Curso Primeros Pasos en R

Clase 3: Manipulación e importación de datos

Profesor: Diego Muñoz Ureta

Pontificia Universidad Católica de Chile

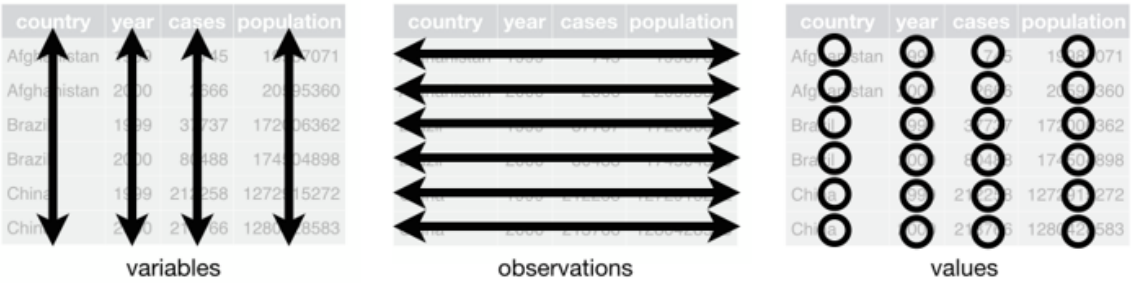
Noviembre 2021

Clase 3: Manipulación e importación de datos

- Dataframe: Manipulación de DF y creación de DF
- Tibble: ventajas y creación de un tibble
- Importación y exportación de datoss
- Análisis exploratorio de datos
- Problemas en la importación de datos
- Actividad

Data frames y tibble

En el curso trabajaremos principalmente con bases de datos ordenadas:



Cuando carguemos bases de datos en R, estas pueden ser de dos tipos: `data frame` o `tibble`.

Las funciones `tibble()` y `as_tibble()` del paquete `tibble` nos permiten crear este tipo de objetos y transformar un data frame a este formato.

Data frame

Creación de DF

Los data frame en R se pueden generar directamente con el comando `data.frame()` o se pueden transformar otras estructuras definidas. Por ejemplo, una matriz, con el comando `as.data.frame()`

Forma 1

```
DF1 <- data.frame(ID = 1:100,
                  Numero = rnorm(100),
                  Sexo = rbinom(100, 1, 0.4))

head(DF1)
```

```
##   ID      Numero Sexo
## 1  1 -0.81703224    0
## 2  2 -0.64072176    0
## 3  3  0.76330663    0
## 4  4 -0.02254978    1
## 5  5  0.31928365    1
## 6  6  0.07727245    1
```

Creación de DF

Forma 2

```
ID <-  1:100
Numero <-  rnorm(100)
Sexo <-  rbinom(100, 1, 0.4)

Matriz <- cbind(ID, Numero, Sexo)
class(Matriz)
```

```
## [1] "matrix" "array"
```

```
### Convertimos la matriz en DF
DF2 <- as.data.frame(Matriz)
class(DF2)
```

```
## [1] "data.frame"
```

Manipulación de DF

Una vez cargada la base de datos, se pueden ralizar en ella, diversas operaciones:

- `head(data, k)`
- `tail(data, k)`
- `dim(data)`
- `length(data)`
- `names(data)`
- `attach(data)` (no recomendado)
- `Operador "$"`

Taller práctico

Utilice las operaciones anteriores en la base de datos `paises` de la librería `datos`.

- Se crea el DF

```
# install.packages("datos")

library(datos)
```

```
## Warning: package 'datos' was built under R version 4.1.1
```

```
datos <- datos::paises
```

- Se visualiza los primeros 3 valores del DF

```
head(datos, 3)
```

```
## # A tibble: 3 x 6
##   pais       continente  anio esperanza_de_vida poblacion pib_per_capita
##   <fct>      <fct>      <int>             <dbl>     <int>          <dbl>
## 1 Afganistán Asia        1952              28.8   8425333           779.
## 2 Afganistán Asia        1957              30.3   9240934           821.
## 3 Afganistán Asia        1962              32.0  10267083           853.
```

Taller práctico

- Se visualizan los últimos 3 valores del DF

```
tail(datos, 3)
```

```
## # A tibble: 3 x 6
##   pais     continente  anio esperanza_de_vida poblacion pib_per_capita
##   <fct>    <fct>      <int>             <dbl>     <int>          <dbl>
## 1 Zimbabue África      1997              46.8  11404948           792.
## 2 Zimbabue África      2002              40.0  11926563           672.
## 3 Zimbabue África      2007              43.5  12311143           470.
```

- ¿Cuántas observaciones y variables tiene el DF?

```
dim(datos)
```

```
## [1] 1704    6
```

Recuerde que la primera posición representa el número de observaciones y la segunda posición el número de variables.

Taller práctico

- ¿Cuántos elementos tiene el DF?

```
length(datos)
```

```
## [1] 6
```

- ¿Cuáles son los nombres de las variables del DF?

```
names(datos)
```

```
## [1] "pais"              "continente"        "anio"
## [4] "esperanza_de_vida" "poblacion"         "pib_per_capita"
```

- ¿Cómo rescatamos una variable en particular?

```
head(datos$anio, 15)
```

```
##  [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002 2007 1952 1957 1962
```

Tibble

Creación de un tibble

Recordando la clase pasada, los tibble se definen de la misma forma que un data frame. Volviendo al ejemplo anterior:

```
library(tibble)

tibble_DF1 <- tibble(ID = 1:100,
                     Numero = rnorm(100),
                     Sexo = rbinom(100, 1, 0.4))


tibble_DF1
```

```
## # A tibble: 100 x 3
##       ID Numero  Sexo
##    <int>  <dbl> <int>
##  1     1  2.04      1
##  2     2  0.314     1
##  3     3  0.382     0
##  4     4  0.774     0
##  5     5 -0.145     0
##  6     6 -0.241     1
##  7     7 -0.155     0
##  8     8 -1.09      1
##  9     9  1.51      0
## 10    10  1.91      0
## # ... with 90 more rows
```

Comparar DF y tibble

```
head(DF1, 3)
```

```
##   ID     Numero Sexo
## 1  1 -0.8170322    0
## 2  2 -0.6407218    0
## 3  3  0.7633066    0
```

```
head(DF1$Num, 3)
```

```
## [1] -0.8170322 -0.6407218  0.7633066
```

DF no es sensible a la completación de los nombres de las variables.

```
head(tibble_DF1, 3)
```

```
## # A tibble: 3 x 3
##      ID Numero  Sexo
##   <int>  <dbl> <int>
## 1     1  2.04      1
## 2     2  0.314     1
## 3     3  0.382     0
```

```
tibble_DF1$Num
```

```
## Warning: Unknown or uninitialised column: `Nu
```

```
## NULL
```

En cambio tibble si es sensible a la completación de los nombres de las variables (es menos riesgoso).

Importación y exportación de datos

Importación y exportación de datos

La importación de una base de datos dependerá básicamente del formato del archivo. Para ello, existen diferentes funciones que permiten llevar a cabo tal procedimiento:

Importar archivos `txt`

```
read.table("<Ruta del archivo>", header = TRUE, ... )
readr::read_csv("<Ruta del archivo>", col_names = TRUE,...)
```

Importar archivos `csv` Dependiendo de cómo esté codificado el archivo csv, hay distintas funciones para importar:

- **`read_csv`** : Si los decimales están con puntos y las variables se separan por comas.
- **`read_csv2`**: Si los decimales están con comas y las variables se separan por punto y coma (;).

```
readr::read_csv("<Ruta del archivo>", col_names = TRUE, ...)
readr::read_csv2("<Ruta del archivo>", col_names = TRUE, ...)
```

Importación de datos

Importar archivos `excel`

La librería `readxl` tiene múltiples funciones para cargar archivos en formato excel. Un argumento importante de estas funciones es **`sheet`**, en donde se puede indicar cuál hoja se importará. Sus principales funciones son:

```
readxl::read_excel("<Ruta del archivo>", col_names = TRUE, ...)
readxl::read_xls()
readxl::read_xlsx()
```

Importar múltiples tipos de archivos

El paquete `haven` contiene multiples funciones para importar archivos de SPSS, STATA y SAS con funciones, tales como: `read_sas()`, `read_por()`, `read_sav()` y `read_dta()`

El paquete `rio` y su función `import` permiten importar numerosos tipos de archivo de formato, incluyendo Excel, SAS, SPSS, STATA, Minitab, Matlab, JSON, etc. Es recomendable su uso si la base a cargar es limpia y no tiene problemas de importación.

```
rio::import("<Ruta del archivo>")
```

Análisis exploratorio

Análisis exploratorio

Las principales funciones para realizar los primeros pasos de un análisis de datos exploratorio, son las siguientes:

```
dplyr::glimpse(DF1)
```

```
## Rows: 100
## Columns: 3
## $ ID     <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, ~
## $ Numero <dbl> -0.81703224, -0.64072176, 0.76330663, -0.02254978, 0.31928365, ~
## $ Sexo   <int> 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, ~
```

```
str(DF1)
```

```
## 'data.frame':    100 obs. of  3 variables:
##  $ ID    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Numero: num  -0.817 -0.6407 0.7633 -0.0225 0.3193 ...
##  $ Sexo  : int  0 0 0 1 1 1 1 0 1 0 ...
```

```
summary(DF1)
```

```
##        ID             Numero             Sexo      
##  Min.   :  1.00   Min.   :-2.9586   Min.   :0.00  
##  1st Qu.: 25.75   1st Qu.:-0.8420   1st Qu.:0.00  
##  Median : 50.50   Median :-0.1147   Median :0.00  
##  Mean   : 50.50   Mean   :-0.1442   Mean   :0.44  
##  3rd Qu.: 75.25   3rd Qu.: 0.4358   3rd Qu.:1.00  
##  Max.   :100.00   Max.   : 4.3836   Max.   :1.00
```

```
skimr::skim(DF1)
```

Table: Data summary

Name	DF1
Number of rows	100
Number of columns	3
_	
Column type frequency:	
numeric	3
__	
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	h
ID	0	1	50.50	29.01	1.00	25.75	50.50	75.25	100.00	▇
Numero	0	1	-0.14	1.15	-2.96	-0.84	-0.11	0.44	4.38	▂
Sexo	0	1	0.44	0.50	0.00	0.00	0.00	1.00	1.00	▇

Problemas en la importación

Problemas en la importación de datos

Delimitadores

Algunas veces las observaciones de las bases de datos están separadas por distintos delimitadores. Esto puede configurarse para realizar correctamente la lectura de los datos en las distintas funciones de importación:

```
read.table(..., sep = "<delimitador>")
readr::read_csv(..., sep = "<delimitador>")
```

Saltos de fila

El argumento `skip` presente en la mayoría de las funciones para cargar datos, permite saltarse un número de filas de observaciones para realizar la lectura de datos. Esto sirve para cargar archivos en donde la base de datos no empieza desde la primera fila.

```
readr::read_csv(...,skip = n)
readxl::read_excel(...,skip = n)
```

Codificación de datos faltantes (NA)

Es común que en las bases de datos, especialmente en las encuestas, exista una codificación de valores faltantes distinto a una celda vacía (por ejemplo, *, 88 o 99). Dependiendo de cómo se importen los datos, estos puede recodificarse de distintas formas.

Directamente al importar

Algunas funciones de importación tienen argumentos que permiten codificar observaciones como NA dado un vector de referencia:

```
readr::read_csv("archivo", na = vector, ... )
readxl::read_excel("archivo", na = vector, ... )
```

Después de importar

Uno puede usar distintas herramientas para recodificar elementos como NA. A continuación se presentan dos formas equivalentes, usando R base y `dplyr` respectivamente.

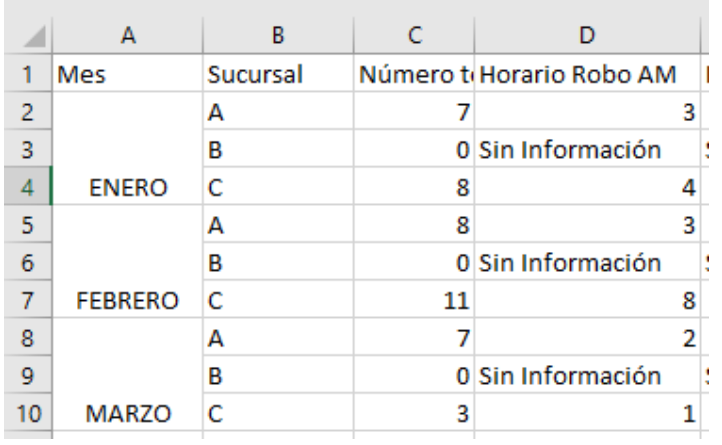
```
nombre_base[ nombre_base %in% vector ] <- NA

nombre_base %>% dplyr::na_if(vector)
```

Observaciones agrupadas

Algunas bases de datos contienen filas agrupadas. Esto es problemático dado que solo se cargará información a la primera celda contenida en esta agrupación. Es decir:

Mes	Sucursal	...
ENERO	A	...
NA	B	...
NA	C	...



	A	B	C	D
1	Mes	Sucursal	Número t	Horario Robo AM
2		A	7	3
3		B	0	Sin Información
4	ENERO	C	8	4
5		A	8	3
6		B	0	Sin Información
7	FEBRERO	C	11	8
8		A	7	2
9		B	0	Sin Información
10	MARZO	C	3	1

Si el documento es un Excel, esto se puede solucionar usando el argumento `fillMergedCells = TRUE` al cargar el archivo usando el comando `openxlsx::read.xlsx()`.

Si el archivo ya está cargado, el comando `tidyr::fill` permite llenar bases de datos con variables incompletas de muchas formas, gracias a su argumento `direction`.

Actividad

Actividad

La base de datos `viviendasRM.xlsx` contiene avisos de viviendas usadas que se venden en la Región Metropolitana de Chile recolectados de Chile Propiedades en mayo 2020. La base de datos contiene 1139 observaciones de 13 variables (una descripción de esta base se encuentra en la siguiente slide). **Fuente:** Kaggle.

Descripción de la base de datos:

Variable	Descripción
Comuna	Comuna de la región en la cual se encuentra la casa.
Link	Enlace al aviso de la vivienda.
Tipo_Vivienda	Tipo de vivienda.
N_Habitaciones	Número de habitaciones.
N_Banos	Número de baños.
N_Estacionamientos	Número de estacionamientos.
Total_Superficie_M2	Total de superficie en mt2.
Superficie_Construida_M2	Superficie construida en mt2: metros de la construcción habitable.
Valor_UF	Valor en Unidades de fomentos de la propiedad declarado en el portal.
Valor_CLP	Valor en pesos chilenos.
Direccion	Dirección de la casa del aviso.
Quien_Vende	Nombre de la persona que vende la vivienda.
Corredor	Indica nombre de la empresa que vende.

Actividad

Utilizando los contenidos abarcados en la clase 2 y 3, realice las siguientes actividades:

a. Importar base de datos y corrobore su clase (¿es tibble o data frame?)

b. Si su base fue importada como DF, convierta a tibble

c. Realice un pequeño análisis exploratorio de la base

d. Rescate la información contenida en el cuarto registro

e. Compare los resultados obtenidos con glimpse y str

¡Gracias!

Diego Muñoz Ureta
dimunoz1@uc.cl

Felipe Moya
felipe.moya@uc.cl