Sorting Data Tables

Sorting Data Tables

Start of transcript. Skip to the end.

0:00 / 0:00 1.0x

RAFAEL IRIZARRY: When examining a data set,

it is often convenient to sort the table by different columns.

We know about the order and sort functions.

But for ordering entire tables, the function arrange, in dplyr,

is very useful.

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Textbook link

This video corresponds to the textbook section on sorting data frames.

Key points

- The `arrange()` function from **dplyr** sorts a data frame by a given column.
- By default, `arrange()` sorts in ascending order (lowest to highest). To instead sort in descending order, use the function `desc()` inside of `arrange()`.
- You can `arrange()` by multiple levels: within equivalent values of the first level, observations are sorted by the second level, and so on.
- The `top_n()` function shows the top results ranked by a given variable, but the results are not ordered. You can combine `top_n()` with `arrange()` to return the top results in order.

Code

```
# libraries and data
library(tidyverse)
library(dslabs)
data(murders)

# set up murders object
murders <- murders %>%
    mutate(murder_rate = total/population * 100000)

# arrange by population column, smallest to largest
murders %>% arrange(population) %>% head()

# arrange by murder rate, smallest to largest
murders %>% arrange(murder_rate) %>% head()

# arrange by murder rate in descending order
murders %>% arrange(desc(murder_rate)) %>% head()

# arrange by region alphabetically, then by murder rate within each reg
murders %>% arrange(region, murder_rate) %>% head()

# show the top 10 states with highest murder rate, not ordered by rate
murders %>% top_n(10, murder_rate)

# show the top 10 states with highest murder rate, ordered by rate
murders %>% arrange(desc(murder_rate)) %>% top_n(10)
```

Learn About Verified Certificates