

# Taller 2: Exploración y validación de datos

José Daniel Conejeros

Magíster(c) Sociología UC - Licenciado en Ciencias Sociales

Junio 2020

 Código del taller  [jdconejeros@uc.cl](mailto:jdconejeros@uc.cl)  JDConejeros

 [joseconejerosp](#)  [Jose\\_Conejeros](#)

# ¿Qué hemos visto hasta el momento?



- Instalación de R y RStudio
- Explorar espacios de trabajo
- Comprender la lógica de objetos
- Explorar vectores, listas y matrices
- Estructura de funciones y argumentos
- Instalar y cargar librerías/paquetes

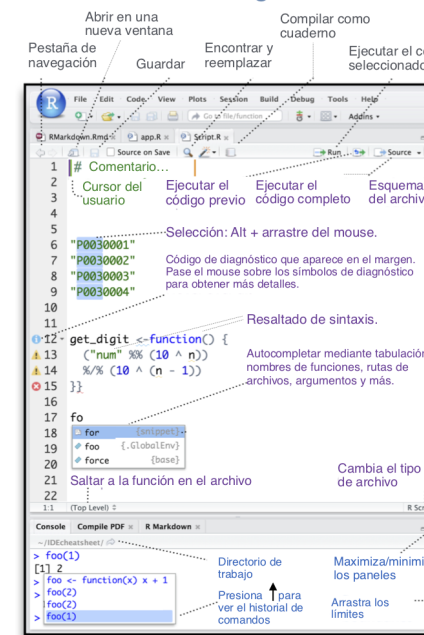
## Documentos y apps

RStudio desarrolla herramientas gratuitas y abiertas para R. Su entorno de desarrollo integrado (IDE) facilita el análisis de datos con R.

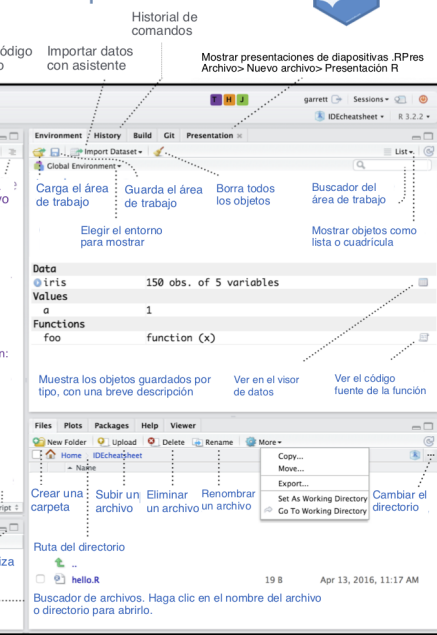
También ofrece: muchos paquetes R (e.g. tidyverse, sparklyr, ggplot2, dplyr), incluidos **Shiny** (crea aplicaciones web sencillas en R) y **R Markdown** (te permite convertir tus análisis en documentos, informes, presentaciones y paneles de alta calidad; compartílos y reproducílos).



## Escribe tu código

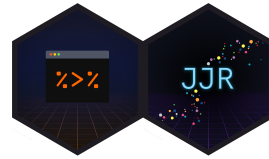


## Soporte R

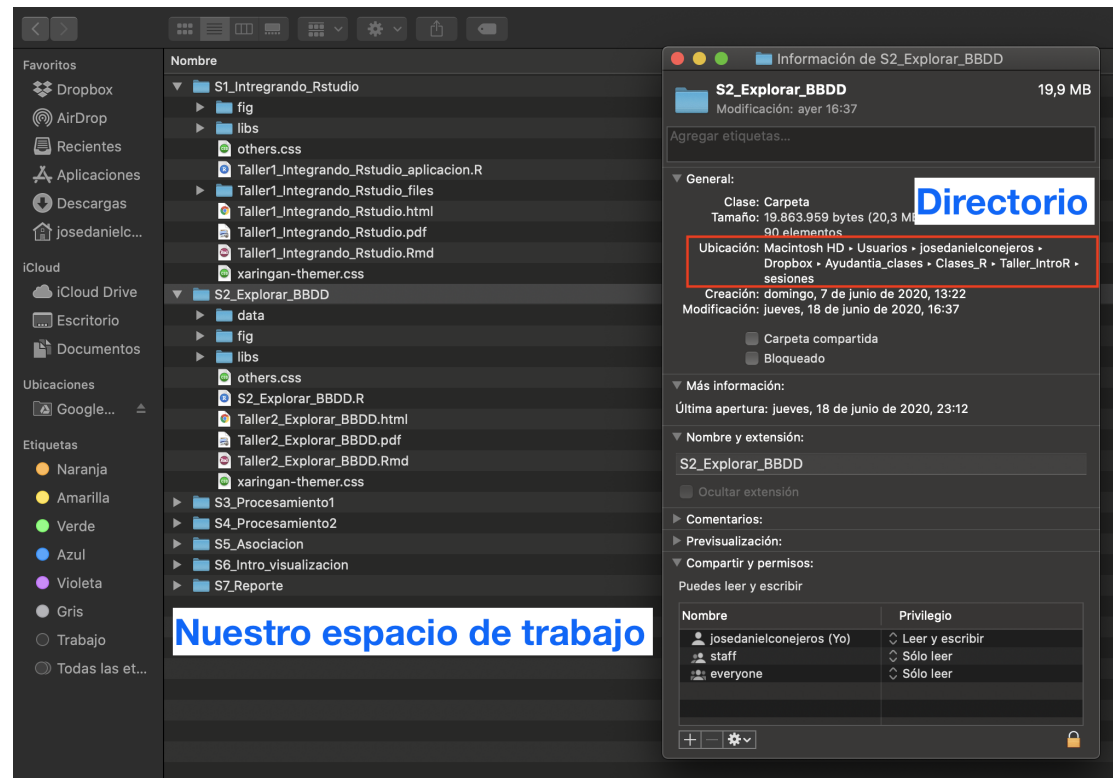


¿Alguna pregunta?

# Hoy: Definir nuestro espacio de trabajo



```
getwd() # Directorio de trabajo actual  
setwd("ruta") # Establecer directorio de trabajo
```



También podemos generar proyectos de trabajo

# Estructura de una base de datos



Las filas representan las **observaciones** - Las columnas representan las **variables**

```
id ← c(1, 2, 2, 4, 4)
edad ← c(23, 45, 67, 201)
sexo ← c(1, 0, 1, 0, NA)
peso ← c(80, 60, 70, 50, 55)
altura ← c(180, 160, 200, 140, 300)

data ← as.data.frame(cbind(id, edad, sexo,
                             peso, altura))

data ← as.data.frame(data)
data
```

```
##   id edad sexo peso altura
## 1   1  23    1  80    180
## 2   2  45    0  60    160
## 3   2  67    1  70    200
## 4   4 201    0  50    140
## 5   4  23   NA  55    300
```

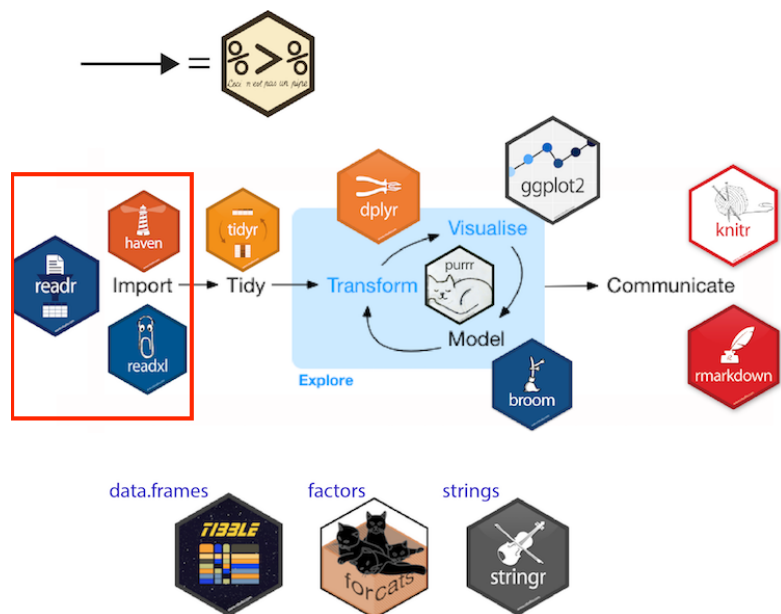
Dim	Homogeneous	Heterogeneous
1d	Atomic Vector	List
2d	Matrix	Data - Frame
nd	Array	

Mayor detalle pueden revisar el siguiente [enlace](#)

# Importar una base de datos



```
install.packages(c("haven", "readxl", "readr"))  
library(haven)  
library(readxl)  
library(readr)
```



Mayor detalle del paquete [readr](#) aquí

Mayor detalle del paquete [haven](#) aquí

# Explorar nuestra base de datos



```
dim(data)  # Observaciones y variables
```

```
## [1] 5 5
```

```
names(data) # Nombre de nuestras variables
```

```
## [1] "id"      "edad"    "sexo"    "peso"    "altura"
```

```
str(data)  # Visor de nuestras variables
```

```
## 'data.frame':    5 obs. of  5 variables:
## $ id      : num  1 2 2 4 4
## $ edad    : num  23 45 67 201 23
## $ sexo    : num  1 0 1 0 NA
## $ peso    : num  80 60 70 50 55
## $ altura  : num  180 160 200 140 300
```

# Explorar nuestra base de datos



```
head(data) # Primeras 6 observaciones
```

```
##   id edad sexo peso altura
## 1  1   23    1   80   180
## 2  2   45    0   60   160
## 3  2   67    1   70   200
## 4  4  201    0   50   140
## 5  4   23   NA   55   300
```

```
tail(data) # Últimas 6 observaciones
```

```
##   id edad sexo peso altura
## 1  1   23    1   80   180
## 2  2   45    0   60   160
## 3  2   67    1   70   200
## 4  4  201    0   50   140
## 5  4   23   NA   55   300
```

# Explorar nuestra base de datos



```
summary(data)

library(skimr)
skim(data)
```

skim_type	skim_variable	n_missing	numeric.mean	numeric.sd	numeric.p0	numeric.p25	numeric.p50	numeric.p75	numeric.p100	numeric.hist
numeric	id	0	2.6	1.3416408	1	2	2.0	4	4	
numeric	edad	0	71.8	74.4929527	23	23	45.0	67	201	
numeric	sexo	1	0.5	0.5773503	0	0	0.5	1	1	
numeric	peso	0	63.0	12.0415946	50	55	60.0	70	80	
numeric	altura	0	196.0	62.2896460	140	160	180.0	200	300	

¿Qué podemos ver de nuestras variables?



# Validación de una base de datos



## Identificar observaciones duplicadas

```
# Nuestros casos únicos  
unique(data)
```

```
##      id edad sexo peso altura  
## 1  1   23    1   80   180  
## 2  2   45    0   60   160  
## 3  2   67    1   70   200  
## 4  4  201    0   50   140  
## 5  4   23   NA   55   300
```

```
# Nuestros casos únicos  
unique(data$id)
```

```
## [1] 1 2 4
```

```
# Nuestros casos duplicados  
duplicated(data$id)
```

```
## [1] FALSE FALSE  TRUE FALSE  TRUE
```

# Validación de una base de datos



## Identificar observaciones duplicadas

```
# Nuestros casos duplicados
data$caso ← duplicated(data$id)
# Resumen de duplicados
table(data$caso)
```

```
##
## FALSE  TRUE
##      3     2
```

```
# Nuestros casos duplicados
data[duplicated(data$id),]
```

```
##   id edad sexo peso altura caso
## 3  2   67    1   70    200  TRUE
## 5  4   23   NA   55    300  TRUE
```

# Validación de una base de datos



## Identificar observaciones duplicadas

```
# Quitar los casos duplicados
data2 <- data[unique(data$id),]
data2
```

```
##   id edad sexo peso altura caso
## 1  1   23    1   80    180 FALSE
## 2  2   45    0   60    160 FALSE
## 4  4  201    0   50    140 FALSE
```

## Revisar valores fuera de rango

```
# Mínimo
min(data$edad)
```

```
## [1] 23
```

```
# Máximo
max(data$edad)
```

```
## [1] 201
```

# Validación de una base de datos



## Revisar valores fuera de rango

```
# Tabla de valores
table(data$edad)
```

```
##
##  23  45  67 201
##    2   1   1   1
```

```
table(data$sexo)
```

```
##
##  0  1
##  2  2
```

```
# Descriptivos
summary(data$edad)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      23.0   23.0   45.0   71.8   67.0   201.0
```

# Validación de una base de datos



## Explorar codificación de los casos perdidos

```
# Revisar casos perdidos
table(data$sexo, useNA = "ifany")
```

```
##
##      0      1 <NA>
##      2      2      1
```

```
# Mostrar los casos perdidos
is.na(data)
```

```
##           id  edad  sexo  peso  altura  caso
## [1,] FALSE FALSE FALSE FALSE  FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE  FALSE FALSE
## [3,] FALSE FALSE FALSE FALSE  FALSE FALSE
## [4,] FALSE FALSE FALSE FALSE  FALSE FALSE
## [5,] FALSE FALSE  TRUE  FALSE  FALSE FALSE
```

# Validación de una base de datos



## Explorar codificación de los casos perdidos

```
which(is.na(data$sexo)) # Indica la fila
```

```
## [1] 5
```

```
# Eliminar casos perdidos  
data3 ← na.omit(data)  
data3
```

```
##   id edad sexo peso altura caso  
## 1  1  23    1  80    180 FALSE  
## 2  2  45    0  60    160 FALSE  
## 3  2  67    1  70    200  TRUE  
## 4  4 201    0  50    140 FALSE
```

# Validación de una base de datos



## Explorar codificación de los casos perdidos

```
# Recodificar casos perdidos (imputar)
data4 <- data
data4$sexo[is.na(data$sexo)] <- 1
data
```

##	id	edad	sexo	peso	altura	caso
## 1	1	23	1	80	180	FALSE
## 2	2	45	0	60	160	FALSE
## 3	2	67	1	70	200	TRUE
## 4	4	201	0	50	140	FALSE
## 5	4	23	NA	55	300	TRUE

data4

##	id	edad	sexo	peso	altura	caso
## 1	1	23	1	80	180	FALSE
## 2	2	45	0	60	160	FALSE
## 3	2	67	1	70	200	TRUE
## 4	4	201	0	50	140	FALSE
## 5	4	23	1	55	300	TRUE

- Chequear efectividad de los filtros (si es que los hay): próximo taller
- Chequear que las variables estén en un formato correcto para el análisis: próximo taller

¿Qué otros temas podríamos validar?

# Exportar una base de datos



Podemos guardar nuestra base como texto plano:

```
write.table(data, file="data.txt", sep="\t")
```

Podemos guardar en un archivo separado por comas (csv):

```
# Separado por comas  
write.csv(data, file="data.csv", row.names = FALSE)  
  
# Otras opciones de separación  
write.csv(data, file="data.csv", row.names = FALSE, sep = "-")
```

¿Qué ventaja tienen estos formatos?

Podemos guardar en un excel tradicional:

```
library(openxlsx)  
write.xlsx(data, file="data.xlsx")
```



# Exportar una base de datos



Podemos guardar en el formato de [SPSS](#):

```
library(haven)  
write_sav(data, "data.sav")
```

Podemos guardar en el formato de [SAS](#):

```
write_sas(data, "data.sas7bdat")
```

Podemos guardar en el formato de [Stata](#):

```
write_dta(data, "data.dta")
```

Podemos guardar en el formato de [R](#):

```
save(data, file="data.Rdata")
```

# Próximo taller



- Encadenar operaciones: `%>%` (pipe)
- Filtrar y ordenar datos: `filter()` & `arrange()`
- Seleccionar columnas: `select()`
- Generar nuevas variables: `mutate` & `$`
- Recodificar y etiquetar: `indexar` `code` & `dplyr`

# Próximo taller



## Data Transformation with dplyr : : CHEAT SHEET

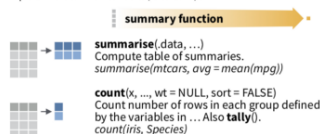


dplyr functions work with pipes and expect tidy data. In tidy data:



### Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

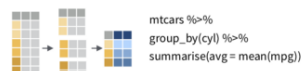


#### VARIATIONS

**summarise\_all()** - Apply funs to every column.  
**summarise\_at()** - Apply funs to specific columns.  
**summarise\_if()** - Apply funs to all cols of one type.

### Group Cases

Use **group\_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



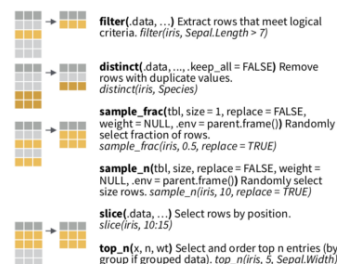
**group\_by(data, ..., add = FALSE)**  
Returns copy of table grouped by ...  
`g_iris <- group_by(iris, Species)`

**ungroup(x, ...)**  
Returns ungrouped copy of table.  
`ungroup(g_iris)`

### Manipulate Cases

#### EXTRACT CASES

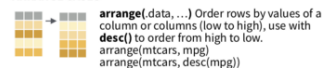
Row functions return a subset of rows as a new table.



#### Logical and boolean operators to use with filter()

`<` `<=` `is.na()` `%in%` `|` `xor()`  
`>` `>=` `!is.na()` `!` `&`  
See ?base:logic and ?Comparison for help.

#### ARRANGE CASES



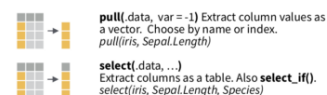
#### ADD CASES

**add\_row(data, ..., before = NULL, after = NULL)**  
Add one or more rows to a table.  
`add_row(faithful, eruptions = 1, waiting = 1)`

### Manipulate Variables

#### EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

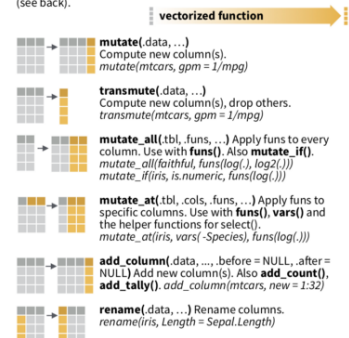


Use these helpers with **select()**, e.g. `select(iris, starts_with("Sepal"))`

**contains(match)** **num\_range(prefix, range)** **ends\_with(match)** **one\_of(...)** **matches(match)** **starts\_with(match)**  
e.g. `mpg:cyl`  
e.g. `Species`

#### MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).



RStudio is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more with browseVignettes(package = "dplyr", "tidbldr") • dplyr 0.7.0 • tidbldr 1.2.0 • Updated: 2017-03

Pueden adelantar revisando el siguiente [enlace](#)

# Donación



Se donará el 25% de lo recaudado en el taller: **\$350.000**.

Se está gestionando **cajas de alimentos** para personas con problemas económicos en La comuna de la Pintana, sector el castillo.

¡Se les avisará cuando se entreguen estas cajas! (en las **próximas 2 semanas**)

Sugerencias: ✉ **Buzón anónimo**

# ¡Gracias!

Presentación generada en:



✉ [jdconejeros@uc.cl](mailto:jdconejeros@uc.cl)  [joseconejerosp](#)

 [JDConejeros](#)  [Jose\\_Conejeros](#)