

ME720 - Modelos Lineares Generalizados

Parte 13 - Criando a sua própria função glm

Profa. **Larissa Avila Matos**

- Funções para calcular o vetor de escore e a matriz Hessiana dependendo da distribuição escolhida.

```
Poisson <- function(y, X, beta){  
  mu <- exp(X%*%beta)  
  f <- dpois(y,mu)  
  u <- t(y - mu)%*%X #vetor escore  
  W <- diag(as.vector(mu))  
  H <- -t(X)%*%W%*%X #matriz Hessiana  
  output_list <- list(f = f, u = u, H = H)  
  return(output_list)  
}
```

```

Binaria <- function(y, X, beta){
  pi <- exp(X%*%beta)/(1+exp(X%*%beta))
  f <- prod(ifelse(y==1,pi,1-pi))
  u <- t(t(y- pi)%*%X)
  W <- diag(as.vector(pi*(1-pi)))
  H <- -t(X)%*%W%*%X
  output_list <- list(f = f, u = u, H = H)
  return(output_list)
}

```

```

Binomial <- function(y,X,beta){
  pi <- exp(X%*%beta)/(1+exp(X%*%beta)) # vetor de probabilidades
  f <- prod(ifelse(y==1,pi,1-pi))
  df <- t(t(y- pi)%*%X) # primeira derivada
  W <- diag(as.vector(pi*(1-pi)))
  ddf <- -t(X)%*%W%*%X # segunda derivada
  output_list <- list(f = f, df = df,ddf = ddf)
  return(output_list)
}

```

■ Newton–Raphson.

```
newton <- function(func, init, eps=1e-16, maxiter=50){
  params <- init
  out <- matrix(NA, nrow=maxiter+1, ncol=length(t(beta)))
  out[1,] <- t(params)
  i <- 1
  continue <- T
  while(continue){
    i <- i+1
    funcOut <- func(params)
    params <- params - solve(funcOut$H)%*%funcOut$u
    if(sum(is.na(params))>0){stop("NA nas estimativas")}
    out[i,] <- t(params)
    continue <- (sqrt(t(funcOut$u)%*%funcOut$u) > eps) && (i <= maxiter)
  }
  if (i > maxiter) {
    warning("Máximo número de iterações atingido")
  }
  out <- out[!is.na(out[,1]),]
  output_list <- list(params = params, out = out, iter = i)
  return(output_list)
}
```

■ Função para o ajuste do modelo.

```
ajusteMLG <-function(y, X, init, eps=1e-13, maxiter=50, ditr="Poisson"){  
  
  n<-dim(X)[1]  
  p=length(t(init))  
  
  if(ditr=="Poisson"){  
    outNewton <- newton(function(input){Poisson(y = y, X = X, beta = input)},  
                          init, eps, maxiter)  
    est <- outNewton$params  
    v.ajustado <- exp(X%*%est)  
    preditor.linear=X%*%est  
    desvio <- sum(2*(y*(ifelse(y==0,0,log(y))-log(v.ajustado))-y+v.ajustado))  
    AIC <- -2*(sum(dpois(y,v.ajustado,log=T))) + 2*p  
    BIC <- -2*(sum(dpois(y,v.ajustado,log=T))) + 2*p*log(n)  
    W <- diag(as.vector(v.ajustado))  
    H<-diag(sqrtm(W)%*%X%*%solve(t(X)%*%W%*%X)%*%t(X)%*%solve(sqrtm(W)))  
    res <- sqrt(2*(y*(ifelse(y==0,0,log(y))-log(v.ajustado))-y+v.ajustado))/sqrt(1-H)  
    residuos <- ifelse(y-v.ajustado>0,res,-res)  
    residuos.pearson.p <- (y-v.ajustado)/sqrt(v.ajustado*(1-H))  
  }  
}
```

```

if(ditr=="Binaria"){
  outNewton <- newton(function(input){Binaria(y = y, X = X, beta = input)},
    init, eps, maxiter)
  est <- outNewton$params
  v.ajustado <- exp(X%%est)/(1+exp(X%%est))
  predictor.linear=X%%est
  desvio <- sum(-2*log(v.ajustado))
  AIC <- -2*(sum(dbinom(y,size=1,prob=v.ajustado,log=T))) + 2*p
  BIC <- -2*(sum(dbinom(y,size=1,prob=v.ajustado,log=T))) + 2*p*log(n)
  W <- diag(as.vector(v.ajustado))
  H<-diag(sqrtm(W)%*%X%%solve(t(X)%*%W%%X)%*%t(X)%*%solve(sqrtm(W)))
  residuos.pearson <- (y-v.ajustado)/sqrt(v.ajustado*(1-v.ajustado))
  res <- -2*log(v.ajustado)/sqrt(1-H)
  residuos <- ifelse(y-v.ajustado>0,res,-res)
  residuos.pearson.p <- residuos.pearson/sqrt(1-H)
}

```

```

I=t(X)%*%W%*%X
se=sqrt(diag(solve(I)))
z.value=est/se
p.value=pnorm(abs(z.value), lower.tail = F)

z.alpha<-qnorm(0.975)
LI=est-z.alpha*se
LS=est+z.alpha*se

saida<-list(out=outNewton$out, est=est, iter=outNewton$iter, se=se,
            v.ajustado=v.ajustado, preditor.linear=preditor.linear,
            z.value=z.value, p.value=p.value, LI=LI, LS=LS,
            AIC=AIC, BIC=BIC, desvio=desvio, gl.res=(n-p),
            residuos=residuos, residuos.pearson.p=residuos.pearson.p)
return(saida)
}

```

■ Função geral do MLG.

```
glmLarissa = function(y, X, init, eps=1e-13, maxiter=50, ditr="Poisson", show=T){  
  
  # Validações  
  
  ly <- length(y)  
  if(nrow(X)!=ly){stop("Número de linhas na matriz X difere do número de  
    observações em y")}  
  
  if(ncol(X)!=nrow(init)){stop("Valores inválidos para os parâmetros iniciais")}  
  
  if(maxiter <= 0 | maxiter%%1 != 0) stop("maxiter (número máximo de iterações)  
    precisa ser um inteiro positivo.")  
  
  if(eps <=0 | eps > 1) stop("o erro precisa pertencer ao intervalo (0,1]")  
  
  if(ditr!="Poisson"|ditr!="Binaria"|ditr!="Binomial") stop("Distribuição  
    não definida")  
  
  if(show!=T | show!=F) stop("essa variável precisa ser T ou F")  
}
```


Outputs

```
out = ajusteMLG(y, X, init, eps, maxiter, ditr)
betas = round(out$est, 4)
se = round(out$se, 4)
z.value = round(out$z.value, 4)
p.value = out$p.value
LI = round(out$LI, 2)
LS = round(out$LS, 2)
```

Criterios

```
desvio = round(out$desvio,3)
desvioinf = matrix(c(desvio,out$gl.res),1,2)
AIC = round(out$AIC,3)
BIC = round(out$BIC,3)
iter = matrix(c(out$iter),1,1)
```

```
Estimativas = cbind(betas,se,z.value,p.value)
```

```
colx = ncol(as.matrix(X))
namesx = paste0('x',1)
IC=paste0('[' ,LI[1],',' ,LS[1],"]")
if(ncol(as.matrix(X))>1){
  for(i in 2:ncol(as.matrix(X))){namesx = cbind(namesx, paste0('x',i))
  IC = rbind(IC,paste0('[' ,LI[i],',' ,LS[i],"]"))}}
```

```

Estimativas<-data.frame(Estimativas,IC=IC)
dimnames(Estimativas) = list(c(namesx[1:colx]),c("Estimativas", "SE", "valor z",
                                                "p-valor", "IC's"))

dimnames(desvioinf) = list(c('Desvio:  '),c(" ", 'g.l'))
dimnames(iter) = list(c('Número de iterações:  '),c(" "))
criteria = as.matrix(c(AIC,BIC))
dimnames(criteria) = list(c("AIC:", "BIC:"),c(" "))

if(show==T){
  cat('\n')
  cat('-----\n')
  cat('                                MLG                                \n')
  cat('-----\n')
  print(Estimativas)
  cat('-----\n')
  print(desvioinf)
  cat('-----\n')
  print(iter)
  cat('-----\n')
  cat('Critérios de seleção do Modelo \n')
  print(criteria)
  cat('-----\n')}

class(out) <- "mlg"
return(invisible(out))
}

```

- Gerando dados para um modelo de dados Binários, onde $Y = 0, 1$.

```
x1=rnorm(100,10,2)
x2=rnorm(100,5,1)
x=cbind(x1,x2)
beta <- as.matrix(c(-1,2))
y=rbinom(n = 100, size = 1, prob = exp(x%*%beta)/(1+exp(x%*%beta)))
```

```
fit.bin<-glmLarissa(y, X=x, init=beta, eps=1e-10, maxiter=50,  
                    ditr="Binaria", show=T)
```

MLG

	Estimativas	SE	valor z	p-valor	IC's
x1	-1.107	0.0708	-15.6333	2.159386e-55	[-1.25;-0.97]
x2	2.242	0.1177	19.0440	3.687466e-81	[2.01;2.47]

g.l
Desvio: 329.583 98

Número de iterações: 6

Critérios de seleção do Modelo

AIC: 71.864

BIC: 86.285

```
fit.glm<-glm(y~x1+x2-1,family=binomial(link = "logit"))
summary(fit.glm)
```

Call:

```
glm(formula = y ~ x1 + x2 - 1, family = binomial(link = "logit"))
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.1734	-0.4673	-0.1226	0.4506	2.6155

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
x1	-1.1070	0.2241	-4.940	7.81e-07 ***
x2	2.2420	0.4656	4.816	1.47e-06 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 138.629 on 100 degrees of freedom
Residual deviance: 67.864 on 98 degrees of freedom
AIC: 71.864

Number of Fisher Scoring iterations: 6

- Gerando um conjunto de dados para um modelo Poisson.

```
x=cbind(rep(1,100),runif(100,0,2))  
beta<-c(1,3)  
y=rpois(n=100,lambda=exp(x%%beta))  
init=solve(t(x)%%x)%%t(x)%%sqrt(y)
```

```
fit.poisson<-glmLarissa(y, X=x, init=beta, eps=1e-10, maxiter=50,  
                        ditr="Poisson", show=T)
```

MLG

	Estimativas	SE	valor z	p-valor	IC's
x1	0.9983	0.0383	26.0747	3.534848e-150	[0.92;1.07]
x2	3.0059	0.0222	135.6132	0.000000e+00	[2.96;3.05]

g.l
Desvio: 101.633 98

Número de iterações: 5

Critérios de seleção do Modelo

AIC: 701.336
BIC: 715.757

```
fit.glm<-glm(y~x-1,family=poisson(link = "log"))
summary(fit.glm)
```

Call:

```
glm(formula = y ~ x - 1, family = poisson(link = "log"))
```

Deviance Residuals:

	Min	1Q	Median	3Q	Max
	-2.06475	-0.78880	-0.01273	0.72430	2.53309

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
x1	0.99830	0.03829	26.07	<2e-16 ***
x2	3.00591	0.02217	135.61	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 205217.13 on 100 degrees of freedom
Residual deviance: 101.63 on 98 degrees of freedom
AIC: 701.34

Number of Fisher Scoring iterations: 4

Exemplo Exposição de Bactérias

```
bacterias<-c(175,108,95,82,71,50,49,31,28,17,16,11)
exposicao<-c(1,2,3,4,5,6,7,8,9,10,11,12)
fit<-glm(bacterias~exposicao,family=poisson(link = "log"))
summary(fit)
```

Call:

```
glm(formula = bacterias ~ exposicao, family = poisson(link = "log"))
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7703	-0.5715	-0.1019	0.5496	1.2794

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	5.30557	0.06348	83.58	<2e-16 ***
exposicao	-0.22890	0.01270	-18.02	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 393.6292 on 11 degrees of freedom

```
X<-cbind(rep(1,length(exposicao)),exposicao)
init<-matrix(solve(t(X)%*%X)%*%t(X)%*%sqrt(bacterias),2,1)
fit.mlg<-glmLarissa(y=bacterias, X=X, init=init, eps=1e-10,
                    maxiter=50, ditr="Poisson", show=T)
```

MLG

	Estimativas	SE	valor z	p-valor	IC's
x1	5.3056	0.0635	83.5750	0.000000e+00	[5.18;5.43]
x2	-0.2289	0.0127	-18.0234	6.386627e-73	[-0.25;-0.2]

g.1

Desvio: 8.422 10

Número de iterações: 13

Critérios de seleção do Modelo

AIC: 80.182

BIC: 86.122

```
residuos<-cbind(residuals.glm(fit),fit.mlg$residuos)
colnames(residuos)<-c("glm","mlgPoisson")
round(residuos,3)
```

	glm	mlgPoisson
1	1.149	1.150
2	-1.770	-1.772
3	-0.641	-0.641
4	0.151	0.151
5	0.842	0.842
6	-0.143	-0.143
7	1.279	1.281
8	-0.227	-0.227
9	0.452	0.453
10	-0.780	-0.783
11	-0.061	-0.061
12	-0.548	-0.552

```
residuos<-cbind(residuals.glm(fit,type = "pearson"),fit.mlg$residuos.pearson.p)
colnames(residuos)<-c("glm","mlgPoisson")
round(residuos,3)
```

	glm	mlgPoisson
1	1.166	1.168
2	-1.723	-1.725
3	-0.634	-0.634
4	0.151	0.152
5	0.856	0.857
6	-0.143	-0.143
7	1.322	1.323
8	-0.225	-0.225
9	0.459	0.460
10	-0.757	-0.760
11	-0.061	-0.061
12	-0.534	-0.537

```
valores.ajustados<-cbind(bacterias,fit$fitted.values,fit.mlg$v.ajustado)
colnames(valores.ajustados)<-c("bacterias","glm","mlgPoisson")
valores.ajustados
```

	bacterias	glm	mlgPoisson
1	175	160.24051	160.24051
2	108	127.45716	127.45716
3	95	101.38091	101.38091
4	82	80.63955	80.63955
5	71	64.14163	64.14163
6	50	51.01900	51.01900
7	49	40.58110	40.58110
8	31	32.27868	32.27868
9	28	25.67484	25.67484
10	17	20.42206	20.42206
11	16	16.24394	16.24394
12	11	12.92062	12.92062

- Agresti, A. (2015). *Foundations of Linear and Generalized Linear Models*. Wiley series in probability and statistics.