Tarea - Ordenando y Seleccionando Datos con dplyr

Jesus Mudarra Luján

2022-10-31

Pregunta 1

Piensa cómo podrías usar la función `arrange()` para colocar todos los valores `NA` al inicio. Pista: puedes la función `is.na()` en lugar de la función `desc()` como argumento de `arrange`.

```
arrange(flights, desc(is.na(flights)))
```

```
## # A tibble: 336,776 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     2       NA       1545      NA      NA    1910      NA AA
##  2  2013     1     2       NA       1601      NA      NA    1735      NA UA
##  3  2013     1     3       NA        857      NA      NA    1209      NA UA
##  4  2013     1     3       NA        645      NA      NA     952      NA UA
##  5  2013     1     4       NA        845      NA      NA    1015      NA 9E
##  6  2013     1     4       NA       1830      NA      NA    2044      NA 9E
##  7  2013     1     5       NA        840      NA      NA    1001      NA 9E
##  8  2013     1     7       NA        820      NA      NA     958      NA 9E
##  9  2013     1     8       NA       1645      NA      NA    1838      NA US
## 10  2013     1     9       NA        755      NA      NA    1012      NA 9E
## # ... with 336,766 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
```

Pregunta 2

Ordena los vuelos de `flights` para encontrar los vuelos más retrasados en la salida. ¿Qué vuelos fueron los que salieron los primeros antes de lo previsto?

```
arrange(flights, desc(dep_delay))
```

```
## # A tibble: 336,776 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     9      641        900    1301    1242    1530    1272 HA
##  2  2013     6    15     1432       1935    1137    1607    2120    1127 MQ
##  3  2013     1    10     1121       1635    1126    1239    1810    1109 MQ
##  4  2013     9    20     1139       1845    1014    1457    2210    1007 AA
##  5  2013     7    22      845       1600    1005    1044    1815     989 MQ
##  6  2013     4    10     1100       1900     960    1342    2211     931 DL
##  7  2013     3    17     2321        810     911     135    1020     915 DL
##  8  2013     6    27      959       1900     899    1236    2226     850 DL
```

```
##  9  2013     7    22     2257        759     898     121    1026     895 DL
## 10  2013    12     5      756       1700     896    1058    2020     878 AA
## # ... with 336,766 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
```

```
head(arrange(flights, dep_delay))
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep~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    12     7     2040        2123     -43      40    2352      48 B6
## 2  2013     2     3     2022        2055     -33    2240    2338     -58 DL
## 3  2013    11    10     1408        1440     -32    1549    1559     -10 EV
## 4  2013     1    11     1900        1930     -30    2233    2243     -10 DL
## 5  2013     1    29     1703        1730     -27    1947    1957     -10 F9
## 6  2013     8     9      729         755     -26    1002     955       7 MQ
## # ... with 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
```

El vuelo N592JB que fue de `JFK` a `DEN` salió 43 minutos antes de lo previsto.

Pregunta 3

Ordena los vuelos de `flights` para encontrar los vuelos más rápidos. Usa el concepto de rapidez que consideres.

```
flights %>%
  arrange(air_time) %>%
  select(tailnum, origin, dest, air_time)
```

```
## # A tibble: 336,776 x 4
##    tailnum origin dest  air_time
##    <chr>   <chr>  <chr>    <dbl>
##  1 N16911  EWR    BDL         20
##  2 N12167  EWR    BDL         20
##  3 N27200  EWR    BDL         21
##  4 N13913  EWR    PHL         21
##  5 N13955  EWR    BDL         21
##  6 N12921  EWR    PHL         21
##  7 N947UW  LGA    BOS         21
##  8 N8501F  JFK    PHL         21
##  9 N12160  EWR    BDL         21
## 10 N16987  EWR    BDL         21
## # ... with 336,766 more rows
```

Pregunta 4

¿Qué vuelos tienen los trayectos más largos? Busca en Wikipedia qué dos aeropuertos del dataset alojan los vuelos más largos.

Honolulu (HNL) - New York (JFK): 8.020 kilómetros

Pregunta 5

¿Qué vuelos tienen los trayectos más cortos? Busca en Wikipedia qué dos aeropuertos del dataset alojan los vuelos más cortos.

Newark (EWR) - New York (LGA): 27 kilómetros

Pregunta 6

Dale al coco para pensar cuántas más maneras posibles de seleccionar los campos `dep_time`, `dep_delay`, `arr_time` y `arr_delay` del dataset de `flights`.

```
select(flights, dep_time, dep_delay, arr_time, arr_delay)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
select(flights, c(dep_time, dep_delay, arr_time, arr_delay))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```
select(flights, starts_with("dep"), starts_with("arr"))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
```

```
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

Pregunta 7

¿Qué ocurre si pones el nombre de una misma variable varias veces en un `select()`?

No ocurre nada, por muchas veces que introduzcas la misma variable en la función `select()` solo aparecerá una vez.

Pregunta 8

Investiga el uso de la función `one_of()` de `dplyr`

Utilizando la función `one_of()` nos mostrará las columnas de todas las variables que le indiquemos en la función `select()` independientemente de si existe o no.

Pregunta 9

Investiga cómo puede ser útil la función `one_of()` de la pregunta anterior en conjunción con el vector de variables

```
c("year", "month", "day", "dep_delay", "arr_delay")
```

Sin `one_of()` obtendremos un error en el `select()`:

```
select(flights, c("year", "month", "day", "dep_delay", "arr_delay", "variable_inexistente"))
```

Con `one_of()` nos permite añadir variables que no se encuentran en el data frame:

```
select(flights, one_of(c("year", "month", "day", "dep_delay", "arr_delay", "variable_inexistente")))
```

```
## Warning: Unknown columns: `variable_inexistente`
```

```
## # A tibble: 336,776 x 5
##     year month   day dep_delay arr_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1         2        11
##  2  2013     1     1         4        20
##  3  2013     1     1         2        33
##  4  2013     1     1        -1       -18
##  5  2013     1     1        -6       -25
##  6  2013     1     1        -4        12
##  7  2013     1     1        -5        19
##  8  2013     1     1        -3       -14
##  9  2013     1     1        -3        -8
## 10  2013     1     1        -2         8
## # ... with 336,766 more rows
```

Pregunta 10

Intenta averiguar el resultado del siguiente código. Luego, ejecútalo y a ver si el resultado te sorprende.
`select(flights, contains("time"))`

Intenta averiguar cómo lo hacen las funciones de ayuda de la función `select` para tratar el caso por defecto y cómo lo puedes cambiar.

Con el párametro `contains` mostrará todas las columnas de las variables que contengan la palabra “time”.