Curso Primeros Pasos en R

Clase 5: Introducción a Tidyverse

Pontificia Universidad Católica de Chile

Noviembre 2021

Clase 5: Introducción a Tidyverse

- Introducción a `tidyverse`
- Sintaxis en `tidyverse`
- Funciones `dplyr`
- Estructura de control
- Actividad 1 `dplyr`
- Actividad 2 `dplyr`

Anexos

- Comando `if`.
- Comando `else`.
- Comando `ifelse`.

Paquete `tidyverse`

Paquete `tidyverse`

El `tidyverse` es un universo de paquetes para Data Science en R. Algunos de sus paquetes centrales son:

- **ggplot2 :** Visualización de datos.
- **dplyr :** Manipulación de datos.
- **tidyr :** Herramientas para ordenar datos.
- **tibble :** Reinvención de data frames.
- **stringr :** Manejo con cadenas de texto (strings).
- **readr :** Importación de datos.

Y muchos más: **purrr**, **forcats**, **lubridate**, **readxl**, **haven**, etc.

Instalación/carga de tidyverse.

En la clase anterior instalamos el paquete `tidyverse`. Para cargarlo en nuestra sesión tenemos que usar el comando `library()`. (Si estás usando RStudioCloud tendrán que instalarlo nuevamente):

```
install.packages("tidyverse") # Instalar tidyverse

library(tidyverse) # Cargarlo en la sesión
```

Esto carga los ocho paquetes centrales del `tidyverse`: `dplyr, tidyr, tibble, ggplot2, purrr, stringr, forcats, readr`. El resto de los paquetes es necesario cargarlos de manera individual.

Sintaxis en Tidyverse

Sintaxis en Tidyverse

La sintaxis del `tidyverse` se ha popularizado en los últimos años porque permite leer el código y programar de forma más parecida a como leemos (de izquierda a derecha y de arriba hacia abajo). Esto es posible gracias al operador "pipe" `%>%` (Ctrl/Cmd + Shift + M), que funciona de la siguiente forma:

funcion(objeto, ...)

objeto %>% funcion(...)

Ejemplo R base:

```
mean(paises[paises$anio == 2007,
            ]$poblacion)
```

Ejemplo `tidyverse`:

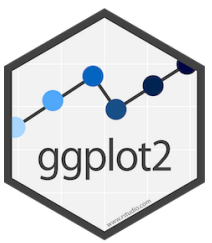
```
paises %>%
  filter(anio == 2007) %>%
  summarize(mean(poblacion))
```

Paquete `dplyr`

Paquete `dplyr`

`dplyr` es un paquete enfocado en la manipulación de datos. Es decir, nos permite realizar acciones como seleccionar columnas, filtrar filas, crear y ordenar variables, unir bases de datos, entre otras.

Paquete `ggplot2`



`ggplot2` es un paquete que permite la creación de gráficos en R. Es la herramienta de visualización más populr, dadas la gran cantidad de opciones de personalización que ofrece.

Funciones más usadas de `dplyr`

Funciones más usadas de `dplyr`

`select()` : Permite seleccionar variables (columnas).

```
base %>%
  select(nombre_variable)
```

`filter()` : Permite filtrar las observaciones de la base dado un criterio definido.

```
base %>%
  filter(<Condición asociada a una variable>)
```

`group_by()` : Agrupa filas a partir de categoría de una o más variables.

```
base %>%
  group_by(nombre_variable)
```

`summarise()` : Calcula un resumen a base a funciones de R.

```
base %>%
  summarise(nombre_calculo = <Función de una variable>, ...)
```

`mutate()` : Crea nuevas variables (o modifica las existentes).

```
base %>%
  mutate(nueva_variable = <Cálculo en base a una variable>,...)
```

`rename()` : Renombra columnas.

```
base %>%
  rename(nuevo_nombre = nombre_variable)
```

`count()` : Cuenta el número de observaciones en cada grupo.

```
base %>%
  count(nombre_variable)
```

`slice()` : Permite extraer observaciones (filas) de acuerdo a su posición.

```
base %>%
  slice(vector_de_filas)
```

`arrange()` : Ordena las filas según los valores de una columna.

```
base %>%
  arrange(nombre_variable)
```

Comando `case_when`

Comando `case_when`

El comando `case_when` de la librería `dplyr`, permite evaluar múltiples `if` e `ifelse` al mismo tiempo. Es el equivalente de la sentencia `CASE WHEN` de SQL. Este funciona de la siguiente forma:

```
case_when(<Condición 1> ~ <Resultado si condición 1 es TRUE>,
          <Condición 2> ~ <Resultado si condición 2 es TRUE>,
          ...)
```

Es un comando especialmente útil para recodificar variables categóricas.

Ejemplo: Definamos un `case_when()` que nos permita replicar el ejemplo anterior:

```
case_when(nota < 1 ~ "Error, ingrese un número entre 1 y 7.",
          nota > 7 ~ "Error, ingrese un número entre 1 y 7.",
          is.numeric(nota) == FALSE ~ "Error, ingrese un número.",
          nota >= 4 ~ "¡Felicitaciones!",
          nota <4 ~ "Reprobaste")
```

Actividad 1

Actividad 1

La base de datos `playstore.xlsx` contiene información sobre algunas aplicaciones para dispositivo móvil ofrecidas en la plataforma. Importe la base de datos `playstore.xlsx` y realice lo siguiente:

1.- Ordene según la variable `Tamaño`

2.- Filtre por las aplicaciones que son *pagadas*

3.- Agregue la variable `Ganancias` definidas como el precio ponderado por la cantidad de descargas

4.- Agrupe por `Categoría`

5.- Obtenga la `Ganancia` promedio por las distintas `Categorías`

Actividad 2

Actividad 2 propuesto

Vuelva a utilizar la base de datos `viviendasRM.xlsx` que contiene avisos de viviendas usadas que se venden en la Región Metropolitana de Chile. El objetivo es replicar la actividad de hace unas clases, pero con la lógica de dplyr, es decir:

Usando funciones de dplyr junto con el operador pipe `%>%` indique:

a. Toda la información del cuarto registro.

b. El número de baños de la vivienda del registro 8.

c. El numero de estacionamientos de la vivienda 14.

d. Total de superficie M2 y valor en UF, para las viviendas desde la quinta a la décima.

e. Listado de superficie M2, corredor y valor en UF de viviendas cuya superficie M2 es igual a 100 m2.

Anexos

Anexos

En programación, una estructura de control es un tipo de comando que nos permite controlar cuándo se ejecuta nuestro código, mediante sentencias condicionales, las cuales se definen usando operadores lógicos.

Ejemplos de sentencias condicionales:

Sentencia condicional	Código
Es un número es mayor o igual a 4.	numero >= 4
Es un número par.	numero %% 2 == 0
Nació antes del 1998	anho_nacimiento < 1998
Tiene educación superior completa.	nivel_edu == "Superior Completa"

Comando `if`

El comando `if()` se usa para ejecutar un código sólo cuando se cumpla una condición específica. Es equivalente a un si lógico (sí), esto se usa de la siguiente forma:

```
if( Condición ){
  Código a ejecutar si la condición es VERDADERA
}
```

Ejemplo: Si la calificación de un estudiante es **mayor o igual a 4**, entonces imprima en la consola un mensaje de felicitaciones.

```
if(nota >= 4){
  print("¡Felicitaciones! :)")
}
```

Comando `else`

Comando `else`

El comando `else()` se usa para añadir condiciones a un if. Equivale a decir "en otro caso", esto se usa de la siguiente forma:

```
if( Condición ){
  Código a ejecutar si la condición es VERDADERA
}else{
  Código a ejecutar si la condición es FALSA
}
```

Ejemplo: Si la calificación de un estudiante es **mayor o igual a 4**, imprima un mensaje de felicitaciones. En otro caso, ejecute un mensaje de reprobación.

```
if(nota >= 4){
  print("¡Felicitaciones! :)")
}else{
  print("Reprobaste :(")
}
```

Comando `ifelse`

Comando `ifelse`

El comando `ifelse()` acorta la escritura anterior, pudiendo escribir un `if()` y un `else()` con sólo un comando. Este se usa de la siguiente forma:

```
ifelse(Condición,
       Código si condición es VERDADERA,
       Código si condición es FALSA)
```

Con `if()` y `else()`

```
if(nota >= 4){
  print("¡Felicitaciones!")
}else{
  print("Reprobaste")
}
```

Con `ifelse()`

```
ifelse(nota >= 4,
       print("¡Felicitaciones!"),
       print("Reprobaste") )
```

¿Qué pasa si ingresamos una nota negativa o una nota mayor a 7? ¿Tiene sentido, en el primer caso, considerar un error de tipeo como un "reprobado"?

Encadenar `if` y `else`

Se puede encadenar `if()` y `else()` para generar condiciones más robustas, por ejemplo, en el caso anterior se puede añadir una condición que entregue un error si se ingresa un número fuera del rango o un objeto que no sea numérico:

```
if((nota < 1) | (nota > 7) | (is.numeric(nota) == FALSE)) {
  print("Error, ingrese un número entre 1 y 7")
}else if (nota >= 4) {
    print("¡Felicitaciones!")
  } else{
    print("Reprobaste")
  }
}
```

Tambien, se puede hacer `ifelse()`:

```
ifelse( (nota < 1) | (nota > 7) | (is.numeric(nota) == FALSE),
       print("Error, ingrese un número entre 1 y 7"),
       ifelse(nota >= 4,
              print("¡Felicitaciones!"),
              print("Reprobaste")))
```

Referencias y material complementario

Capítulo sobre visualización de datos del libro "R para Ciencia de Datos"`, de Hadley Wickham y Garrett Grolemund.

Cheatsheet Domar Datos con `dplyr` y `tidyr`, Traducido por Frans van Dunné.

Cheatsheet Transformación de Datos con `dplyr`

¡Gracias!

Diego Muñoz
dimunoz1@uc.cl

Felipe Moya
felipe.moya@uc.cl