0.1) 5.18 22
## 4  (0.99, 0.16) 3.52 23
## 5   (0.74, 0.2) 5.25 24
## 6  (0.92, 0.11) 3.10 25
## 7  (0.85, 0.75) 5.62 26
## 8  (0.22, 0.31) 3.95 27
## 9  (0.32, 0.85) 6.03 28
## 10 (0.81, 0.69) 4.88 29

df2<- data.frame(xy, df)
coordinates(df2)<- ~xc+yc


df2[1:2,]

##    coordinates   z1 z2
## 1 (0.22, 0.02) 4.72 20
## 2  (0.7, 0.97) 5.76 21

as.data.frame(df2)[1:2,]

##     xc   yc   z1 z2
## 1 0.22 0.02 4.72 20
## 2 0.70 0.97 5.76 21
```

Note that in this form, coordinates by setting (specifying) the coordinates promotes its argument, an object of class data.frame to an object of class Spatial Points Data Frame.

The method as data.frame coerces back to the original data.frame. When used on a right-hand side of an equation, coordinates retrieves the matrix with coordinates :

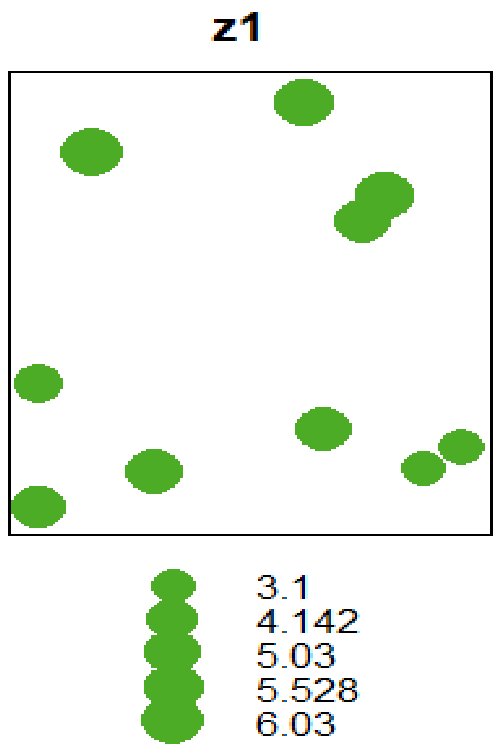
```
coordinates(df2)[1:2]

## [1] 0.22 0.70
```

Elements (columns) in the data.frame part of an object can be retrieved, assigned directly :

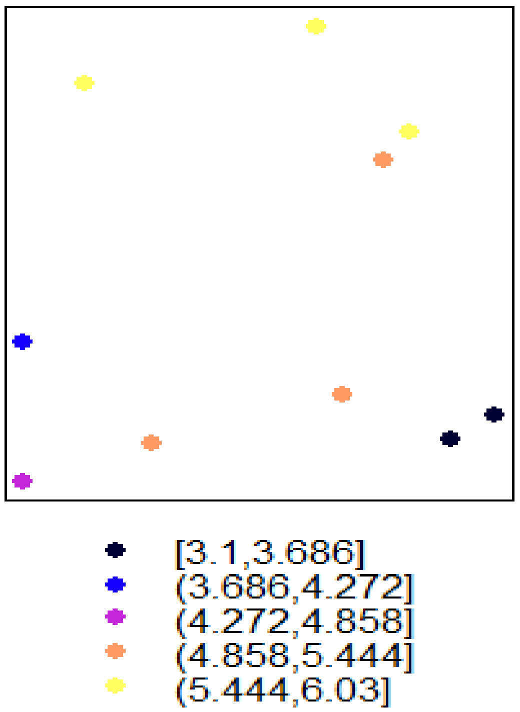
```
df2[["z2"]]

##  [1] 20 21 22 23 24 25 26 27 28 29

df2[["z2"]][10]<- 20

df2[["z3"]]<- 1:10

summary(df2)

## Object of class SpatialPointsDataFrame
## Coordinates:
##     min  max
## xc 0.22 0.99
## yc 0.02 0.97
## Is projected: NA
## proj4string : [NA]
## Number of points: 10
## Data attributes:
##        z1               z2               z3
##  Min.   :3.100   Min.   :20.00   Min.   : 1.00
##  1st Qu.:4.143   1st Qu.:21.25   1st Qu.: 3.25
##  Median :5.030   Median :23.50   Median : 5.50
##  Mean   :4.801   Mean   :23.60   Mean   : 5.50
##  3rd Qu.:5.527   3rd Qu.:25.75   3rd Qu.: 7.75
##  Max.   :6.030   Max.   :28.00   Max.   :10.00
```

Plotting attribute data can be done by using either spplot to colour symbols, or bubble which uses symbol size :

```
bubble(df2, "z1", key.space= "bottom")
```



```
spplot(df2, "z1", key.space= "bottom")
```


[3.1,3.686]
(3.686,4.272]
(4.272,4.858]
(4.858,5.444]
(5.444,6.03]