Introducción a R

Sesión 3

Agosto 2022

Índice

Paquetería tidyr:

- Limpieza de datos
- Mutate
- unique
- topper

Material de trabajo

Carpeta de drive:

https://drive.google.com/drive/u/0/folders/14QDyRnqnLUjosCBlAzBqXL-qglOMjNJL

Base a descargar ile_completa

Liga de zoom https://us02web.zoom.us/j/83841692580

ID de reunión: 838 4169 2580

Que hemos visto:

- Paneles en R studio
- Estructura de datos
- Paqueterías o librerias
- Importar o subir archivos
- Función unique(), str(), View()
- Simbolo "$"

Más formas de subir o importar archivos:

Hemos visto read_excel de la paquetería readlx

- **Archivos xlxs**

```
library(readxl)
```

```
#Cargar datos de un archivo de extension xlsx
datos <- read_excel("~/ile_completa.xlsx")
```

- **Leer desde una nube o hoja de sheet**

```
#datos<-read.csv("https://docs.google.com/spreadsheets/d/e/2PACX-1vRt8YMj-fA6OqTnS
```

- **Archivos csv**

```
#Cargar datos de un archivo de extension csv
datos<-read.csv("C:/Users/Paola Viridiana/Desktop/ile_completa.csv", encoding="UTF
```

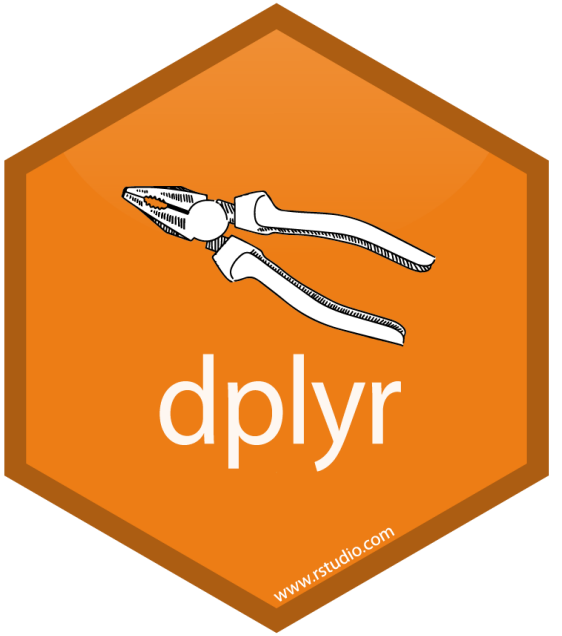
Encoding o codificación de caracteres

¿Quién no ha visitado alguna vez un sitio web con signos extraños reemplazando las eñes, las letras con acento u otros caracteres especiales? Como desarrolladores, ¿cuántas veces hemos encontrado que al subir contenido de texto a nuestros sitios aparecen esos caracteres raros? Si uno no los puso allí, ¿por qué aparecen?

El encoding («codificación» en inglés) es el proceso a través del cual se transforma información textual humana (caracteres alfabéticos y no alfabéticos) en un conjunto más reducido, para ser almacenado o transmitido.

Todo esto para explicar que la codificación que lee *acentos* y *caracter como ñ* se utiliza el encoding "UTF-8".

Este influye al guardar bases de datos y script



dplyr

Paquetería dplyr

Manipulación de datos con 'dplyr'.
El paquete **dplyr** proporciona un conjunto de funciones extremadamente útiles para manipular data frames y así reducir el número de repeticiones, la probabilidad de cometer errores y el número de caracteres que hay que escribir. Como valor extra, puedes encontrar que la gramática de dplyr es más fácil de entender.

Aquí vamos a revisar 6 de sus funciones más usadas, así como a usar los **pipes (%>%)** para combinarlas.

- select()
- filter()
- group_by()
- summarize()
- mutate()

```
library(dplyr)
```

El operador pipe %>%

El operador pipeline %>% es útil para concatenar múltiples dplyr operaciones. Obsérvese en el siguiente ejemplo, que cada vez que queremos aplicar mas de una función, la instrucción es una secuencia de llamadas a funciones de forma anidada:

Seleccionar con pipes

```
head(datos %>% select(year, mes),5)
```

```
##   year   mes
## 1 2016 ENERO
## 2 2016 ENERO
## 3 2016 ENERO
## 4 2016 ENERO
## 5 2016 ENERO
```

Se inserta con ctrl + tecla flecha arriba + M

Empezamos con la limpieza de datos

Limpieza de datos

La limpieza de datos (en inglés data cleansing o data scrubbing) es el acto de descubrimiento y corrección o eliminación de registros de datos erróneos de una tabla o base de datos. El proceso de limpieza de datos permite identificar datos incompletos, incorrectos, inexactos, no pertinentes, etc. y luego substituir, modificar o eliminar estos datos sucios ("data duty"). Después de la limpieza, la base de datos podrá ser compatible con otras bases de datos similares en el sistema.

La limpieza de datos es la parte que más horas se le asigna en un análisis y procesamiento de datos aldedor del 60%-70%. Es la talacha en otras palabras.

Reclasificación de observaciones de una variable

En la data de Ile_completa se observan diferentes observaciones que se pueden reclasificar para agrupar una variable, por ejemplo:

En la variable **nivel_edu** Primaria~ Secundaria incompleta se podría reclasificar estas dos observaciones como **primaria.**

unique()

Extrae los valores unicos de una variable o columna.

mutate()

La función mutate() sirve para crear nuevas variables.

case_when

Permite definir una variable, la cual toma un valor particular para cada condición establecida. En caso de no cumplir ninguna de las condiciones establecidas la variable tomara valor NA. La función mutate() sirve para crear nuevas variables.

toupper

Con esta función se convierte a mayúscula una columna o varias

```
# con signo "$" se especifica una variable
datos$mes<- toupper(datos$mes)
```

Ejemplo de reclasificación - variable edocivil_descripcion

Paso 1: extraer valores unicos.

```
unique(datos$edocivil_descripcion)
```

```
##  [1] "soltera"              "unión libre"          "casado(a)"
##  [4] "divorciada"           "N/E"                  "no sabe/ sin respuesta"
##  [7] "casada (o)"           "separada"             "Soltera"
## [10] "Viuda"                "SOLTERA"              "VIUDA"
## [13] "viuda"                "S/INFORMACION"        "SE DESCONOCE"
## [16] "divociada"            "NO RESPONDE"          "UNION LIBRE"
## [19] "CASADA"               "SEPARADA"             "DIVORCIADA"
## [22] "soltero(a)"           "divorciado(a)"        "viudo"
## [25] "union libre"          "separado (a)"         NA
## [28] "Casada"               "Divorciada"           "Separada"
## [31] "UNIÓN LIBRE"          "SOLTERO/A"            "SEPARADO/A"
## [34] "DIVORCIADO/A"         "casada"               "SIN ESPECIFICAR"
## [37] "Sin especificar"
```

Ejemplo de reclasificación - variable edocivil_descripcion

Paso dos: pasar a mayúsculas.

```
# con signo "$" se especifica una variable
datos$edocivil_descripcion<- toupper(datos$edocivil_descripcion)
```

vemos valores unicos ahora que estan en mayúsculas

```
unique(datos$edocivil_descripcion)
```

```
##  [1] "SOLTERA"              "UNIÓN LIBRE"          "CASADO(A)"
##  [4] "DIVORCIADA"           "N/E"                  "NO SABE/ SIN RESPUESTA"
##  [7] "CASADA (O)"           "SEPARADA"             "VIUDA"
## [10] "S/INFORMACION"        "SE DESCONOCE"         "DIVOCIADA"
## [13] "NO RESPONDE"          "UNION LIBRE"          "CASADA"
## [16] "SOLTERO(A)"           "DIVORCIADO(A)"        "VIUDO"
## [19] "SEPARADO (A)"         NA                     "SOLTERO/A"
## [22] "SEPARADO/A"           "DIVORCIADO/A"         "SIN ESPECIFICAR"
```

Paso 3:reclasificar obsevaciones con case_when cuando no se espeficica alguna obsevación por default se pasa a NA o valor perdido.

```
datos<- datos %>%
  mutate(
    EDO_CIVIL_DESCRIPCION_LIMPIA = case_when(
      edocivil_descripcion == "CASADO(A)" ~ "CASADA",
      edocivil_descripcion == "SOLTERA" ~ "SOLTERA",
      edocivil_descripcion == "UNIÓN LIBRE" ~ "UNIÓN LIBRE",
      edocivil_descripcion == "DIVORCIADA" ~ "DIVORCIADA",
      edocivil_descripcion == "CASADA (O)" ~ "CASADA",
      edocivil_descripcion == "UNION LIBRE" ~ "UNIÓN LIBRE",
      edocivil_descripcion == "SOLTERO(A)" ~ "SOLTERA",
      edocivil_descripcion == "DIVORCIADO(A)" ~ "DIVORCIADA",
      edocivil_descripcion == "VIUDO" ~ "VIUDA",
      edocivil_descripcion == "SEPARADO (A)" ~ "SEPARADA",
      edocivil_descripcion == "SOLTERO/A" ~ "SOLTERA",
      edocivil_descripcion == "CASADA" ~ "CASADA",
      edocivil_descripcion == "SOLTERO(A)" ~ "SOLTERA",
      edocivil_descripcion == "SEPARADO/A" ~ "SEPARADA",
      edocivil_descripcion == "DIVORCIADO/A" ~ "DIVORCIADA"
    ))
```

Esta crea una nueva variable llamada EDO_CIVIL_DESCRIPCION_LIMPIA se añadé a la base en la última columna.