Assignment

Operational Statistics for SAR Imagery

Professor Alejandro C. Frery

Zhangyue Xiong

23020181154246

August 10, 2019

Contents

Assignment #1

Task

Read, summarize and comment this article [1].

Summary and Comment

Mathematical modelling aims at solving complex problems which can be described in a rigorous mathematical way that enables the use of computers for finding the solution. There are high number of variables requires strong computational effort in their solutions. In complex problems, there are many variables and values. minute errors in obtaining the eigenvalue and eigenvector or a matrix determinant, in calculating an average, a standard deviation or correlation coefficient, can lead to erroneous. To search for best approximate solutions, we need to consider some reasonable bounds of errors, imposes tight accuracy requirements on the computational platforms and its libraries or functions. However, little attention has been drawn to assess different platforms under the diversity of operational systems and hardware considering the accuracy of the results. Authers aim to test the accuracy of different platforms used in computational modeling, running on the same operational system and hardware.

They test the accuracy of three platforms used in computational modelling: MatLab, Octave and Scilab, running on i386 architecture and three operating systems(Windows, Ubuntu and Mac OS). They submitted them to numerical tests using standard data sets and using the function provided by each platform. In order to verify the stability of the results with respect to small departures from the original input, a Monte Carlo study was conducted in some of the datasets. They also used data provided by NIST(National Institute of Standards and Technology), a protocol which includes the computation of basic univariate statistics, linear regression and extremes of probability distributions.

The assessment was made comparing the results computed by the platforms with certified values, that is, known results, computing the number of correct significant digits. Regarding the computation of basic statics, Scilab is not only the worst performance platform but also the worst platform with respect to the stability of the results. Scilab presented the best performance when dealing with the binomial and t-Student deviation, and also when computing the cumulative distribution function of the Possion Law. When computing the F distributions, Octave produced the best result. Regrading the normal distribution, MatLab and Octave obtained the same good results, while Scilab produced bad results. The three platforms were acceptable when dealing with the gamma law. Matlab and Octave failed at computing the t-Student distribution. No single platform can be advised as safe for the linear regression problems here considered. Regarding the variability among operating systems, MatLab and Octave were equivalent and more consistent than Scilab in most of the situa-

tions under assessment.

I think the results of this article will provide some suggestions for selecting computational platforms in my future research.

Assignment #2

Task

Exercise 4.4. Obtain the expression of the density of $\mathcal{K}$-distributed amplitude data via the transformation $Z_A = \sqrt{Z_I}$. Compute its moments. Illustrate.

Exercise 4.8. Obtain the expression of the density of $\mathcal{G}^0$-distributed amplitude data via the transformation $Z_A = \sqrt{Z_I}$. Compute its moments. Illustrate.

Answer

Exercise 4.4.
The density of $\mathcal{K}$-distributed:

$$f_{Z_I}(z_I;\alpha,\lambda,L) = \frac{2\lambda L}{\Gamma(\alpha)\Gamma(L)}(\lambda L z_I)^{\frac{\alpha+L}{2}-1}K_{\alpha-L}(2\sqrt{\lambda L z_I})$$

When $Z_A = \sqrt{Z_I}$, which is $z_I = z_A^2$, the density of amplitude data Z_A is:

$$\begin{aligned}
f_{Z_A}(z_A;\alpha,\lambda,L) =& \frac{2\lambda L}{\Gamma(\alpha)\Gamma(L)}(z_A^2)'(\lambda L z_A^2)^{\frac{\alpha+L}{2}-1}K_{\alpha-L}2\sqrt{\lambda L z_A^2} \\
=& \frac{2\lambda L}{\Gamma(\alpha)\Gamma(L)}2z_A(\lambda L z_A^2)^{\frac{\alpha+L}{2}-1}K_{\alpha-L}2\sqrt{\lambda L z_A^2} \\
=& \frac{4\lambda L}{\Gamma(\alpha)\Gamma(L)}(\lambda L)^{\frac{\alpha+L}{2}-1}z_A^{\alpha+L-1}K_{\alpha-L}2\sqrt{\lambda L z_A^2}
\end{aligned}$$

The k-order moments of Z_I are:

$$E(Z_I^k) = (\lambda L)^{-k}\frac{\Gamma(L+k)\Gamma(\alpha+k)}{\Gamma(L)\Gamma(\alpha)}$$

The k-order moments of Z_A are:

$$\begin{aligned}
E(Z_A^k) =& E(Z_I^{\frac{k}{2}}) \\
=& (\lambda L)^{-\frac{k}{2}}\frac{\Gamma(L+\frac{k}{2})\Gamma(\alpha+\frac{k}{2})}{\Gamma(L)\Gamma(\alpha)}
\end{aligned}$$

Exercise 4.8.
The the density of $\mathcal{G}^0$ distribution:

$$f_{Z_I}(z_I;\alpha,\gamma,L) = \frac{L^L\Gamma(L-\alpha)}{\gamma^\alpha\Gamma(L)\Gamma(-\alpha)}\frac{z_I^{L-1}}{(\gamma+Lz_I)^{L-\alpha'}}$$

When $Z_A = \sqrt{Z_I}$, which is $z_I = z_A^2$, the density of amplitude data Z_A is:

$$\begin{aligned} f_{Z_A}(z_A;\alpha,\gamma,L) =& \frac{L^L\Gamma(L-\alpha)}{\gamma^\alpha\Gamma(L)\Gamma(-\alpha)}(z_A^2)'\frac{(z_A^2)^{L-1}}{(\gamma+Lz_A^2)^{L-\alpha'}} \\ =& \frac{L^L\Gamma(L-\alpha)}{\gamma^\alpha\Gamma(L)\Gamma(-\alpha)}2z_A\frac{(z_A^2)^{L-1}}{(\gamma+Lz_A^2)^{L-\alpha'}} \\ =& \frac{2L^L\Gamma(L-\alpha)}{\gamma^\alpha\Gamma(L)\Gamma(-\alpha)}\frac{z_A^{2L-1}}{(\gamma+Lz_A^2)^{L-\alpha'}} \end{aligned}$$

The k-order moments of Z_I are:

$$E(Z_I^k) = (\gamma/L)^k\frac{\Gamma(L+k)\Gamma(-\alpha-k)}{\Gamma(L)\Gamma(-\alpha)}$$

The k-order moments of Z_A are:

$$\begin{aligned} E(Z_A^k) =& E(Z_I^{\frac{k}{2}}) \\ =& (\gamma/L)^{\frac{k}{2}}\frac{\Gamma(L+\frac{k}{2})\Gamma(-\alpha-\frac{k}{2})}{\Gamma(L)\Gamma(-\alpha)} \end{aligned}$$

Assignment #3

Task

Including today's activities

Activities

Test function

Try to implement some function using R language

```
require(ggplot2)
require(ggthemes)

# square function

f.square <- function(x){
x^2
}
ggplot(data = data.frame(x = seq(-pi,pi,length.out = 500)),aes(x=x)) +
stat_function(fun = f.square, geom = "line", size=2, col="black") +
theme_classic() +
theme(text = element_text(size = 20)) +
xlab("x") + ylab("Square")
ggsave(file = "F:/assignment/Figures/square.pdf")
```

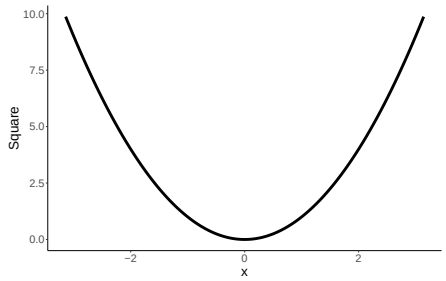


Figure 1: Square

```r
# power function

f.power <- function(x,a){
x^a
}
ggplot(data = data.frame(x = seq(-pi,pi,length.out = 500)),aes(x=x)) +
stat_function(fun = f.power, geom = "line", size=2, col="blue",args = (a=1)) +
stat_function(fun = f.power, geom = "line", size=2, col="red",args = (a=2)) +
stat_function(fun = f.power, geom = "line", size=2, col="green",args = (a=3)) +
theme_classic() +
theme(text = element_text(size = 20)) +
xlab("x") + ylab("Power")
ggsave(file = "F:/assignment/Figures/power.pdf")
```

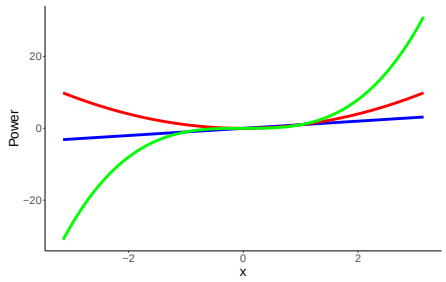


Figure 2: Power

```r
# sin(1/x)*x function

f.sinx <- function(x){
```

```
x*sin(1/x)
}
ggplot(data = data.frame(x = seq(-pi,pi,length.out = 500)),aes(x=x)) +
stat_function(fun = f.sinx, geom = "line", size=2, col="black") +
theme_classic() +
theme(text = element_text(size = 20)) +
xlab("x") + ylab("sinx")
ggsave(file = "F:/assignment/Figures/sinx.pdf")
```

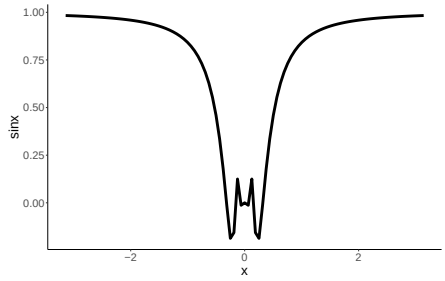


Figure 3: Sinx

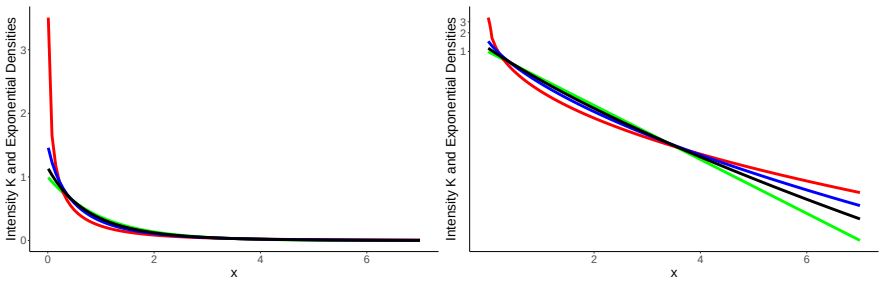
$\mathcal{K}$-distributed
Implement $\mathcal{K}$-distributed using R language

```
# Intensity K distributions

f.K <- function(x,p_alpha,p_lambda,p_L){
(2*p_lambda*p_L/(gamma(p_alpha)*gamma(p_L)))*((p_lambda*p_L*x)^((p_alpha+p_L)/2-1))*besselK(2*sqrt(p_lambda*p_L*x),p_alpha-p_L)
}
ggplot(data=data.frame(x=seq(0.01, 7, length.out = 500)), aes(x=x)) +
stat_function(fun=dexp, geom = "line", size=2, col="green", args = (mean=1)) +
stat_function(fun=f.K, geom = "line", size=2, col="red", args = list(p_alpha=1, p_lambda=1, p_L=1)) +
stat_function(fun=f.K, geom = "line", size=2, col="blue", args = list(p_alpha=3, p_lambda=3, p_L=1)) +
stat_function(fun=f.K, geom = "line", size=2, col="black", args = list(p_alpha=8, p_lambda=8, p_L=1)) +
theme_classic() +
theme(text = element_text(size=20)) +
```

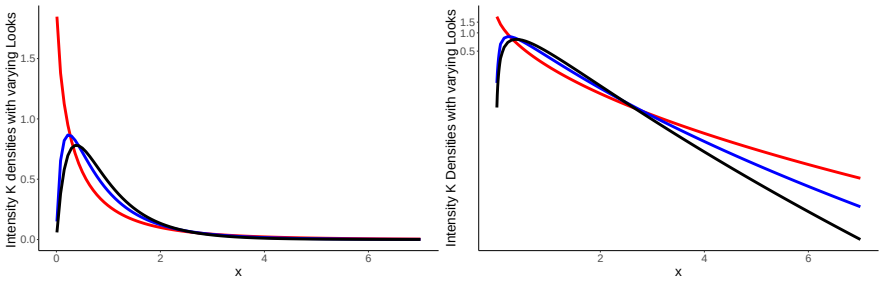
```
13 xlab("x") + ylab("Intensity K and Exponential Densities")
14 ggsave(file = "F:/assignment/Figures/KIDensities.pdf")
15
16 ggplot(data=data.frame(x=seq(0.01, 7, length.out = 500)),
       aes(x=x)) +
17 stat_function(fun=dexp, geom = "line", size=2, col="green
     ", args = (mean=1)) +
18 stat_function(fun=f.K, geom = "line", size=2, col="red",
     args = list(p_alpha=1, p_lambda=1, p_L=1)) +
19 stat_function(fun=f.K, geom = "line", size=2, col="blue",
       args = list(p_alpha=3, p_lambda=3, p_L=1)) +
20 stat_function(fun=f.K, geom = "line", size=2, col="black"
     , args = list(p_alpha=8, p_lambda=8, p_L=1)) +
21 theme_classic() +
22 theme(text = element_text(size=20)) +
23 coord_trans(y="log10") +
24 xlab("x") + ylab("Intensity K and Exponential Densities")
25 ggsave(file="F:/assignment/Figures/KIDensitiesSemilog.pdf
     ")
26
27 ggplot(data=data.frame(x=seq(0.01, 7, length.out = 500)),
       aes(x=x)) +
28 stat_function(fun=f.K, geom = "line", size=2, col="red",
     args = list(p_alpha=2, p_lambda=2, p_L=1)) +
29 stat_function(fun=f.K, geom = "line", size=2, col="blue",
       args = list(p_alpha=2, p_lambda=2, p_L=3)) +
30 stat_function(fun=f.K, geom = "line", size=2, col="black"
     , args = list(p_alpha=2, p_lambda=2, p_L=8)) +
31 theme_classic() +
32 theme(text = element_text(size=20)) +
33 xlab("x") + ylab("Intensity K densities with varying
     Looks ")
34 ggsave(file="F:/assignment/Figures/KIDensitiesLooks.pdf")
35
36 ggplot(data=data.frame(x=seq(0.01, 7, length.out = 500)),
       aes(x=x)) +
37 stat_function(fun=f.K, geom = "line", size=2, col="red",
     args = list(p_alpha=2, p_lambda=2, p_L=1)) +
38 stat_function(fun=f.K, geom = "line", size=2, col="blue",
       args = list(p_alpha=2, p_lambda=2, p_L=3)) +
39 stat_function(fun=f.K, geom = "line", size=2, col="black"
     , args = list(p_alpha=2, p_lambda=2, p_L=8)) +
40 theme_classic() +
41 theme(text = element_text(size=20)) +
42 coord_trans(y="log10") +
43 xlab("x") + ylab("Intensity K Densities with varying
```

```
    Looks")
ggsave(file="F:/assignment/Figures/
    KIDensitiesSemilogLooks.pdf")
```



(a) Densities (b) Densities in semilog scale

Figure 4: Densities in linear and semilogarithmic scale of the E(1) (green) and $\mathcal{K}$ distributions with unitary mean ($\alpha \in \{1, 3, 8\}$ in red, blue, and black, resp.



(a) Densities (b) Densities in semilog scale

Figure 5: Densities in linear and semilogarithmic scale $\mathcal{K}(2, 2, L)$ distributions with unitary mean and $L \in \{1, 3, 8\}$ in red, blue, and black, resp.

$\mathcal{G}^0$-distributed
Implement $\mathcal{G}^0$ using R language

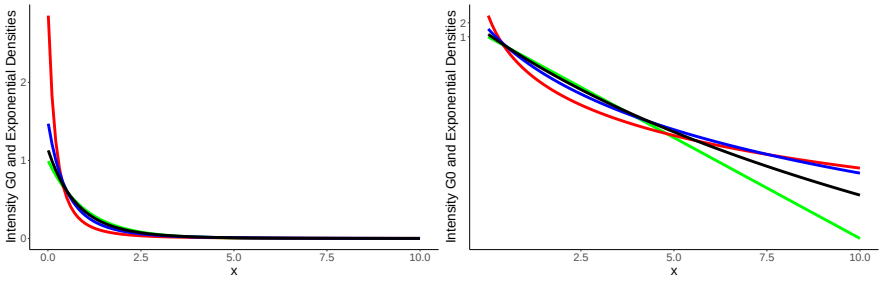
```
# Intensity GI0 distributions

f.GI0 <- function(x,p_alpha,p_gamma,p_L){
((p_L^p_L*gamma(p_L-p_alpha))/(p_gamma^p_alpha*gamma(p_L)
    *gamma(-p_alpha)))*(x^(p_L-1)/(p_gamma+p_L*x)^(p_L-p_
    alpha))
}

ggplot(data=data.frame(x=seq(0.01, 10, length.out = 500))
    , aes(x=x)) +
```

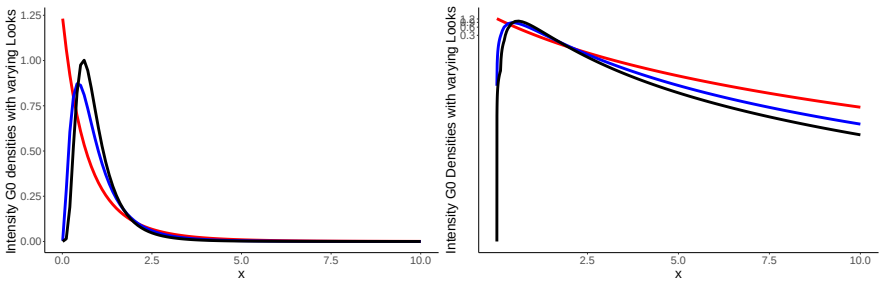
```
8  stat_function(fun=dexp, geom = "line", size=2, col="green
       ", args = (mean=1)) +
9  stat_function(fun=f.GI0, geom = "line", size=2, col="red"
       , args = list(p_alpha=-1.5, p_gamma=.5, p_L=1)) +
10 stat_function(fun=f.GI0, geom = "line", size=2, col="blue
       ", args = list(p_alpha=-3, p_gamma=2, p_L=1)) +
11 stat_function(fun=f.GI0, geom = "line", size=2, col="
       black", args = list(p_alpha=-8, p_gamma=7, p_L=1)) +
12 theme_classic() +
13 theme(text = element_text(size=20)) +
14 xlab("x") + ylab("Intensity G0 and Exponential Densities"
       )
15 ggsave(file="F:/assignment/Figures/GI0Densities.pdf")
16
17 ggplot(data=data.frame(x=seq(0.01, 10, length.out = 500))
       , aes(x=x)) +
18 stat_function(fun=dexp, geom = "line", size=2, col="green
       ", args = (mean=1)) +
19 stat_function(fun=f.GI0, geom = "line", size=2, col="red"
       , args = list(p_alpha=-1.5, p_gamma=.5, p_L=1)) +
20 stat_function(fun=f.GI0, geom = "line", size=2, col="blue
       ", args = list(p_alpha=-3, p_gamma=2, p_L=1)) +
21 stat_function(fun=f.GI0, geom = "line", size=2, col="
       black", args = list(p_alpha=-8, p_gamma=7, p_L=1)) +
22 theme_classic() +
23 theme(text = element_text(size=20)) +
24 coord_trans(y="log10") +
25 xlab("x") + ylab("Intensity G0 and Exponential Densities"
       )
26 ggsave(file="F:/assignment/Figures/GI0DensitiesSemilog.
       pdf")
27
28 ggplot(data=data.frame(x=seq(0.01, 10, length.out = 500))
       , aes(x=x)) +
29 stat_function(fun=f.GI0, geom = "line", size=2, col="red"
       , args = list(p_alpha=-5, p_gamma=4, p_L=1)) +
30 stat_function(fun=f.GI0, geom = "line", size=2, col="blue
       ", args = list(p_alpha=-5, p_gamma=4, p_L=3)) +
31 stat_function(fun=f.GI0, geom = "line", size=2, col="
       black", args = list(p_alpha=-5, p_gamma=4, p_L=8)) +
32 theme_classic() +
33 theme(text = element_text(size=20)) +
34 xlab("x") + ylab("Intensity G0 densities with varying
       Looks ")
35 ggsave(file="F:/assignment/Figures/GI0DensitiesLooks.pdf"
       )
```

```
36
37 ggplot(data=data.frame(x=seq(0.01, 10, length.out = 500))
       , aes(x=x)) +
38 stat_function(fun=f.GI0, geom = "line", size=2, col="red"
       , args = list(p_alpha=-5, p_gamma=4, p_L=1)) +
39 stat_function(fun=f.GI0, geom = "line", size=2, col="blue
       ", args = list(p_alpha=-5, p_gamma=4, p_L=3)) +
40 stat_function(fun=f.GI0, geom = "line", size=2, col="
       black", args = list(p_alpha=-5, p_gamma=4, p_L=8)) +
41 theme_classic() +
42 theme(text = element_text(size=20)) +
43 coord_trans(y="log10") +
44 xlab("x") + ylab("Intensity G0 Densities with varying
       Looks")
```



(a) Densities (b) Densities in semilog scale

Figure 6: Densities in linear and semi-logarithmic scale of the E(1) (green) and $\mathcal{G}^0$ distributions with unitary mean and $\alpha \in \{-1.5, -3, -8\}$ in red, blue, and black, resp.



(a) Densities (b) Densities in semilog scale

Figure 7: Densities in linear and semilogarithmic scale $\mathcal{G}^0(-5, 4, L)$ distributions with unitary mean and $L \in \{1, 3, 8\}$ in red, blue, and black, resp.

Assignment #4

Task

Including today's activities

Activities

Solving the maximum likelihood function estimate.
Step:

1. $D(\theta)$
2. $f_X(\underline{X};\theta)$
3. $L(\theta;\underline{X})$
4. $argmax L(\theta;\underline{X})$

Assignment #5

Task

Including today's activities

Activities

Parameter Estimation
Build estimators to analysis the data from an urban

```
# estimator
GI0.Estimator.m1m2 <- function(z,L){
m1 <- mean(z)
m2 <- mean(z^2)
m212 <- m2/m1^2

a <- -2 - (L+1) / (L * m212)
g <- m1 * (2 + (L+1) / (L *m212))

return(list("alpha"=a,"gamma"=g))
}


# LogLikelihoodLknown
LogLikelihoodLknown <- function(params){
```

```
16 p_alpha <- -abs(params[1])
17 p_gamma <- abs(params[2])
18 p_L <- abs(params[3])
19
20 n <- length(z)
21
22 return(
23 n*(lgamma(p_L-p_alpha) - p_alpha*log(p_gamma) - lgamma(-p
       _alpha)) + (p_alpha-p_L)*sum(log(p_gamma + z*p_L))
24 )
25 }
26
27 Intensity_restricted <- subset(vUrbanHV$UHV, subset =
       vUrbanHV$UHV <= 100000)
28 binwidth_restricted <- 2*IQR(Intensity_restricted)*length
       (Intensity_restricted)^(-1/3)
29 vUrbanHV <- data.frame(UHV=as.vector(UrbanHV
       [90:200,50:100]))
30 meanUHV <- mean(vUrbanHV$UHV)
31 secondUHV <- mean(vUrbanHV$UHV^2)
32 a <- 2 * (1-CoeffVariation2) / (CoeffVariation2-2)
33 g <- meanUHV * (-a-1)
34 z <- vUrbanHV$UHV
35
36 estim.Urban <- GI0.Estimator.m1m2(UrbanHV,1)
37 estim.UrbanML <- maxNR(LogLikelihoodLknown,
38 start = c(estim.Urban$alpha,estim.Urban$gamma,1),
39 activePar = c(TRUE,TRUE,FALSE))$estimate[1:2]
40
41 ggplot(data=vUrbanHV, aes(x=UHV)) +
42 geom_histogram(aes(y=..density..),
43 binwidth = binwidth_restricted) +
44 xlim(0,200000) +
45 stat_function(fun=dexp, args=list(rate=1/meanUHV),
46 col="red", lwd=2, alpha=.7) +
47 stat_function(fun=f.GI0, args = list(p_alpha=estim.Urban$
       alpha, p_gamma=estim.Urban$gamma, p_L=1),
48 col="blue", lwd=2, alpha=.7) +
49 stat_function(fun=f.GI0, args = list(p_alpha=estim.
       UrbanML[1], p_gamma=estim.UrbanML[2], p_L=1),
50 col="green", lwd=2, alpha=.7) +
51 xlab("Intensities from the Urban Area") +
52 ylab("Histogram, and fitted Exponential and G0 Laws") +
53 ggtitle("Restricted Histogram") +
54 theme_few()
```

```
55 ggsave(file="F:/assignment/Figures/
       HistogramRestrictedUrban.pdf")
```

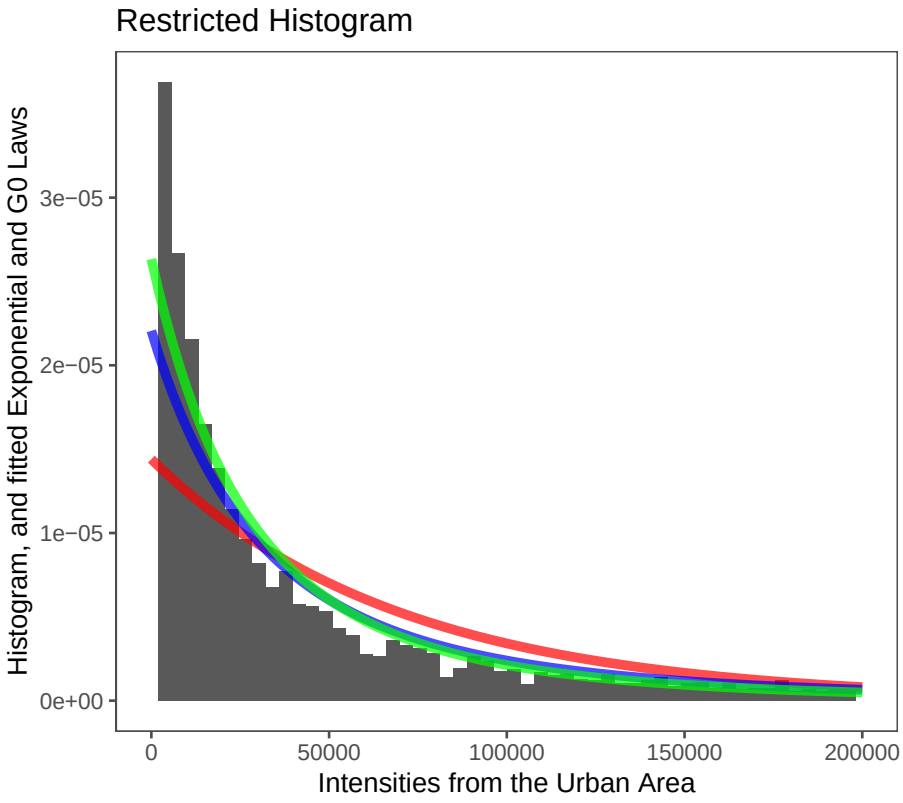


Figure 8: Restricted histogram of the urban area data, with fitted exponential (red) and $\mathcal{G}^0$ densities; the latter is shown with the parameters estimated by the moments (blue) and maximum likelihood (green) methods.

The better quality of the fit with the $\mathcal{G}^0$ model is noticeable. There seems to be evidence that the maximum likelihood estimates provide a better fit than that with the moments estimates.

References

[1] Eliana S de Almeida, Antonio C Medeiros, and Alejandro C Frery. How good are matlab, octave and scilab for computational modelling? *Computational & Applied Mathematics*, 31(3):523–538, 2012.