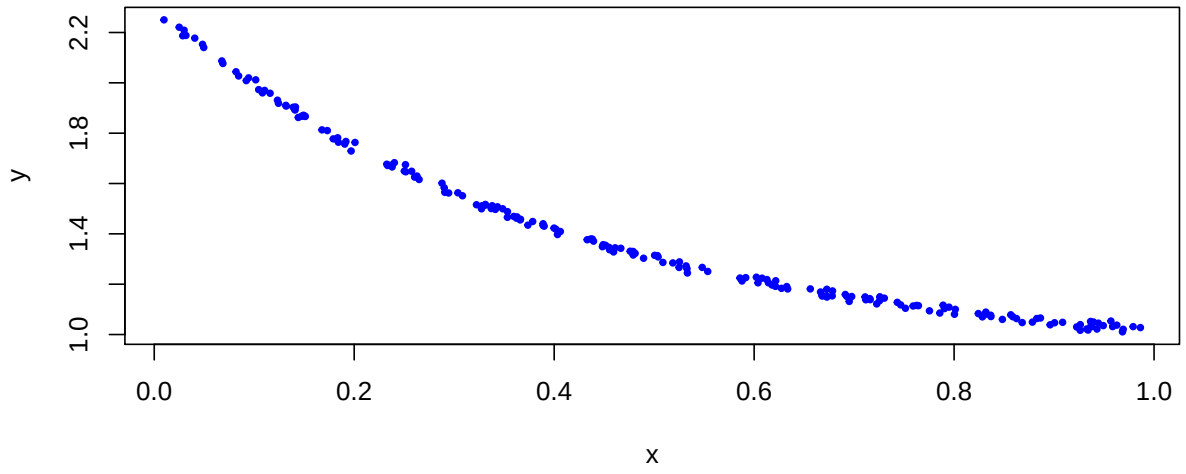
Lista 11

Matheus Cougias

20/03/2021

Gerando os Dados
Através do gráfico abaixo temos uma noção do comportamento da distribuição dos valores de Y e de X.

```
n <- 200
x <- runif(n)
y <- 0.9 + 1.4*exp(-2.5*x) + rnorm(n, sd=0.01)
dados <- data.frame(y=y, x=x)
plot(y ~ x, dados, pch=19, col="blue", cex=0.5)
```

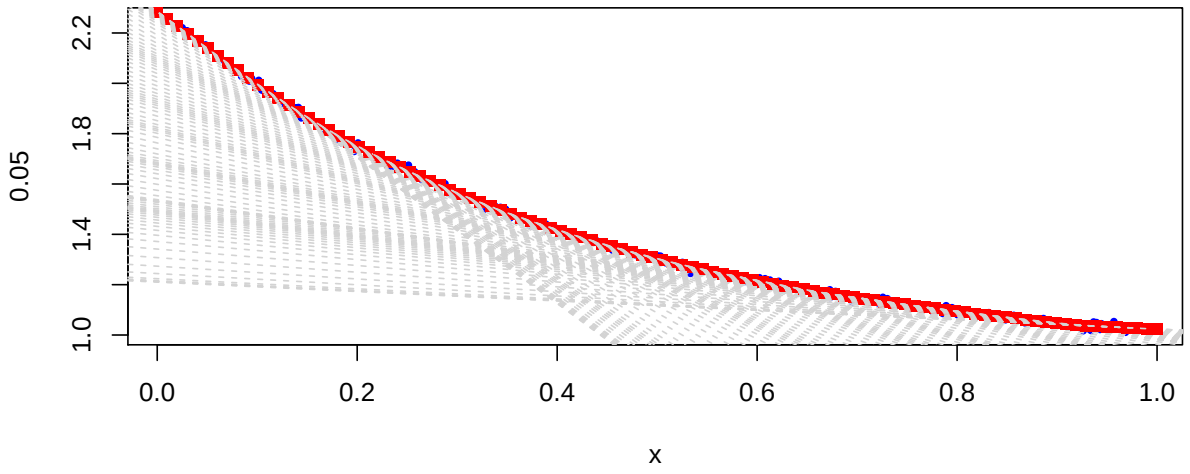


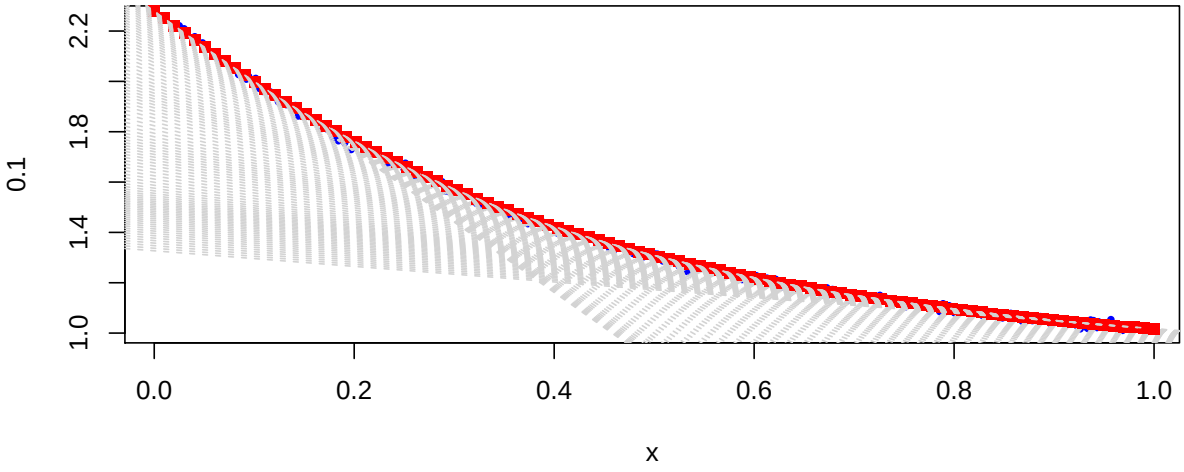
Mínimos Quadrados Ponderados
A primeira técnica de regressão utilizada é de mínimos quadrados ponderados, tendo a largura da banda como parâmetro de entrada. Tentei de algumas maneiras cacular o valor de R^2 preditivo para que uma função de otimização gerasse o lambda ótimo, porém os cálculos não saíram da maneira que esperava. Dessa maneira, sem conseguir calcular esse valor do R^2, resolvi testar alguns valores para que a observação do ajuste fosse realizada baseao nos gráficos. Nessa análise visual, percebo que os valores de lambda acima de 0.1 não conseguem se ajustar corretamente aos pontos gerados pela equação. Assim, analisando os gráficos para lambda 0.05 e 0.1, percebo um melhor ajuste para 0.05, pois a quantidade de pontos fora da curva de regressão são menores.

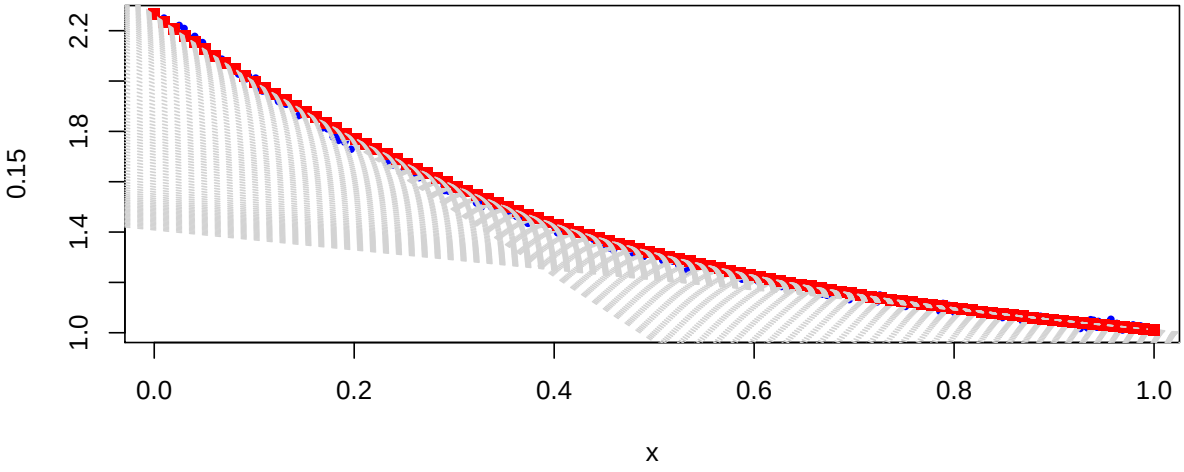
```
Ponderado <- function(lambda)
{
```

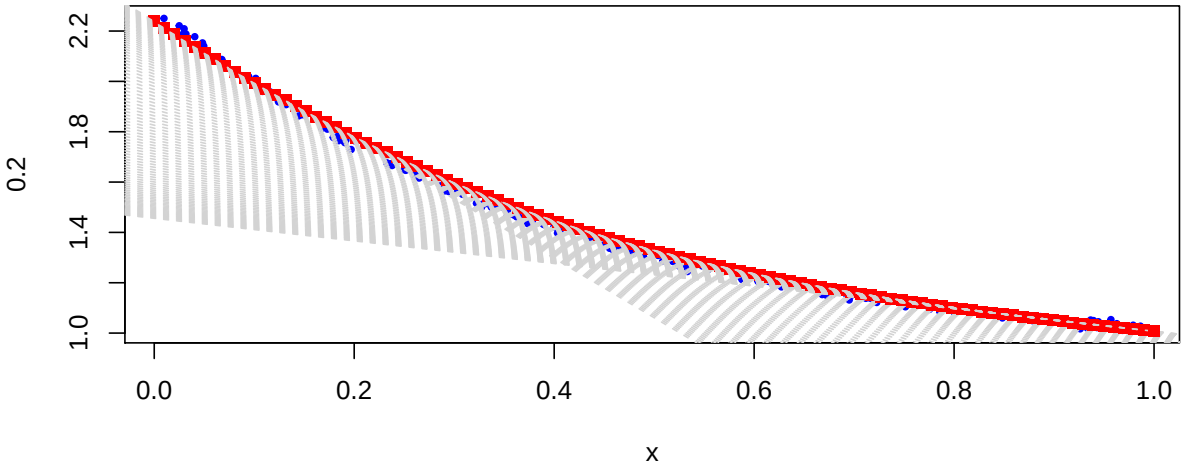
```
  plot(y ~ x, dados, pch=19, col="blue", cex=0.5, ylab = lambda)
  X <- as.matrix( cbind(1, dados$x) )
  Y <- as.matrix(dados$y)
  for(x in seq(0, 1, length=100)){
    x0    <- x
    pesos <- exp( - ( (dados$x - x0)/lambda )^2 )
    pesos  <- pesos/sum(pesos)
    W    <- diag(pesos)
    beta <- solve(t(X)%*%W%*%X)%*%t(X)%*%W%*%Y
    abline(a=beta[1], b=beta[2], lty=2, col="light grey")
    yhat <- matrix(c(1,x0), ncol=2)%*%beta
    points(yhat ~ x0, pch=15, col="red")
  }
}

for(lambda in c(0.05, 0.1, 0.15, 0.2))
{
  Ponderado(lambda)
}
```




0.1
2.2
1.8
1.4
1.0
0.0
0.2
0.4
0.6
0.8
1.0
x


0.15
2.2
1.8
1.4
1.0
0.0
0.2
0.4
0.6
0.8
1.0
x



Mínimos Quadrados Não-Lineares Aplicando os mínimos quadrados não-lineares, temos (dependerá da rodada da geração dos dados, foi baseada na anterior à montagem do PDF):
-> Beta0 = 2.1277355 e Beta1 = -0.8849776;
-> Para 2.5%, Beta0 = 2.1057309 e Beta1 = -0.9081198;
-> Para 97.5%, Beta0 = 2.1497401 e Beta1 = -0.8618353.

```
Y <- as.matrix(dados$y)
beta <- as.matrix(c(mean(Y), 0))

for(iter in 1:10)
  {
   dBeta0 <- exp(beta[2]*dados$x)
   dBeta1 <- beta[1]*exp(beta[2]*dados$x)*dados$x
   J <- as.matrix( cbind(dBeta0, dBeta1) )
   f <- beta[1]*exp(beta[2]*dados$x)
   Z <- Y - f + J%*%beta
   New.beta <- solve(t(J)%*%J)%*%t(J)%*%Z
   cbind(beta, New.beta)
   beta     <- New.beta

}
sigma2 <- sum( (Y - f)^2 )/(length(Y) - 2)
sigma <- sqrt(diag( sigma2 * solve(t(J)%*%J) ) )
(beta)
```

```
##             [,1]
## dBeta0  2.099865
## dBeta1 -0.859267
```

```
(beta - 2*sigma)
```

```
##              [,1]
## dBeta0  2.0760853
## dBeta1 -0.8843678
```

```
(beta + 2*sigma)
```

```
##              [,1]
## dBeta0  2.1236452
## dBeta1 -0.8341661
```

Pacote nls() Aplicando o pacote nls, temos (dependerá da rodada da geração dos dados, foi baseada na anterior à montagem do PDF):
-> Beta0 = 2.1277355 e Beta1 = -0.8849776;
-> Para 2.5%, Beta0 = 2.1057309 e Beta1 = -0.9081198;
-> Para 97.5%, Beta0 = 2.1497401 e Beta1 = -0.8618353.

Dessa maneira, pode-se observar que ambos os métodos geram os mesmos resultados de beta, então ambos podem ser utilizados para regressão.

```
dados  <- data.frame(y=y, x=x)
modelo <- nls( y ~ b0*exp(b1*x), data=dados, start=list(b0=mean(y), b1=0))
summary(modelo)
```

```
##
## Formula: y ~ b0 * exp(b1 * x)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## b0  2.09986    0.01189  176.61   <2e-16 ***
## b1 -0.85927    0.01255  -68.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06673 on 198 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 5.185e-06
```

```
(beta)
```

```
##              [,1]
## dBeta0  2.099865
## dBeta1 -0.859267
```

```
confint(modelo)
```

```
## Waiting for profiling to be done...
```

```
##          2.5%      97.5%
## b0  2.0761118  2.1237356
## b1 -0.8846438 -0.8339974
```