Report II - Analysis of SAR Samples

Ma siteng(simon), Liang shuang, Liu zhening(NiuNiu), Zhang yu

2020/10/21

Implement the densities of the K and G0 distributions for intensity data

Import the bright image data from the “ImagematrXl” library for analysis.

```
# Use your paths
source("D:/Program Files/R/RStudio/documents/Codes/imagematrix.R")
load("D:/Program Files/R/RStudio/documents/Data/R/bright.Rdata")
# Inspect what they are in the Environment window
```

We usually start reporting the very basic information about the data, including its type, dimension, range, etc.

```
typeof(bright)
```

```
## [1] "double"
```

```
dim(bright)
```

```
## [1] 109 214   3
```

```
range(bright)
```

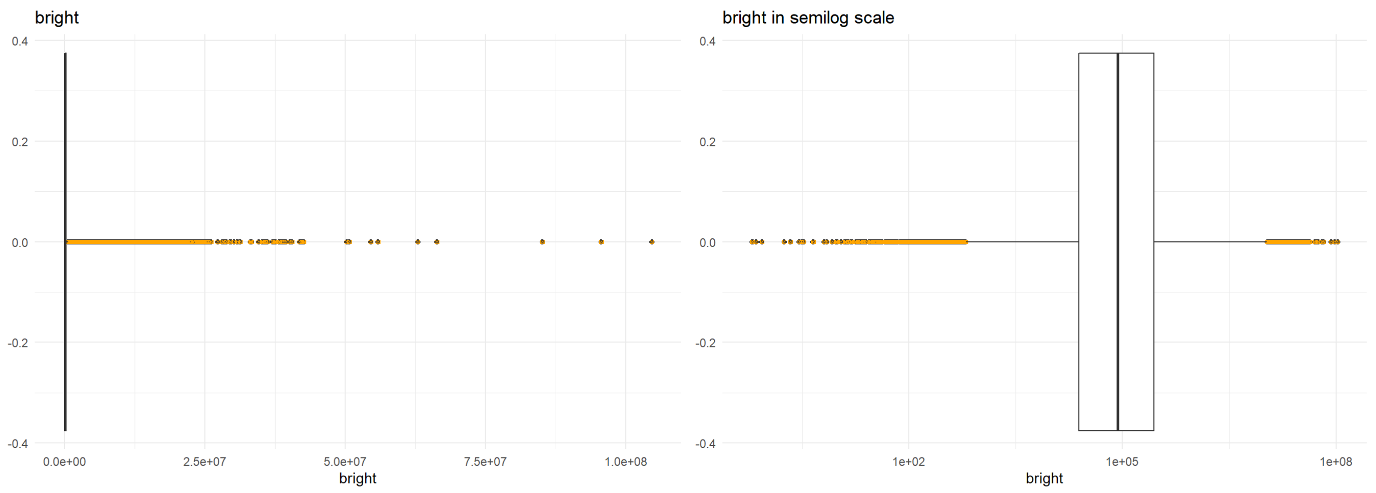
```
## [1] 6.265993e-01 1.046492e+08
```

```
vector.bright <- data.frame(bright=as.vector(bright))
summary(vector.bright)
```

```
##      bright
##  Min.   :        1
##  1st Qu.:    24325
##  Median :    85451
##  Mean   :   469894
##  3rd Qu.:   277819
##  Max.   :104649217
```

Let’s move to graphical representations of the data. We will analyze the data with boxplot and histogram The outliers are shown in orange.

```
ggplot(vector.bright, aes(x=bright)) +
  geom_boxplot(notch = TRUE)+
  geom_boxplot(outlier.colour="orange", outlier.shape=7,outlier.size=1)+
  ggtitle("bright")

ggplot(vector.bright, aes(x=bright)) +
  geom_boxplot(notch = TRUE) +
  geom_boxplot(outlier.colour="orange", outlier.shape=7,outlier.size=1)+
  scale_x_log10()+
  ggtitle("bright in semilog scale")
```

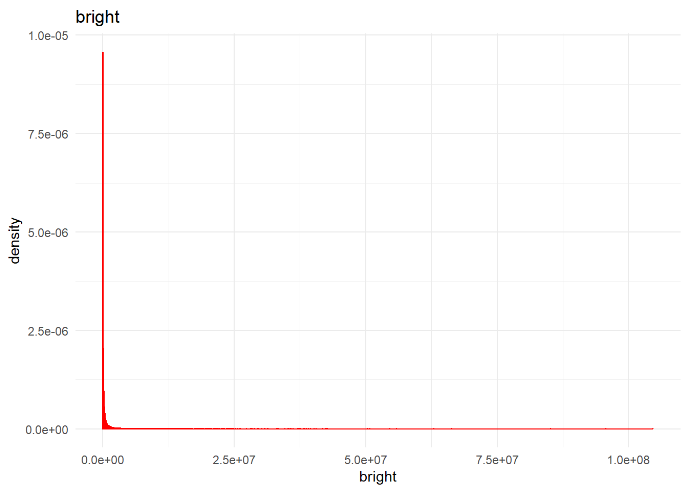


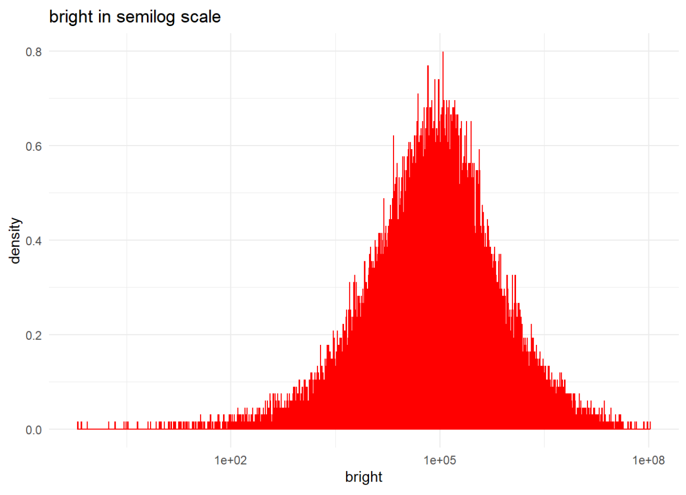
Boxplots with notches

First, we get a boxplot of the bright image data, i.e., the graph on the left named "bright". We can see from the "bright" graph that the boxes are so flattened. Therefore, we perform a logarithmic transformation of the data to get the boxplot titled "Bright at half logarithmic scale" in the figure below. As we can see from the "bright at half-logarithmic scale" graph, the mean value of the data is around 10^5 and does not fluctuate much.

Histograms of the data complement the information shown by the boxplots. We will use the Freedman-Diaconis rule for building the histograms. It consists of using bins of equal size: $\frac{3}{2}\mathrm{IQR}(z)/n^{1/3}$.

```
ggplot(vector.bright, aes(x=bright)) +
  geom_histogram(aes(y=..density..),
               bins=nclass.FD(unlist(vector.bright)),
               col="red", fill="white")+
  ggtitle("bright")

ggplot(vector.bright, aes(x=bright)) +
  geom_histogram(aes(y=..density..),
               bins=nclass.FD(unlist(vector.bright)),
               col="red", fill="white") +
  scale_x_log10()+
  ggtitle("bright in semilog scale")
```

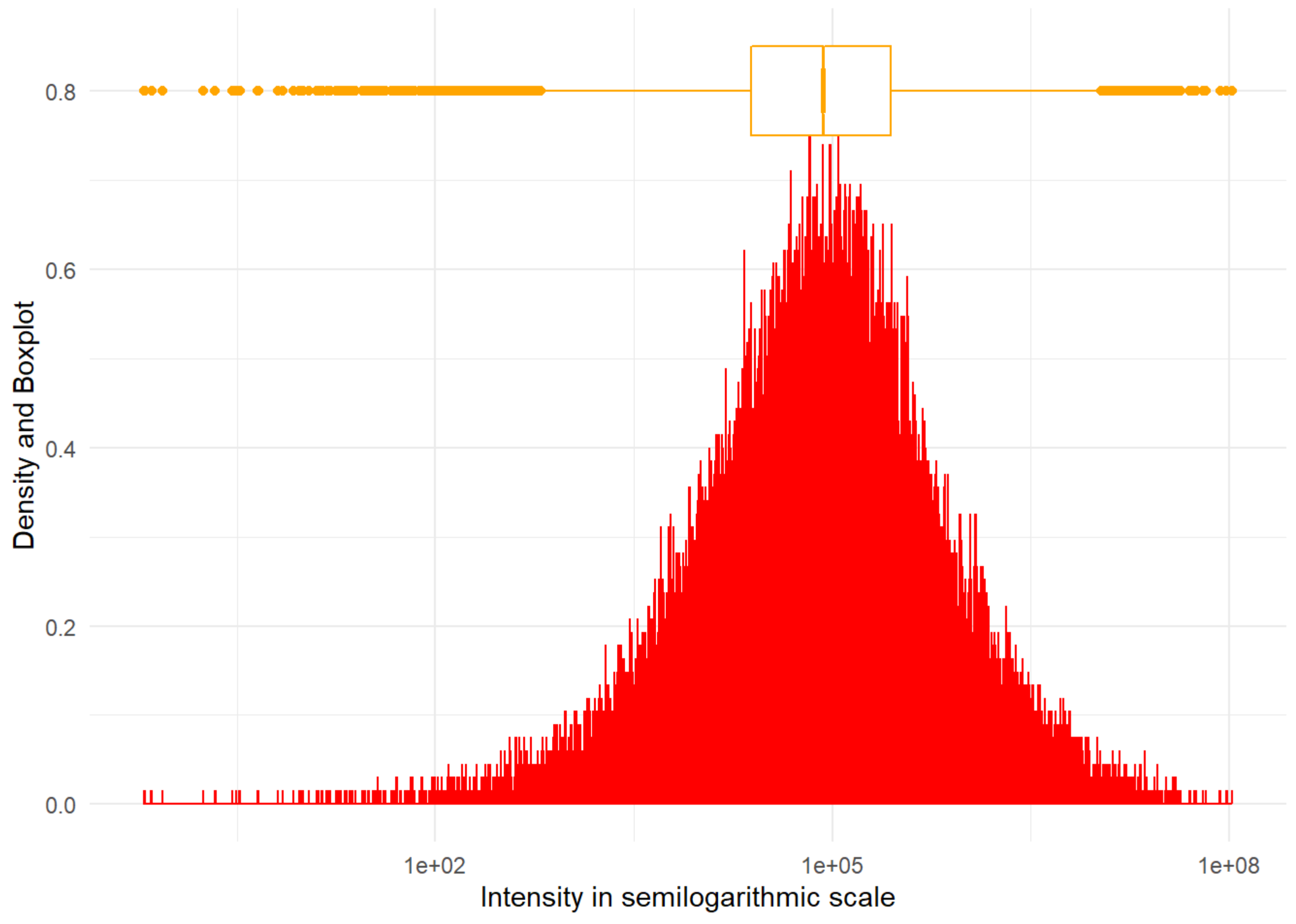




Histograms

Now we mix the second boxplot with the second histogram.

```
ggplot(vector.bright, aes(x=bright)) +
  geom_histogram(aes(y=..density..),
                 bins=nclass.FD(unlist(vector.bright)),
                 col="red", fill="white") +
  geom_boxplot(aes(y=0.8), width=.1,col="orange", notch=TRUE) +
  ylab("Density and Boxplot") +
  xlab("Intensity in semilogarithmic scale") +
  scale_x_log10()
```



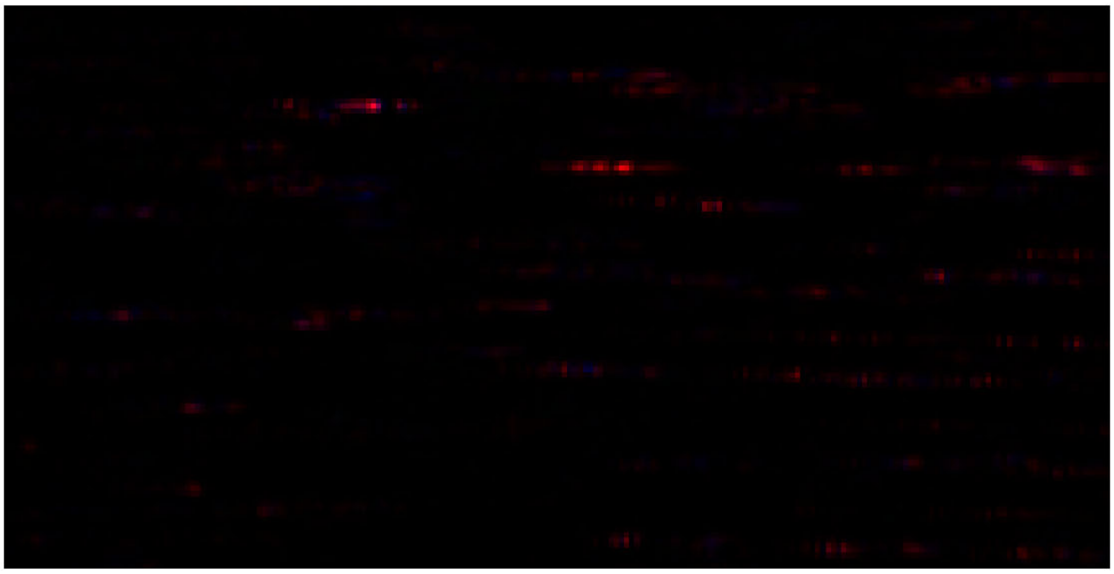
```
# Try reducing the number of bins
```

Through the above analysis, we can obtain the following three conclusions:

- The data are positive
- The data have a very large dynamic range
- The data are some symmetric

Now it's time to look at the "natural" domain of the data: as an image. The "imagematrix" library requires the data to be in $[0, 1]$. It provides a function to do this mapping.

```
plot(imagematrix(normalize(bright)))
```



We can barely see anything. There are a few very large values that "flatten" most of the observations into very deep greyscales. We want to "use" all possible values equally. In other words, we want to have a uniform histogram.

Consider the cumulative distribution function F_Z of the continuous random variable $Z colon \Omega \to \mathbb{R}$. The random variable $W = F_Z(Z)$ has a uniform distribution.

Prove that the cumulative distribution function of W is $F_W(w) = /Pr(W/leqw) = /Pr(F_Z(Z)/leqw)$. Since Z is continuous, there exists the reciprocal of its cumulative distribution function, F_Z^{-1}. Then we can write it as $F_W(w) = Pr(F_Z(Z)/leqw) = Pr(F_Z^{-1}(F_Z(Z)) \leq F_Z^{-1}(w)) = Pr(Z \leq F_Z^{-1}(w))$. This is exactly the cumulative distribution function of Z at $F_Z^{-1}(w)$, so $F_W(w) = F_Z(F_Z^{-1}(w)) = w$, which characterizes the uniform random variables on $(0, 1)$.

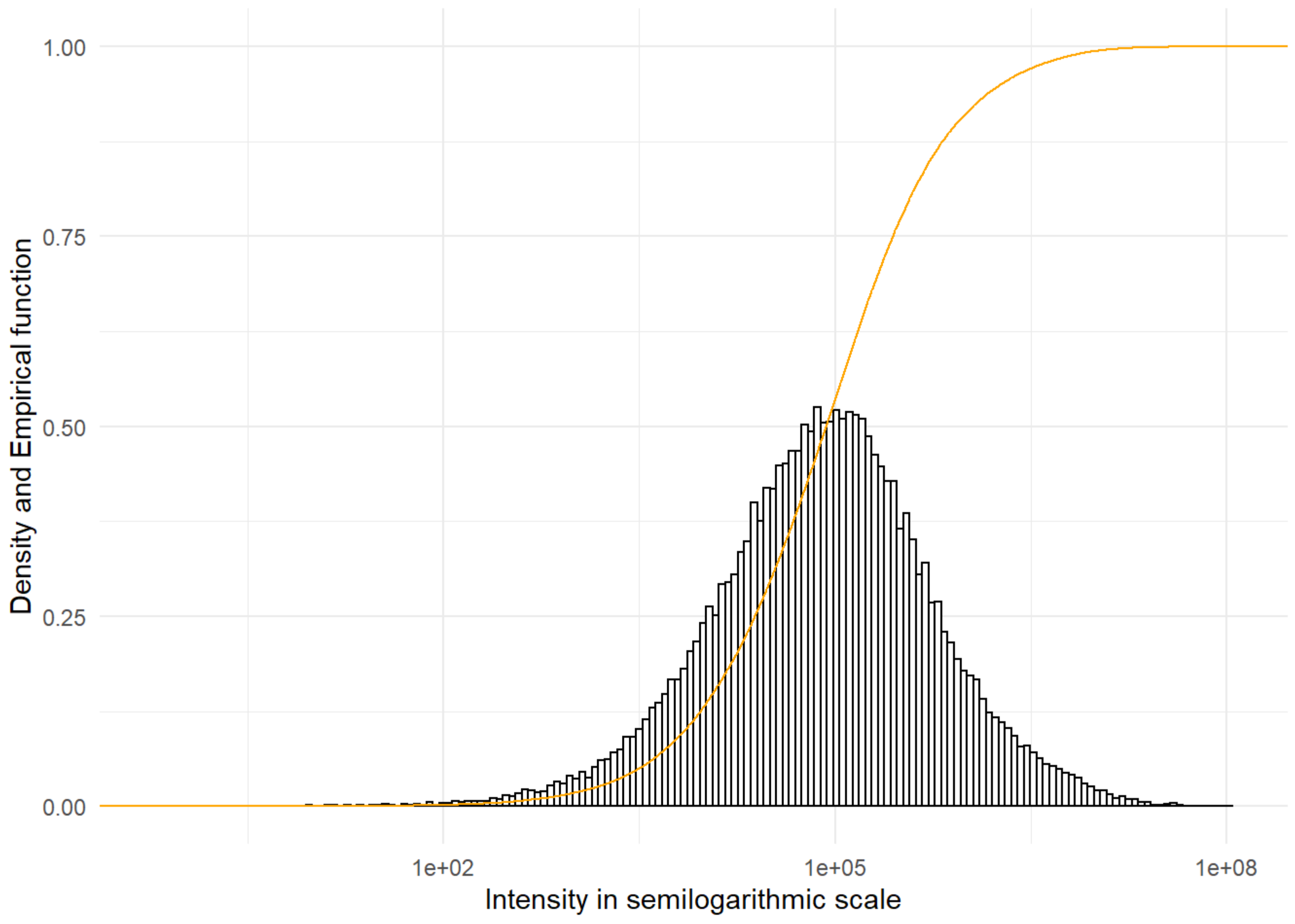
So all we need to do is apply the cumulative distribution function of the random variable generating the data to the data itself, and we'll get a uniformly distributed sample. But do we have this function? Usually we don't have it, but we can estimate it.

One of the simplest ways to estimate the cumulative distribution function from the data $z = (z_1, \ldots, z_n)$ is to use the empirical cumulative distribution function, or simply the "empirical function". It is only a finite approximation of the definition of the cumulative distribution function, given by the following formula

\begin{equation} wide-hat {F}(t) = frac1n # # # j # text{ such that} z_j/leq t/} , of the Dusk {formula} where $\boxed{\#}$ denotes the number of elements in the set.

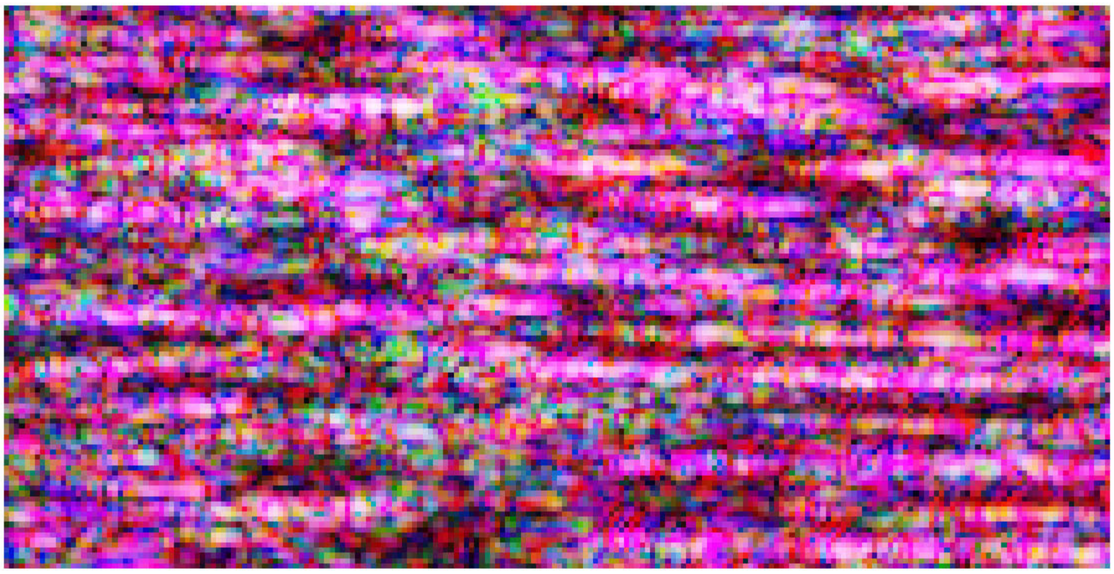
Let us see the original data and its empirical function.

```
ggplot(vector.bright, aes(x=bright)) +
  geom_histogram(aes(y=..density..),
                 bins=nclass.FD(unlist(vector.bright))/50,
                 col="black", fill="white") +
  stat_ecdf(col="orange") +
  ylab("Density and Empirical function") +
  xlab("Intensity in semilogarithmic scale") +
  scale_x_log10()
```



Let us implement this idea.

```
# First, we compute the empirical function
ecdf.bright <- ecdf(unlist(vector.bright))

# Then, we apply this function to the data
eq.bright <- ecdf.bright(unlist(vector.bright))

# Finally, we restore the matrix organization of the data
dim(eq.bright) <- dim(bright)

# And we see the result
plot(imagematrix(eq.bright))
```

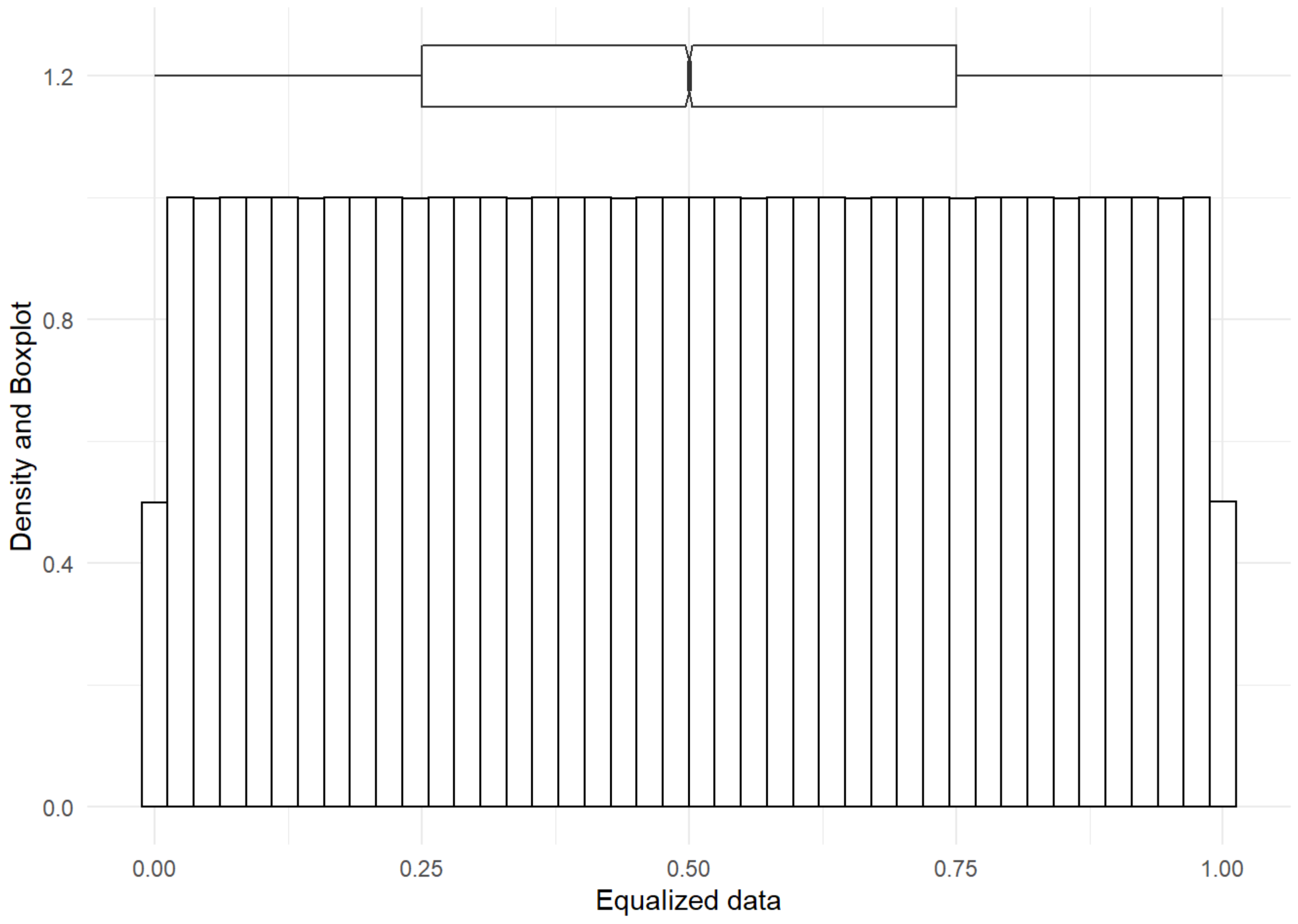


Let us now perform an EDA on the equalized data.

```
summary(as.vector(eq.bright))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 1.43e-05 2.50e-01 5.00e-01 5.00e-01 7.50e-01 1.00e+00
```

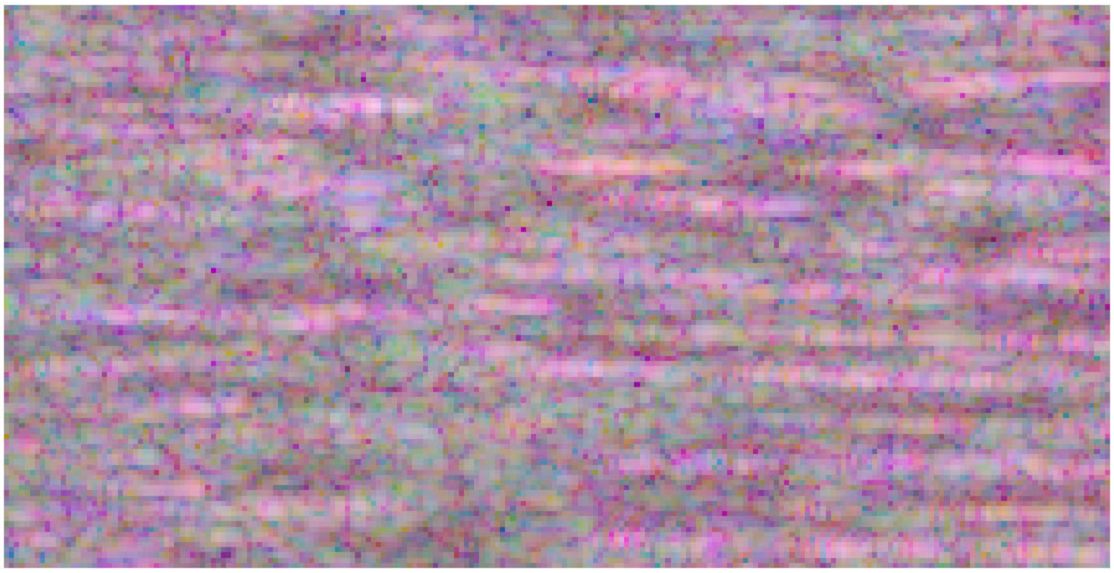
```
vector.eq.bright <- data.frame(eq.bright=as.vector(eq.bright))

ggplot(vector.eq.bright, aes(x=eq.bright)) +
  geom_histogram(aes(y=..density..),
                 bins=nclass.FD(unlist(vector.eq.bright)),
                 col="black", fill="white") +
  geom_boxplot(aes(y=1.2), width=.1, notch=TRUE) +
  ylab("Density and Boxplot") +
  xlab("Equalized data")
```



In the figure above, we can see that after the transformation, the new image has a uniform pixel distribution.

Of course, for the sake of simplicity, we can also directly perform a logarithmic transformation of the data, and the result is shown in the figure below.

```
plot(imagematrix(normalize(log(bright))))
```



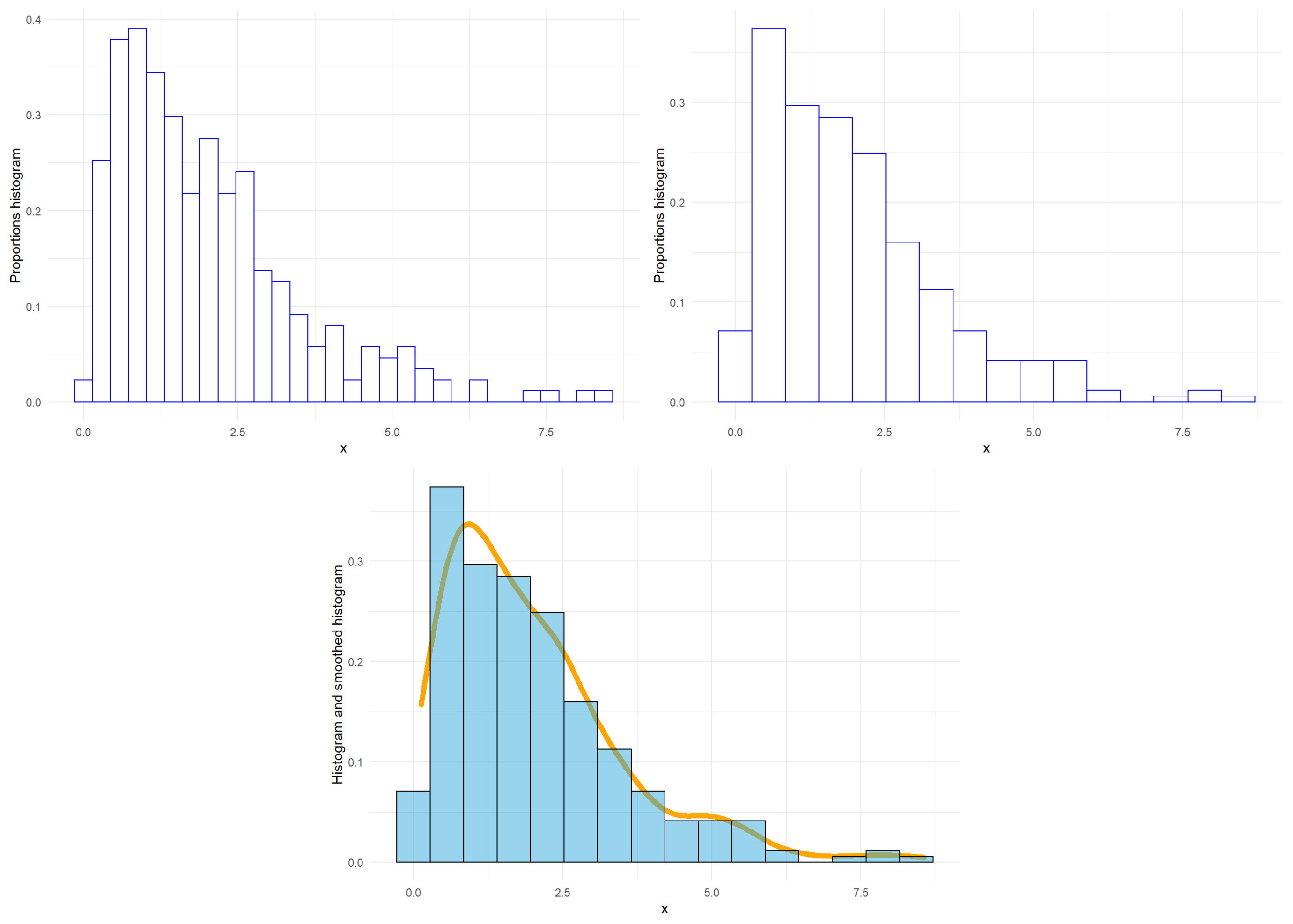
When we change the parameters in the histogram, we can see the following results. We will look at an example of this effect with a sample of size $n = 300$.

#Gamma Distribution

```
x <- data.frame(x=rgamma(300, shape=2, scale=1))

ggplot(x, aes(x=x)) +
  geom_histogram(aes(y=..density..),
                 col="blue", fill="white") +
  ylab("Proportions histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
ggplot(x, aes(x=x)) +
  geom_histogram(aes(y=..density..),
                 bins=nclass.FD(unlist(x)),
                 col="blue", fill="white") +
  ylab("Proportions histogram")

ggplot(x, aes(x=x)) +
  geom_density(col="orange", size=2) +
  geom_histogram(aes(y=..density..),
                 bins=nclass.FD(unlist(x)),
                 alpha=0.5, fill="#33AADE", color="black") +
  ylab("Histogram and smoothed histogram")
```

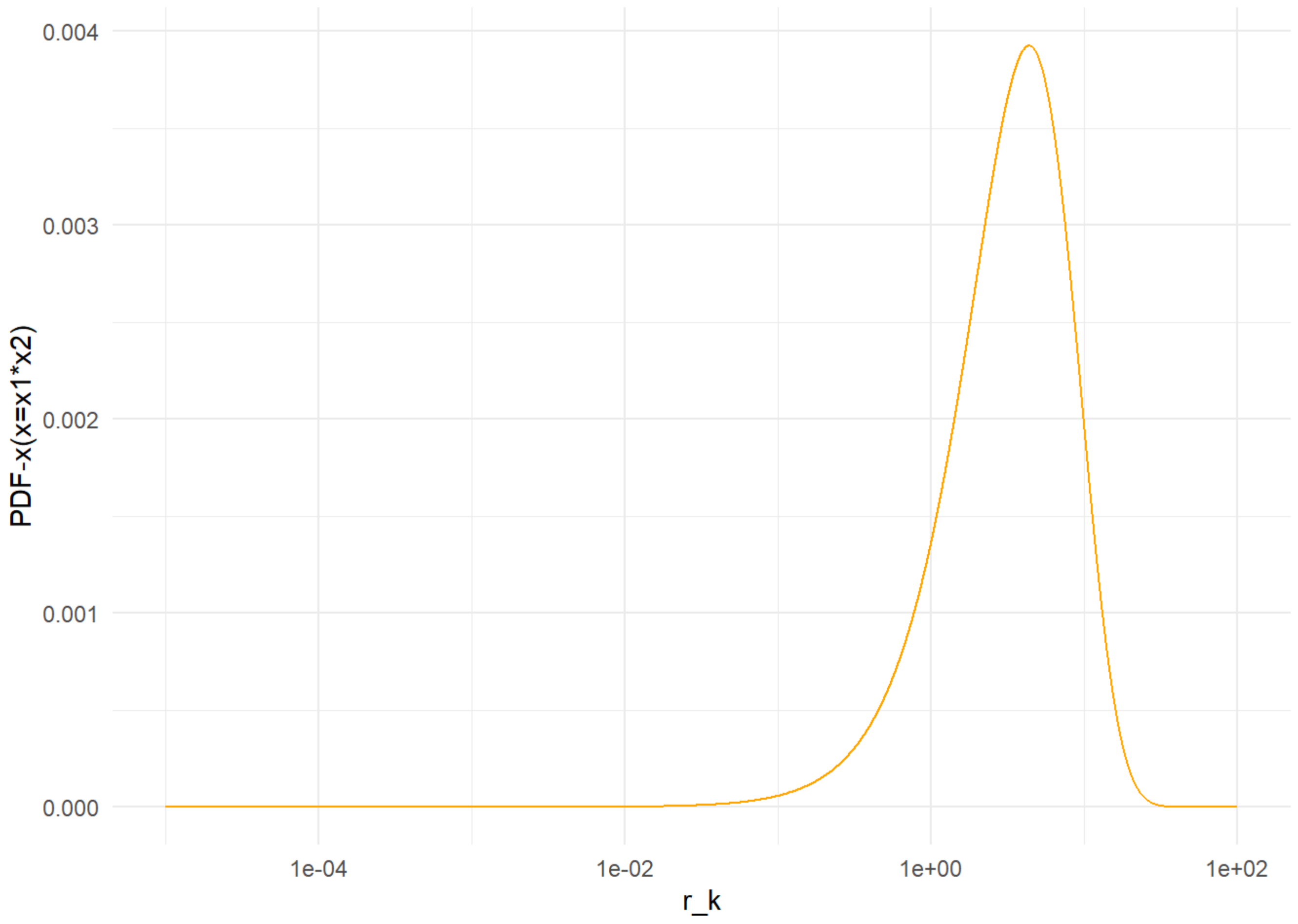


Histograms

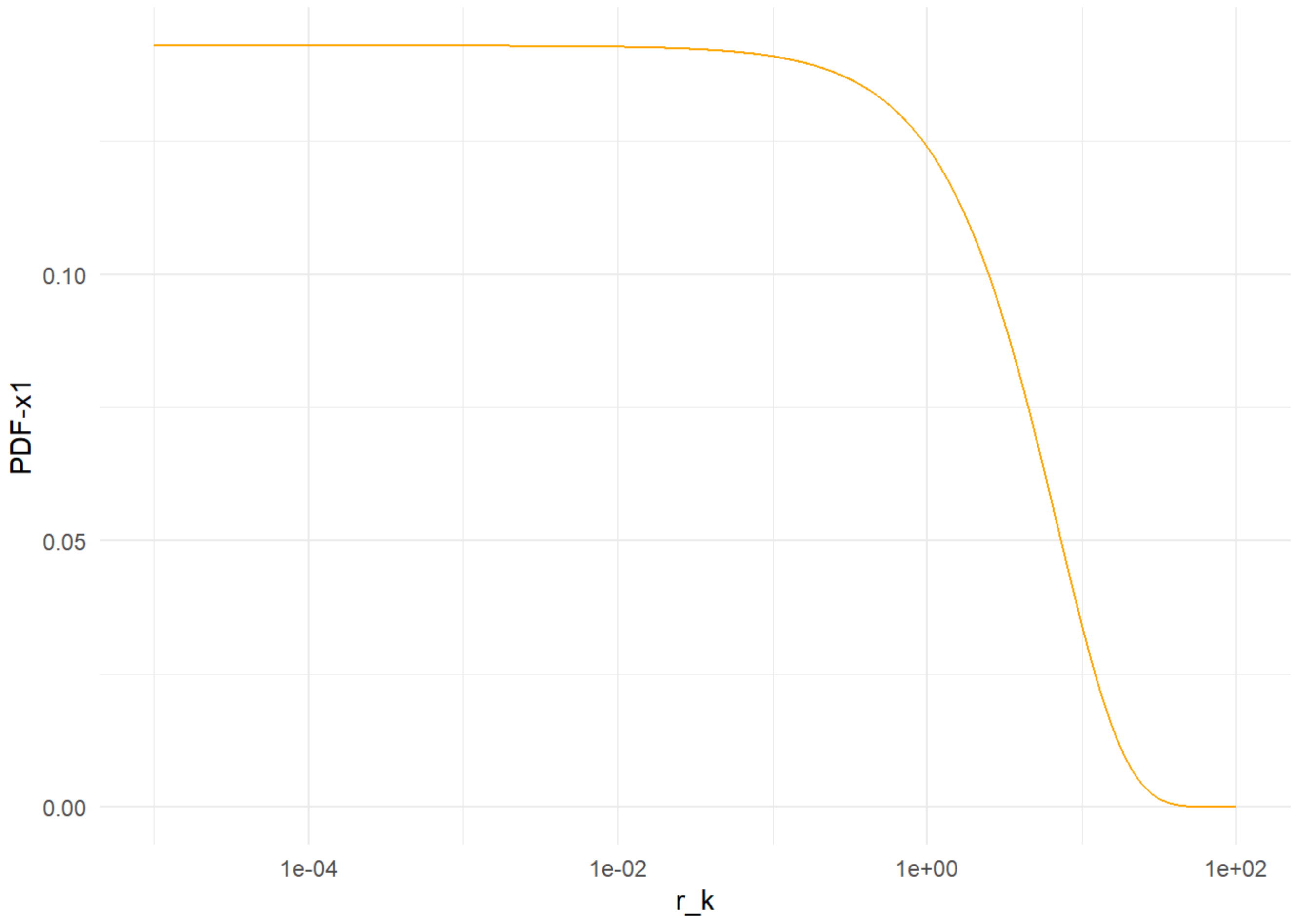
By comparing the first and second graphs, we can see that the larger the number of boxes, the more data we can get and the more difficult it is to process.

#K Distribution

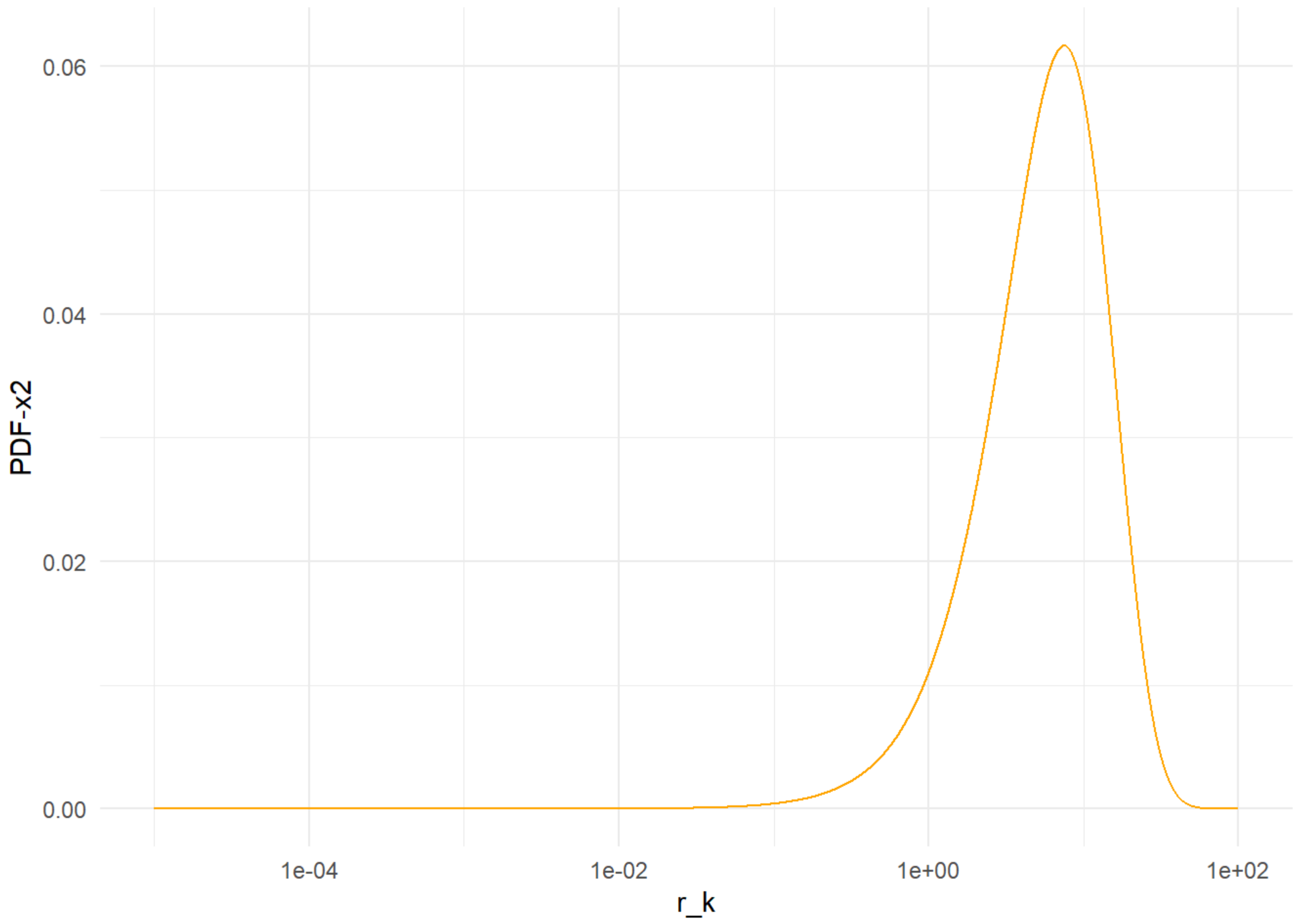
```
#Calculation process
a_K=5
v_K=2
L_K=7
pk_num=1e5
array_K=1e5

r_k<-seq(1e-5,1e2,length.out = array_K) # array
x1 <- dgamma(r_k, shape=1, scale=L_K)
x2 <- dgamma(r_k, shape=a_K/v_K, scale=a_K)
x=x1*x2

vector.x <- data.frame(r_G0=as.vector(r_k),x=as.vector(x))
vector.x1 <- data.frame(r_G0=as.vector(r_k),x=as.vector(x1))
vector.x2 <- data.frame(r_G0=as.vector(r_k),x=as.vector(x2))

ggplot(vector.x, aes(x=r_k, y=x, group=1)) +
  geom_line(linetype="solid",col="orange")+
  ylab("PDF-x(x=x1*x2)")+
  scale_x_log10()
```



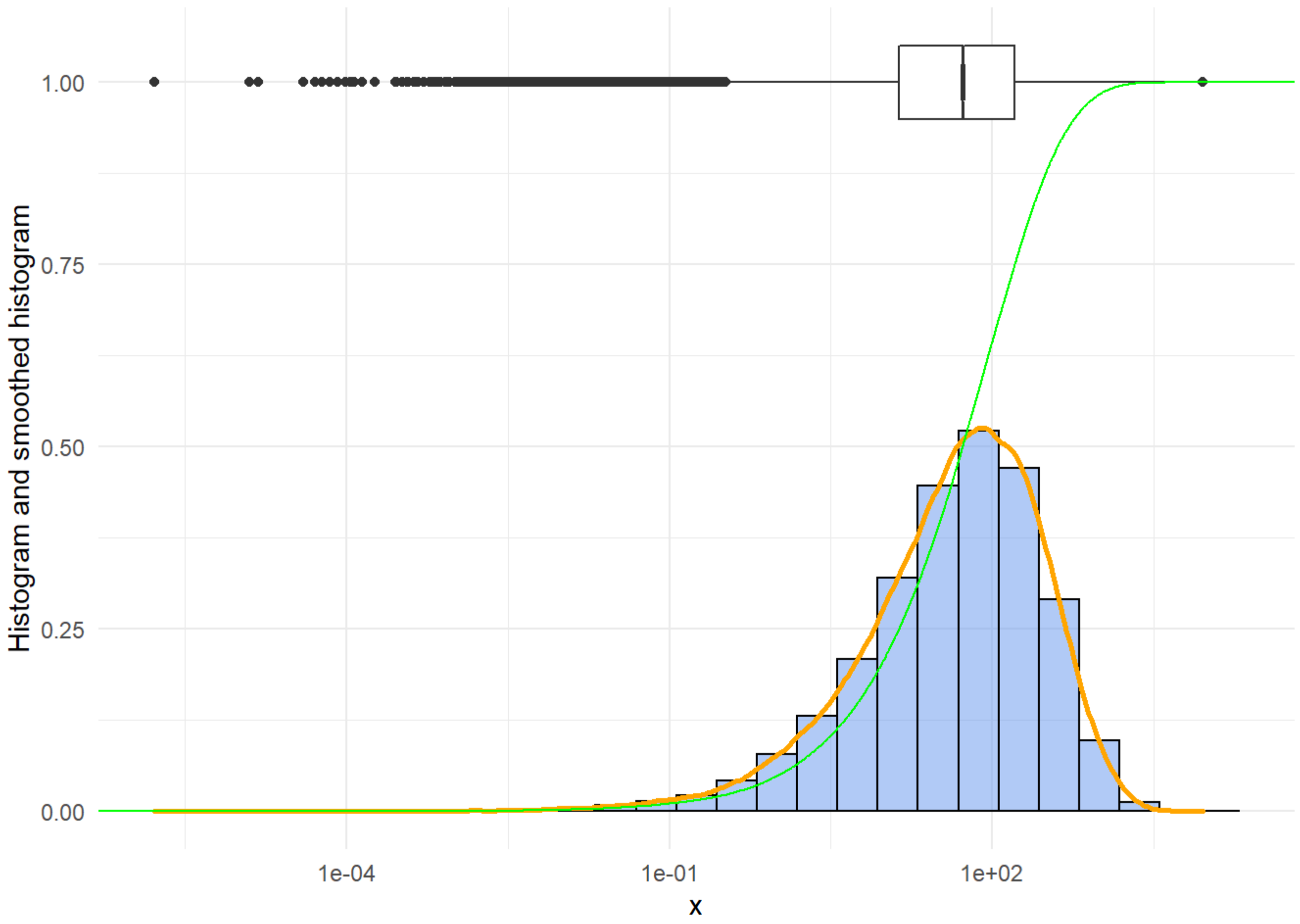
```
ggplot(vector.x1, aes(x=r_k, y=x1, group=1)) +
  geom_line(linetype="solid",col="orange")+
  ylab("PDF-x1")+
  scale_x_log10()
```



```
ggplot(vector.x2, aes(x=r_k, y=x2, group=1)) +
  geom_line(linetype="solid",col="orange")+
  ylab("PDF-x2")+
  scale_x_log10()
```

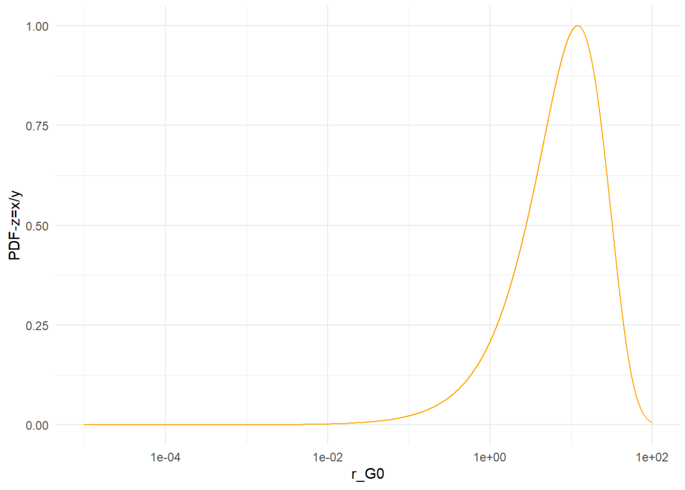


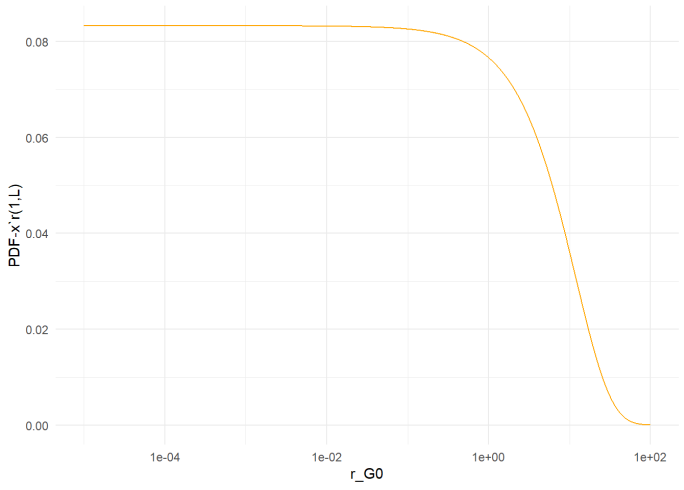
```
# K-random
r_k<-runif(pk_num, min = 0, max = 1) # 0-1 random
x1 <- qgamma(r_k, shape=1, scale=L_K)
x2 <- qgamma(r_k, shape=a_K/v_K, scale=a_K)
x=x1*x2
x<-data.frame(x) # x~K(a,v,L)

ggplot(x, aes(x=x)) +
  geom_histogram(aes(y=..density..),alpha=0.5, fill="#6495ED", color="black",bins=nclass.FD(unl
ist(x))/50) +
  geom_density(col="orange", size=1) +
  geom_boxplot(aes(y=1), width=.1, notch=TRUE)+
  stat_ecdf(col="green") +
  ylab("Histogram and smoothed histogram")+
  scale_x_log10()
```

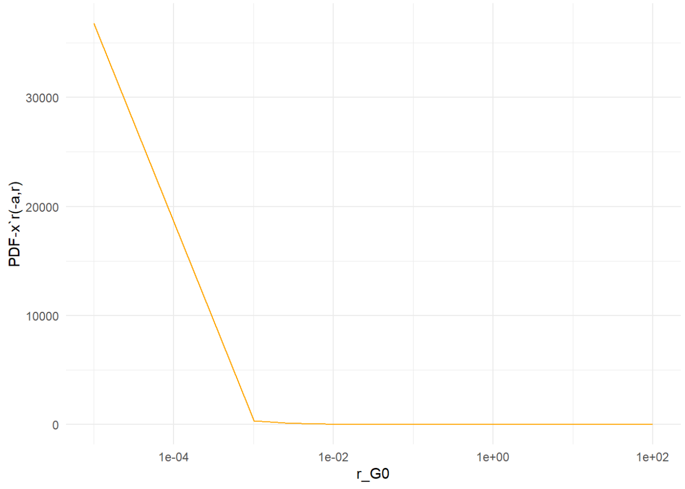


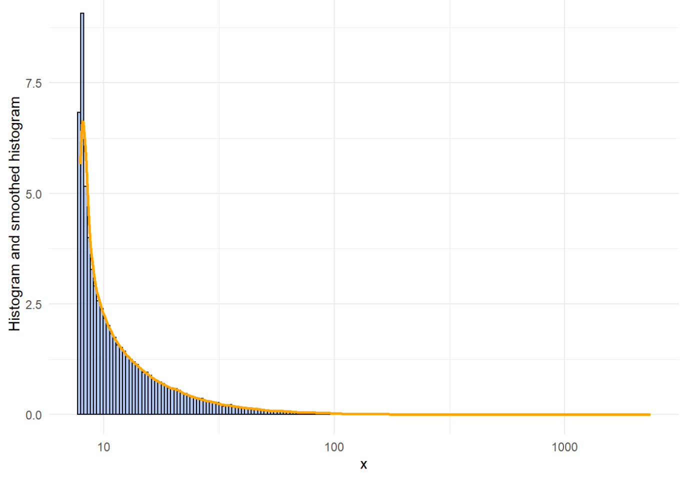
#G0 Distribution

```
a_G0=-2
r_G0=5
L_G0=12
pG0_num=1e5
array_G0=1e5

#Calculation process
r_G0<-seq(1e-5,1e2,length.out = array_G0) #  array
x1 <- dgamma(r_G0, shape=1, scale=L_G0)
x2 <- dgamma(r_G0, shape=-a_G0, scale=r_G0)
x=x1/x2
x<-data.frame(x)

vector.x <- data.frame(r_G0=as.vector(r_G0),x=as.vector(x))
vector.x1 <- data.frame(r_G0=as.vector(r_G0),x=as.vector(x1))
vector.x2 <- data.frame(r_G0=as.vector(r_G0),x=as.vector(x2))

ggplot(vector.x, aes(x=r_G0, y=x, group=1)) +
  geom_line(linetype="solid",col="orange")+
  ylab("PDF-z=x/y")+
  scale_x_log10()
ggplot(vector.x1, aes(x=r_G0, y=x1, group=1)) +
  geom_line(linetype="solid",col="orange")+
  ylab("PDF-x`r(1,L)")+
  scale_x_log10()
ggplot(vector.x2, aes(x=r_G0, y=x2, group=1)) +
  geom_line(linetype="solid",col="orange")+
  ylab("PDF-x`r(-a,r)")+
  scale_x_log10()

# G0-random
r_G0<-runif(pG0_num, min = 0, max = 1) # 0-1 random
x1 <- qgamma(r_G0, shape=1, scale=L_G0)
x2 <- qgamma(r_G0, shape=-a_G0, scale=r_G0)
x=x1/x2
x<-data.frame(x) # x~G0(a,r,L)

ggplot(x, aes(x=x)) +
  geom_histogram(aes(y=..density..),alpha=0.5, fill="#6495ED",color="black",bins=nclass.FD(unli
st(x))/50) +
  geom_density(col="orange", size=1) +
  ylab("Histogram and smoothed histogram")+
  scale_x_log10()
```


PDF-z=x/y
1.00
0.75
0.50
0.25
0.00
1e-04
1e-02
1e+00
1e+02
r_G0


PDF-x`r(1,L)
0.08
0.06
0.04
0.02
0.00
1e-04
1e-02
1e+00
1e+02
r_G0


PDF-x`r(-a,r)
30000
20000
10000
0
1e-04
1e-02
1e+00
1e+02
r_G0


Histogram and smoothed histogram
7.5
5.0
2.5
0.0
10
100
1000
x

Histograms