R_Básico_3

J.Ballesta

17/10/24

Resumen

Resumen En esta publicación continuaremos explorando las posibilidades de ggplot2, el package DataExplorer para el análisis exploratorio de datos y la comprobación del supuesto de normalidad de un conjunto de datos.

In this post, we will continue exploring the possibilities of ggplot2, the DataExplorer package for exploratory data analysis, and testing the assumption of normality for a dataset.

In diesem Beitrag werden wir die Möglichkeiten von ggplot2, dem DataExplorer-Paket für die explorative Datenanalyse, sowie die Überprüfung der Annahme der Normalverteilung eines Datensatzes weiter untersuchen.

Introducción

Seguimos trabajando con R(2024a), Quarto(Allaire et al. 2022) y RStudio(Posit team 2024) y conociendo las posibilidades de análisis de datos que nos ofrecen. En esta publicación añadiremos alguna visualización adicional con *ggplot2*(Wickham 2016a) y presentaremos el package *DataExplorer*(Cui 2024a) para análisis exploratorio de datos (en inglés EDA) y *patchwork*(Pedersen 2024) para la composición de gráficos de *ggplot2*(Wickham 2016b).

```
#
#    Normalmente en el bloque inicial es conveniente declarar las librerias/packages que vamos
a usar, en este caso:
library(ggplot2)    # Visualización de gráficos
library(patchwork) # Composición de los gráficos de ggplot2
library(scales)    # Formateo de los ejes de los gráficos en ggplot2
library(DataExplorer) # Uno de los muchos packages disponibles para EDA
library (moments)   # cálculo de la asimetria y curtosis de la distribución
# si están en el bloque inicial siempre sabemos donde buscar la librería y en caso de que el
# script alcance muchas líneas de código no hay que recorrer todo el código buscando en que
# línea hemos cargado la librería-
#
#    Damos un valor de semilla para la reproducibilidad de los resultados obtenidos en este
# script.
set.seed(123)
```

En esta publicación usaremos el conjunto de datos (*dataset*) disponible en R: Orange, contiene 35 registros y tres variables, que recogen el crecimiento de unos naranjos.

```
#
#    Asignamos el dataset a una variable de trabajo.
datos <- Orange
# comprobamos la estructura de la variable.
str(datos)
```

```
Classes 'nfnGroupedData', 'nfGroupedData', 'groupedData' and 'data.frame':	35 obs. of	3 variables:
 $ Tree        : Ord.factor w/ 5 levels "3"<"1"<"5"<"2"<..: 2 2 2 2 2 2 2 4 4 4 ...
 $ age         : num  118 484 664 1004 1231 ...
 $ circumference: num  30 58 87 115 120 142 145 33 69 111 ...
 - attr(*, "formula")=Class 'formula'  language circumference ~ age | Tree
  .. ..- attr(*, ".Environment")=<environment: R_EmptyEnv>
 - attr(*, "labels")=List of 2
  ..$ x: chr "Time since December 31, 1968"
  ..$ y: chr "Trunk circumference"
 - attr(*, "units")=List of 2
  ..$ x: chr "(days)"
  ..$ y: chr "(mm)"

# comprobamos cabecera y fin de los datos cargados
head(datos, 5)

  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120

#
tail(datos, 5)

   Tree  age circumference
31    5  664            81
32    5 1004           125
33    5 1231           142
34    5 1372           174
35    5 1582           177

# hacemos un resumen de algunas magnitudes estadísticas de los datos
summary(datos)

 Tree       age         circumference
 3:7   Min.   : 118.0   Min.   : 30.0
 1:7   1st Qu.: 484.0   1st Qu.: 65.5
 5:7   Median :1004.0   Median :115.0
 2:7   Mean   : 922.1   Mean   :115.9
 4:7   3rd Qu.:1372.0   3rd Qu.:161.5
       Max.   :1582.0   Max.   :214.0

#
```

Las variables en el *dataset*, muestran:

- *$Tree* : un factor ordenado que indica el árbol en el cual se ha hecho la medida.
- *$age* : un valor numérico que muestra la edad del árbol en días desde 31/12/1.968.

- *$circumference* : un valor numérico de diámetro de la circunferencia del tronco en mm.

Análisis exploratorio de datos.

R(2024b) es un lenguaje de programación Open Source y dispone de una amplia gama de packages (aunque en el script se llamen mediante la función *library()*) desarrollados por la comunidad de usuarios, que facilitan la programación de análisis para los problemas más diversos. En esta ocasión, optamos por un *package* de R llamado *DataExplorer*(Cui 2024b), especializado para el análisis exploratorio de datos.

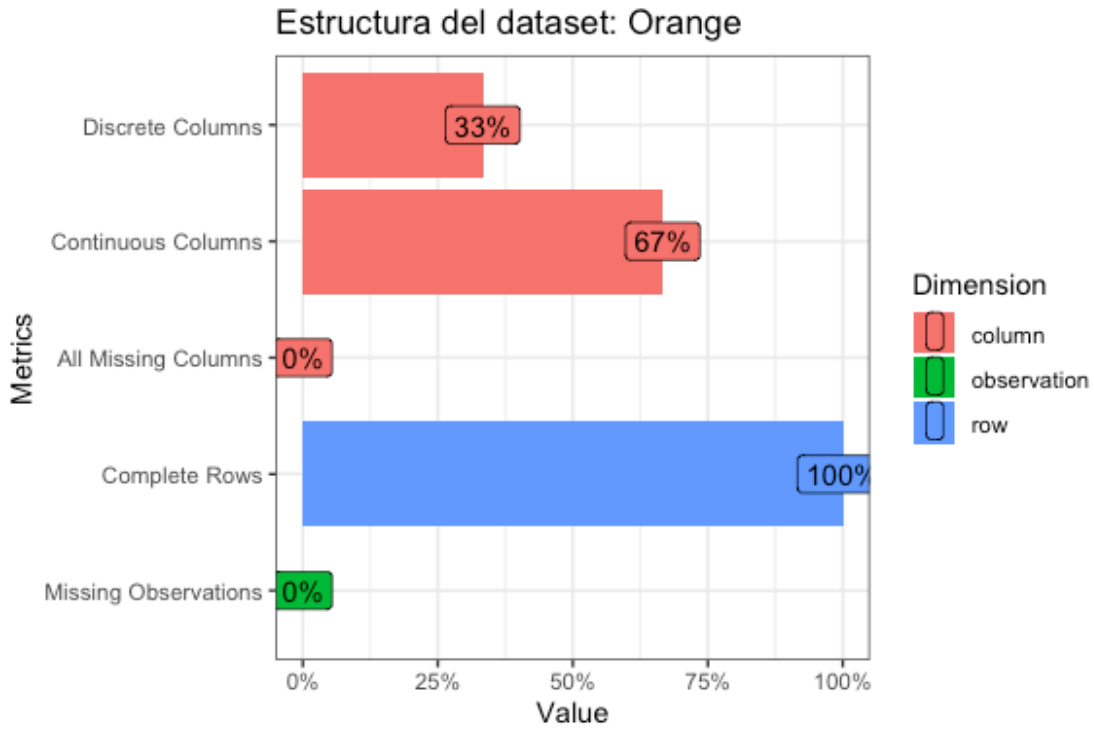
La lista de *packages* disponibles en R a través de CRAN : Link

El análisis exploratorio de datos (EDA en inglés) es un estudio preliminar para conocer los datos de los cuales disponemos y que puntos o que hipótesis debemos comprobar.

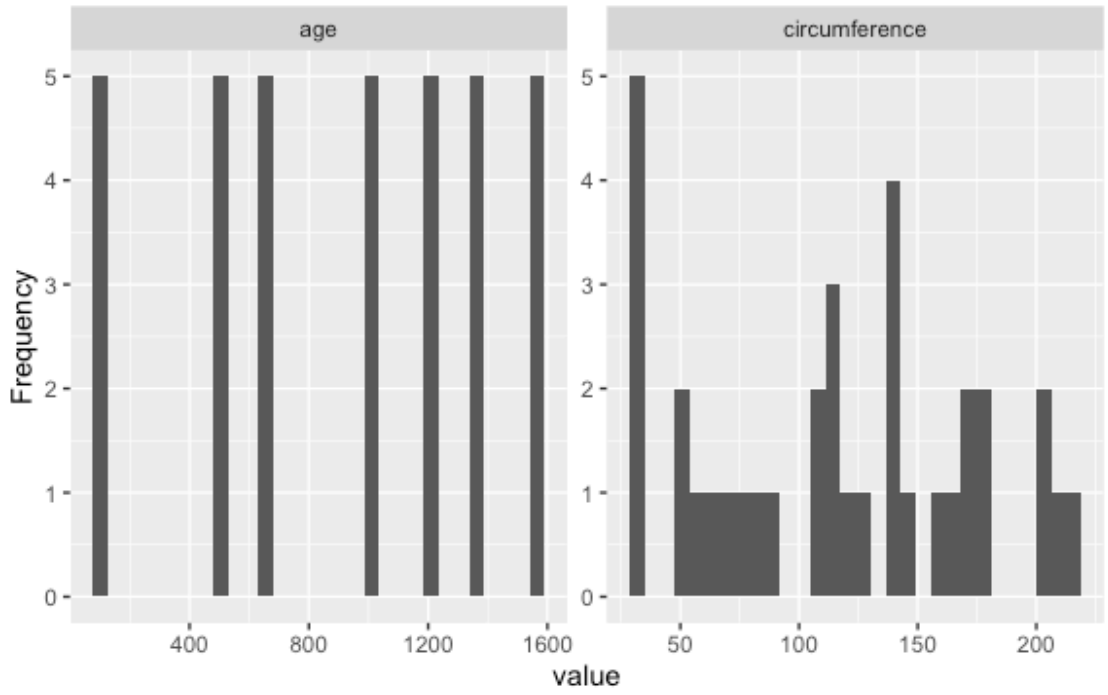
```
#
#    En el bloque inicial del script ya hemos cargado la librería DataExplorer
# en primer lugar vemos información básica de los datos, como numero de filas, columnas
# /variables, columnas con valores discretos, columnas con valores continuos...
introduce(datos)
```

```
  rows columns discrete_columns continuous_columns all_missing_columns
1   35       3               1                 2                   0
  total_missing_values complete_rows total_observations memory_usage
1                    0           35                105        3040
```

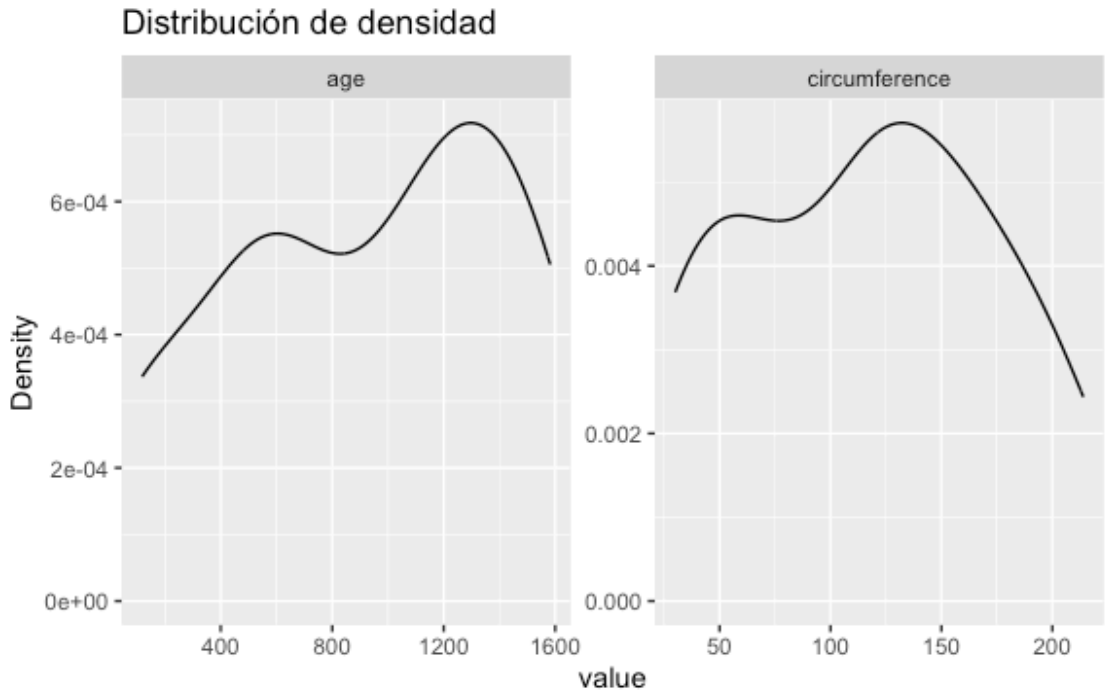
```
#
#    a los datos anteriores les podemos dar una salida gráfica mediante la función plot_intro()
# DataExplorer usa la base de ggplot2 para algunas de sus salidas, podemos usar algunos de los parámetros de ggplot2.
plot_intro (datos,
            title= "Estructura del dataset: Orange",
            ggtheme= theme_bw()
            )
```



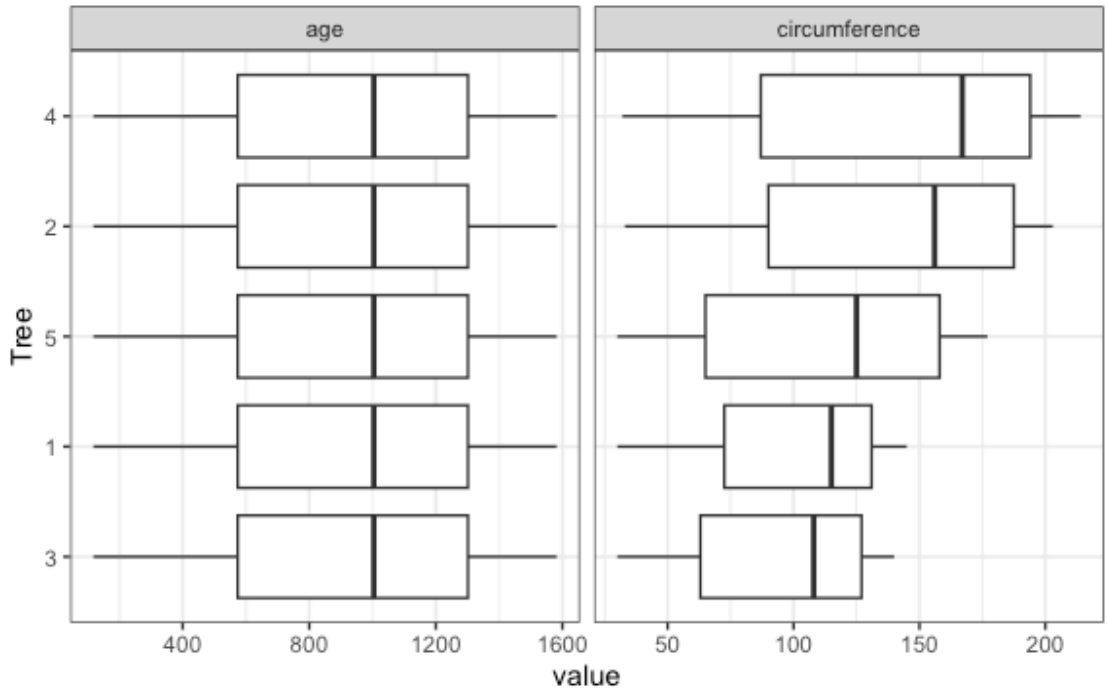

```
#
#    también es posible hacer una visualización de la distribución de frecuencias de las
# variables numéricas
plot_histogram (datos)
```




```
# o de su distribución de densidad
plot_density(datos,
             title = "Distribución de densidad")
```




```
#
#    podemos hacer una visualización por diagrama de cajas, respecto del tipo de árbol
plot_boxplot(datos, by="Tree",
             ggtheme= theme_bw())
```



DataExplorer, fuentes de información.

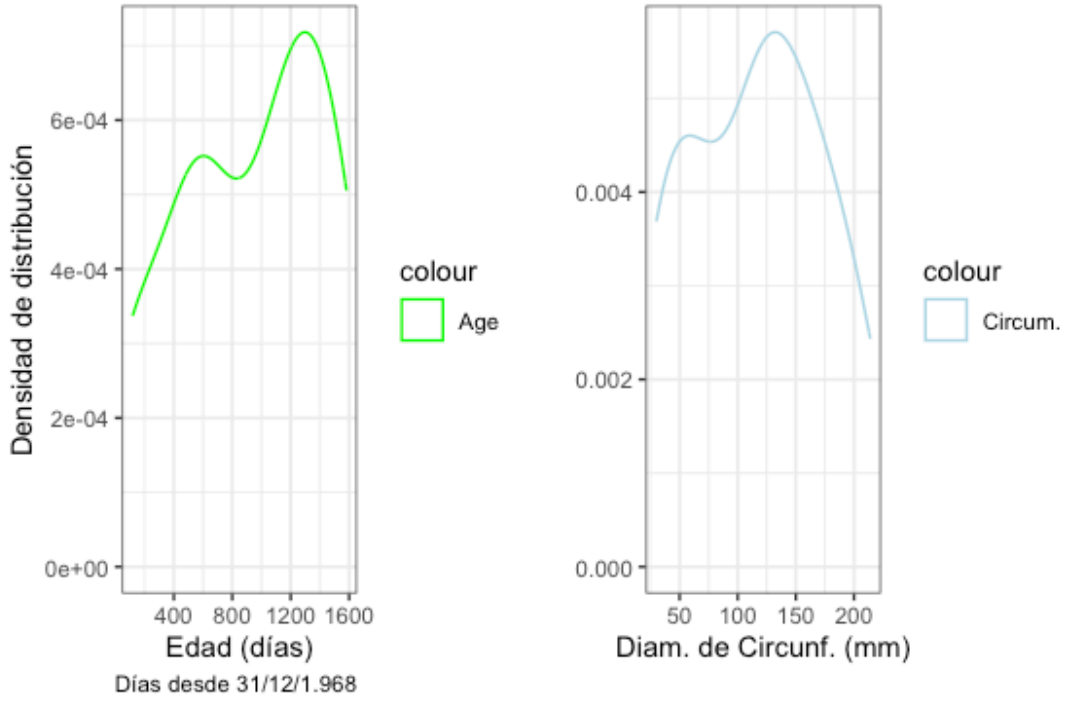
1. Cran : DataExplorer: Automate Data Exploration and Treatment

2. Introduction to DataExplorer

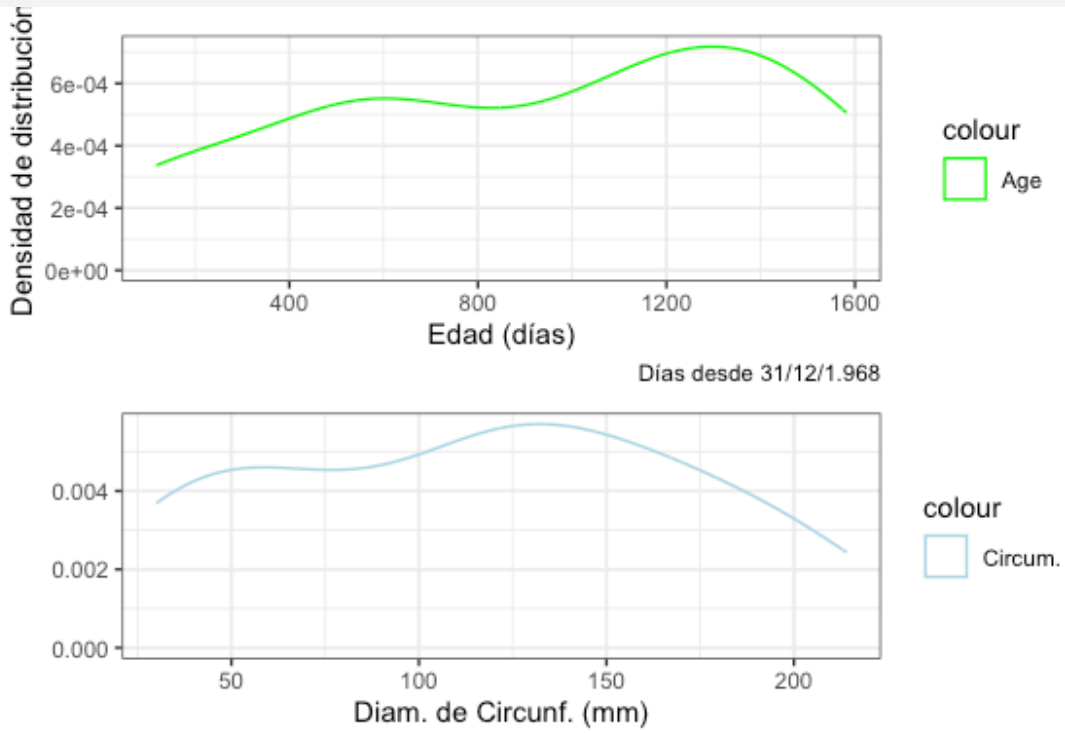
Análisis gráfico.

Continuamos con las posibilidades que nos ofrece *ggplot2*(Wickham 2016c), de presentar gráficamente nuestros datos en análisis.

```
#
#    En esta ocasión haremos uso del package patchwork para componer los dos gráficos de
# densidad de distribución de datos$age y datos$circumference
p1 <- ggplot(data=datos)+
        geom_density(aes(x=age, color="Age"))+
        scale_color_manual(values= c("Age"="green", "Circum."="lightblue"))+
        xlab("Edad (días)")+
        ylab("Densidad de distribución")+
        labs(caption = "Días desde 31/12/1.968")+
        theme_bw()
#
p2 <- ggplot(data=datos)+
        geom_density(aes(x=circumference, color="Circum."))+
        scale_color_manual(values= c("Age"="green", "Circum."="lightblue"))+
        xlab("Diam. de Circunf. (mm)")+
        ylab("")+
        theme_bw()
# componemos los gráficos almacenados en la variables p1 y p2 uno al lado del otro
p1+p2
```

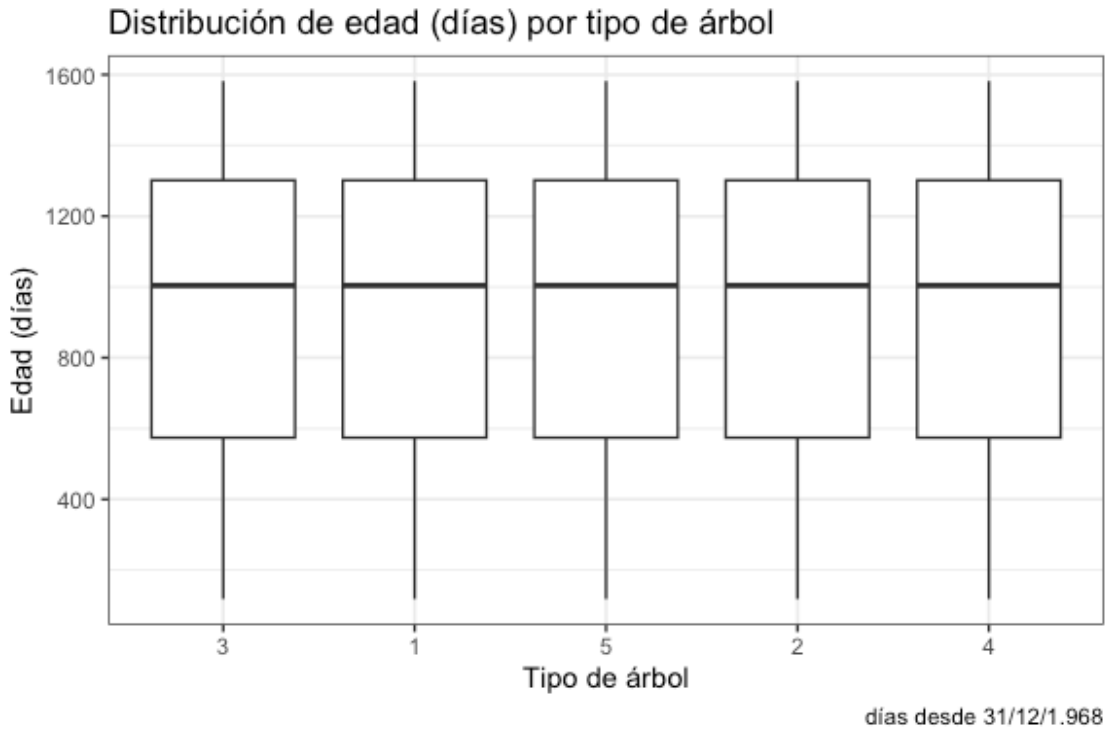


```
#
# para mostrar los gráficos apilados, cambiamos el operador.
p1/p2
```

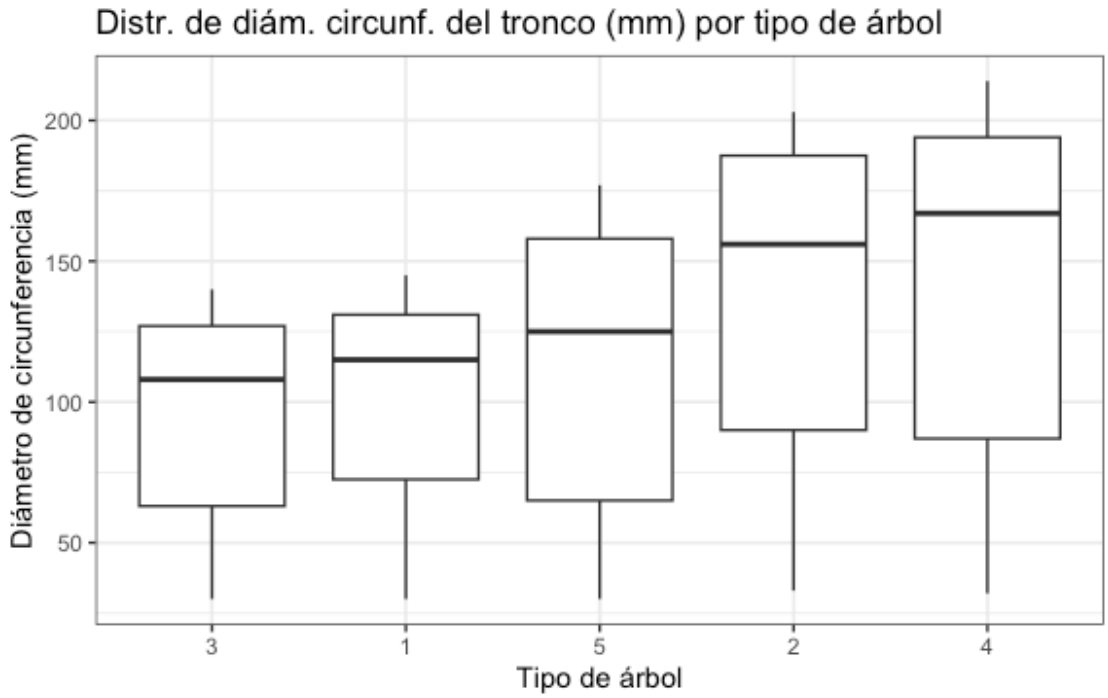


Para el caso de la edad y diámetro de circunferencia del tronco en función del tipo de árboles, podemos optar por *geom_boxplot()* de ggplot2(Wickham 2016c).

```
#
#    representamos edad y diámetro de circunferencia en funcio del tipo de árbol (datos$Tree)
# de nuestro estudio.
ggplot(data=datos)+
    geom_boxplot (aes(y=age, x=Tree),
                  outlier.color="red",
                  outlier.shape = 1)+
    labs( title = "Distribución de edad (días) por tipo de árbol",
          caption= " días desde 31/12/1.968")+
    xlab("Tipo de árbol")+
    ylab("Edad (días)")+
    theme_bw()
```

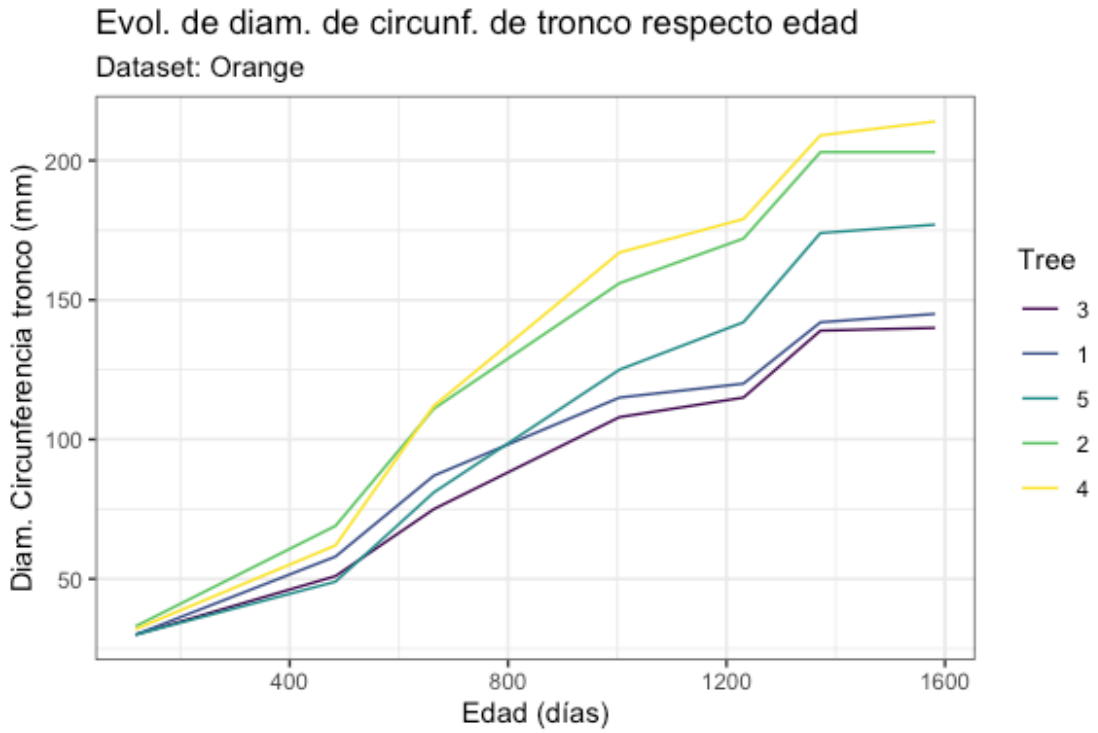



```
#
ggplot(data=datos)+
    geom_boxplot (aes(y=circumference, x=Tree),
                  outlier.color="red",
                  outlier.shape = 1)+
    labs( title = "Distr. de diám. circunf. del tronco (mm) por tipo de árbol",
          caption= "")+
    xlab("Tipo de árbol")+
    ylab("Diámetro de circunferencia (mm)")+
    theme_bw()
```



En el siguiente gráfico vemos mediante la variable *$Tree* (factor de 5 niveles) que codifica el tipo de árbol, cómo los arboles tipo 4, a partir de una determinada edad superan en diámetro de tronco al resto de tipo de árboles.

```
#
#    gráficamente como evoluciona con la edad el diametro de circunferencia del tronco
ggplot(data=datos)+
    geom_line(aes(x=age, y=circumference, color=Tree))+
    labs(title= "Evol. de diam. de circunf. de tronco respecto edad",
         subtitle= "Dataset: Orange")+
    xlab("Edad (días)")+
    ylab ("Diam. Circunferencia tronco (mm)")+
    theme_bw()
```



Comprobación de distribución Normal.

En algunos casos nos interesa conocer si nuestros datos siguen una distribución tipo Normal, comprobar si los valores que obtenemos están agrupados entorno a un valor central, llamado media. Si la distribución es Normal, los valores de la media y la mediana coinciden.

A efectos de demostración, consideraremos nuestros datos como un único grupo y comprobaremos si los datos de la medida del diámetro del tronco (*datos$circumference*) siguen una distribución normal

```
#
#       para la comprobación de la normalidad podemos optar por una visualización como el gráfico
# qqplot, donde se compara la distribución de los datos respecto a una linea teórica que
# deberían seguir si siguieran una distribución normal.
ggplot (data.frame(sample= datos$circumference), aes(sample=sample))+
    stat_qq(aes(color="Diam."))+
    stat_qq_line(linetype="dashed")+
    labs(title="QQ-Plot para verificar la nomalidad de Diam. Circumf.",
         subtitle="Dataset: Orange")+
    xlab ("Cuantiles teóricos")+
    ylab("Cuantiles de la muestra")+
```

```
  scale_color_manual(values=c("Diam."="blue"))+
  theme_bw()
```

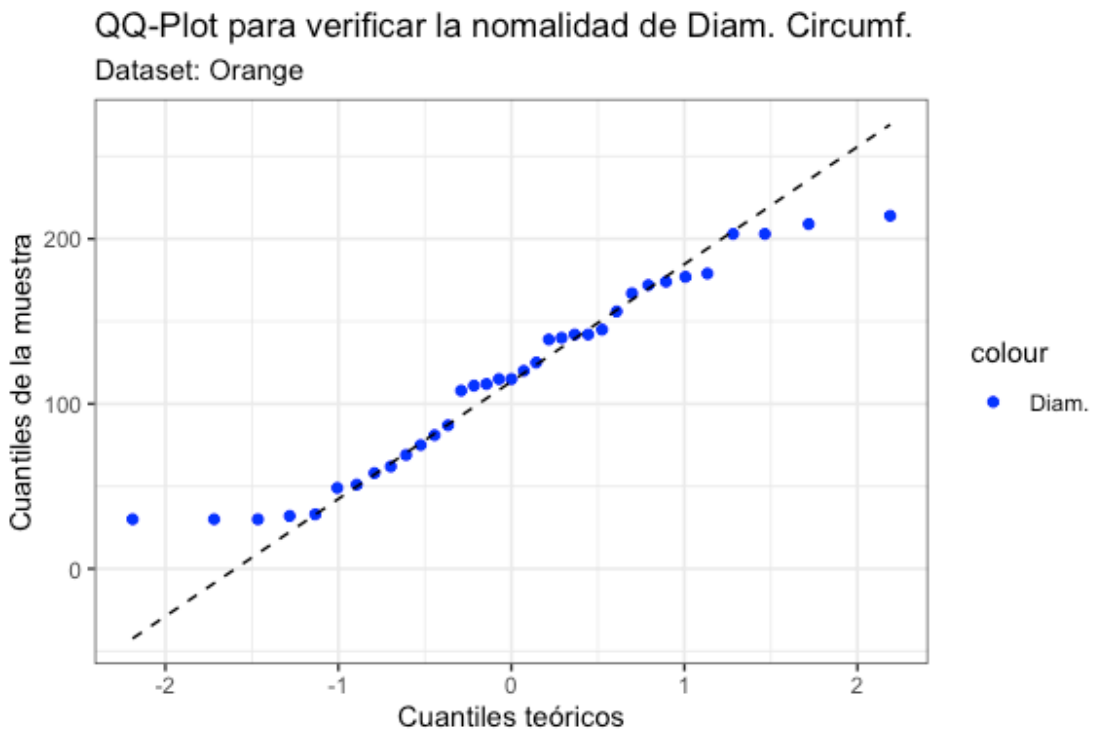
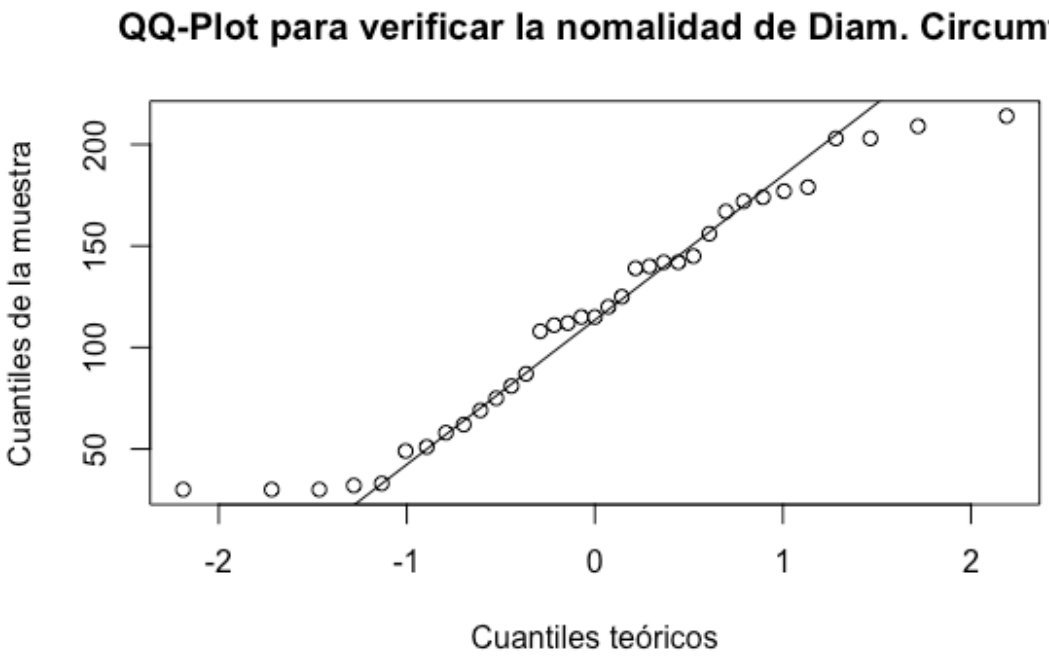


Figura 1: Diagrama QQ-Plot

En Figura 1 gráficamente vemos que, sobre todo en los extremos de la distribución los datos se alejan de una distribución tipo normal, representada por la linea diagonal discontinua.

R standard ofrece la función *qqnorm()*:

```
#
# que districución presenta la variable:
qqnorm (datos$circumference,
        main= "QQ-Plot para verificar la nomalidad de Diam. Circumf.",
        xlab= "Cuantiles teóricos",
        ylab="Cuantiles de la muestra")
# como sería si la distribución de la variable fuera tipo Normal
qqline(datos$circumference)
```



R nos ofrece varios test de normalidad para comprobar la condición de distribución normal numéricamente:

```
#
#    Test de Shapiro-Wilk
# ideal para muestras pequeñas (n<50)
# H0 = los datos vienen de una distribución normal
# si p-value es mayor que el nivel del significancia (usualmente 0,05), no se rechaza la
# hipotesis nula, lo que sugiere que los datos deben venir de una distribución normal
shapiro.test(datos$circumference)
```

```
    Shapiro-Wilk normality test

data:  datos$circumference
W = 0.94591, p-value = 0.08483
```

```
#    Test de Kolmogorov-Smirnov
# ideal para muestras grandes y pequeñas
# H0 = los datos no difieren significativamente de una distribución normal
# aceptación : Similar al test de Shapiro-Wilk
ks.test(datos$circumference, "pnorm")
```

```
Warning in ks.test.default(datos$circumference, "pnorm"): ties should not be
present for the one-sample Kolmogorov-Smirnov test

    Asymptotic one-sample Kolmogorov-Smirnov test

data:  datos$circumference
```

```
D = 1, p-value < 2.2e-16
alternative hypothesis: two-sided

# ...
```

La mayor dificultad con estos test numéricos es que parten del supuesto de normalidad de los datos. Como resultado, los datos deben seguir claramente otra distribución antes de que el bajo valor de *p-value* obtenido en el test de normalidad te haga rechazar la hipótesis nula. Esto hace que los test de normalidad sean muy conservadores, tendiendo al error en el aspecto de la normalidad de la distribución.

En relación a los valores de asimetría (en inglés skewness) y curtosis de la distribución podemos valorar la diferencia respecto a una distribución normal (en la cual los valores son skewness=0 y curtosis=3)

```
#
#    cálculo de la curtosis  y la asimetría del paquete momments, redondeado a tres decimales.
asimetria <- round(skewness(datos$circumference), 3)
curtosis <- round(kurtosis(datos$circumference), 3)
#
print(paste("Asimetría:", asimetria))
```

```
[1] "Asimetría: 0"
```

```
print(paste("Curtosis:", curtosis))
```

```
[1] "Curtosis: 1.866"
```

Normalmente, recurriremos a una combinación de métodos gráficos y numéricos para valorar la condición de normalidad de la distribución que tenemos en estudio.

Referencias

Allaire, J. J., Charles Teague, Yihui Xie, y Christophe Dervieux. 2022. «Quarto», enero. https://doi.org/10.5281/ZENODO.5960048.

Cui, Boxuan. 2024a. «DataExplorer: Automate Data Exploration and Treatment». https://CRAN.R-project.org/package=DataExplorer.

———. 2024b. «DataExplorer: Automate Data Exploration and Treatment». https://CRAN.R-project.org/package=DataExplorer.

Pedersen, Thomas Lin. 2024. «patchwork: The Composer of Plots». https://CRAN.R-project.org/package=patchwork.

Posit team. 2024. *RStudio: Integrated Development Environment for R*. Boston, MA: Posit Software, PBC. http://www.posit.co/.

R Core Team. 2024a. «R: A Language and Environment for Statistical Computing». https://www.R-project.org/.

———. 2024b. «R: A Language and Environment for Statistical Computing». https://www.R-project.org/.

Wickham, Hadley. 2016a. «ggplot2: Elegant Graphics for Data Analysis». https://ggplot2.tidyverse.org.

———. 2016b. «ggplot2: Elegant Graphics for Data Analysis». https://ggplot2.tidyverse.org.

———. 2016c. «ggplot2: Elegant Graphics for Data Analysis». https://ggplot2.tidyverse.org.