Regresión cuantílica: modelos GAMLSS

Joaquín Amat Rodrigo j.amatrodrigo@gmail.com

Mayo, 2020

Tabla de contenidos

Versión PDF: Github

Más sobre ciencia de datos: cienciadedatos.net o joaquinamatrodrigo.github.io

Introducción

La predicción de una variable Y en función de uno o varios predictores X es un problema de aprendizaje supervisado que puede resolverse con múltiples métodos de *Machine Learning* y aprendizaje estadístico. Algunos de ellos consideran que la relación entre Y y X es únicamente lineal (regresión lineal, GLM), mientras que otros permiten incorporar relaciones no lineales o incluso interacciones entre predictores (SVM, Random Forest, Boosting).

De una forma u otra, todos ellos tratan de inferir la relación entre X e Y para obtener información sobre la distribución condicional de la variable respuesta en función de variables predictoras. Sin embargo, la gran mayoría de los modelos de regresión únicamente modelan la media de la variable respuesta $E(Y|X = x)$, asumiendo que el resto de características de la distribución (dispersión, asimetría...) son constantes. Esto supone una limitación importante a la hora de modelar distribuciones complejas, sobretodo si se pretende predecir intervalos de confianza o cuantiles.

GAMLSS

En este documento se hace un resumen muy breve de los modelos GAMLSS. Para conocer más sobre este tipo de modelos consultar Introducción a los modelos GAMLSS.

Los modelos aditivos generalizados para posición, escala y forma **GAMLSS** (*Generalized Additive Models for Location, Scale and Shape*), son modelos de regresión semi-paramétricos. Paramétricos en cuanto a que requieren asumir que la variable respuesta sigue una determinada distribución paramétrica (normal, beta, *gamma*...) y *semi* porque los parámetros de esta distribución pueden ser modelados, cada uno de forma independiente, siguiendo funciones no paramétricas (lineales, aditivas o no lineales). Esta versatilidad hace de los *GAMLSS* una herramienta adecuada para modelar variables que siguen todo un abanico de distribuciones (no normales, asimétricas, con varianza no constante...).

Los modelos *GAMLSS* asumen que la variable respuesta tiene una función de densidad definida por hasta 4 parámetros (μ, σ, ν, τ) que determinan su posición (p.ej. media), escala (p.ej. desviación estándar) y forma (p.ej. *skewness* y *kurtosis*), y que cada uno de ellos puede variar independientemente de los otros en función de los predictores. Estos modelos aprenden por lo tanto hasta 4 funciones, donde cada una establece la relación entre uno de los parámetros y las variables predictoras. Por ejemplo, en la regresión gaussiana, la variable respuesta depende de dos parámetros: la media μ y de la desviación típica σ. En lugar de asumir que σ es constante (como hacen los modelos *LM* y *GAM*), los modelos *GAMLSS* modelan ambos parámetros en función de las variables predictoras.

$$\mathbf{Y} \sim N(\mu, \sigma)$$

$$\mu = \eta_\mu = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p)$$

$$\sigma = \eta_\sigma = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_p(x_p)$$

donde $f_i(x_i)$ representa el efecto de cada predictor sobre el parámetro modelado y pueden ser funciones tanto lineales como no lineales (*smooth*).

Los modelos *GAMLSS* son capaces de caracterizar la distribución completa, permitiendo generar intervalos probabilísticos y predicción de cuantiles, sin tener que asumir que la varianza es constante ni que las relaciones son únicamente lineales.

Ejemplo 1

Véase el siguiente ejemplo simulado (y muy simplificado) sobre la evolución del consumo eléctrico de todas las casas de una ciudad en función de la hora del día. Ver *Anexo*[1] con el código empleado para la simulación.

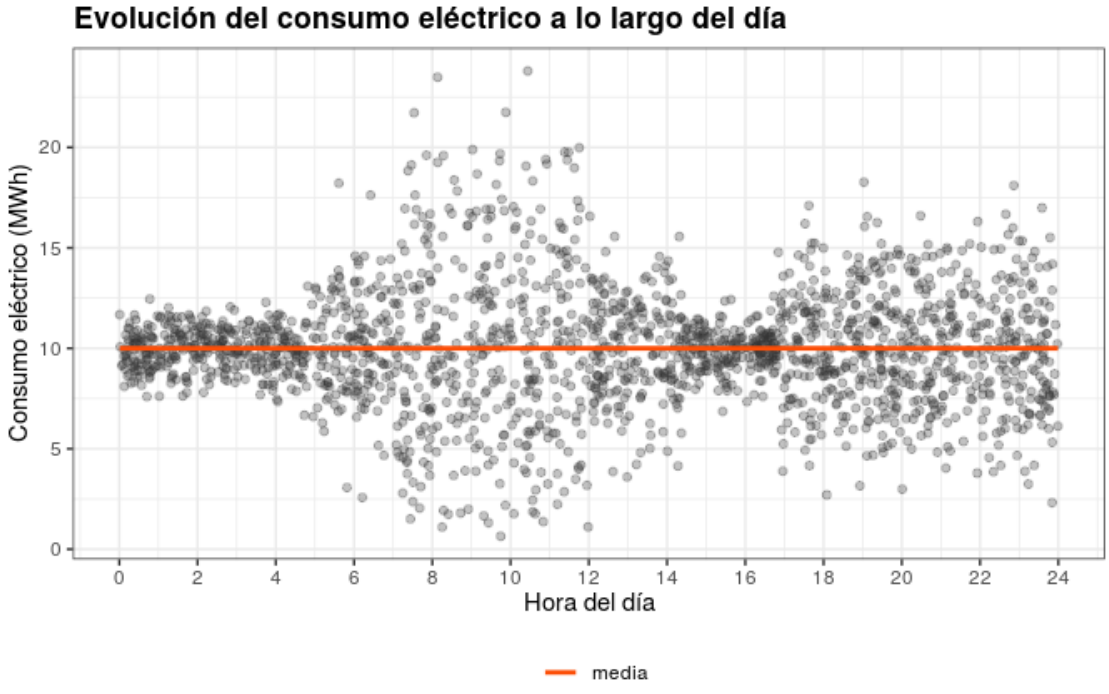
Datos

Ver *Anexo*[1] para conocer más detalles de la simulación.

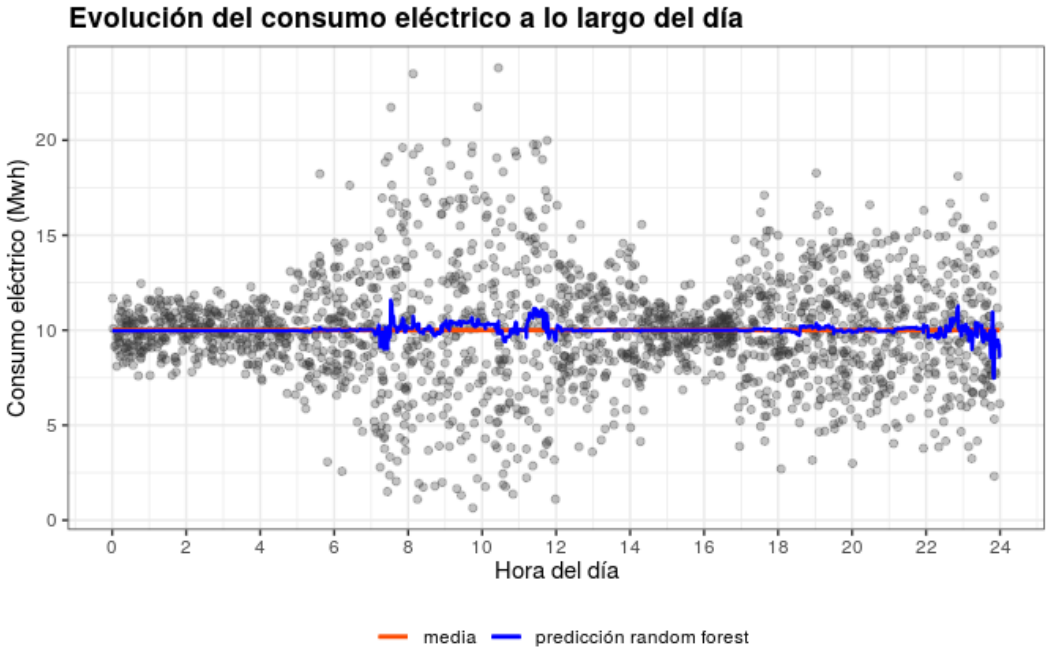
```
library(tidyverse)
# Simulación distribución no uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(12345)
n <- 2000
x <- runif(min = 0, max = 24, n = n) %>% sort()
y <- rnorm(
        n,
        mean = 10,
        sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
             1.5*(12 < x & x < 14.4) + 2*(x > 16.8)
     )
# Cálculo del cuantil 0.1 y 0.9 para cada posición de x simulada.
cuantil_10 <- qnorm(
                p = 0.1,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(12 < x & x < 14.4) + 2*(x > 16.8)
              )
cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(12 < x & x < 14.4) + 2*(x > 16.8)
              )
datos <- data.frame(consumo = y, hora = x, cuantil_10, cuantil_90)
# No puede haber consumos negativos
datos <- datos %>%
         filter(consumo >= 0)
datos <- datos %>%
         mutate(dentro_intervalo = ifelse(
                                     consumo > cuantil_10 & consumo < cuantil_90,
                                     TRUE, FALSE
                                   )
        )
```

```
p <- ggplot() +
      geom_point(
        data = datos,
        aes(x = hora, y = consumo),
        alpha = 0.3,
        color = "gray20") +
      geom_line(
        data = datos,
        aes(x = hora , y = 10, color = "media"),
        linetype = "solid",
        size  = 1) +
      scale_color_manual(name = "",
                         breaks = c("media"),
                         values = c("media" = "#FC4E07")) +
      labs(title = "Evolución del consumo eléctrico a lo largo del día",
           x = "Hora del día",
           y = "Consumo eléctrico (MWh)") +
      theme_bw() +
      theme(legend.position = "bottom",
            plot.title = element_text(face = "bold"))

p <- p +
     scale_x_continuous(breaks = seq(0, 24, 2),
                        labels = seq(0, 24, 2))
p
```



La media del consumo eléctrico es la misma durante todo el día, $\overline{consumo} = 10Mwh$, sin embargo, su dispersión no es constante (heterocedasticidad). Véase el resultado de predecir el consumo medio en función de la hora del día con un modelo Random Forest.

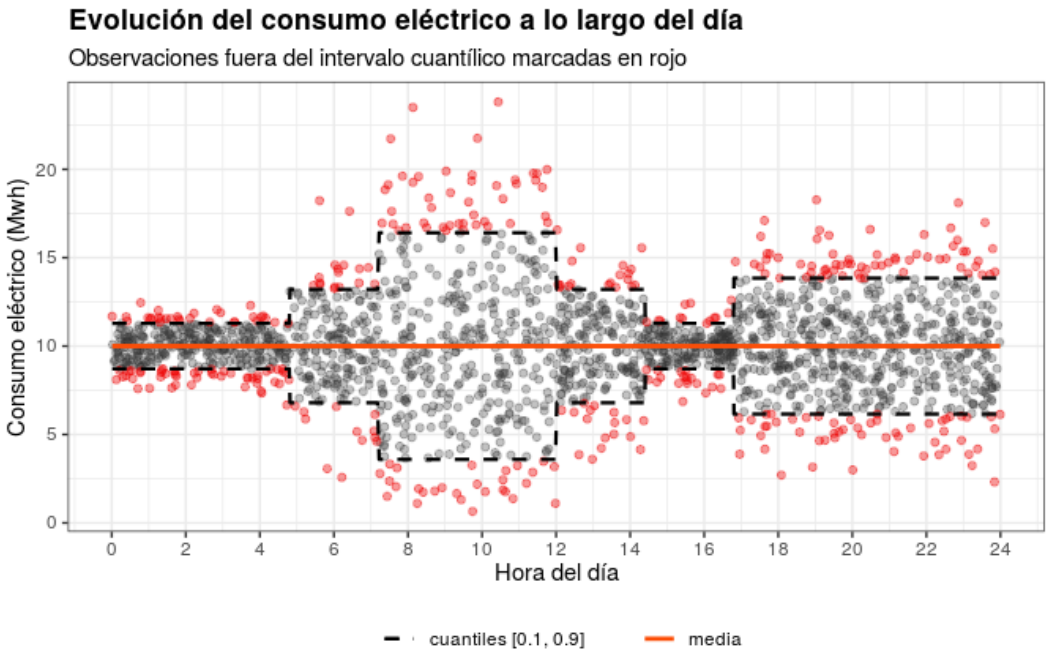


El valor predicho es muy próximo a la media real, es decir, el modelo es bueno prediciendo el consumo medio esperado. Ahora, imagínese que la compañía encargada de suministrar la electricidad debe de ser capaz de provisionar, en un momento dado, con hasta un 50% de electricidad extra respecto al promedio. Esto significa un máximo de 15 Mwh. Estar preparado para suministrar este extra de energía implica gastos de personal y maquinaría, por lo que la compañía se pregunta si es necesario estar preparado para producir tal cantidad durante todo el día, o si, por lo contrario, podría evitarse durante algunas horas, ahorrando así gastos.

Un modelo que predice únicamente el promedio no permite responder a esta pregunta, ya que tanto para las 2h de la mañana como para las 8h, el consumo promedio predicho es en torno a 10 Mwh, sin embargo, la probabilidad de que se alcancen consumos de 15 Mwh a las 2h es prácticamente nula mientras que esto ocurra a las 8h sí es razonable.

Una forma de describir la dispersión de una variable es el uso de cuantiles. El cuantil de orden τ $(0 < \tau < 1)$ de una variable continua Y es el valor y_τ tal que, una proporción τ de valores de la población, es menor o igual que dicho valor $(Prob(Y \leq y_\tau) = \tau)$. Por ejemplo, el cuantil de orden 0.36 deja un 36% de valores por debajo y el cuantil de orden 0.50 el 50% (se corresponde con la mediana de la distribución).

Dado que los datos se han simulado empleando distribuciones normales, se conoce el valor de los cuantiles teóricos para cada X. Se muestra de nuevo el mismo gráfico pero esta vez añadiendo los cuantiles 0.1 y 0.9.



Si como resultado del modelo, además de la predicción de la media, se predice también el valor de los cuantiles, se dispone de una caracterización mayor de la distribución de la variable respuesta, y con ello se puede responder a más preguntas. Por ejemplo, en el caso de la energía, se tendría cierta seguridad al decir que, durante los intervalos de 0h a 4h y de 12h a 14h, es poco probable que se alcancen consumos de 15 Mwh.

Otros casos en los que conocer la distribución de cuantiles puede ser útil son:

- Identificación de regiones en las que la variable respuesta tiene mayor dispersión en torno a su media.
- Entrenar modelos que predicen la mediana (cuantil 0.5) en lugar de la media. Estos modelos son más robustos frente a *outliers*.
- Detectar anomalías, identificando aquellas observaciones que están fuera de un determinado intervalo cuantílico.

En este ejemplo se describe cómo los modelos *GAMLSS* permiten caracterizar toda la distribución y por lo tanto predecir los cuantiles.

Nota: Otras aproximaciones que tratan de resolver este mismo problema son Quantile Regression Forest, Distributional Regression Forest: Random Forest probabilístico y Gradient Boosting Quantile Regression.

Modelo

El primer paso para ajustar un modelo *GAMLSS* es identificar qué tipo de distribución paramétrica sigue la variable respuesta. En este caso, como los datos han sido simulados, se sabe que siguen una distribución normal. Aunque para este primer ejemplo ilustrativo se asume como cierto, en la práctica, esta información se desconoce, por lo que es necesario un primer estudio de la distribución (ver ejemplo 2).

Con la función `gamlss()` del paquete `gamlss` se realiza el entrenamiento del modelo. Es necesario indicar una fórmula que defina la relación entre los predictores y cada uno de los parámetros de la distribución. Es en esta fórmula donde se indica también si el predictor se transforma empleando alguna función no lineal de tipo *smooth*. Para este ejemplo se ajusta un modelo *GAMLSS* con las siguientes características:

- Distribución normal `NO` para la variable respuesta.
- Tanto *mu* como *sigma* se modelan en función de la variable `hora`.
- En ambos casos se emplean *P-splines* para permitir una relación no lineal con entre el predictor y la variable respuesta.

```
library(gamlss)
modelo <- gamlss(
            formula = consumo~ pb(hora),
            sigma.formula = ~ pb(hora),
            family  = NO,
            data    = datos,
            control = gamlss.control(trace = FALSE)
          )
summary(modelo)
```

```
## ******************************************************************
## Family:  c("NO", "Normal")
##
## Call:  gamlss(formula = consumo ~ pb(hora), sigma.formula = ~pb(hora),
##     family = NO, data = datos, control = gamlss.control(trace = FALSE))
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.9549493  0.0591163 168.396   <2e-16 ***
## pb(hora)    -0.0008836  0.0053652  -0.165    0.869
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.481390   0.032502   14.81   <2e-16 ***
## pb(hora)    0.028841   0.002298   12.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
## ii) Std. Error for the linear terms maybe are not accurate.
## ------------------------------------------------------------------
## No. of observations in the fit:  1990
## Degrees of Freedom for the fit:  18.02599
##       Residual Deg. of Freedom:  1971.974
##                       at cycle:  4
##
## Global Deviance:     8980.308
##             AIC:     9016.36
##             SBC:     9117.232
## ******************************************************************
```

Predicción

La predicción de un modelo *GAMLSS* es el valor estimado de los parámetros de la distribución (en este caso μ y σ) para cada valor del predictor `hora`.

```
# Se predice todo el rango de hora para representar los cuantiles
grid_predictor <- seq(0, 24, length.out = 2500)

predicciones <- predictAll(
                  modelo,
                  newdata = data.frame(hora = grid_predictor),
                  type = "response"
                )

## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)
## new prediction
## New way of prediction in pb()  (starting from GAMLSS version 5.0-3)

predicciones <- as.data.frame(predicciones)
predicciones <- bind_cols(data.frame(hora = grid_predictor), predicciones)
predicciones %>% head()
```

```
##          hora       mu     sigma
## 1 0.000000000 9.954907 0.9048709
## 2 0.009603842 9.954899 0.9065713
## 3 0.019207683 9.954891 0.9082748
## 4 0.028811525 9.954882 0.9099789
## 5 0.038415366 9.954874 0.9116814
## 6 0.048019208 9.954865 0.9133818
```

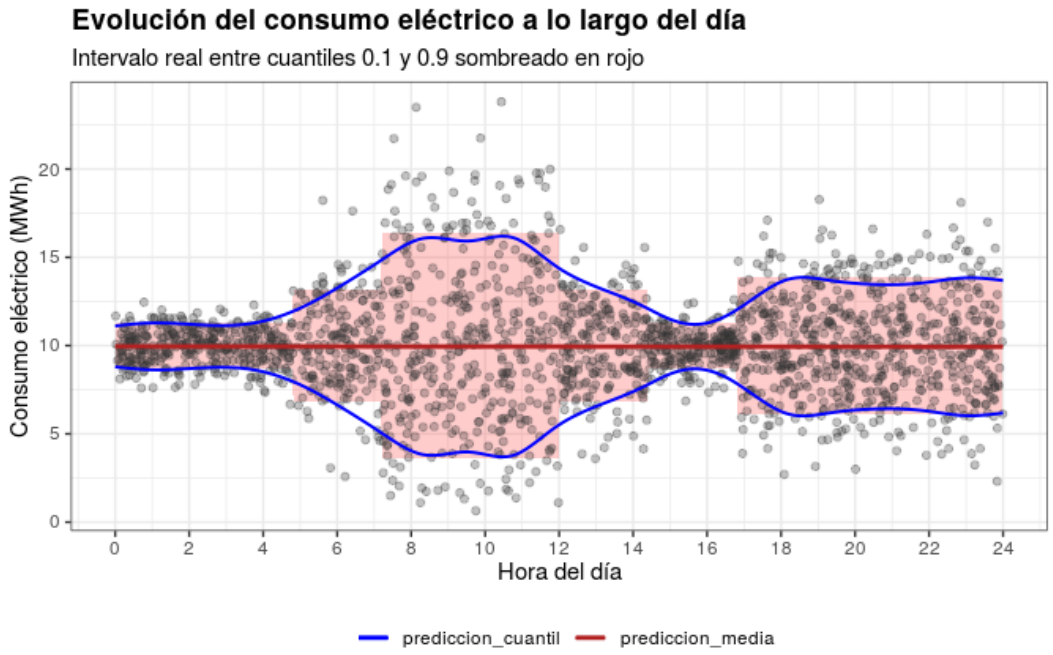
Una vez predichos los parámetros que caracterizan a la distribución en función del predictor, se puede calcular la probabilidad de cada observación o el intervalo que acumula un determinado porcentaje de probabilidad (intervalo cuantílico). Todas las distribuciones de los paquetes `gamlss` y `gamlss.dist` disponen de funciones `d`, `p`, `q` y `r` para calcular probabilidad, densidad, cuantiles y generar valores aleatorios.

```
# Cálculo de los cuantiles teóricos para establecer el intervalo central que
acumula
# un 90% de probabilidad empleando los parámetros predichos.
predicciones <- predicciones %>%
                mutate(
                cuantil_10_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma),
                                    .f = function(mu, sigma) {qNO(p = 0.1, mu, sigma)}
                                  ),
                cuantil_90_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma),
                                    .f = function(mu, sigma) {qNO(p = 0.9, mu, sigma)}
                                  )
                )

p <- ggplot() +
      geom_ribbon(
        data = datos,
        aes(x = hora, ymin = cuantil_10, ymax = cuantil_90),
        fill = "red",
        alpha = 0.2) +
      geom_point(
        data = datos,
        aes(x = hora, y = consumo),
        alpha = 0.3,
        color = "gray20") +
      geom_line(
        data = predicciones,
        aes(x = hora, y = cuantil_10_pred, color = "prediccion_cuantil"),
        size = 0.7) +
      geom_line(
        data = predicciones,
        aes(x = hora, y = cuantil_90_pred, color = "prediccion_cuantil"),
        size = 0.7) +
      geom_line(
        data = predicciones,
        aes(x = hora, y = mu, color = "prediccion_media"),
        size = 1) +
```

```
      scale_color_manual(name = "",
                         breaks = c("prediccion_cuantil", "prediccion_media"),
                         values = c("prediccion_cuantil" = "blue",
                                    "prediccion_media"   = "firebrick")) +
      labs(title = "Evolución del consumo eléctrico a lo largo del día",
           subtitle = "Intervalo real entre cuantiles 0.1 y 0.9 sombreado en rojo",
           x = "Hora del día",
           y = "Consumo eléctrico (MWh)") +
      theme_bw() +
      theme(legend.position = "bottom",
            plot.title = element_text(face = "bold"))

p <- p +
     scale_x_continuous(breaks = seq(0, 24, 2),
                        labels = seq(0, 24, 2))
p
```



Si por ejemplo, se desea conocer la probabilidad de que a las 8h el consumo supere los 15 *MWh*, primero se predicen los parámetros de la distribución a para ($hora = 8$) y después se calcula la probabilidad de $consumo \geq 15$ con su función de distribución.

```
prediccion <- predictAll(
                modelo,
                newdata = data.frame(hora = 8),
                type = "response"
              )

prediccion
```

```
## $mu
## [1] 9.947882
##
## $sigma
## [1] 4.607569
##
## attr(,"family")
## [1] "NO"     "Normal"
```

```
# Se calcula la probabilidad
probabilidad_consumo <- pNO(
                         q     = 15,
                         mu    = prediccion$mu,
                         sigma = prediccion$sigma,
                         lower.tail = FALSE
                        )
probabilidad_consumo
```

```
## [1] 0.1364339
```

Acorde al modelo, la probabilidad de que a las 8h el consumo sea igual o superior a 15 *Mwh* es del 13.6%.

Ejemplo 2

Datos

El set de datos `dbbmi` del paquete `gamlss.data` contiene información sobre la edad (*Age*) e índice de masa corporal (*BMI*) de 7294 jóvenes holandeses de entre 0 y 20 años. El objetivo es obtener un modelo capaz de predecir cuantiles del índice de masa coprporal en función de la edad. Estos cuantiles son uno de los estándares empleados para detectar casos anómalos que pueden requerir atención médica.

```
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(tidyverse)
library(ggpubr)
library(ggforce)

data(dbbmi)
datos <- dbbmi
head(datos)
```

```
##    age      bmi
## 1 0.03 13.23529
## 2 0.04 12.43877
## 3 0.04 14.54177
## 4 0.04 11.77395
## 5 0.04 15.32561
## 6 0.05 13.21439
```

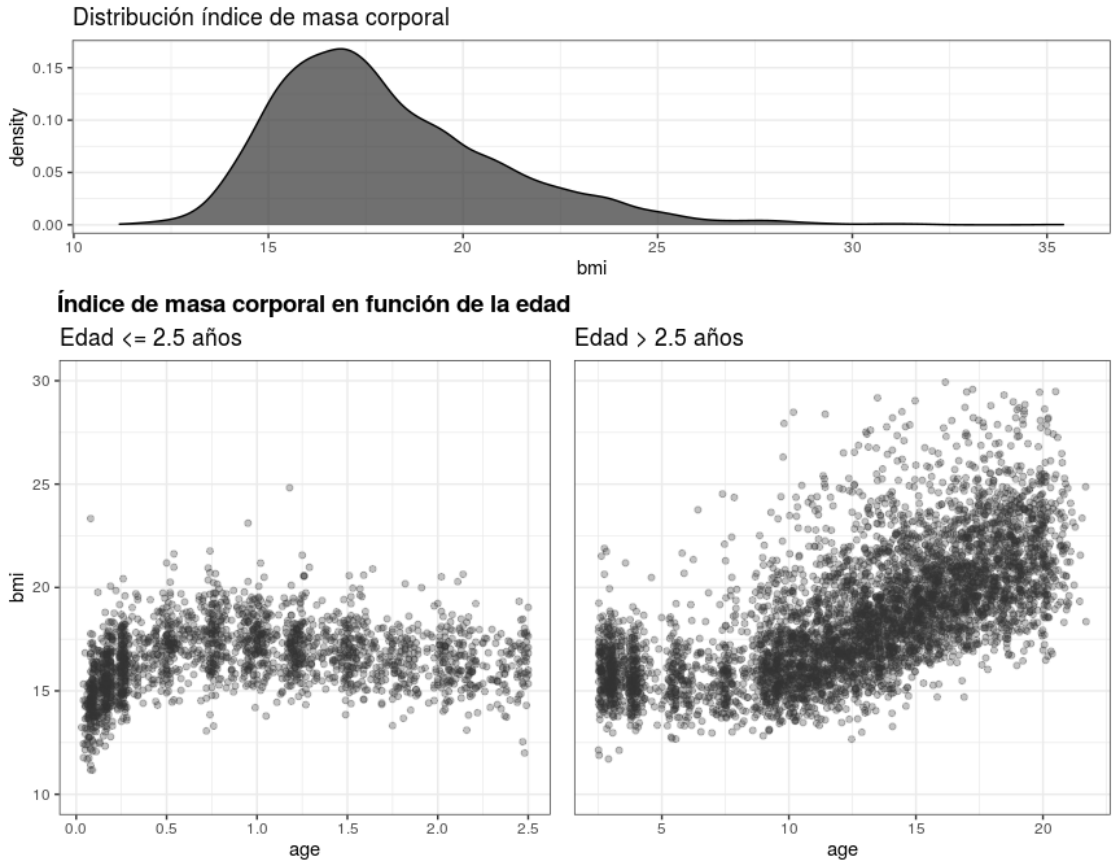
```
p1 <- ggplot(
        data = datos %>% filter(age <= 2.5),
        aes(x = age, y = bmi)) +
      geom_point(alpha = 0.3, color = "gray20") +
      lims(y = c(10,30)) +
      labs(title = "Edad <= 2.5 años") +
      theme_bw()

p2 <- ggplot(
        data = datos %>% filter(age > 2.5),
        aes(x = age, y = bmi)) +
      lims(y = c(10,30)) +
      geom_point(alpha = 0.3, color = "gray20") +
      labs(title = "Edad > 2.5 años") +
      theme_bw() +
      theme(
```

```
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank())

p3 <- ggplot(data = datos, aes(x = bmi)) +
      geom_density(alpha = 0.7, fill = "gray20") +
      labs(title = "Distribución índice de masa corporal") +
      theme_bw()

ggarrange(
  p3,
  ggarrange(p1, p2, align = "h")  %>%
    ggpubr::annotate_figure(
    top = text_grob("Índice de masa corporal en función de la edad",
                    color = "Black",
                    face  = "bold",
                    size  = 14,
                    x     = 0.28)
    ),
  nrow = 2,
  heights = c(1,2)
)
```



Esta distribución muestra tres características importantes que hay que tener en cuenta a la hora de modelarla:

- La relación entre la edad y el índice de masa corporal no es lineal ni constante. Tiene una relación positiva notable hasta los 0.25 años, después se estabiliza hasta los 10 años y vuelve a ascender notablemente de los 10 a los 20 años.
- La varianza es heterogénea (heterocedasticidad), siendo esta menor en edades bajas mayor en edades altas.
- La distribución de la variable respuesta no es de tipo normal, muestra asimetría y una cola positiva.

Dadas estas características, se necesita un modelo que:

- Sea capaz de aprender relaciones no lineales.
- Sea capaz de modelar explícitamente la varianza en función de los predictores, ya que esta no es constante.
- Sea capaz de aprender distribuciones asimétricas con una marcada cola positiva.

Distribución

La función `fitDist()` ajusta toda las distribuciones paramétricas disponibles de una determinada familia, y las compara acorde al *GAIC* (*generalized Akaike information criterion*). La familia de distribuciones se especifica con el argumento `type` y puede ser: `"realAll"`, `"realline"`, `"realplus"`, `"real0to1"`, `"counts"` y `"binom"`. Para conocer más información sobre el ajuste de distribuciones consultar Introducción a los modelos GAMLSS.

```
distribuciones <- fitDist(
                    y = datos$bmi,
                    k = log(length(datos$bmi)),
                    type  = "realplus",
                    trace = FALSE,
                    try.gamlss = TRUE,
                    parallel = "multicore",
                    ncpus = 3L
                  )

distribuciones$fits %>%
  enframe(name = "distribucion", value = "GAIC") %>%
  arrange(GAIC)
```

```
## # A tibble: 23 x 2
##    distribucion   GAIC
##    <chr>         <dbl>
##  1 exGAUS       35001.
##  2 GG           35003.
##  3 GB2          35010.
##  4 BCPEo        35013.
##  5 BCPE         35013.
##  6 BCCGo        35013.
##  7 BCCG         35013.
##  8 BCT          35022.
##  9 BCTo         35022.
## 10 IGAMMA       35215.
## # … with 13 more rows
```

El objeto devuelto por `fitDist()` almacena la mejor de entre todas las distribuciones probadas.

```
summary(distribuciones)
```

```
## ******************************************************************
## Family:  c("exGAUS", "ex-Gaussian")
##
## Call:  gamlssML(formula = y, family = DIST[i], parallel = "multicore",
##     ncpus = 3L, data = sys.parent())
##
## Fitting method: "nlminb"
##
##
## Coefficient(s):
##             Estimate  Std. Error  t value   Pr(>|t|)
## eta.mu    15.2952362   0.0420600 363.6531 < 2.22e-16 ***
## eta.sigma  0.2787316   0.0245323  11.3618 < 2.22e-16 ***
## eta.nu     1.0048731   0.0184961  54.3290 < 2.22e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Degrees of Freedom for the fit: 3 Residual Deg. of Freedom   7291
## Global Deviance:     34974.5
##             AIC:     34980.5
##             SBC:     35001.2
```

Acorde al criterio *GAIC*, la distribución que mejor describe los datos es la *Exponentially modified Gaussian (ex-Gaussian)*. Es importante tener en cuenta que esta elección no es infalible, siempre hay que ponerla en contexto con el conocimiento que se tenga sobre la variable que se está intentando modelar. Por ejemplo, los autores de los modelos *GAMLSS*, Rigby y Stasinopoulos, proponen que el índice de masa corporal sigue una distribución *Box-Cox t-distribution* BCT(μ, σ, ν, τ).

Modelo

```
library(gamlss)
modelo <- gamlss(
            formula       = bmi ~ pb(age),
            sigma.formula = ~ pb(age),
            nu.formula    = ~ pb(age),
            family  = exGAUS,
            data    = datos,
            control = gamlss.control(trace = FALSE)
          )
summary(modelo)
```

```
## ******************************************************************
## Family:  c("exGAUS", "ex-Gaussian")
##
## Call:  gamlss(formula = bmi ~ pb(age), sigma.formula = ~pb(age),
##     nu.formula = ~pb(age), family = exGAUS, data = datos,
##     control = gamlss.control(trace = FALSE))
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.114532   0.046034  328.33   <2e-16 ***
## pb(age)      0.083005   0.004904   16.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06699    0.02516   2.663  0.00777 **
## pb(age)      0.01287    0.00251   5.130 2.97e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Nu link function:  log
## Nu Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.199469   0.052653  -3.788 0.000153 ***
## pb(age)      0.063712   0.004141  15.385  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
## ii) Std. Error for the linear terms maybe are not accurate.
## ------------------------------------------------------------------
## No. of observations in the fit:  7294
## Degrees of Freedom for the fit:  32.25977
##       Residual Deg. of Freedom:  7261.74
##                       at cycle:  20
##
## Global Deviance:     29143.7
##             AIC:     29208.22
##             SBC:     29430.65
## ******************************************************************
```

Predicción

La predicción del modelo es el valor estimado de los parámetros de la distribución (en este caso μ, σ y ν) para cada valor de edad `age`.

```
# Se predice todo el rango de age para representar los cuantiles
grid_predictor <- seq(0, 20, length.out = 1000)

predicciones <- predictAll(
                  modelo,
                  newdata = data.frame(age = grid_predictor),
                  type = "response"
                )

predicciones <- as.data.frame(predicciones)
predicciones <- bind_cols(data.frame(age = grid_predictor), predicciones)
predicciones %>% head()
```

```
##          age       mu    sigma        nu
## 1 0.00000000 13.04813 1.131026 0.6738751
## 2 0.02002002 13.28288 1.130088 0.6750617
## 3 0.04004004 13.51708 1.129151 0.6762503
## 4 0.06006006 13.74291 1.128215 0.6774413
## 5 0.08008008 13.95852 1.127279 0.6786345
## 6 0.10010010 14.16413 1.126344 0.6798301
```

Una vez predichos los parámetros que caracterizan a la distribución en función del predictor, se puede calcular la probabilidad de cada observación o el intervalo que acumula un

determinado porcentaje de probabilidad (intervalo cuantílico). Todas las distribuciones de los paquetes `gamlss` y `gamlss.dist` disponen de funciones `d`, `p`, `q` y `r` para calcular probabilidad, densidad, cuantiles y generar valores aleatorios.

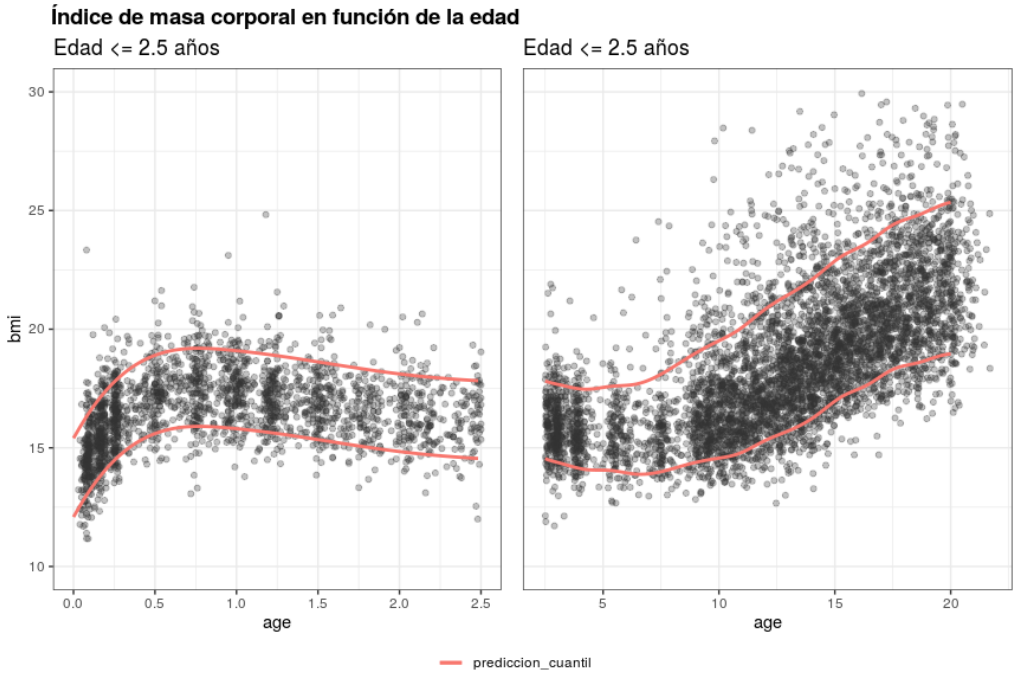
```
# Cálculo de los cuantiles teóricos para establecer el intervalo central que
acumula
# un 90% de probabilidad empleando los parámetros predichos.
predicciones <- predicciones %>%
                mutate(
                cuantil_10_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma, nu = nu),
                                    .f = function(mu, sigma, nu) {
                                            qexGAUS(p = 0.1, mu, sigma, nu)
                                        }
                                  ),
                cuantil_90_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma, nu = nu),
                                    .f = function(mu, sigma, nu) {
                                            qexGAUS(p = 0.9, mu, sigma, nu)
                                        }
                                  )
                )

p1 <- ggplot() +
      geom_point(
        data = datos %>% filter(age <= 2.5),
        aes(x = age, y = bmi),
        alpha = 0.3,
        color = "gray20") +
      geom_line(
        data = predicciones %>% filter(age <= 2.5),
        aes(x = age, y = cuantil_10_pred, color = "prediccion_cuantil"),
        size = 1) +
      geom_line(
        data = predicciones %>% filter(age <= 2.5),
        aes(x = age, y = cuantil_90_pred, color = "prediccion_cuantil"),
        size = 1) +
      scale_color_manual(name = "",
                         breaks = c("prediccion_cuantil"),
                         values = c("prediccion_cuantil" = "#F8766D")) +
      lims(y = c(10,30)) +
      labs(title = "Edad <= 2.5 años") +
      theme_bw()

p2 <- ggplot() +
      geom_point(
        data = datos %>% filter(age > 2.5),
        aes(x = age, y = bmi),
        alpha = 0.3,
```

```
        color = "gray20") +
      geom_line(
        data = predicciones %>% filter(age > 2.5),
        aes(x = age, y = cuantil_10_pred, color = "prediccion_cuantil"),
        size = 1) +
      geom_line(
        data = predicciones %>% filter(age > 2.5),
        aes(x = age, y = cuantil_90_pred, color = "prediccion_cuantil"),
        size = 1) +
      scale_color_manual(name = "",
                         breaks = c("prediccion_cuantil"),
                         values = c("prediccion_cuantil" = "#F8766D")) +
      lims(y = c(10,30)) +
      labs(title = "Edad <= 2.5 años") +
      theme_bw() +
      theme(
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank())

ggarrange(p1, p2, nrow = 1, ncol = 2, align = "h",
          common.legend = TRUE, legend = "bottom")  %>%
    ggpubr::annotate_figure(
    top = text_grob("Índice de masa corporal en función de la edad",
                    color = "Black",
                    face  = "bold",
                    size  = 14,
                    x     = 0.28)
)
```



Si por ejemplo, se desea conocer la probabilidad de que un niño de 10 años tenga un índice de masa corporal igual o superior a 20, primero se predicen los parámetros de la distribución a para $age = 10$ y después se calcula la probabilidad de $bmi \geq 20$ con su función de distribución.

```
prediccion <- predictAll(
                modelo,
                newdata = data.frame(age = 10),
                type = "response"
              )

prediccion
```

```
## $mu
## [1] 14.87463
##
## $sigma
## [1] 0.9554561
##
## $nu
## [1] 1.910097
##
## attr(,"family")
## [1] "exGAUS"      "ex-Gaussian"
```

```
# Se calcula la probabilidad
probabilidad_bmi <- pexGAUS(
                      q     = 20,
                      mu    = prediccion$mu,
                      sigma = prediccion$sigma,
                      nu    = prediccion$nu,
                      lower.tail = FALSE
                    )
probabilidad_bmi
```

```
## [1] 0.07744431
```

Acorde al modelo, la probabilidad de que un niño de 10 años tenga un índice de masa corporal igual o superior a 20 es del 7.7%.

Ejemplo 3

Datos

El set de datos `db` del paquete `gamlss.data` contiene información sobre la edad (*age*) y la circunferencia de la cabeza (*head*) de 7294 jóvenes holandeses de entre 0 y 20 años. El objetivo es obtener un modelo capaz de predecir cuantiles del índice de masa corporal en función de la edad. Estos cuantiles son uno de los estándares empleados para detectar casos anómalos que pueden requerir atención médica.

```
library(gamlss)
library(gamlss.dist)
library(gamlss.data)
library(tidyverse)
library(ggpubr)
library(ggforce)

data(db)
datos <- db
head(datos)
```

```
##   head  age
## 1 33.6 0.03
## 2 33.6 0.04
## 3 33.7 0.04
## 4 35.0 0.04
## 5 36.1 0.04
## 6 36.6 0.05
```

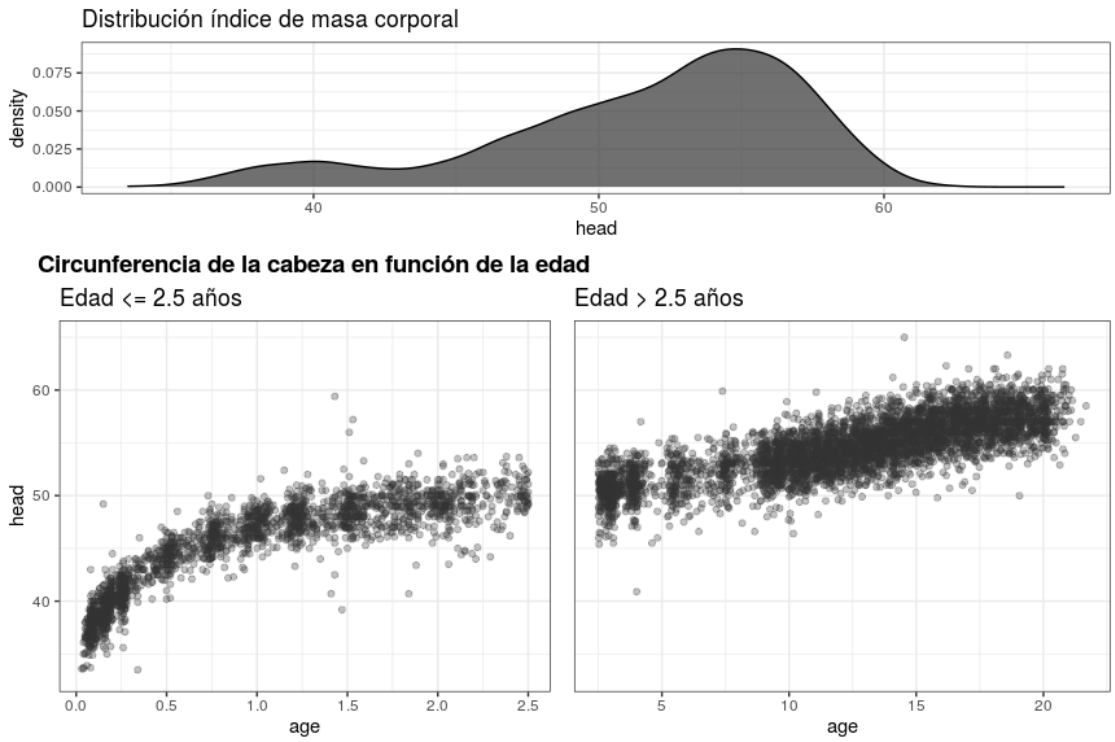
```
p1 <- ggplot(
        data = datos %>% filter(age <= 2.5),
        aes(x = age, y = head)) +
      geom_point(alpha = 0.3, color = "gray20") +
      lims(y = c(33, 65)) +
      labs(title = "Edad <= 2.5 años") +
      theme_bw()

p2 <- ggplot(
        data = datos %>% filter(age > 2.5),
        aes(x = age, y = head)) +
      lims(y = c(33, 65)) +
      geom_point(alpha = 0.3, color = "gray20") +
      labs(title = "Edad > 2.5 años") +
      theme_bw() +
      theme(
```

```
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank())

p3 <- ggplot(data = datos, aes(x = head)) +
      geom_density(alpha = 0.7, fill = "gray20") +
      labs(title = "Distribución índice de masa corporal") +
      theme_bw()

ggarrange(
  p3,
  ggarrange(p1, p2, align = "h")  %>%
    ggpubr::annotate_figure(
    top = text_grob("Circunferencia de la cabeza en función de la edad",
                    color = "Black",
                    face  = "bold",
                    size  = 14,
                    x     = 0.28)
    ),
  nrow = 2,
  heights = c(1,2)
)
```



Esta distribución muestra tres características importantes que hay que tener en cuenta a la hora de modelarla:

- La relación entre la edad y la circunferencia de la cabeza no es lineal. Tiene una relación positiva notable hasta los 0.5 años, después se estabiliza hasta los 10 años y vuelve a ascender ligeramente de los 10 a los 20 años.
- La varianza es heterogénea (heterocedasticidad), siendo aumenta con la edad.
- La distribución de la variable respuesta es muy asimétrica.

Distribución

La función `fitDist()` ajusta toda las distribuciones paramétricas disponibles de una determinada familia, y las compara acorde al *GAIC* (*generalized Akaike information criterion*). La familia de distribuciones se especifica con el argumento `type` y puede ser: `"realAll"`, `"realline"`, `"realplus"`, `"real0to1"`, `"counts"` y `"binom"`. Para conocer más información sobre el ajuste de distribuciones consultar Introducción a los modelos GAMLSS.

```
distribuciones <- fitDist(
                    y = datos$head,
                    k = 3,
                    type  = "realplus",
                    trace = FALSE,
                    try.gamlss = TRUE,
                    parallel = "multicore",
                    ncpus = 3L
                  )

distribuciones$fits %>%
  enframe(name = "distribucion", value = "GAIC") %>%
  arrange(GAIC)
```

```
## # A tibble: 23 x 2
##    distribucion   GAIC
##    <chr>         <dbl>
##  1 GB2          42463.
##  2 BCPEo        42513.
##  3 BCPE         42513.
##  4 GG           42516.
##  5 BCCGo        42517.
##  6 BCCG         42517.
##  7 BCTo         42520.
##  8 BCT          42520.
##  9 WEI3         42776.
## 10 WEI          42776.
## # … with 13 more rows
```

Acorde al criterio *GAIC*, la distribución que mejor describe los datos es la *Generalized beta 2*. Sin embargo, entre las primeras destacan las distribuciones de la familia *Box-Cox* (*BCPEo*, *BCPE*, *BCCGo*, *BCCG*). Teniendo en cuenta esto, junto con información de otras publicaciones sobre la distribución de este tipo valores, se emplea la distribución *Box-Cox t-distribution*.

```
BCT()
```

```
##
## GAMLSS Family: BCT Box-Cox t
## Link function for mu    : identity
## Link function for sigma: log
## Link function for nu    : identity
## Link function for tau   : log
```

Modelo

```
library(gamlss)
modelo <- gamlss(
            formula        = head ~ pb(age),
            sigma.formula  = ~ pb(age),
            nu.formula     = ~ pb(age),
            tau.formula    = ~ pb(age),
            family  = BCT,
            data    = datos,
            control = gamlss.control(trace = FALSE)
          )
summary(modelo)
```

```
## ******************************************************************
## Family:  c("BCT", "Box-Cox t")
##
## Call:  gamlss(formula = head ~ pb(age), sigma.formula = ~pb(age),
##     nu.formula = ~pb(age), tau.formula = ~pb(age),
##     family = BCT, data = datos, control = gamlss.control(trace = FALSE))
##
## Fitting method: RS()
##
## ------------------------------------------------------------------
## Mu link function:  identity
## Mu Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 45.083516   0.028339  1590.8   <2e-16 ***
## pb(age)      0.723949   0.002836   255.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## ------------------------------------------------------------------
## Sigma link function:  log
## Sigma Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) -3.602497   0.021307 -169.077   <2e-16 ***
## pb(age)      0.005282   0.001830    2.887   0.0039 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Nu link function:  identity
## Nu Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.30827    0.61971   5.338 9.67e-08 ***
## pb(age)     -0.14253    0.05305  -2.687  0.00724 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## Tau link function:  log
## Tau Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.79955    0.11735  15.334  < 2e-16 ***
## pb(age)      0.07955    0.01950   4.079 4.58e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## ------------------------------------------------------------------
## NOTE: Additive smoothing terms exist in the formulas:
##  i) Std. Error for smoothers are for the linear effect only.
## ii) Std. Error for the linear terms maybe are not accurate.
## ------------------------------------------------------------------
## No. of observations in the fit:  7040
## Degrees of Freedom for the fit:  29.25792
##       Residual Deg. of Freedom:  7010.742
##                       at cycle:  11
##
## Global Deviance:     26754.34
##             AIC:     26812.85
##             SBC:     27013.54
## ******************************************************************
```

Predicción

La predicción del modelo es el valor estimado de los parámetros de la distribución (en este caso μ, σ, ν y τ) para cada valor de edad `age`.

```
# Se predice todo el rango de age para representar los cuantiles
grid_predictor <- seq(0, 20, length.out = 1000)

predicciones <- predictAll(
                  modelo,
                  newdata = data.frame(age = grid_predictor),
                  type = "response"
                )

predicciones <- as.data.frame(predicciones)
predicciones <- bind_cols(data.frame(age = grid_predictor), predicciones)
predicciones %>% head()
```

```
##          age       mu      sigma       nu      tau
## 1 0.00000000 35.77451 0.02663940 3.308269 6.046878
## 2 0.02002002 36.25825 0.02664713 3.305415 6.056516
## 3 0.04004004 36.74113 0.02665487 3.302562 6.066170
## 4 0.06006006 37.21120 0.02666261 3.299708 6.075839
## 5 0.08008008 37.66559 0.02667035 3.296855 6.085524
## 6 0.10010010 38.10464 0.02667810 3.294002 6.095223
```

Una vez predichos los parámetros que caracterizan a la distribución en función del predictor, se puede calcular la probabilidad de cada observación o el intervalo que acumula un determinado porcentaje de probabilidad (intervalo cuantílico). Todas las distribuciones de los paquetes `gamlss` y `gamlss.dist` disponen de funciones `d`, `p`, `q` y `r` para calcular probabilidad, densidad, cuantiles y generar valores aleatorios.

```
# Cálculo de los cuantiles teóricos para establecer el intervalo central que
acumula
# un 90% de probabilidad empleando los parámetros predichos.
predicciones <- predicciones %>%
                mutate(
                cuantil_10_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma,
                                              nu = nu, tau = tau),
                                    .f = function(mu, sigma, nu, tau) {
                                              qBCT(p = 0.1, mu, sigma, nu, tau)
                                         }
                                  ),
                cuantil_90_pred = purrr::pmap_dbl(
                                    .l = list(mu = mu, sigma = sigma,
                                              nu = nu, tau = tau),
```

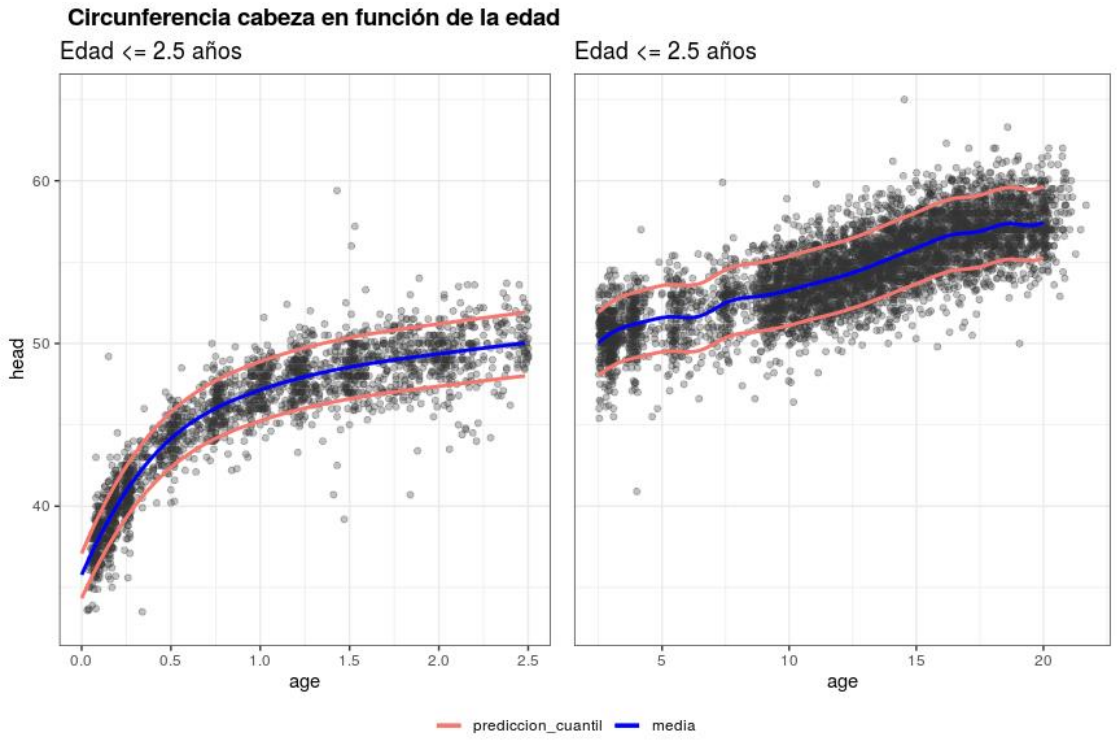
```
                                    .f = function(mu, sigma, nu, tau) {
                                              qBCT(p = 0.9, mu, sigma, nu, tau)
                                         }
                                    )
                )

p1 <- ggplot() +
      geom_point(
        data = datos %>% filter(age <= 2.5),
        aes(x = age, y = head),
        alpha = 0.3,
        color = "gray20") +
      geom_line(
        data = predicciones %>% filter(age <= 2.5),
        aes(x = age, y = cuantil_10_pred, color = "prediccion_cuantil"),
        size = 1) +
      geom_line(
        data = predicciones %>% filter(age <= 2.5),
        aes(x = age, y = cuantil_90_pred, color = "prediccion_cuantil"),
        size = 1) +
      geom_line(
        data = predicciones %>% filter(age <= 2.5),
        aes(x = age, y = mu, color = "media"),
        size = 1) +
      scale_color_manual(name = "",
                         breaks = c("prediccion_cuantil", "media"),
                         values = c("prediccion_cuantil"="#F8766D", "media"="blue")) +
      lims(y = c(33,65)) +
      labs(title = "Edad <= 2.5 años") +
      theme_bw()

p2 <- ggplot() +
      geom_point(
        data = datos %>% filter(age > 2.5),
        aes(x = age, y = head),
        alpha = 0.3,
        color = "gray20") +
      geom_line(
        data = predicciones %>% filter(age > 2.5),
        aes(x = age, y = cuantil_10_pred, color = "prediccion_cuantil"),
        size = 1) +
      geom_line(
        data = predicciones %>% filter(age > 2.5),
        aes(x = age, y = cuantil_90_pred, color = "prediccion_cuantil"),
        size = 1) +
      geom_line(
        data = predicciones %>% filter(age > 2.5),
        aes(x = age, y = mu, color = "media"),
        size = 1) +
      scale_color_manual(name = "",
```

```
                      breaks = c("prediccion_cuantil", "media"),
                      values = c("prediccion_cuantil"="#F8766D", "media"="blue")) +
      lims(y = c(33,65)) +
      labs(title = "Edad <= 2.5 años") +
      theme_bw() +
      theme(
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank())

ggarrange(p1, p2, nrow = 1, ncol = 2, align = "h",
          common.legend = TRUE, legend = "bottom")  %>%
    ggpubr::annotate_figure(
    top = text_grob("Circunferencia cabeza en función de la edad",
                    color = "Black",
                    face  = "bold",
                    size  = 14,
                    x     = 0.28)
)
```



Si se quiere conocer la probabilidad de que un niño de 2 años tenga un valor de circunferencia craneal igual o superior a 50, primero se predicen los parámetros de la distribución a para $age = 2$ y después se calcula la probabilidad de $head \geq 50$ con su función de distribución.

```
prediccion <- predictAll(
                modelo,
                newdata = data.frame(age = 2),
                type = "response"
              )

prediccion
```

```
## $mu
## [1] 49.36223
##
## $sigma
## [1] 0.02742632
##
## $nu
## [1] 3.023218
##
## $tau
## [1] 7.089748
##
## attr(,"family")
## [1] "BCT"       "Box-Cox t"
```

```
# Se calcula la probabilidad
probabilidad_head <- pBCT(
                        q     = 50,
                        mu    = prediccion$mu,
                        sigma = prediccion$sigma,
                        nu    = prediccion$nu,
                        tau   = prediccion$tau,
                        lower.tail = FALSE
                      )
probabilidad_head
```

```
## [1] 0.3237674
```

Acorde al modelo, la probabilidad de que un niño de 2 años tenga una circunferencia de cabeza igual o superior a 50 es del 32.4%.

Anexos

Anexo 1

Simulación ligeramente modificada del ejemplo publicado en *XGBoostLSS – An extension of XGBoost to probabilistic forecasting Alexander März.*

La ecuación empleada para generar los datos del ejemplo es:

$$y \sim \mathcal{N}(10, (1 + 1.5(4.8 < x < 7.2) + 4(7.2 < x < 12) + 1.5(12 < x < 14.4) + 2(x > 16.8))$$

Para el rango de valores $0 < x < 4.8$ los datos se distribuyen según una normal de media 10 y desviación típica 1. Para el rango de $4.8 < x < 7.2$ la desviación típica aumenta a 2.5. Para el rango $7.2 < x < 12$ pasa a 5, a continuación de $12 < x < 14.4$ desciende 2.5 y de $14.4 < x < 14.4$ a 1. Finalmente, para $x > 16.8$ la desviación aumenta a 3. El valor medio se mantiene constante (10) teniendo en todo momento la media de 10.

```
library(dplyr)

# Simulación distribución uniforme en el rango X
# ------------------------------------------------------------------------------
set.seed(123)
x <- rep(x = seq(0, 1, length.out = 96), each = 30)
y <- rnorm(
        length(x),
        mean = 10,
        sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
             1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
     )

# Cálculo del cuantil 0.1 y 0.9 para cada posición de x simulada.
cuantil_01 <- qnorm(
                p = 0.1,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )
cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )
```

```
datos    <- data.frame(y, x, cuantil_01, cuantil_90)
# No puede haber consumos negativos
datos <- datos %>%
         filter(y >=0)

# Simulación distribución no uniforme en el rango X
# ------------------------------------------------------------------------
set.seed(12345)
n <- 2000
x <- runif(min = 0, max = 24, n = n)
y <- rnorm(
        n,
        mean = 10,
        sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
             1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
     )
# Cálculo del cuantil 0.1 y 0.9 para cada posición de x simulada.
cuantil_01 <- qnorm(
                p = 0.1,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )

cuantil_90 <- qnorm(
                p = 0.9,
                mean = 10,
                sd = 1 + 1.5*(4.8 < x & x < 7.2) + 4*(7.2 < x & x < 12) +
                     1.5*(7.2 < x & x < 14.4) + 2*(x > 16.8)
              )
datos <- data.frame(y, x, cuantil_01, cuantil_90)

# No puede haber consumos negativos
datos <- datos %>%
         filter(y >=0)

datos <- datos %>%
         mutate(dentro_intervalo = ifelse(
                                     y > cuantil_01 & y < cuantil_90,
                                     TRUE,
                                     FALSE
                                   )
        )
```

Bibliografía

Stasinopoulos, Mikis & Rigby, Bob & Akantziliotou, Calliope. (2008). Instructions on how to use the gamlss package in R Second Edition.

Stasinopoulos, Dm & Rigby, Robert & Heller, Gillian & Voudouris, Vlasios & De Bastiani, Fernanda. (2017). Flexible regression and smoothing: Using GAMLSS in R. 10.1201/b21973.

Stasinopoulos, D., & Rigby, R. (2007). Generalized Additive Models for Location Scale and Shape (GAMLSS) in R. Journal of Statistical Software, 23(7), 1 - 46. doi:http://dx.doi.org/10.18637/jss.v023.i07

Rigby, R.A. and Stasinopoulos, D.M. (2005), Generalized additive models for location, scale and shape. Journal of the Royal Statistical Society: Series C (Applied Statistics), 54: 507-554. doi:10.1111/j.1467-9876.2005.00510.x

Hothorn, Torsten, Thomas Kneib, and Peter Bühlmann. "Conditional Transformation Models." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 76.1 (2013)

Hofner, B., Mayr, A., & Schmid, M. (2016). gamboostLSS: An R Package for Model Building and Variable Selection in the GAMLSS Framework. Journal of Statistical Software, 74(1), 1 - 31. doi:http://dx.doi.org/10.18637/jss.v074.i01