P2 - Análise Exploratória de Dados

Nome: Davi dos Santos Mattos - 119133049 ; Lucas Jesus Lapa - 117086199

Questão 1

Tendo $log(y_{ij}) = \mu + \alpha_i + \beta_j + e_{ij}$, para que e_{ij} seja um ruído branco, precisamos que $y_{ij} = e^{\mu+\alpha_i+\beta_j+e_{ij}}$, para que quando calcularmos a mediana do quociente $\frac{resíduo}{valor\_de\_comparação}$, teremos $k = 1$.

como poderemos ver ao executar o código abaixo.

Código R utilizado:

```
linhas <- 4
colunas <- 5

mu <- 5

alfa <- c(-2,-1,1,2)
beta <- c(-5,-3,0,3,5)

erro <- rnorm(20,0,0.2)
e <- matrix(erro, linhas,colunas, byrow = TRUE)

y <- matrix(0,linhas,colunas)

for (i in 1:linhas) {
   for (j in 1:colunas) {
      y[i,j] <- exp((mu + alfa[i] + beta[j] + e[i,j])) #Funçao inversa de log(x)
   }
}

yhat <- c()
delta <- matrix(0,4,5)
for(j in 1:5){
    delta[,j] <- y[,j] - median(y[,j]) # matriz delta
    yhat <- c(yhat, median(y[,j])) # yhat
}

centr <- median(yhat) #Valor central

rowef <- c()
for(i in 1:4){
    rowef <- c(rowef, median(delta[i,])) # efeito de linha
}

colef <- numeric(5)
for (j in 1:5) {
   colef[j] <- yhat[j] - centr # efeito de coluna
}

res <- matrix(0,4,5) #Resíduo
for(i in 1:4){
    res[i,] <- delta[i,] - median(delta[i,])
}

comparacao_valores <- matrix(0,4,5)
for(i in 1:4){
    for(j in 1:5){
        comparacao_valores[i,j] <- (rowef[i]*colef[j])/centr
```

```
    }
}

cv<-numeric(20)
sr<-numeric(20)
for (i in 1:4) {
   for (j in 1:5) {
      cv[(i-1)*5+j] <- comparacao_valores[i,j]
      sr[(i-1)*5+j] <- res[i,j]
   }
}

ls.fit=lm(sr~cv)
ls.fit

incl=sr/cv
incli=na.omit(incl)
median(incli)
```

Questão 2

a)

Quando aplicamos a metodologia da tabela de duas entradas nos dados sem nenhuma transformação prévia, ao calcular o coeficiente angular da reta nós obtemos que $k = 0.8018$, pelos mínimos quadrados, e $k = 0.8655438$, pela média do quociente. Ambos próximo de 1 (um), o que nos indica que a melhor transformação de variável a ser aplicada é $log(y)$ se $k = 1$.

Código R utilizado:

```
cells <- c(73.77574, 81.28325, 94.07271, 104.92151, 118.88436, 86.69515,
95.23071, 109.83399, 127.11574, 148.41316, 86.68249, 98.59934, 112.64397,
123.08970, 136.94859)

rnames <- c("Ate34","35a49","50+")
cnames <- c("1","2","3","4","5")
tabela_wage <- matrix(cells,3,5,byrow = TRUE,dimnames=list(rnames,cnames))

yhat <- c()
delta <- matrix(0,3,5)
for(j in 1:5){
    delta[,j] <- tabela_wage[,j] - median(tabela_wage[,j]) # matriz delta
    yhat <- c(yhat, median(tabela_wage[,j])) # yhat
}

centr <- median(yhat) #Valor central

rowef <- c()
for(i in 1:3){
    rowef <- c(rowef, median(delta[i,])) # efeito de linha
}

colef <- numeric(5)
for (j in 1:5) {
   colef[j] <- yhat[j] - centr # efeito de coluna
}

res <- matrix(0,3,5) #Erro
for(i in 1:3){
    res[i,] <- delta[i,] - median(delta[i,])
}
```

```
comparacao_valores <- matrix(0,3,5)
for(i in 1:3){
    for(j in 1:5){
        comparacao_valores[i,j] <- (rowef[i]*colef[j])/centr
    }
}

cv<-numeric(length(rnames)*length(cnames))
sr<-numeric(length(rnames)*length(cnames))
for (i in 1:length(rnames)) {
   for (j in 1:length(cnames)) {
      cv[(i-1)*5+j] <- comparacao_valores[i,j]
      sr[(i-1)*5+j] <- res[i,j]
   }
}

ls.fit=lm(sr~cv)
ls.fit

incl=sr/cv
incli=na.omit(incl)
median(incli)
```

b)

Após aplicar a transformação de variável, nós temos:

Valor Central	4.69897

Efeito de Linha	-0.1583625055	0.0001460396	0		
Efeito de Coluna	-0.2367181	0.1426676	0	0.1139433	0.2206356

Resíduos:

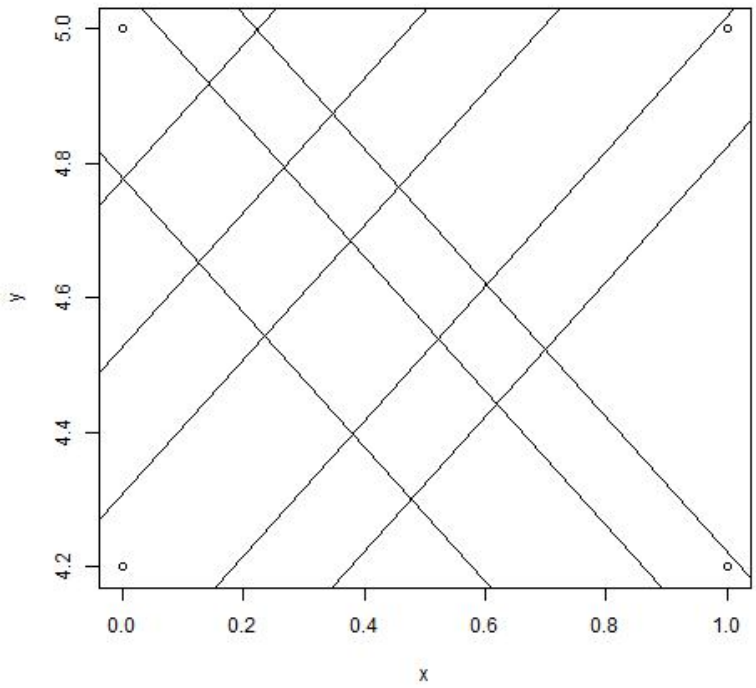
-0.002859446	0	0.0034604554	-0.001338306	0.01690816
0	-0.0001460396	-0.0001460396	0.03203861	0.08024837
0	0.03476209	0.02526209	0	0

Código R utilizado:

```
cells <- c(73.77574, 81.28325, 94.07271, 104.92151, 118.88436, 86.69515,
95.23071, 109.83399, 127.11574, 148.41316, 86.68249, 98.59934, 112.64397,
123.08970, 136.94859)
novos_valores <- c()
for(i in 1:length(cells)){
    novos_valores <- c(novos_valores, log(cells[i]))
}
rnames <- c("Ate34","35a49","50+")
cnames <- c("1","2","3","4","5")
tabela_wage <- matrix(novos_valores,3,5,byrow = TRUE,dimnames=list(rnames,cnames))
yhat <- c()
delta <- matrix(0,3,5)
for(j in 1:5){
    delta[,j] <- tabela_wage[,j] - median(tabela_wage[,j]) # matriz delta
    yhat <- c(yhat, median(tabela_wage[,j])) # yhat
}

centr <- median(yhat) #Valor central

rowef <- c()
```

```
for(i in 1:3){
    rowef <- c(rowef, median(delta[i,])) # efeito de linha
}

colef <- numeric(5)
for (j in 1:5) {
   colef[j] <- yhat[j] - centr # efeito de coluna
}

res <- matrix(0,3,5) #Erro
for(i in 1:3){
    res[i,] <- delta[i,] - median(delta[i,])
}
```

c)



As retas do gráfico acima com inclinação igual a 1, representam o nível de escolaridade, enquanto a retas com inclinação igual a -1, representam a faixa etária (Até 34, 50+, 35 a 49). Através desse gráfico podemos concluir que o salário aumenta conforme o nível de escolaridade é maior, e que a tendência é de que na faixa etária 50+, o salário estabilize ou diminua.

Código R utilizado:

```
x <- c(-1,-1,1,1)
y <- c(0,0.5,0.5,1)

jpeg("grafico_wage_age.jpeg")
plot(x,y)

abline(4.778151,1)
```

```
abline(4.528275,1)
abline(4.309985,1)
abline(4.017729,1)
abline(3.823909,1)

abline(4.778151,-1)
abline(5.221849,-1)
abline(5.061061,-1)
dev.off()
```

Questão 3

a)

Para esse problema foi utilizado a combinação $span = 0.3, degree = 2, family = "gaussian"$, pois apresentava o menor erro comparado com as outras 11 combinações, como podemos ver abaixo.

```
> menor_erro
   span  degree family      error
1  "0.1" "1"    "gaussian"  "139377.853682376"
2  "0.1" "2"    "gaussian"  "252745.609423955"
3  "0.3" "1"    "gaussian"  "632536.054017686"
4  "0.3" "2"    "gaussian"  "75297.345413104"
5  "0.5" "1"    "gaussian"  "3376276.93339049"
6  "0.5" "2"    "gaussian"  "117741.792061529"
7  "0.1" "1"    "symmetric" "200778.808619933"
8  "0.1" "2"    "symmetric" "298512.939047683"
9  "0.3" "1"    "symmetric" "942987.916058014"
10 "0.3" "2"    "symmetric" "116196.192356177"
11 "0.5" "1"    "symmetric" "4366402.40930338"
12 "0.5" "2"    "symmetric" "167364.996291138"
> min(menor_erro[,4]) # Vendo o menor erro
[1] "116196.192356177"
```

Esse resultado não era previsível, pois o erro é gerado de uma forma aleatória e com o desvio padrão alto, o que acarreta que se gerarmos novos valores para o erro, a melhor combinação pode ser outra, como podemos ver abaixo.

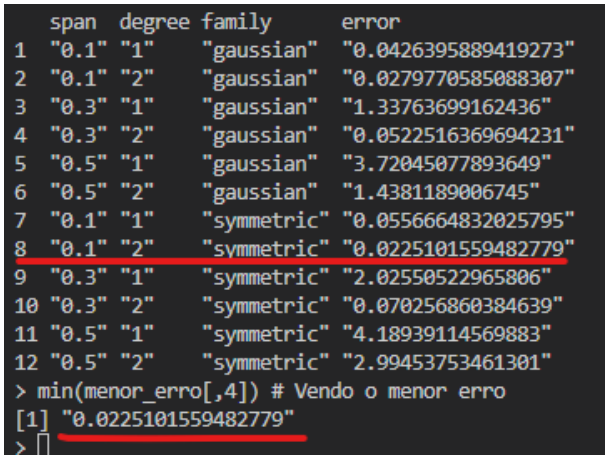
```
> menor_erro
   span  degree family      error
1  "0.1" "1"    "gaussian"  "107882.455806272"
2  "0.1" "2"    "gaussian"  "222969.548072712"
3  "0.3" "1"    "gaussian"  "510186.873061198"
4  "0.3" "2"    "gaussian"  "69436.0174410472"
5  "0.5" "1"    "gaussian"  "3398019.7479664"
6  "0.5" "2"    "gaussian"  "56515.414837298"
7  "0.1" "1"    "symmetric" "108367.33034484"
8  "0.1" "2"    "symmetric" "243045.278996315"
9  "0.3" "1"    "symmetric" "682207.540066887"
10 "0.3" "2"    "symmetric" "70446.4473159773"
11 "0.5" "1"    "symmetric" "4494025.76349425"
12 "0.5" "2"    "symmetric" "59664.7823735953"
> min(menor_erro[,4]) # Vendo o menor erro
[1] "107882.455806272"
```

Código R utilizado

```
n <- 1000
estr <- numeric(n)
e <- rnorm(n, 0, 300)
x <- numeric(n)
y <- numeric(n)

for (i in 1:n) {
    x[i] <- i/10
    estr[i] <- 500 + (-30 * x[i]) + (2 * (x[i])^2) + (-0.02 * (x[i])^3)
    y[i] <- estr[i] + e[i]
}

# Criando Matriz para comparação

ncol <- c("span", "degree", "family", "error")
nlin <- seq(1, 12)
menor_erro <- matrix(0, nrow = 12, ncol = 4, byrow = TRUE, dimnames = list(nlin,ncol))
span1 <- c(0.1,0.3,0.5)
grau <- c(1,2)
familia <- c("gaussian", "symmetric")
verd <- numeric(100)
linha <- 1

for (i in 1:length(familia)){
   for (k in 1:length(span1)) {
      for (j in 1:length(grau)){
        fit1 <- loess(y~x,span=span1[k],degree=grau[j],family=familia[i])
        prec <- 0
        for (m in 1:100) {
            verd[m] <- 500 + (-30 * m) + (2 * (m^2)) + (-0.02 * (m^3))
            prec <- prec + ((predict(fit1,m) - verd[m])^2)
        }
        menor_erro[linha,1] <- span1[k]
        menor_erro[linha,2] <- grau[j]
        menor_erro[linha,3] <- familia[i]
        menor_erro[linha,4] <- prec
        linha <- linha + 1
      }
   }
}
menor_erro
min(menor_erro[,4]) # Vendo o menor erro
```

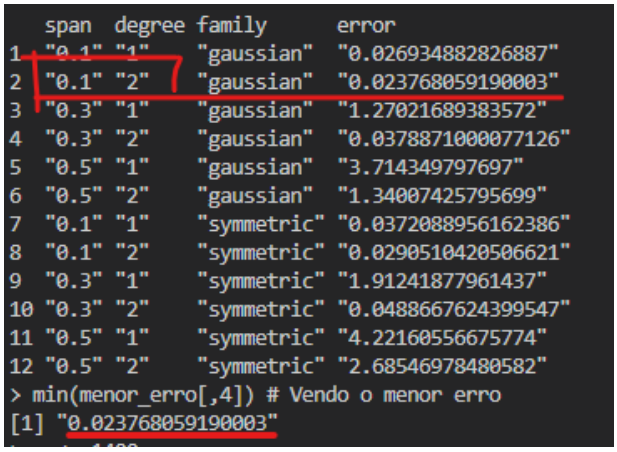
b)

Dentre as 12 combinações testadas a escolhida foi a $span = 0.1, degree = 2, family = symmetric$, pois quando comparada com as outras combinações ela apresenta o menor erro.



```
   span  degree family      error
1  "0.1" "1"    "gaussian"  "0.0426395889419273"
2  "0.1" "2"    "gaussian"  "0.0279770585088307"
3  "0.3" "1"    "gaussian"  "1.33763699162436"
4  "0.3" "2"    "gaussian"  "0.0522516369694231"
5  "0.5" "1"    "gaussian"  "3.72045077893649"
6  "0.5" "2"    "gaussian"  "1.4381189006745"
7  "0.1" "1"    "symmetric" "0.0556664832025795"
8  "0.1" "2"    "symmetric" "0.0225101559482779"
9  "0.3" "1"    "symmetric" "2.02550522965806"
10 "0.3" "2"    "symmetric" "0.070256860384639"
11 "0.5" "1"    "symmetric" "4.18939114569883"
12 "0.5" "2"    "symmetric" "2.99453753461301"
> min(menor_erro[,4]) # Vendo o menor erro
[1] "0.0225101559482779"
>
```

Quando geramos novos valores para e, podemos ver que dois parâmetro (span e degree) se repetem quando calculamos a menor distância entre as curvas. Isso acontece devido ao desvio padrão do erro ser mínimo(0,1), podendo assim se tornar o resultado previsível, como podemos ver abaixo.



```
   span  degree family      error
1  "0.1" "1"    "gaussian"  "0.026934882826887"
2  "0.1" "2"    "gaussian"  "0.023768059190003"
3  "0.3" "1"    "gaussian"  "1.27021689383572"
4  "0.3" "2"    "gaussian"  "0.0378871000077126"
5  "0.5" "1"    "gaussian"  "3.714349797697"
6  "0.5" "2"    "gaussian"  "1.34007425795699"
7  "0.1" "1"    "symmetric" "0.0372088956162386"
8  "0.1" "2"    "symmetric" "0.0290510420506621"
9  "0.3" "1"    "symmetric" "1.91241877961437"
10 "0.3" "2"    "symmetric" "0.0488667624399547"
11 "0.5" "1"    "symmetric" "4.22160556675774"
12 "0.5" "2"    "symmetric" "2.68546978480582"
> min(menor_erro[,4]) # Vendo o menor erro
[1] "0.023768059190003"
```

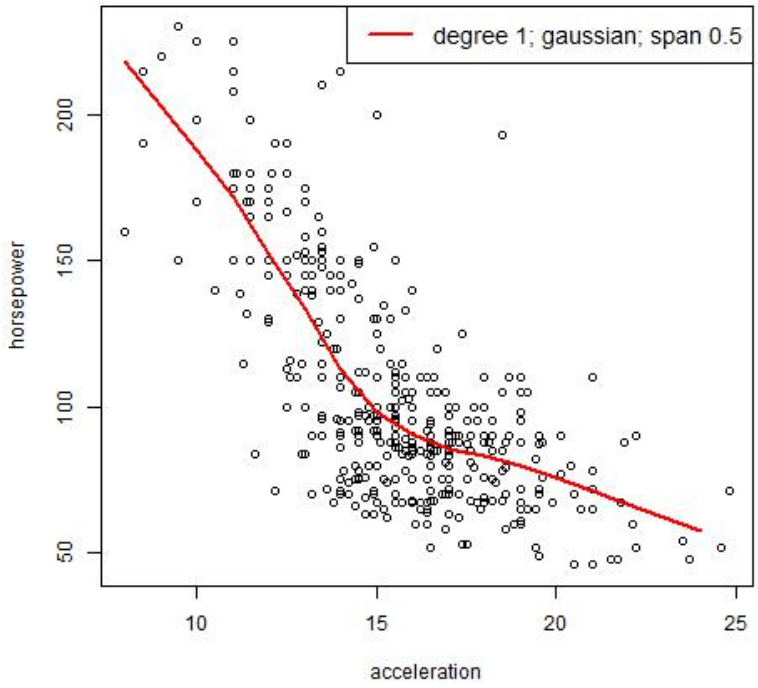
Código R utilizado:

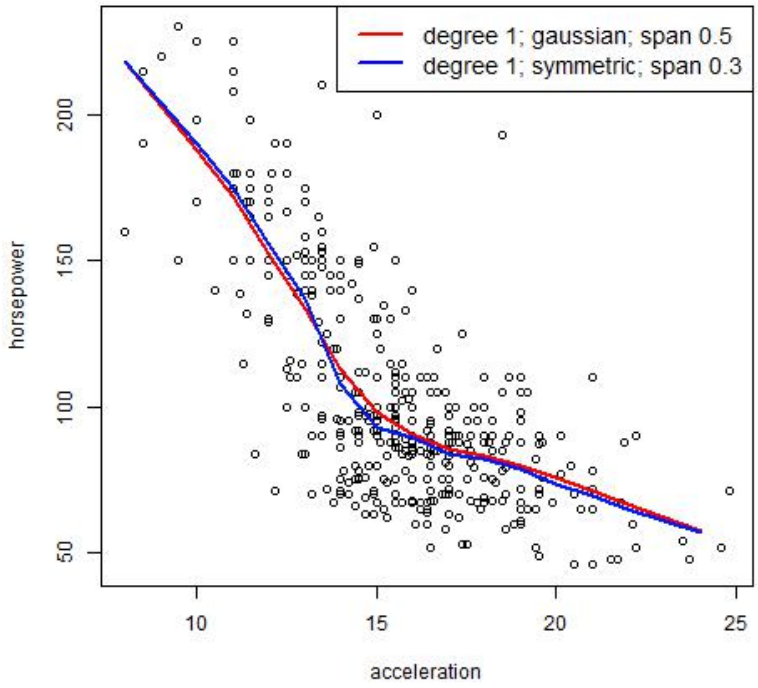
```
n <- 1400
estr <- numeric(n)
e <- rnorm(n, 0, 0.1)
x <- numeric(n)
y <- numeric(n)

for (i in 1:n) {
    x[i] <- i/100
    estr[i] <- exp(-0.2 * x[i]) * sin(x[i])
    y[i] <- estr[i] + e[i]
}
# Criando Matriz para comparação
ncol <- c("span", "degree", "family", "error")
nlin <- seq(1, 12)
menor_erro <- matrix(0, nrow = 12, ncol = 4, byrow = TRUE, dimnames = list(nlin,ncol))
span1 <- c(0.1,0.3,0.5)
grau <- c(1,2)
familia <- c("gaussian", "symmetric")
verd <- numeric(100)
linha <- 1
```

```
for (i in 1:2){
   for (k in 1:3) {
      for (j in 1:2){
        fit1 <- loess(y~x,span=span1[k],degree=grau[j],family=familia[i])
        prec <- 0
        for (m in 1:140) {
            verd[m] <- exp(-0.2*(m/10)) * sin(m/10)
            prec <- prec + ((predict(fit1,(m/10)) - verd[m])^2)
        }
        menor_erro[linha,1] <- span1[k]
        menor_erro[linha,2] <- grau[j]
        menor_erro[linha,3] <- familia[i]
        menor_erro[linha,4] <- prec
        linha <- linha + 1
      }
   }
}
menor_erro
min(menor_erro[,4]) # Vendo o menor erro
```

Questão 4

Dentre as 12 combinações dos parâmetros, a combinação escolhida foi a $span = 0.5, degree = 1, family = gaussian$. Pois como podemos visualizar no gráfico abaixo, a curva não apresenta praticamente nenhuma oscilação ao longo da mesma, diferentemente das outras combinações que apresentavam diversas oscilações.





Código R utilizado:

```
library(ISLR)
attach(Auto)
names(Auto)

plot(acceleration, horsepower)

powlims <- range(acceleration)
pow.grid <-seq(from=powlims[1],to=powlims[2])

plot(acceleration,horsepower,xlim=powlims,cex=.5)
fit1 <- loess(horsepower~acceleration,span=.1,degree=1,family="symmetric")
lines(pow.grid,predict(fit1,data.frame(acceleration=pow.grid)),col="red",lwd=2)
fit2 <- loess(horsepower~acceleration,span=.1,degree=1,family="gaussian")
lines(pow.grid,predict(fit2,data.frame(acceleration=pow.grid)),col="#fffb00",lwd=2)

plot(acceleration,horsepower,xlim=powlims,cex=.5)
fit3 <- loess(horsepower~acceleration,span=.1,degree=2,family="symmetric")
lines(pow.grid,predict(fit3,data.frame(acceleration=pow.grid)),col="#09ff00",lwd=2)
fit4 <- loess(horsepower~acceleration,span=.1,degree=2,family="gaussian")
lines(pow.grid,predict(fit4,data.frame(acceleration=pow.grid)),col="#00ffff",lwd=2)

plot(acceleration,horsepower,xlim=powlims,cex=.5)
fit5 <- loess(horsepower~acceleration,span=.3,degree=1,family="symmetric")
lines(pow.grid,predict(fit5,data.frame(acceleration=pow.grid)),col="#001aff",lwd=2)
fit6 <- loess(horsepower~acceleration,span=.3,degree=1,family="gaussian")
lines(pow.grid,predict(fit6,data.frame(acceleration=pow.grid)),col="#ff00ff",lwd=2)

plot(acceleration,horsepower,xlim=powlims,cex=.5)
fit7 <- loess(horsepower~acceleration,span=.3,degvree=2,family="symmetric")
lines(pow.grid,predict(fit7,data.frame(acceleration=pow.grid)),col="#00ffb3",lwd=2)
fit8 <- loess(horsepower~acceleration,span=.3,degree=2,family="gaussian")
lines(pow.grid,predict(fit8,data.frame(acceleration=pow.grid)),col="#eeff00",lwd=2)

plot(acceleration,horsepower,xlim=powlims,cex=.5)
```

```
fit9 <- loess(horsepower~acceleration,span=.5,degree=1,family="symmetric")
lines(pow.grid,predict(fit9,data.frame(acceleration=pow.grid)),col="#003cff",lwd=2)
#_____________________________________________________________________________________
# Visualmente a mais suave
fit10 <- loess(horsepower~acceleration,span=.5,degree=1,family="gaussian")
lines(pow.grid,predict(fit10,data.frame(acceleration=pow.grid)),col="#ff00d4",lwd=2)
#_____________________________________________________________________________________

plot(acceleration,horsepower,xlim=powlims,cex=.5)
fit11 <- loess(horsepower~acceleration,span=.5,degree=2,family="symmetric")
lines(pow.grid,predict(fit11,data.frame(acceleration=pow.grid)),col="#ff00aa",lwd=2)
fit12 <- loess(horsepower~acceleration,span=.5,degree=2,family="gaussian")
lines(pow.grid,predict(fit12,data.frame(acceleration=pow.grid)),col="#ff0000",lwd=2)
```