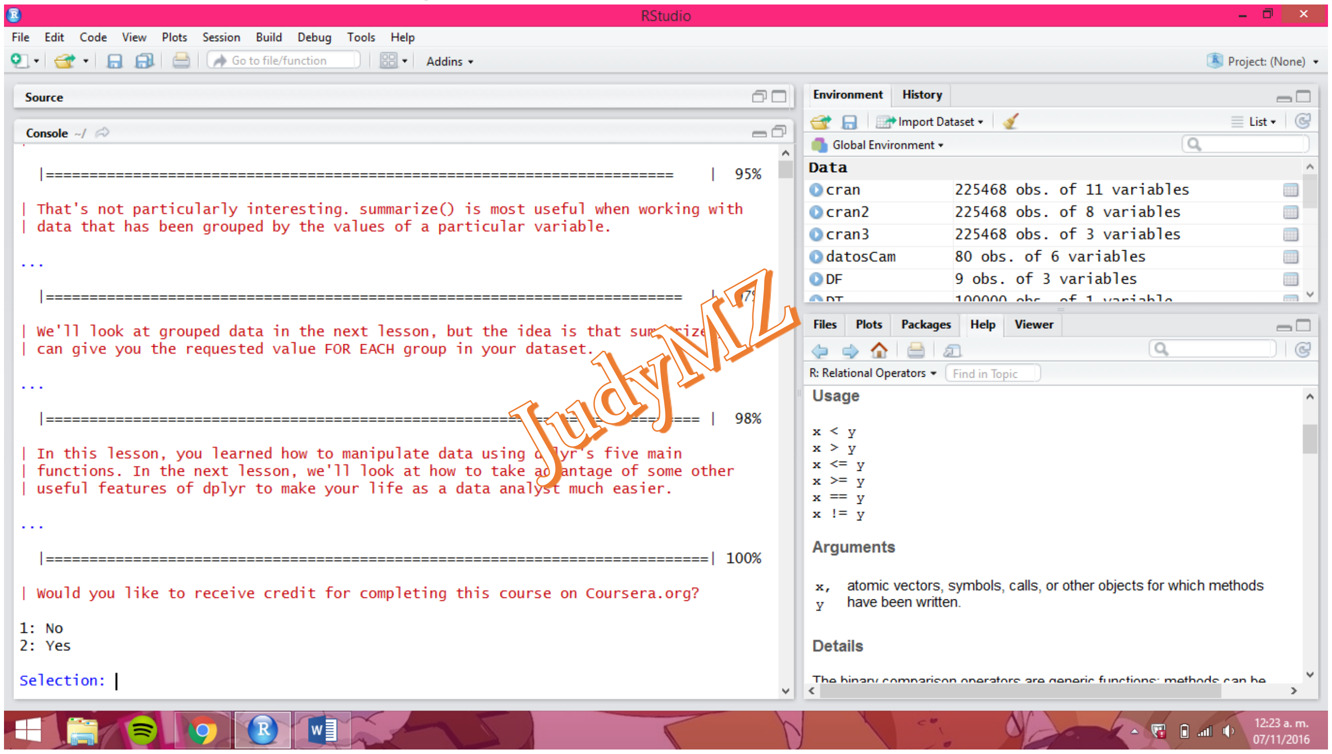
1: Manipulating Data with dplyr



```
  |===========================================================================    |  95%

| That's not particularly interesting. summarize() is most useful when working with
| data that has been grouped by the values of a particular variable.

...

  |=============================================================================  |  97%

| 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: No
2: Yes

Selection: 
```

```
Data
cran        225468 obs. of 11 variables
cran2       225468 obs. of 8 variables
cran3       225468 obs. of 3 variables
datosCam    80 obs. of 6 variables
DF          9 obs. of 3 variables
```

R: Relational Operators

Usage

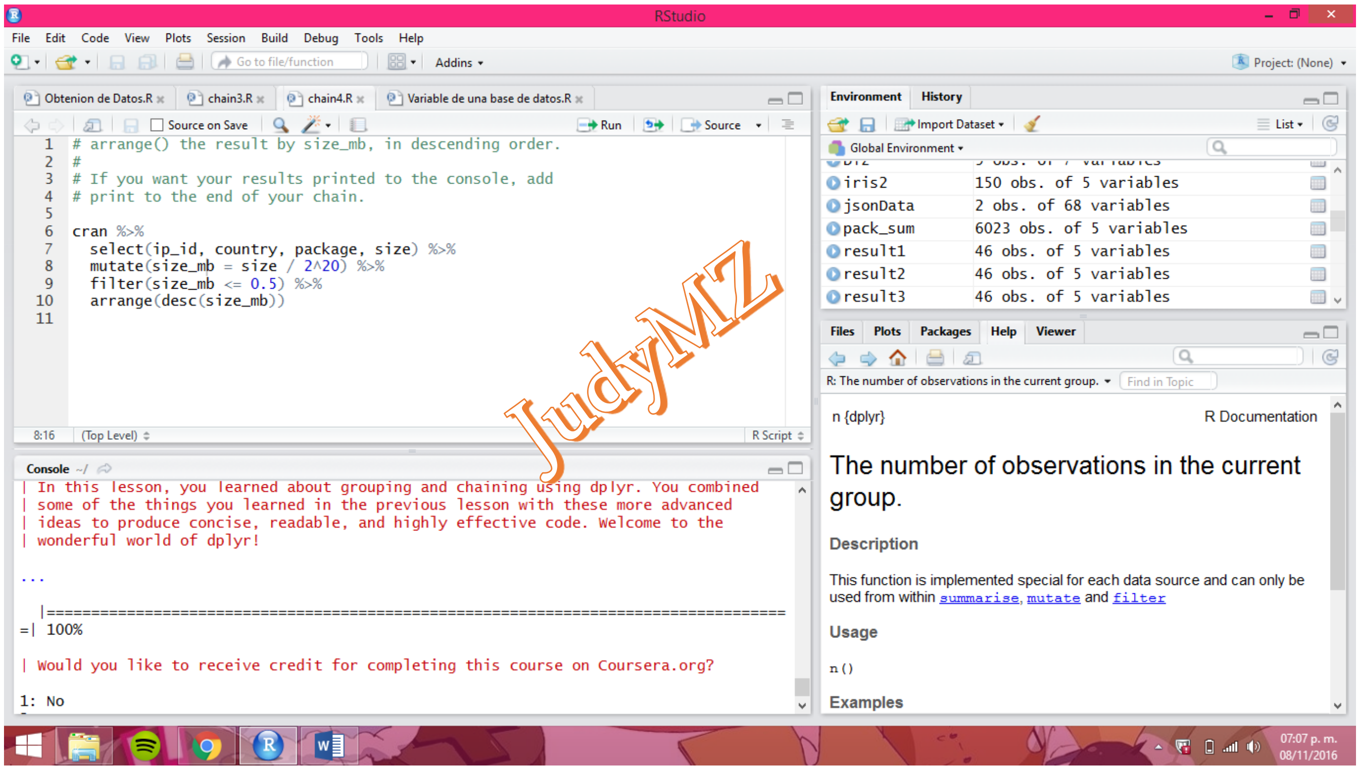
```
x < y
x > y
x <= y
x >= y
x == y
x != y
```

Arguments

x, y — atomic vectors, symbols, calls, or other objects for which methods have been written.

Details

2: Grouping and Chaining with dplyr



```
# arrange() the result by size_mb, in descending order.
#
# If you want your results printed to the console, add
# print to the end of your chain.

cran %>%
  select(ip_id, country, package, size) %>%
  mutate(size_mb = size / 2^20) %>%
  filter(size_mb <= 0.5) %>%
  arrange(desc(size_mb))
```

```
| 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?

1: No
```

```
iris2       150 obs. of 5 variables
jsonData    2 obs. of 68 variables
pack_sum    6023 obs. of 5 variables
result1     46 obs. of 5 variables
result2     46 obs. of 5 variables
result3     46 obs. of 5 variables
```

n {dplyr} R Documentation

The number of observations in the current group.

Description

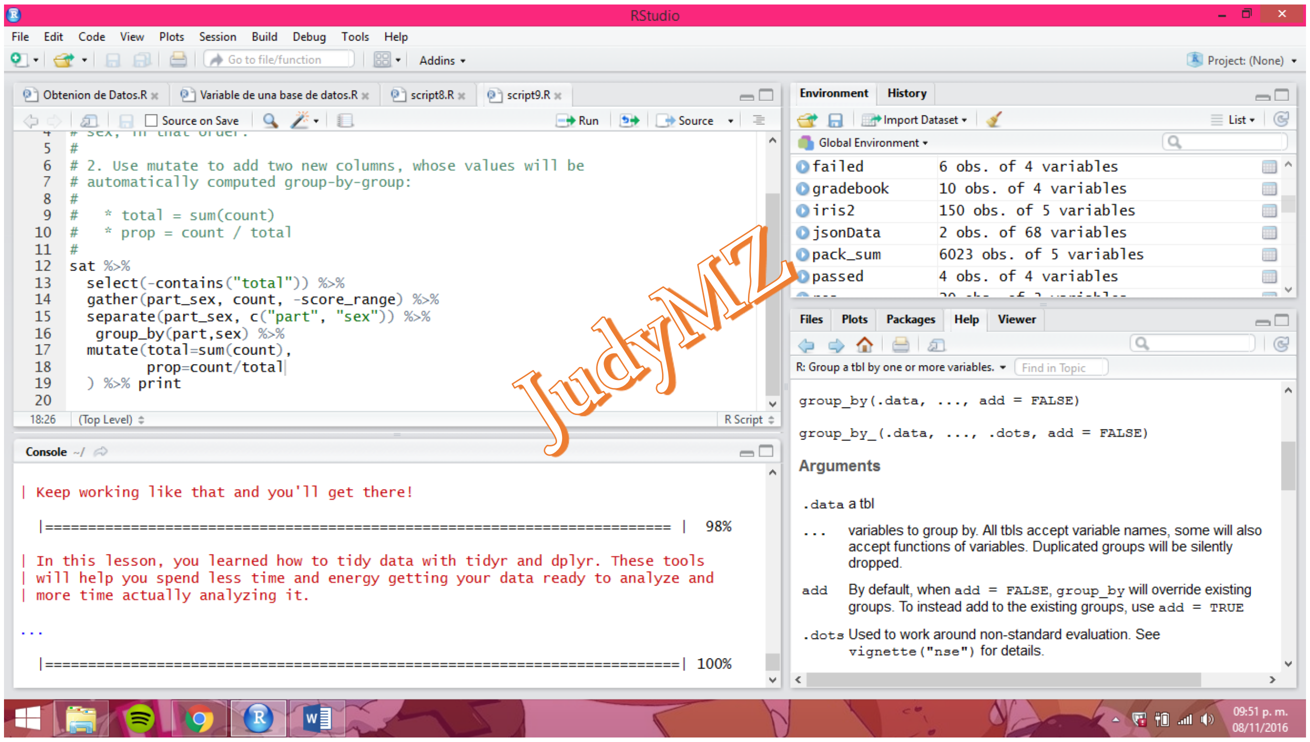
This function is implemented special for each data source and can only be used from within summarise, mutate and filter

Usage

`n()`

Examples

3: Tidying Data with tidyr



```r
# sex, in that order.
#
# 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")) %>%
   group_by(part,sex) %>%
  mutate(total=sum(count),
         prop=count/total
  ) %>% print
```

```
failed        6 obs. of 4 variables
gradebook     10 obs. of 4 variables
iris2         150 obs. of 5 variables
jsonData      2 obs. of 68 variables
pack_sum      6023 obs. of 5 variables
passed        4 obs. of 4 variables
```

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

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

Arguments

`.data` a tbl

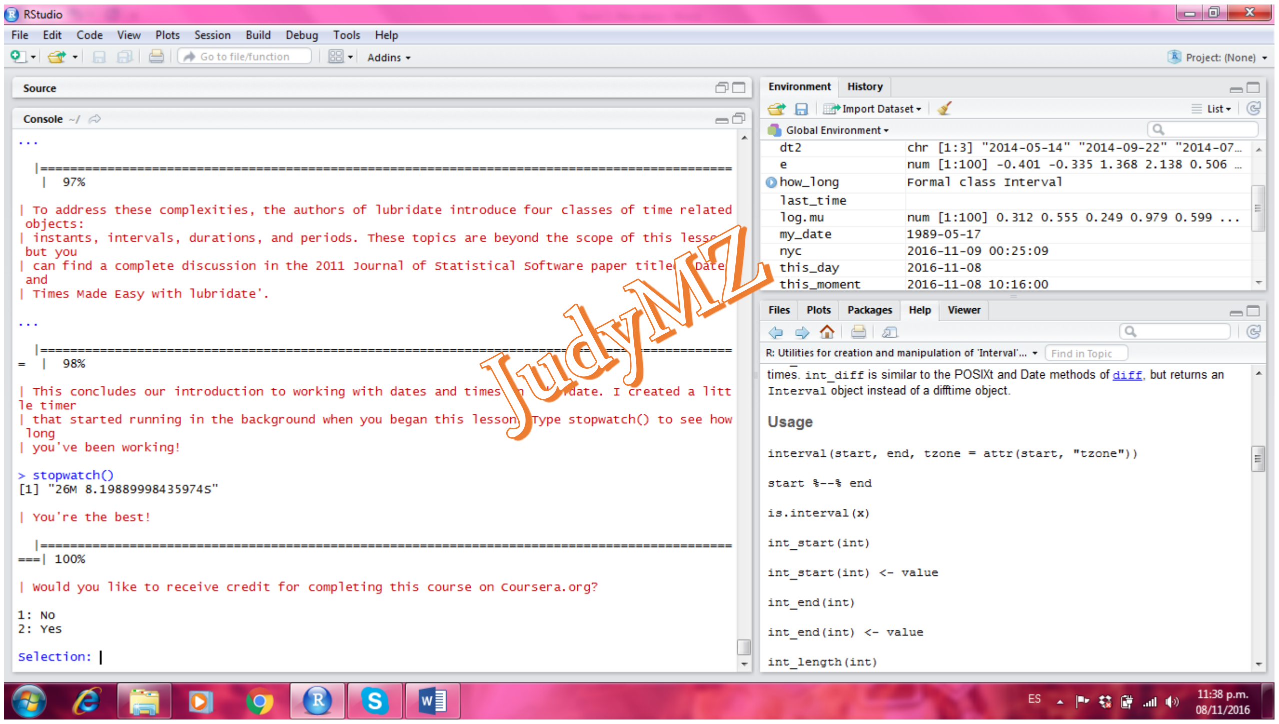
`...` variables to group by. All tbls accept variable names, some will also accept functions of variables. Duplicated groups will be silently dropped.

`add` By default, when `add = FALSE`, `group_by` will override existing groups. To instead add to the existing groups, use `add = TRUE`

`.dots` Used to work around non-standard evaluation. See `vignette("nse")` for details.

```
| Keep working like that and you'll get there!

  |=========================================================================   |  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%
```

4: Dates and Times with lubridate



```
...

  |=============================================================================
  |  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 litt
le timer
| that started running in the background when you began this lesson. Type stopwatch() to see how
 long
| you've been working!

> stopwatch()
[1] "26M 8.19889998435974S"

| You're the best!

  |=============================================================================
===| 100%

| Would you like to receive credit for completing this course on Coursera.org?

1: No
2: Yes

Selection: |
```

```
dt2           chr [1:3] "2014-05-14" "2014-09-22" "2014-07…
e             num [1:100] -0.401 -0.335 1.368 2.138 0.506 …
how_long      Formal class Interval
last_time
log.mu        num [1:100] 0.312 0.555 0.249 0.979 0.599 ...
my_date       1989-05-17
nyc           2016-11-09 00:25:09
this_day      2016-11-08
this_moment   2016-11-08 10:16:00
```

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, tzone = attr(start, "tzone"))

start %--% end

is.interval(x)

int_start(int)

int_start(int) <- value

int_end(int)

int_end(int) <- value

int_length(int)
```