Exercícios Aula 7

Giovana Santos Oliveira
NUSP 10266976

12 de Maio de 2020

Exercício 1

Use os dados da tabela tab_virgo.dat para ajustar a relação Dn−σ. Use a função lm() para fazer uma regressão linear ordinária e a função rlm() para fazer uma regressão robusta. Compare os resultados.

O ajuste lm() foi feito usando o seguinte código em R:

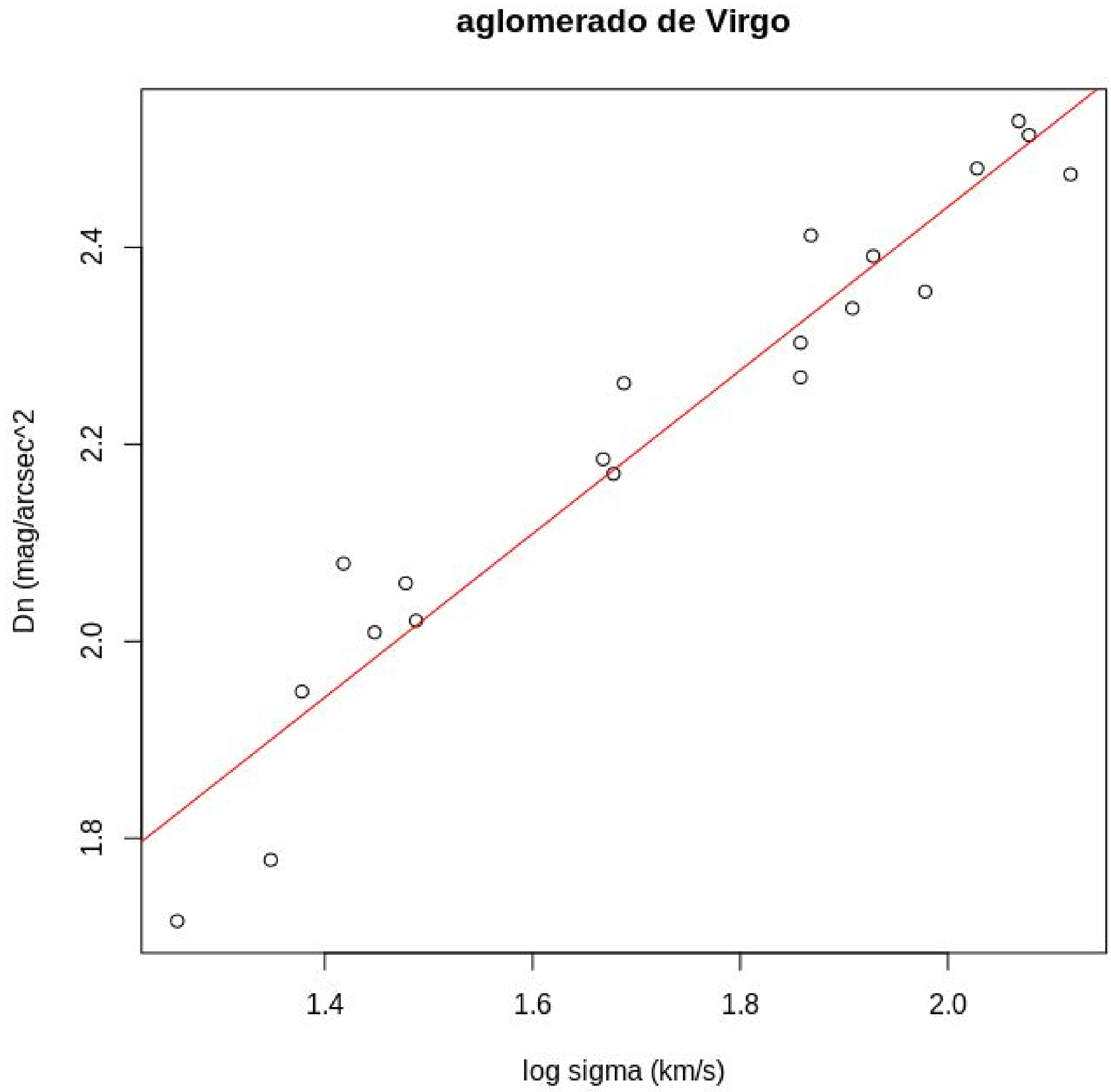
```
dados <- read.table(file = "tab_virgo.dat", header = TRUE)

log_Dn <- dados[,4]
log_sigma <- dados[,8]

x <- log_Dn
y <- log_sigma

modelo = lm(y~x)
plot(x,y,main='aglomerado de Virgo',xlab='log sigma (km/s)',ylab='Dn (mag/arcsec^2')
abline(modelo, col='red')

print(modelo)
```

Na saída foi obtido as seguintes informações a serem analisadas adiante:

Call:
lm(formula = y ~ x)

Coefficients:

(Intercept) *x*
0.7806 *0.8303*



Para a regressão robusta, rlm(), foi escrito o seguinte código em R, onde é feito um ajuste assim como o feito acima usando apenas lm(y~x), mas com dados que seriam necessários para utilizar o rlm(). Foi colocado nos dados um outlier e feito um ajuste normal com este outlier e um ajuste robusto com o outlier para comparar. Segue o código e suas saídas:

```
dados1 = data.frame(cbind(x,y))
#vou ajustar uma reta
ajuste = lm(y~x, data = dados1)
```

print(ajuste)

Call:
lm(formula = y ~ x, data = dados1)

Coefficients:
(Intercept)   x
0.7806   0.8303

#irei introduzir um outilier (x,y) = (0.5,19)
outlier = data.frame(x=0.5,y=19)
#introduzirei este outlier no final da tabela 'dados'
novos_dados1 = rbind(dados1, outlier)

# e refazer o ajuste:
ajuste1 = lm(y~x, data = novos_dados1)
print(ajuste1)

Call:
lm(formula = y ~ x, data = novos_dados1)

Coefficients:
(Intercept)   x
13.631   -6.363

#ajuste robusto usando o pacote MASS - diminui o peso dos outliers

library(MASS)
ajuste2 = rlm(y~x, data = novos_dados1)
print(ajuste2)

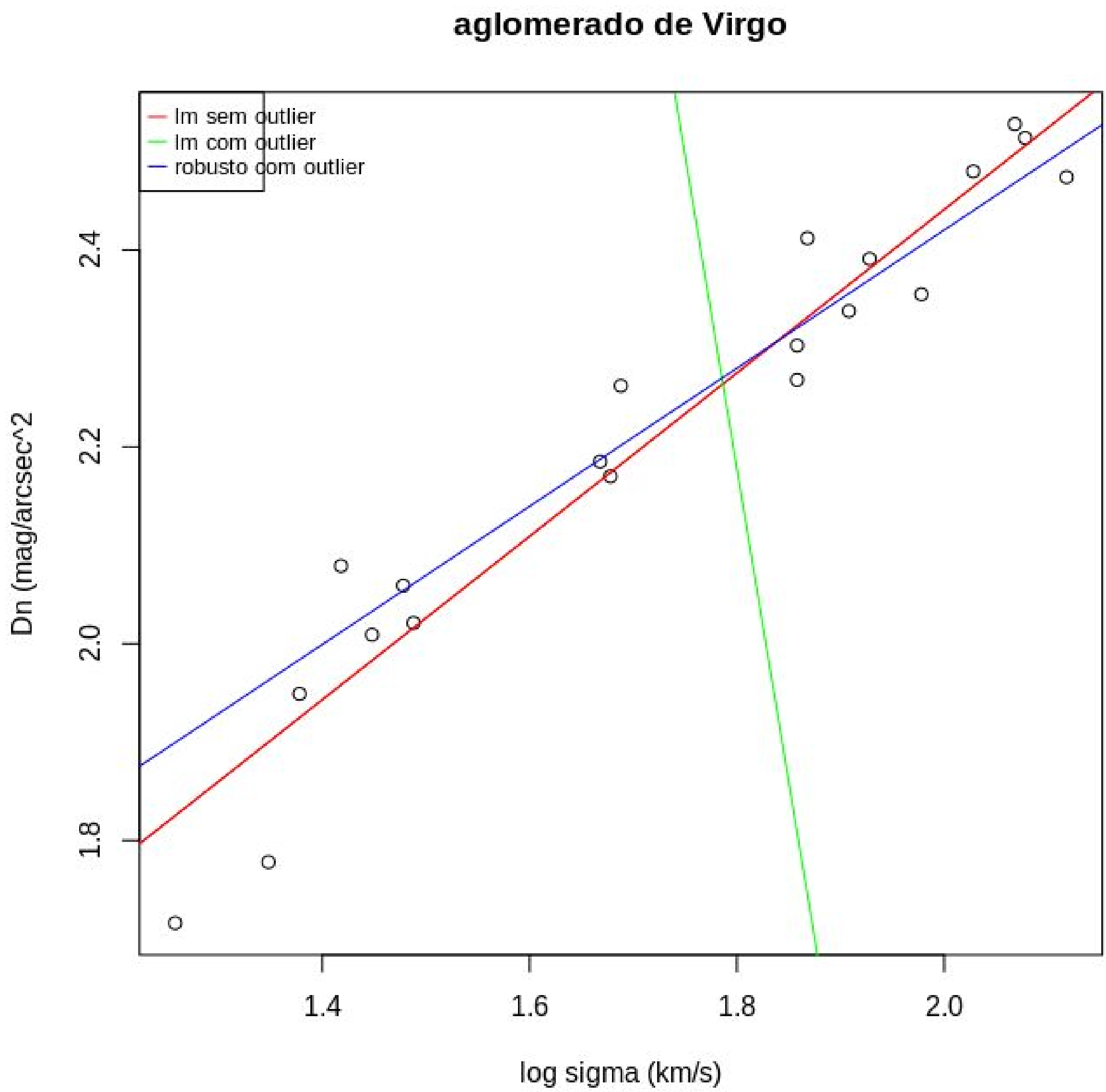
Call:
rlm(formula = y ~ x, data = novos_dados1)
Converged in 10 iterations

Coefficients:
(Intercept)   x
1.0163625   0.7020272

Degrees of freedom: 21 total; 19 residual
Scale estimate: 0.0594

```
print(coef(summary(ajuste)))
print(coef(summary(ajuste1)))
print(coef(summary(ajuste2)))

#visualização
plot(novos_dados, main = 'aglomerado de Virgo')
abline(ajuste, col = 'red')
abline(ajuste1, col = 'green')
abline(ajuste2, col = 'blue')
legend('topleft',legend=c("lm sem outlier", "lm com outlier", "robusto com outlier"),
                          col = c("red","green","blue"), lty=1, cex=0.8)
```

```
> print(coef(summary(ajuste)))
         Estimate Std. Error   t value         Pr(>|t|)
(Intercept) 0.7805959 0.09212393  8.473323 1.070633e-07
x        0.8303151 0.05270288 15.754644 5.654133e-12
> print(coef(summary(ajuste1)))
         Estimate Std. Error   t value         Pr(>|t|)
(Intercept) 13.630669   2.845708  4.789903 0.0001273261
x        -6.362815   1.664793 -3.821985 0.0011505594
> print(coef(summary(ajuste2)))
                  Value Std. Error  t value
(Intercept) 1.0163625 0.06624069 15.34348
x        0.7020272 0.03875206 18.11587
```



Análise dos resultados:

No ajuste lm() normal sem outlier teve-se como parâmetros para y e x os resultados *0.7806 0.8303*. No ajuste lm() normal com outlier os resultados foram *13.631 -6.363*. No ajuste robusto, rlm(), com outlier, o que se obteve foi *1.0163625 0.7020272*. Observa-se pelo gráfico segundo gráfico que mesmo com o outlier, um ponto tão fora do conjunto que o ajuste normal com ele foi uma reta que vai totalmente em outra direção, mesmo assim o ajuste robusto com o outlier conseguiu ter parâmetros próximos do ajuste normal sem outlier.

Exercício 2

A tabela tab_virgo.dat contém também um indicador de metalicidade denominado Mg2, associado ao magnésio. Usando mínimos quadrados linear, ajuste este indice em função do brilho superficial e da dispersão de velocidades (conjuntamente!).

Para fazer este exercício foi escrito o seguinte código em R:

```
dados <- read.table(file = "tab_virgo.dat", header = TRUE)

Sigmae <- dados[,7]
log_sigma <- dados[,8]
Mg2 <- dados[,9]

x <- log_sigma
y <- Sigmae
z <- Mg2

chi_2 = lm(z~x+y)
print(summary(chi_2))
```

Obteve-se como saída do programa os seguintes resultados:

```
Call:
lm(formula = z ~ x + y)

Residuals:
      Min      1Q    Median       3Q      Max
-0.023324 -0.009331 -0.002051  0.011551  0.020893

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.308387   0.093733  -3.290  0.00432 **
x       0.228062   0.013817  16.506 6.72e-12 ***
y       0.003663   0.003972   0.922  0.36931
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01413 on 17 degrees of freedom
Multiple R-squared:  0.942,    Adjusted R-squared:  0.9352
```

F-statistic: 138.1 on 2 and 17 DF, p-value: 3.068e-11

O ajuste feito com mínimos quadrados utilizando duas váriaveis independetes e uma dependente teve o seguinte resultado para seus parâmetros:

Mg2   -0.308387
log_sigma   0.228062
Sigmae   0.003663

Exercício 3

Um aglomerado de galáxias tem um perfil projetado de intensidade de raios-X como o descrito na tabela raiosX.dat. Nessa tabela a intensidade está em unidades arbitrárias e o raio em Mpc. Ajuste estes dados com um perfil do tipo:

I(R) =I0*[1+(R/Rc)^2]^(−3β+12),

com 3 parâmetros: I0, Rc e β. Use nls(). Compare esse resultado com o que se obtém usando no ajuste log10(I). O que você conclui?

Os ajustes feitos tanto para a função original quanto para o log10 foi feito com os códigos em R abaixo e seus respectivos resultados:

```
raiosX <- read.table("raiosX.dat", header=T)

attach(raiosX)

#nls
raiosX.fit <- nls(I~I0*(1+(R_Mpc/Rc)^2)^(-3*B + 1/2), data=raiosX,
                start = list(I0=1.,Rc=1.,B=1.), model=T)

print(summary(raiosX.fit))
```

----

*Formula: I ~ I0 * (1 + (R_Mpc/Rc)^2)^(-3 * B + 1/2)*

Parameters:
```
   Estimate Std. Error t value Pr(>|t|)
I0   8.3040     0.5597  14.836  0.00012 ***
Rc   0.3128   0.1041   3.005  0.03975 *
```

B   0.8791   0.3295   2.668 0.05593 .
---
*Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1*

Residual standard error: 0.2499 on 4 degrees of freedom

Number of iterations to convergence: 12
Achieved convergence tolerance: 3.347e-06

```
#nls
raiosX.fit <- nls(log10(I)~log10(I0)+(-3*B+1/2)*log10(1+(R_Mpc/Rc)^2), data=raiosX,
                  start = list(I0=1.,Rc=1.,B=1.), model=T)
```

```
Formula: log10(I) ~ log10(I0) + (-3 * B + 1/2) * log10(1 + (R_Mpc/Rc)^2)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
I0  9.34165    1.25157   7.464  0.00172 **
Rc  0.21365  0.03262   6.551  0.00281 **
B   0.64645   0.03987  16.215 8.46e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.04632 on 4 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 5.263e-06
```

Os parâmetros para o ajuste da função original foram:

I0   8.3040
Rc   0.3128
B   0.8791

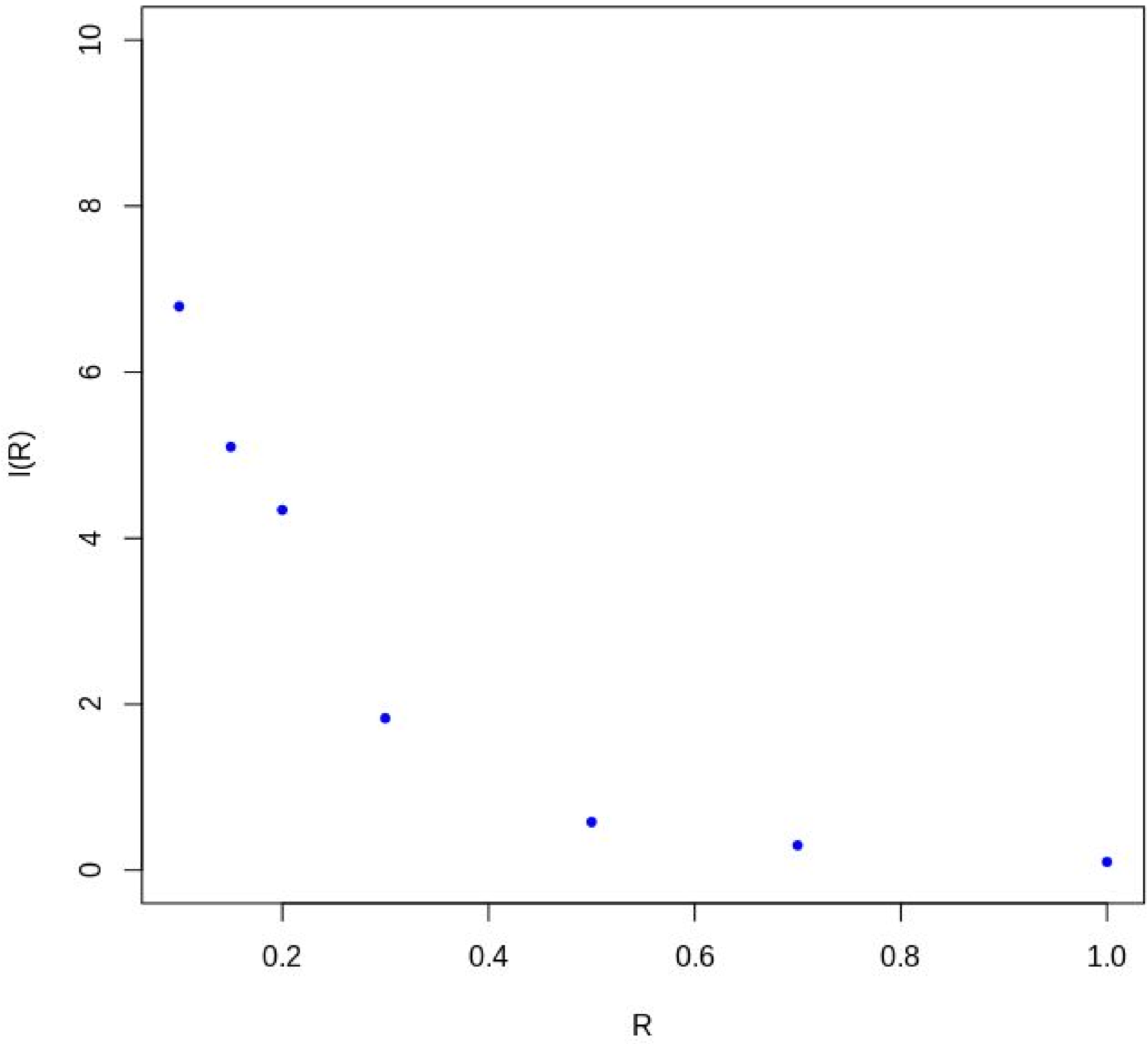
Para o ajuste com log10 foram obtidos os seguintes valores:

I0 9.34165
Rc 0.21365

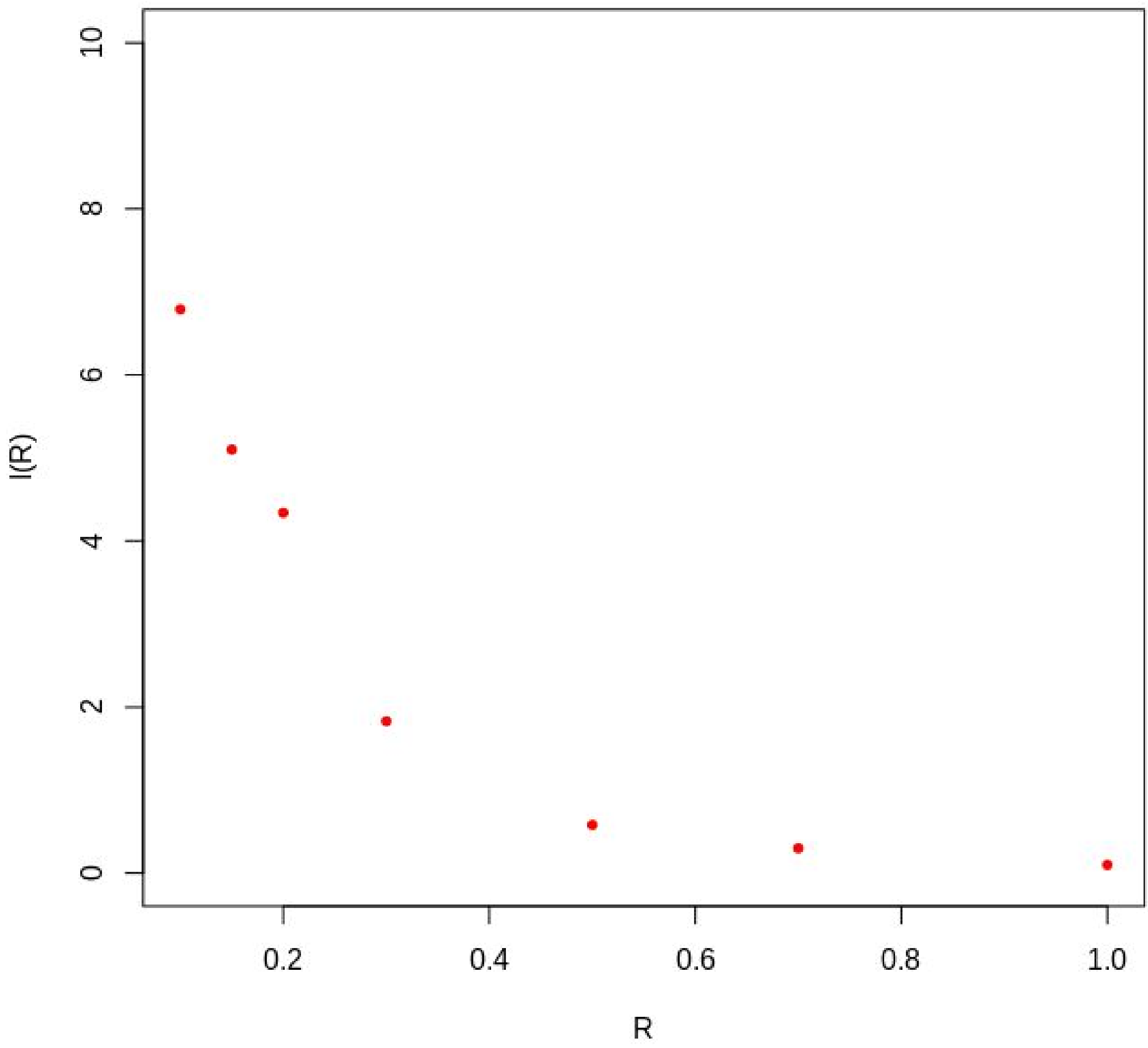
B 0.64645

Plotando um gráfico para visualização:

Função original



Função com log10



Olhando nos gráficos as diferenças são muito sutis, pequenas, mas mesmo assim pode se ver que o modelo feito com a função log10 é melhor (Não consegui traçar uma linha para as funções, mas no primeiro gráfico os pontos estão bem "perturbados" e já no segundo gráfico os pontos estão mais organizados, seguem uma certa ordem bonitinha).