

# Assessment 07 - Basic Data Wrangling

*Alessandro Corradini - Harvard Data Science Professional*

## dplyr

Load the dplyr package and the murders dataset.

```
library(dplyr)
library(dslabs)
data(murders)
```

You can add columns using the dplyr function `mutate`. This function is aware of the column names and inside the function you can call them unquoted. Like this:

```
murders <- mutate(murders, population_in_millions = population / 10^6)
```

Note that we can write `population` rather than `murders$population`. The function `mutate` knows we are grabbing columns from `murders`.

### Instructions

- Use the function `mutate` to add a murders column named `rate` with the per 100,000 murder rate.
- Make sure you redefine `murders` as done in the example code above.

Remember the murder rate is defined the total divided by the population size times 100,000

```
# Loading data
library(dslabs)
data(murders)

# Loading dplyr
library(dplyr)

##
## 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

# Redefine murders so that it includes column named rate with the per 100,000 murder rates
murders <- mutate(murders, rate = total / population * 100000)
```

## mutate

Note that if `rank(x)` gives you the ranks of `x` from lowest to highest, `rank(-x)` gives you the ranks from highest to lowest.

### Instructions

Use the function `mutate` to add a column `rank` containing the rank, from highest to lowest murder rate. Make sure you redefine `murders`.

```
# Note that if you want ranks from highest to lowest you can take the negative and then compute the rank
x <- c(88, 100, 83, 92, 94)
rank(-x)
```

```
## [1] 4 1 5 3 2
```

```
# Defining rate
rate <- murders$total/ murders$population * 100000

# Redefine murders to include a column named rank
# with the ranks of rate from highest to lowest
murders <- mutate(murders, rank = rank(-rate))
```

## select

With `dplyr` we can use `select` to show only certain columns. For example with this code we would only show the states and population sizes:

```
select(murders, state, population)
```

## Instructions

Use `select` to show the state names and abbreviations in `murders`. Just show it, do not define a new object.

```
# Load dplyr
library(dplyr)

# Use select to only show state names and abbreviations from murders
select(murders, state, abb)
```

```
##           state abb
## 1      Alabama  AL
## 2       Alaska  AK
## 3      Arizona  AZ
## 4     Arkansas  AR
## 5    California  CA
## 6      Colorado  CO
## 7   Connecticut  CT
## 8      Delaware  DE
## 9 District of Columbia DC
## 10     Florida  FL
## 11     Georgia  GA
## 12      Hawaii  HI
## 13      Idaho  ID
## 14     Illinois  IL
## 15     Indiana  IN
## 16       Iowa  IA
## 17      Kansas  KS
## 18     Kentucky  KY
## 19    Louisiana  LA
## 20       Maine  ME
## 21     Maryland  MD
## 22 Massachusetts  MA
## 23      Michigan  MI
## 24     Minnesota  MN
## 25    Mississippi  MS
## 26     Missouri  MO
```

```
## 27          Montana MT
## 28          Nebraska NE
## 29          Nevada NV
## 30      New Hampshire NH
## 31          New Jersey NJ
## 32          New Mexico NM
## 33          New York NY
## 34      North Carolina NC
## 35          North Dakota ND
## 36          Ohio OH
## 37          Oklahoma OK
## 38          Oregon OR
## 39          Pennsylvania PA
## 40          Rhode Island RI
## 41      South Carolina SC
## 42          South Dakota SD
## 43          Tennessee TN
## 44          Texas TX
## 45          Utah UT
## 46          Vermont VT
## 47          Virginia VA
## 48          Washington WA
## 49      West Virginia WV
## 50          Wisconsin WI
## 51          Wyoming WY
```

## filter

The `dplyr` function `filter` is used to choose specific rows of the data frame to keep. Unlike `select` which is for columns, `filter` is for rows. For example you can show just New York row like this:

```
filter(murders, state == "New York")
```

You can use other logical vector to filter rows.

## Instructions

Use `filter` to show the top 5 states with the highest murder rates. After we add murder rate and rank, do not change the `murders` dataset, just show the result. Note that you can filter based on the `rank` column.

```
# Add the necessary columns
murders <- mutate(murders, rate = total/population * 100000, rank = rank(-rate))

# Filter to show the top 5 states with the highest murder rates
filter(murders, rank <= 5)
```

##	state	abb	region	population	total	rate	rank
## 1	District of Columbia	DC	South	601723	99	16.452753	1
## 2	Louisiana	LA	South	4533372	351	7.742581	2
## 3	Maryland	MD	South	5773552	293	5.074866	4
## 4	Missouri	MO	North Central	5988927	321	5.359892	3
## 5	South Carolina	SC	South	4625364	207	4.475323	5

## filter with !=

We can remove rows using the `!=` operator. For example to remove Florida we would do this:

```
no_florida <- filter(murders, state != "Florida")
```

## Instructions

- Create a new data frame called `no_south` that removes states from the South region.
- How many states are in this category? You can use the function `nrow` for this.

```
# Use filter to create a new data frame no_south
no_south <- filter(murders, region != "South")
# Use nrow() to calculate the number of rows
nrow(no_south)
```

```
## [1] 34
```

## filter with %in%

We can also use the `%in%` to filter with `dplyr`. For example you can see the data from New York and Texas like this:

```
filter(murders, state %in% c("New York", "Texas"))
```

## Instructions

- Create a new data frame called `murders_nw` with only the states from the Northeast and the West.
- How many states are in this category?

```
# Create a new data frame called murders_nw with only the states from the northeast and the west
murders_nw <- filter(murders, region %in% c("Northeast", "West"))
# Number of states (rows) in this category
nrow(murders_nw)
```

```
## [1] 22
```

## filtering by two conditions

Suppose you want to live in the Northeast or West and want the murder rate to be less than 1. We want to see the data for the states satisfying these options. Note that you can use logical operators with `filter`:

```
filter(murders, population < 5000000 & region == "Northeast")
```

## Instructions

- Add a murder rate column and a rank column as done before
- Create a table, call it `my_states`, that satisfies both the conditions: it is in the Northeast or West and the murder rate is less than 1.
- Use `select` to show only the state name, the rate and the rank

```
# add the rate column
murders <- mutate(murders, rate = total / population * 100000, rank = rank(-rate))

# Create a table, call it my_states, that satisfies both the conditions
my_states <- filter(murders, region %in% c("Northeast", "West") & rate < 1)
# Use select to show only the state name, the murder rate and the rank
select(my_states, state, rate, rank)
```

```
##           state      rate rank
## 1      Hawaii 0.5145920   49
## 2       Idaho 0.7655102   46
## 3        Maine 0.8280881   44
## 4 New Hampshire 0.3798036   50
## 5        Oregon 0.9396843   42
## 6         Utah 0.7959810   45
```

```
## 7      Vermont 0.3196211  51
## 8      Wyoming 0.8871131  43
```

## Using the pipe %>%

The pipe %>% can be used to perform operations sequentially without having to define intermediate objects. After redefining `murders` to include rate and rank.

```
library(dplyr)
murders <- mutate(murders, rate = total / population * 100000, rank = (-rate))
```

in the solution to the previous exercise we did the following:

```
# Created a table
my_states <- filter(murders, region %in% c("Northeast", "West") & rate < 1)
```

```
# Used select to show only the state name, the murder rate and the rank
select(my_states, state, rate, rank)
```

The pipe %>% permits us to perform both operation sequentially and without having to define an intermediate variable `my_states`

For example we could have mutated and selected in the same line like this:

```
mutate(murders, rate = total / population * 100000, rank = (-rate)) %>%
  select(state, rate, rank)
```

Note that `select` no longer has a data frame as the first argument. The first argument is assumed to be the result of the operation conducted right before the %>%

## Instructions

- Repeat the previous exercise, but now instead of creating a new object, show the result and only include the state, rate, and rank columns.
- Use a pipe %>% to do this in just one line.

```
# Load library
library(dplyr)

## Define the rate column
murders <- mutate(murders, rate = total / population * 100000, rank = rank(-rate))

# show the result and only include the state, rate, and rank columns, all in one line
filter(murders, region %in% c("Northeast", "West") & rate < 1) %>% select(state, rate, rank)
```

```
##      state      rate rank
## 1    Hawaii 0.5145920  49
## 2    Idaho 0.7655102  46
## 3    Maine 0.8280881  44
## 4 New Hampshire 0.3798036  50
## 5    Oregon 0.9396843  42
## 6     Utah 0.7959810  45
## 7    Vermont 0.3196211  51
## 8    Wyoming 0.8871131  43
```

## mutate, filter and select

### Instructions

Now we will make `murders` the original table one gets when loading using `data(murders)`. Use just one line to create a new data frame, called, `my_states` that has murder rate and rank column, consider only states in the Northeast or West, which have a murder rate lower than 1 and contain only the state, rate, and rank columns. The line should have four components separated by three `%>%`.

- The original dataset `murders`
- A call to `mutate` to add the murder rate and the rank.
- A call to `filter` to keep only the states from the Northeast or West and that have a murder rate below 1
- A call to `select` that keeps only the columns with the state name, the murder rate and the rank.

The line should look something like this `my_states <- murders %>% mutate something %>% filter something %>% select something`. Please, make sure the columns in the final data frame must be in the order: `state`, `rate`, `rank`.

```
# Loading the libraries
```

```
library(dplyr)
```

```
data(murders)
```

```
# Create new data frame called my_states (with specifications in the instructions)
```

```
my_states <- murders %>% mutate(rate = total / population * 100000, rank = rank(-rate)) %>% filter(reg
```