**C11**

**# Exercise 1:** **Working With Data Frames (review)**

**# Install devtools package: allows installations from GitHub**

install.packages("devtools")

**# Install "fueleconomy" dataset from GitHub**

devtools::install\_github("hadley/fueleconomy")

**# Use the `libary()` function to load the "fueleconomy" package**

library(fueleconomy)

**# You should now have access to the `vehicles` data frame. To inspect:** View(vehicles)

**# Select the different manufacturers (makes) of the cars in this data set.**

# Save this vector in a variable makes <- vehicles$make

**# Use the `unique()` function to determine how many different car manufacturers are represented by the data set**

length(unique(makes))

**# Filter the data set for vehicles manufactured in 1997**

cars\_1997 <- vehicles[vehicles$year == 1997, ]

**# Arrange the 1997 cars by highway (`hwy`) gas milage**

# Hint: use the `order()` function to get a vector of indices in order by value

# See also: https://www.r-bloggers.com/r-sorting-a-data-frame-by-the-contents-of-a-column/

cars\_1997 <- cars\_1997[order(cars\_1997$hwy), ]

**# Mutate the 1997 cars data frame to add a column `average` that has the average gas milage (between city and highway mpg) for each car**

cars\_1997$average <- (cars\_1997$hwy + cars\_1997$cty) / 2

**# Filter the whole vehicles data set for 2-Wheel Drive vehicles that get more than 20 miles/gallon in the city. Save this new data frame in a variable.**

two\_wheel\_20\_mpg <- vehicles[vehicles$drive == "2-Wheel Drive" & vehicles$cty > 20, ]

**# Of the above vehicles, what is the vehicle ID of the vehicle with the worst hwy mpg?**

# Hint: filter for the worst vehicle, then select its ID.

worst\_hwy <- two\_wheel\_20\_mpg$id[two\_wheel\_20\_mpg$hwy == min(two\_wheel\_20\_mpg$hwy)]

**# Write a function that takes a `year\_choice` and a `make\_choice` as parameters, and returns the vehicle model that gets the most hwy miles/gallon of vehicles of that make in that year.**

# You'll need to filter more (and do some selecting)!

make\_year\_filter <- function(make\_choice, year\_choice) {

filtered <- vehicles[vehicles$make == make\_choice & vehicles$year == year\_choice, ]

filtered[filtered$hwy == max(filtered$hwy), "model"] }

**# What was the most efficient Honda model of 1995?**

make\_year\_filter("Honda", 1995)

**# Exercise 2: Working With `dplyr`**

# Note that this exercise repeats the analysis from Exercise 1, but should be performed using `dplyr` (do not directly access or manipulate the data frames)

**# Install and load the "fueleconomy" package**

# install.packages("devtools")

# devtools::install\_github("hadley/fueleconomy")

library(fueleconomy)

**# Install and load the "dplyr" library**

install.packages("dplyr")

library("dplyr")

**# Select the different manufacturers (makes) of the cars in this data set.**

makes <- select(vehicles, make)

**# Use the `distinct()` function to determine how many different car manufacturers are represented by the data set**

nrow(distinct(vehicles, make))

length(unique(makes$make)) # without deplyr

**# Filter the data set for vehicles manufactured in 1997**

cars\_1997 <- filter(vehicles, year == 1997)

**# Arrange the 1997 cars by highway (`hwy`) gas milage**

cars\_1997 <- arrange(cars\_1997, hwy)

**# Mutate the 1997 cars data frame to add a column `average` that has the average gas milage (between city and highway mpg) for each car**

cars\_1997 <- mutate(cars\_1997, average = (hwy + cty) / 2)

**# Filter the whole vehicles data set for 2-Wheel Drive vehicles that get more**

**# than 20 miles/gallon in the city. Save this new data frame in a variable.**

two\_wheel\_20\_mpg <- filter(vehicles, drive == "2-Wheel Drive", cty > 20)

**# Of the above vehicles, what is the vehicle ID of the vehicle with the worst hwy mpg?**

# Hint: filter for the worst vehicle, then select its ID.

filtered <- filter(two\_wheel\_20\_mpg, hwy == min(hwy))

worst\_hwy <- select(filtered, id)

**# Write a function that takes a `year\_choice` and a `make\_choice` as parameters, and returns the vehicle model that gets the most hwy miles/gallon of vehicles of that make in that year.**

# You'll need to filter more (and do some selecting)!

make\_year\_filter <- function(make\_choice, year\_choice) {

filtered <- filter(vehicles, make == make\_choice, year == year\_choice)

filtered <- filter(filtered, hwy == max(hwy))

selected <- select(filtered, model)

selected }

**# Exercise 3: Using The Pipe Operator**

**# Install (if needed) and load the "dplyr" library** install.packages("dplyr")

library("dplyr")

**# Install (if needed) and load the "fueleconomy" package** install.packages("devtools")

# devtools::install\_github("hadley/fueleconomy") library("fueleconomy")

**# Which 2015 Acura model has the best hwy MGH? (Use dplyr, but without method chaining or pipes--use temporary variables!)**

acuras <- filter(vehicles, make == "Acura", year == 2015)

best\_acura <- filter(acuras, hwy == max(hwy))

best\_model <- select(best\_acura, model)

**# Which 2015 Acura model has the best hwy MPG? (Use dplyr, nesting functions)**

best\_model <- select(filter(

filter(vehicles, make == “Acura”, year == 2015), hwy == max(hwy)

), model)

**# Which 2015 Acura model has the best hwy MPG? (Use dplyr and the pipe operator)**

best\_model <- filter(vehicles, make == "Acura", year == 2015) %>%

filter(hwy == max(hwy)) %>%

select(model)

**### Bonus**

**# Write 3 functions, one for each approach. Then, test how long it takes to perform each one 1000 times**

**# Without chaining**

temp\_vars\_best\_model <- function() {

acuras <- filter(vehicles, make == "Acura", year == 2015)

best.acura <- filter(acuras, hwy == max(hwy))

best.model <- select(best\_acura, model) }

**# Nested functions**

nested\_best\_model <- function() {

best\_model <- select(

filter(

filter(vehicles, make == "Acura", year == 2015), hwy == max(hwy)

), model

)

**}**

**# Pipe operator**

pipe\_best\_model <- function() {

best\_model <- filter(vehicles, make == "Acura", year == 2015) %>%

filter(hwy == max(hwy)) %>%

select(model)

}

**# Pretty similar results; use which is most readable!**

system.time(for (i in 1:1000) temp\_vars\_best\_model())

system.time(for (i in 1:1000) nested\_best\_model())

system.time(for (i in 1:1000) pipe\_best\_model())

**# Exercise 4:** **Practicing with dplyr**

In this exercise, you'll practice using the `dplyr` package to ask questions of a data set: specifically, airline on-time data for flights departing NYC in 2013 from the [`nycflights13`](https://cran.r-project.org/web/packages/nycflights13/index.html) package.

**# Install the `"nycflights13"` package. Load (`library()`) the package.**

**# You'll also need to load `dplyr`**

install.packages("nycflights13")

library(nycflights13)

library(dplyr)

**# The data frame `flights` should now be accessible to you.**

**# Use functions to inspect it: how many rows and columns does it have?**

**# What are the names of the columns?**

**# Use `??flights` to search for documentation on the data set (for what the columns represent)**

nrow(flights)

ncol(flights)

colnames(flights)

?flights

**# Use `dplyr` to give the data frame a new column that is the amount of time gained or lost while flying (that is: how much of the delay arriving occurred during flight, as opposed to before departing).**

flights <- mutate(flights, gain\_in\_air = arr\_delay - dep\_delay)

**# Use `dplyr` to sort your data frame in descending order by the column you just created. Remember to save this as a variable (or in the same one!)**

flights <- arrange(flights, desc(gain\_in\_air))

View(head(flights))

**# For practice, repeat the last 2 steps in a single statement using the pipe operator. You can clear your environmental variables to "reset" the data frame**

flights <- flights %>% mutate(gain\_in\_air = arr\_delay - dep\_delay) %>% arrange(desc(gain\_in\_air))

**# Make a histogram of the amount of time gained using the `hist()` function**

hist(flights$gain\_in\_air)

**# On average, did flights gain or lose time?**

**# Note: use the `na.rm = TRUE` argument to remove NA values from your aggregation**

mean(flights$gain\_in\_air, na.rm = TRUE) # Gained 5 minutes!

**# Create a data.frame of flights headed to SeaTac ('SEA'), only including the origin,**

**destination, and the "gain\_in\_air" column you just created**

to\_sea <- flights %>% select(origin, dest, gain\_in\_air) %>% filter(dest == "SEA")

**# On average, did flights to SeaTac gain or loose time?**

mean(to\_sea$gain\_in\_air, na.rm = TRUE) # Gained 11 minutes!

**# Consider flights from JFK to SEA. What was the average, min, and max air time**

**# of those flights? Bonus: use pipes to answer this question in one statement**

**# (without showing any other data)!**

filter(flights, origin == "JFK", dest == "SEA") %>%

summarize(

avg\_air\_time = mean(air\_time, na.rm = TRUE),

max\_air\_time = max(air\_time, na.rm = TRUE),

min\_air\_time = min(air\_time, na.rm = TRUE)

)

**# Exercise 5: dplyr grouped operations**

**# Install the `"nycflights13"` package. Load (`library()`) the package.**

**# You'll also need to load `dplyr`**

# install.packages("nycflights13") # should be done already

library("nycflights13")

library("dplyr")

**# What was the average departure delay in each month?**

**# Save this as a data frame `dep\_delay\_by\_month`**

**# Hint: you'll have to perform a grouping operation then summarizing your data**

dep\_delay\_by\_month <- flights %>%

group\_by(month) %>%

summarize(delay = mean(dep\_delay, na.rm = TRUE))

dep\_delay\_by\_month

**# Which month had the greatest average departure delay?**

filter(dep\_delay\_by\_month, delay == max(delay)) %>% select(month)

**# If your above data frame contains just two columns (e.g., "month", and "delay"**

**# in that order), you can create a scatterplot by passing that data frame to the**

# `plot()` function

plot(dep\_delay\_by\_month)

**# To which destinations were the average arrival delays the highest?**

**# Hint: you'll have to perform a grouping operation then summarize your data**

**# You can use the `head()` function to view just the first few rows**

arr\_delay\_by\_month <- flights %>%

group\_by(dest) %>%

summarise(delay = mean(arr\_delay, na.rm = TRUE)) %>%

arrange(-delay)

head(arr\_delay\_by\_month)

**# You can look up these airports in the `airports` data frame!**

filter(airports, faa == arr\_delay\_by\_month$dest[1]) # for example

**# Which city was flown to with the highest average speed?**

city\_fasted\_speed <- flights %>%

mutate(speed = distance / air\_time \* 60) %>%

group\_by(dest) %>%

summarise(avg\_speed = mean(speed, na.rm = TRUE)) %>%

filter(avg\_speed == max(avg\_speed, na.rm = TRUE))

city\_fasted\_speed

**# Exercise 6:** **dplyr join operations**

In this exercise, you'll practice using the dplyr's join functions on the [`nycflights13`](https://cran.r-project.org/web/packages/nycflights13/index.html) data set.

**# Install the `"nycflights13"` package. Load (`library()`) the package.**

**# You'll also need to load `dplyr`**

# install.packages("nycflights13") # should be done already

library("nycflights13")

library("dplyr")

**# Create a dataframe of the average arrival delays for each \_destination\_, then**

**# use `left\_join()` to join on the "airports" dataframe, which has the airport info**

**# Which airport had the largest average arrival delay?**

largest\_arrival\_delay <- flights %>%

group\_by(dest) %>%

summarise(avg\_delay = mean(arr\_delay, na.rm = TRUE)) %>%

mutate(faa = dest) %>%

left\_join(airports, by = "faa") %>%

filter(avg\_delay == max(avg\_delay, na.rm = TRUE))

largest\_arrival\_delay

**# Create a dataframe of the average arrival delay for each \_airline\_, then use**

**# `left\_join()` to join on the "airlines" dataframe**

**# Which airline had the smallest average arrival delay?**

smallest\_airline\_delay <- flights %>%

group\_by(carrier) %>%

summarise(avg\_delay = mean(arr\_delay, na.rm = TRUE)) %>%

left\_join(airlines, by = "carrier") %>%

filter(avg\_delay == max(avg\_delay, na.rm = TRUE))

smallest\_airline\_delay

**# Exercise 7:** **using dplyr on external data**

In this exercise, you'll practice using the dplyr's library to work with an external data set, specifically NBA team statistics from the 2015-2016 season.

**# Load the `dplyr` library**

library(dplyr)

# Use the `read.csv()` function to read in the included data set. Remember to save it as a variable.

team\_data <- read.csv("data/nba\_teams\_2016.csv", stringsAsFactors = FALSE)

**# View the data frame you loaded, and get some basic information about the number of rows/columns.**

**# Note the "X" preceding some of the column titles as well as the "\*" following**

**# the names of teams that made it to the playoffs that year.**

View(team\_data)

**# Add a column that gives the turnovers to steals ratio (TOV / STL) for each team**

team\_data <- mutate(team\_data, Ratio = TOV / STL)

**# Sort the teams from lowest turnover/steal ratio to highest**

**# Which team has the lowest turnover/steal ratio?**

team\_data %>%

filter(Ratio == min(Ratio)) %>%

select(Team)

**# Using the pipe operator, create a new column of assists per game (AST / G)**

**# AND sort the data.frame by this new column in descending order.**

team\_data <- mutate(team\_data, ASTGM = AST / G) %>% arrange(-ASTGM)

**# Create a data frame called `good\_offense` of teams that scored more than 8700 points (PTS) in the season**

good\_offense <- filter(team\_data, PTS > 8700)

**# Create a data frame called `good\_defense` of teams that had more than 470 blocks (BLK)**

good\_defense <- filter(team\_data, BLK > 470)

**# Create a data frame called `offense\_stats` that only shows offensive**

**# rebounds (ORB), field-goal % (FG.), and assists (AST) along with the team name.**

offense\_stats <- select(team\_data, Team, ORB, FG., AST)

**# Create a data frame called `defense\_stats` that only shows defensive**

# rebounds (DRB), steals (STL), and blocks (BLK) along with the team name.

defense\_stats <- select(team\_data, Team, DRB, STL, BLK)

**# Create a function called `better\_shooters` that takes in two teams and returns**

# a data frame of the team with the better field-goal percentage. Include the

# team name, field-goal percentage, and total points in your resulting data frame

better\_shooters <- function(team1, team2) {

better\_team <- filter(team\_data, Team %in% c(team1, team2)) %>%

filter(FG. == max(FG.)) %>%

select(Team, FG., PTS)

better\_team

}

**# Call the function on two teams to compare them (remember the `\*` if needed)**

better.shooter <- BetterShooters("ggpGolden State Warriors\*", "Cleveland Cavaliers\*")

better.shooter

**# Exercise 8:** **Exploring data sets**

In this exercise, you'll practice using dplyr to explore a data set, specifically information about *\_Pulitzer Prize Winning Newspapers\_*. The dataset comes from [Five Thirty Eight.](https://github.com/fivethirtyeight/data/blob/master/pulitzer/pulitzer-circulation-data.csv). More information on the [Pulitzer Prize](http://www.pulitzer.org/) can be found on their website.

**# Load the `dplyr` library**

library(dplyr)

**# Read in the data (from `data/pupulitzer-circulation-data.csv`). Remember to**

**# not treat strings as factors!**

pulitzer <- read.csv("data/pulitzer-circulation-data.csv", stringsAsFactors = FALSE)

**# View in the data set. Start to understand what the data set contains**

View(pulitzer)

**# Print out the names of the columns for reference**

colnames(pulitzer)

**# Use the 'str()' function to also see what types of values are contained in each column (you're looking at the second column after the `:`)**

**# Did any value type surprise you? Why do you think they are that type?**

str(pulitzer)

**# Add a column to the data frame called 'Pulitzer.Prize.Change` that contains the difference in the number of times each paper was a winner or finalist**

**# (hereafter "winner") during 2004-2014 and during 1990-2003**

mutate(pulitzer,

Pulitzer.Prize.Change =

Pulitzer.Prize.Winners.and.Finalists..2004.2014 -

Pulitzer.Prize.Winners.and.Finalists..1990.2003

)

**# What was the name of the publication that has the most winners between 2004-2014?**

filter(pulitzer, max(Pulitzer.Prize.Winners.and.Finalists..2004.2014) ==

Pulitzer.Prize.Winners.and.Finalists..2004.2014) %>%

select(Newspaper)

**# Which publication with at least 5 winners between 2004-2014 had the biggest decrease(negative) in daily circulation numbers?**

filter(pulitzer, Pulitzer.Prize.Winners.and.Finalists..2004.2014 >= 5) %>%

filter(min(Change.in.Daily.Circulation..2004.2013) == Change.in.Daily.Circulation..2004.2013) %>%

select(Newspaper)

**C12**

**# Exercise 1: analyzing avocado sales with the `tidyr` package**

The data has the following columns:

- \*\*Date\*\*: Week of sales data.

- \*\*AveragePrice\*\*: Average price during that week.

- \*\*Total Volume\*\*: Total volume of avocados sold (includes price lookup codes `4046`, `4770`, `4225` \*\*and\*\* others).

- \*\*4046\*\*: Total volume of small Hass avocados sold.

- \*\*4225\*\*: Total volume of large Hass avocados sold.

- \*\*4770\*\*: Total volume of extra-large Hass avocados sold.

- \*\*Total Bags\*\*: Total bags of avocados sold.

- \*\*Small Bags\*\*: Number of small bags of avocados sold.

- \*\*Large Bags\*\*: Number of large bags of avocados sold.

- \*\*XLarge Bags\*\*: Number of extra large bags of avocados sold.

- \*\*type\*\*: Type of avocados sold.

- \*\*year\*\*: Year of sale.

**# Load necessary packages (`tidyr`, `dplyr`, and `ggplot2`)**

library("tidyr")

library("dplyr")

library("ggplot2")

**# Set your working directory using the RStudio menu:**

# Session > Set Working Directory > To Source File Location

**# Load the `data/avocado.csv` file into a variable `avocados`**

**# Make sure strings are \*not\* read in as factors**

avocados <- read.csv("data/avocado.csv", stringsAsFactors = F)

**# To tell R to treat the `Date` column as a date (not just a string)**

**# Redefine that column as a date using the `as.Date()` function**

**# (hint: use the `mutate` function)**

avocados <- avocados %>%

mutate(Date = as.Date(Date))

**# The file had some uninformative column names, so rename these columns:**

**# `X4046` to `small\_haas`**

**# `X4225` to `large\_haas`**

**# `X4770` to `xlarge\_haas`**

avocados <- avocados %>%

rename(small\_haas = X4046, large\_haas = X4225, xlarge\_haas = X4770)

**# The data only has sales for haas avocados. Create a new column `other\_avos`**

**# that is the Total.Volume minus all haas avocados (small, large, xlarge)**

avocados <- avocados %>%

mutate(other\_avos = Total.Volume - small\_haas - large\_haas - xlarge\_haas)

**# To perform analysis by avocado size, create a dataframe `by\_size` that has**

**# only `Date`, `other\_avos`, `small\_haas`, `large\_haas`, `xlarge\_haas`**

by\_size <- avocados %>%

select(Date, other\_avos, small\_haas, large\_haas, xlarge\_haas)

**# In order to visualize this data, it needs to be reshaped. The four columns `other\_avos`, `small\_haas`, `large\_haas`, `xlarge\_haas` need to be \*\*gathered\*\* together into a single column called `size`. The volume of sales (currently stored in each column) should be stored in a new column called `volume`.**

**# Create a new dataframe `size\_gathered` by passing the `by\_size` data frame to the `gather()` function. `size\_gathered` will only have 3 columns: `Date`, `size`, and `volume`.**

size\_gathered <- by\_size %>%

gather(key = size, value = volume, -Date)

**# Using `size\_gathered`, compute the average sales volume of each size**

**# (hint, first `group\_by` size, then compute using `summarize`)**

average\_sales <- size\_gathered %>%

group\_by(size) %>%

summarise(mean\_volume = mean(volume))

# This shape also facilitates the visualization of sales over time

# (how to write this code is covered in Chapter 16)

ggplot(size\_gathered) +

geom\_smooth(mapping = aes(x = Date, y = volume, col = size), se = F)

# We can also investigate sales by avocado type (conventional, organic).

# Create a new data frame `by\_type` by grouping the `avocados` dataframe by

# `Date` and `type`, and calculating the sum of the `Total.Volume` for that type

# in that week (resulting in a data frame with 2 rows per week).

by\_type <- avocados %>%

group\_by(Date, type) %>%

summarise(volume = sum(Total.Volume))

# To make a (visual) comparison of conventional versus organic sales, you

# need to \*\*spread\*\* out the `type` column into two different columns. Create a

# new data frame `by\_type\_wide` by passing the `by\_type` data frame to the `spread()` function!

by\_type\_wide <- by\_type %>%

spread(key = type, value = volume)

# Now you can create a scatterplot comparing conventional to organic sales!

ggplot(by\_type\_wide) +

geom\_point(mapping = aes(x = conventional, y = organic, color = Date))