CE043 - GAMLSS

Estimação e Inferência I: A Função `gamlss()`

Silva, J.P; Taconeli, C.A.

03/08/2020

Vamos explorar com mais detalhes a função `gamlss()`.

Argumentos da função `gamlss()`

```
library(gamlss)
#help(gamlss)
```

Algumas observações sobre o uso das funções:

- `weights`: os pesos devem ser usados nas condições
  - para ponderar observações (pesos iguais a 1 ou 0) e
  - para uma análise de verossimilhança em que a contribuição das observações é ponderada por `weights`; tipicamente isso é apropriado quando algumas linhas dos dados são idênticas e os pesos representam as frequências destas linhas.

Qualquer outro uso não é recomendado e pode ter alguns efeitos indesejados.

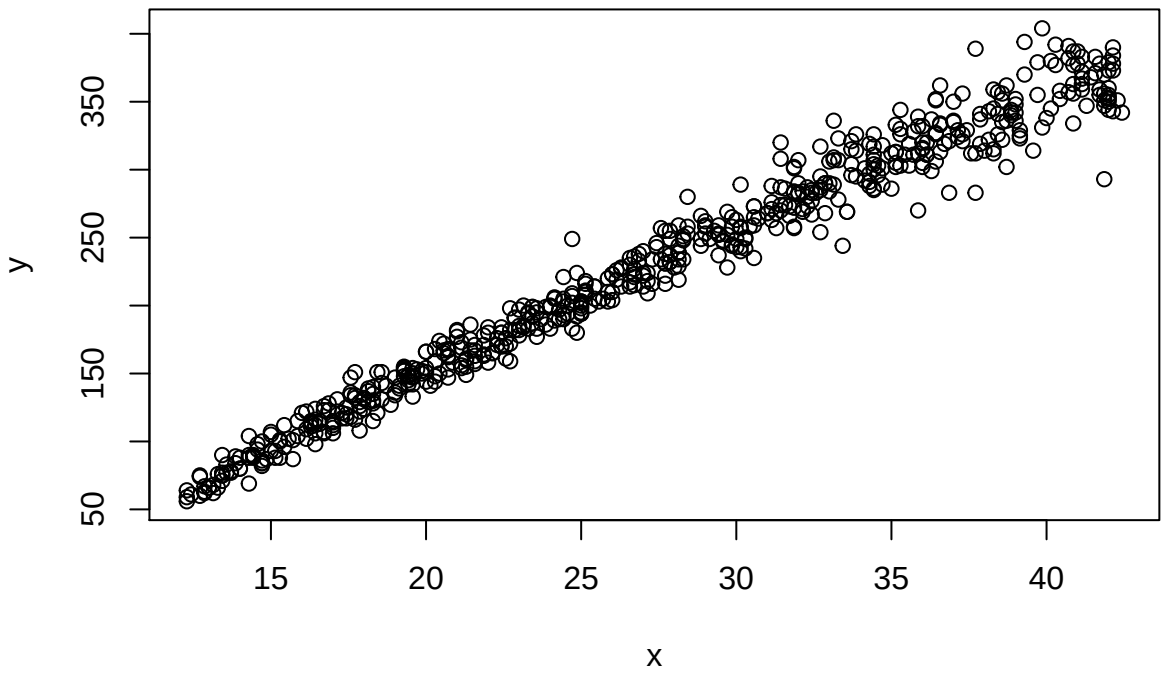
- `na.action` e `subset`: os argumentos `na.action` e `subset`, comuns em funções como `lm()` e `glm()` foram removidos da função `gamlss()`.

A justificada é que, embora haja apenas um conjunto de dados no modelo, são criados até quatro conjuntos diferentes (um para cada parâmetro) e, por consistência, é mais fácil aplicar `subset` e `na.action` em todo o conjunto de dados. Para subconjuntos use `data=subset(mydata, subset=<the relevant condition>)` e para `na.action` use `data=na.omit(mydata)`.

O exemplo `abdom`

Vamos ilustrar o uso da função `gamlss()` com o conjunto de dados abdom. A resposta é circunferência abdominal (y) e a variável explicativa é idade gestacional em semanas (x). Os dados compreendem 610 observações.

```
data(abdom)
# help(abdom)
with(abdom, plot(y~x))
```



A seguir, ajustaremos termos suaves em x, tanto para μ como σ assumindo uma distribuição normal usando o algoritmo RS (padrão).

```
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=NO, data=abdom)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GAMLSS-RS iteration 3: Global Deviance = 4785.696
```

A *deviance* aumenta levemente durante as iterações. Isso pode ocorrer se são usados termos aditivos de suavização, já que os graus de liberdade nos diferentes ajustes podem mudar.

O ajuste com o algoritmo CG é dado por

```
h <- gamlss(y~pb(x),sigma.fo=~pb(x),family=NO,data=abdom,method=CG())
```

```
## GAMLSS-CG iteration 1: Global Deviance = 6165.522
## GAMLSS-CG iteration 2: Global Deviance = 5633.981
## GAMLSS-CG iteration 3: Global Deviance = 5204.626
## GAMLSS-CG iteration 4: Global Deviance = 4936.139
## GAMLSS-CG iteration 5: Global Deviance = 4820.149
## GAMLSS-CG iteration 6: Global Deviance = 4788.767
## GAMLSS-CG iteration 7: Global Deviance = 4785.825
## GAMLSS-CG iteration 8: Global Deviance = 4785.695
## GAMLSS-CG iteration 9: Global Deviance = 4785.695
```

E a versão mista

```
h <- gamlss(y~pb(x),sigma.fo=~pb(x),family=NO,data=abdom, method=mixed(2,20))
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GAMLSS-CG iteration 1: Global Deviance = 4785.696
```

No exemplo acima foram necessários dois ciclos do RS e um ciclo do CG. Todos os métodos chegaram essencialmente no mesmo modelo ajustado.

Funções de controle do algoritmo

A função `gamlss.control` controla as iterações externas do algoritmo.

```
#?gamlss.control
```

A função `glim.control` controla as iterações internas do algoritmo.

```
#?glim.control
```

A seguir um exemplo de como mudar o critério de convergência `c.crit`. Primeiro ajustamos um modelo com o critério *default* de 0.001:

```
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=NO, data=abdom)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GAMLSS-RS iteration 3: Global Deviance = 4785.696
```

Os argumentos de `gamlss.control` ou `glim.control` podem ser mudados diretamente dentro da função `gamlss`:

```
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=NO, data=abdom, c.crit=0.000001)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GAMLSS-RS iteration 3: Global Deviance = 4785.696
## GAMLSS-RS iteration 4: Global Deviance = 4785.696
## GAMLSS-RS iteration 5: Global Deviance = 4785.696
```

Alternativamente, podemos mudar o critério de convergência usando o argumento `control` definido dentro de `gamlss.control()`:

```
control1<-gamlss.control(c.crit=0.000001)
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=NO, data=abdom, gamlss.control=control1)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GAMLSS-RS iteration 3: Global Deviance = 4785.696
```

Vamos mudar o valor default de trace diretamente dentro do `gamlss()`:

```
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=NO, data=abdom, glm.trace=TRUE)
```

```
## GLIM iteration 1 for mu: Global Deviance = 6607.265
## GLIM iteration 2 for mu: Global Deviance = 6607.265
## GLIM iteration 1 for sigma: Global Deviance = 6036.217
## GLIM iteration 2 for sigma: Global Deviance = 5522.867
## GLIM iteration 3 for sigma: Global Deviance = 5124.676
## GLIM iteration 4 for sigma: Global Deviance = 4889.783
## GLIM iteration 5 for sigma: Global Deviance = 4801.039
## GLIM iteration 6 for sigma: Global Deviance = 4787.047
## GLIM iteration 7 for sigma: Global Deviance = 4786.698
## GLIM iteration 8 for sigma: Global Deviance = 4786.697
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GLIM iteration 1 for mu: Global Deviance = 4785.707
## GLIM iteration 2 for mu: Global Deviance = 4785.707
## GLIM iteration 1 for sigma: Global Deviance = 4785.695
## GLIM iteration 2 for sigma: Global Deviance = 4785.695
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GLIM iteration 1 for mu: Global Deviance = 4785.696
## GLIM iteration 1 for sigma: Global Deviance = 4785.696
## GAMLSS-RS iteration 3: Global Deviance = 4785.696
```

Alternativamente `trace` pode ser mudada usando o argumento `i.control` definido dentro de `glim.control()`:

```
control2 <- glim.control(glm.trace=TRUE)
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=NO, data=abdom, i.control=control2)
```

```
## GLIM iteration 1 for mu: Global Deviance = 6607.265
## GLIM iteration 2 for mu: Global Deviance = 6607.265
## GLIM iteration 1 for sigma: Global Deviance = 6036.217
## GLIM iteration 2 for sigma: Global Deviance = 5522.867
## GLIM iteration 3 for sigma: Global Deviance = 5124.676
## GLIM iteration 4 for sigma: Global Deviance = 4889.783
## GLIM iteration 5 for sigma: Global Deviance = 4801.039
## GLIM iteration 6 for sigma: Global Deviance = 4787.047
## GLIM iteration 7 for sigma: Global Deviance = 4786.698
## GLIM iteration 8 for sigma: Global Deviance = 4786.697
## GAMLSS-RS iteration 1: Global Deviance = 4786.697
## GLIM iteration 1 for mu: Global Deviance = 4785.707
## GLIM iteration 2 for mu: Global Deviance = 4785.707
## GLIM iteration 1 for sigma: Global Deviance = 4785.695
## GLIM iteration 2 for sigma: Global Deviance = 4785.695
## GAMLSS-RS iteration 2: Global Deviance = 4785.695
## GLIM iteration 1 for mu: Global Deviance = 4785.696
## GLIM iteration 1 for sigma: Global Deviance = 4785.696
## GAMLSS-RS iteration 3: Global Deviance = 4785.696
```

Isso é útil para checar a convergência de parâmetros individuais, mas a menos que se suspeite de um problema, é melhor `trace` no valor *default*.

- Recomendação: se for usado um grande conjunto de dados (mais que 10.000 observações), e o usuário realiza uma análise exploratória que exige o ajuste de muitos modelos de forma rápida, é recomendável mudar `c.crit` em para algo como 0.01 ou mesmo 0.1.

Vamos usar uma distribuição da família t para os dados. A opção `family` para a distribuição t é `TF` e o parâmetro de graus de liberdade é ν, que é ajustado como uma constante por padrão:

```
h <- gamlss(y~pb(x),sigma.fo=~pb(x),family=TF,data=abdom,trace=FALSE)
```

O valor ajustado para ν é 11.42, obtido como:

```
fitted(h,"nu")[1]# ou
```

```
##        1
## 11.42469
```

```
exp(coef(h,"nu"))
```

```
## (Intercept)
##    11.42469
```

Em algumas situações o usuário pode desejar fixar parâmetros da distribuição em um algum específico valor. Por exemplo, fixar os graus de liberdade da t em 10. Isso pode ser feito com os argumentos `nu.start` e `nu.fix`:

```
h1 <- gamlss(y~pb(x), sigma.fo=~pb(x), family=TF, data=abdom, nu.start=10, nu.fix=TRUE,
             trace=FALSE)
```

As funções `refit` e `update`

- A função `refit()` é usada se o componente `converged` do objeto `gamlss` ajustado é `FALSE`, isto é, quando o número máximo de iterações (`n.cyc`) foi alcançado sem convergência. O valor padrão para `n.cyc` é 20, que é suficiente para a maioria dos problemas. Aqui um exemplo artificial no qual forçamos o algoritmo parar na terceira iteração e continuar com `refit()`.

```
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=TF, data=abdom, n.cyc=3)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4780.234
## GAMLSS-RS iteration 2: Global Deviance = 4777.493
## GAMLSS-RS iteration 3: Global Deviance = 4777.519
```

```
## Warning in RS(): Algorithm RS has not yet converged
```

```
h <- refit(h)
```

```
## GAMLSS-RS iteration 4: Global Deviance = 4777.52
```

- A função `update()` é usada para atualizar fórmulas ou outros argumentos de um objeto gamlss ajustado. Para atualizar fórmulas, `update()` use a função `update.formula()`.

```
#?update.gamlss
```

O objeto `gamlss`

A função `gamlss` retorna um objeto da classe `gamlss S3`.

```
h <- gamlss(y~pb(x), sigma.fo=~pb(x), family=TF, data=abdom)
```

```
## GAMLSS-RS iteration 1: Global Deviance = 4780.234
## GAMLSS-RS iteration 2: Global Deviance = 4777.493
## GAMLSS-RS iteration 3: Global Deviance = 4777.519
## GAMLSS-RS iteration 4: Global Deviance = 4777.52
```

Usando a função `names` podemos checar os componentes do objeto `h`.

```
names(h)#ver ?gamlss
```

```
##  [1] "family"             "parameters"         "call"
##  [4] "y"                  "control"            "weights"
##  [7] "G.deviance"         "N"                  "rqres"
## [10] "iter"               "type"               "method"
## [13] "contrasts"          "converged"          "residuals"
## [16] "noObs"              "mu.fv"              "mu.lp"
## [19] "mu.wv"              "mu.wt"              "mu.link"
## [22] "mu.terms"           "mu.x"               "mu.qr"
## [25] "mu.coefficients"    "mu.offset"          "mu.xlevels"
## [28] "mu.formula"         "mu.df"              "mu.nl.df"
## [31] "mu.s"               "mu.var"             "mu.coefSmo"
## [34] "mu.lambda"          "mu.pen"             "df.fit"
## [37] "pen"                "df.residual"        "sigma.fv"
```

```
## [40] "sigma.lp"           "sigma.wv"           "sigma.wt"
## [43] "sigma.link"         "sigma.terms"        "sigma.x"
## [46] "sigma.qr"           "sigma.coefficients" "sigma.offset"
## [49] "sigma.xlevels"      "sigma.formula"      "sigma.df"
## [52] "sigma.nl.df"        "sigma.s"            "sigma.var"
## [55] "sigma.coefSmo"      "sigma.lambda"       "sigma.pen"
## [58] "nu.fv"              "nu.lp"              "nu.wv"
## [61] "nu.wt"              "nu.link"            "nu.terms"
## [64] "nu.x"               "nu.qr"              "nu.coefficients"
## [67] "nu.offset"          "nu.formula"         "nu.df"
## [70] "nu.nl.df"           "nu.pen"             "P.deviance"
## [73] "aic"                "sbc"
```

Por exemplo...

```
h$family# família usada
```

```
## [1] "TF"       "t Family"
```

```
h$parameters# parâmetros
```

```
## [1] "mu"    "sigma" "nu"
```

```
h$G.deviance# deviance
```

```
## [1] 4777.519
```

```
h$N# comprimento da variáveis resposta
```

```
## [1] 610
```

```
h$noObs# número de observações (coincide com h$N na ausência de pesos)
```

```
## [1] 610
```

```
h$iter# número de iterações externas
```

```
## [1] 4
```

Métodos e funções para objetos `gamlss`

Há vários métodos e funções que podem ser aplicados a um objeto `gamlss`. Um método, no `R`, é uma função genérica que pode ser usada para mostrar informações de objeto de uma classe específica. Essas funções são exploradas ao longo da disciplina.

Alguns exemplos são:

- `AIC()`: para extrair o AIC generalizado
- `coef()`: para extrair os coeficientes lineares de qualquer parâmetro da distribuição
- `confint()`: para extrair intervalos de confiança
- `deviance()`: para extrair a *deviance* global
- `fitted()`: para extrair os valores ajustados
- `lp()`: para extrair o preditor linear para um parâmetro da distribuição
- `plot()`: para plotar o diagnóstico de resíduos
- `predict()`: para fazer predições com base em um novo conjunto de dados
- `print()`: para mostrar (imprimir) um objeto `gamlss`
- `residuals()`: para extrair os resíduos quantílicos normalizados de um objeto `gamlss` ajustado
- `summary()`: para sumarizar o ajuste de um objeto `gamlss`
- `vcov()`: para extrair a matriz de variância-covariância das estimativas de beta