Supervised Methods in Machine Learning report guideline: Exploration and Application of R for Data Science

Eduard Romero

2023-02-25

Basic coding in R.

Please create a script in R to print the prime numbers from 1 to 100.

- Solution :

```
for (i in 1:100) {
  # Check if i is a prime number
  is_prime <- TRUE
  for (j in 2:(i-1)) {
    if (i %% j == 0) {
      is_prime <- FALSE
      break
    }
  }
  # Print the prime number
  if (is_prime && i > 1) {
    print(i)
  }
}
```

Package installation and basic use of tidyverse.

From the website: https://r4ds.had.co.nz/transform.html, solve and discuss in detail the following exercises:

- 5.2.4 Exercises: items 1, and 2
- 5.3.1 Exercises: all items
- 5.4.1 Exercises: items 2, 3, and 4
- 5.5.2 Exercises: items 1, and 2
- 5.6.7 Exercises: item 1
- 5.7.1 Exercises: item 2
- Solution 5.2.4 item 1: Find all flights that Had an arrival delay of two or more hours.

```
#Import data frame
library(nycflights13)
#Import library for commands
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.4     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
#Import library for commands
library(dplyr)
#Call a tibble
flights
# Delay is a local varibale for use the command filter
Delay <- filter(flights, arr_delay >= 120 )
Delay
```

- Solution 5.2.4 item 2: Find all flights that flew to Houston (IAH or HOU)

```
# fly_houston is local variable for use the command filter
fly_houston <- filter(flights, dest == "IAH" | dest == "HOU")
fly_houston
```

- Solution 5.3.1 item 1:How could you use arrange() to sort all missing values to the start?

```
# The operator pipeline %>$% is useful for concatenating multiple dplyr operations
flights %>%
  # the arrange() function put na values last
  #  command desc sorted in descending order
    arrange(desc(is.na(dep_time)),
           desc(is.na(dep_delay)),
           desc(is.na(arr_time)),
           desc(is.na(arr_delay)),
           desc(is.na(tailnum)),
           desc(is.na(air_time)))
```

- Solution 5.3.1 item 2: Sort flights to find the most delayed flights. Find the flights that left earliest.

```
# when we use the arrange and desc functions we are making our table sort in descending order
arrange(flights, desc(dep_delay))
# when we use the arrange function only we are making our table sort in ascending order
arrange(flights,(dep_delay))
```

- Solution 5.3.1 item 3: Sort flights to find the fastest (highest speed) flights.

```
# When we use the arrange function only we are making our table sort in ascending order
arrange(flights, (air_time))
```

- Solution 5.3.1 Item 4:Which flights traveled the farthest? Which traveled the shortest?

```
# when we use the arrange and desc functions we are making our table sort in descending order
arrange (flights, desc(distance))
# When we use the arrange function only we are making our table sort in ascending order
arrange(flights, distance)
```

> [!NOTE]
>
> The function arrage is for order the vector ascending, Function arrage, desc is for order the vector descending

- Solution 5.4.1 item 2:What happens if you include the name of a variable multiple times in a select() call?

  [!NOTE]

  The function select() call ignores the duplication, Any duplicated variables are only included once

```
# when we use the select() function it will only show the variables mentioned within the table
select(flights, year, month,day,day,day)
```

- Solution 5.4.1 item 3:What does the any_of() function do? Why might it be helpful in conjunction with this vector?

  [!NOTE]

  the function any_of() select variables contained in a character vector. They are especially useful for programming with selecting functions

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
select(flights, any_of(vars))
```

- Solution 5.4.1 item 4:Does the result of running the following code surprise you? How do the select helpers deal with case by default? How can you change that default?

  [!NOTE]

  when we use this expression it will only show the variables what contains values of time, he default helper functions are insensitive to case. This can be changes by setting

```
select(flights, contains("TIME", ignore.case = FALSE))
```

- Solution 5.5.2 item 1:Currently dep_time and sched_dep_time are convenient to look at, but hard to compute with because they're not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

```
# Method with division integer and modulus
mutate(flights,
       dep_time = (dep_time %/% 100) * 60 + (dep_time %% 100),
       sched_dep_time = (sched_dep_time %/% 100) * 60 + (sched_dep_time %% 100))
```

- Solution 5.2.2 item 2:Compare air_time with arr_time - dep_time. What do you expect to see? What do you see? What do you need to do to fix it?

```
flight_exercise <- select(flights, dep_time, arr_time, air_time)
mutate(flight_exercise,
       arr_min = (arr_time %/% 100)*60 + (arr_time %% 100),
       dep_min = (dep_time %/% 100)*60 + (dep_time %% 100),
       fly_short = arr_min - dep_min)
```

- Solution 5.6.7 item 1: Brainstorm at least 5 different ways to assess the typical delay characteristics of a group of flights. Consider the following scenarios
- A flight is 15 minutes early 50% of the time, and 15 minutes late 50% of the time.
- A flight is always 10 minutes late.
- A flight is 30 minutes early 50% of the time, and 30 minutes late 50% of the time.
- 99% of the time a flight is on time. 1% of the time it's 2 hours late.
- Solution 5.7.1 item 1:Refer back to the lists of useful mutate and filtering functions. Describe how each operation changes when you combine it with grouping.

It works with groups. For example, you can use mutation to create new variables using only group statistics, or you can filter all groups or filter as usual.

Reporting with Rmarkdown

Discuss in detail 2 of the functions for data transformation explained in https://r4ds.had.co.nz/transform.html. Discussion should be original (i.e. is not textually taken from any external source). It is expected that you at least:

- Use one chunk per explanation
- Change some of the chunk options and explain how the options affect the report.
  - Solution 1:

  [!NOTE]

  In this case we have created a dataframe with 4x4, using the mutate function we add another column where the professional performance of each minipak company worker will be evaluated.

```
#Load the library
library(dplyr)
# Creating a dataframe manual

Minipak <- data.frame(
  Employees_Name = c("Eduard Romero", "John Rodriguez", "Sergio Mendoza", "Andres Lote"),
    English_Level =c(6, 7, 5, 4),
    Jobs_Trips = c(1, 5, 2 ,1),
    Lvl_Engeenering = c(8, 10, 10, 10)
)
#Mutate Function
mutate(Minipak,AverageProfessional = English_Level+ Lvl_Engeenering*Jobs_Trips)
```

```
##   Employees_Name English_Level Jobs_Trips Lvl_Engeenering AverageProfessional
## 1  Eduard Romero             6          1               8                  14
## 2 John Rodriguez             7          5              10                  57
## 3 Sergio Mendoza             5          2              10                  25
## 4    Andres Lote             4          1              10                  14
```

- Solution 2:

```
#Load the library
library(dplyr)
# Creating a dataframe manual
UECCi <- data.frame(
  NameStudent =c("Juan","Eduard","Nicol","Daniela","Sergio", "David","Duvar", "Javier"),
  CareerName =c("Ing Mecatronica","Ing Electronica","Ing Mecanica","Ing Industrial","Ing Mecatronica",
   Score = c("3.5","3.0","4.0","4.5","3.8","3.1","3.9","4.0")
)
arrange(UECCi,Score  )
```

```
##   NameStudent      CareerName Score
## 1      Eduard Ing Electronica   3.0
## 2       David Ing Electronica   3.1
## 3        Juan Ing Mecatronica   3.5
## 4      Sergio Ing Mecatronica   3.8
## 5       Duvar      Lic Fisica   3.9
## 6       Nicol    Ing Mecanica   4.0
```

```
## 7      Javier    Ing Mecanico   4.0
## 8     Daniela  Ing Industrial   4.5
```

[!NOTE]

In this case we have created a dataframe, The arrange function allows us to organize our data by receiving the desired variable as a parameter and in ascending order.

Chunks Options:

In this document we use two options of configurations, the first is {r results='Hide'} This configuration allows us to evaluate the code but does not show its output

The Second is {r results='hold'} the results="hold" option means to "hold all the output pieces and push them to the end of a chunk"