DSiBA: Series 3

Data Wrangling – 1

Table of contents

1 Introduction

In this series, we focus on data wrangling in R, covering key concepts from data manipulation. These exercises are designed to help you practice tasks such as widening and longening data, handling missing values, selecting and mutating columns, filtering data, and summarising information using the `tidyverse` package.

For this exercise, we will use the `flights` dataset from the `nycflights13` package, which contains information about flights departing from New York City in 2013. The dataset includes details such as flight times, delays, distances, and carrier information. We will explore various data wrangling techniques to manipulate and summarise this dataset effectively.

```
library(tidyverse)
library(nycflights13)
```

2 Counting the Data

2.1 Counting Flights per Tail Number, Carrier, and Month

Question

Count the number of flights per `tailnum`, `carrier`, and `month`. This will help you understand how often each aircraft is used by different carriers across different months. Name this dataset `flights_count`.

- Use `count()` to calculate the number of flights per `tailnum`, `carrier`, and `month`.

```
flights_count
```

```
# A tibble: 38,028 x 4
   tailnum carrier month     n
   <chr>   <chr>   <int> <int>
 1 D942DN  DL          2     1
 2 D942DN  DL          3     2
 3 D942DN  DL          7     1
 4 NOEGMQ  MQ          1    41
 5 NOEGMQ  MQ          2    28
 6 NOEGMQ  MQ          3    30
 7 NOEGMQ  MQ          4    29
 8 NOEGMQ  MQ          5    13
 9 NOEGMQ  MQ          6    32
10 NOEGMQ  MQ          7    11
# i 38,018 more rows
```

3 Widening and Longening the Data

3.1 Widening the Data

Question

Create a wide version of the `flights_count` dataset named `wide_flights`, where each `carrier` has a separate column for the number of flights per month. Use `tailnum` and `month` as identifiers for each aircraft.

- Use `pivot_wider()` to create a wide table with separate columns for each `carrier`.

```
wide_flights
```

```
# A tibble: 37,988 x 18
   tailnum month    DL    MQ    EV    US    UA  `9E`    FL    B6    AA    WN
   <chr>   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
 1 D942DN      2     1     0     0     0     0     0     0     0     0     0
 2 D942DN      3     2     0     0     0     0     0     0     0     0     0
 3 D942DN      7     1     0     0     0     0     0     0     0     0     0
 4 NOEGMQ      1     0    41     0     0     0     0     0     0     0     0
 5 NOEGMQ      2     0    28     0     0     0     0     0     0     0     0
 6 NOEGMQ      3     0    30     0     0     0     0     0     0     0     0
 7 NOEGMQ      4     0    29     0     0     0     0     0     0     0     0
 8 NOEGMQ      5     0    13     0     0     0     0     0     0     0     0
 9 NOEGMQ      6     0    32     0     0     0     0     0     0     0     0
10 NOEGMQ      7     0    11     0     0     0     0     0     0     0     0
# i 37,978 more rows
# i 6 more variables: F9 <int>, AS <int>, VX <int>, HA <int>, OO <int>,
#   YV <int>
```

3.2 Longening the Data

Question

Create a long version of the `wide_flights` dataset named `long_flights`, where each observation represents a `carrier` and its corresponding number of flights per month for each aircraft (`tailnum`).

- Use `pivot_longer()` to convert the wide table back to a long format.
- Ensure that the final dataset has columns for `tailnum`, `month`, `carrier`, and `n_flights`.

```
long_flights
```

```
# A tibble: 607,808 x 4
   tailnum month carrier n_flights
   <chr>   <int> <chr>       <int>
 1 D942DN      2 DL              1
 2 D942DN      2 MQ              0
 3 D942DN      2 EV              0
 4 D942DN      2 US              0
 5 D942DN      2 UA              0
 6 D942DN      2 9E              0
 7 D942DN      2 FL              0
 8 D942DN      2 B6              0
 9 D942DN      2 AA              0
10 D942DN      2 WN              0
# i 607,798 more rows
```

4 Handling Missing Values

4.1 Removing Rows with Missing Arrival Delay

Question

Remove rows where the `arr_delay` (arrival delay) is missing, as these might indicate cancelled flights. After removing the rows, calculate how many rows were removed from the dataset and explain why removing these rows is a reasonable approach.

- Use `filter()` to exclude rows with missing `arr_delay`.
- Use `nrow()` to calculate the number of rows before and after filtering.

```
num_removed
```

```
[1] 9430
```

5 Selecting and Renaming Columns

5.1 Selecting Columns Starting with "arr"

Question

Use `select()` to create a new table containing only columns that start with "arr" (e.g., `arr_delay`, `arr_time`). These columns provide information related to the arrival of flights.

- Use `select()` with `starts_with("arr")` to select the columns.

```
arr_columns
```

```
# A tibble: 336,776 x 2
   arr_time arr_delay
      <int>     <dbl>
 1      830        11
 2      850        20
 3      923        33
 4     1004       -18
 5      812       -25
 6      740        12
 7      913        19
 8      709       -14
 9      838        -8
10      753         8
# i 336,766 more rows
```

5.2 Renaming `tailnum` to `aircraft_id`

Question

Rename the `tailnum` column to `aircraft_id` to make it more descriptive, and relocate it to the first column in the dataset.

- Use `rename()` to change the column name.
- Use `relocate()` to move the column to the first position.

```
flights_renamed
```

```
# A tibble: 336,776 x 19
   aircraft_id  year month   day dep_time sched_dep_time dep_delay arr_time
   <chr>       <int> <int> <int>    <int>          <int>     <dbl>    <int>
 1 N14228       2013     1     1      517            515         2      830
 2 N24211       2013     1     1      533            529         4      850
 3 N619AA       2013     1     1      542            540         2      923
 4 N804JB       2013     1     1      544            545        -1     1004
 5 N668DN       2013     1     1      554            600        -6      812
 6 N39463       2013     1     1      554            558        -4      740
 7 N516JB       2013     1     1      555            600        -5      913
 8 N829AS       2013     1     1      557            600        -3      709
 9 N593JB       2013     1     1      557            600        -3      838
10 N3ALAA       2013     1     1      558            600        -2      753
# i 336,766 more rows
# i 11 more variables: sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
#   flight <int>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

6 Mutating Columns

6.1 Adding Average Speed Column

Question

Add a new column called `avg_speed` that calculates the average speed (in miles per hour) for each flight using the `distance` and `air_time` columns. Relocate `avg_speed` after the `arr_time` column in the dataset. Note that `air_time` is in minutes, so you will need to convert it to hours for the calculation.

- Use `mutate()` to create the new column.
- Use `.after = arr_time` argument in `mutate()` to place the new column after `arr_time`.

```
flights_speed
```

```
# A tibble: 336,776 x 20
    year month   day dep_time sched_dep_time dep_delay arr_time avg_speed
   <int> <int> <int>    <int>          <int>     <dbl>    <int>     <dbl>
 1  2013     1     1      517            515         2      830      370.
 2  2013     1     1      533            529         4      850      374.
 3  2013     1     1      542            540         2      923      408.
 4  2013     1     1      544            545        -1     1004      517.
 5  2013     1     1      554            600        -6      812      394.
 6  2013     1     1      554            558        -4      740      288.
 7  2013     1     1      555            600        -5      913      404.
 8  2013     1     1      557            600        -3      709      259.
 9  2013     1     1      557            600        -3      838      405.
10  2013     1     1      558            600        -2      753      319.
# i 336,766 more rows
# i 12 more variables: 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>
```

7 Filtering and Excluding Records

7.1 Excluding Certain Flights

Question

Exclude flights with a `distance` of less than 500 miles or an `air_time` of less than 60 minutes. Calculate how many flights remain in the dataset after applying this filter.

- Use `filter()` to apply the conditions.
- Use `nrow()` to count the remaining flights.

```
remaining_flights
```

```
[1] 250677
```

8 Summarising Data

8.1 Creating a Summary Table

Question

Create a summary table that shows the **average arrival delay** and **average departure delay** for each combination of `origin` and `carrier`. This will help you understand the delays for different carriers at different origins.

- Use `group_by()` and `summarise()` to calculate the mean delays.
- Ignore missing values using `na.rm = TRUE` in the `mean()` function.

```
summary_table
```

```
# A tibble: 35 x 4
# Groups:   origin [3]
   origin carrier mean_arr_delay mean_dep_delay
   <chr>  <chr>            <dbl>          <dbl>
 1 EWR    9E               1.62            5.95
 2 EWR    AA               0.978          10.0
 3 EWR    AS              -9.93            5.80
 4 EWR    B6               9.39           13.1
 5 EWR    DL               8.78           12.1
 6 EWR    EV              17.0            20.2
 7 EWR    MQ              16.3            17.5
 8 EWR    OO              21.5            20.8
 9 EWR    UA               3.48           12.5
10 EWR    US               0.977           3.74
# i 25 more rows
```

9 Conclusion

In this exercise, we've practiced various data wrangling techniques using the `tidyverse` package in R. These included counting and reshaping data, handling missing values, selecting and mutating columns, filtering records, and summarising data. Mastery of these techniques is essential for effective data analysis and interpretation.