

QTM 151

Week 5 – dplyr

Umberto Mignozzetti

Feb 26

Recap

- We learned:
 - `qplot`: quick way to make ggplot graphs.
 - `ggplotly`: transform ggplot objects into nice plotly viz.
 - `plot_ly`: create nice plotly graphs.
- Do you have any questions about any of these functions?
- The quiz will be posted today, at 4:00PM
- Our GitHub page is: <https://github.com/umbertomig/qtm151>

Today's Agenda

`dplyr` package for data manipulation:

- Observation/row:
 - filter
 - arrange
- Variable/column:
 - select
 - rename
 - mutate
- Get summaries:
 - summarise
 - group_by: changes the scope of each function from operating on the entire dataset to operating on it group-by-group

Getting Started

Getting Started: loading packages

```
# Loading tidyverse
```

```
library(tidyverse)
```

```
## — Attaching packages ————— tidyverse
```

```
## ✓ ggplot2 3.3.2      ✓ purrr 0.3.4
```

```
## ✓ tibble 3.0.4       ✓ dplyr 1.0.2
```

```
## ✓ tidyr 1.1.2        ✓ stringr 1.4.0
```

```
## ✓ readr 1.4.0        ✓ forcats 0.5.0
```

```
## — Conflicts ————— tidyverse_0.1.0
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::lag()     masks stats::lag()
```

Loading datasets

```
# Loading tips dataset
```

```
tips ← read.csv('https://raw.githubusercontent.com/umbertomig/qtn  
head(tips, 2)
```

```
##      obs totbill  tip sex smoker day  time size  
## 1      1   16.99 1.01  F      No Sun Night    2  
## 2      2   10.34 1.66  M      No Sun Night    3
```

```
# Loading PErisk dataset
```

```
PErisk ← read.csv('https://raw.githubusercontent.com/umbertomig/c  
head(PErisk, 2)
```

```
##      country courts      barb2 prsexp2 prscorr2      gdpw2  
## 1 Argentina      0 -0.7207754      1      3  9.69017  
## 2 Australia      1 -6.9077550      5      4 10.30484
```

select: variable selection

select: variable selection

- Suppose we want to select only the numeric variables in the `PErisk` dataset.
- We do the following:

```
PErisk_num ← select(PErisk, barb2, gdpw2)  
head(PErisk_num, 2)
```

```
##           barb2      gdpw2  
## 1 -0.7207754    9.69017  
## 2 -6.9077550   10.30484
```


select: variable selection

The syntax to use the `select` is the following:

```
dat_final ← select(dat_initial, var1, var2, var3, ...)
```

Your turn: select the variables `tip` and `totbill` in the `tips` dataset.

select: variable selection

We can select slices and by characteristics. For example:

```
aux ← select(PErisk, country, gdpw2)
```

```
aux ← PErisk %>%  
  select(country, gdpw2)
```

```
aux ← select(PErisk, -c(courts, prsexp2, prscorr2))
```

```
aux ← select(PErisk, courts:prsexp2)
```

```
aux ← select(PErisk, starts_with("co"))
```

select: variable selection

And the methods we can apply in the select are the following:

Method	Effect
<code>v1, v2, v3 (etc)</code>	Select given variables
<code>starts_with('xyz')</code>	Select starting with <code>xyz</code>
<code>ends_with('xyz')</code>	Select ending with <code>xyz</code>
<code>contains('xyz')</code>	Select variables that have <code>xyz</code> in their names
<code>vk:vn</code>	All variables between <code>vn</code> and <code>vk</code>
<code>-(vk:vn)</code>	All but <code>vk</code> to <code>vn</code>

We can even rename variables using `select`.

rename: change names variables

rename: change names variables

We can change the names of the variables using `rename`.

Basic syntax:

```
new_dat ← rename(dat,  
                  new_name1 = old_name1,  
                  ... )
```

rename: change names variables

Let's rename the `PErisk` courts to `indep_judiciary`:

```
PErisk_new ← rename(PErisk, indep_judiciary = courts)
head(PErisk_new, 2)
```

```
##      country indep_judiciary      barb2 prsexp2 prscorr2      gdpw2
## 1 Argentina           0 -0.7207754         1         3  9.69017
## 2 Australia           1 -6.9077550         5         4 10.30484
```

Your turn: rename the `totbill` to `totalbill` in the dataset `tips`.

filter: select pieces of the dataset

filter: select pieces of the dataset

We can select desired rows of the dataset using the function `filter`.

Basic syntax:

```
new_dat ← filter(dat, condition1, condition2, ... )
```


filter: select pieces of the dataset

Let's filter the `PErisk` to keep all countries without independent judiciary:

```
PErisk_new <- filter(PErisk,  
                      courts = 0)  
head(PErisk_new, 2)
```

```
##      country courts      barb2 prsexp2 prscorr2      gdpw2  
## 1  Argentina      0 -0.7207754      1      3 9.690170  
## 2 Bangladesh      0  0.7759748      1      0 8.379768
```

filter: select pieces of the dataset

- The filter operators are the following:

```
#| Operator | Meaning |
#|-----|-----|
#| < and ≤ | Smaller than and Smaller than or equal |
#| > and ≥ | Greater than and Treated than or equal |
#| =       | Equal |
#| ≠      | Different |
#| !      | Negation |
#| |      | Or |
#| &      | And |
```

Your turn: filter the `tips` dataset, keeping only the tips above \$4.52, by Non-smokers.

arrange: sort the dataset

arrange: sort the dataset

We can sort the dataset by levels, according to the needs of our analysis.

Basic syntax:

```
new_dat ← arrange(dat, var1,  
                  desc(var2),  
                  desc(var3), var4, ... )
```

arrange: sort the dataset

Let's arrange the `PErisk` by the `gdpw2`, in descending order:

```
PErisk_new ← arrange(PErisk, desc(gdpw2))  
head(PErisk_new, 2)
```

```
##           country courts      barb2 prsexp2 prscorr2      gdpw2  
## 1      Canada      1 -6.907755      5      5 10.41018  
## 2 Switzerland      1 -6.907755      5      5 10.34110
```

Your turn: Arrange the `tips` by the size of the tip, in ascending order. See the head and the tail of the arranged data.

mutate: make transformation in the
dataset

mutate: make transformation in the

We can make transformations in the dataset using the function `mutate`

Basic syntax:

```
new_dat ← mutate(dat,  
                  new_var = do_something_func(old_var),  
                  ... )
```

mutate: make transformation in the

Let's mutate the `PErisk`, to show the gdp in dollars:

```
PErisk_new ← mutate(PErisk,  
                     gdppc = exp(gdpw2))  
head(PErisk_new, 2)
```

```
##      country courts      barb2 prsexp2 prscorr2      gdpw2      gdppc  
## 1 Argentina      0 -0.7207754      1      3  9.69017 16157.99  
## 2 Australia      1 -6.9077550      5      4 10.30484 29876.87
```

Your turn: Let's find the percentage of the tips based on the total bill. Divide the tip by the total bill, and multiply by 100. Sort the data by this variable.

summarise: summaries of the dataset

summarise: summaries of the dataset

We can compute summaries of the dataset, performed by a desired variable.

Basic syntax:

```
new_dat ← summarise(dat,  
                     my_summary = do_smt_func(vars),  
                     ... )
```

summarise: summaries of the dataset

Let's compute the mean and standard deviation of the `gdpw2`, and of `barb2`:

```
PErisk_new ← summarise(PErisk,
                        mean_gdp = mean(gdpw2), std_gdp = sd(gdpw2),
                        mean_barb2 = mean(barb2), std_barb2 = sd(barb2))
head(PErisk_new, 2)
```

```
##   mean_gdp   std_gdp mean_barb2 std_barb2
## 1  9.041875  0.9702639  -2.925557  2.707211
```

Your turn: Compute the mean and standard deviation of the tips.

summarise and group_by: summaries of
the dataset by groups

summarise and group_by: summaries of

We can compute summaries of the dataset, performed by a desired variable.

Basic syntax:

```
new_dat ← dat %>%  
  group_by(var_to_group) %>%  
  summarise(my_summary = do_smt_func(vars),  
            ... )
```

summarise and group_by: summaries of

Let's compute the mean and standard deviation of the `gdpw2`, by expropriation risk:

```
PERisk_new <- PERisk %>% group_by(prsexp2) %>%  
  summarise(mean_gdp = mean(gdpw2), std_gdp = sd(gdpw2))  
  
## `summarise()` ungrouping output (override with `.groups` argument)  
  
head(PERisk_new, 2)  
  
## # A tibble: 2 x 3  
##   prsexp2 mean_gdp std_gdp  
##   <int>     <dbl>   <dbl>  
## 1      0      8.98    0.555  
## 2      1      8.48    1.04
```

summarise and group_by: summaries of

Your turn: Compute the mean and standard deviation of the tips, grouped by the smoker and non-smoker.

Questions?

Have a great weekend!
