##############################################################

### Title: Bootcamp example code

### Author: Magdalena Bennett

### Date Created: 08/23/2023

### Last edit: [08/23/2023] - Created code

##############################################################

#Clear memory

rm(list = ls())

#Clear the console

cat("\014")

#Turn off scientific notation (turn back on with 0)

options(scipen = 999)

# Load packages

library(tidyverse) #includes dplyr and ggplot2!

# If there is a package you don't have installed, you can use install.packages("tidyverse")

# Only run once! (no need to install packages every time you run your code)

# Load data (this is loading data directly from Github)

sales = read.csv("https://raw.githubusercontent.com/maibennett/sta235/main/exampleSite/content/bootcamp/data/US\_Regional\_Sales\_Data.csv")

## Inspecting your data

# Exercise 1: Let's explore the data. How many variables and observations do we have? What type of variables do we have?

## We can see in the environment pane that we have 7991 obs and 16 variables. Most of them are character variables (chr), but some are numeric (int and num).

# Exercise 2: Install the package vtable, load it, and run the code sumtable(sales). What do you get? Use the ?vtable to see the options for this function.

#install.packages(vtable)

library(vtable)

?sumtable

sumtable(sales)

## Data wrangling

# Exercise 1: Unit cost and unit price should be numeric. Let's change this! (hint: you can use the function gsub() to replace "," for "", and as.numeric() to transform a variable!).

## Keep the same names for the variables and the dataset.

sales = sales %>% mutate(unit\_cost = gsub(",", "", unit\_cost),

unit\_price = gsub(",","", unit\_price))

sales = sales %>% mutate(unit\_cost = as.numeric(unit\_cost),

unit\_price = as.numeric(unit\_price))

# Exercise 2: What are the different values for the sales channel in this dataset? Use the function table() to see!

## Create a new dataset for in-store and online sales. Call it "sales\_min". How many variables do we have?

sales %>% select(sales\_channel) %>% table()

sales\_min = sales %>% filter(sales\_channel == "In-Store" | sales\_channel == "Online")

# Exercise 3: Use the original dataset "sales", and create a new variable called "minority",

## which takes the value of 1 if the sales channel is in-store or online, and 0 in another case.

sales = sales %>% mutate(minority = ifelse(sales\_channel == "In-Store" | sales\_channel == "Online", 1, 0))

# Exercise 4: What is the average price for sales made through a minority channel vs a non-minority channel?

sales %>% group\_by(minority) %>% summarize(unit\_price = mean(unit\_price))

## Plotting data!

# Exercise 1: Create a scatter plot between unit cost (x axis) and unit price (y axis)

ggplot(data = sales, aes(x = unit\_cost, y = unit\_price)) + geom\_point()

# Exercise 2: Now, let's make that plot pretty. Use theme\_minimal() to get rid of the grey background. Color the points with the color "deepskyblue3",

## and change the axis titles to something more informative (e.g. Unit price ($)). This can be done with xlab() and ylab().

ggplot(data = sales, aes(x = unit\_cost, y = unit\_price)) + geom\_point(color = "deepskyblue3") + theme\_minimal() +

xlab("Unit Cost ($)") + ylab("Unit Price ($)")

# Exercise 3: Using the same code as before, now we want to color observations from the minority sales channel in one color, and the non-minority in another color.

## Write some code that does that (e.g. you will need to change your aesthetics!)

ggplot(data = sales, aes(x = unit\_cost, y = unit\_price, color = factor(minority))) + geom\_point() + theme\_minimal() +

xlab("Unit Cost ($)") + ylab("Unit Price ($)")

# Exercise 4: Finally, using the same code as in exercise 2, include a regression line in this plot using geom\_smooth().

ggplot(data = sales, aes(x = unit\_cost, y = unit\_price)) + geom\_point(color = "deepskyblue3") + theme\_minimal() +

xlab("Unit Cost ($)") + ylab("Unit Price ($)") + geom\_smooth(method = "lm")

## Regressions

# Let's load a new dataset: The Gapminder

gapminder = read.csv("https://raw.githubusercontent.com/maibennett/sta235/main/exampleSite/content/bootcamp/data/gapminder.csv")

# Exercise 1: What type of data do we have?

## You can see there is numeric data, but also factor (also referred to as categorical variables).

## Factors are useful because they enter a regression as individual dummies

# Exercise 2: Transform population into millions (divide pop by 10^6), and then regress life expectancy on gdp per capita and population. What do you obtain?

gapminder = gapminder %>% mutate(pop = pop/10^6)

lm1 = lm(lifeExp ~ gdpPercap + pop, data = gapminder)

summary(lm1)

# Exercise 3: Include now continent in the previous regression. Do your results change? How does it look when you include a factor variable in a regression?

lm2 = lm(lifeExp ~ gdpPercap + pop + continent, data = gapminder)

summary(lm2)

## Bringing everything together

# Exercise 1: Create a new variable called gdpPercap\_log, which is the logarithm of the GDP per capita. Now plot life expectancy against the log(GDP per capita),

## and describe the relationship.

gapminder = gapminder %>% mutate(gdpPercap\_log = log(gdpPercap))

ggplot(data = gapminder, aes(x = gdpPercap\_log, y = lifeExp)) +

geom\_point() + theme\_minimal()

# Exercise 2: Using the same plot as before, now color the points by continent and make the size proportional by population (in millions).

ggplot(data = gapminder, aes(x = gdpPercap\_log, y = lifeExp, size = pop, color = continent)) +

geom\_point() + theme\_minimal()

# Exercise 3: Do the same thing as before (exercise 2), but only for Europe!

gapminder %>% filter(continent == "Europe") %>%

ggplot(data = ., aes(x = gdpPercap\_log, y = lifeExp, size = pop)) +

geom\_point() + theme\_minimal()

# Exercise 4: Finally, run a regression that helps you estimate the association between life expectancy and GDP per capita, conditional on population,

## for the year 2007 and then, another regression for the year 1982.

lm3 = gapminder %>% filter(year == 2007) %>% lm(lifeExp ~ gdpPercap + pop, data = .)

summary(lm3)

lm4 = gapminder %>% filter(year == 1982) %>% lm(lifeExp ~ gdpPercap + pop, data = .)

summary(lm4)