Homework07

Peyton Hall

03/14/2024

Complete the activities mentioned in the videos

```
# Everything copied and pasted from slide 20:
# Example

# df <- data.frame(a = 1:5, b = c(10, 20, 30, 40, 50))
# df <- df %>%mutate(c = a + b)
# df

# perform multiple operations in a single mutate() call
# df <- df %>%
#   mutate(
#   c = a + b,
#   d = a * 2,
#   e = b/2
#   )
# print(df)
```

```
# Everything copied and pasted from slide 20:
# Overwrite existing columns:

# df <- df %>%
#   mutate(a = a * 100)
# print(df)

# Using helper functions: dplyr provides several helper functions that can be used inside mutate(). For
# instance, if_else() can be useful for conditional changes:

# df <- df %>%
#   mutate(f = if_else(a > 200, "High", "Low"))
# print(df)
```

```
# Example:
# 1. use the Orange data,
#Choose the trees with circumference >100
# Add a column to show the unit circumference using circumference/age
#(name that variable ad unitc)
# Save the changes as new data NewOrange

# NewOrange<-Orange%>%
#   filter(circumference>100)%>%
#   mutate(unitc = circumference/age)
# print(NewOrange)

# Use the InsectSprays data, and add a new column "Odor" to re-group the sprays: If the spray
# is A, B or C, group them as group LowOdor, otherwise call the spray High odor. Use the ifelse
# function to do this.

# library(tidyverse)

# InsectSprays%>%
#   mutate(Odor=ifelse( spray %in% c("A", "B", "C"), "LowOdor", "HighOdor"))
```

```
# use the flights data from the nycflights13 package
# Choose the flights in January, from JFK to MSP
# Select the variables: dep_time, dep_delay, arr_time, arr_delay, air_time, and
#distance
# Add a column to show the speed of selected flights: speed=distance/ air time
# Create a new file "newflight" to save the above changes

library(nycflights13)
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
```

```
# Filter flights from JFK to MSP in January
january_flights <- filter(flights, month == 1, origin == "JFK", dest == "MSP")

# Select required variables
selected_flights <- select(january_flights, dep_time, dep_delay, arr_time, arr_delay, air_time, distanc

# Add a column for speed
selected_flights <- mutate(selected_flights, speed = distance / air_time)

# Save the modified data frame to a new file
write.csv(selected_flights, "newflight.csv", row.names = FALSE)
```

```
# Package: "dplyr"
# The arrange() function is used to reorder rows of a dataframe by column values.

# data %>% arrange(variablename)

# Default: sort by ascending order
# If want to sort by descending order, use desc(variable)
```

```r
# Example:Assume you have a dataframe df with columns A and B
df <- data.frame(A = c(3, 1, 4), B = c(6, 5, 7))
#  arrange df by column A in ascending order as:
arrange(df, A)
```

```
##   A B
## 1 1 5
## 2 3 6
## 3 4 7
```

```r
# OR
df %>% arrange(A)
```

```
##   A B
## 1 1 5
## 2 3 6
## 3 4 7
```

```r
# Descending order: If you want to sort in descending order, you can use the desc()
# function:
arrange(df, desc(A))
```

```
##   A B
## 1 4 7
## 2 3 6
## 3 1 5
```

```r
# OR
df %>%arrange(desc(A))
```

```
##   A B
## 1 4 7
## 2 3 6
## 3 1 5
```

```r
# Sorting by multiple columns: If you want to sort by multiple columns, simply add the
# columns in the order you want:
arrange(df, A, B)
```

```
##   A B
## 1 1 5
## 2 3 6
## 3 4 7
```

```r
# OR
df %>% arrange(A,B)
```

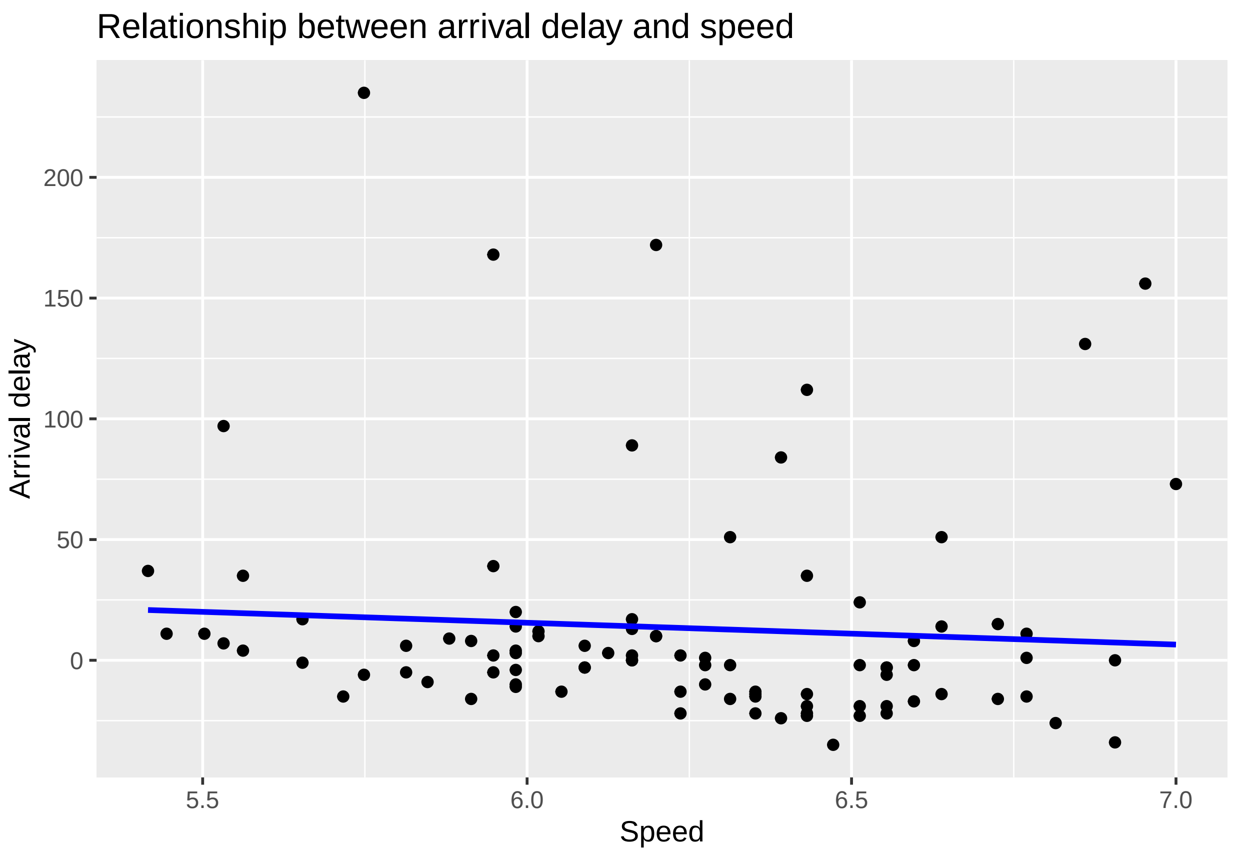
```
##   A B
## 1 1 5
## 2 3 6
## 3 4 7
```

```r
# This will first sort by column A and then by column B for rows with the same values of
```

```r
# Example: Sort the OrangeNew data by circumference

# NewOrange%>%
#   arrange(circumference)
```

```r
# Activity 03
# Use the newflight data created from the previous activity
# Sort by the arrival delay time
# Generate an appropriate graph to see the relationship between arrival
# delay and speed
# Is there a strong correlation?

library(nycflights13)
library(dplyr)
library(ggplot2)

# Load the previously created CSV file into R
newflight <- read.csv("newflight.csv")

# Sort by the arrival delay time
sorted_flights <- newflight %>% arrange(arr_delay)

# Generate a scatterplot to visualize the relationship between arrival delay and speed
ggplot(sorted_flights, aes(x = speed, y = arr_delay)) +
  geom_point(color = "black") +
  geom_smooth(method = "lm", se = FALSE, color = "blue") +
  labs(title = "Relationship between arrival delay and speed",
       y = "Arrival delay",
       x = "Speed")
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```



```
# Package: "dplyr"
# group_by()
# The group_by() function groups data by one or more columns.
# The result is a grouped data frame, but it looks almost the same as the original data
# frame.
# The difference becomes apparent when you use it with other dplyr verbs.
data <- data.frame(
  Category = c('A', 'B', 'A', 'A', 'B', 'B'),
  Value = c(10, 20, 30, 40, 50, 60)
  )
grouped_data <- data %>%
group_by(Category)
```

```
# Package: "dplyr"
# summarize()
# The summarize() function is used to compute summary statistics for each group.
# It can be used with any function that returns a single value (like mean, sum, min, etc.).
# When you use summarize() on a grouped data frame, it computes the summary
# statistics for each group and returns a new data frame.
# Example:
summary_data <- grouped_data %>%
  summarize(
    Total = sum(Value),
    Average = mean(Value)
  )
summary_data
```

```
## # A tibble: 2 x 3
##   Category Total Average
##   <chr>    <dbl>   <dbl>
## 1 A           80    26.7
## 2 B          130    43.3
```

```
# group_by() and summarize()
# group_by() and summarize() usually go together
# Example:
data %>%
group_by(Category) %>%
  summarize(
    Total = sum(Value),
    Average = mean(Value)
  )
```

```
## # A tibble: 2 x 3
##   Category Total Average
##   <chr>    <dbl>   <dbl>
## 1 A           80    26.7
## 2 B          130    43.3
```

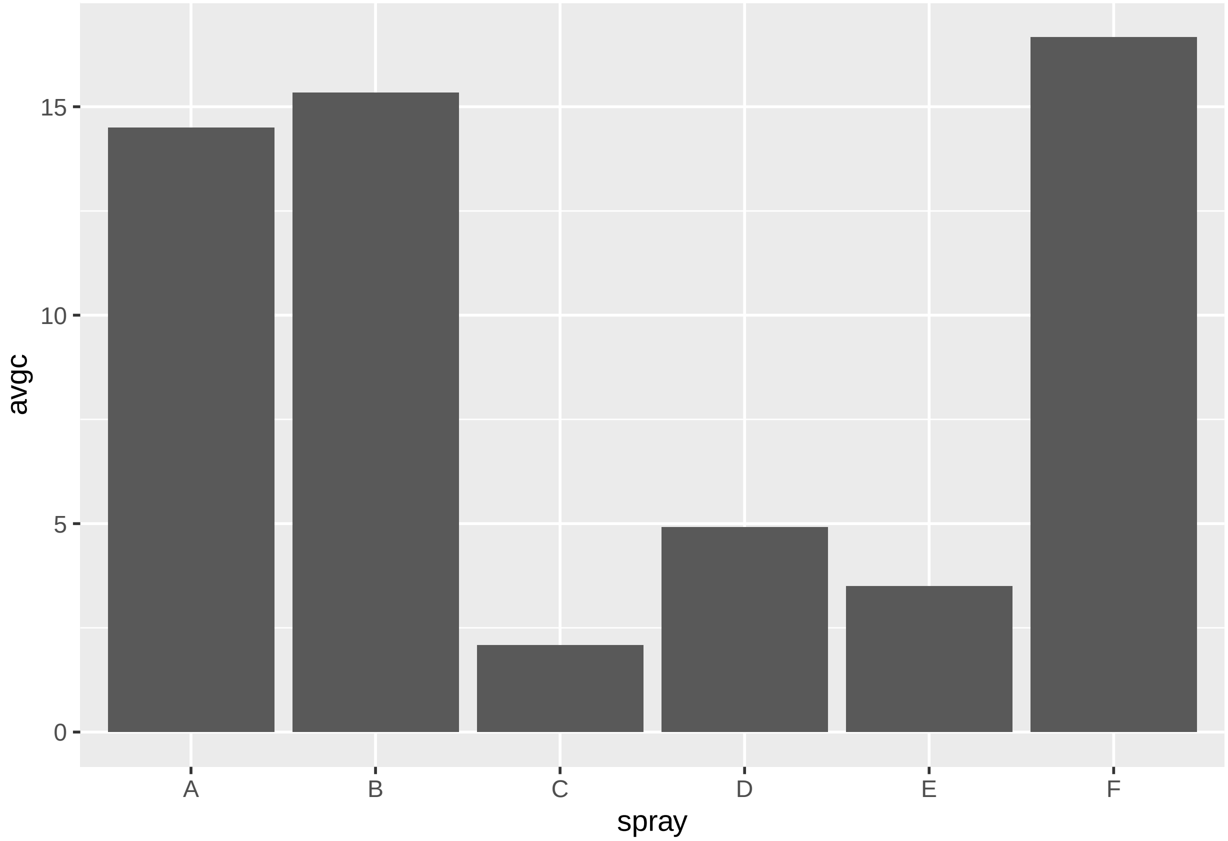
```
# Example:
# Use the InsectSprays data to get the mean count of each spray, and plot the
# means using a bar chart.
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
head(InsectSprays)
```

```
##   count spray
## 1    10     A
## 2     7     A
## 3    20     A
## 4    14     A
## 5    14     A
## 6    12     A
```

```
InsectSprays%>%
  group_by(spray)%>%
  summarize(avgc=mean(count))%>%
  ggplot(aes(x=spray, y=avgc))+geom_bar(stat = "identity")
```



```
mpg%>%
  filter(!is.na(hwy))%>%
  group_by(manufacturer)%>%
  summarize(avghwy=mean(hwy))
```

```
## # A tibble: 15 x 2
##    manufacturer avghwy
##    <chr>         <dbl>
##  1 audi           26.4
##  2 chevrolet      21.9
##  3 dodge          17.9
##  4 ford           19.4
##  5 honda          32.6
##  6 hyundai        26.9
##  7 jeep           17.6
##  8 land rover     16.5
##  9 lincoln        17
## 10 mercury        18
## 11 nissan         24.6
## 12 pontiac        26.4
## 13 subaru          25.6
## 14 toyota          24.9
## 15 volkswagen      29.2
```

```
mpg %>%
  group_by(manufacturer) %>%
  summarize(total_models = n()) %>%
  arrange(desc(total_models))
```

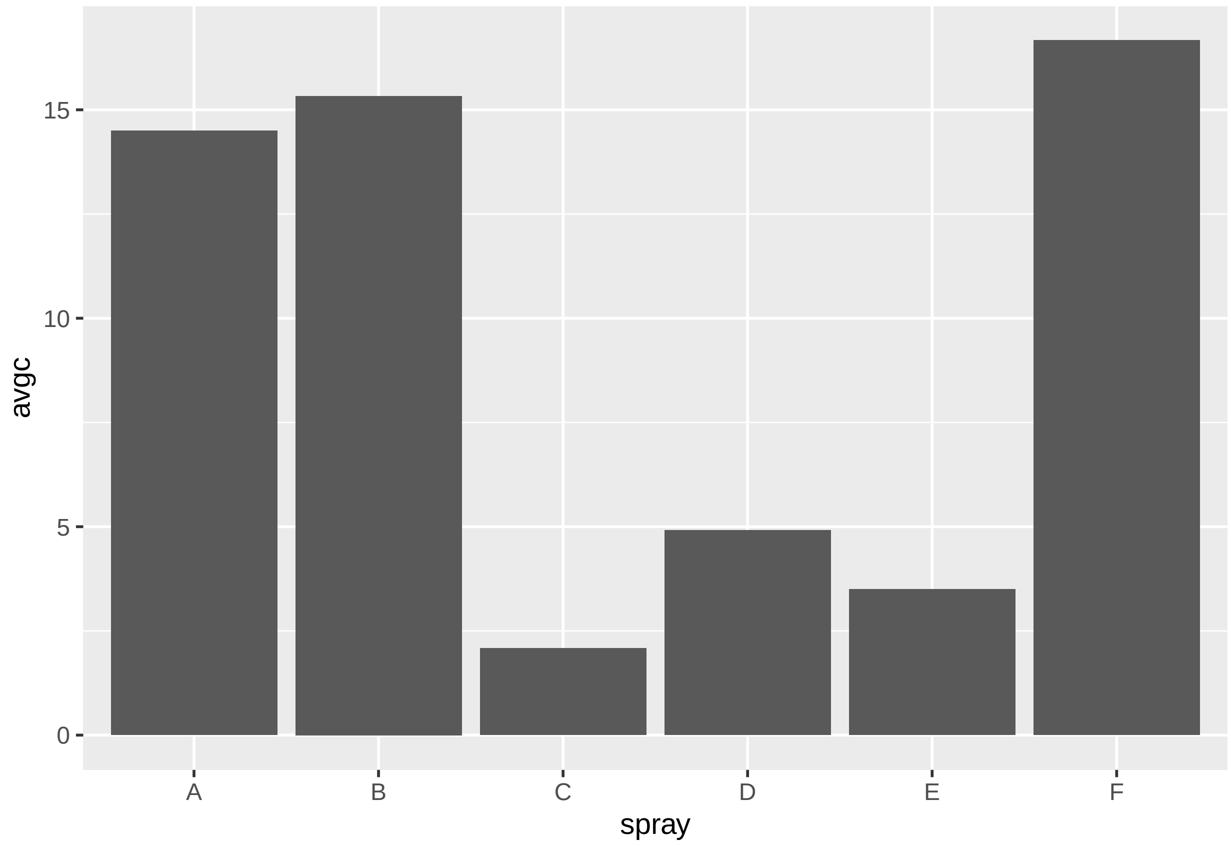
```
## # A tibble: 15 x 2
##    manufacturer total_models
##    <chr>               <int>
##  1 dodge                  37
##  2 toyota                 34
##  3 volkswagen             27
##  4 ford                   25
##  5 chevrolet              19
##  6 audi                   18
##  7 hyundai                14
##  8 subaru                 14
##  9 nissan                 13
## 10 honda                   9
## 11 jeep                    8
## 12 pontiac                 5
## 13 land rover              4
## 14 mercury                 4
## 15 lincoln                 3
```

```
# Values of NA is considered as missing values
# If there are missing values, we can't find the mean
# To find the descriptive statistics without missing values:

# filter(!is.na(variable)) %>%
#   group_by(...) %>%
#   summarize(average=mean(variable))

# OR

# group_by() %>%
#   summarize(average=mean(variable, na.rm=TRUE))

# To find the total of none missing individuals:

# filter(!is.na(variable) %>%
# group_by() %>%
# summarize(total=n())
```

```
# Example: Use the mpg data to do the following
# Keep all the none NA's from the hwy, and find the average highway
# miles per gallon by manufacturer. Which manufacturer has the
# highest highway miles per gallon?
# Group by manufacturer and find the total number of each
# manufacturer of cars. (there is no missing value of manufacturer, so no
# worry about missing values here
library(tidyverse)
library(ggplot2)
head(InsectSprays)
```

```
##   count spray
## 1    10     A
## 2     7     A
## 3    20     A
## 4    14     A
## 5    14     A
## 6    12     A
```

```
InsectSprays%>%
  group_by(spray)%>%
  summarize(avgc=mean(count))%>%
  ggplot(aes(x=spray, y=avgc))+geom_bar(stat = "identity")
```



```
mpg%>%
  filter(!is.na(hwy))%>%
  group_by(manufacturer)%>%
  summarize(avghwy=mean(hwy))
```

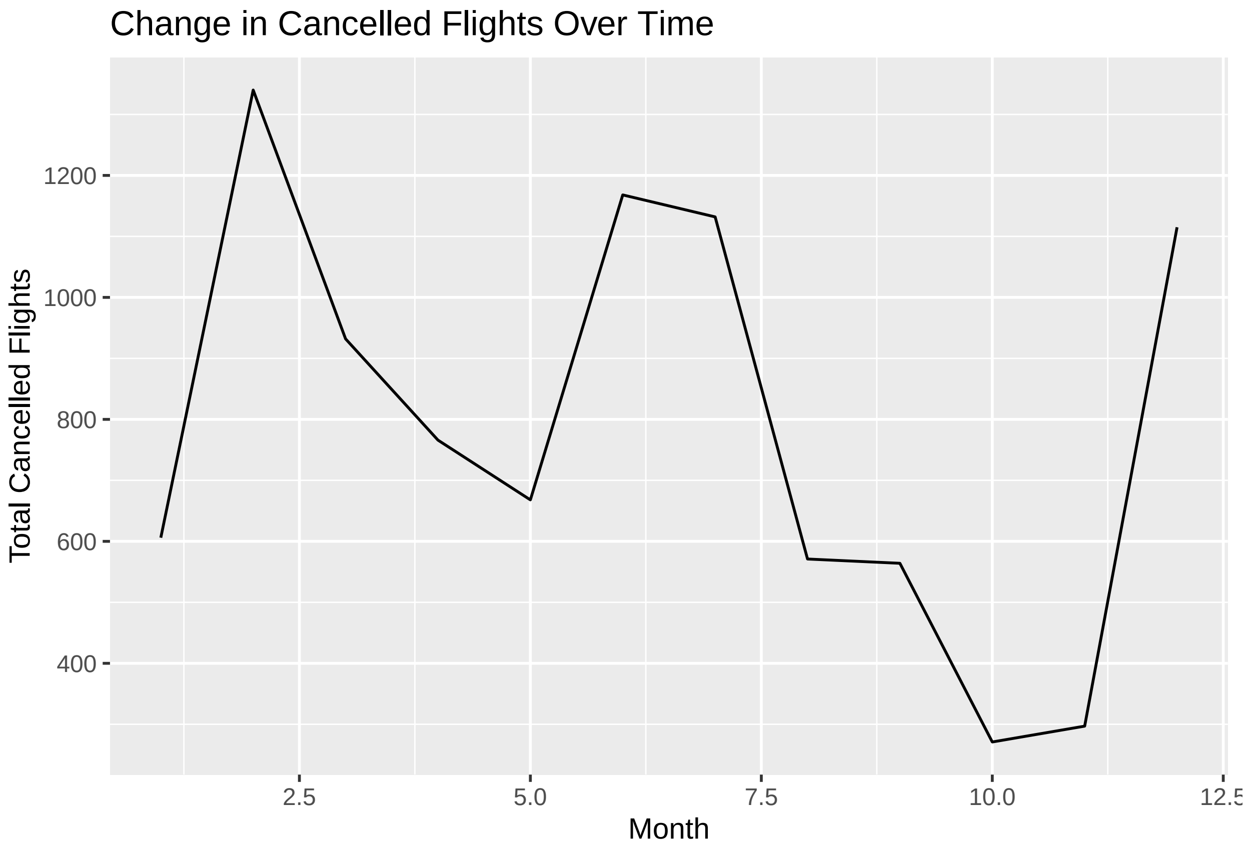
```
## # A tibble: 15 x 2
##    manufacturer avghwy
##    <chr>         <dbl>
##  1 audi           26.4
##  2 chevrolet      21.9
##  3 dodge          17.9
##  4 ford           19.4
##  5 honda          32.6
##  6 hyundai        26.9
##  7 jeep           17.6
##  8 land rover     16.5
##  9 lincoln        17
## 10 mercury        18
## 11 nissan         24.6
## 12 pontiac        26.4
## 13 subaru         25.6
## 14 toyota         24.9
## 15 volkswagen     29.2
```

```
mpg %>%
  group_by(manufacturer) %>%
  summarize(totalm = n()) %>%
  arrange(desc(totalm))
```

```
## # A tibble: 15 x 2
##    manufacturer totalm
##    <chr>         <int>
##  1 dodge            37
##  2 toyota           34
##  3 volkswagen       27
##  4 ford             25
##  5 chevrolet        19
##  6 audi             18
##  7 hyundai          14
##  8 subaru           14
##  9 nissan           13
## 10 honda             9
## 11 jeep              8
## 12 pontiac           5
## 13 land rover        4
## 14 mercury           4
## 15 lincoln           3
```

```
# Activity 04
# Use the flights data
# Keep the none NA's of departure delay values, and find the mean departure
# delay by destination
# Find the total missing departure delay flights by destination
# (Hint: keep all NA's of departure delays and find the total by destination)
# The missing values of air_time show the cancelled flights, find the total cancelled
# flights by month, and generate a line chart to see the change over time.
library(nycflights13)
library(dplyr)
library(ggplot2)
```

```
# Keep non-NA departure delay values and find mean departure delay by destination
mean_dep_delay <- flights %>%
  filter(!is.na(dep_delay)) %>%
  group_by(dest) %>%
  summarize(mean_dep_delay = mean(dep_delay, na.rm = TRUE))

# Find total missing departure delay flights by destination
total_missing_dep_delay <- flights %>%
  filter(is.na(dep_delay)) %>%
  group_by(dest) %>%
  summarize(total_missing_dep_delay = n())

# Find total cancelled flights by month
cancelled_flights <- flights %>%
  filter(is.na(air_time)) %>%
  group_by(month) %>%
  summarize(total_cancelled_flights = n())

# Generate a line chart to visualize the change in cancelled flights over time
ggplot(cancelled_flights, aes(x = month, y = total_cancelled_flights)) +
  geom_line() +
  labs(x = "Month", y = "Total Cancelled Flights", title = "Change in Cancelled Flights Over Time")
```



```
# This function returns all rows from x where there are matching values in y, and all
# columns from x and y. If there are multiple matches between x and y, all
# combinations of the matches are returned.
# Rows in x with no match in y and rows in y with no match in x will not be in the
# result.
library(dplyr)
  df1 <- data.frame(ID = c(1, 2, 3), Name = c("A", "B", "C"))
  df2 <- data.frame(ID = c(2, 3, 4), Score = c(80, 90, 100))
inner_join(df1, df2, by = "ID")
```

```
##   ID Name Score
## 1  2    B    80
## 2  3    C    90
```

```
# left_join
# This function returns all rows from x, and all columns from x and y. Rows in x with
# no match in y will still be returned, but with NA in the columns from y.
# In other words, it keeps all rows from the "left" dataset (i.e., x), regardless of
# whether there is a matching row in the "right" dataset (y).
left_join(df1, df2, by = "ID")
```

```
##   ID Name Score
## 1  1    A    NA
## 2  2    B    80
## 3  3    C    90
```

```
# right_join
# This function returns all rows from y, and all columns from x and y. Rows in y with
# no match in x will still be returned, but with NA in the columns from x.
# It's the opposite of a left_join(). It keeps all rows from the "right" dataset (i.e., y),
# regardless of whether there is a matching row in the "left" dataset (x).
right_join(df1, df2, by = "ID")
```

```
##   ID Name Score
## 1  2    B    80
## 2  3    C    90
## 3  4 <NA>   100
```

```
# Inner_join, left_join, and right_join
# Example: join the datasets of flights and airlines by "carrier"
# If there are more than 1 common variable between two data frames, use
# by=c()
library(nycflights13)
inner_join(flights, airlines, by = "carrier")
```

```
## # A tibble: 336,776 x 20
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # i 336,766 more rows
## # i 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>, name <chr>
```

```
# Activity 05
# Merge the following two data sets by both "country" and "year"

# Country, Year, x1
# Canada,  2000, 21
# Canada,  2001, 25
# USA,     2000, 23
# USA,     2001, 28
# Japan,   2000, 29
# Japan,   2001, 30

# Country, Year, x2
# Canada,  2000, 2
# Canada,  2001, 6
# USA,     2000, 7
# USA,     2001, 12
# Japan,   2000, 6
# Japan,   2001, 30

# Create the first dataset
df1 <- data.frame(
  country = c("Canada", "Canada", "USA", "USA", "Japan", "Japan"),
  year = c(2000, 2001, 2000, 2001, 2000, 2001),
  x1 = c(21, 25, 23, 28, 29, 30)
)

# Create the second dataset
df2 <- data.frame(
  country = c("Canada", "Canada", "USA", "USA", "Japan", "Japan"),
  year = c(2000, 2001, 2000, 2001, 2000, 2001),
  x2 = c(2, 6, 7, 12, 6, 30)
)

# Merge the datasets by both "country" and "year"
merged_df <- merge(df1, df2, by = c("country", "year"))

# View the merged dataset
print(merged_df)
```

```
##   country year x1 x2
## 1  Canada 2000 21  2
## 2  Canada 2001 25  6
## 3   Japan 2000 29  6
## 4   Japan 2001 30 30
## 5     USA 2000 23  7
## 6     USA 2001 28 12
```