Chapter 5

Data Transformation

Icaro Franco Hernandes

2022-09-26

Remember that to view a whole data set we can execute, for example, `view(nycflights13::flights)`. This is a tibble, and tibble $\neq$ table. Tibbles work better for the tidyverse. To check what kind of variable we are working with, we can use the following command:

```
typeof(nycflights13::flights$time_hour)
```

```
## [1] "double"
```

Filtering

Selecting all flights from January first:

```
nycflights13::flights%>%
  dplyr::filter(month==1,day==1)->jan1
#remeber that dplyr does not change the original dataset (always try to be as pure as possible).
```

If we want to also print the new data set, just put between parenthesis:

```
(nycflights13::flights%>%
  dplyr::filter(month==1,day==1)->jan1)
```

```
## # A tibble: 842 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013     1     1      517        515       2     830     819      11 UA
##  2  2013     1     1      533        529       4     850     830      20 UA
##  3  2013     1     1      542        540       2     923     850      33 AA
##  4  2013     1     1      544        545      -1    1004    1022     -18 B6
##  5  2013     1     1      554        600      -6     812     837     -25 DL
##  6  2013     1     1      554        558      -4     740     728      12 UA
##  7  2013     1     1      555        600      -5     913     854      19 B6
##  8  2013     1     1      557        600      -3     709     723     -14 EV
##  9  2013     1     1      557        600      -3     838     846      -8 B6
## 10  2013     1     1      558        600      -2     753     745       8 AA
## # ... with 832 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

Boolean operators are: `&` for **and**, `|` for **or** and `!` for **is not**.

Inclusion operator: `%in%`. For example:

```r
nycflights13::flights%>%
  dplyr::filter(month %in% c(11,12))
```

```
## # A tibble: 55,403 x 19
##     year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier
##    <int> <int> <int>    <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>
##  1  2013    11     1        5       2359       6     352     345       7 B6
##  2  2013    11     1       35       2250     105     123    2356      87 B6
##  3  2013    11     1      455        500      -5     641     651     -10 US
##  4  2013    11     1      539        545      -6     856     827      29 UA
##  5  2013    11     1      542        545      -3     831     855     -24 AA
##  6  2013    11     1      549        600     -11     912     923     -11 UA
##  7  2013    11     1      550        600     -10     705     659       6 US
##  8  2013    11     1      554        600      -6     659     701      -2 US
##  9  2013    11     1      554        600      -6     826     827      -1 DL
## 10  2013    11     1      554        600      -6     749     751      -2 DL
## # ... with 55,393 more rows, 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, and abbreviated variable names
## #   1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## #   5: arr_delay
```

This will filter all flights that happened in november **or** december. The filter already excludes `NA` values.

Exercises 5.2.4

1

Find all flights that

- Had an arrival delay of two or more hours

```r
nycflights13::flights%>%
  dplyr::filter(arr_delay>=120)->f1
```

- Flew to Houston (`IAH` or `HOU`)

```r
nycflights13::flights%>%
  dplyr::filter(dest %in% c("IAH", "HOU"))->f2
```

- Were operated by United, American, or Delta

```r
nycflights13::flights%>%
  dplyr::filter(carrier %in% c("UA", "AA", "DL"))->f3
```

- Departed in Summer (July, August, and September)

```r
nycflights13::flights%>%
  dplyr::filter(month %in% c(7, 8, 9))->fsummer
```

- Arrived more than two hours late, but did not leave late

```r
nycflights13::flights%>%
  dplyr::filter(arr_delay>120,dep_time<=sched_dep_time)->f5
```

- Were delayed by at least an hour, but made up over 30 minutes in flight

```r
nycflights13::flights%>%
  dplyr::filter(arr_delay>=60,air_time>30)->f6
```

- Departed between midnight and 6am (inclusive)

```
nycflights13::flights%>%
  dplyr::filter(hour %in% c(seq(0,6)))->f7

or
```

```
## function (e1, e2)  .Primitive("|")
```

```
nycflights13::flights%>%
  dplyr::filter(hour >= 0 & hour<= 6)->f71

# nycflights13::flights%>%
#   dplyr::filter(0 <= hour <= 6) -> this does not work!
```

2

Another useful `dplyr` filter helper is `between()`. What does it do? Can you use it to simplify the code needed to answer the precious questions?

According to the documentation, `between()` let us pick any values between to boundaries, and it is a shortcut for `x>= & x<=`. They would be useful in the cases where we had to filter for the summer months and the flights between midnight and 6 a.m.:

```
nycflights13::flights%>%
  dplyr::filter(between(month, 7, 9))->f8

nycflights13::flights%>%
  dplyr::filter(between(hour, 0, 6))->f9
```

3

How many flights have a missing `dep_time`? What other variables are missing? What might these rows represent?

```
nycflights13::flights%>%
  dplyr::filter(is.na(dep_time))%>%
  dplyr::summarise(n = dplyr::n())->na

na
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1  8255
```

Using the count operator from `dplyr` we can see that 8255 flights have missing values for the departure time. This means that theses flights were canceled. If we do not have the departure time, we also cannot check the airtime, the departure delay and the arrival delay,**Remember this count operator (within the summarise function) from dplyr**.

4

Why is `NA^0`not missing? Why is `NA|TRUE` not missing? Whys is `FALSE & NA` not missing? Can you figure out the general rule? (`NA*0` is a tricky counterexample!)

```
NA^0
```

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

```
NA|TRUE
```

```
## [1] TRUE
```

```
FALSE&NA
```

```
## [1] FALSE
```

Since we are working with boolean operators here, the general rule is that R avoids the NA values and does let them contaminate the operation. It is different from the case if we calculate the average of some values with an NA (in that case it does contaminate the average).

```
v1<-c(1,1, NA)
mean(v1)
```

```
## [1] NA
```

```
mean(v1, na.rm = T)
```

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

The command `na.rm=TRUE` discards the `NA` values from the calculation!.