Programming for Data Science

Data Transformation with dplyr

Marco Beccuti

Università degli Studi di Torino

Dipartimento di Informatica

Data Transformation in R

- It is rare that you get the data in exactly the right form you need;
- You will need to create some new variables or summaries, to rename the variables and to reorder the observations;
- To deal with this we will use the *dplyr* package;
- The *dplyr* package can be loaded using:

  $>$ *library*(*tidyverse*)

dplyr Basics

- The five key *dplyr* functions are:
  - *filter()* to pick observations by their values;
  - *arrange()* to reorder the rows;
  - *select()* to pick variables by their names;
  - *mutate()* to create new variables with functions of existing variables;
  - *summary()* to collapse many values down to a single summary.
- All these functions can be used in conjunction with *group_by()*;
- It changes the scope of each function from operating on the entire dataset to operating on it group-by-group.

dplyr Basics

- All these function similarly:
  - The first argument is a data frame (i.e. a tibble);
  - The subsequent arguments describe what to do with the data frame, using the variable names (without quotes);
  - The result is a new data frame.

Filter Rows with filter()

- It allows you to subset observations based on their values;
- The first argument is the name of the data frame;
- The others arguments are the expressions that filter the data frame.
- From data frame *flights* (in package nycflights13) we can select all flights on January 1st with:

$> library(nycflights13)$

$> filter(flights, month == 1, day == 1)$

```
#> # A tibble: 842 × 19
#>    year month   day dep_time sched_dep_time dep_delay
#>   <int> <int> <int>    <int>          <int>     <dbl>
#> 1  2013     1     1      517            515         2
#> 2  2013     1     1      533            529         4
#> 3  2013     1     1      542            540         2
#> 4  2013     1     1      544            545        -1
#> 5  2013     1     1      554            600        -6
#> 6  2013     1     1      554            558        -4
#> # ... with 836 more rows, and 13 more variables:
#> #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
#> #   carrier <chr>, flight <int>, tailnum <chr>,origin <chr>,
#> #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> #   minute <dbl>, time_hour <dttm>
```

Filter Rows with filter()

- R either prints out the results, or saves them to a variable;
- If you want to do both, you can wrap the assignment in parentheses:

$> (dec25 = filter(flights, month == 12, day == 25))$

```
#> # A tibble: 719 × 19
#>    year month   day dep_time sched_dep_time dep_delay
#>   <int> <int> <int>    <int>          <int>     <dbl>
#> 1  2013    12    25      456            500        -4
#> 2  2013    12    25      524            515         9
#> 3  2013    12    25      542            540         2
#> 4  2013    12    25      546            550        -4
#> 5  2013    12    25      556            600        -4
#> 6  2013    12    25      557            600        -3
#> # ... with 713 more rows, and 13 more variables:
#> #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
#> #   carrier <chr>, flight <int>, tailnum <chr>,origin <chr>,
#> #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> #   minute <dbl>, time_hour <dttm>
```

Filter Rows with filter()

- To find all flights that departed in November or December:

  $> (novdec = filter(flights, month == 12 \mid month == 11))$

  $> (novdec = filter(flights, month \ \%in\% \ c(11, 12))$

- To find flights that were not delayed (on arrival or departure) by more than two hours:

  $> filter(flights, !(arr\_delay > 120 \mid dep\_delay > 120))$

  $> filter(flights, arr\_delay <= 120, dep\_delay <= 120)$

Arrange Rows with arrange()

- The function *arrange()* changes the rows ordering;
- It takes a data frame and a set of column names (or more complicated expressions) to order by;
- If you provide more than one column name, each additional column will be used to break ties in the values of preceding columns:

  > *arrange*(*flights*, *year*, *month*, *day*)

```
#> # A tibble: 336,776 × 19
#>    year month   day dep_time sched_dep_time dep_delay
#>   <int> <int> <int>    <int>          <int>     <dbl>
#> 1  2013     1     1      517            515         2
#> 2  2013     1     1      533            529         4
#> 3  2013     1     1      542            540         2
#> 4  2013     1     1      544            545        -1
#> 5  2013     1     1      554            600        -6
#> 6  2013     1     1      554            558        -4
#> # ... with 3.368e+05 more rows, and 13 more variables:
#> #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
#> #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#> #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#> #   minute <dbl>, time_hour <dttm>
```

Arrange Rows with arrange()

- Use *desc()* to reorder by a column in descending order:

  $> arrange(flights, desc(arr\_delay))$

- Missing values are always sorted at the end:

  $> df = tibble(x = c(5, 2, NA))$

  $> arrange(df, x)$

```
#> # A tibble: 3 × 1
#>       x
#>   <dbl>
#> 1     2
#> 2     5
#> 3    NA
```

  $> arrange(df, desc(x))$

```
#> # A tibble: 3 × 1
#>       x
#>   <dbl>
#> 1     5
#> 2     2
#> 3    NA
```

Select Columns with select()

- Function *select()* allows you to rapidly zoom in on a useful subset using operations based on the names of the variables;

  > *select*(*flights*, *year*, *month*, *day*)

```
#> # A tibble: 336,776 × 3
#>    year month   day
#>   <int> <int> <int>
#> 1  2013     1     1
#> 2  2013     1     1
#> 3  2013     1     1
#> 4  2013     1     1
#> 5  2013     1     1
#> 6  2013     1     1
#> # ... with 3.368e+05 more rows
```

Select Columns with select()

- Select all columns between *year* and *day* (inclusive):

  $>$ *select*(*flights*, *year* : *day*)

```
#> # A tibble: 336,776 × 3
#>    year month   day
#>   <int> <int> <int>
#> 1  2013     1     1
#> 2  2013     1     1
#> 3  2013     1     1
#> 4  2013     1     1
#> 5  2013     1     1
#> 6  2013     1     1
#> # ... with 3.368e+05 more rows
```

Select Columns with select()

- Select all columns except those from *year* to *day* (inclusive):

  $> select(flights, -(year : day))$

```
#> # A tibble: 336,776 × 16
#>   dep_time sched_dep_time dep_delay arr_time sched_arr_time
#>      <int>          <int>     <dbl>    <int>          <int>
#> 1      517            515         2      830            819
#> 2      533            529         4      850            830
#> 3      542            540         2      923            850
#> 4      544            545        -1     1004           1022
#> 5      554            600        -6      812            837
#> 6      554            558        -4      740            728
#> # ... with 3.368e+05 more rows, and 12 more variables:
#> #   arr_delay <dbl>, carrier <chr>, flight <int>,
#> #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#> #   distance <dbl>,  hour <dbl>, minute <dbl>,
#> #   time_hour <dttm>
```

Select Columns with select()

- There are a number of helper functions you can use within *select()*:
  - *starts_with("abc")* matches names that begin with "abc";
  - *ends_with("xyz")* matches names that end with "xyz";
  - *contains("ijk")* matches names that contain "ijk";
  - *matches(".a.")* selects variables that match a regular expression;
  - *num_range("x", 1:3)* matches x1, x2, and x3;
  - To use *select()* in conjunction with the *everything()*. This is useful to move variables to the start of the data frame.

```
> select(flights, time_hour, air_time, everything())
```

```
#> # A tibble: 336,776 × 19
#>             time_hour air_time  year month   day dep_time
#>                 <dttm>    <dbl> <int> <int> <int>    <int>
#> 1 2013-01-01 05:00:00      227  2013     1     1      517
#> 2 2013-01-01 05:00:00      227  2013     1     1      533
#> 3 2013-01-01 05:00:00      160  2013     1     1      542
#> 4 2013-01-01 05:00:00      183  2013     1     1      544
#> 5 2013-01-01 06:00:00      116  2013     1     1      554
```

Add New Variables with mutate()

- Function *mutate()* adds new columns that are functions of existing columns;
- it always adds new columns at the end of a dataset:

$> fl = select(flights, year : day, ends\_with("delay"), distance, air\_time)$

$> mutate(fl, gain = arr\_delay - dep\_delay, speed = distance/air\_time * 60)$

```
# A tibble: 336,776 × 9
    year month   day dep_delay arr_delay distance air_time  gain speed
   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl> <dbl>
 1  2013     1     1         2        11     1400      227     9  370.
 2  2013     1     1         4        20     1416      227    16  374.
 3  2013     1     1         2        33     1089      160    31  408.
 4  2013     1     1        -1       -18     1576      183   -17  517.
 5  2013     1     1        -6       -25      762      116   -19  394.
 6  2013     1     1        -4        12      719      150    16  288.
 7  2013     1     1        -5        19     1065      158    24  404.
 8  2013     1     1        -3       -14      229       53   -11  259.
 9  2013     1     1        -3        -8      944      140    -5  405.
10  2013     1     1        -2         8      733      138    10  319.
# … with 336,766 more rows
```

Add New Variables with mutate()

- If you only want to keep the new variables, use *transmute()*:

  $> transmute(fl, gain = arr\_delay - dep\_delay, speed = distance/air\_time * 60)$

```
# A tibble: 336,776 × 2
    gain speed
   <dbl> <dbl>
 1     9  370.
 2    16  374.
 3    31  408.
 4   -17  517.
 5   -19  394.
 6    16  288.
 7    24  404.
 8   -11  259.
 9    -5  405.
10    10  319.
# … with 336,766 more rows
```

Grouped Summaries with summarize()

- Function *summarize()* collapses a data frame to a single row:

  $> summarize(flights, delay = mean(dep\_delay, na.rm = TRUE))$

```
#> # A tibble: 1 × 1
#>   delay
#>   <dbl>
#> 1  12.6
```

- Function *summarize()* is not terribly useful unless we pair it with *group_by()*.
- This changes the unit of analysis from the complete dataset to individual groups:

  $> by\_day = group\_by(flights, year, month, day)$

  $> summarize(by\_day, delay = mean(dep\_delay, na.rm = TRUE))$

```
#> Source: local data frame [365 x 4]
#> Groups: year, month [?]
#>
#>    year month   day delay
#>   <int> <int> <int> <dbl>
#> 1  2013     1     1 11.55
#> 2  2013     1     2 13.86
#> 3  2013     1     3 10.99
#> 4  2013     1     4  8.95
#> 5  2013     1     5  5.73
#> 6  2013     1     6  7.15
#> # ... with 359 more rows
```