ME720 - Modelos Lineares Generalizados

Laboratório - Dados Binários

Profa. **Larissa Avila Matos**

Frogs Data

Descrição dos dados:

```
library(DAAG)
data(frogs)
help(frogs)
```

O conjunto de dados possui 212 linhas e 11 colunas.

Esses dados são sobre a distribuição dos sapos da espécie *Southern Corroboree*, na área de Snowy Mountains, em New South Wales, Austrália.

Conjunto de Dados:

`pres.abs`: 0 = sapos ausentes na região, 1 = sapos presentes na região

`northing`: ponto de referência do norte

`easting`: ponto de referência do leste

`altitude`: altitude, em metros

`distance`: distância em metros à população existente mais próxima

`NoOfPool`: número de potenciais criadouros

`NoOfSites`: número de potenciais criadouros num raio de 2 km

`avrain`: precipitação média da chuva para o período da primavera

`meanmin`: temperatura mínima média da primavera

`meanmax`: temperatura máxima média da primavera

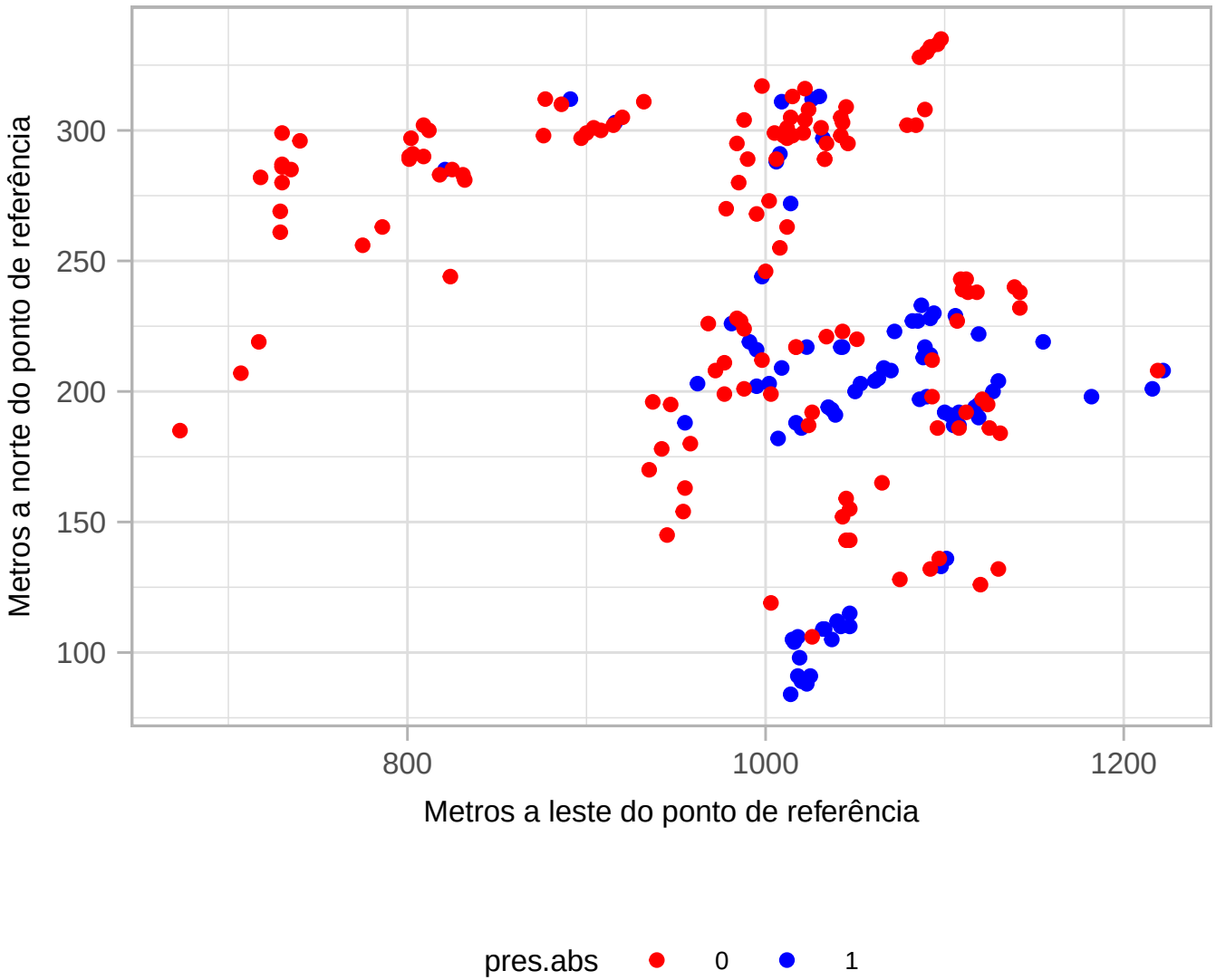
```r
dim(frogs)
head(frogs)
```

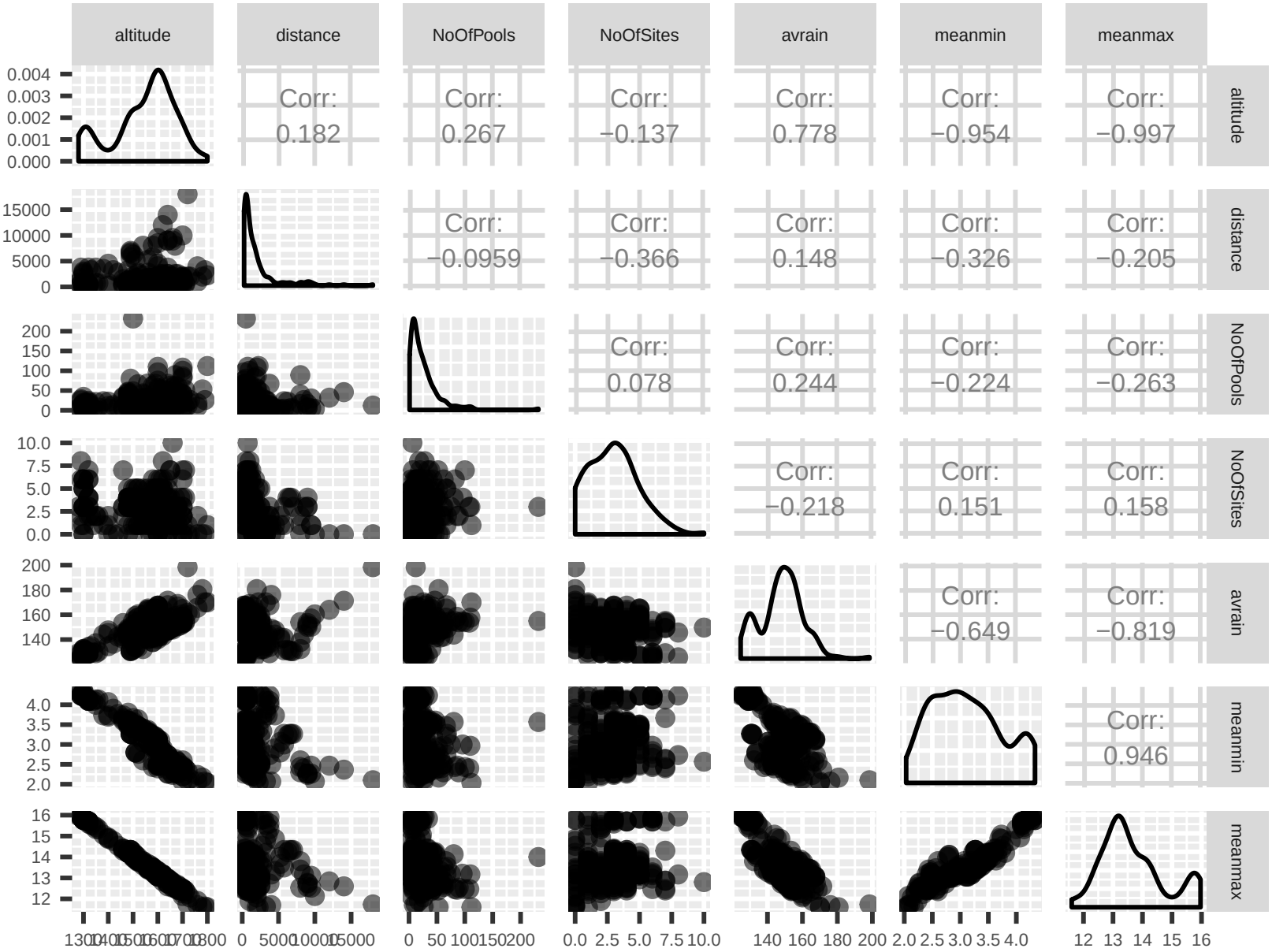
```
[1] 212  10
```

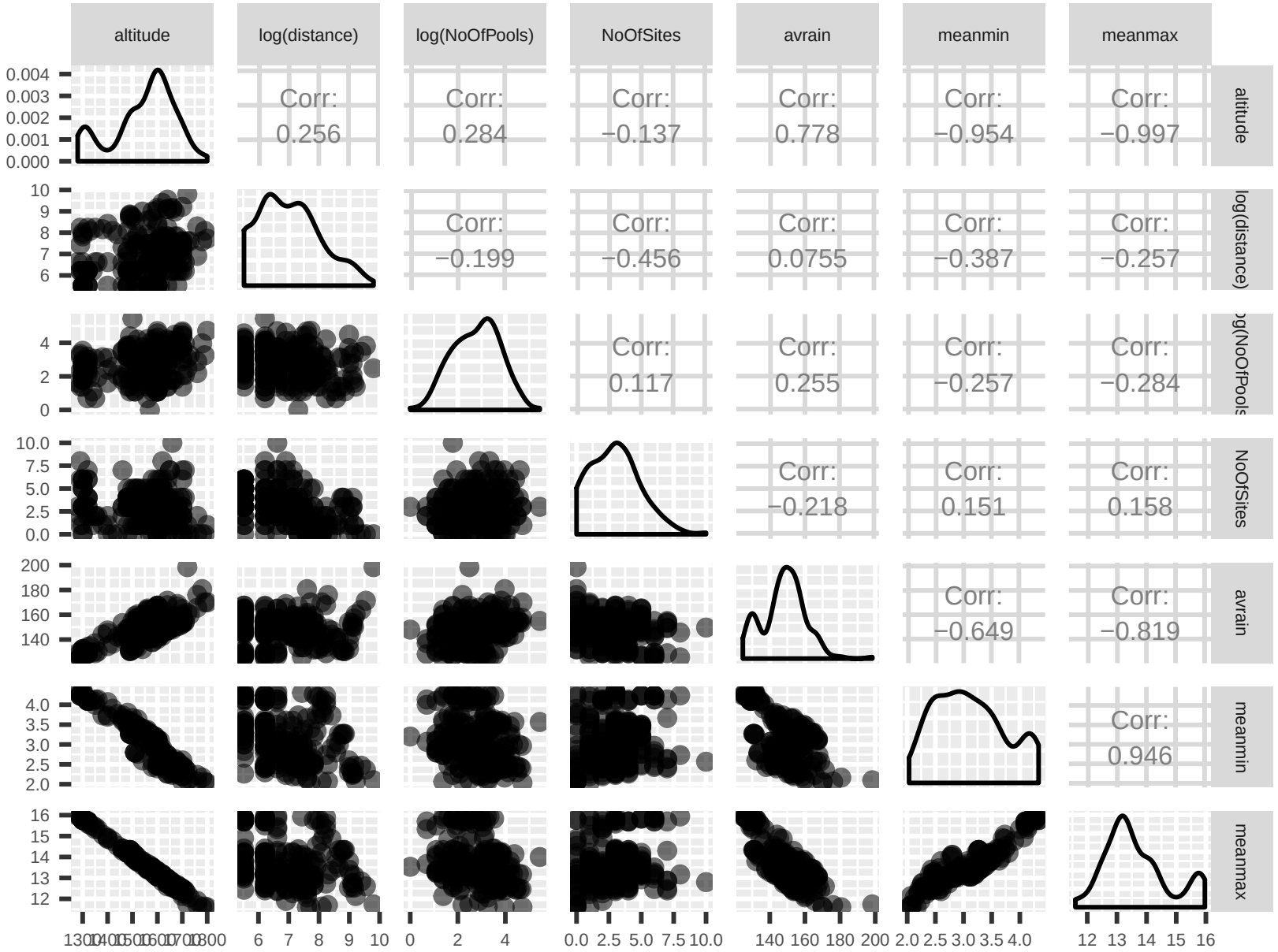
```
  pres.abs northing easting altitude distance NoOfPools NoOfSites   avrain
2        1      115    1047     1500      500       232          3 155.0000
3        1      110    1042     1520      250        66          5 157.6667
4        1      112    1040     1540      250        32          5 159.6667
5        1      109    1033     1590      250         9          5 165.0000
6        1      109    1032     1590      250        67          5 165.0000
7        1      106    1018     1600      500        12          4 167.3333
   meanmin  meanmax
2 3.566667 14.00000
3 3.466667 13.80000
4 3.400000 13.60000
5 3.200000 13.16667
6 3.200000 13.16667
7 3.133333 13.06667
```

```
summary(frogs)
```

```
   pres.abs          northing        easting          altitude
Min.   :0.0000   Min.   : 84.0   Min.   : 673.0   Min.   :1280
1st Qu.:0.0000   1st Qu.:192.0   1st Qu.: 977.8   1st Qu.:1480
Median :0.0000   Median :222.5   Median :1023.0   Median :1580
Mean   :0.3726   Mean   :228.2   Mean   :1004.6   Mean   :1547
3rd Qu.:1.0000   3rd Qu.:290.0   3rd Qu.:1086.2   3rd Qu.:1625
Max.   :1.0000   Max.   :335.0   Max.   :1222.0   Max.   :1800
   distance        NoOfPools       NoOfSites         avrain
Min.   :  250   Min.   :  1.00   Min.   : 0.000   Min.   :124.7
1st Qu.:  500   1st Qu.:  8.00   1st Qu.: 1.000   1st Qu.:141.7
Median : 1000   Median : 18.00   Median : 3.000   Median :148.8
Mean   : 1933   Mean   : 25.11   Mean   : 2.939   Mean   :148.1
3rd Qu.: 2000   3rd Qu.: 32.00   3rd Qu.: 4.000   3rd Qu.:155.0
Max.   :18000   Max.   :232.00   Max.   :10.000   Max.   :198.3
   meanmin          meanmax
Min.   :2.033   Min.   :11.60
1st Qu.:2.567   1st Qu.:12.97
Median :3.000   Median :13.38
Mean   :3.120   Mean   :13.67
3rd Qu.:3.567   3rd Qu.:14.21
Max.   :4.333   Max.   :15.97
```


Metros a norte do ponto de referência
300
250
200
150
100
800
1000
1200
Metros a leste do ponto de referência
pres.abs
0
1


altitude
distance
NoOfPools
NoOfSites
avrain
meanmin
meanmax
Corr: 0.182
Corr: 0.267
Corr: −0.137
Corr: 0.778
Corr: −0.954
Corr: −0.997
Corr: −0.0959
Corr: −0.366
Corr: 0.148
Corr: −0.326
Corr: −0.205
Corr: 0.078
Corr: 0.244
Corr: −0.224
Corr: −0.263
Corr: −0.218
Corr: 0.151
Corr: 0.158
Corr: −0.649
Corr: −0.819
Corr: 0.946


altitude
log(distance)
log(NoOfPools)
NoOfSites
avrain
meanmin
meanmax
Corr: 0.256
Corr: 0.284
Corr: −0.137
Corr: 0.778
Corr: −0.954
Corr: −0.997
Corr: −0.199
Corr: −0.456
Corr: 0.0755
Corr: −0.387
Corr: −0.257
Corr: 0.117
Corr: 0.255
Corr: −0.257
Corr: −0.284
Corr: −0.218
Corr: 0.151
Corr: 0.158
Corr: −0.649
Corr: −0.819
Corr: 0.946

```r
frogs.glm0 <- glm(formula = pres.abs ~ altitude + log(distance) +
                    log(NoOfPools) + NoOfSites + avrain + meanmin + meanmax,
                  family = binomial, data = frogs)
summary(frogs.glm0)
```

```
Call:
glm(formula = pres.abs ~ altitude + log(distance) + log(NoOfPools) +
    NoOfSites + avrain + meanmin + meanmax, family = binomial,
    data = frogs)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9795  -0.7193  -0.2785   0.7964   2.5658

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     4.090e+01  1.327e+02   0.308 0.757845
altitude       -6.648e-03  3.866e-02  -0.172 0.863466
log(distance)  -7.593e-01  2.554e-01  -2.973 0.002945 **
log(NoOfPools)  5.727e-01  2.162e-01   2.649 0.008083 **
NoOfSites      -8.979e-04  1.074e-01  -0.008 0.993330
avrain         -6.793e-03  5.999e-02  -0.113 0.909848
meanmin         5.305e+00  1.543e+00   3.439 0.000584 ***
meanmax        -3.173e+00  4.839e+00  -0.656 0.512048
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 279.99  on 211  degrees of freedom
Residual deviance: 197.62  on 204  degrees of freedom
AIC: 213.62
```

```r
n <- dim(frogs)[1]
p <- 7
y <- frogs$pres.abs
ybar <- mean(y)
ybar
```

```
[1] 0.3726415
```

```r
phat <- fitted(frogs.glm0)
X<-cbind(rep(1,212),frogs$altitude,log(frogs$distance),log(frogs$NoOfPools),
         frogs$NoOfSites,frogs$avrain,frogs$meanmin,frogs$meanmax)
pred<-X%*%frogs.glm0$coefficients
fitted<-cbind(fitted(frogs.glm0), frogs.glm0$fitted.values,exp(pred)/(1+exp(pred)))
fitted[1:5,]
```

```
       [,1]      [,2]      [,3]
2 0.9411072 0.9411072 0.9411072
3 0.9261453 0.9261453 0.9261453
4 0.9045402 0.9045402 0.9045402
5 0.8126903 0.8126903 0.8126903
6 0.9319679 0.9319679 0.9319679
```

```r
# Desvio
my.res.dev <- -2 * sum(y * log(phat) + (1 - y) * log(1 - phat))
c(my.res.dev, summary(frogs.glm0)$deviance)
```

```
[1] 197.6249 197.6249
```

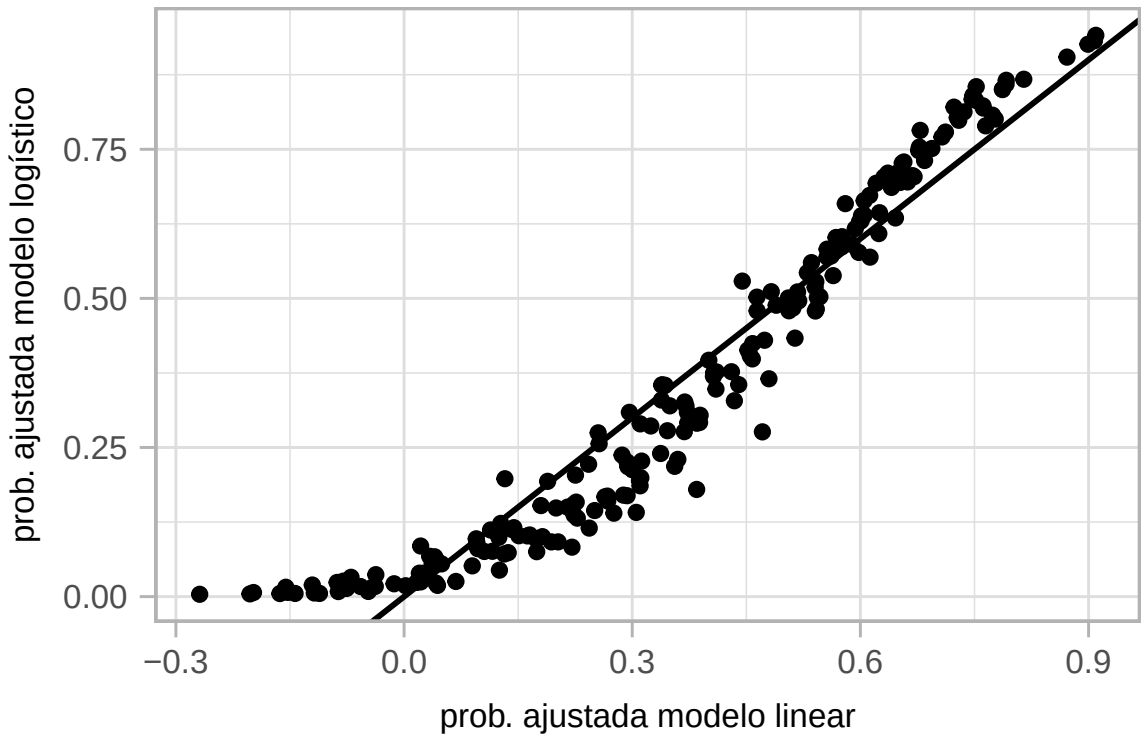
```r
# graus de liberdade do desvio
c(n-p-1, summary(frogs.glm0)$df.residual)
```

```
[1] 204 204
```

```r
# desvio saturado
my.null.dev <- -2 * n * (ybar * log(ybar) + (1 - ybar) * log(1 - ybar))
c(my.null.dev, summary(frogs.glm0)$null.deviance)
```

```
[1] 279.987 279.987
```

```
frogs.lm <- lm(formula = pres.abs ~ altitude + log(distance) +
                    log(NoOfPools) + NoOfSites + avrain + meanmin + meanmax,
                  data = frogs)
phat.lm <- fitted(frogs.lm)
```



Predição em regressão logística

Suponha que obtenhamos dados para um novo site e desejemos prever se os sapos estarão no site ou não.

Se usarmos nosso modelo ajustado, obteremos uma estimativa da probabilidade da presença de sapos.

Mas como transformamos isso em uma previsão?

Devemos usar um ponto de corte de 0,5 e prever sim se a probabilidade estimada for $\geq$ 0,5?

Considere a seguinte matriz de confusão.

	$\hat{Y} = 0$	$\hat{Y} = 1$
$Y = 0$	a	b
$Y = 1$	c	d

Gostaríamos que a e d fossem grandes e b e c pequenos.

Vamos comparar os valores de $b + c$ nos dados fornecidos, para vários valores do ponto de corte.

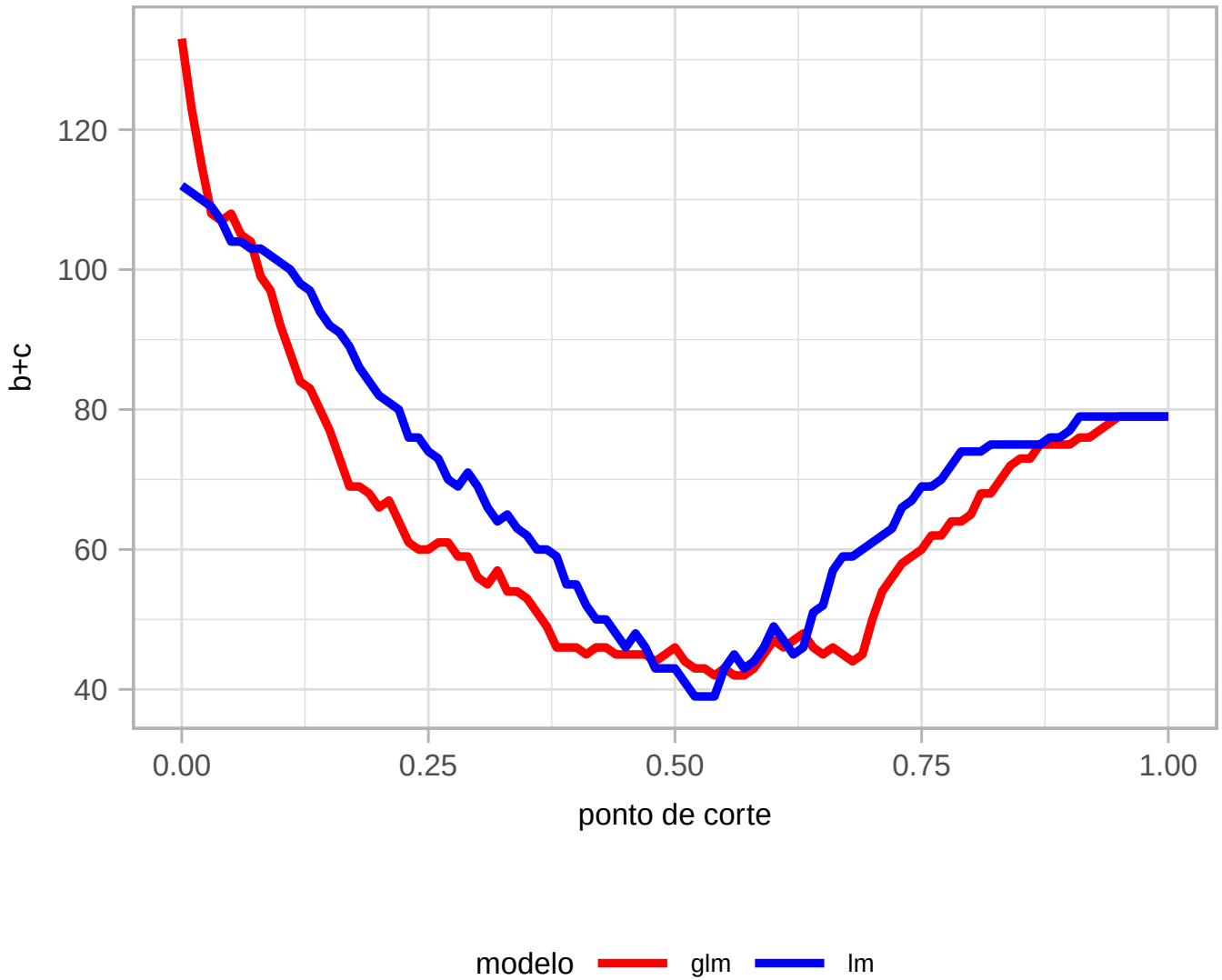
```r
thres.vec <- seq(0, 1, by=0.01)
thres.vec[1:10]
```

```
 [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09
```

```r
conf <- matrix(0, nrow=length(thres.vec), ncol=4)
conf.lm <- matrix(0, nrow=length(thres.vec), ncol=4)
for (i in 1:length(thres.vec)){
  thres <- thres.vec[i]
  conf[i, 1] <- sum((!y) & (phat < thres)) # !y: y=0 -> TRUE
  conf[i, 2] <- sum((!y) & (phat >= thres))
  conf[i, 3] <- sum(y & (phat < thres))
  conf[i, 4] <- sum(y & (phat >= thres))}

for (i in 1:length(thres.vec)){
  thres <- thres.vec[i]
  conf.lm[i, 1] <- sum((!y) & (phat.lm < thres))
  conf.lm[i, 2] <- sum((!y) & (phat.lm >= thres))
  conf.lm[i, 3] <- sum(y & (phat.lm < thres))
  conf.lm[i, 4] <- sum(y & (phat.lm >= thres))}
```

```
bplusc <- conf[,2] + conf[,3]
bplusc.lm <- conf.lm[,2] + conf.lm[,3]
```



```
thres.vec[which.min(bplusc.lm)]
```

```
[1] 0.52
```

```
thres.vec[which.min(bplusc)]
```

```
[1] 0.54
```

Vemos que 0,54 é o melhor ponto de corte para a regressão logística se queremos minimizar $b + c$, o número de erros.

Nossa escolha de $b + c$ foi arbitrário e assumiu que consideramos os dois tipos de erros (falso positivo, falso negativo) como igualmente ruins.

Por exemplo, suponha que um banco queira decidir se aceita a solicitação de empréstimo de um cliente. Existem dois tipos de erros:

1. O banco nega a solicitação de empréstimo, mas o cliente teria reembolsado o empréstimo.
2. O banco concede a solicitação de empréstimo, mas o cliente vai à falência.

O segundo erro é mais grave para o banco; portanto, nessa situação, seria mais interessante darmos mais peso a um erro do que ao outro.

Um segundo exemplo seria, considere um exame médico para detectar uma doença letal.

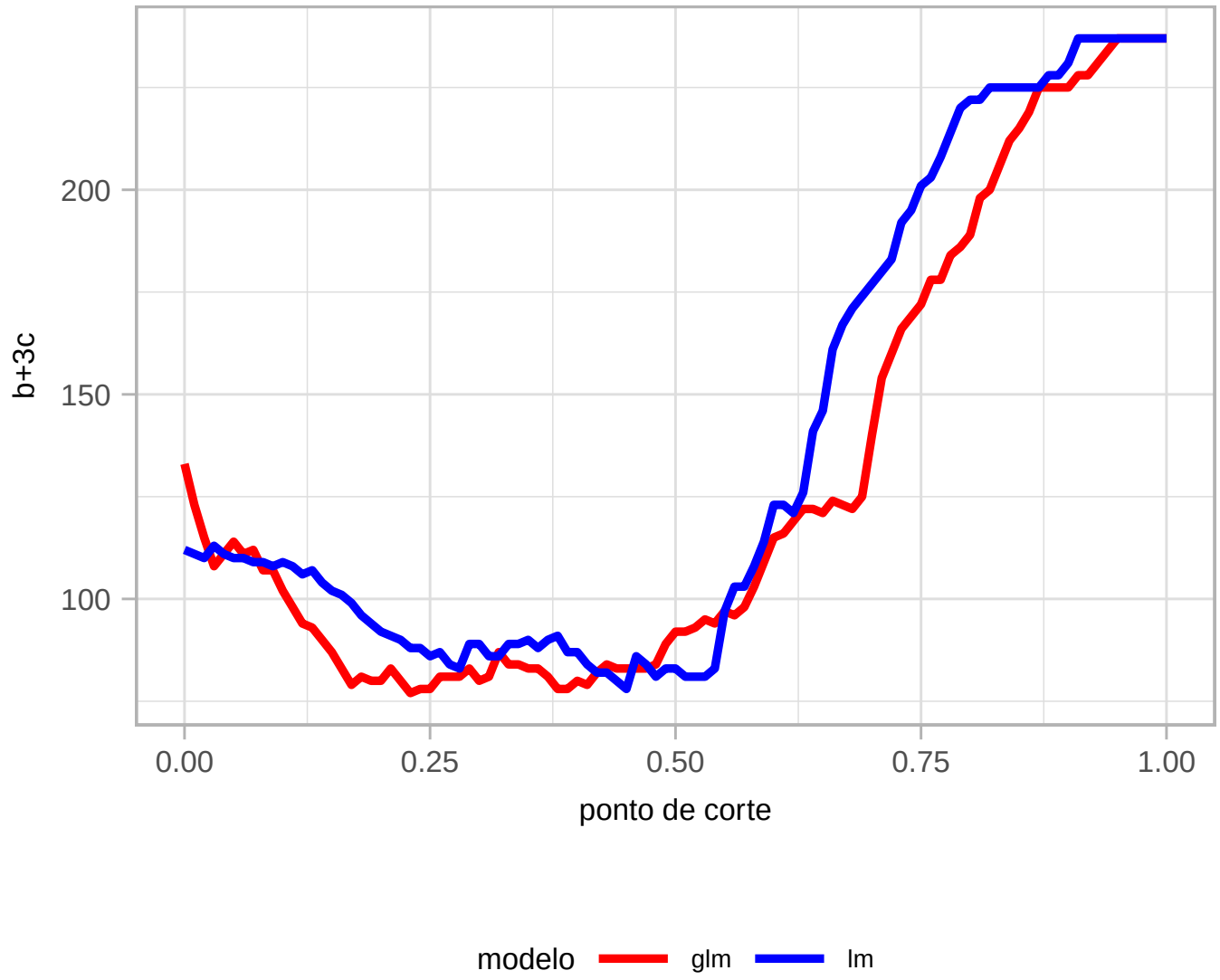
1. O teste indica que o paciente tem a doença, mas o paciente é realmente saudável.
2. O paciente realmente tem a doença, mas o teste não a detecta.

Dependendo da sua perspectiva, você pode colocar mais peso em um erro do que em outro.

Vamos ver o que acontece se usamos $b + 3c$ para o exemplo dos sapo. (Você realmente deseja encontrar sapos!! Você ficará muito triste se não visitar um site que tem sapos apenas porque seu modelo previu por engano que não há sapos por lá.)

```
bplus3c <- conf[,2] + 3*conf[,3]
bplus3c.lm <- conf.lm[,2] + 3*conf.lm[,3]
thres.vec[which.min(bplus3c)]
```

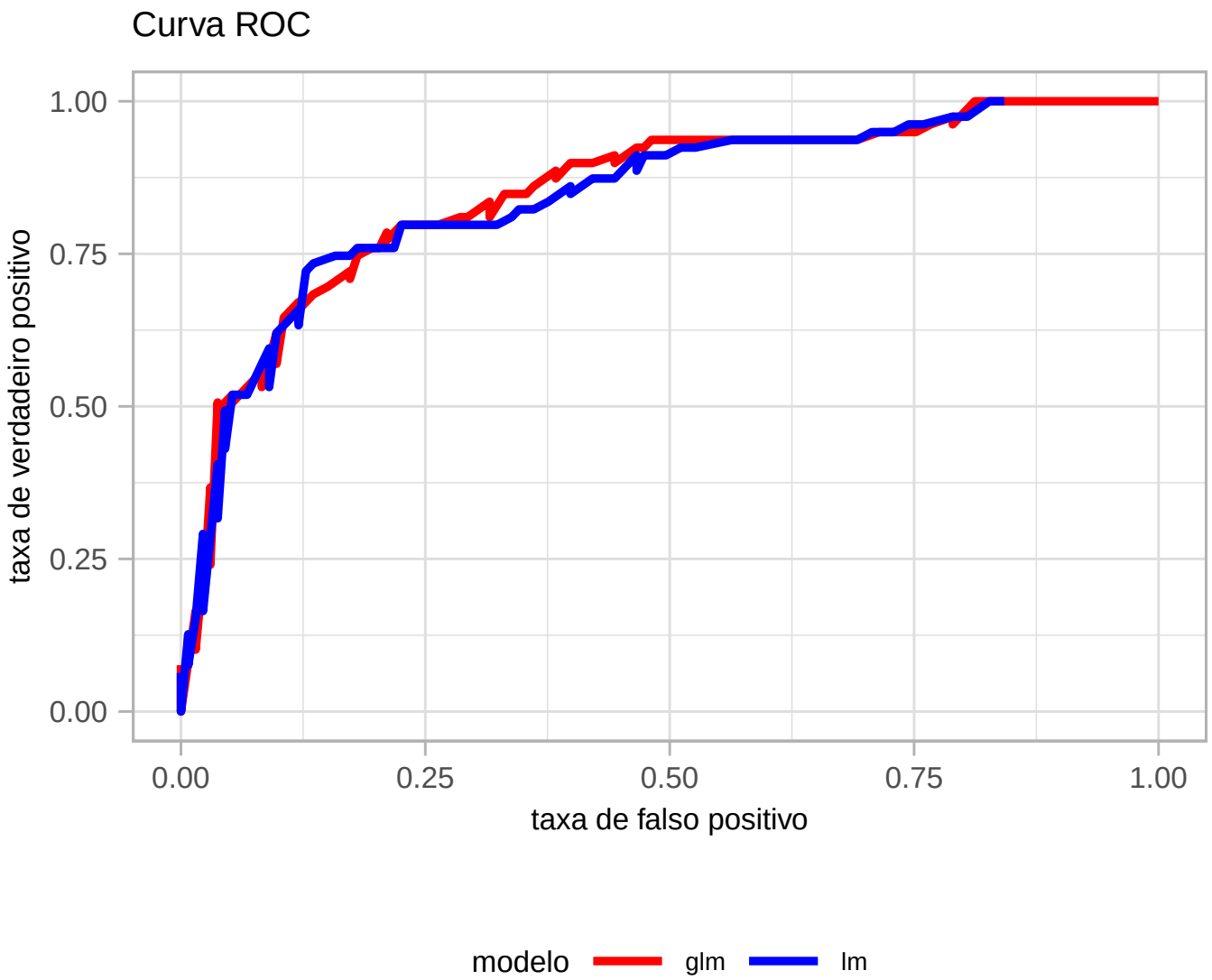
```
[1] 0.23
```



Como esperado, o melhor ponto de corte (em nossos dados) diminuiu (para 0,25), pois queremos rotular os sites como “sim” para evitar a falta de sites com problemas.

A curva ROC (receiver operator characteristic (ROC) é um gráfico da taxa de verdadeiro positivo $\frac{d}{c+d}$ contra taxa de falsos positivos $\frac{b}{a+b}$ conforme você altera o ponto de corte.

```
tpr <- conf[,4] / (conf[,3] + conf[,4])
fpr <- conf[,2] / (conf[,1] + conf[,2])
tpr.lm <- conf.lm[,4] / (conf.lm[,3] + conf.lm[,4])
fpr.lm <- conf.lm[,2] / (conf.lm[,1] + conf.lm[,2])
```


Curva ROC
1.00
0.75
0.50
0.25
0.00
taxa de verdadeiro positivo
0.00
0.25
0.50
0.75
1.00
taxa de falso positivo
modelo
glm
lm

```r
predct1<-ifelse(phat>=0.54,1,0)
conf <- matrix(0, nrow=2, ncol=2)
conf[1, 1] <- sum((!y) & (!predct1))
conf[1, 2] <- sum((!y) & (predct1))
conf[2, 1] <- sum(y & (!predct1))
conf[2, 2] <- sum(y & (predct1))
conf
```

```
     [,1] [,2]
[1,]  117   16
[2,]   26   53
```

```r
ac <- (conf[1,1]+conf[2,2])/(sum(!y)+sum(y))
acuracia <- ac
acuracia
```

```
[1] 0.8018868
```

```r
confusionMatrix(as.factor(predct1), as.factor(y))
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0  16  53
         1 117  26

               Accuracy : 0.1981
                 95% CI : (0.1467, 0.2582)
    No Information Rate : 0.6274
    P-Value [Acc > NIR] : 1

                  Kappa : -0.4728

 Mcnemar's Test P-Value : 1.353e-06

            Sensitivity : 0.12030
            Specificity : 0.32911
         Pos Pred Value : 0.23188
         Neg Pred Value : 0.18182
             Prevalence : 0.62736
         Detection Rate : 0.07547
   Detection Prevalence : 0.32547
      Balanced Accuracy : 0.22471
```

Considerando só as covariáveis significativas:

```
frogs.glm <- glm(formula = pres.abs ~ log(distance) + log(NoOfPools) +
                    meanmin + meanmax, family = binomial, data = frogs)
summary(frogs.glm)
```

```
Call:
glm(formula = pres.abs ~ log(distance) + log(NoOfPools) + meanmin +
    meanmax, family = binomial, data = frogs)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9753  -0.7224  -0.2780   0.7974   2.5736

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)    18.5268     5.2673   3.517 0.000436 ***
log(distance)  -0.7547     0.2261  -3.338 0.000844 ***
log(NoOfPools)  0.5707     0.2152   2.652 0.007999 **
meanmin         5.3791     1.1928   4.509  6.5e-06 ***
meanmax        -2.3821     0.6234  -3.821 0.000133 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 279.99  on 211  degrees of freedom
Residual deviance: 197.66  on 207  degrees of freedom
AIC: 207.66

Number of Fisher Scoring iterations: 5
```

```
phat <- fitted(frogs.glm)
X<-cbind(rep(1,212),log(frogs$distance),log(frogs$NoOfPools),
         frogs$meanmin,frogs$meanmax)
pred<-X%*%frogs.glm$coefficients
fitted<-cbind(fitted(frogs.glm), frogs.glm$fitted.values,exp(pred)/(1+exp(pred)))
fitted[1:5,]
```

```
       [,1]      [,2]      [,3]
2 0.9416691 0.9416691 0.9416691
3 0.9259228 0.9259228 0.9259228
4 0.9029415 0.9029415 0.9029415
5 0.8119619 0.8119619 0.8119619
6 0.9314070 0.9314070 0.9314070
```

```r
# Desvio
my.res.dev <- -2 * sum(y * log(phat) + (1 - y) * log(1 - phat))
c(my.res.dev, summary(frogs.glm)$deviance)
```

```
[1] 197.6561 197.6561
```

```r
# graus de liberdade do desvio
c(n-p-1, summary(frogs.glm)$df.residual)
```

```
[1] 204 207
```

```r
# desvio saturado
my.null.dev <- -2 * n * (ybar * log(ybar) + (1 - ybar) * log(1 - ybar))
c(my.null.dev, summary(frogs.glm)$null.deviance)
```

```
[1] 279.987 279.987
```

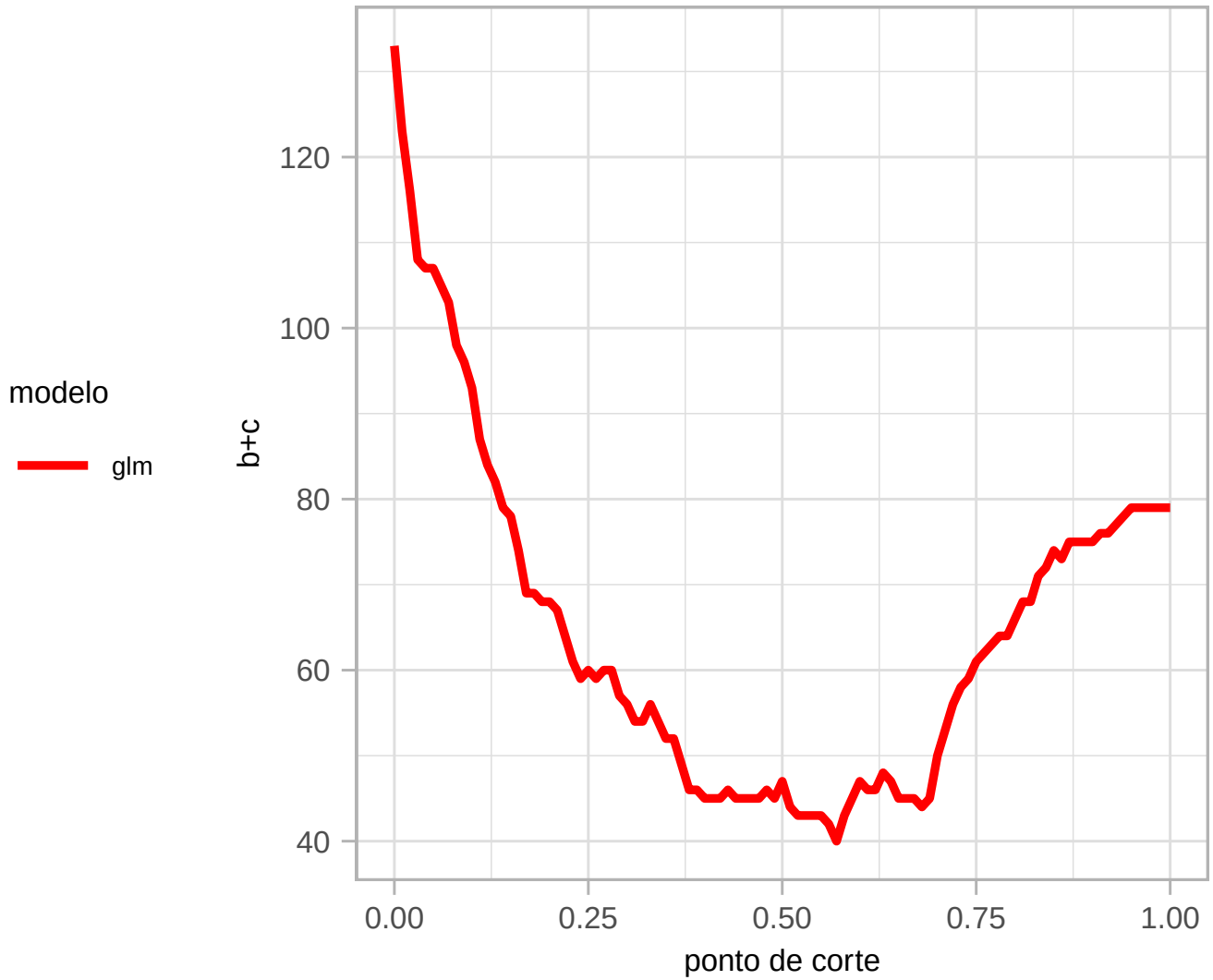
```r
thres.vec <- seq(0, 1, by=0.01)
thres.vec[1:10]
```

```
 [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09
```

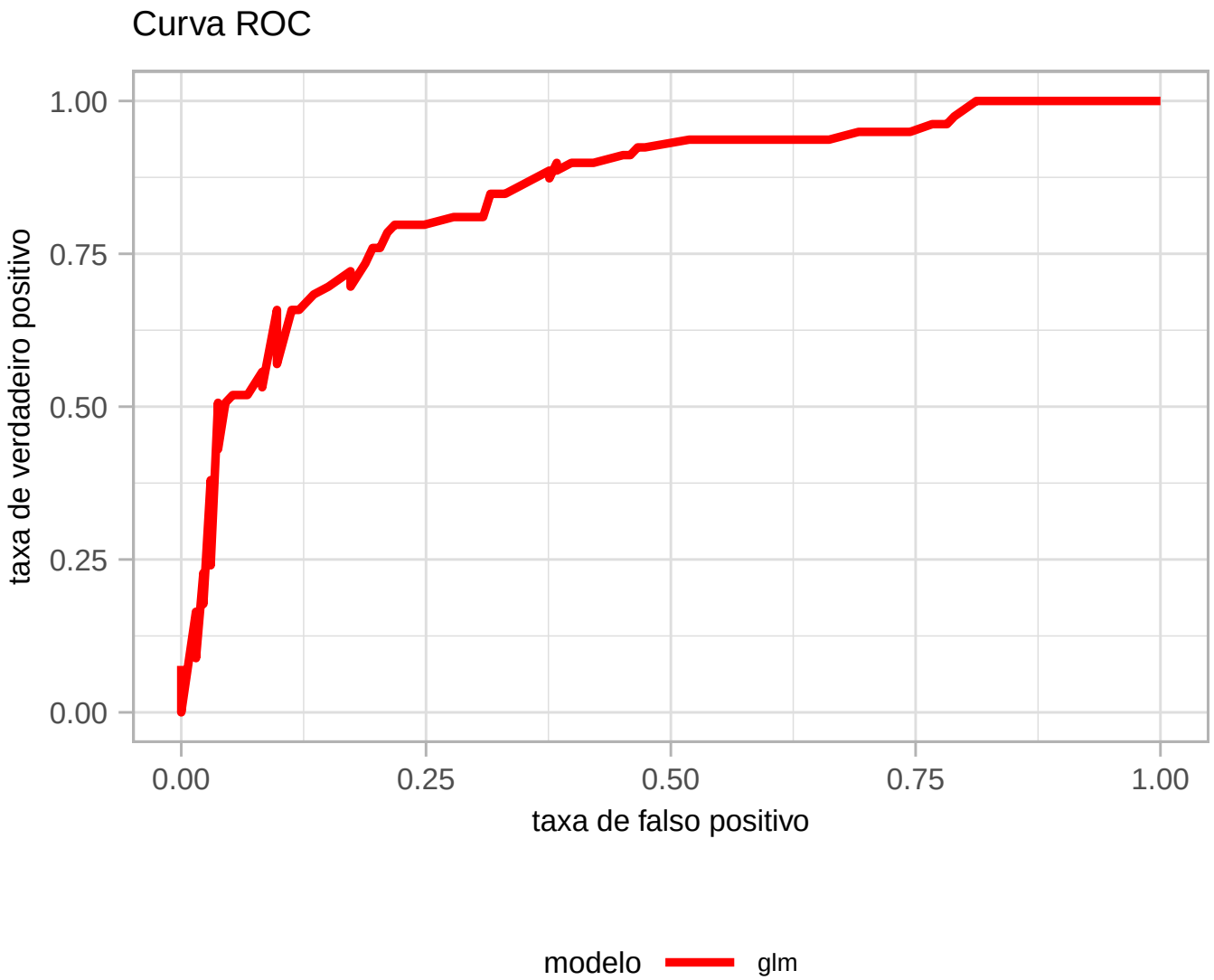
```r
conf <- matrix(0, nrow=length(thres.vec), ncol=4)
conf.lm <- matrix(0, nrow=length(thres.vec), ncol=4)
for (i in 1:length(thres.vec)){
  thres <- thres.vec[i]
  conf[i, 1] <- sum((!y) & (phat < thres))
  conf[i, 2] <- sum((!y) & (phat >= thres))
  conf[i, 3] <- sum(y & (phat < thres))
  conf[i, 4] <- sum(y & (phat >= thres))}
```

```
bplusc <- conf[,2] + conf[,3]
thres.vec[which.min(bplusc)]
```

```
[1] 0.57
```



```r
tpr <- conf[,4] / (conf[,3] + conf[,4])
fpr <- conf[,2] / (conf[,1] + conf[,2])
```



Área sob a curva (AUC):

```
auc(fpr,tpr, type = 'spline')
```

```
[1] 0.850367
```

Exercício

1. Fazer a análise para os dados `spam` do mesmo pacote do `frogs`.

2. Prever a sobrevivência no Titanic, utilizando o conjunto de dados `titanic` do pacote `titanic`.

Referência

- Notas de aula do Prof. Gilberto de Paula.
- Agresti, A. (2015). *Foundations of Linear and Generalized Linear Models*. Wiley series in probability and statistics.
- Faraway, J. J. (2006). *Extending the Linear Model with R. Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Chapaman and Hall/CRC.
- STAT151A: Lab 11 - https://www.stat.berkeley.edu/~blfang/STAT151A/STAT151A_lab11.html