Exploration and Application of R for Data Science

Andres Felipe Urrego Estrada - Juan Esteban Rincon - Aristizabal Brayan Alexander Suesca Bolivar - Solivan Santiago Ruiz - Edison Santiago Triana

2023-02-25

##Prime Numbers

#this line will be executed first  
  
print("Numeros Primos")

## [1] "Numeros Primos"

for(i in 2:100){  
 cont <- 0  
 for (k in 1:i) {  
 if(i %% k == 0){  
 cont=cont+1  
   
 }  
 }  
 if(cont<=2){  
 print(i)  
 }  
}

## [1] 2  
## [1] 3  
## [1] 5  
## [1] 7  
## [1] 11  
## [1] 13  
## [1] 17  
## [1] 19  
## [1] 23  
## [1] 29  
## [1] 31  
## [1] 37  
## [1] 41  
## [1] 43  
## [1] 47  
## [1] 53  
## [1] 59  
## [1] 61  
## [1] 67  
## [1] 71  
## [1] 73  
## [1] 79  
## [1] 83  
## [1] 89  
## [1] 97

In the first line of code the range of numbers from 2 to 100 is traversed by means of a for loop, in the next line a variable is declared which is initialized to 0, in the third line of code with the help of another for loop and a new variable “K” which is in the range 1 to i, in the next line of code with the help of a conditional if the module between the operation of the variables i/k is taken, if the module is equal to 0, the variable cont will add +1, once this cycle is closed, with the help of a second conditional it is verified that the variable cont is not greater than 2, if it fulfills this condition the number is printed on the screen.

## First the library is loaded

library(nycflights13)  
library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.4.0 ✔ purrr 1.0.1  
## ✔ tibble 3.1.8 ✔ dplyr 1.1.0  
## ✔ tidyr 1.3.0 ✔ stringr 1.5.0  
## ✔ readr 2.1.3 ✔ forcats 1.0.0  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(ggplot2)

These libraries are loaded into RStudio so that the functions and objects they contain can be used. For example, take the nycflights13 library, which contains a set of New York City flight data that can be used for analysis and visualization. By loading this library, you can access this data and use it in your code. The Tidyverse library is a collection of data analysis packages, including ggplot2 (for visualization), dplyr (for data processing), tidyr (for cleaning and transforming data) and others. By loading this library, the functions of these packages can be used more easily and the syntax between them is consistent. Finally, the ggplot2 library is one of the most widely used libraries for creating statistical plots in R. By loading this library, you can use the functions you need to create attractive and customized visualizations in R.

## We assign the dataset

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights

The dataframe is assigned to a new variable with the assignment operator “<-” the dataframe will be stored under the name “Bd”.

## First point 5.2.4

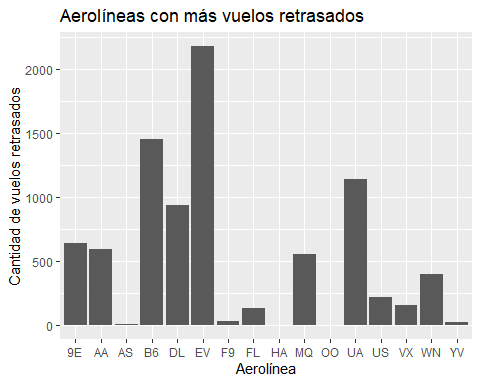
#Ítem 1 Flights that had an arrival delay of two or more hours.

#'Se crea una nueva funcion  
#'con el llamado a la función se extrae todos los vuelos con 2 horas o mas de retrasos  
#'@return Vuelos con 2 o mas horas de retraso  
#'@examples R2 <- RetrasoMas2Horas()  
RetrasoMas2Horas <- function(){  
 RetrasoM2Horas <- Bd %>%  
 filter(arr\_delay>=120, dep\_delay>=120)  
}

In this code, the nycflights13 library and dplyr are loaded and linked using the %>% function. In the second line, use the filter() function to filter out flights with a delay of two hours or more and store the result in the “RetrasoM2Horas” variable.

The following is a list of the new Dataset

R2 <- RetrasoMas2Horas()  
 ggplot(data = R2, aes(x = carrier)) +  
 geom\_bar() +  
 ggtitle("Aerolíneas con más vuelos retrasados") +  
 xlab("Aerolínea") +  
 ylab("Cantidad de vuelos retrasados")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of airline flight delays. The title is added with ggtitle() and the x and y axis labels are added with xlab() and ylab() respectively.

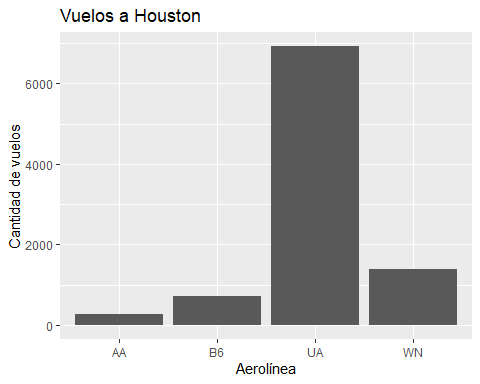
#ítem 2 Fly to Houston ( IAH o HOU)

#'Se realiza una nueva función para conocer los vuelos que salierón a Houston  
#'Con el llamado a la funcion VuelosHouston se extrae todos los vuelos que salieron a Houston  
#'@examples Vh <- VuelosHouston()  
#'@return retorna los vuelos a Houston  
VuelosHouston <- function(){  
 VueloHouston <- Bd %>%  
 filter(dest=="IAH"| dest=="HOU")  
}

In this code, the “Bd” dataframe is used and linked to “VueloHouston” using the %>% function. In the second line, use the filter() function to filter the flights that were made to houston and store the result in the variable “VueloHouston.”

The following is a list of the new Dataset

Vh <- VuelosHouston()  
ggplot(data = Vh, aes(x = carrier)) +  
geom\_bar() +  
ggtitle("Vuelos a Houston") +  
xlab("Aerolínea") +  
ylab("Cantidad de vuelos")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of flights to Houston by airlines. The title is added with ggtitle() and the x and y axis labels are added with xlab() and ylab() respectively.

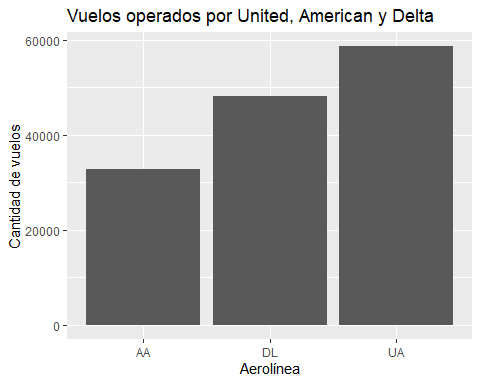
#ítem3 Flights that were operated by United, American or Delta

#'Se realiza una nueva función para conocer los vuelos que fueron operados por United, American y Delta  
#'Con el llamado a la funcion VuelosHouston se extrae todos los vuelos que fueron operados por United, American y Delta  
#'@examples Op <- VoperadosUUAUDL()  
#'@return retorna los vuelos que fueron operados por United, American y Delta  
VoperadosUUAUDL <- function(){  
 OperadosAAUADL <- Bd %>%  
 filter(carrier=="AA" | carrier=="UA" | carrier=="DL")  
}

In this code, the dataframe Bd is used and linked to OperadosAAUADL using the function %>%. In the second line, use the filter() function to filter the flights operated by United, American and Delta airlines and store the result in the variable OperadosAAUADL.”

The following is a list of the new Dataset

Op <- VoperadosUUAUDL()  
ggplot(data = Op, aes(x = carrier)) +  
geom\_bar() +  
ggtitle("Vuelos operados por United, American y Delta") +  
xlab("Aerolínea") +  
ylab("Cantidad de vuelos")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of flights operated by United, American and Delta airlines. The title is added with ggtitle() and the x- and y-axis labels are added with xlab() and ylab() respectively.

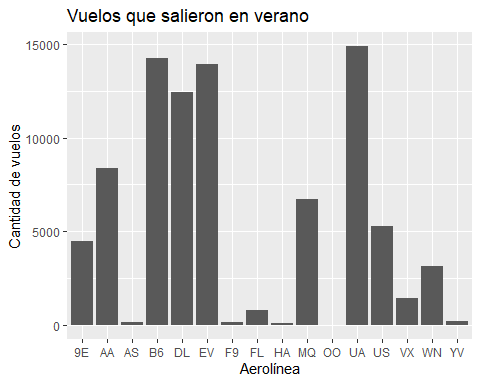
#Ítem 4 Flights departing in summer (July, August and September)

#'Se realiza una nueva función para conocer los vuelos que salierón en verano  
#'Con el llamado a la funcion VuelosHouston se extrae todos los vuelos que salierón en verano  
#'@examples Sv <- SalidaVerano()  
#'@return retorna los vuelos que salierón en verano  
SalidaVerano <- function(){  
 SalidaVeranon <- Bd%>%  
 filter(month==7 | month==8 | month==9)  
}

In this code, the Bd dataframe is used and linked to the SalidaVerano function %>%. In the second line, use the filter() function to filter the flights that departed in summer and store the result in the variable SalidaVerano.

The following is a list of the new Dataset

Sv <- SalidaVerano()  
ggplot(data = Sv, aes(x = carrier)) +  
geom\_bar() +  
ggtitle("Vuelos que salieron en verano") +  
xlab("Aerolínea") +  
ylab("Cantidad de vuelos")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of flights that departed in summer The title is added with ggtitle() and the x and y axis labels are added with xlab() and ylab() respectively.

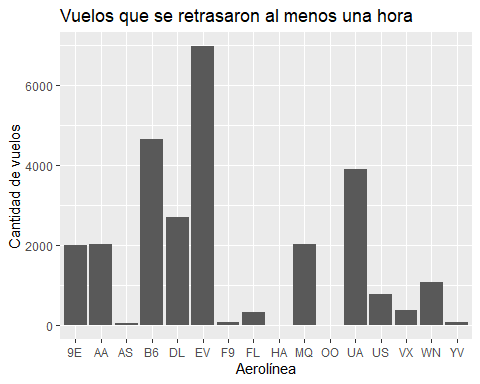
#ítem 5 Flights that arrived more than two hours late, but did not leave late

#'Se realiza una nueva función para conocer los vuelos que se retrasaron al menos 1 hora  
#'Con el llamado a la funcion VuelosHouston se extrae todos los vuelos que se retrasaron al menos 1 hora  
#'@examples R1h <- RetrasoAlM1hora()  
#'@return retorna los vuelos que se retrasaron al menos 1 hora  
RetrasoAlM1hora <- function(){  
 RetrasaronPr <- Bd %>%  
 filter(dep\_delay>=60)  
}

In this code, the dataframe “Bd” is used and linked to “LlegoTarde” with the function %>%. In the second line, use the filter() function to filter out flights that arrived late but left on time and store the result in the variable “LlegoTarde.”

The following is a list of the new Dataset

R1h <- RetrasoAlM1hora()  
ggplot(data = R1h, aes(x = carrier)) +  
geom\_bar() +  
ggtitle("Vuelos que se retrasaron al menos una hora") +  
xlab("Aerolínea") +  
ylab("Cantidad de vuelos")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of flights that arrived late but left on time.The title is added with ggtitle() and the x and y axis labels are added with xlab() and ylab() respectively.

#Ítem 6

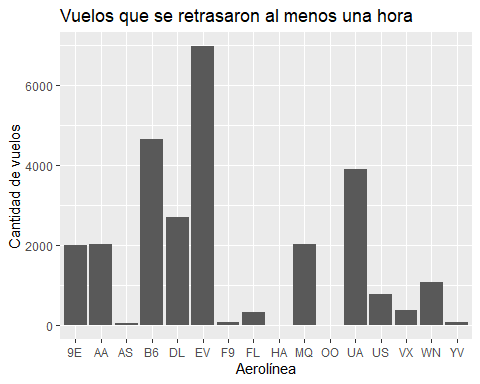
Flights that were delayed at least one hour, but made up more than 30 minutes in flight

#'Se realiza una nueva función para conocer los vuelos que se retrasaron al menos 1 hora  
#'Con el llamado a la funcion VuelosHouston se extrae todos los vuelos que se retrasaron al menos 1 hora  
#'@examples R1h <- RetrasoAlM1hora()  
#'@return retorna los vuelos que se retrasaron al menos 1 hora  
RetrasoAlM1hora <- function(){  
 RetrasaronPr <- Bd %>%  
 filter(dep\_delay>=60)  
}

In this code, the dataframe Bd is used and linked to DelayedPr with the function %>%. In the second line, use the filter() function to filter out flights that arrived late but left on time and store the result in the DelayedPr variable.

The following is a list of the new Dataset

R1h <- RetrasoAlM1hora()  
ggplot(data = R1h, aes(x = carrier)) +  
geom\_bar() +  
ggtitle("Vuelos que se retrasaron al menos una hora") +  
xlab("Aerolínea") +  
ylab("Cantidad de vuelos")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of flights that arrived late but left on time.The title is added with ggtitle() and the x and y axis labels are added with xlab() and ylab() respectively.

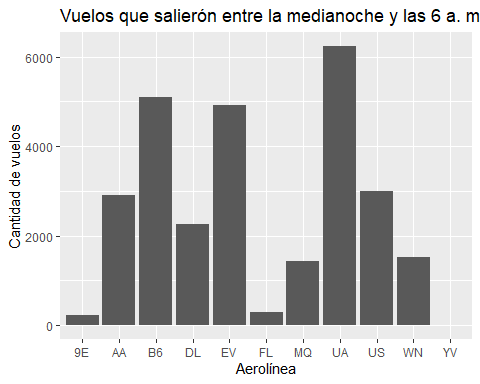
#ítem 7 Flights departing between midnight and 6 a.m. (included)

#'Se realiza una nueva función para conocer los vuelos que salieron entre la media noche hasta las 6 de la mañana  
#'Con el llamado a la funcion VuelosHouston se extrae todos los vuelos que salieron entre la media noche hasta las 6 de la mañana  
#'@examples Ve12To6 <- Vuelos12To6()  
#'@return retorna los vuelos que salieron entre la media noche hasta las 6 de la mañana  
Vuelos12To6 <- function(){  
 Salida12amto6am <- Bd %>%  
 filter(hour>=0 & hour<=6)  
}

In this code, the dataframe Bd is used and linked to RetDeparture12amto6pm with the function %>%. In the second line, use the filter() function to filter the flights that departed between midnight and 6am and store the result in the variable Depart12amto6pm.

The following is a list of the new Dataset

Ve12To6 <- Vuelos12To6()  
ggplot(data = Ve12To6, aes(x = carrier)) +  
geom\_bar() +  
ggtitle("Vuelos que salierón entre la medianoche y las 6 a. m.") +  
xlab("Aerolínea") +  
ylab("Cantidad de vuelos")

 Then use the ggplot2 function ggplot() and the geom\_bar() layer to create a bar chart showing the number of flights that arrived late but left on time.The title is added with ggtitle() and the x and y axis labels are added with xlab() and ylab() respectively.

##Second Point 5.2.4 #ítem 1 Another useful dplyr filtering helper is between(). What does it do? Can you use it to simplify the code needed to answer the above challenges?

#Example 1 We will select flights that departed between 6 and 8 am:

Between1 <- Bd %>%   
 filter(between(hour, 6, 8))

The new Dataset is as follows

Between1

## # A tibble: 76,014 × 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 1 554 600 -6 812 837 -25 DL   
## 2 2013 1 1 555 600 -5 913 854 19 B6   
## 3 2013 1 1 557 600 -3 709 723 -14 EV   
## 4 2013 1 1 557 600 -3 838 846 -8 B6   
## 5 2013 1 1 558 600 -2 753 745 8 AA   
## 6 2013 1 1 558 600 -2 849 851 -2 B6   
## 7 2013 1 1 558 600 -2 853 856 -3 B6   
## 8 2013 1 1 558 600 -2 924 917 7 UA   
## 9 2013 1 1 558 600 -2 923 937 -14 UA   
## 10 2013 1 1 559 600 -1 941 910 31 AA   
## # … with 76,004 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

#Example 2 We will select flights departing in May or June:

Between2 <- Bd%>%   
 filter(between(month, 5, 6))

The new Dataset is as follows

Between2

## # A tibble: 57,039 × 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 5 1 9 1655 434 308 2020 408 VX   
## 2 2013 5 1 451 500 -9 641 640 1 US   
## 3 2013 5 1 537 540 -3 836 840 -4 AA   
## 4 2013 5 1 544 545 -1 818 827 -9 UA   
## 5 2013 5 1 548 600 -12 831 854 -23 B6   
## 6 2013 5 1 549 600 -11 804 810 -6 MQ   
## 7 2013 5 1 553 600 -7 700 712 -12 EV   
## 8 2013 5 1 553 600 -7 655 701 -6 US   
## 9 2013 5 1 554 600 -6 731 756 -25 DL   
## 10 2013 5 1 554 600 -6 707 725 -18 UA   
## # … with 57,029 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

#Example 3 We will select flights that arrived on time (i.e. no delays):

Between3 <- Bd %>%   
 filter(between(arr\_delay, -1, 0))

The new Dataset is as follows

Between3

## # A tibble: 10,855 × 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 1 627 630 -3 1018 1018 0 US   
## 2 2013 1 1 732 735 -3 857 858 -1 B6   
## 3 2013 1 1 758 800 -2 1053 1054 -1 B6   
## 4 2013 1 1 807 810 -3 1043 1043 0 DL   
## 5 2013 1 1 931 930 1 1237 1238 -1 B6   
## 6 2013 1 1 956 1000 -4 1241 1241 0 DL   
## 7 2013 1 1 1042 1040 2 1325 1326 -1 B6   
## 8 2013 1 1 1124 1125 -1 1445 1445 0 DL   
## 9 2013 1 1 1219 1220 -1 1415 1415 0 AA   
## 10 2013 1 1 1240 1235 5 1415 1415 0 MQ   
## # … with 10,845 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

#Example 4 We will select flights that arrived on time (i.e. no delays):

Between4 <- Bd %>%   
 filter(between(dep\_delay, 1, 30) & between(arr\_delay, 31, Inf))

The new Dataset is as follows

Between4

## # A tibble: 8,032 × 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 1 542 540 2 923 850 33 AA   
## 2 2013 1 1 608 600 8 807 735 32 MQ   
## 3 2013 1 1 702 700 2 1058 1014 44 B6   
## 4 2013 1 1 933 904 29 1252 1210 42 B6   
## 5 2013 1 1 1003 959 4 1408 1329 39 US   
## 6 2013 1 1 1123 1110 13 1410 1336 34 UA   
## 7 2013 1 1 1208 1158 10 1540 1502 38 B6   
## 8 2013 1 1 1222 1159 23 1512 1429 43 EV   
## 9 2013 1 1 1246 1225 21 1424 1348 36 B6   
## 10 2013 1 1 1252 1245 7 1624 1550 34 AA   
## # … with 8,022 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

#Example 5 We will select flights that arrived on time (i.e. no delays):

Between4 <- Bd %>%   
 filter(between(dep\_delay, 1, 30) & between(arr\_delay, 31, Inf))

The new Dataset is as follows

Between4

## # A tibble: 8,032 × 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 1 542 540 2 923 850 33 AA   
## 2 2013 1 1 608 600 8 807 735 32 MQ   
## 3 2013 1 1 702 700 2 1058 1014 44 B6   
## 4 2013 1 1 933 904 29 1252 1210 42 B6   
## 5 2013 1 1 1003 959 4 1408 1329 39 US   
## 6 2013 1 1 1123 1110 13 1410 1336 34 UA   
## 7 2013 1 1 1208 1158 10 1540 1502 38 B6   
## 8 2013 1 1 1222 1159 23 1512 1429 43 EV   
## 9 2013 1 1 1246 1225 21 1424 1348 36 B6   
## 10 2013 1 1 1252 1245 7 1624 1550 34 AA   
## # … with 8,022 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

## Point 5.3.1

## Point 5.3.1

#1 Point

-The function “arrange()” its a function that can be used to organice treams of data for 1 or more variables

-The function na.rom its used to erase trash data, o skip N/A

## 2 Point A

* Now we are going to add the code of the 2nd point:

library(nycflights13)  
library(tidyverse)  
  
nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
latest\_flights <- Bd%>%  
 filter(arr\_delay>120)  
  
latest\_flights

## # A tibble: 10,034 × 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 1 811 630 101 1047 830 137 MQ   
## 2 2013 1 1 848 1835 853 1001 1950 851 MQ   
## 3 2013 1 1 957 733 144 1056 853 123 UA   
## 4 2013 1 1 1114 900 134 1447 1222 145 UA   
## 5 2013 1 1 1505 1310 115 1638 1431 127 EV   
## 6 2013 1 1 1525 1340 105 1831 1626 125 B6   
## 7 2013 1 1 1549 1445 64 1912 1656 136 EV   
## 8 2013 1 1 1558 1359 119 1718 1515 123 EV   
## 9 2013 1 1 1732 1630 62 2028 1825 123 EV   
## 10 2013 1 1 1803 1620 103 2008 1750 138 MQ   
## # … with 10,024 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

-You have to declare the variable ‘lated\_flights’ and then we call the package Bd that contains each one of the variables from the flights

-With the operator “<-”, we create a filter with the function filter to the variable “lated\_flights”

## 2 Point B

You can also embed plots, for example:

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
earliest\_flights <- Bd%>%  
 filter(dep\_delay< -10)   
earliest\_flights

## # A tibble: 6,578 × 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 1 839 850 -11 1027 1035 -8 MQ   
## 2 2013 1 1 940 955 -15 1226 1220 6 MQ   
## 3 2013 1 1 946 959 -13 1146 1202 -16 EV   
## 4 2013 1 1 1716 1730 -14 1947 1953 -6 F9   
## 5 2013 1 1 1849 1900 -11 2131 2129 2 FL   
## 6 2013 1 1 2030 2045 -15 2150 2225 -35 AA   
## 7 2013 1 1 2217 2229 -12 249 315 -26 B6   
## 8 2013 1 2 1102 1115 -13 1430 1425 5 AA   
## 9 2013 1 2 1416 1429 -13 1736 1803 -27 UA   
## 10 2013 1 2 1444 1455 -11 1607 1628 -21 B6   
## # … with 6,568 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

-We declare the variable ‘earliest\_flights’ and then we call the package Bd that contains each variable of the flights

-With the operator “<-”, we create a filter with the function filter to the variable “earliest\_flights”

##3 Point

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
fastest\_flights <- Bd%>%  
 filter(hour>1)  
fastest\_flights

## # A tibble: 336,775 × 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 336,765 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

-We declare the variable ‘fastest\_flights’ and then we call the package Bd that contains each variable of the flights

-With the operator “<-”, we create a filter with the function filter to the variable “fastest\_flights”

##4 Point A

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
fartest\_flights <- Bd%>%  
 filter(distance>=4000)  
fartest\_flights

## # A tibble: 707 × 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 1 857 900 -3 1516 1530 -14 HA   
## 2 2013 1 1 1344 1344 0 2005 1944 21 UA   
## 3 2013 1 2 909 900 9 1525 1530 -5 HA   
## 4 2013 1 2 1344 1344 0 1940 1944 -4 UA   
## 5 2013 1 3 914 900 14 1504 1530 -26 HA   
## 6 2013 1 3 1418 1341 37 2006 1935 31 UA   
## 7 2013 1 4 900 900 0 1516 1530 -14 HA   
## 8 2013 1 4 1343 1341 2 1932 1935 -3 UA   
## 9 2013 1 5 858 900 -2 1519 1530 -11 HA   
## 10 2013 1 5 1329 1335 -6 1850 1935 -45 UA   
## # … with 697 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

-We declare the variable ‘fartest\_flights’ and then we call the package Bd that contains each variable of the flights

-With the operator “<-”, we create a filter with the function filter to the variable “fartest\_flights”

##4 Point B

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
shortest <- Bd%>%  
 filter(distance<90)  
shortest

## # A tibble: 50 × 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 3 2127 2129 -2 2222 2224 -2 EV   
## 2 2013 1 4 1240 1200 40 1333 1306 27 EV   
## 3 2013 1 4 1829 1615 134 1937 1721 136 EV   
## 4 2013 1 4 2128 2129 -1 2218 2224 -6 EV   
## 5 2013 1 5 1155 1200 -5 1241 1306 -25 EV   
## 6 2013 1 6 2125 2129 -4 2224 2224 0 EV   
## 7 2013 1 7 2124 2129 -5 2212 2224 -12 EV   
## 8 2013 1 8 2127 2130 -3 2304 2225 39 EV   
## 9 2013 1 9 2126 2129 -3 2217 2224 -7 EV   
## 10 2013 1 10 2133 2129 4 2223 2224 -1 EV   
## # … with 40 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

-We declare the variable ‘shortest’ and then we call the package Bd that contains each variable of the flights

-With the operator “<-”, we create a filter with the function filter to the variable “shortest”

## Solution 5.5.2

# ítem1

Bd <- nycflights13::flights  
Solucion1 <- Bd %>%  
 mutate(dep\_time\_mins = (dep\_time %/% 100) \* 60 + (dep\_time %% 100),  
 sched\_dep\_time\_mins = (sched\_dep\_time %/% 100) \* 60 + (sched\_dep\_time %% 100))  
##head(flights[, c("dep\_time", "dep\_time\_mins", "sched\_dep\_time", "sched\_dep\_time\_mins")])

En esta expresión, usamos el operador %/% para obtener la parte entera de una hora y el operador %% para obtener la parte fraccionaria de una hora, luego multiplicamos la parte entera por 60 para convertirla en minutos y sumamos un decimal. a minutos para que podamos expresar más convenientemente dev\_time y sched\_dev\_time en variables minutos desde la medianoche, la función mutate() se usa para agregar nuevas variables al conjunto de datos original.

The new Dataset is as follows

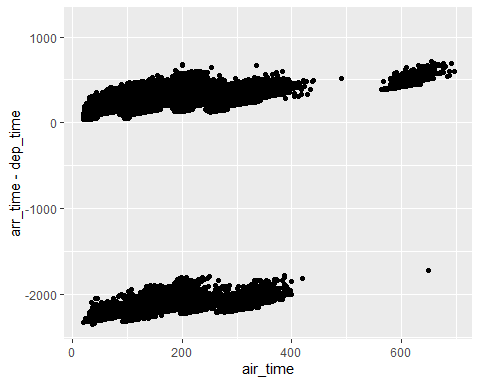
Solucion1

## # A tibble: 336,776 × 21  
## 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 336,766 more rows, 11 more variables: flight <int>, tailnum <chr>,  
## # origin <chr>, dest <chr>, air\_time <dbl>, distance <dbl>, hour <dbl>,  
## # minute <dbl>, time\_hour <dttm>, dep\_time\_mins <dbl>,  
## # sched\_dep\_time\_mins <dbl>, and abbreviated variable names ¹​sched\_dep\_time,  
## # ²​dep\_delay, ³​arr\_time, ⁴​sched\_arr\_time, ⁵​arr\_delay

# Ítem 2

When comparing air\_time with arr\_time - dep\_time, we expect a linear relationship between the two variables. air\_time is the actual flight time and arr\_time - dep\_time is the time from departure to arrival, including the actual flight time plus any delays en route. We can compare two variables using a scatter plot to see if there is a linear relationship. In the ggplot2 library, we can create scatter plots by plotting points using the ggplot() function and the geom\_point() function:

## Warning: Removed 9430 rows containing missing values (`geom\_point()`).



What we see is a massive diffuse point cloud. Although we can see a linear trend in the cloud, there are many points that do not follow this trend. This can be due to several factors, such as weather changes or air traffic. To avoid this, we can consider filtering the data to eliminate flights with severe arrival delays, as this can significantly affect the total flight time. We can also consider other variables that may cause data variability, such as airlines or departure/destination airports.

## Point 5.6.7

* A continuacion se adjunta el codigo del 1er punto:

## 1st form (Calculation of the percentage of delayed flights)

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
X<-mean(Bd$dep\_delay, na.rm = TRUE)  
view(X)

* With this code we can calculate the average of de delay from the departure of every flight in the dataframe nycflights13.

## 2nd form (Calculation of the percentage of on-time flights)

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
sum(Bd$dep\_delay > 0, na.rm = TRUE) / nrow(Bd) \* 100

## [1] 38.13573

* With this code we can calculate the average of the puntual flights in the dataframe nycflights13.

## 3rd form (Distribution analysis)

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
Porcentaje<- sum(Bd$dep\_delay > 0, na.rm = TRUE) / nrow(Bd) \* 100  
view(Porcentaje)

* With this code we can calculate the the distribution analisis of the of every flight in the dataframe nycflights13.

##4th form (Trend analysis)

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
Distribucion<- summary(Bd$dep\_delay)  
view(Distribucion)

* This codes provide us information about the distribution of the times of delay in the dataframe nycflights13, including the medium, the standar deviation and other stadistic descriptive data

## 5th form (Root cause analysis)

nycflights13::flights

## # A tibble: 336,776 × 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 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

Bd <- nycflights13::flights  
  
sd(Bd$dep\_delay, na.rm = TRUE)

## [1] 40.21006

* This code can calculate the average of delay on the departure of every flight in the dataframe nycflights13.

## Point 5.7.1

## 2nd Point

* Now we are going to add the code of the 2do point:

nycflights13::flights

## # A tibble: 336,776 × 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 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

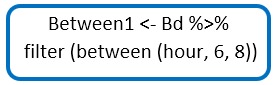
Bd <- nycflights13::flights  
  
worst\_arrive\_score <- Bd%>%  
 filter(arr\_delay==430)

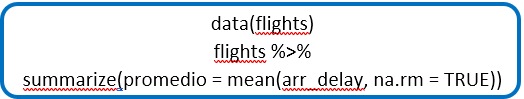
-We declare the variable ‘worst\_arrive\_score’ and then we call the package Bd that contains each variable of the flights

-With the operator “<-”, we create a filter with the function filter to the variable “worst\_arrive\_score”

#Solution 3 ## Between The function between is used in the RStudio programming language to verify if a value is in a range of values of a specific data frame. The syntax and form to express this function is: x

where x is the value to test, left is the left boundary of the range and right is the right boundary of the range. The function returns TRUE if the value of x is between the left and right limits; otherwise, it returns FALSE. The between function is useful to perform filtering or data selection operations, here is an example performed during the execution of the assigned points where you want to know the flights that had departed between 6 and 8 o’clock.

 ##summarise The sumarrise function is used in the R programming language where it basically facilitates the user to acquire new specific information by generating a single output or value of an entire data frame or database, as its name suggests it helps to summarize and obtain data from a large set of data, a clear example are the measures of central tendency such as the mean. The first step to start using this function is to run the dplyr library as it is the tool that allows us to make use of functions such as filtering, selection, aggregation, sorting and transformation of data, then you must select the database to be studied, the next step is to call the summarise function and a new variable will be assigned the operation either total amount, mean, mode, etc. of the database column to be analyzed.



insertando imagen



insertando imagen

#para asignar el nombre al chunk dejamos un espacio   
#si lo configuramos como setup se ejecutara ese chunk de primero  
#child NULL A character vector of filenames. Knitr will knit the files and place them into the ma  
#engine 'R' Knitr will evaluate the chunk in the named language, e.g. engine = 'python'. Run #names(knitr::knit\_engines$get()) to  
#see supported languages.  
#purl TRUE If FALSE, knitr will not include the chunk when running purl() to extract the source code.  
#error TRUE If FALSE, knitr will not display any error messages generated by the code.  
#comment '##' A character string. Knitr will append the string to the start of each line of results in the final document

## Point 4

A repository is created where the codes of the development of the first deliverable are found, the documentation of several functions that were included among which are functions such as: find the planes that had a delay of more than 1 hour. planes that departed during the summer flights that departed between midnight and 6 a.m. flights operated by united, delta and american the url of this repository is the following: <https://github.com/AndresFelipe120320/ProyectoR>

In the R folder the file 5.2.4 is where you can find the development of the activity points.

In the man folder you will find the documentation of the different functions created, with some examples of use.

Next you will see the function “retrieve\_answer” which according to the parameter from 1 to 6 returns the answer of point 5.2.4 item 1.

retrieve\_answer <- function(v){  
  
 if(v==1){  
 R2 <- RetrasoMas2Horas()  
 print("Vuelos retrasado mas de 2 horas")  
 return(R2)  
  
 }else if(v==2){  
 Vh <- VuelosHouston()  
 print("Vuelos a houston")  
 return(Vh)  
 }  
 else if(v==3){  
 Op <- VoperadosUUAUDL()  
 print("operados por united, america, delta")  
 return(Op)  
 }  
 else if(v==4){  
 Sv <- SalidaVerano()  
 print("Vuelos que salieron en verano")  
 return(Op)  
 }  
 else if(v==5){  
 R1h <- RetrasoAlM1hora()  
 print("Retraso de al menos 1 hora")  
 return(R1h)  
 }  
 else if(v==6){  
 Ve12To6 <- Vuelos12To6()  
 print("Vuelos de media noche a 6 de la mañana")  
 return(Ve12To6)  
 }else{  
 print("Valor no encontrado")  
 }  
}

some examples # Exmples

Retrasos <- retrieve\_answer(1)

## [1] "Vuelos retrasado mas de 2 horas"

Retrasos

## # A tibble: 8,482 × 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 1 848 1835 853 1001 1950 851 MQ   
## 2 2013 1 1 957 733 144 1056 853 123 UA   
## 3 2013 1 1 1114 900 134 1447 1222 145 UA   
## 4 2013 1 1 1815 1325 290 2120 1542 338 EV   
## 5 2013 1 1 1842 1422 260 1958 1535 263 EV   
## 6 2013 1 1 1856 1645 131 2212 2005 127 AA   
## 7 2013 1 1 1934 1725 129 2126 1855 151 MQ   
## 8 2013 1 1 1938 1703 155 2109 1823 166 EV   
## 9 2013 1 1 1942 1705 157 2124 1830 174 MQ   
## 10 2013 1 1 2006 1630 216 2230 1848 222 EV   
## # … with 8,472 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