Coding\_Assignment\_4

Noah Foilb

9/23/2020

#### Step 1) Load the nycflights13, dplyr, tidyr, and mosaic libraries.

* Once you are sure that these load properly, you can suppress the warnings from appearing in your final knitted output file by changing {r} at the beginning of the code chunck to {r, message=FALSE} if you want.

#load the library's by using the library() function  
  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(mosaic)

## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggplot2

## Loading required package: ggstance

##   
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':  
##   
## geom\_errorbarh, GeomErrorbarh

##   
## New to ggformula? Try the tutorials:   
## learnr::run\_tutorial("introduction", package = "ggformula")  
## learnr::run\_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix

##   
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':  
##   
## expand, pack, unpack

## Registered S3 method overwritten by 'mosaic':  
## method from   
## fortify.SpatialPolygonsDataFrame ggplot2

##   
## The 'mosaic' package masks several functions from core packages in order to add   
## additional features. The original behavior of these functions should not be affected by this.  
##   
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.  
##   
## Have you tried the ggformula package for your plots?

##   
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':  
##   
## mean

## The following object is masked from 'package:ggplot2':  
##   
## stat

## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally

## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,  
## quantile, sd, t.test, var

## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

library(nycflights13)

#### Step 2) We will be working with a subset of the flights data frame. Starting with the data frame called flights, create a new data frame called f2 (since this is step 2) that includes all the following eight pieces of information about each flight: month, dep\_delay, arr\_delay, carrier, origin, dest, air\_time, distance.

* As usual, inspect this newly created data frame using ?, str, and head before proceeding.
* Comments are not necessary in this part.

f2 <- flights %>%  
 select(month,dep\_delay,arr\_delay,carrier,origin,dest,air\_time,distance)  
str(f2)

## tibble [336,776 x 8] (S3: tbl\_df/tbl/data.frame)  
## $ month : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...  
## $ dep\_delay: num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...  
## $ arr\_delay: num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...  
## $ carrier : chr [1:336776] "UA" "UA" "AA" "B6" ...  
## $ origin : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...  
## $ dest : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...  
## $ air\_time : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...  
## $ distance : num [1:336776] 1400 1416 1089 1576 762 ...

f2

## # A tibble: 336,776 x 8  
## month dep\_delay arr\_delay carrier origin dest air\_time distance  
## <int> <dbl> <dbl> <chr> <chr> <chr> <dbl> <dbl>  
## 1 1 2 11 UA EWR IAH 227 1400  
## 2 1 4 20 UA LGA IAH 227 1416  
## 3 1 2 33 AA JFK MIA 160 1089  
## 4 1 -1 -18 B6 JFK BQN 183 1576  
## 5 1 -6 -25 DL LGA ATL 116 762  
## 6 1 -4 12 UA EWR ORD 150 719  
## 7 1 -5 19 B6 EWR FLL 158 1065  
## 8 1 -3 -14 EV LGA IAD 53 229  
## 9 1 -3 -8 B6 JFK MCO 140 944  
## 10 1 -2 8 AA LGA ORD 138 733  
## # ... with 336,766 more rows

#### Step 3) Starting with data frame f2, create a table showing the number of flights leaving New York City each month, sorted from highest to lowest.

* You will need the group\_by, summarize, and arrange commands.
* Name the column showing the number of flights monthly\_flights.
* In this step and all subsequent steps, make sure you comment your code to explain what you’re doing.
* If done correctly, you will find July is the top month for flights with 29,425.

# Calling the dataset to be able to manipulate it  
f2 %>%  
   
 # Using Group\_By to arrange the data with relation to the month  
 group\_by(month) %>%  
   
 # Using summarise to be able to count the amount of flights per month  
 summarise(monthly\_flights = n()) %>%  
   
 # Using arrange to be able to set the monthly flights from highest to lowest  
 arrange(desc(monthly\_flights))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 12 x 2  
## month monthly\_flights  
## <int> <int>  
## 1 7 29425  
## 2 8 29327  
## 3 10 28889  
## 4 3 28834  
## 5 5 28796  
## 6 4 28330  
## 7 6 28243  
## 8 12 28135  
## 9 9 27574  
## 10 11 27268  
## 11 1 27004  
## 12 2 24951

#### Step 4) Starting with data frame f2, create a table that counts the number of flights leaving New York City each month from each airport. Display the month, airport of origin, and the number of flights. Only show months and airports where the number of flights exceeds 10,000.

* You will need to group\_by two different variables and also use summarize, filter, and arrange.
* Name the column showing the number of flights monthly\_flights.
* If done correctly, EWR will have 10,592 flights leaving in May as the top result.

# Calling the dataset to be able to manipulate it  
f2 %>%  
   
 # Using Group\_By to arrange the data with relation to the month and the origin  
 group\_by(month,origin) %>%  
   
 # Using summarise to be able to count the amount of flights per month  
 summarise(monthly\_flights = n()) %>%  
   
 # Using Filter to make the table only display the values over 10,000  
 filter(monthly\_flights>10000) %>%  
   
 # Using arrange to be able to set the monthly flights from highest to lowest  
 arrange(desc(monthly\_flights))

## `summarise()` regrouping output by 'month' (override with `.groups` argument)

## # A tibble: 8 x 3  
## # Groups: month [7]  
## month origin monthly\_flights  
## <int> <chr> <int>  
## 1 5 EWR 10592  
## 2 4 EWR 10531  
## 3 7 EWR 10475  
## 4 3 EWR 10420  
## 5 8 EWR 10359  
## 6 6 EWR 10175  
## 7 10 EWR 10104  
## 8 7 JFK 10023

#### Step 5) Starting with data frame f2, create a table showing the fastest air speed (Fastest\_speed), slowest air speed (Slowest\_speed), and average air speed (Average\_speed) for all flights in the f2 data frame where speed is measured in miles per hour (mph). Only include flights for which both air\_time and distance are available.

* Use drop\_na to remove air\_time and distance entries listed as NA
* Create a new column called air\_speed where you calculate air speed in mph from the distance (reported in miles) and the air\_time (reported in minutes). Make sure you convert from minutes to hours!
* Within summarise, calculate the fastest, slowest, and average air speed.
* If done correctly, max, min, and avg air speeds are 703.4, 76.8, and 394.3 mph.

# Calling the dataset to be able to manipulate it  
f2 %>%  
   
 # Using Drop\_Na to drop the values that do not exist so it does not interfere with the data  
 drop\_na(air\_time,distance)%>%  
   
 # Using Mutate to be able to create a column names "AirSpeed" as said in the instructions  
 mutate(air\_speed = distance/(air\_time/60)) %>%  
   
 # Using Summarize to be able to display the data I want to  
 summarise(  
   
 #First column: assigning the variable to the max airspeed and rounding it  
 Fastest\_speed = round(max(air\_speed),1),   
   
 #Second column: assigning the variable to the min airspeed and rounding it  
 Slower\_speed = round(min(air\_speed),1),   
   
 #Third column: assigning the variable to the average airspeed and rounding it  
 Average\_speed = round(mean(air\_speed),1))

## # A tibble: 1 x 3  
## Fastest\_speed Slower\_speed Average\_speed  
## <dbl> <dbl> <dbl>  
## 1 703. 76.8 394.

#### Step 6) Starting with data frame f2, create a table showing all the flights grouped by month for which departure delay is available. For each month, list: the total number of flights having the dep\_delay available as Flights\_with\_dep\_delay\_info, the number of delayed departures as Delayed\_departures, and the percent of delayed flights as Percent\_delayed (report percent as a number between 0 and 100). Sort the table from the lowest to highest percentage of delayed departures.

* Use drop\_na to remove flights with missing departure delays
* Use group\_by, summarise, and arrange
* Within the summarise command:
  + use sum or count to find all flights that were delayed (dep\_delay>0), and
  + calculate the Percent\_delayed.
* If done correctly, September will be the month with the lowest percent of flights with departure delays with 27,122 total flights, 7,815 departure delays, and 28.8% of the flights delayed.

# Calling the dataset to be able to manipulate it  
f2 %>%  
   
 # Using Drop\_Na to drop the values that do not exist so it does not interfere with the data  
 drop\_na(dep\_delay) %>%  
   
 # Using Group\_By to arrange the data with relation to the month  
 group\_by(month) %>%  
   
 # Using Summarize to be able to display the data I want to  
 summarise(  
   
 #First column: assigning the variable to n() to be able to count the amount of drop delay info available  
 Flights\_with\_dep\_delay\_info = n(),  
   
 #Second column: assigning the variable to count the amount of delayed departed that are >0 to count the ones that matter   
 Delayed\_departures = count(dep\_delay > 0 ),  
   
 #Third column: assigning the varaible to dived Flights\_wit.. column by Delated\_depar.. column to be able to get the percent  
 Percent\_delayed = round(Delayed\_departures\*100/Flights\_with\_dep\_delay\_info,1))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 12 x 4  
## month Flights\_with\_dep\_delay\_info Delayed\_departures Percent\_delayed  
## <int> <int> <int> <dbl>  
## 1 1 26483 9662 36.5  
## 2 2 23690 9124 38.5  
## 3 3 27973 11209 40.1  
## 4 4 27662 10543 38.1  
## 5 5 28233 11291 40   
## 6 6 27234 12655 46.5  
## 7 7 28485 13909 48.8  
## 8 8 28841 11713 40.6  
## 9 9 27122 7815 28.8  
## 10 10 28653 8722 30.4  
## 11 11 27035 8239 30.5  
## 12 12 27110 13550 50

#### Step 7) Starting with data frame f2, create a table containing all the flights for which both the departure delay and arrival delay are available. In this table, report the total number of such flights as Flights\_with\_both\_delay\_info, the number of flights which left late but arrived on time or early as Left\_late\_but\_arrived\_early, and the percent that fell into this category as Percent (report percent as a number between 0 and 100).

* Use drop\_na() to remove flights with missing departure and arrival delays.
* Within the summarise command:
  + use sum or count to find all flights that left late but arrived early (use a logical operator to specify this compound condition), and
  + calculate the Percent.
* If done correctly, there will be 327,346 flights with this info available, 35,442 will have left late but arrived early, and the percent will be 10.8%

# Calling the dataset to be able to manipulate it  
f2 %>%  
   
 # Using Drop\_Na to drop the values that do not exist so it does not interfere with the data  
 drop\_na(arr\_delay,dep\_delay) %>%  
   
   
 summarise(  
   
 #First column: assigning the variable to n() to be able to count the amount of both drop delay info available  
 Flights\_with\_both\_delay\_info = n(),   
   
 #First column: assigning the variable to n() to be able to count when it is late but arrived on time  
 Left\_late\_but\_arrived\_early = count(dep\_delay >0 & arr\_delay <= 0),  
   
 #First column: assigning the variable to n() to be able to calculate the percent   
 Percent = round(Left\_late\_but\_arrived\_early\*100/Flights\_with\_both\_delay\_info,1))

## # A tibble: 1 x 3  
## Flights\_with\_both\_delay\_info Left\_late\_but\_arrived\_early Percent  
## <int> <int> <dbl>  
## 1 327346 35442 10.8

#### Step 8) Starting with data frame f2, create a table showing the three most popular destinations and the number of flights that flew to each. Provide an appropriate name for the column counting the number of flights.

* If done correctly, there will be 17,283 flights to the most popular destination.

# Calling the dataset to be able to manipulate it  
f2 %>%  
   
 # Using Group\_By to arrange the data with relation to the   
 group\_by(dest) %>%  
   
 # Using summarise to be able to count the number of times flown to the location  
 summarise(Number\_of\_times\_flown\_to = n()) %>%  
   
 # Using arrange to be able to set the Number of times flown to from highest to lowest  
 arrange(desc(Number\_of\_times\_flown\_to))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 105 x 2  
## dest Number\_of\_times\_flown\_to  
## <chr> <int>  
## 1 ORD 17283  
## 2 ATL 17215  
## 3 LAX 16174  
## 4 BOS 15508  
## 5 MCO 14082  
## 6 CLT 14064  
## 7 SFO 13331  
## 8 FLL 12055  
## 9 MIA 11728  
## 10 DCA 9705  
## # ... with 95 more rows

#### Step 9) Inspect the result of the previous question to identify the most popular destination. Create a table showing the TWO carriers that flew the most fights to this (most popular) destination and the total number of flights they each flew there.

* If done correctly, UA will have 6,984 flights and AA 6,059.

# Calling the dataset to be able to manipulate it  
f2 %>%  
 filter(dest == "ORD") %>%  
   
 # Using Group\_By to arrange the data with relation to the   
 group\_by(carrier) %>%  
   
 # Using summarise to be able to count the number of flights  
 summarise(Number\_of\_Flights = n()) %>%  
   
 # Using arrange to be able to set the Number of Flights from highest to lowest  
 arrange(desc(Number\_of\_Flights))

## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 7 x 2  
## carrier Number\_of\_Flights  
## <chr> <int>  
## 1 UA 6984  
## 2 AA 6059  
## 3 MQ 2276  
## 4 9E 1056  
## 5 B6 905  
## 6 EV 2  
## 7 OO 1