

UNIVERSIDAD DE GUANAJUATO



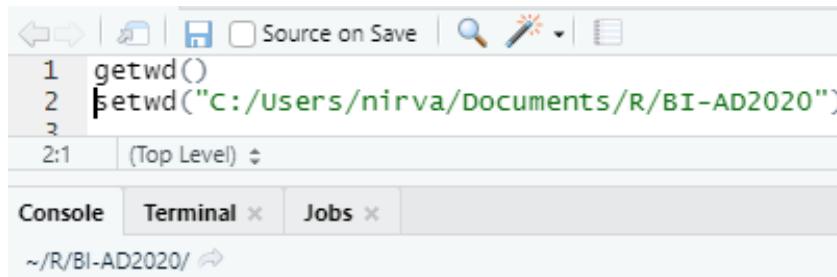
Inteligencia de Negocios

Agosto-Diciembre 2020

Martes - Jueves

10:30 - 13:00

- R maneja directorios de trabajo. Básicamente, es donde se almacenan los archivos relacionados al proyecto
 - **getwd()** – obtiene el directorio actual de trabajo
 - **setwd()** – establece una ruta para el directorio de trabajo



The screenshot shows the RStudio interface. In the top-left pane, there is a code editor with the following R code:

```
1 getwd()
2 setwd("C:/users/nirva/Documents/R/BI-AD2020")
3
```

The second line contains a partially typed command, "setwd("C:/users/nirva/Documents/R/BI-AD2020)". Below the code editor is a status bar showing "2:1 (Top Level) ⌂".

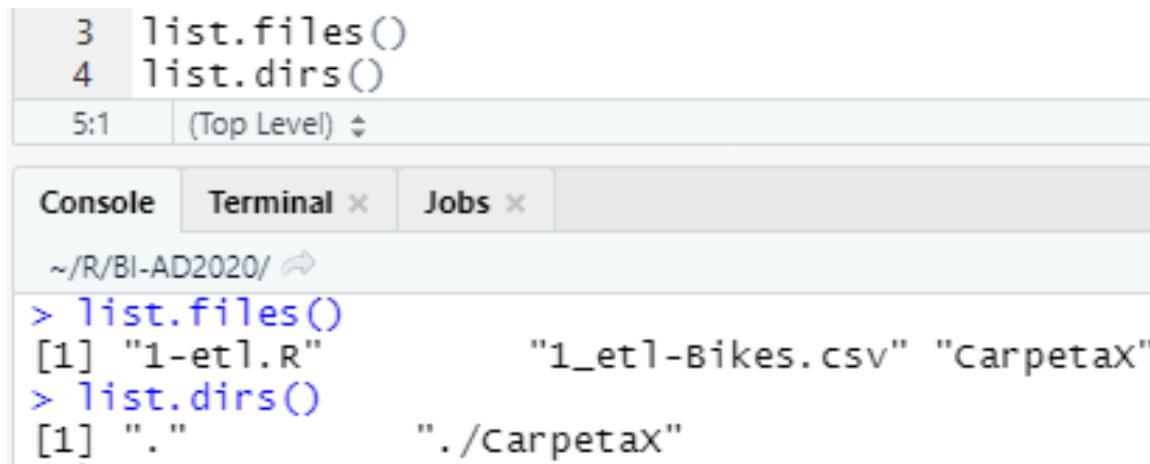
At the bottom of the interface, there is a tab bar with "Console", "Terminal x", and "Jobs x". The "Console" tab is selected.

In the bottom-right pane, the R console window displays the following output:

```
~/R/BI-AD2020/ ⌂
> getwd()
[1] "C:/Users/nirva/Documents"
> setwd("C:/users/nirva/Documents/R/BI-AD2020")
> getwd()
[1] "C:/Users/nirva/Documents/R/BI-AD2020"
```

[Más info](#)

- **list.files()** – muestra todos los archivos en el directorio de trabajo.
- **list.dirs()** – muestra todas las carpetas en el directorio de trabajo.



3 list.files()
4 list.dirs()
5:1 (Top Level) ↴

Console Terminal × Jobs ×

~/R/BI-AD2020/ ↴

```
> list.files()
[1] "1-etl.R"          "1_etl-Bikes.csv" "Carpetax"
> list.dirs()
[1] "."               "./Carpetax"
```



EXTRACCIÓN, TRANSFORMACIÓN Y CARGA

- ETL es un proceso enfocado a obtener, manipular y almacenar datos para cumplir con las necesidades de negocio o análisis. Puede considerarse como el punto de inicio de un proyecto de Inteligencia de Negocios.



- Es una compañía, iniciada en 2011, que opera una flota de bicicletas de alquiler público en un área metropolitana. Sus clientes son oficinistas de gobierno y empresas, y estudiantes del área urbana. Los clientes pueden fácilmente rentar bicicletas dentro de una red de estaciones en toda la ciudad; siendo así, que pueden rentar una bicicleta en una estación y dejarla en otra.

- Bicicleta compartida ha tenido un crecimiento continuo. Debido a esto, se estableció un grupo de Inteligencia de Negocios para realizar seguimiento de los datos almacenados acerca de sus transacciones, clientes y factores relacionados con las rentas. En 2014, comenzaron a entender como podrían usar los datos almacenados para guiar decisiones relacionadas a ventas, operaciones y publicidad.
- Como parte del grupo de Inteligencia de Negocios, se le ha proporcionado los datos de las rentas del periodo del 01/01/2011 – 31/12/2012.

- **datetime:** Hourly date + timestamp
- **season:** 1 = spring, 2 = summer, 3 = fall, 4 = winter
- **holiday:** Whether the day is considered a holiday
- **workingday:** Whether the day is neither a weekend nor holiday
- **weather:**
 - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
 - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
 - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
 - 4: Heavy Rain + Ice Pellets + Thunderstorm + Mist, Snow + Fog
- **temp:** Temperature in Celsius
- **atemp:** *Feels like* temperature in Celsius
- **humidity:** Relative humidity
- **windspeed:** Wind speed
- **casual:** Number of non-registered user rentals initiated
- **registered:** Number of registered user rentals initiated
- **count:** Number of total rentals

Bicicleta compartida: Vista del archivo



UNIVERSIDAD DE
GUANAJUATO

	A	B	C	D	E	F	G	H	I	J	K	L
1	datetime	season	holiday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	count
2	01/01/2011 00:00	1	0	0	1	9.84	14.395	81	0	3	13	16
3	01/01/2011 01:00	1	0	0	1	9.02	13.635	80	0	8	32	40
4	01/01/2011 02:00	1	0	0	1	9.02	13.635	80	0	5	27	32
5	01/01/2011 03:00	1	0	0	1	9.84	14.395	75	0	3	10	13
6	01/01/2011 04:00	1	0	0	1	9.84	14.395	75	0	0	1	1
7	01/01/2011 05:00	1	0	0	2	9.84	12.88	75	6.0032	0	1	1
8	01/01/2011 06:00	1	0	0	1	9.02	13.635	80	0	2	0	2
9	01/01/2011 07:00	1	0	0	1	8.2	12.88	86	0	1	2	3
10	01/01/2011 08:00	1	0	0	1	9.84	14.395	75	0	1	7	8
11	01/01/2011 09:00	1	0	0	1	13.12	17.425	76	0	8	6	14
12	01/01/2011 10:00	1	0	0	1	15.58	19.695	76	16.9979	12	24	36
13	01/01/2011 11:00	1	0	0	1	14.76	16.665	81	19.0012	26	30	56
14	01/01/2011 12:00	1	0	0	1	17.22	21.21	77	19.0012	29	55	84
15	01/01/2011 13:00	1	0	0	2	18.86	22.725	72	19.9995	47	47	94
16	01/01/2011 14:00	1	0	0	2	18.86	22.725	72	19.0012	35	71	106
17	01/01/2011 15:00	1	0	0	2	18.04	21.97	77	19.9995	40	70	110
18	01/01/2011 16:00	1	0	0	2	17.22	21.21	82	19.9995	41	52	93
19	01/01/2011 17:00	1	0	0	2	18.04	21.97	82	19.0012	15	52	67
20	01/01/2011 18:00	1	0	0	3	17.22	21.21	88	16.9979	9	26	35



- **datos <- read.csv("{nombre_archivo}.csv")**
- **datos <-read.table("{nombre_archivo}.txt", sep=",", header=TRUE)**

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Shows R script code for setting the working directory, listing files and directories, and reading CSV files into two datasets: `rentas1` and `rentas2`.
- Console:** Displays the same R commands being run.
- Environment:** Shows the global environment with two data frames: `rentas1` and `rentas2`, both containing 17379 observations and 12 variables.
- File Explorer:** Shows the project structure with files `1-etl.R`, `1_etl-Bikes.csv`, and a folder `CarpetaX`.

**Investigar: ¿Cómo importar datos de bds
relacionales en R?**

Extraer datos de los recursos: Visualizando datos importados



UNIVERSIDAD DE
GUANAJUATO

• **str(objeto)**

The screenshot shows the RStudio interface. In the top-left, there's a code editor window titled "1-etri.R" containing R code. In the top-right, the "Environment" tab of the global environment is selected, showing a dataset named "rentas". Below the code editor is the "Console" tab, which displays the output of the `str(rentas)` command.

```
1 getwd() #ruta actual del directorio de trabajo
2 setwd("C:/Users/nirva/Documents/R/BI-AD2020") #cambio de ruta del directorio de trabajo
3 list.files() #listar los archivos del directorio de trabajo
4 list.dirs() #listar las carpetas del directorio de trabajo
5 rentas <- read.csv("1_etl-Bikes.csv") #importar datos de csv
6 str(rentas)#muestra información compacta de un objeto
7
```

7:1 (Top Level) R Script

The screenshot shows the RStudio interface with the "Console" tab active. It displays the output of the `str(rentas)` command, which provides a detailed summary of the "rentas" dataset, including the number of observations, variables, and their types and values.

```
> str(rentas)#muestra información compacta de un objeto
'data.frame': 17379 obs. of 12 variables:
 $ datetime : chr "1/1/2011 0:00" "1/1/2011 1:00" "1/1/2011 2:00" "1/1/2011 3:00" ...
 $ season   : int 1 1 1 1 1 1 1 1 1 ...
 $ holiday  : int 0 0 0 0 0 0 0 0 0 ...
 $ workingday: int 0 0 0 0 0 0 0 0 0 ...
 $ weather  : int 1 1 1 1 2 1 1 1 1 ...
 $ temp     : num 9.84 9.02 9.02 9.84 9.84 ...
 $ atemp    : num 14.4 13.6 13.6 14.4 14.4 ...
 $ humidity : int 81 80 80 75 75 75 80 86 75 76 ...
 $ windspeed: num 0 0 0 0 0 ...
 $ casual   : int 3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int 13 32 27 10 1 1 0 2 7 6 ...
 $ count    : int 16 40 32 13 1 1 2 3 8 14 ...
```

>

The screenshot shows the RStudio interface with the "Environment" tab active. The "rentas" dataset is listed under the "Data" section, showing it has 17379 observations and 12 variables.

The screenshot shows the RStudio interface with the "Files" tab active. It displays the contents of the "BI-AD2020" folder, which includes files like "1-etri.R" and "1_etl-Bikes.csv", and a folder named "CarpetaX".

- Instalar paquete *dplyr* para la manipulación de datos
 - `install.package("nombre_paquete")`

The screenshot shows the RStudio interface with the following details:

- Script Editor:** The file "1-eti.R" is open, displaying R code to set the working directory, read a CSV file, and install packages.
- Console Output:** The console shows the execution of the script, including the download of several R package files from CRAN (e.g., dplyr_1.0.2.zip, assertthat_0.3.0.zip, utf8_1.2.2.zip) and their successful unpacking and MD5 sum verification.

```
1 eti.R* Source on Save Run Source
1 getwd() #ruta actual del directorio de trabajo
2 setwd("C:/users/nirva/Documents/R/BI-AD2020") #cambio de ruta del directorio de trabajo
3 list.files() #listar los archivos del directorio de trabajo
4 list.dirs() #listar las carpetas del directorio de trabajo
5 rentals <- read.csv("1_eti-Bikes.csv") #importar datos de csv
6 str(rentals)#muestra información compacta de un objeto
7 install.packages("dplyr") #instalación de paquete
7:50 (Top Level) R Script

Console Terminal × Jobs ×
~/R/BI-AD2020/ ↵
probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/R6_2.4.1.zip'
Content type 'application/zip' length 59273 bytes (57 KB)
downloaded 57 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/rlang_0.4.7.zip'
Content type 'application/zip' length 1136794 bytes (1.1 MB)
downloaded 1.1 MB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/tibble_3.0.3.zip'
Content type 'application/zip' length 416148 bytes (406 KB)
downloaded 406 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/tidyselect_1.1.0.zip'
Content type 'application/zip' length 203209 bytes (198 KB)
downloaded 198 KB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/vctrs_0.3.4.zip'
Content type 'application/zip' length 1193987 bytes (1.1 MB)
downloaded 1.1 MB

probando la URL 'https://cran.rstudio.com/bin/windows/contrib/4.0/dplyr_1.0.2.zip'
Content type 'application/zip' length 1308048 bytes (1.2 MB)
downloaded 1.2 MB

package 'assertthat' successfully unpacked and MD5 sums checked
package 'utf8' successfully unpacked and MD5 sums checked
```



- Cargar paquete
 - **library("nombre_paquete")**

A screenshot of the RStudio interface. The top panel shows an R script with two lines of code: 'library(dplyr) #cargar paquete para su uso'. The bottom panel shows the 'Console' tab with the command 'library(dplyr) #cargar paquete para su uso' entered and its output: 'Attaching package: 'dplyr''. It also lists masked objects from 'package:stats' ('filter, lag') and 'package:base' ('intersect, setdiff, setequal, union').

```
8 library(dplyr) #cargar paquete para su uso
9 |
9:1 (Top Level) R Script
Console Terminal × Jobs ×
~/R/BI-AD2020/ ↗
> library(dplyr) #cargar paquete para su uso
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':
  filter, lag

The following objects are masked from 'package:base':
  intersect, setdiff, setequal, union
```

- Para extraer (filtrar) un subconjunto de filas que cumplen con un criterio con base a condiciones lógicas, se usa:
 - filtrado <- **filter(*objeto* , *condiciones*)**
- Operadores lógicos

Logical Operators in the R Environment			
<	Less than	>	Greater than
<=	Less than or equal to	>=	Greater than or equal to
==	Equal to	!=	Not equal to
%in%	Group membership	&, , !, xor, any, and all	Boolean operators
is.na	Is NA	!is.na	Is not NA



- Listar los registros donde no hay ninguna renta iniciada por usuarios registrados durante la temporada de primavera (1) o verano (2)

The screenshot shows an RStudio interface. On the left, the script editor window contains the following R code:

```
6 str(rentas) #muestra información compacta de un objeto
7 install.packages("dplyr") #instalación de paquete
8 library(dplyr) #cargar paquete para su uso
9 norss <- filter(rentas, registered == 0 & (season == 1 | season == 2)) #filtra filas de acuerdo a condición
10 dim(norss)
11
12
```

The console window below shows the execution of the code:

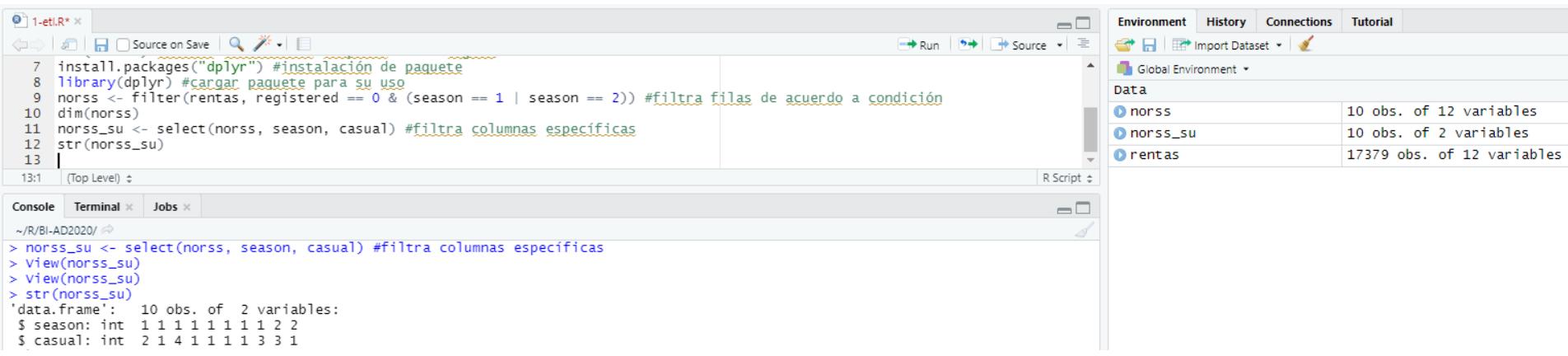
```
> norss <- filter(rentas, registered == 0 & (season == 1 | season == 2)) #filtra filas de acuerdo a condición
> dim(norss)
[1] 10 12
```

On the right, the environment pane displays two data objects:

Global Environment	Data
	<ul style="list-style-type: none">norss: 10 obs. of 12 variablesrentas: 17379 obs. of 12 variables

Nota: **dim(objeto)** regresa la dimensión
de un objeto

- Para extraer (seleccionar) las columnas que se desean conservar, se usa
 - seleccionando `<- select(objeto, atr_1, ... atr_n)`
- De los registros filtrados, solo mostrar la temporada y la cantidad de clientes casuales que iniciaron una renta



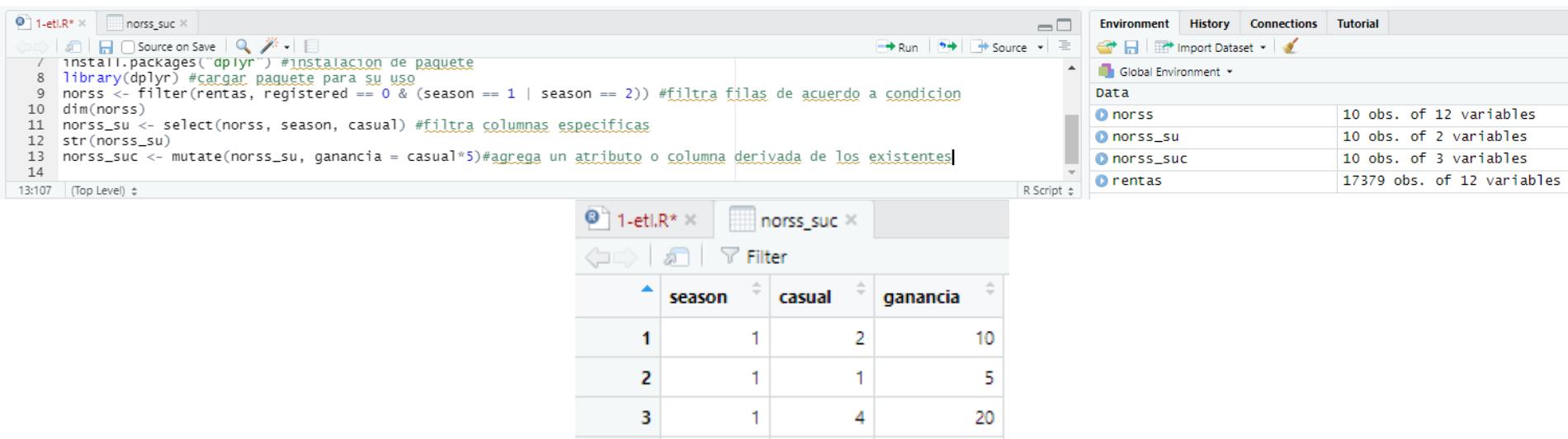
The screenshot shows an RStudio interface with the following details:

- Code Editor:** Shows R script code for installing dplyr, loading it, filtering rental data, and selecting specific columns (season and casual).

```
7 install.packages("dplyr") #instalación de paquete
8 library(dplyr) #cargar paquete para su uso
9 norss <- filter(rentas, registered == 0 & (season == 1 | season == 2)) #filtra filas de acuerdo a condición
10 dim(norss)
11 norss_su <- select(norss, season, casual) #filtra columnas específicas
12 str(norss_su)
13
```
- Console:** Shows the execution of the script and the resulting output for the selected subset.

```
> norss_su <- select(norss, season, casual) #filtra columnas específicas
> View(norss_su)
> View(norss_su)
> str(norss_su)
'data.frame': 10 obs. of 2 variables:
 $ season: int 1 1 1 1 1 1 1 2 2
 $ casual: int 2 1 4 1 1 1 1 3 3 1
```
- Environment:** Shows the global environment with objects: norss (10 obs. of 12 variables), norss_su (10 obs. of 2 variables), and rentas (17379 obs. of 12 variables).

- Para generar una columna con valores derivados de los datos existentes, se usa:
 - derivado <- **mutate**(*objeto* , *nueva_columna=calculo*)
- De la consulta previa, calcule los costos de renta obtenidos de los usuarios casuales. El costo por el pase de un día es de \$5.00



The screenshot shows the RStudio interface with the following details:

- Code Editor:** The script pane contains R code for data manipulation. Lines 1-14 show the installation of the dplyr package, filtering of rental data for seasons 1 and 2, selecting specific columns for casual users, and creating a new column 'ganancia' by multiplying 'casual' by 5.
- Environment View:** Shows the global environment with four objects: norss (10 obs. of 12 variables), norss_su (10 obs. of 2 variables), norss_suc (10 obs. of 3 variables), and rentas (17379 obs. of 12 variables).
- Data View:** A data grid displays the first few rows of the norss_suc dataset, which includes columns for season, casual count, and calculated ganancia.

	season	casual	ganancia
1	1	2	10
2	1	1	5
3	1	4	20



- Para generar resúmenes de datos, son necesarias las siguientes dos funciones:
 - grupo <- **group_by**(*objeto, atributo1, … , atributoN*)
 - resumen <- **summarise**(*obj_agrupado, función_agregación(atributoX), … , función_agregación(atributoZ)*)

- Reportar la cantidad de clientes casuales y el total de ganancias obtenidas de estos.

The screenshot shows the RStudio interface with the following components:

- Code Editor:** The file "1-etl.R" contains R code for data transformation. The code reads a CSV file, imports it, installs the "dplyr" package, filters rows where registered == 0 and season is 1 or 2, selects specific columns, adds a derived column "ganancia" (casual * 5), groups by "season", and summarizes the data to get the sum of casual clients and total ganancia.
- Environment View:** Shows the global environment with objects like gpo_norss, norss, norss_su, norss_suc, rentas, and sum_norss, along with their dimensions.
- Data View:** A data grid titled "sum_norss" showing the summarized data. The table has columns: season, sum(casual), and sum(ganancia). It shows two rows: one for season 1 with values 14 and 70, and one for season 2 with values 4 and 20.

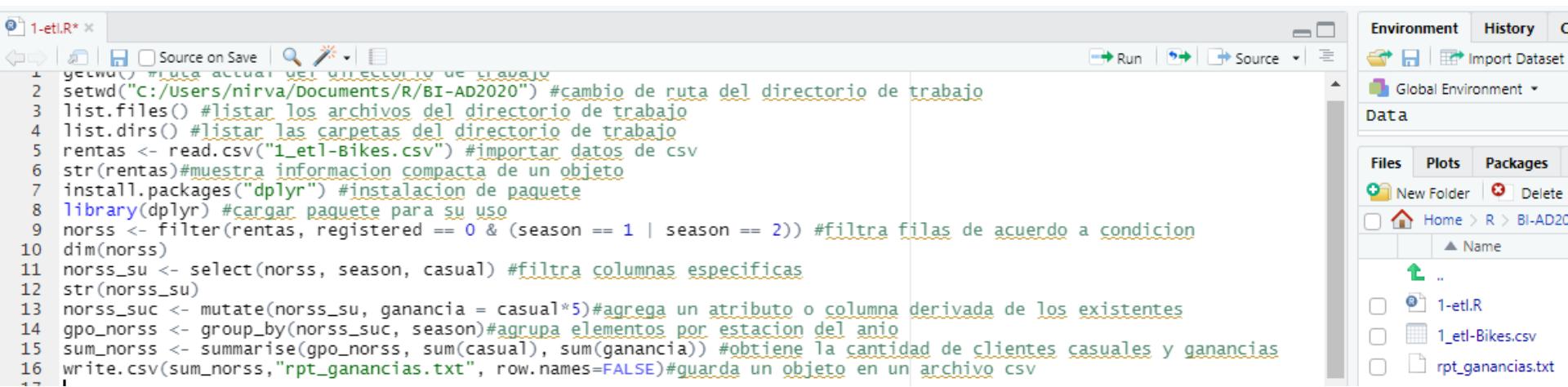
```

1 etl.R* x
2
3 rentas <- read.csv("1-EET-BIKES.CSV") #importar valores de csv
4 str(rentas) #muestra informacion compacta de un objeto
5
6 install.packages("dplyr") #instalacion de paquete
7 library(dplyr) #cargar paquete para su uso
8
9 norss <- filter(rentas, registered == 0 & (season == 1 | season == 2)) #filtra filas de acuerdo a condicion
10 dim(norss)
11 norss_su <- select(norss, season, casual) #filtra columnas especificas
12 str(norss_su)
13 norss_suc <- mutate(norss_su, ganancia = casual*5) #agrega un atributo o columna derivada de los existentes
14 gpo_norss <- group_by(norss_suc, season) #agrupa elementos por estacion del año
15 sum_norss <- summarise(gpo_norss, sum(casual), sum(ganancia)) #obtiene la cantidad de clientes casuales y ganancias
16

```

	season	sum(casual)	sum(ganancia)
1	1	14	70
2	2	4	20

- Los datos transformados pueden ser guardados mediante las siguientes funciones:
 - **write.csv(objeto, “{nombre_archivo}.csv”, row.names=FALSE)**
 - **write.table(objeto, “{nombre_archivo}.txt”, row.names=FALSE, sep=“\t”)**

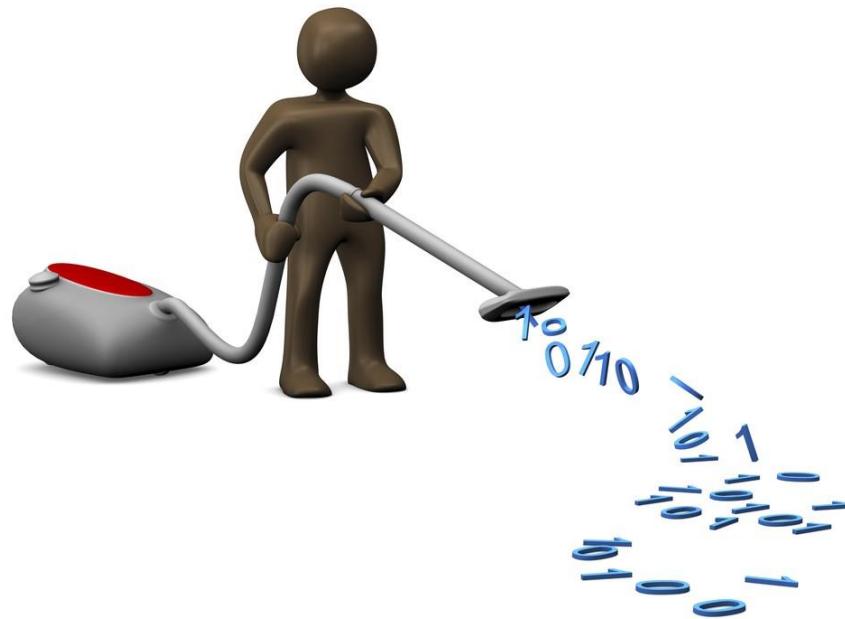


```
1 getwd() #se establece la ruta actual de trabajo
2 setwd("C:/Users/nirva/Documents/R/BI-AD2020") #cambio de ruta del directorio de trabajo
3 list.files() #listar los archivos del directorio de trabajo
4 list.dirs() #listar las carpetas del directorio de trabajo
5 rentas <- read.csv("1_etl-Bikes.csv") #importar datos de csv
6 str(rentas) #muestra informacion compacta de un objeto
7 install.packages("dplyr") #instalacion de paquete
8 library(dplyr) #cargar paquete para su uso
9 norss <- filter(rentas, registered == 0 & (season == 1 | season == 2)) #filtra filas de acuerdo a condicion
10 dim(norss)
11 norss_su <- select(norss, season, casual) #filtra columnas especificas
12 str(norss_su)
13 norss_suc <- mutate(norss_su, ganancia = casual*5) #agrega un atributo o columna derivada de los existentes
14 gpo_norss <- group_by(norss_suc, season) #agrupa elementos por estacion del año
15 sum_norss <- summarise(gpo_norss, sum(casual), sum(ganancia)) #obtiene la cantidad de clientes casuales y ganancias
16 write.csv(sum_norss, "rpt_ganancias.txt", row.names=FALSE) #guarda un objeto en un archivo csv
```



LIMPIEZA DE LOS DATOS

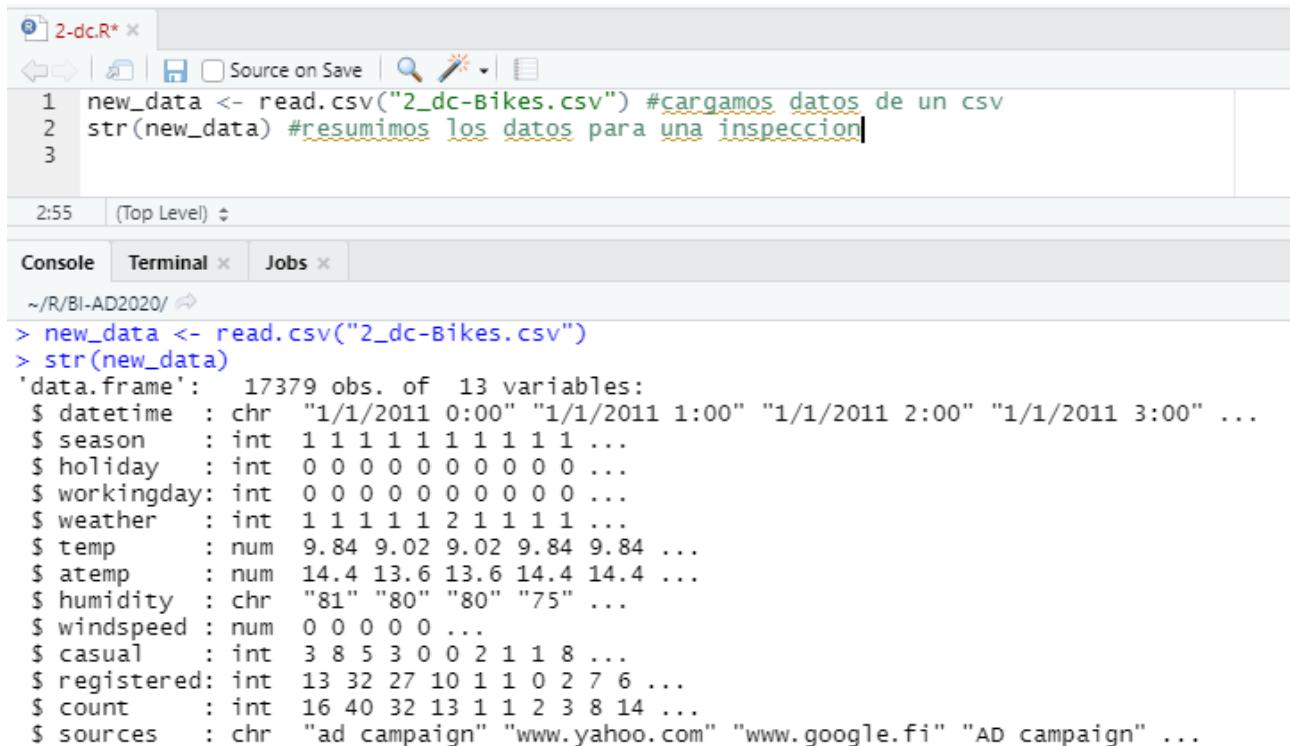
- Datos pobres o de baja calidad es la principal razón de problemas en análisis de inteligencia de negocios.
- La limpieza de los datos es el proceso de transformar datos brutos en datos utilizables.





- Como analista de negocios, te han llegado nuevos datos. A diferencia del primer conjunto de datos, los nuevos están en bruto. Es necesario que realices una limpieza a estos datos y tenerlos listos para futuros análisis.
- Siga el enfoque Resumir-Reparar-Convertir-Adaptar (Summarize-Fix-Convert-Adapt -SFCA-) para realizar la limpieza de los datos.

- Para resumir un conjunto de datos, se hace uso de la función **str(objeto)**



2-dc.R*

```
1 new_data <- read.csv("2_dc-Bikes.csv") #cargamos datos de un csv
2 str(new_data) #resumimos los datos para una inspección
3
```

2:55 (Top Level) ↴

Console Terminal × Jobs ×

```
~ /R/BI-AD2020/ ↴
> new_data <- read.csv("2_dc-Bikes.csv")
> str(new_data)
'data.frame': 17379 obs. of 13 variables:
 $ datetime : chr "1/1/2011 0:00" "1/1/2011 1:00" "1/1/2011 2:00" "1/1/2011 3:00" ...
 $ season   : int 1 1 1 1 1 1 1 1 1 ...
 $ holiday  : int 0 0 0 0 0 0 0 0 0 ...
 $ workingday: int 0 0 0 0 0 0 0 0 0 ...
 $ weather   : int 1 1 1 1 1 2 1 1 1 ...
 $ temp      : num 9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num 14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : chr "81" "80" "80" "75" ...
 $ windspeed : num 0 0 0 0 ...
 $ casual    : int 3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int 13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int 16 40 32 13 1 1 2 3 8 14 ...
 $ sources   : chr "ad campaign" "www.yahoo.com" "www.google.fi" "AD campaign" ...
```

¿Qué podemos observar?

- Otras funciones para ver características de los datos, son:
 - **`dim(objeto)`** : obtiene número de renglones y columnas
 - **`head(objeto)`** : permite observar los primeros 6 registros de los datos
 - **`tail(objeto)`** : permite observar los últimos 6 registros de los datos



- Los valores faltantes (NA) o nulos (NULL) provocan diversas dificultades para los métodos de análisis.
- Para encontrar valores faltantes, existe la función:

The screenshot shows the RStudio interface with the following details:

- Code Editor:** The script pane contains R code to read a CSV file, inspect its structure, and count NA values. The code is as follows:

```
1 new_data <- read.csv("2_dc-Bikes.csv") #cargamos datos de un csv
2 str(new_data) #resumimos los datos para una inspección
3 table(is.na(new_data)) #regresa una tabla con la cantidad de datos que son NA y los que no
```

- Console:** The output shows the results of the `table` command:

```
~ /R/BI-AD2020/ > table(is.na(new_data)) #regresa una tabla con la cantidad de datos que son NA y los que no
FALSE TRUE
225373 554
```

The right pane shows the **Data** view with the dataset **new_data** selected, displaying 17379 observations and 13 variables.



- Los valores faltantes o nulos provocan diversas dificultades para los métodos de análisis.
- Para encontrar valores faltantes, existe la función:
 - **is.na(objeto)**

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Shows an R script named "2-dc.R" with the following code:

```
1 new_data <- read.csv("2_dc-Bikes.csv") #cargamos datos de un csv
2 str(new_data) #resumimos los datos para una inspección
3 table(is.na(new_data)) #regresa una tabla con la cantidad de datos que son NA y los que no
```
- Console:** Shows the command `> table(is.na(new_data))` and its output:

```
~ /R/BI-AD2020/ >
> table(is.na(new_data)) #regresa una tabla con la cantidad de datos que son NA y los que no
FALSE    TRUE
225373     554
```
- Environment View:** Shows a data frame named "new_data" with 17379 observations and 13 variables.

- $\pm \text{Infinity}$: resulta de almacenar un 'numero muy grande o el resultado de una división entre cero.
- **NaN**: aplica a valores numéricos (reales y complejos) e indica que no es algo no es un numero.
- **NA (Not Available)**: Constante lógica que indica valor faltante
- **Null**: es un objeto devuelto cuando una expresión o función resulta en un valor indefinido.

```
> x <- c(1.5, Inf, -Inf, 3)
> is.infinite(x)
[1] FALSE TRUE TRUE FALSE
> x <- c(-Inf, -1, 0/0, 1, Inf)
> x
[1] -Inf -1 NaN 1 Inf
> is.nan(x)
[1] FALSE FALSE TRUE FALSE FALSE
> x <- c(-Inf, -1, 0/0, 1, Inf, NA)
> x
[1] -Inf -1 NaN 1 Inf NA
> is.na(x)
[1] FALSE FALSE TRUE FALSE FALSE TRUE
> x <- c(-Inf, -1, 0/0, 1, Inf, NULL, NULL, NA)
> x
[1] -Inf -1 NaN 1 Inf NA
> is.null(x)
[1] FALSE
> x <- c(NULL,NULL)
> is.null(x)
[1] TRUE
> x
NULL
```



- Para trabajar con los valores faltantes, se puede:
 - Eliminar las observaciones con problemas
 - Imputar los datos - reparar los datos faltantes basado en algún método, como:
 - Tomar muestras aleatorias
 - Usar la media de una variable o de un grupo
 - Generar valores basados en modelos predictivos

- Las funciones del paquete *stringr*, nos permiten detectar cadenas
 - **install.packages("stringr")**
 - **library(stringr)**
- Para encontrar los elementos con la cadena “NA”, se usa:
 - **str_detect(objeto, cadena)**

```
5 install.packages("stringr") #instalacion de paquete para reparar valores NA
6 library(stringr)
7 str_detect(new_data, "NA") #Encontrar una cadena en un conjunto de valores
8
8:1 (Top Level) RS
Console Terminal × Jobs ×
~/R/BI-AD2020/ ↵
> str_detect(new_data, "NA") #Encontrar una cadena en un conjunto de valores
[1] FALSE TRUE
Warning message:
In stri_detect_regex(string, pattern, negate = negate, opts_regex = opts(pattern)) :
  argument is not an atomic vector; coercing
```



- El valor “NA” se encuentra en la columna #13

The screenshot shows an RStudio interface. In the top-left corner, there are two code snippets:

```
8 table(is.na(new_data[13])) #Detectar valores NA en la columna 13
9 <--
```

Below this, the status bar shows "9:1 (Top Level) ⇣".

In the main console area, the command `table(is.na(new_data[13])) #Detectar valores NA en la columna 13` is entered, followed by its output:

FALSE	TRUE
16825	554

- ¿La solución? se verá al llegar a la parte de Adaptar en el proceso SFCA.

- Un error detectado en la primera parte de SFCA, es el que la variable humedad en lugar de ser numérica, es almacenada como tipo carácter. El problema se puede encontrar buscando todas las instancias con valores diferentes a los numéricos en dicha variable. Para esto, se usara la siguiente función del paquete *stringr*:

– coincidencias <- **str_subset(objeto, reg_ex)**

```
9 humedad_chr <- str_subset(new_data$humidity, "[a-z A-Z]") #detecta las cadenas de caracteres en humidity
10 print(humedad_chr) #imprime las cadenas de caracteres encontradas en humidity
9:105 (Top Level) ⇡ R Script
Console Terminal × Jobs ×
~/R/BI-AD2020/ ↗
> humedad_chr <- str_subset(new_data$humidity, "[a-z A-Z]")
> print(humedad_chr)
[1] "x61"
```

- El error encontrado parece ser tipográfico y se decide no eliminar el registro, sino imputarlo:

```
11 indice_x61 <- str_detect(new_data$humidity, humedad_chr) #encuentra los índices de celdas con errores
12 print(new_data[indice_x61,]) #imprime los registros con el error en la 13va columna
12:84 (Top Level) ▾
Console Terminal × Jobs ×
~ / R / BI - AD2020 / ↵
> indice_x61 <- str_detect(new_data$humidity, humedad_chr) #encuentra los índices de celdas con errores
> print(new_data[indice_x61,]) #imprime las posiciones con el error en la 13va columna
  datetime season holiday workingday weather  temp atemp humidity windspeed casual registered
14177 3 / 18 / 2012 21:00      3       0       0     1 27.06 31.06      x61        0       90      248
  count sources
14177    338 www.bing.com
```

- Para reparar este error, se usará la siguiente función del paquete *stringr*:
 - objeto <- **str_replace_all(objeto, índices, nvo_val)**

Reparando valores defectuosos: Valores faltantes



```
13 new_data$humidity <- str_replace_all(new_data$humidity, humedad_chr, "61") #reemplaza el valor x61 por 61
14 print(new_data[indice_x61,]) #imprime los registros corregidos
15
```

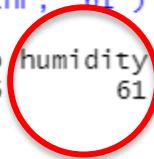
15:1 (Top Level) ↴

R Script

Console Terminal × Jobs ×

~/R/BI-AD2020/ ↴

```
> new_data$humidity <- str_replace_all(new_data$humidity, humedad_chr, "61") #reemplaza el valor x61 por 61
> print(new_data[indice_x61,]) #imprime los registros corregidos
   datetime season holiday workingday weather  temp atemp humidity windspeed casual registered
14177 8/18/2012    21:00      3       0       0     1 27.06 31.06      61          0       90      248
  count sources
14177    338 www.bing.com
```



- Diferentes tipos de datos encontrados en el archivo de la compañía Bicicleta Compartida:

Data type	Explanation	Example
Numeric	A number having a decimal value	9.84
Integer	A number without decimals	3
Character	A string variable	"www.google.com"
Factor	A categorical variable that has a character and integer representation	"ad campaign", "blog": 1, 2
Date	A date or time in various formats	2016-02-16 18:56:57 EST

¿Por qué se necesita conversión entre tipos de datos?

- Para realizar una conversión de carácter a número, R cuenta con funciones de conversión de tipo:
 - `as.{type}`
- En los datos nuevos, se ha corregido el problema tipográfico en `humidity`, sin embargo, el tipo de dato aun es carácter.
Solución:

```
15 str(new_data$humidity) #visualizar un resumen de los datos
16 new_data$humidity <- as.numeric(new_data$humidity) #cambia los datos en humidity de caracter a numero
17 str(new_data$humidity) #visualizar un resumen de los datos|
```

17:59 (Top Level) ↴

R

Console Terminal × Jobs ×

~/R/BI-AD2020/ ↗

```
> str(new_data$humidity) #visualizar un resumen de los datos
chr [1:17379] "81" "80" "80" "75" "75" "75" "80" "86" "75" "76" "76" "81" "77" "72" "72" "77" "82" ...
> new_data$humidity <- as.numeric(new_data$humidity) #cambia los datos en humidity de caracter a numero
> str(new_data$humidity) #visualizar un resumen de los datos
num [1:17379] 81 80 80 75 75 75 80 86 75 76 ...
```

- En R, los factores son la forma mas natural de representar puntos de datos que caen dentro de un número finito de distintas categorías; en lugar de pertenecer a un espacio continuo de valores.

- Para transformar variables numéricas que realmente son categóricas, se hace a través de factores, la forma es la siguiente:
 - `obj_factor <- factor(objeto, levels = v_niveles,
labels = v_etiquetas)`

- Es conveniente convertir los valores de las columnas *holiday* y *workingday* de 0 y 1 a “no” y “yes”, respectivamente, para una mejor lectura de los datos:

```
18 str(new_data$holiday) #visualizar un resumen de los datos
19 str(new_data$workingday) #visualizar un resumen de los datos
20 #convertir valores de 0 o 1 en columnas holiday y working day a factores con etiquetas no o si
21 new_data$holiday <- factor(new_data$holiday, levels = c(0,1), labels = c("no", "yes"))
22 new_data$workingday <- factor(new_data$workingday, levels = c(0,1), labels = c("no", "yes"))
23 str(new_data$holiday) #visualizar un resumen de los datos
24 str(new_data$workingday) #visualizar un resumen de los datos
25 n
```

19:54 | (Top Level) ↵

```
Console Terminal × Jobs ×
~/R/BI-AD2020/ ↵
> str(new_data$holiday) #visualizar un resumen de los datos
int [1:17379] 0 0 0 0 0 0 0 0 0 0 ...
> str(new_data$workingday) #visualizar un resumen de los datos
int [1:17379] 0 0 0 0 0 0 0 0 0 0 ...
> #convertir valores de 0 o 1 en columnas holiday y working day a factores con etiquetas no o si
> new_data$holiday <- factor(new_data$holiday, levels = c(0,1), labels = c("no", "yes"))
> new_data$workingday <- factor(new_data$workingday, levels = c(0,1), labels = c("no", "yes"))
> str(new_data$holiday) #visualizar un resumen de los datos
Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
> str(new_data$workingday) #visualizar un resumen de los datos
Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
```



- Para transformar variables numéricas que realmente son categóricas, se hace a través de factores ordenados, la forma es la siguiente:
 - `obj_factor <- factor(objeto, levels = v_niveles,
labels = v_etiquetas, ordered = TRUE)`

- Ahora, resulta conveniente convertir los valores de las columnas *season* y *weather* en factores ordenados, para tener la noción de secuencia en las estaciones o variación del clima:

```
25 str(new_data$season) #visualizar un resumen de los datos
26 str(new_data$weather) #visualizar un resumen de los datos
27 new_data$season <- factor(new_data$season, levels = c(1, 2, 3, 4),
28                           labels = c("spring", "summer", "fall", "winter"),
29                           ordered = TRUE)
30 new_data$weather <- factor(new_data$weather, levels = c(1, 2, 3, 4),
31                            labels = c("clr_part_cloud", "mist_cloudy", "lt_rain_snow", "hv_y_rain_snow"),
32                            ordered = TRUE)
33 str(new_data$season) #visualizar un resumen de los datos
34 str(new_data$weather) #visualizar un resumen de los datos
35 n
```

```
34:28 (Top Level) ⇡ R
Console Terminal × Jobs ×
```

```
~/R/BI-AD2020/ ↵
> str(new_data$season) #visualizar un resumen de los datos
int [1:17379] 1 1 1 1 1 1 1 1 1 1 ...
> str(new_data$weather) #visualizar un resumen de los datos
int [1:17379] 1 1 1 1 1 2 1 1 1 1 ...
> new_data$season <- factor(new_data$season, levels = c(1, 2, 3, 4),
+                           labels = c("spring", "summer", "fall", "winter"),
+                           ordered = TRUE)
> new_data$weather <- factor(new_data$weather, levels = c(1, 2, 3, 4),
+                           labels = c("clr_part_cloud", "mist_cloudy", "lt_rain_snow", "hv_y_rain_snow"),
+                           ordered = TRUE)
> str(new_data$season) #visualizar un resumen de los datos
ord.factor w/ 4 Levels "spring" <"summer" <...: 1 1 1 1 1 1 1 1 1 ...
> str(new_data$weather) #visualizar un resumen de los datos
ord.factor w/ 4 levels "clr_part_cloud" <...: 1 1 1 1 1 2 1 1 1 1 ...
```



- Las funciones del paquete *lubridate*, nos permiten manipular fechas
 - `install.packages("lubridate")`
 - `library(lubridate)`
- Las funciones del paquete *lubridate* son nombradas usando letras representando el orden de los datos de entrada, por ejemplo:
 - `dmy_hm(objeto)` : muestra la fecha en formato de día, mes, año, hora y minutos
 - `mdy_hm(objeto)` : muestra la fecha en formato de mes, día, año, hora y minutos

- Convertir los valores de la columna *datetime* de carácter a fecha en formato mes-día-año:

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Shows R code in the script pane:

```
36 library(lubridate)
37 str(new_data$datetime) #visualizar un resumen de los datos
38 new_data$datetime <- mdy_hm(new_data$datetime)
39 str(new_data$datetime) #visualizar un resumen de los datos
40
```
- Console:** Shows the output of the executed code:

```
40:1 (Top Level) ▾
[1] ~ /R/BI-AD2020/ ↵
> str(new_data$datetime) #visualizar un resumen de los datos
chr [1:17379] "1/1/2011 0:00" "1/1/2011 1:00" "1/1/2011 2:00" "1/1/2011 3:00" "1/1/2011 4:00" ...
> new_data$datetime <- mdy_hm(new_data$datetime)
> str(new_data$datetime) #visualizar un resumen de los datos
POSIXct[1:17379], format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" "2011-01-01 02:00:00" "2011-01-01 03:00:00" ...
```
- Terminal:** Not visible in the screenshot.
- Jobs:** Not visible in the screenshot.
- R Script:** Tab is visible in the top right corner.

Nuestro conjunto de datos hasta ahora...



UNIVERSIDAD DE
GUANAJUATO

```
> str(new_data)
'data.frame': 17379 obs. of 13 variables:
 $ datetime : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" "2011-01-01 02:00:00" ...
 $ season   : Ord.factor w/ 4 levels "spring"<"summer"<..: 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
 $ workingday: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
 $ weather   : Ord.factor w/ 4 levels "clr_part_cloud"<..: 1 1 1 1 1 1 2 1 1 1 ...
 $ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
 $ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
 $ humidity  : num  81 80 80 75 75 75 80 86 75 76 ...
 $ windspeed : num  0 0 0 0 0 ...
 $ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
 $ registered: int  13 32 27 10 1 1 0 2 7 6 ...
 $ count     : int  16 40 32 13 1 1 2 3 8 14 ...
 $ sources   : chr  "ad campaign" "www.yahoo.com" "www.google.fi" "AD campaign" ...
> head(new_data)
   datetime season holiday workingday      weather temp atemp humidity windspeed casual
1 2011-01-01 spring     no      no clr_part_cloud 9.84 14.395    81 0.0000    3
2 2011-01-01 spring     no      no clr_part_cloud 9.02 13.635    80 0.0000    8
3 2011-01-01 spring     no      no clr_part_cloud 9.02 13.635    80 0.0000    5
4 2011-01-01 spring     no      no clr_part_cloud 9.84 14.395    75 0.0000    3
5 2011-01-01 spring     no      no clr_part_cloud 9.84 14.395    75 0.0000    0
6 2011-01-01 spring     no      no mist_cloudy  9.84 12.880    75 6.0032    0
   registered count sources
1          13    16 ad campaign
2          32    40 www.yahoo.com
3          27    32 www.google.fi
4          10    13 AD campaign
5           1     1 Twitter
6           1     1 www.bing.com
```



- Revisemos que valores hay en la columna *source*. Esto lo logramos mediante la función:
 - **unique(*objeto*)**

```
40 unique(new_data$sources) #permite obtener los valores únicos en un vector
41
41:1 (Top Level) ⇣
Console Terminal × Jobs ×
~/R/BI-AD2020/ ↵
> unique(new_data$sources) #permite obtener los valores únicos en un vector
[1] "ad campaign"      "www.yahoo.com"    "www.google.fi"   "AD campaign"      "Twitter"
[6] "www.bing.com"      "www.google.co.uk" "facebook page"  "Ad Campaign"      "Twitter"
[11] NA                 "www.google.com"  "direct"          "blog"
```

¿Convendrá hacer de esta columna un factor?

¿Cuántos valores tendría nuestro factor?



- Empecemos por estandarizar los valores en *source*.
- Para evitar duplicados por variaciones de mayúsculas y minúsculas, dejemos todo en minúsculas con la siguiente función:
 - objeto <- **tolower**(*objeto_str*)
- Para evitar redundancia de valores por espacios “de más”, usaremos la función:
 - objeto <- **str_trim**(*objt_str*)
- Por último, cambiaremos los valores NA por la palabra “unknown” con una función ya vista (**is.na**).



```
41 new_data$sources <- tolower(new_data$sources) #cambia todas las letras a minusculas
42 unique(new_data$sources)
43 new_data$sources <- str_trim(new_data$sources) #quita espacios en blanco al inicio y fin de la cadena
44 unique(new_data$sources)
45 indice_na <- is.na(new_data$sources) #detecta las incidencias NA
46 new_data$sources[indice_na] <- "unknown"
47 unique(new_data$sources)
48
```

40:11 (Top Level) ↴

R S

Console Terminal × Jobs ×

~/R/BI-AD2020/ ↴

```
> new_data$sources <- tolower(new_data$sources) #cambia todas las letras a minusculas
> unique(new_data$sources)
[1] "ad campaign"      "www.yahoo.com"    "www.google.fi"    "twitter"        "www.bing.com"
[6] "www.google.co.uk" "facebook page"   "twitter"         "NA"              "www.google.com"
[11] "direct"           "blog"
> new_data$sources <- str_trim(new_data$sources) #quita espacios en blanco al inicio y fin de la cadena
> unique(new_data$sources)
[1] "ad campaign"      "www.yahoo.com"    "www.google.fi"    "twitter"        "www.bing.com"
[6] "www.google.co.uk" "facebook page"   "NA"              "www.google.com"  "direct"
[11] "blog"
> indice_na <- is.na(new_data$sources) #detecta las incidencias NA
> new_data$sources[indice_na] <- "unknown"
> unique(new_data$sources)
[1] "ad campaign"      "www.yahoo.com"    "www.google.fi"    "twitter"        "www.bing.com"
[6] "www.google.co.uk" "facebook page"   "unknown"         "www.google.com"  "direct"
[11] "blog"
```

Contar con 11 diferentes categorías ¿es suficiente o adecuado?



- Según la *Ley de Miller* (1956): La mente humana trabaja mejor con “cosas” o categorías que aparecen en cantidades de 7 (+/- 2). Con base a esto, intentemos quedarnos con una cantidad de categorías entre 5-7.
- Como tip: Las personas de publicidad, no consideran importante conocer el motor de búsqueda sino que una persona encontró la compañía en la Web.



- Las funciones del paquete *DataCombine*, nos permiten combinar datos para definir nuevas categorías
 - `install.packages("DataCombine")`
 - `library(DataCombine)`



- En R, los Data Frames son la forma de presentar un conjunto de datos formado por una colección de observaciones registradas para una o más variables.
- Al igual que las listas, los Data Frames no tienen restricción en los tipos de datos de las variables.
- Un aspecto importante, es que los miembros del Data Frame deben ser vectores del mismo tamaño.



- Creamos un patrón para los sitios web mediante una expresión regular: "(www.[a-z]*.[a-z]*)"
- Obtenemos los valores que coinciden con este criterio de búsqueda, esto a través de las funciones **str_subset** y **unique**.
- Generamos un vector del tamaño de las incidencias de sitio web, esto mediante la función:
 - objeto <- **rep(valor, cantidad)**



- Definimos un data frame que contenga las incidencias de sitios web y la cadena de remplazo:
 - objeto <- **data.frame**(*col1=vec1, ... , coln=vecn*)
- Reemplazamos las incidencias de sitios web por la palabra “web” con base al data frame creado
 - objeto <- **FindReplace**(*data=objeto, var=columna, from=val_original_df, to=val_cambio_df, exact=FALSE*)



- Definimos la columna *source* como factor
 - objeto <- **as.factor(valores)**
- Escribimos un csv con los datos limpios

Adaptando cadenas de caracteres a un estándar de valor



```
48 install.packages("DataCombine") #instalacion de paquete para combinar datos
49 library(DataCombine)
50 wb <- "(www. [a-z]*.[a-z]*)" #expresion regular para buscar combinaciones tipo sitio web
51 sts <- unique(str_subset(new_data$sources, wb)) #se obtienen las cadenas que son tipo sitio web
52 rplc <- rep("web", length(sts)) #crea un vector con la palabra web de tamaño tantos sitios haya
53 rplcmts <- data.frame(ori = sts, rep = rplc) #crea un vector con la asociacion de sitio y reemplazo
54 #FindReplace permite cambiar datos en from por los de to en un objeto cadena
55 new_data <- FindReplace(data=new_data, Var="sources", rplcmts, from="ori", to="rep", exact = FALSE)
56 unique(new_data$sources)
57 new_data$sources <- as.factor(new_data$sources)
58 str(new_data)
59 write.csv(new_data,"cleaned_new_data.csv", row.names=FALSE)
60
61
```

32:39 (Top Level) R

onsole Terminal × Jobs ×

```
./R/BI-AD2020/ ↵
?FindReplace
wb <- "(www. [a-z]*.[a-z]*)" #expresion regular para buscar combinaciones tipo sitio web
sts <- unique(str_subset(new_data$sources, wb)) #se obtienen las cadenas que son tipo sitio web
rplc <- rep("web", length(sts)) #crea un vector con la palabra web de tamaño tantos sitios haya
rplcmts <- data.frame(ori = sts, rep = rplc) #crea un vector con la asociacion de sitio y reemplazo
#FindReplace permite cambiar datos en from por los de to en un objeto cadena
new_data <- FindReplace(data=new_data, var="sources", rplcmts, from="ori", to="rep", exact = FALSE)
unique(new_data$sources)
1] "ad campaign"      "web"           "twitter"        "facebook page" "unknown"       "direct"
7] "blog"
new_data$sources <- as.factor(new_data$sources)
str(new_data)
data.frame': 17379 obs. of 13 variables:
$ datetime : POSIXct, format: "2011-01-01 00:00:00" "2011-01-01 01:00:00" "2011-01-01 02:00:00" ...
$ season   : Ord.factor w/ 4 levels "spring"<"summer"<..: 1 1 1 1 1 1 1 1 1 ...
$ holiday  : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
$ workingday: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 ...
$ weather   : Ord.factor w/ 4 levels "clr_part_cloud"<...: 1 1 1 1 1 2 1 1 1 ...
$ temp      : num  9.84 9.02 9.02 9.84 9.84 ...
$ atemp     : num  14.4 13.6 13.6 14.4 14.4 ...
$ humidity  : num  81 80 80 75 75 75 80 86 75 76 ...
$ windspeed : num  0 0 0 0 0 ...
$ casual    : int  3 8 5 3 0 0 2 1 1 8 ...
$ registered: int  13 32 27 10 1 1 0 2 7 6 ...
$ count     : int  16 40 32 13 1 1 2 3 8 14 ...
$ sources   : Factor w/ 7 levels "ad campaign",...: 1 7 7 1 5 7 1 7 7 7 ...
```

UNIVERSIDAD DE
GUANAJUATO

