- Tutorial
- Exercícios
- Apostila

7a. Regressão Linear Simples

Ajuste e diagnóstico de uma regressão

Vamos usar o objeto `Animals`, do pacote *MASS*

```
library(MASS)
data(Animals)
str(Animals)
```

Nele estão as massas corporais e do cérebro de 28 espécies de vertebrados, em gramas. Vamos explorar a relação entre tamanho do cérebro e do corpo em um gráfico de dispersão:

```
plot(brain~body, data=Animals)
```

A relação claramente não é linear, então podemos tentar em escala *log*:

```
plot(brain~body,data=Animals, log="xy")
```

Ou transformar os dados em seus logaritmos, dentro da função `plot`:

```
plot(log(brain)~log(body), data=Animals)
```

Agora a relação parece linear, mas há 3 pontos bem discrepantes, de animais muito grandes mas com cérebros pequenos. Quem são?

```
Animals[log(Animals$body) > 8 & log(Animals$brain) < 6, ]
```

São os dinossauros! Ainda assim vamos ajustar uma regressão linear com todos os dados, transformados para seu logaritmos. Note que o argumento que damos para a função `lm` é o mesmo que demos para a função `plot`:

```
anim.m1 <- lm(log(brain)~log(body),data=Animals)
```

E vamos adicionar a linha de regressão ao gráfico:

```
abline(anim.m1, col="blue")
```

Antes de prosserguir, vamos inspecionar os gráficos de diagnóstico (um a um, aperte `Enter` para passar para o próximo):

```
plot(anim.m1)
```

Há vários problemas. Você os identifica?

Vamos agora ajustar uma regressão sem os dinossauros, retirados com o argumento `subset`:

```
anim.m2 <- lm(log(brain)~log(body),data=Animals,
subset=!(log(Animals$body)>8&log(Animals$brain)<6))
```

Um gráfico com as retas dos dois modelos, para comparação; veja como a exclusão destes três pontos mudou bastante a inclinação da reta:

```
plot(log(brain)~log(body), data=Animals)
abline(anim.m1, col="blue", lty=2)
abline(anim.m2, col="red")
```

Novos Diagnósticos (4 gráficos de uma vez) mostram que agora ainda temos alguns problemas (quais?):

```
par(mfrow=c(2,2))
plot(anim.m2)
par(mfrow=c(1,1))
```

Mas vamos prosseguir. Fazemos a ANOVA e o resumo do modelo:

```
anova(anim.m2)
summary(anim.m2)
```

Calculamos os coeficientes estimados e os intervalos de confiança:

```
coef(anim.m2)
confint(anim.m2)
```

Como a relação entre os logaritmos dos valores é linear, na escala original a relação é de uma equação de potência (*power law*), muito usada para expressar relações alométricas:

$$y \ = \ Cx^b $$

Onde $b =$ inclinação e $C = e^a$, sendo a o intercepto na escala log-log, se usamos logaritmos naturais.

Agora vamos acrescentar à planilha os valores previstos, já na escala original:

```
cf <- as.numeric(coef(anim.m2))
Animals$prev <- round(exp(cf[1])*Animals$body^cf[2],1)
Animals
```

Para quais animais a estimativa foi melhor? E para quais foi pior? Você consegue relacionar isto com os diagnósticos que fizemos acima?

Simulação com dados Não Normais

Neste tutorial vamos investigar a robustez do modelo de regressão linear simples (Gaussiana) à violação de seus pressupostos. Para isto, vamos criar uma variável resposta cujo valor médio é uma função linear de uma variável preditora, mas com uma distribuição muito diferente da Normal. Antes de mais nada, defina uma semente de números aleatórios, para reproduzir a mesma simulação, e verificar os resultados relatados abaixo.

```
set.seed(42)
```

Vamos então simular 30 valores da variável preditora, sorteados de uma distribuição uniforme:

```
x <- runif(30,0,10)
```

Em seguida criamos uma variável resposta, vinda de uma população com distribuição exponencial, cuja a média equivale a (2 + 5 * preditora). A distribuição exponencial tem apenas um parâmetro, λ, que corresponde ao inverso da média.

```
y <- rexp(30,rate=1/(2+5*x))
```

Inspecionando o gráfico de dispersão:

```
plot(y~x)
```

Ajustando a regressão linear:

```
m1 <- lm(y~x)
```

Os gráficos de diagnóstico mostram desvios importantes em relação às premissas do modelo de regressão, como esperado: os resíduos não são normais e não têm variância constante. Há alguns pontos com alavancagem (*leverage*) forte.

```
par(mfrow=c(2,2))
plot(m1)
par(mfrow=c(1,1))
```

Quais os coeficientes estimados pela regressão?

```
coef(m1)
```

São bem diferentes dos valores reais de intercepto (a = 2) e inclinação (b = 5). Será que este padrão se mantém? Podemos investigar isto repetindo várias vezes a tomada de uma amostra de y e o ajuste do modelo, e para cada uma delas guardar os valores dos coeficientes. Vamos fazer isso, e também guardar os limites dos intervalos de confiança a 5% dos coeficientes, para verificar quantas vezes eles de fato incluem os valores verdadeiros.

Primeiro criamos uma matriz para guardar os resultados. Cada linha guarda o resultado de uma aleatorização, e deixamos seis colunas, para as estimativas do intercepto e inclinação, e para os limites de seus intervalos de confiança:

```
result <- matrix(ncol=6,nrow=2000)
```

Em seguida usamos um *loop* (função `for`) para repetir as simulações 2000 vezes. A cada vez guardamos os coeficientes e limites de seus intervalos de confiança:

```
for(i in 1:2000){
  yr <- rexp(30,rate=1/(2+5*x))
  m <- lm(yr~x)
  CIs <- confint(m)
  result[i,1:2] <- as.numeric(coef(m))# guarda os coeficientes nas duas
primeiras colunas
  result[i,3:4] <- CIs[1,]#guarda limites dos ICs do intercepto
  result[i,5:6] <- CIs[2,]#guarda limites dos ICs da inclinacao
}
```

Agora vamos verificar a distribuição dos valores dos interceptos estimados, que estão guardadas na primeira coluna da matriz de resultados:

```
hist(result[,1])
```

Acrescentamos a média destes valores:

```
abline(v=mean(result[,1]), lty=2, col="blue")
```

E o valor real de a = 2

```
abline(v=2, col="red", lty=2)
```

A diferença é minúscula, o que indica que o valor esperado das estimativas (i.e. sua média ou esperança estatística), é muito próximo do valor do parâmetro.

Agora vamos verificar o mesmo para as inclinações estimadas, que guardamos na segunda coluna da matriz de resultados:

```
hist(result[,2])
abline(v=mean(result[,2]), lty=2, col="blue")
abline(v=5, col="red", lty=2)
```

Nos dois casos o viés das estimativas parece ser muito pequeno. Vamos calcular este viés em percentual em relação ao valor real para o intercepto:

```
100*(2-mean(result[,1]))/2 ## 7.8%
```

e para a inclinação:

```
100*(5-mean(result[,2]))/5 ## -0.3%
```

Agora vamos verificar as estimativas pelo intervalo de confiança. Quantas vezes os intervalos estimados incluíram o valor real? Como o intervalo foi calculado a 5%, esperaríamos que dos 2000

intervalos calculados, 1900 incluíssem o valor real.

Vamos fazer esta contagem. Como guardamos os limites do intervalo de confiança do intercepto nas colunas 3 e 4 da matriz de resultados, podemos fazer a contagem como a soma de um teste lógico:

```
sum(result[,3]<=2&result[,4]>=2)
```

Em 1995 das 2000 simulações o intervalo incluiu a média, ou seja, a significância do intervalo é de 5/2000 = 0,0025, o que é menor do que a significância nominal de 0,05.

Para a inclinação temos:

```
sum(result[,5]<=5&result[,6]>=5)
```

Aqui as simulações mostram que o intervalo de confiança teve uma significância um pouco maior que a nominal (0,055).

Em resumo, surpreendentemente, as estimativas dos coeficientes da regressão por ponto (valores estimados) e por intervalo (intervalos de confiança) mostraram-se muito robustas a este caso de violação das premissas. No entanto, **não generalize este resultado**, pois não sabemos como variações desta simulação se comportariam.

Ajuste de Polinômios

Galileu Galilei investigou a distância percorrida por um corpo lançado de uma rampa a diferentes alturas. Para isto ele usou um plano inclinado a uma certa altura do chão, de onde soltou bolas de metal. Com este experimento ele verificou que a relação entre a altura de lançamento e a distância percorrida não é linear. Mais detalhes na ajuda do objeto galileo, no pacote UsingR. Este tutorial é retirado do livro companheiro deste pacote[1)].

As alturas de lançamentos e as distâncias percorridas obtidas por Galileu são[2)]:

```
init.h = c(600, 700, 800, 950, 1100, 1300, 1500)
h.d = c(253, 337, 395, 451, 495, 534, 573)
```

Vamos fazer o gráfico de dispersão:

```
plot(h.d~init.h)
```

Galileu tinha razões teóricas para supor que a relação seria uma parábola, e não uma reta, como propunha o conhecimento vigente. Hoje podemos tratar isto como um problema simples de seleção de modelos, em que os modelos concorrentes são uma equação da reta:

$$y \ = \ a+bx$$

e uma equação da parábola, que é um polinômio de segundo grau:

$$y \ = \ a+bx+cx^2$$

O primeiro modelo é um caso particular do segundo, no qual o coeficiente $$c$$ é zero. Portanto,

podemos ajustar os dois modelos e usar o comando `anova` para testar se o acréscimo do termo quadrático implicou em melhora. O ajuste da reta é feito do modo usual:

```
mod1 <- lm(h.d~init.h)
```

Mas para ajustar a parábola corretamente usamos a função `I()` (de *insulate*), necessária para isolar a operação matemática de elevar ao quadrado. Sem isto, o R entenderá o termo `init.h^2` como uma interação, detalhes aqui, na seção “Uma Palavra sobre o Argumento Fórmula”.

O comando para ajustar a equação da parábola é:

```
mod2 <- update(mod1,.~. +I(init.h^2))
```

E agora podemos comparar os dois modelos:

```
anova(mod1,mod2)
```

O modelo de parábola é claramente superior, o que fica evidente se acrescentamos as linhas dos previstos pelos dois modelos ao gráfico:

```
abline(mod1)
cf.m2 <- coef(mod2)
curve(cf.m2[1]+cf.m2[2]*x+cf.m2[3]*x^2, add=T, lty=2)
```

Por fim, verifique o resumo do modelo selecionado:

```
summary(mod2)
```

Este procedimento pode ser usado sempre que se deseje avaliar se seus dados seguem uma equação de reta. Se você suspeita de curvatura, a comparação com o modelo quadrático é um teste simples e rápido.

1)
John Verzani. Using R for Introductory Statistics. Chapman & Hall/CRC, Boca Raton, FL, 2005.

2)
as duas medidas estão em *puntos*, uma unidade que corresponde a 169/180 mm