



Introduction to Data Science: Operations

Héctor Corrada Bravo

University of Maryland, College Park, USA

2019-08-01

Operations that select attributes

`select`

In our data set we have a large number of attributes describing each arrest.

We only want to study patterns in these arrests based on a smaller number of attributes.

In that case we would like to create a data frame that contains only those attributes of interest.

We use the `select` function for this.

Operations that select attributes



Operations that select attributes

Let's create a data frame containing only the age, sex and district attributes

```
select(arrest_tab, age, sex, district)
```

```
## # A tibble: 104,528 x 3
```

```
##       age sex  district
```

```
##    <dbl> <chr> <chr>
```

```
##  1     23 M    <NA>
```

```
##  2     37 M    SOUTHERN
```

```
##  3     46 M    NORTHEASTERN
```

```
##  4     50 M    WESTERN
```

Operations that select attributes

We can use an operator to describe ranges. E.g., 1:5 would be attributes 1 through 5:

```
select(arrest_tab, 1:5)
```

```
## # A tibble: 104,528 x 5
```

```
##      arrest    age race  sex  arrestDate
```

```
##      <dbl> <dbl> <chr> <chr> <chr>
```

```
##  1 11126858    23  B    M    01/01/2011
```

```
##  2 11127013    37  B    M    01/01/2011
```

```
##  3 11126887    46  B    M    01/01/2011
```

```
##  4 11126873    50  B    M    01/01/2011
```

Operations that select entities



Operations that select entities

slice

We can choose specific entities by their row position. For instance, to choose entities in rows 1,3 and 10, we would use the following:

```
slice(arrest_tab, c(1, 3, 10))
```

```
## # A tibble: 3 x 15
```

```
##   arrest    age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
## 1 1.11e7    23 B     M    01/01/2011 00'00"    <NA>
```

```
## 2 1.11e7    46 B     M    01/01/2011 01'00"    2800 Mayfield...
```


Operations that select entities

As before, the first argument is the data frame to operate on.

The second argument is a *vector* of indices.

We used the `c` function (for concatenate) to create a vector of indices.

Operations that select entities

We can also use the range operator here:

```
slice(arrest_tab, 1:5)
```

```
## # A tibble: 5 x 15
```

```
##   arrest    age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
## 1 1.11e7    23 B    M    01/01/2011 00'00"    <NA>
```

```
## 2 1.11e7    37 B    M    01/01/2011 01'00"    2000 Wilkens ...
```

```
## 3 1.11e7    46 B    M    01/01/2011 01'00"    2800 Mayfield...
```

```
## 4 1.11e7    50 B    M    01/01/2011 04'00"    2100 Ashburto...
```

```
## 5 1.11e7    33 B    M    01/01/2011 05'00"    4000 Wilsby A...
```

Operations that select entities

To create general sequences of indices we would use the `seq` function. For example, to select entities in even positions we would use the following:

```
slice(arrest_tab, seq(2, nrow(arrest_tab), by=2))
```

```
## # A tibble: 52,264 x 15
```

```
##   arrest   age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
##  1 1.11e7   37 B    M    01/01/2011 01'00"    2000 Wilkens ...
```

```
##  2 1.11e7   50 B    M    01/01/2011 04'00"    2100 Ashburto...
```

```
##  3 1.11e7   41 B    M    01/01/2011 05'00"    2900 Spellman...
```

Operations that select entities

filter

We can also select entities based on attribute properties. For example, to select arrests where age is less than 18 years old, we would use the following:

```
filter(arrest_tab, age < 18)
```

```
## # A tibble: 463 x 15
```

```
##   arrest   age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
##  1 1.11e7   17  B    M    01/03/2011 15:00    <NA>
```

Operations that select entities

The second argument is an expression that evaluates to a logical value (TRUE or FALSE), if the expression evaluates to TRUE for a given entity (row) then that entity (row) is part of the resulting data frame.

Operations that select entities

Operators used frequently include:

`==`, `!=`: tests equality and inequality respectively (categorical, numerical, datetimes, etc.)

`<`, `>`, `<=`, `>=`: tests order relationships for ordered data types (not categorical)

`!`, `&`, `|`: not, and, or, logical operators

Operations that select entities

To select arrests with ages between 18 and 25 we can use

```
filter(arrest_tab, age >= 18 & age <= 25)
```

```
## # A tibble: 35,770 x 15
```

```
##   arrest    age race  sex  arrestDate arrestTime arrestLocation
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
## 1 1.11e7    23 B    M    01/01/2011 00:00      <NA>
## 2 1.11e7    20 W    M    01/01/2011 00:05      5200 Moravia ...
## 3 1.11e7    24 B    M    01/01/2011 00:07      2400 Gainsdbo...
## 4 1.11e7    25 B    M    01/01/2011 00:20      2800 Violet A...
## 5 1.11e7    24 B    M    01/01/2011 00:40      3900 Greenmou...
```

Operations that select entities

The filter function can take multiple logical expressions. In this case they are combined with &. So the above is equivalent to

```
filter(arrest_tab, age >= 18, age <= 25)
```

```
## # A tibble: 35,770 x 15
```

```
##   arrest    age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
##  1 1.11e7    23 B     M    01/01/2011 00:00      <NA>
```

```
##  2 1.11e7    20 W     M    01/01/2011 00:05      5200 Moravia ...
```

```
##  3 1.11e7    24 B     M    01/01/2011 00:07      2400 Gainsdbo...
```

```
##  4 1.11e7    25 B     M    01/01/2011 00:20      2800 Violet A...
```


Operations that select entities

`sample_n` and `sample_frac`

Frequently we will want to choose entities from a data frame at random. The `sample_n` function selects a specific number of entities at random:

```
sample_n(arrest_tab, 10)
```

```
## # A tibble: 10 x 15
```

```
##   arrest   age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
## 1 1.26e7   25 B    M    09/26/2012 22:25    0 N Howard St
```

```
## 2 1.14e7   22 B    F    11/10/2011 18:00    2700 Kinsey St
```

Operations that select entities

The `sample_frac` function selects a fraction of entities at random:

```
sample_frac(arrest_tab, .1)
```

```
## # A tibble: 10,453 x 15
```

```
##   arrest   age race  sex  arrestDate arrestTime arrestLocation
```

```
##   <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>
```

```
## 1 1.13e7   34 B    M    09/26/2011 19:30    <NA>
```

```
## 2 1.25e7   20 B    M    04/05/2012 04:30    1300 N Calhou...
```

```
## 3 1.11e7   26 B    M    02/04/2011 10:10    <NA>
```

```
## 4 1.25e7   20 B    M    09/05/2012 10:45    <NA>
```

```
## 5 1.26e7   32 B    M    11/08/2012 08:35    3800 Brehms Ln
```

Operations that select entities

Pipelines of operations

All of the functions implementing our first set of operations have the same argument/value structure.

They take a data frame as a first argument and return a data frame. We refer to this as the *data-->transform-->data* pattern.

This is the core a lot of what we will do in class as part of data analyses.

Specifically, we will combine operations into *pipelines* that manipulate data frames.

Operations that select entities

In R, the `dplyr` package introduces *syntactic sugar* to make this pattern explicit.

```
arrest_tab %>%  
  sample_frac(.1)
```

```
## # A tibble: 10,453 x 15
```

```
##      arrest   age race  sex  arrestDate arrestTime arrestLocation  
##      <dbl> <dbl> <chr> <chr> <chr>      <time>      <chr>  
##  1  1.24e7    59 W     M    01/14/2012 17:50    600 Monroe St  
##  2  1.12e7    44 B     M    05/01/2011 00:30    <NA>  
##  3  1.24e7    26 W     M    03/28/2012 02:15    <NA>
```

Operations that select entities

The `%>%` binary operator takes the value to its **left** and inserts it as the first argument of the function call to its **right**. So the expression `LHS %>% f(another_argument)` is **equivalent** to the expression `f(LHS, another_argument)`.

In pandas, you can chain `.` calls.

Operations that select entities

Using the `%>%` operator and the *data-->transform-->data* pattern of the functions we've seen so far, we can create pipelines.

Operations that select entities

For example, let's create a pipeline that:

1) filters our dataset to arrests between the ages of 18 and 25 2) selects attributes `sex`, `district` and `arrestDate` (renamed as `arrest_date`) 3) samples 50% of those arrests at random

We will assign the result to variable `analysis_tab`

Operations that select entities

```
analysis_tab <- arrest_tab %>%  
  filter(age >= 18, age <= 25) %>%  
  select(sex, district, arrest_date=arrestDate) %>%  
  sample_frac(.5)  
analysis_tab
```

```
## # A tibble: 17,885 x 3  
  
##   sex    district    arrest_date  
##   <chr> <chr>        <chr>  
  
## 1 M      EASTERN      12/14/2012  
## 2 F      <NA>          08/10/2011  
## 3 M      <NA>          03/09/2012
```


Operations that select entities

Exercise: Create a pipeline that:

1) filters dataset to arrests from the "SOUTHERN" district occurring before "12:00" (arrestTime) 2) selects attributes, sex, age 3) samples 10 entities at random

Principles: More Operations

In the previous section we introduced our first few operations to manipulate data frames. Next, we learn a few more: sorting, creating new attributes, summarizing and grouping. Finally we will take a short detour through a discussion on vectors.