Final Project

```
# import library
library(readr)
```

```
Warning: package 'readr' was built under R version 4.3.3
```

```
# reading in the dataframe we are working with
obesity <- read_csv("obes.csv")
```

```
Rows: 284142 Columns: 31
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (25): RowId, LocationAbbr, LocationDesc, Datasource, Class, Topic, Quest...
dbl  (6): YearStart, YearEnd, Data_Value, Data_Value_Alt, Low_Confidence_Lim...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# show first 6 rows
head(obesity)
```

```
# A tibble: 6 × 31
  RowId       YearStart YearEnd LocationAbbr LocationDesc Datasource Class Topic
  <chr>           <dbl>   <dbl> <chr>        <chr>        <chr>      <chr> <chr>
1 BRFSS~2022…      2022    2022 PA           Pennsylvania BRFSS      Ment… Freq…
2 BRFSS~2022…      2022    2022 SD           South Dakota BRFSS      Ment… Freq…
3 BRFSS~2022…      2022    2022 ID           Idaho        BRFSS      Ment… Freq…
4 BRFSS~2022…      2022    2022 MD           Maryland     BRFSS      Ment… Freq…
5 BRFSS~2022…      2022    2022 WI           Wisconsin    BRFSS      Ment… Freq…
6 BRFSS~2022…      2022    2022 IA           Iowa         BRFSS      Ment… Freq…
# ℹ 23 more variables: Question <chr>, Data_Value_Unit <chr>,
#   DataValueTypeID <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Alt <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Low_Confidence_Limit <dbl>,
#   High_Confidence_Limit <dbl>, StratificationCategory1 <chr>,
#   Stratification1 <chr>, StratificationCategory2 <chr>,
#   Stratification2 <chr>, Geolocation <chr>, ClassID <chr>, TopicID <chr>, …
```

```
# import library
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.3.3


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
```

```
# filtering the dataframe so it only displays rows that are in the years 2015-2019
obes <- filter(obesity, YearStart %in% c(2015, 2016, 2017, 2018, 2019))
obes
```

```
# A tibble: 171,627 × 31
   RowId      YearStart YearEnd LocationAbbr LocationDesc Datasource Class Topic
   <chr>          <dbl>   <dbl> <chr>        <chr>        <chr>      <chr> <chr>
 1 BRFSS~201…      2019    2022 NY           New York     BRFSS      Cogn… Subj…
 2 BRFSS~201…      2019    2022 UT           Utah         BRFSS      Cogn… Subj…
 3 BRFSS~201…      2019    2022 WA           Washington   BRFSS      Cogn… Subj…
 4 BRFSS~201…      2019    2022 WA           Washington   BRFSS      Cogn… Subj…
 5 BRFSS~201…      2019    2022 ND           North Dakota BRFSS      Cogn… Func…
 6 BRFSS~201…      2019    2022 OH           Ohio         BRFSS      Cogn… Func…
 7 BRFSS~201…      2019    2022 NV           Nevada       BRFSS      Cogn… Func…
 8 BRFSS~201…      2019    2022 OH           Ohio         BRFSS      Cogn… Func…
 9 BRFSS~201…      2019    2022 OR           Oregon       BRFSS      Cogn… Func…
10 BRFSS~201…      2019    2022 WA           Washington   BRFSS      Cogn… Need…
# ℹ 171,617 more rows
# ℹ 23 more variables: Question <chr>, Data_Value_Unit <chr>,
#   DataValueTypeID <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Alt <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Low_Confidence_Limit <dbl>,
#   High_Confidence_Limit <dbl>, StratificationCategory1 <chr>,
#   Stratification1 <chr>, StratificationCategory2 <chr>, …
```

```
# filters the dataframe to only include rows that have a Topic of Obesity
obes <- filter(obes, Topic == 'Obesity')
obes
```

```
# A tibble: 6,916 × 31
   RowId      YearStart YearEnd LocationAbbr LocationDesc Datasource Class Topic
   <chr>          <dbl>   <dbl> <chr>        <chr>        <chr>      <chr> <chr>
 1 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 2 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 3 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 4 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 5 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 6 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 7 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 8 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
 9 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
```

```
10 BRFSS~201…      2019    2019 AK           Alaska       BRFSS      Nutr… Obes…
# ℹ 6,906 more rows
# ℹ 23 more variables: Question <chr>, Data_Value_Unit <chr>,
#   DataValueTypeID <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Alt <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Low_Confidence_Limit <dbl>,
#   High_Confidence_Limit <dbl>, StratificationCategory1 <chr>,
#   Stratification1 <chr>, StratificationCategory2 <chr>, …
```

```
# Arrange the dataframe by the LocationDesc column to sort the data alphabetically by location des
obes1 <- arrange(obes, LocationDesc)
obes1
```

```
# A tibble: 6,916 × 31
   RowId      YearStart YearEnd LocationAbbr LocationDesc Datasource Class Topic
   <chr>          <dbl>   <dbl> <chr>        <chr>        <chr>      <chr> <chr>
 1 BRFSS~201…      2019    2019 AL           Alabama      BRFSS      Nutr… Obes…
 2 BRFSS~201…      2019    2019 AL           Alabama      BRFSS      Nutr… Obes…
 3 BRFSS~201…      2016    2016 AL           Alabama      BRFSS      Nutr… Obes…
 4 BRFSS~201…      2015    2015 AL           Alabama      BRFSS      Nutr… Obes…
 5 BRFSS~201…      2016    2016 AL           Alabama      BRFSS      Nutr… Obes…
 6 BRFSS~201…      2015    2015 AL           Alabama      BRFSS      Nutr… Obes…
 7 BRFSS~201…      2018    2018 AL           Alabama      BRFSS      Nutr… Obes…
 8 BRFSS~201…      2016    2016 AL           Alabama      BRFSS      Nutr… Obes…
 9 BRFSS~201…      2018    2018 AL           Alabama      BRFSS      Nutr… Obes…
10 BRFSS~201…      2019    2019 AL           Alabama      BRFSS      Nutr… Obes…
# ℹ 6,906 more rows
# ℹ 23 more variables: Question <chr>, Data_Value_Unit <chr>,
#   DataValueTypeID <chr>, Data_Value_Type <chr>, Data_Value <dbl>,
#   Data_Value_Alt <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Low_Confidence_Limit <dbl>,
#   High_Confidence_Limit <dbl>, StratificationCategory1 <chr>,
#   Stratification1 <chr>, StratificationCategory2 <chr>, …
```

```
# select only the columns that are relevant to our analysis
obes2 <- obes1 %>%
  select(YearStart, LocationDesc, Question, Data_Value)
obes2
```

```
# A tibble: 6,916 × 4
   YearStart LocationDesc Question                                    Data_Value
       <dbl> <chr>        <chr>                                           <dbl>
 1      2019 Alabama      Percentage of older adults who are current…      NA
 2      2019 Alabama      Percentage of older adults who are current…      NA
 3      2016 Alabama      Percentage of older adults who are current…      44.2
 4      2015 Alabama      Percentage of older adults who are current…      39
 5      2016 Alabama      Percentage of older adults who are current…      NA
 6      2015 Alabama      Percentage of older adults who are current…      38
 7      2018 Alabama      Percentage of older adults who are current…      NA
```

```
 8      2016 Alabama      Percentage of older adults who are current…       39.6
 9      2018 Alabama      Percentage of older adults who are current…       42.3
10      2019 Alabama      Percentage of older adults who are current…       31.7
# ℹ 6,906 more rows
```

```
# Arrange the dataframe by the YearStart column to sort the data numerically by year
obes3 <- arrange(obes2, YearStart)
obes3
```

```
# A tibble: 6,916 × 4
   YearStart LocationDesc Question                                    Data_Value
       <dbl> <chr>        <chr>                                           <dbl>
 1      2015 Alabama      Percentage of older adults who are current…       39
 2      2015 Alabama      Percentage of older adults who are current…       38
 3      2015 Alabama      Percentage of older adults who are current…       NA
 4      2015 Alabama      Percentage of older adults who are current…       50.9
 5      2015 Alabama      Percentage of older adults who are current…       NA
 6      2015 Alabama      Percentage of older adults who are current…       30
 7      2015 Alabama      Percentage of older adults who are current…       27.3
 8      2015 Alabama      Percentage of older adults who are current…       36.2
 9      2015 Alabama      Percentage of older adults who are current…       43.6
10      2015 Alabama      Percentage of older adults who are current…       NA
# ℹ 6,906 more rows
```

```
# Drop na values and aggregate each row using means so that each state has 1 row for each year
obes3 <- na.omit(obes3)
obes3 <- obes3 %>% group_by(YearStart, LocationDesc) %>% summarize(Data_Value = mean(Data_Value))
```

```
`summarise()` has grouped output by 'YearStart'. You can override using the
`.groups` argument.
```

```
obes3
```

```
# A tibble: 290 × 3
# Groups:   YearStart [5]
   YearStart LocationDesc         Data_Value
       <dbl> <chr>                     <dbl>
 1      2015 Alabama                    38.1
 2      2015 Alaska                     34.5
 3      2015 Arizona                    30.6
 4      2015 Arkansas                   35.8
 5      2015 California                 26.0
 6      2015 Colorado                   24.3
 7      2015 Connecticut                29.8
 8      2015 Delaware                   32.2
 9      2015 District of Columbia       29.6
10      2015 Florida                    30.3
# ℹ 280 more rows
```

```
obes3 <- obes3 %>%
  rename(
    Date = YearStart, # Rename the YearStart column to 'Date' for clarity
    State = LocationDesc) # Rename the 'LocationDesc column to 'State' for clarity
obes3
```

```
# A tibble: 290 × 3
# Groups:   Date [5]
    Date State                Data_Value
   <dbl> <chr>                     <dbl>
 1  2015 Alabama                    38.1
 2  2015 Alaska                     34.5
 3  2015 Arizona                    30.6
 4  2015 Arkansas                   35.8
 5  2015 California                 26.0
 6  2015 Colorado                   24.3
 7  2015 Connecticut                29.8
 8  2015 Delaware                   32.2
 9  2015 District of Columbia       29.6
10  2015 Florida                    30.3
# ℹ 280 more rows
```

```
# read in second dataframe
ppi <- read_csv("ProductPriceIndex.csv")
```

```
Rows: 15766 Columns: 8
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr  (7): productname, farmprice, atlantaretail, chicagoretail, losangelesre...
date (1): date

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# show first 5 rows
head(ppi)
```

```
# A tibble: 6 × 8
  productname  date       farmprice atlantaretail chicagoretail losangelesretail
  <chr>        <date>     <chr>     <chr>         <chr>         <chr>
1 Strawberries 2019-05-19 $1.16     $2.23         $1.70         $1.99
2 Romaine Let… 2019-05-19 $0.35     $1.72         $2.00         $1.69
3 Red Leaf Le… 2019-05-19 $0.32     $1.84         $1.84         $1.69
4 Potatoes     2019-05-19 $1.50     $5.32         $5.14         $3.99
5 Oranges      2019-05-19 $0.41     $1.42         $1.45         $1.34
6 Iceberg Let… 2019-05-19 $0.35     $1.39         $1.46         $1.69
# ℹ 2 more variables: newyorkretail <chr>, averagespread <chr>
```

```
# import library
library(lubridate)
```

```
Warning: package 'lubridate' was built under R version 4.3.3


Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```
# Create a new column 'Year' in the 'ppi' dataframe
# ymd(ppi$date): This function converts the 'date' column in 'ppi' to Date format (Year-Month-Day
ppi$Year <- year(ymd(ppi$date))

# year(): This function extracts the year from the Date object created.
ppi
```

```
# A tibble: 15,766 × 9
   productname date       farmprice atlantaretail chicagoretail losangelesretail
   <chr>       <date>     <chr>     <chr>         <chr>         <chr>
 1 Strawberri… 2019-05-19 $1.16     $2.23         $1.70         $1.99
 2 Romaine Le… 2019-05-19 $0.35     $1.72         $2.00         $1.69
 3 Red Leaf L… 2019-05-19 $0.32     $1.84         $1.84         $1.69
 4 Potatoes    2019-05-19 $1.50     $5.32         $5.14         $3.99
 5 Oranges     2019-05-19 $0.41     $1.42         $1.45         $1.34
 6 Iceberg Le… 2019-05-19 $0.35     $1.39         $1.46         $1.69
 7 Green Leaf… 2019-05-19 $0.31     $1.72         $1.70         $1.69
 8 Celery      2019-05-19 $2.83     $1.89         $2.36         $2.52
 9 Cauliflower 2019-05-19 $1.22     $3.24         $4.35         $4.03
10 Carrots     2019-05-19 $0.24     $0.95         $0.95         $0.99
# ℹ 15,756 more rows
# ℹ 3 more variables: newyorkretail <chr>, averagespread <chr>, Year <dbl>
```

```
# removes columns date, farmprice, and averagespread
ppi <-select(ppi, -date, -farmprice, -averagespread)
ppi
```

```
# A tibble: 15,766 × 6
   productname  atlantaretail chicagoretail losangelesretail newyorkretail  Year
   <chr>        <chr>         <chr>         <chr>            <chr>         <dbl>
 1 Strawberries $2.23         $1.70         $1.99            $2.54          2019
 2 Romaine Let… $1.72         $2.00         $1.69            $1.99          2019
 3 Red Leaf Le… $1.84         $1.84         $1.69            $1.89          2019
 4 Potatoes     $5.32         $5.14         $3.99            $6.22          2019
 5 Oranges      $1.42         $1.45         $1.34            $2.05          2019
 6 Iceberg Let… $1.39         $1.46         $1.69            $1.56          2019
```

```
 7 Green Leaf … $1.72         $1.70         $1.69           $1.71          2019
 8 Celery       $1.89         $2.36         $2.52           $2.49          2019
 9 Cauliflower  $3.24         $4.35         $4.03           $2.97          2019
10 Carrots      $0.95         $0.95         $0.99           $1.22          2019
# ℹ 15,756 more rows
```

```
# selects the column year and moves it to be the first column displayed and everything else after
ppi <-ppi %>%
  select(Year, everything())
ppi
```

```
# A tibble: 15,766 × 6
    Year productname  atlantaretail chicagoretail losangelesretail newyorkretail
   <dbl> <chr>        <chr>         <chr>         <chr>            <chr>
 1  2019 Strawberries $2.23         $1.70         $1.99            $2.54
 2  2019 Romaine Let… $1.72         $2.00         $1.69            $1.99
 3  2019 Red Leaf Le… $1.84         $1.84         $1.69            $1.89
 4  2019 Potatoes     $5.32         $5.14         $3.99            $6.22
 5  2019 Oranges      $1.42         $1.45         $1.34            $2.05
 6  2019 Iceberg Let… $1.39         $1.46         $1.69            $1.56
 7  2019 Green Leaf … $1.72         $1.70         $1.69            $1.71
 8  2019 Celery       $1.89         $2.36         $2.52            $2.49
 9  2019 Cauliflower  $3.24         $4.35         $4.03            $2.97
10  2019 Carrots      $0.95         $0.95         $0.99            $1.22
# ℹ 15,756 more rows
```

```
ppi2 <- ppi %>%
  rename(
    Georgia = atlantaretail, # renames atlantaretail column to Georgia
    Illinois = chicagoretail, # renames chicagoretail column to Illinois
    California = losangelesretail,# renames losangelesretail column to California
    `New York` = newyorkretail,# renames newyorkretail column to NewYork
    Date = Year) # renames Year column to Date
ppi2
```

```
# A tibble: 15,766 × 6
    Date productname        Georgia Illinois California `New York`
   <dbl> <chr>              <chr>   <chr>    <chr>      <chr>
 1  2019 Strawberries       $2.23   $1.70    $1.99      $2.54
 2  2019 Romaine Lettuce    $1.72   $2.00    $1.69      $1.99
 3  2019 Red Leaf Lettuce   $1.84   $1.84    $1.69      $1.89
 4  2019 Potatoes           $5.32   $5.14    $3.99      $6.22
 5  2019 Oranges            $1.42   $1.45    $1.34      $2.05
 6  2019 Iceberg Lettuce    $1.39   $1.46    $1.69      $1.56
 7  2019 Green Leaf Lettuce $1.72   $1.70    $1.69      $1.71
 8  2019 Celery             $1.89   $2.36    $2.52      $2.49
 9  2019 Cauliflower        $3.24   $4.35    $4.03      $2.97
10  2019 Carrots            $0.95   $0.95    $0.99      $1.22
# ℹ 15,756 more rows
```

```
# import libraries
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.3.3

Warning: package 'ggplot2' was built under R version 4.3.3

Warning: package 'tibble' was built under R version 4.3.3

Warning: package 'tidyr' was built under R version 4.3.3

Warning: package 'purrr' was built under R version 4.3.3

Warning: package 'forcats' was built under R version 4.3.3

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ forcats 1.0.0     ✓ stringr 1.5.1
✓ ggplot2 3.5.1     ✓ tibble  3.2.1
✓ purrr   1.0.2     ✓ tidyr   1.3.1
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```
# Transform the dataframe from a wide format to a long format
# This function reshapes the dataframe so that columns Georgia, Illinois, California, and New Yorl
# cols: Specifies the columns to pivot (Georgia, Illinois, California, NewYork).
# names_to: The new column name that will contain the original column names after transformation
# values_to: The new column name for the data values from the original columns (Price).
ppi22 <- ppi2 %>%
  pivot_longer(cols = c(Georgia, Illinois, California, `New York`),
               names_to = "State",
               values_to = "Price")
ppi22
```

```
# A tibble: 63,064 × 4
    Date productname      State      Price
   <dbl> <chr>            <chr>      <chr>
 1  2019 Strawberries     Georgia    $2.23
 2  2019 Strawberries     Illinois   $1.70
 3  2019 Strawberries     California $1.99
 4  2019 Strawberries     New York   $2.54
 5  2019 Romaine Lettuce  Georgia    $1.72
 6  2019 Romaine Lettuce  Illinois   $2.00
 7  2019 Romaine Lettuce  California $1.69
 8  2019 Romaine Lettuce  New York   $1.99
 9  2019 Red Leaf Lettuce Georgia    $1.84
```

```
10  2019 Red Leaf Lettuce Illinois   $1.84
# ℹ 63,054 more rows
```

```
# selects the columns so they are displayed in the order of Date, productname, State, Price
ppi22 <- ppi22%>%
  select(Date, productname, State, Price)
ppi22
```

```
# A tibble: 63,064 × 4
    Date productname      State      Price
   <dbl> <chr>            <chr>      <chr>
 1  2019 Strawberries     Georgia    $2.23
 2  2019 Strawberries     Illinois   $1.70
 3  2019 Strawberries     California $1.99
 4  2019 Strawberries     New York   $2.54
 5  2019 Romaine Lettuce  Georgia    $1.72
 6  2019 Romaine Lettuce  Illinois   $2.00
 7  2019 Romaine Lettuce  California $1.69
 8  2019 Romaine Lettuce  New York   $1.99
 9  2019 Red Leaf Lettuce Georgia    $1.84
10  2019 Red Leaf Lettuce Illinois   $1.84
# ℹ 63,054 more rows
```

```
# Perform an inner join between the 'ppi22' and 'obes3' dataframes
# The result is a new dataframe 'join' that contains only rows where there is a match in the 'Dat
join <- inner_join(ppi22, obes3)
```

```
Joining with `by = join_by(Date, State)`
```

```
join
```

```
# A tibble: 13,392 × 5
    Date productname      State      Price Data_Value
   <dbl> <chr>            <chr>      <chr>      <dbl>
 1  2019 Strawberries     Georgia    $2.23       38.9
 2  2019 Strawberries     Illinois   $1.70       35.6
 3  2019 Strawberries     California $1.99       27.7
 4  2019 Strawberries     New York   $2.54       31.0
 5  2019 Romaine Lettuce  Georgia    $1.72       38.9
 6  2019 Romaine Lettuce  Illinois   $2.00       35.6
 7  2019 Romaine Lettuce  California $1.69       27.7
 8  2019 Romaine Lettuce  New York   $1.99       31.0
 9  2019 Red Leaf Lettuce Georgia    $1.84       38.9
10  2019 Red Leaf Lettuce Illinois   $1.84       35.6
# ℹ 13,382 more rows
```

```
# get and list the names of unique fruits and vegetables
product_names <- unique(join$productname)
```

```
for (i in 1:length(product_names)) {
  product_names[i] <- tolower(product_names[i])
  print(sprintf("%d: %s", i, product_names[i]))
}
```

```
[1] "1: strawberries"
[1] "2: romaine lettuce"
[1] "3: red leaf lettuce"
[1] "4: potatoes"
[1] "5: oranges"
[1] "6: iceberg lettuce"
[1] "7: green leaf lettuce"
[1] "8: celery"
[1] "9: cauliflower"
[1] "10: carrots"
[1] "11: cantaloupe"
[1] "12: broccoli crowns"
[1] "13: avocados"
[1] "14: broccoli bunches"
[1] "15: asparagus"
[1] "16: flame grapes"
[1] "17: thompson grapes"
[1] "18: honeydews"
[1] "19: tomatoes"
[1] "20: plums"
[1] "21: peaches"
[1] "22: nectarines"
```

```
# print out prompts for the user
print("Please choose 1 or more products to include in analysis")
```

```
[1] "Please choose 1 or more products to include in analysis"
```

```
print("Enter your choices as a comma separated list of numbers or names:")
```

```
[1] "Enter your choices as a comma separated list of numbers or names:"
```

```
# when compiling an html document, the readline() call is skipped; this line sets a default value
if (!interactive()) {
  user_input <- "1, 2, 3"

} else {

  # get the user input
  user_input <- readline()
}
```

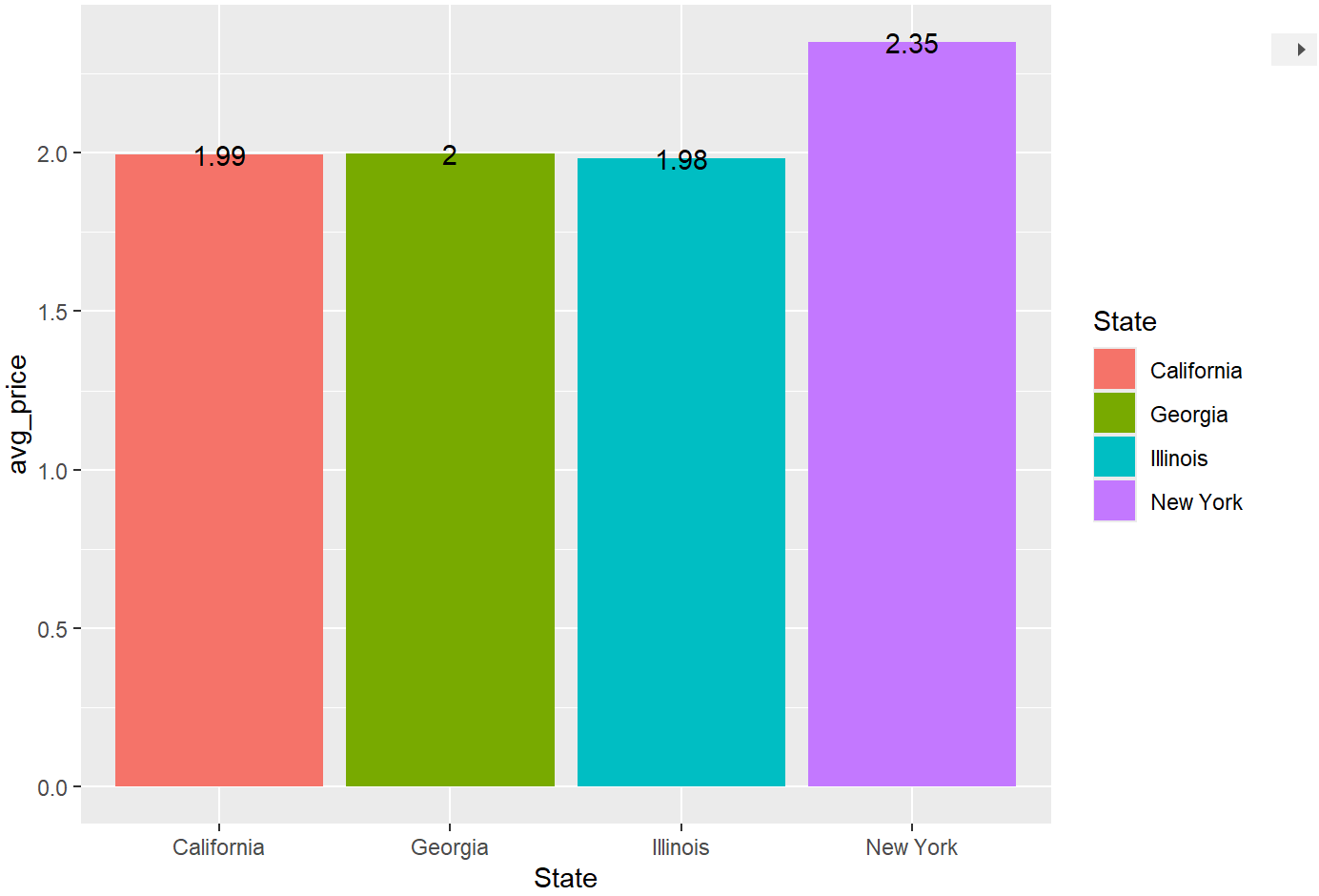
```
# validate user input
user_input <- strsplit(user_input, ",")[[1]]
valid_names <- c()
for (i in 1:length(user_input)) {

  # check the user input name for an item and add it too the list if it matches
  # one in the list
  if (tolower(trimws(user_input[i])) %in% product_names) {
    valid_names <- c(valid_names, tolower(trimws(user_input[i])))
  }

  # try to convert the string to a number, if it succeeds and is valid, add
  # the corresponding product to the list
  integer_format <- as.integer(tolower(trimws(user_input[i])))
  if (!is.na(integer_format) && integer_format > 0 && integer_format < 23) {
    valid_names <- c(valid_names, product_names[integer_format])
  }
}

# check to make sure there was at least one valid product specified, if there
# is not we will be unable to perform analysis, so quit immediately
if (length(valid_names) == 0) {
  stop("No valid fruit/vegetable names specified by user, quitting")
}
```

```
# update the elements in the database to match the names we just found
join$productname <- tolower(join$productname)

# now take the new list of names and filter the dataset accordingly
final_dataset <- join[join$productname %in% valid_names, ]
final_dataset
```

```
# A tibble: 2,704 × 5
    Date productname      State      Price Data_Value
   <dbl> <chr>            <chr>      <chr>      <dbl>
 1  2019 strawberries     Georgia    $2.23       38.9
 2  2019 strawberries     Illinois   $1.70       35.6
 3  2019 strawberries     California $1.99       27.7
 4  2019 strawberries     New York   $2.54       31.0
 5  2019 romaine lettuce  Georgia    $1.72       38.9
 6  2019 romaine lettuce  Illinois   $2.00       35.6
 7  2019 romaine lettuce  California $1.69       27.7
 8  2019 romaine lettuce  New York   $1.99       31.0
 9  2019 red leaf lettuce Georgia    $1.84       38.9
10  2019 red leaf lettuce Illinois   $1.84       35.6
# ℹ 2,694 more rows
```

```
# import libraries
library(ggplot2)
```

```
# Remove dollar signs from Price and convert to numeric
data_clean <- join %>% mutate(Price = as.numeric(gsub("\\$", "", Price)),
  Data_Value = as.numeric(Data_Value)) %>%  # Convert Data_Value column to numeric
  filter(!is.na(Price) & !is.na(Data_Value)) %>%  # Filter out rows with missing Price or Data_Va

# Group by State column and summarize to get the average price and data value per state
group_by(State) %>% summarize(avg_price = mean(Price), avg_data_value = mean(Data_Value))

# Plotting the average price per state
ggplot(data_clean, aes(x = State, y = avg_price, fill = State)) +
  geom_col() + # Create a column plot for average price
  geom_text(aes(label = round(avg_price, 2)))  # Add text showing exact values
```

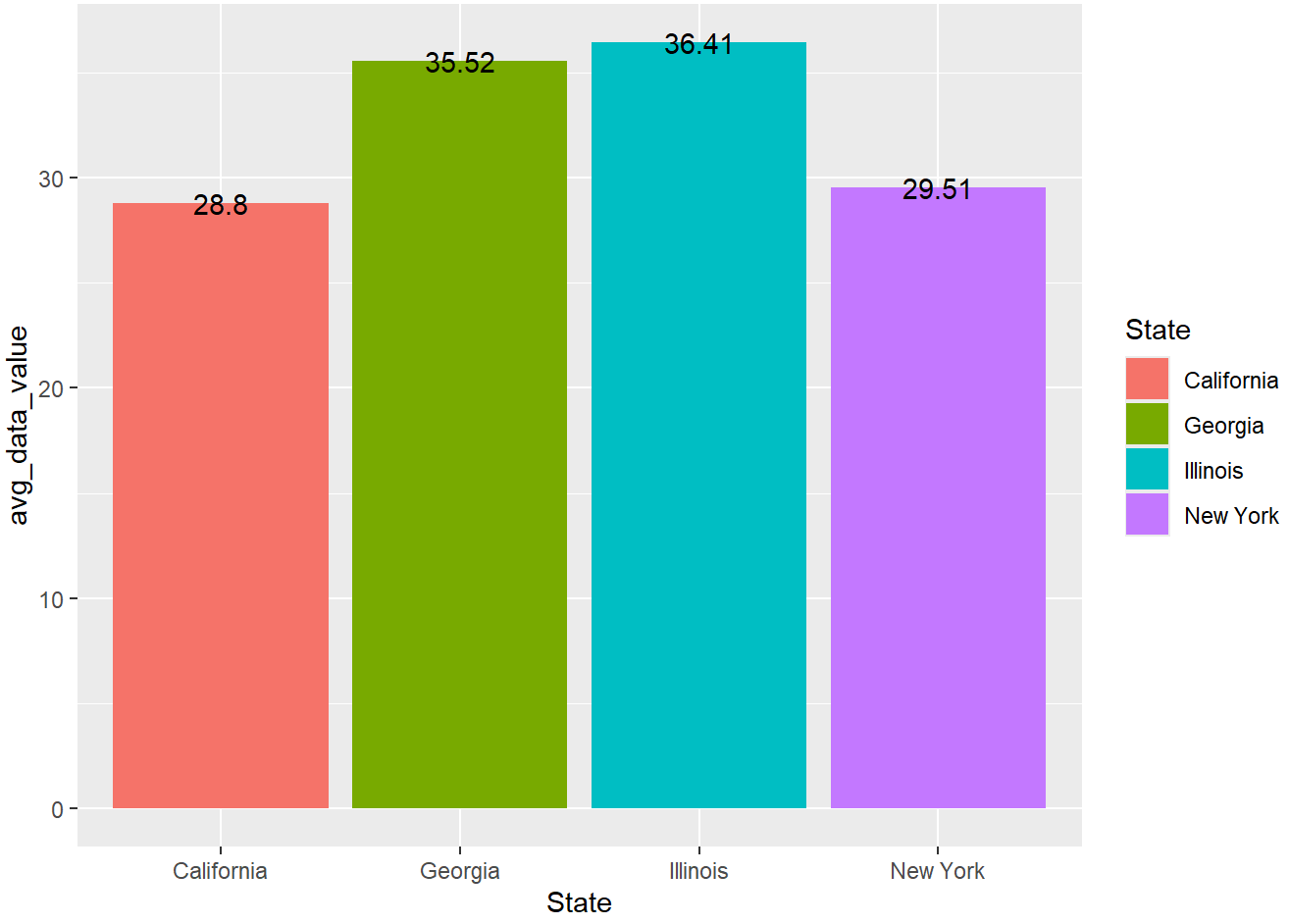



```
  labs(title = "Average Price per State", # label for title, x-axis, and y-axis
       y = "Average Price",
       x = "State")
```

```
$y
[1] "Average Price"

$x
```

```
[1] "State"

$title
[1] "Average Price per State"

attr(,"class")
[1] "labels"
```

```
# Plotting the  average obesity percentage per state
ggplot(data_clean, aes(x = State, y = avg_data_value, fill = State)) +
  geom_col() + # Create a column plot for average obesity percentage
  geom_text(aes(label = round(avg_data_value, 2))) # Add text showing exact values
```




```
  labs(title = "Average Obesity Perctange Per State", # label for title, x-axis, and y-axis
       y = "Average Obesity Perctange",
       x = "State")
```

```
$y
[1] "Average Obesity Perctange"

$x
[1] "State"
```

```
$title
[1] "Average Obesity Perctange Per State"

attr(,"class")
[1] "labels"
```

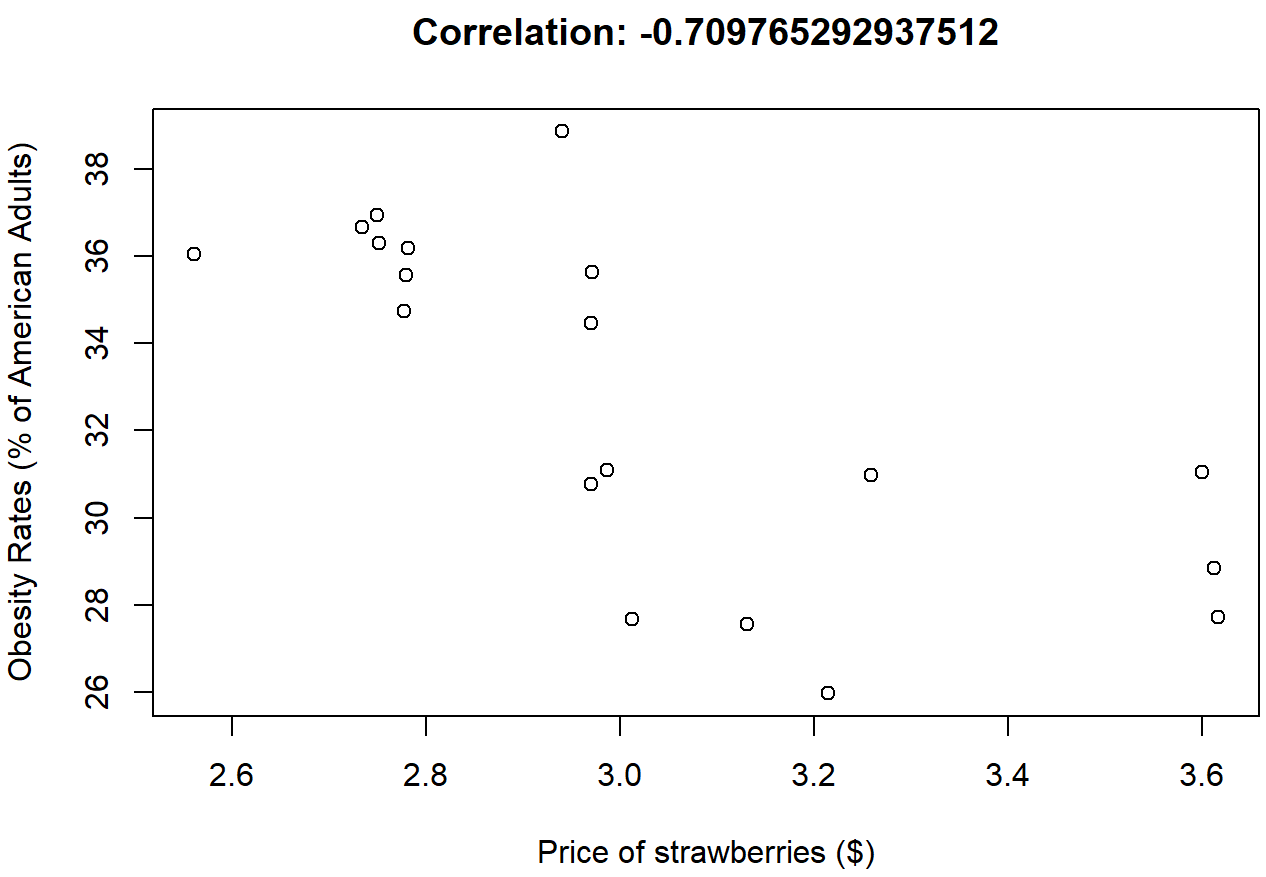
```
correlation_dataset = final_dataset

# remove any stray NA values that made it through
correlation_dataset <- na.omit(correlation_dataset)

# use regex to remove the $ from the price and convert it to a double
correlation_dataset$Price <- as.numeric(sub('.', '', correlation_dataset$Price))

# some rows include items with values recorded on different days of the
# same year, let's aggregate them so that the average values for each years are used
# instead so that it matches the obesity data
correlation_dataset <- correlation_dataset %>%
  group_by(Date, productname, State) %>%
  summarize(Price = mean(Price), Data_Value = mean(Data_Value))
```

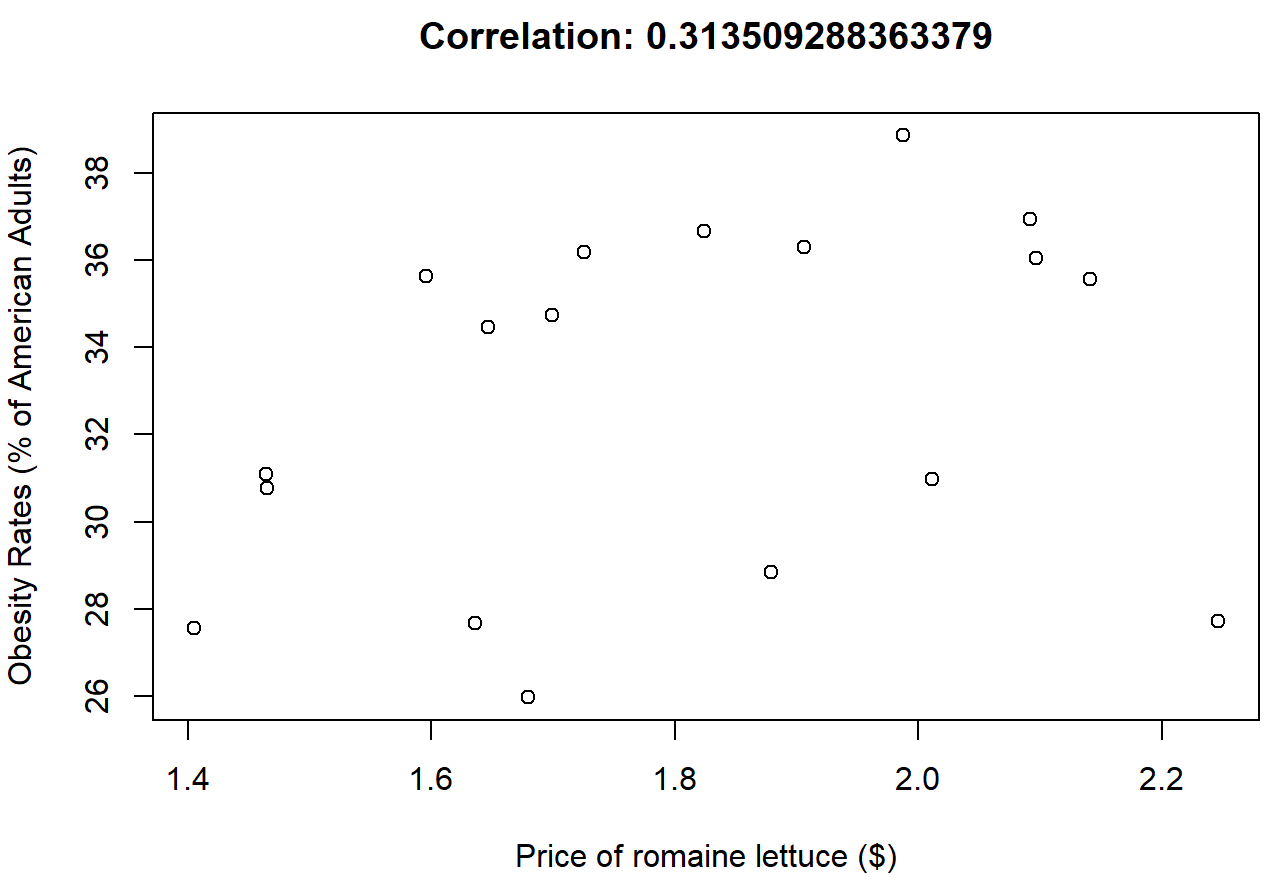
```
`summarise()` has grouped output by 'Date', 'productname'. You can override
using the `.groups` argument.
```

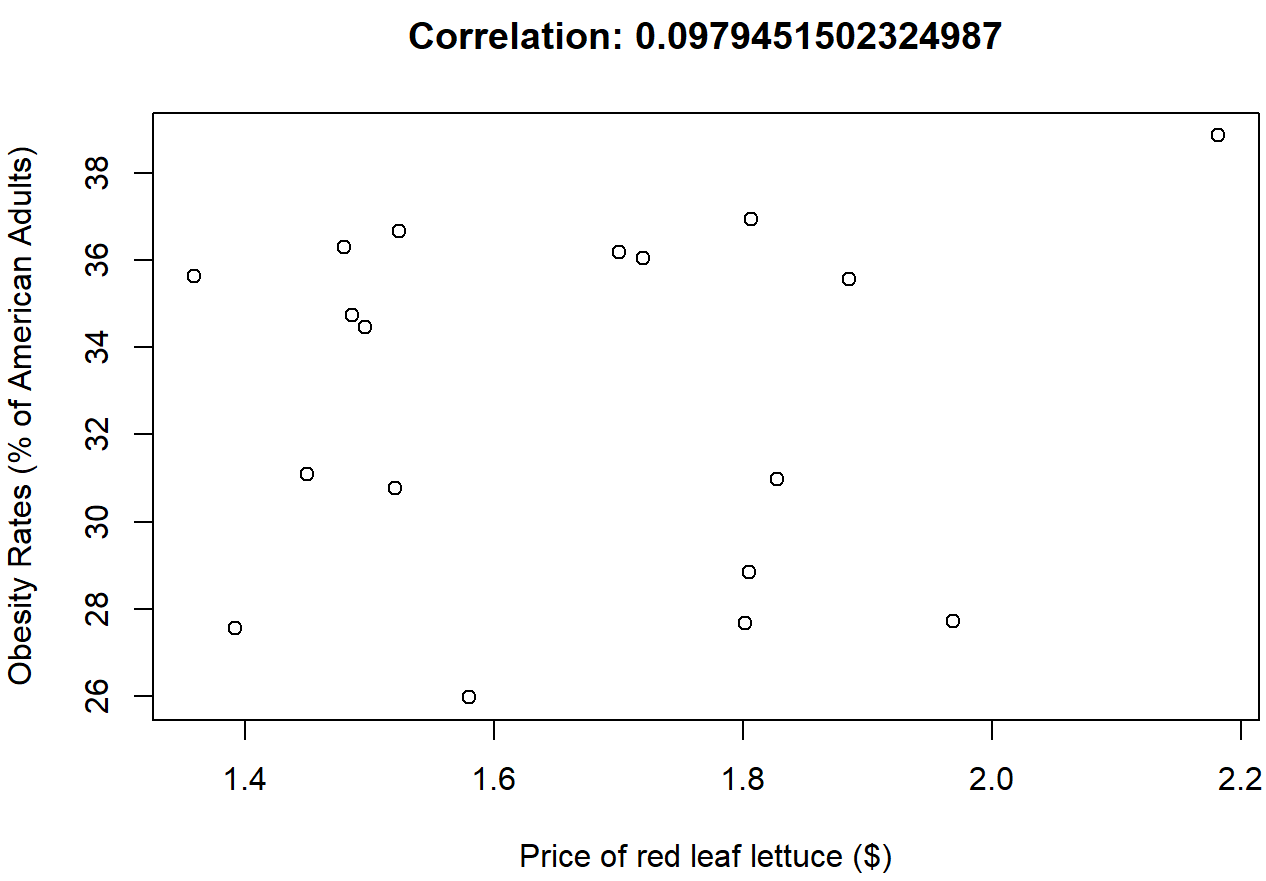
```
# get the necessary columns for calculating the correlation
correlation_dataset <- correlation_dataset %>% select(productname, State, Price, Data_Value)
```

```
Adding missing grouping variables: `Date`
```

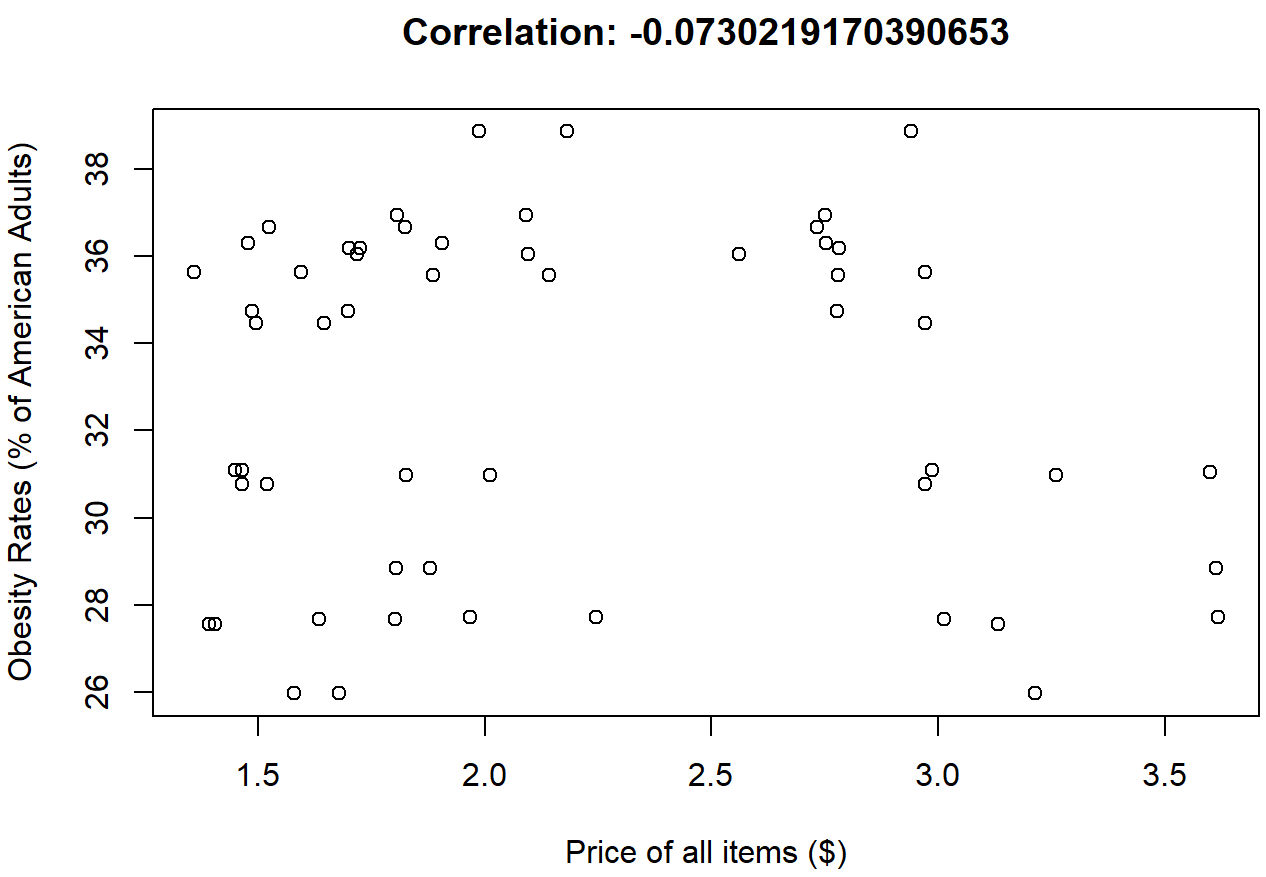
```
# Loop through each product to calculate and plot the correlation for each
for (i in 1:length(valid_names)) {

  # Filter the dataset for the current product
  single_item_correlation_dataset <- filter(correlation_dataset, productname == valid_names[i])

  # Remove rows with NA values from the subset
  single_item_correlation_dataset <- na.omit(single_item_correlation_dataset)

  # Plot the correlation for the current product
  plot(single_item_correlation_dataset$Price, single_item_correlation_dataset$Data_Value,
       xlab = paste("Price of", valid_names[i], "($)", sep = " "),
       ylab = "Obesity Rates (% of American Adults)",
       main = paste("Correlation:", cor(single_item_correlation_dataset$Price, single_item_correla
}
```


Correlation: -0.709765292937512
Obesity Rates (% of American Adults)
26
28
30
32
34
36
38
2.6
2.8
3.0
3.2
3.4
3.6
Price of strawberries ($)


Correlation: 0.313509288363379
Obesity Rates (% of American Adults)
26
28
30
32
34
36
38
1.4
1.6
1.8
2.0
2.2
Price of romaine lettuce ($)




```
# Ensure NA values are handled for the full dataset correlation
# Use the 'use = "complete.obs"' parameter to handle NA values
correlation_value_full <- cor(correlation_dataset$Price, correlation_dataset$Data_Value, use = "c

# Plot the full dataset correlation
plot(correlation_dataset$Price, correlation_dataset$Data_Value,
     xlab = "Price of all items ($)",
     ylab = "Obesity Rates (% of American Adults)",
     main = paste("Correlation:", cor(correlation_dataset$Price, correlation_dataset$Data_Value,
```


Correlation: -0.0730219170390653
Obesity Rates (% of American Adults)
26
28
30
32
34
36
38
1.5
2.0
2.5
3.0
3.5
Price of all items ($)