

Getting and Cleaning Data

Lección 1

The screenshot displays the RStudio interface. The editor window contains R code for creating a directory and downloading a file from a URL. The console window shows a summary of the current lesson, mentioning grouped data, dplyr functions, and a Coursera credit offer. The R Documentation pane on the right shows the help() function documentation, including its description, usage, arguments, and details about the topic parameter.

R Code in Editor:

```
##### Crear una carpeta para guardar los datos descargados #####
1 if(!file.exists("data")){dir.create("data")}
2
3 ##### Descargar datos de este curso #####
4 url <- "https://data.baltimorecity.gov/api/views/2tuc-2s/access.json?access_token=1234567890"
5 download.file(url, destfile = "data/paquetes_descargados.json", quiet = TRUE)
6
7 # Crear una carpeta para guardar los datos descargados
```

Console Output:

```
| We'll look at grouped data in the next lesson, but the idea is that summarize() can give
| you the requested value FOR EACH group in your dataset.
...
|=====| 98%
| In this lesson, you learned how to manipulate data using dplyr's five main functions. In
| the next lesson, we'll look at how to take advantage of some other useful features of
| dplyr to make your life as a data analyst much easier.
...
|=====| 100%
| Would you like to receive credit for completing this course on Coursera.org?
1: Yes
2: No
Selection: |
```

R Documentation (help() function):

Description
help is the primary interface to the help systems.

Usage
help(topic, package = NULL, lib.loc = NULL, verbose = getOption("verbose"), try.all.packages = getOption("help.try.all"), help_type = getOption("help_type"))

Arguments
topic usually, a [name](#) or character string specifying the topic for which help is sought. A character string (enclosed in explicit single or double quotes) is always taken as naming a topic.
If the value of topic is a length-one character vector the topic is taken to be the value of the only element. Otherwise topic must be a name or a [reserved](#)

Lección 2

The screenshot displays the RStudio interface with the following components:

- Script Editor:** Contains R code using `dplyr` to filter and arrange data based on `size_mb`.


```
1 # arrange() the result by size_mb, in descending order.
2 #
3 # If you want your results printed to the console, add
4 # print to the end of your chain.
5
6 cran %>%
7   select(ip_id, country, package, size_mb) %>%
8   mutate(size_mb = size / 2^20) %>%
9   filter(size_mb <= 0.5) %>%
10  # Your call to arrange() goes here
11  arrange(desc(size_mb))
```
- Console:** Shows the output of the script execution, including a message about Coursera credit.


```
| That's a job well done!
|===== | 98
%
| In this lesson, you learned about grouping and chaining using dplyr. You combined
| some of the things you learned in the previous lesson with these more advanced ideas
| to produce concise, readable, and highly effective code. Welcome to the wonderful
| world of dplyr!
...
|===== | 100
%
| Would you like to receive credit for completing this course on Coursera.org?
```
- Environment Pane:** Displays the `group_by` function from the `dplyr` package.

group_by (dplyr) R Documentation

Group a tbl by one or more variables.

Description

Most data operations are useful done on groups defined by variables in the the dataset. The `group_by` function takes an existing tbl and converts it into a grouped tbl where operations are performed "by group".

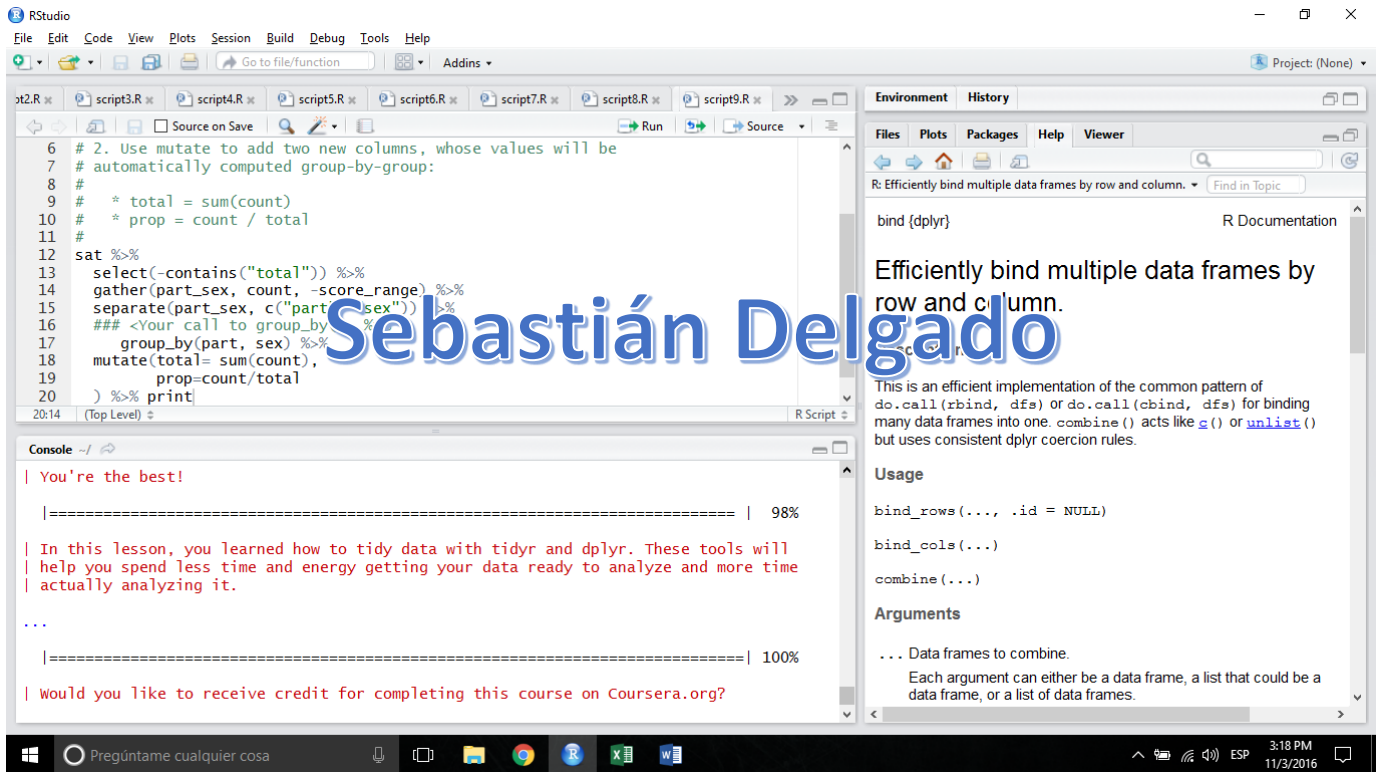
Usage

```
group_by(.data, ..., add = FALSE)
group_by_(.data, ..., .dots, add = FALSE)
```

Arguments

 - `.data` a tbl
 - `...` variables to group by. All tbls accept variable names, some will also accept functions of variables. Duplicated groups will be silently dropped.
 - `add` By default, when `add = FALSE`, `group_by` will override existing groups. To instead add to the existing groups, use `add = TRUE`.

Lección 3



RStudio interface showing the script editor, console, and help pane. The script editor contains R code using dplyr functions. The console shows progress bars and a message about Coursera credit. The help pane shows documentation for the bind function.

```
# 2. Use mutate to add two new columns, whose values will be
# automatically computed group-by-group:
#
# * total = sum(count)
# * prop = count / total
#
sat %>%
  select(-contains("total")) %>%
  gather(part_sex, count, -score_range) %>%
  separate(part_sex, c("part", "sex")) %>%
  ## <Your call to group_by()>
  group_by(part, sex) %>%
  mutate(total = sum(count),
         prop = count / total)
  ) %>% print()
```

Console output:

```
| You're the best!
|=====| 98%
| In this lesson, you learned how to tidy data with tidyr and dplyr. These tools will
| help you spend less time and energy getting your data ready to analyze and more time
| actually analyzing it.
...
|=====| 100%
| Would you like to receive credit for completing this course on Coursera.org?
```

Help pane content:

bind {dplyr} R Documentation

Efficiently bind multiple data frames by row and column.

This is an efficient implementation of the common pattern of `do.call(rbind, dfs)` or `do.call(cbind, dfs)` for binding many data frames into one. `combine()` acts like `c()` or `unlist()` but uses consistent dplyr coercion rules.

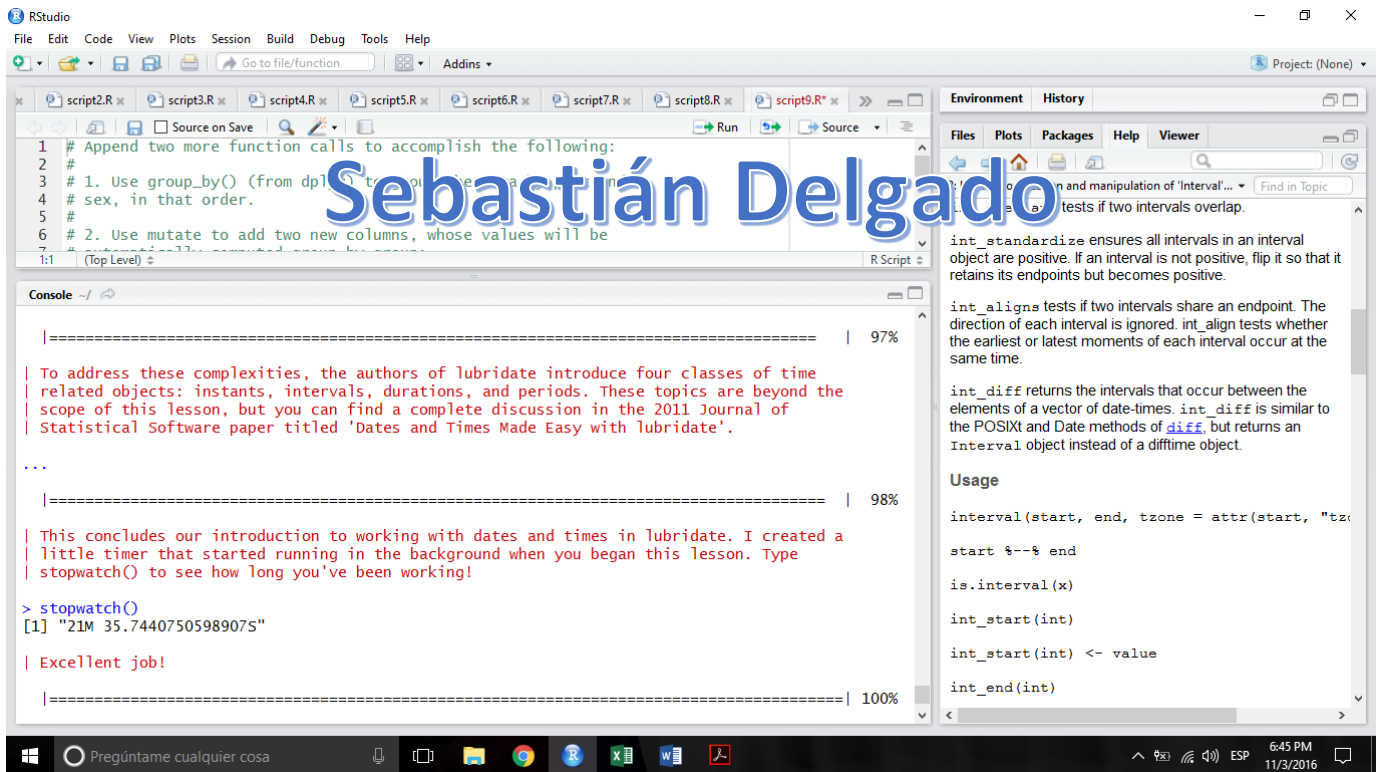
Usage

```
bind_rows(..., .id = NULL)
bind_cols(...)
combine(...)
```

Arguments

```
... Data frames to combine.
Each argument can either be a data frame, a list that could be a
data frame, or a list of data frames.
```

Lección 4



RStudio interface showing the script editor, console, and help pane. The script editor contains R code using lubridate functions. The console shows progress bars and a message about working with dates and times. The help pane shows documentation for the interval function.

```
# Append two more function calls to accomplish the following:
#
# 1. Use group_by() (from dplyr) to group by the variable 'sex', in that order.
#
# 2. Use mutate to add two new columns, whose values will be
# calculated as follows:
#   - 'total' is the sum of 'count' for each 'sex'
#   - 'prop' is 'count' divided by 'total'
#
```

Console output:

```
|=====| 97%
| To address these complexities, the authors of lubridate introduce four classes of time
| related objects: instants, intervals, durations, and periods. These topics are beyond the
| scope of this lesson, but you can find a complete discussion in the 2011 Journal of
| Statistical Software paper titled 'Dates and Times Made Easy with lubridate'.
...
|=====| 98%
| This concludes our introduction to working with dates and times in lubridate. I created a
| little timer that started running in the background when you began this lesson. Type
| stopwatch() to see how long you've been working!
> stopwatch()
[1] "21M 35.74407505989075"
| Excellent job!
|=====| 100%
```

Help pane content:

interval R Documentation

Interval objects and manipulation of 'Interval' objects.

int_standardize ensures all intervals in an interval object are positive. If an interval is not positive, flip it so that it retains its endpoints but becomes positive.

int_aligns tests if two intervals share an endpoint. The direction of each interval is ignored. int_align tests whether the earliest or latest moments of each interval occur at the same time.

int_diff returns the intervals that occur between the elements of a vector of date-times. int_diff is similar to the POSIXt and Date methods of diff, but returns an Interval object instead of a difftime object.

Usage

```
interval(start, end, tzzone = attr(start, "tzzone"))
start %--% end
is.interval(x)
int_start(int)
int_start(int) <- value
int_end(int)
```