3.Manipulacion de datos

Paula Vargas Pellicer

15/02/2022

Empezando a trabajar con R

Importar y checar base de datos

Vamos a utilizar un conjunto de datos disponible públicamente de registros de ocurrencia para varias especies de animales, plantas y hongos (NBN Gateway). Descarga los registros de 2000-2016 de GitHub y guárdalos como edidiv.csv.

Ahora que tiene los datos guardados en tu computadora, impórtalos en RStudio, puedes hacer clic en el botón Importar base de datos y navegar hasta donde guardaste tu archivo, o usar el comando `read.csv()`. Si usas el botón, aparecerá una ventana con una vista previa de tus datos. Asegúrate de que junto a Encabezado (Header) hayas seleccionado Sí (esto le dice a R que trate la primera fila de sus datos como los nombres de las columnas) y haz clic en Importar. En la consola, verás el código para su importación, que incluye la ruta del archivo; es una buena idea copiar este código en tu secuencia de comandos, para que puedas consultar en el futuro de dónde provienen los conjunto de datos.

R funciona mejor con archivos .csv (valores separados por comas). Si ingresaste tus datos en Excel, deberás hacer clic en Guardar como y seleccionar csv como extensión de archivo. Al ingresar datos en Excel, no coloques espacios en los nombres de las filas, ya que confundirán R más tarde (por ejemplo, utiliza: altura_metros en lugar de altura (m). Algunas computadoras guardan archivos .csv con punto y coma, no comas, como los separadores. Esto suele suceder cuando el inglés no es el primer o único idioma en la computadora. Si tus archivos están separados por punto y coma, usa read.csv2 en lugar de read.csv, o alternativamente usa el argumento "sep" (para separador) en la función read.csv: r.csv("your-file-path", sep = ";").

```
edidiv <- read.csv("~/Downloads/CC_course_stream1-master/01_Getting_started/edidiv.csv")  # Esta es la ruta de archivo donde se guardo la base de datos, la tuya puede ser distinta
```

Recuerda guardar tu script de vez en cuando

Un paso importante es verificar que tus datos se hayan importado sin errores. Es una buena práctica ejecutar siempre este código y verificar el resultado en la consola. ¿Ves algún valor faltante, los números/nombres tienen sentido? Si pasas directamente al análisis, corre el riesgo de descubrir más tarde que R no leyó los datos correctamente y tener que volver a hacerlo, o peor aún, analizar datos incorrectos sin darse cuenta. Para obtener una vista previa de algo más que las primeras líneas, también puede hacer clic en

el objeto en el panel Entorno y aparecerá como una hoja de cálculo en una nueva pestaña junto a tu script abierto. Es posible que los archivos grandes no se muestren por completo, así que ten en cuenta que podrían faltar filas o columnas.

```
head(edidiv)                # Muestra las primeras filas
tail(edidiv)                # Muestra las ultimas filas
str(edidiv)                 # Muestra la estructura de las variables (integer
, continuo, categorico, caracter)
```

Notarás que la variable `taxonGroup` se muestra como una variable de caracteres, pero debería ser un factor (variable categórica). Cuando desees acceder a una sola columna de un marco de datos, agregua el nombre de la variable al nombre del objeto con un signo de dólar $. Esta sintaxis te permite ver, modificar y/o reasignar esta variable.

```
head(edidiv$taxonGroup)
class(edidiv$taxonGroup)
edidiv$taxonGroup <- as.factor(edidiv$taxonGroup)
```

- Vuelve a escribir `class(edidiv$taxonGroup)` ¿Qué clase tiene ahora?
- Si notas, la variable `year` es integer, también cámbiala a factor

Calcular la riqueza de especies

Nuestro objeto `edidiv` tiene registros de ocurrencia de varias especies recolectadas en Edimburgo desde 2000 hasta 2016. Para explorar la biodiversidad de Edimburgo, crearemos un gráfico que muestre cuántas especies se registraron en cada grupo taxonómico. Podríamos calcular la riqueza de especies en Excel, pero eso tiene varias desventajas, especialmente cuando trabajamos con grandes conjuntos de datos como el nuestro (25,684 observaciones): no tenemos registro de en qué hicimos clic, cómo clasificamos los datos y (lo mas grave) qué copiamos/eliminamos; los errores pueden pasar desapercibidos sin que te des cuenta. En R, por otro lado, tenemos un "script", que nos permite regresar y verificar todos los pasos en el análisis.

La riqueza de especies es simplemente el número total de especies diferentes en un lugar o grupo determinado. Para saber cuántas especies de aves, plantas, mamíferos, etc. tenemos en Edimburgo, primero debemos dividir `edidiv` en varios objetos, cada uno con filas para un solo grupo taxonómico. Hacemos esto con la función `filter()` del paquete `dplyr`.

Si no tienes instalado el paquete, debes instalarlo antes usando `install.packages("dplyr")` y cargarlo

```
Beetle <- filter(edidiv, taxonGroup == "Beetle")

# En esta funcion, el primer argumento es la base de datos, el segundo es la
condicion que quieres usar para filtrar. Solo queremos, en este caso, a los e
scarabajos, entonces pedimos que la variable elija los campos que concuerdan
EXACTAMENTE (==) con Beetle, todo lo demas lo quita.
# OJO. R es altamente sensible a errores de ortografia, por lo que la palabra
beetle (sin mayusculas) no va a funcionar en este caso).

Bird <- filter(edidiv, taxonGroup == "Bird")  # Se repite para aves
```

- Ahora realiza estos pasos para TODOS los taxones en los datos (plantas con flor, marchantiophytas, hongos, mamiferos, mariposas, hymenopteros, moluscos, libelulas, liquen)

Una vez que hayas creado objetos para cada taxón, podemos calcular la riqueza de especies, es decir, el número de especies diferentes en cada grupo. Para ello anidaremos dos funciones juntas: `unique()`, que identifica diferentes especies, y `length()`, que las cuenta. Puedes probarlos por separado en la consola y ver qué te devuelven

```
a <- length(unique(Beetle$taxonName))

b <- length(unique(Bird$taxonName))

# Puedes usar cualquier nombre, yo elegi a,b,c......
```

- Si escribes a (o como hayas nombrado tus variables de conteo) en la consola, ¿qué devuelve? ¿Qué significa eso? Debe representar el número de especies distintas de escarabajos en el registro.
- Nuevamente, calcula la riqueza de especies para los otros taxones en el conjunto de datos. ¡Probablemente estés notando que esto es bastante repetitivo y usa mucho copiar y pegar! Eso no es particularmente eficiente: en futuras clases, aprenderemos cómo usar más funciones de `dplyr` y lograr el mismo resultado con mucho menos código. Podrás hacer todo lo que acabas de hacer en UNA línea (¡lo prometo!).

Crea un vector y una gráfica

Ahora que tenemos la riqueza de especies para cada taxón, podemos combinar todos esos valores en un vector. A diferencia de un marco de datos, que tiene dos dimensiones (filas y columnas), un vector solo tiene una. Cuando llamas a una columna de un marco de datos como hicimos anteriormente con `edidiv$taxonGroup`, esencialmente estás produciendo un vector, pero se pueden crear desde cero.

Hacemos esto usando la función `c()` (c significa concatenar, o cadena si eso lo hace más fácil de recordar). También podemos agregar etiquetas con la función `names()`, para que los valores no salgan de la nada.

```
biodiv <- c(a,b,c,d,e,f,g,h,i,j,k)     # Estamos encadenando los valores; el
orden en que los pongas, es el orden en que debes nombrarlos

names(biodiv) <- c("Beetle",
                   "Bird",
                   "Butterfly",
                   "Dragonfly",
                   "Flowering.Plants",
                   "Fungus",
                   "Hymenopteran",
                   "Lichen",
                   "Liverwort",
                   "Mammal",
                   "Mollusc")
```

Algunos detalles:

1. Los espacios delante y detrás de `<-` y después de `,` se agregan para facilitar la lectura del código.

2. Todas las etiquetas se han escrito en una nueva línea; de lo contrario, la línea de código se vuelve muy larga y difícil de leer.

3. Ten cuidado de verificar que los valores coinciden con las etiquetas de su vector correctamente; una mejor práctica hubiera sido dar nombres más significativos a nuestros objetos, como beetle_sp, bird_sp, etc.

4. Si resaltas un paréntesis con el mouse, R Studio resaltará el correspondiente en su código. Los paréntesis que faltan, especialmente cuando comienzas a anidar funciones como hicimos anteriormente con length(unique()) son una de las fuentes más comunes de frustración y error cuando comienzas a programar.

Ahora podemos visualizar la riqueza de especies con la función `barplot()`. Los gráficos aparecen en la ventana inferior derecha de RStudio.

```
barplot(biodiv)
```

Hay algunas cosas que no están del todo bien y que debemos corregir: no hay títulos de eje, no todas las etiquetas de columna son visibles y el valor de las especies de plantas (n = 521) excede el valor más alto en el eje y, por lo que necesitamos para extenderlo.

```
barplot(biodiv, xlab="Taxa", ylab="Number of species", ylim=c(0,600),
cex.names= 1.5, cex.axis=1.5, cex.lab=1.5)
```

Checa la pagina de ayuda de `barplot()` para conocer cada elemento.

Crea una base de datos

En la sección anterior creamos vectores, es decir, una serie de valores, cada uno con una etiqueta. Este tipo de objeto es adecuado cuando se trata de un solo conjunto de valores. Sin embargo, a menudo tendrás más de una variable y varios tipos de datos, por ejemplo, algunos continuos, otros categóricos. En esos casos, usamos objetos de base de datos. Las beses de datos son tablas de valores: tienen una estructura bidimensional con filas y columnas, donde cada columna puede tener un tipo de datos diferente. Por ejemplo, una columna llamada “Envergadura” tendría valores numéricos medidos en diferentes aves (21.3, 182.1, 25.1, 8.9), y una columna “Especies” tendría valores de caracteres con los nombres de las especies (“Gorrión común”, “Águila real”, “Martín pescador euroasiático”, “Colibrí garganta rubí”)

Otro posible formato de datos es una matriz: una matriz también puede tener varias filas de datos (por ejemplo, puede combinar vectores en una matriz), pero las variables deben ser todos del mismo tipo. Por ejemplo, todos son numéricos y tienen la misma longitud en cuanto al número de filas.

Una nota: SIEMPRE guarda una copia de tus datos sin procesar tal como los recopilaste por primera vez. La belleza de manipular un archivo en un script R es que las modificaciones viven en el script, no en los datos. Dicho esto, si escribiste un código largo para ordenar un gran conjunto de datos y prepararlo para el análisis, es posible que no desees volver a ejecutar todo el script cada vez que necesitas acceder a los datos limpios. Por lo tanto, es una buena idea guardar tu nuevo objeto como un nuevo archivo csv que puedes cargar, listo para usar, con solo un comando. Ahora crearemos un marco de datos con nuestros datos de riqueza de especies y luego lo guardaremos usando `write.csv()`.

Usaremos la función `data.frame()`, pero primero crearemos un objeto que contenga los nombres de todos los taxones (una columna) y otro objeto con todos los valores de la riqueza de especies de cada taxón (otra columna).

```
# Crea un objeto llamado "taxa" que contiene todos los nombres de los taxones
taxa <- c("Beetle",
          "Bird",
          "Butterfly",
          "Dragonfly",
          "Flowering.Plants",
          "Fungus",
          "Hymenopteran",
          "Lichen",
          "Liverwort",
          "Mammal",
          "Mollusc")
```

- Cambia este objeto a una variable categorica (factor) llamada `taxa_f`
- Combina todos los valores del numero de especies (a,b,c...) en un objeto llamado `richness`
-

```
# Crea la base de datos para los dos factores
biodata <- data.frame(taxa_f, richness)

# Guardalo en un archivo
write.csv(biodata, file="biodata.csv")  # Se va a guardar en tu directorio de
trabajo
```

Si deseas conocer tu directorio de trabajo R actual (o predeterminado), escribe el comando `getwd()`, que significa "obtener directorio de trabajo".

Si queremos crear una grafica de barras usando la base de datos, debemos cambiar ligeramente el código, ya que las bases de datos pueden contener múltiples variables. Debemos decirle a R exactamente cuál queremos que represente. Como antes, podemos especificar columnas de un marco de datos usando $:

```
barplot(biodata$richness, names.arg=c("Beetle",
                                      "Bird",
                                      "Butterfly",
                                      "Dragonfly",
                                      "Flowering.Plants",
                                      "Fungus",
```

```
                                    "Hymenopteran",
                                    "Lichen",
                                    "Liverwort",
                                    "Mammal",
                                    "Mollusc"),
        xlab="Taxa", ylab="Number of species", ylim=c(0,600))
```

Ejercicio

Estos son valores (ficticios) de la envergadura (en cm) medidos en cuatro especies diferentes de aves. * Produce un gráfico de barras de la envergadura media de cada especie y guardarlo en tu computadora? (¿Cuál podría ser la función para calcular la media?)

bird_sp	wingspan
sparrow	22
kingfisher	26
eagle	195
hummingbird	8
sparrow	24
kingfisher	23
eagle	201
hummingbird	9
sparrow	21
kingfisher	25
eagle	185
hummingbird	9