

# Explorando más en detalle el paquete `biometrics`

## Ejemplos

Este documento extiende los ejemplos de distintas funciones del paquete `biometrics` (Salas-Eljatib et al., 2025) implementado en el software R (R Core Team, 2025) para uso interno.

## Índice

<b>1. <code>treestat</code></b>	<b>2</b>
1.1. Para una variable aleatoria . . . . .	2
1.2. Para dos variables aleatorias . . . . .	3
1.3. Una estadística descriptiva completa . . . . .	3
1.4. Sobre el uso de la variable diámetro del árbol . . . . .	4
1.5. Sobre agregar la variable área basal del árbol . . . . .	4
1.6. Agregando el diámetro y área basal al listado de variables . . . . .	5
1.7. Segregado por un factor . . . . .	5
1.8. Listado de árboles medidos en el tiempo . . . . .	6
1.9. Segregado por niveles de un factor . . . . .	7
1.10. Segregado por niveles de un factor y varias variables . . . . .	7
<b>2. <code>bankfit</code></b>	<b>9</b>
<b>3. <code>bankpred</code></b>	<b>12</b>
3.1. Sobre esta función . . . . .	12
3.2. Datos para ejemplos . . . . .	12

## 1. treestat

### Sobre esta función

La función **treestat** tiene por objetivo calcular una serie de estadísticos descriptivos a nivel de árbol por unidad muestral (*i.e.*, parcela). Los estadísticos pueden ser realizados para **una** o **varias** variables aleatorias, que se especifican en la opción **y**. Además, los cálculos se pueden segregar por un factor, o variable categórica, que se especifica en la opción **factvar**.

Note que los estadísticos a calcular están definidos en la función **descstat()** del paquete **datana** (Salas-Eljatib & Campos, 2025).

### Datos para ejemplos

Se empleará los datos de un listado de árboles disponibles en la dataframe **eucaplot2** del paquete **biometrics**.

```
# Dataframe to be used
df <- biometrics::eucaplot2
datana::descstat(df[, c("dap", "atot")])
df$parce <- 1
head(df)
```

```
dap  atot
n      15.000 15.000
Minimum 24.000 26.000
Maximum 37.500 35.000
Mean    30.493 29.667
Median  29.500 30.000
Std. Dev. 3.800 3.039
CV(%)    12.463 10.245
dap sanidad forma clase.copa atot parce
1 28.0      1      1      2 27      1
2 35.5      1      1      1 31      1
3 34.0      1      2      1 31      1
4 37.5      1      2      1 35      1
5 26.5      1      2      2 27      1
6 29.0      1      1      2 26      1
```

A continuación se detallan una serie de ejemplos con los potenciales usos de la función en cuestión.

#### 1.1. Para una variable aleatoria

Se emplea como variable aleatoria de interés a la altura total, la cual está almacenada en la columna **atot**. Un primer ejemplo sería entonces

```
## Using the function
treestat(data = df, plot.id = "parce", y = "atot")
```

```
parce    t n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot
1      1 2025    15      26      35    29.667          30  3.039 10.245
```

## 1.2. Para dos variables aleatorias

Se agrega dentro de las variables aleatorias para las cuales se calcularán los estadísticos descriptivos, a la variable `dap`, por lo tanto ahora, la opción `y` debe ser un vector con el nombre de las dos variables

```
# Now, for two random variables, instead of a single one
treestat(data = df, plot.id = "parce", y = c("dap", "atot"))
```

```
parce    t n.dap min.dap max.dap mean.dap median.dap sd.dap cv.dap n.atot min.atot
1      1 2025    15      24    37.5    30.493      29.5    3.8 12.463    15      26
      max.atot mean.atot median.atot sd.atot cv.atot
1          35    29.667          30    3.039 10.245
```

## 1.3. Una estadística descriptiva completa

Por defecto, los estadísticos a calcular son los definidos en la función `descstat()` del paquete `datana`, sin embargo, se pueden pasar las opciones que tiene dicha función desde la presente `treestat()`, al especificar la opción respectiva en esta última. Por ejemplo, la opción lógica `full` de la función `descstat()`, que por defecto es igual a `FALSE`, sin embargo, si se especifica lo contrario, mire el cambio.

```
## la opcion full de la funcion descstat()
datana::descstat(df[, c("dap","atot")], full=TRUE)
```

```
dap  atot
n      15.000 15.000
Minimum 24.000 26.000
Maximum 37.500 35.000
Mean    30.493 29.667
Median  29.500 30.000
Std. Dev. 3.800 3.039
CV(%)    12.463 10.245
25th Percentile 28.000 27.000
75th Percentile 33.000 31.000
Interq. range   5.000 4.000
Skewness    0.274 0.410
Kurtosis    -1.054 -1.192
```

Un ejemplo de lo anterior, es aplicar la opción `full` de la función `descstat()` al aplicar la función `treestat()` a una variable aleatoria, *i.e.*, `atot`, lo que se logra mediante

```
## Using the function con opcion full
treestat(data = df, plot.id = "parce", y = "atot", full=TRUE)
```

```
parce    t n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot 25p.atot
1      1 2025    15      26      35    29.667          30  3.039 10.245    27
75p.atot iqr.atot ske.atot kurt.atot
1      31      4      0.41    -1.192
```

Note el aumento en el número de columnas del output, lo que representa a una mayor cantidad de estadísticos calculados, en comparación con lo mostrado en §1.1.

#### 1.4. Sobre el uso de la variable diámetro del árbol

El diámetro a la altura del pecho ( $d$ ), es una variable que se mide en todos los árboles dentro de una unidad muestral. Esta variable se puede agregar al listado de variables a las cuales aplicar la función, al especificar la en la opción `d` el nombre de la columna con la variable  $d$  y especificar que la opción `want.add.d` sea `TRUE`, como sigue

```
treestat(data = df, plot.id = "parce", y = "atot", d = "dap", want.add.d = TRUE)
```

Message: The dbh was specified to be the column named: "dap", which will be used accordantly.

```
parce    t n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot n.dap min.dap
1      1 2025    15      26      35    29.667          30  3.039 10.245    15    24
max.dap mean.dap median.dap sd.dap cv.dap
1    37.5    30.493      29.5    3.8 12.463
```

Note lo siguiente:

- Esta salida es equivalente a la dada en §1.2, con la salvedad que ahora, la variable “dap” se agrega al final del listado de variables.
- En la salida aparece un mensaje indicando que columna de los datos fue indicada como la de diámetro a la altura del pecho.

#### 1.5. Sobre agregar la variable área basal del árbol

El área basal de un árbol ( $g$ ) es una variable que se calcula a partir del diámetro  $d$ . Por lo tanto, una vez que se conoce el diámetro es posible calcular  $g$ . En caso que se quiera agregar a la variable  $g$ , se debe especificar que la opción `want.add.g` sea `TRUE`, como sigue

```
treestat(data = df, plot.id = "parce", y = "atot", d = "dap", want.add.g = TRUE)
```

```
parce    t n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot n.g min.g
1      1 2025    15      26      35    29.667          30  3.039 10.245    15 0.045
max.g mean.g median.g sd.g cv.g
1    0.11  0.074    0.068 0.019 25.07
```

Entonces los estadísticos son calculados primero para la variable (o vector de variables) especificada en la opción `y`, a la cual se le agrega (al final) la variable  $g$ . Note lo siguiente:

- El área basal solo se puede calcular en esta función, siempre y cuando, haya sido declarado en qué columna esta el diámetro a la altura del pecho (en la opción `d`).
- Por defecto, el idioma para los mensajes, así como para el output de la función `descstat()` es el Inglés. No obstante, si Ud. desea los resultados en Castellano, especifique que la opción `eng` sea `FALSE`.

```
treestat(data = df, plot.id = "parce", y = "atot", d = "dap", want.add.g = TRUE, eng =
→ FALSE)
```

```
parce    t n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot n.ab min.ab
1      1 2025    15      26      35    29.667          30   3.039 10.245   15  0.045
max.ab mean.ab median.ab sd.ab cv.ab
1    0.11   0.074    0.068 0.019 25.07
```

## 1.6. Agregando el diámetro y área basal al listado de variables

Aunque se conoce a la variable diámetro, lo cual fue especificado en la opción `d`, esta variable no se agregó a los calculos en §1.5, ya que por defecto la opción `want.add.d` es `FALSE`.

Dado lo anterior, si se especifica la columna del diámetro a la altura del pecho, se puede agregar tanto a dicha variable y al área basal, como dos nuevas variables (que se agregan al objeto `y`) a la función en cuestión, como sigue

```
treestat(data = df, plot.id = "parce", y = "atot", d = "dap",
want.add.d = TRUE, want.add.g = TRUE)
```

Message: The dbh was specified to be the column named: "dap", which will be used accordantly.

```
parce    t n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot n.dap min.dap
1      1 2025    15      26      35    29.667          30   3.039 10.245   15    24
max.dap mean.dap median.dap sd.dap cv.dap n.g min.g max.g mean.g median.g sd.g cv.g
1    37.5   30.493    29.5    3.8 12.463  15 0.045  0.11  0.074    0.068 0.019 25.07
```

## 1.7. Segregado por un factor

Todo lo anterior, se puede realizar segregado por un factor, o variable categórica. Para ejemplificar esto, se utilizará al factor "clase de copa" (columna `clase.copa`), para una variable aleatoria, tal como en §1.1, mediante

```
# Do the same as before, but adding the computation by a factor
treestat(data = df, plot.id = "parce", y = "atot", factvar = "clase.copa")
```

```
parce    t factor level n.atot min.atot max.atot mean.atot median.atot sd.atot
2      1 2025 clase.copa    1      8      30      35    32.000          31  2.070
3      1 2025 clase.copa    2      7      26      29    27.000          27  1.000
1      1 2025      None All    15      26      35    29.667          30  3.039
```

```
cv.atot
2    6.469
3    3.704
1   10.245
```

En la salida anterior, note lo siguiente:

- la columna **factor** guarda la variable categórica seleccionada, lo cual busca verificar que la función se esté aplicando correctamente según los requerimientos del usuario.
- la columna **level**, especifica cada nivel del factor empleado para los cálculos.
- la fila identificada como **None** para la columna **factor**, corresponde al resultado de aplicar la función sin considerar un factor para segregar los cálculos. Por lo tanto, estos resultados son equivalentes a los reportados en §1.1.

## 1.8. Listado de árboles medidos en el tiempo

Asuma ahora que la unidad muestral de los datos empleados, ha sido medida en dos ocasiones en el tiempo, tanto en el 2020 como en el 2025.

```
# Creando un set de datos con remediciones (ficticio)
df$time <- 2025
df$time[1:5] <- 2020
df
table(df$time)
```

```
dap sanidad forma clase.copa atot parce time
1 28.0      1      1          2 27      1 2020
2 35.5      1      1          1 31      1 2020
3 34.0      1      2          1 31      1 2020
4 37.5      1      2          1 35      1 2020
5 26.5      1      2          2 27      1 2020
6 29.0      1      1          2 26      1 2025
7 24.0      1      1          2 27      1 2025
8 30.0      1      1          1 30      1 2025
9 29.5      1      2          2 27      1 2025
10 32.0      1      2          1 33      1 2025
11 29.4      1      1          2 26      1 2025
12 28.0      1      2          2 29      1 2025
13 31.5      1      2          1 35      1 2025
14 27.0      1      2          1 31      1 2025
15 35.5      2      2          1 30      1 2025
```

```
2020 2025
5    10
```

Entonces, los cálculos no sólo deben considerar al código de la parcela (columna **parce**), sino que además el año

de medición, que se almacenó en la columna `time`.

```
## Using the function per measurement year
treestat(data = df, plot.id = "parce", y = "atot", t = "time")
```

parce	time	n.atot	min.atot	max.atot	mean.atot	median.atot	sd.atot	cv.atot
1	1 2020	5	27	35	30.2	31.0	3.347	11.082
2	1 2025	10	26	35	29.4	29.5	3.026	10.292

### 1.9. Segregado por niveles de un factor

Se puede además, así como se mostró en §1.7, realizar lo anterior segregado por un factor, es decir, aplicar la función para cada año de medición y cada nivel del factor.

```
## Using the function per measurement year
treestat(data = df, plot.id = "parce", y = "atot", t = "time", factvar = "clase.copa")
```

parce	time	factor	level	n.atot	min.atot	max.atot	mean.atot	median.atot	sd.atot
3	1 2020	clase.copa	1	3	31	35	32.333	31.0	2.309
5	1 2020	clase.copa	2	2	27	27	27.000	27.0	0.000
1	1 2020	None	All	5	27	35	30.200	31.0	3.347
4	1 2025	clase.copa	1	5	30	35	31.800	31.0	2.168
6	1 2025	clase.copa	2	5	26	29	27.000	27.0	1.225
2	1 2025	None	All	10	26	35	29.400	29.5	3.026

  

cv.atot
3 7.142
5 0.000
1 11.082
4 6.817
6 4.536
2 10.292

### 1.10. Segregado por niveles de un factor y varias variables

De la misma manera que se vió en §1.2, se puede ahora también considerar agregar más variables aleatorias a lo anterior, como sigue

```
# Do the same as before, but adding the computation by a factor
treestat(data = df, plot.id = "parce", y = c("dap", "atot"), t = "time",
          factvar = "clase.copa", d = "dap", want.add.g = TRUE)
```

parce	time	factor	level	n.dap	min.dap	max.dap	mean.dap	median.dap	sd.dap	cv.dap
3	1 2020	clase.copa	1	3	34.0	37.5	35.667	35.50	1.756	4.923
4	1 2020	clase.copa	2	2	26.5	28.0	27.250	27.25	1.061	3.892

1	1	2020	None	All	5	26.5	37.5	32.300	34.00	4.804	14.872
5	1	2025	clase.copa	1	5	27.0	35.5	31.200	31.50	3.094	9.918
6	1	2025	clase.copa	2	5	24.0	29.5	27.980	29.00	2.303	8.229
2	1	2025	None	All	10	24.0	35.5	29.590	29.45	3.081	10.412
n.atot min.atot max.atot mean.atot median.atot sd.atot cv.atot n.g min.g max.g mean.g											
3	3		31	35	32.333		31.0	2.309	7.142	3	0.091 0.110 0.100
4	2		27	27	27.000		27.0	0.000	0.000	2	0.055 0.062 0.058
1	5		27	35	30.200		31.0	3.347	11.082	5	0.055 0.110 0.083
5	5		30	35	31.800		31.0	2.168	6.817	5	0.057 0.099 0.077
6	5		26	29	27.000		27.0	1.225	4.536	5	0.045 0.068 0.062
2	10		26	35	29.400		29.5	3.026	10.292	10	0.045 0.099 0.069
median.g sd.g cv.g											
3			0.099	0.010	9.866						
4			0.058	0.005	7.779						
1			0.091	0.024	28.774						
5			0.078	0.015	19.739						
6			0.066	0.010	15.607						
2			0.068	0.014	20.803						



## 2. bankfit

### Sobre esta función

La función `bankfit` permite el ajuste de una lista de modelos para un set de datos. Por lo tanto la función necesita, al menos, de los siguientes inputs:

1. Datos para el ajuste, como un objeto `dataframe` de R.
2. Listado de funciones, como un objeto `list` de R.

Aunque la función `bankfit` es parte del paquete `biometrics` (Salas-Eljatib et al., 2025), para su uso, se debe emplear (o tener instalado) al paquete `datana` (Salas-Eljatib & Campos, 2025).

El listado de modelos a ingresar tiene la siguiente forma:

```
model.list <- list(  
  ## modelo lineal, se ajustará con funcion lm()  
  mod1 = list(expresion_del_modelo = log(y)~x,  
    transformacion_var_respuesta = function(y) exp(y),  
    tipo_del_ajuste = "lm",  
    author_modelo = "Apellido N. 1990"),  
  ## modelo no lineal, se ajustará con funcion nls()  
  mod2 = list(expresion_del_modelo = y~taperpoly(x, n = 2),  
    params = list(b0 = 1, b1 = 0, b2 = 0.5),  
    transformacion_var_respuesta = function(y) sqrt(y),  
    tipo_del_ajuste = "nls",  
    author_modelo = "Apellido N. 1995"),  
  ## modelo ya ajustado, sólo se usará para predicción  
  mod3 = list(expresion_del_modelo = kozak69(hl = df$hl,  
    href = df$href,  
    paramod = c(b0 = 0.4246,  
      b1 = -2.3904,  
      b2 = 1.3973)),  
    transformacion_var_respuesta = function(y, d, ...) y * d,  
    tipo_del_ajuste = "fitted"))
```

La base de datos o `dataframe` debe tener las variables correspondientes para poder ajustar (siguiendo el ejemplo del bloque anterior, columnas `x` e `y`). En caso de no encontrarse alguna variable `bankfit()` no ajustará dicho modelo.

### Datos para ejemplos

Se empleará los datos de una muestra de árboles disponibles en la `dataframe` `treevolruca2` del paquete `biometrics`.

```
library(biometrics)  
library(datana)
```

```
df <- treevolruca2
datana::descstat(df[,c("dap", "atot", "vtot")])
```

dap	atot	vtot
n	382.000	382.000
Minimum	11.500	9.800
Maximum	143.400	53.500
Mean	48.752	27.115
Median	45.250	26.500
Std. Dev.	23.006	8.192
CV(%)	47.190	30.211

En algunos casos, será necesario generar las columnas con los nombres utilizados en la nomenclatura del laboratorio, como sigue

```
df$d <- df$dap # para el dap
df$h <- df$atot # para la altura total
df$v <- df$vtot # para el volumen total
```

A continuación se muestra una lista de modelos de volumen, los que se ajustarán de forma lineal con la función `lm()`.

```
model.list <- list(mod1 = list(expr = v ~ I(d^2) + I(d^2 * h^2) + I(d6),
                             pred.f = function(y, ...) y,
                             type = "lm"),
                  mod2 = list(expr = I(d^2 / v) ~ I(1 / h),
                             pred.f = function(y, d, ...) d^2 / y,
                             type = "lm"),
                  mod3 = list(expr = I(log10(v)) ~ I(log10(d)) + I(log10(h)),
                             pred.f = function(y, ...) 10^y,
                             type = "lm"),
                  mod4 = list(expr = I(log(v)) ~ I(log(d)) + I(log(h)),
                             pred.f = function(y, ...) exp(y),
                             type = "lm"),
                  mod5 = list(expr = I(sqrt(v)) ~ I(log(d)) + I(log(h)),
                             pred.f = function(y, ...) y^2,
                             type = "lm")
)
```

Note que es necesario cargar, antes de usar la función `bankfit()`, la librería `datana`, que desde acá se llama internamente a la función `modresults()`.

Para realizar el ajuste basta con llamar la función `bankfit`:

```
ajuste <- bankfit(models = model.list, data = df)
## vemos los contenidos de `ajuste`
lapply(ajuste, names)
```

```
$mod1
[1] "mod"      "summ"     "data"     "pred.f"   "type"

$mod2
[1] "mod"      "summ"     "data"     "pred.f"   "type"

$mod3
[1] "mod"      "summ"     "data"     "pred.f"   "type"

$mod4
[1] "mod"      "summ"     "data"     "pred.f"   "type"

$mod5
[1] "mod"      "summ"     "data"     "pred.f"   "type"
```

Los resultados del ajuste del modelo `mod1` se pueden verificar manipulando la salida como una lista

```
ajuste$mod1$mod
```

Call:

```
lm(formula = models[[x]][["expr"]], data = data)
```

Coefficients:

(Intercept)	$I(d^2)$	$I(d^2 * h^2)$	$I(d6)$
-1.342e+00	1.021e-04	3.168e-07	7.510e-02

Es posible guardar los resultados en un archivo `.rdata`, lo que es útil y necesario para **predecir** con otra función llamada `bankpred`.

```
## guarda sólo los modelos ajustados y su respectivo pred.f
bankfit(models = model.list, data = df, file = "./resources/fitted.rdata")
## guarda modelos ajustados, pred.f, summary y los datos
bankfit(models = model.list, data = df, file = "./resources/fitted-full.rdata",
        file.full = TRUE)
```

### 3. bankpred

#### 3.1. Sobre esta función

La función **bankpred** fue creada principalmente para los procesos de ajuste y validación de modelos, específicamente esta función toma una lista de modelos **ya ajustados** y (opcionalmente) una base de datos y realiza las respectivas predicciones por modelo.

El listado de modelos ajustados debe venir de la salida de la función **bankfit**, por ejemplo:

```
bankfit(models = lista_modelos, data = mis_datos, file = "ajustado.rdata")
```

Crearé el archivo “ajustado.rdata” para ser usado posteriormente.

#### 3.2. Datos para ejemplos

Se cargarán los modelos ajustados en el ejemplo de §2. Siempre es necesario indicar sobre qué datos se hará la predicción. En este caso usaremos los mismos datos de ajuste (**biometrics::treevolruca2**), indicándolos en la opción **data**:

```
df <- biometrics::treevolruca2
df$d <- df$dap
df$h <- df$atot
df$v <- df$vtot

pred1 <- bankpred(file = "./resources/fitted.rdata", data = df)
head(pred1)
```

	y.hat	y.pred	model
1	12.8773454	12.8773454	mod1
2	0.6064593	0.6064593	mod1
3	1.3660116	1.3660116	mod1
4	-0.3537542	-0.3537542	mod1
5	0.9364930	0.9364930	mod1
6	4.8287693	4.8287693	mod1

Es posible usar datos distintos a los del ajuste (*i.e* validar) mediante la opción **data**. A continuación se usará el **ajuste** hecho con los datos de **biometrics::treevolruca2** (§Datos para ejemplos) en la base de datos **datana::idahohd2**:

```
library(datana)
df <- idahohd2
df$d <- df$dap
df$h <- df$atot

pred2 <- bankpred(file = "./resources/fitted.rdata", data = df)
head(pred2)
```

y.hat	y.pred	model
1	NA	mod1
1100	1448.1708	1.0502905 mod2
2	988.7129	2.3303023 mod2
3	1064.0640	2.5412005 mod2
4	1436.4513	0.9022234 mod2
5	1340.3514	1.0773294 mod2

Se puede ver que las predicciones con datos distintos no son completas, en este caso el modelo 1 no se pudo predecir puesto que la variable respuesta *d6* no existe en la nueva base de datos. En tales casos la predicción quedará con un valor asignado de NA.

## Referencias

- R Core Team. (2025). *R: A language and environment for statistical computing* [<http://www.R-project.org>]. R Foundation for Statistical Computing. Vienna, Austria.
- Salas-Eljatib, C., & Campos, N. (2025). *datana: Data and functions to accompany Análisis de datos con R* [R package version 1.1.4]. <https://doi.org/10.32614/CRAN.package.datana>
- Salas-Eljatib, C., Campos, N., & Marivil, M. (2025). *biometrics: Functions and datasets for forest biometrics and modelling* [R package version 1.0.2]. <https://doi.org/10.32614/CRAN.package.biometrics>