Tema II: Importación de datos.

Maicel Monzón

Pregunta del encuentro anterior

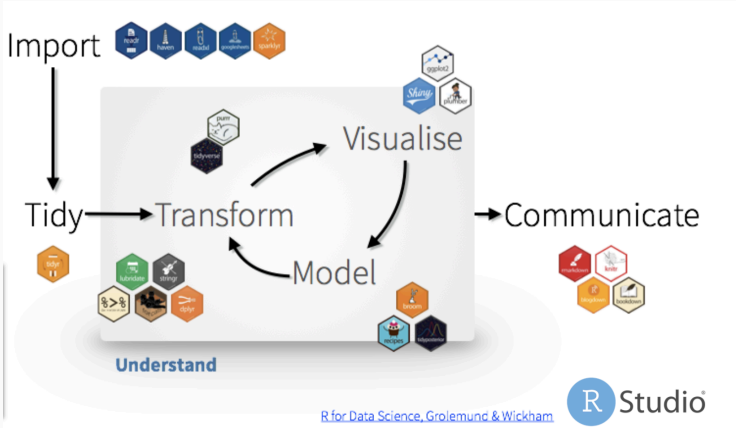
Como se crea un objeto en R?

Respuesta

Los objetos se crean:

- Leyendo datos de un archivo
- Como resultado de un cálculo
- “Asignándoles un valor”
- etc.

Lectura de datos en el proceso de ciencia de datos



Qué datos se pueden leer desde R?

- **Datos estructurados** *(estructura interna identificable filas, columnas con títulos)*

– archivos rectangulares en texto plano (csv,dat,txt)

- **Datos no estructurados** *(datos binarios que no tienen estructura interna identificable)*

– Correos electrónicos, publicaciones en redes sociales (Hojas de cálculo, ficheros SPSS, Stata, SAS, Imágenes digitales, sonido, video, etc.)

Bibliotecas especializadas leer datos

Paquete	formatos
readr*	datos rectangulares (csv, tsv y fwf)
haven	ficheros (SPSS, Stata y SAS)
readxl	ficheros Excel (.xls y .xlsx)
DBI	bases de datos
jsonlite	json
xml2	XML
httr	Web APIs
rvest	HTML (Web Scraping)

Funiones para lectura de datos tabulares a tibbles (readr)

Funiones	delimitador
read_delim()	cualquier
read_csv()	coma
read_csv2()	punto y coma
read_tsv()	tabulaciones
read_fwf()	ancho fijo
fwf_widths()	ubicación

Algunas convenciones

path o camino (ruta hacia un archivo)

“./carpeta/archivo.ext”

separador entre carpetas es una barra inclinada (/),

(./) Espacio de trabajo

```
# obtiene el espacio de trabajo
ws <- getwd()
# contruye una ruta
file.path(".","datos","mifichero.csv")
```

Argumentos de las funciones readr()

file -cadena de texto- Argumento obligatorio

```
read_csv(file = "a,b,c \n
         1,2,3  \n
         4,5,6")
```

```
# A tibble: 2 x 3
      a     b     c
  <dbl> <dbl> <dbl>
1     1     2     3
2     4     5     6
```

Argumetos de las funciones readr

skip -omitir las primeras n líneas-

```
read_csv(file = "La primer línea de metadata
  La segunda línea de metadata
  x,y,z
  1,2,3",
  skip = 2,
  col_types ="i" )
```

Argumetos de las funciones

readr (col_names) -nombres de columna booleano

```
read_csv(file ="41,masculino, blanca
         40, femenino, negra",
         col_names = FALSE)
```

Argumetos de las funciones readr (col_name) -Asignación de nombres de columna-

Argumento col_names -vector de caracteres-

```
read_csv(file ="41,masculino,blanca\n
         40, femenino,negra",
         col_names = c("edad","sexo","color de la piel"))
```

Argumetos de las funciones readr (na) -valores faltantes-

Argumento na vector de caracteres

```
read_csv(file ="a,b,c\n1,2, ", na = " ")
```

Analizar un vector funciones parse_*()

funciones toman un vector de caracteres y devuelven un vector más especializado, como un vector lógico, numérico, o fecha:

- parse_logical()
- parse_integer()
- parse_double()
- parse_number()
- parse_character()
- parse_factor()
- parse_datetime()
- parse_date()

Parseo de números (Ejemplos)

1. uso diferente de decimales *Ej coma o punto*

   `hemoglobina1<-c("12,2","13.5","11.9")` # diferente marca decimal

2. números están rodeados por otros caracteres *Ej unidades*

   `hemoglobina2<-c("12.2","13.5","11.9 g/l")` # texto adicionado

3. caracteres de “agrupación” *Ej “1,000,000”.*

Ejemplo 1 parse_double (locale) uso diferente de decimales

locale -objeto que especifica las opciones de análisis que difieren de un lugar a otro-

decimal_mark “símbolo decimal”

```
parse_double("1.23")

parse_double("1,23",
             locale = locale(decimal_mark = ","))
```

Ejemplo 2 parse_number ignora los caracteres no-numéricos antes y después del número

```
parse_number("123 g/l")

parse_number("la hemoglobina es de 123 g/l")
```

Ejemplo 3 combinación de parse_number() y el locale

```
parse_number("123.456.789",
             locale = locale(grouping_mark = "."))
```

Parseo de Cadenas de caracteres (parse_character())

```
x1 <- "El Ni\xf1o"

parse_character(x1,
                locale = locale(encoding = "Latin1"))
```

Parseo de factores (parse_factor())

Factores

utiliza factores para representar las variables categóricas que tienen un conjunto conocido de valores posibles

```
fruta <- c("manzana", "banana")

parse_factor(c("manzana", "banana", "bananana"), levels = fruta)
```

Parseo de fechas (parse_date())

- Año : %y (2 dígitos)
- Mes : %m (2 dígitos).
- Mes : %b (nombre abreviado, como “ene”).
- Mes :: %B (nombre completo, “enero”).
- Dias : “%d” (2 dígitos)

```
parse_date("01/02/15", "%m/%d/%y")

parse_date("01/02/15", "%d/%m/%y")

parse_date("01/02/15", "%y/%m/%d")

parse_date("01-02-15", "%y-%m-%d")
```

Parseo de tiempos (parse_time())

```
parse_time("20:10:01")
```

readr deduce automáticamente el formato de cada columna para las primeras 1000 filas (guess_parser).

```
guess_parser("2010-10-01")

guess_parser("15:01")

guess_parser(c("TRUE", "FALSE"))

guess_parser(c("1", "5", "9"))

guess_parser(c("12,352,561"))

guess_parser("12.3")
```

Detección automatica (1000 primeras filas)

guess_parser casi siempre detecta los tipos correctos, sin embargo, puede no ser así en todos los casos.

Identificación de problemas de parseo

extraer explícitamente los problemas con la función problems()

```
desafio <- read_csv(readr_example("challenge.csv")) # encuentra la ruta a u

problems(desafio)
```

Estrategia para solución de problemas (paso 1)

1. Comienzó copiando y pegando la especificación de columna en tu llamada a readr, simpre mostrara el resultado de la identificación automática.

```
desafio <- read_csv(
  readr_example("challenge.csv"))
```

Estrategia para solución de problemas (paso 2)

2. Luego puedes ajustar el tipo de la columna x (modificando la llamada del paso anterior)

```
desafio <- read_csv(
  readr_example("challenge.csv"),
  col_types = cols(
    x = col_double(),
    y = col_character()
  )
)
```

Anular las opciones predeterminadas de readr (argumento col_types)

Cada función parse_xyz() tiene su correspondiente función col_xyz().

- parse_xyz() - para vector de caracteres que ya está disponible en R
- col_xyz - cuando quieres decirle a readr cómo cargar los datos

Utilice el argumento col_types para anular las opciones predeterminadas.