Data Analyst Assignment

Rebecca Cockroft

2023-01-16

Problem 1 (Merge)

Combining New York and Detroit purchases:

```
# loading required packages

library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0     ✔ purrr   1.0.1
## ✔ tibble  3.1.8     ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1     ✔ stringr 1.5.0
## ✔ readr   2.1.3     ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
# importing csv files

ny <- read_csv("new_york_purchases.csv")
```

```
## Rows: 27 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): barcode, purchase_timestamp, type
## dbl (2): id, amount
##
## 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.
```

```
detroit <- read_csv("detroit_purchases.csv")
```

```
## Rows: 27 Columns: 5
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (3): barcode, amount, type
## dbl  (1): id
## dttm (1): purchase_timestamp
##
## 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.
```

```
# not required but it might be a good idea to create another "location" variable
# by doing so we can tell which dataset it originally came from, and allows for future filtering

ny$location <- "New York"
detroit$location <- "Detroit"

# let's check the data types to make sure they are the same for ny and detroit

str(detroit)
```

```
## spc_tbl_ [27 × 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id              : num [1:27] 0 1 2 3 4 5 6 7 8 9 ...
##  $ barcode         : chr [1:27] "1835566385273" "7750948061357" "7410144862762" "3470282535786" ...
##  $ amount          : chr [1:27] "$1.61" "$3.86" "$2.56" "$2.3" ...
##  $ purchase_timestamp: POSIXct[1:27], format: "2023-01-01 02:58:07" "2023-01-01 20:34:02" ...
##  $ type            : chr [1:27] "vegetable" "vegetable" "dairy" "vegetable" ...
##  $ location        : chr [1:27] "Detroit" "Detroit" "Detroit" "Detroit" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   barcode = col_character(),
##   ..   amount = col_character(),
##   ..   purchase_timestamp = col_datetime(format = ""),
##   ..   type = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
str(ny)
```

```
## spc_tbl_ [27 × 6] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id              : num [1:27] 0 1 2 3 4 5 6 7 8 9 ...
##  $ barcode         : chr [1:27] "0766635859597" "1170285435077" "6963387314679" "5357547177449" ...
##  $ amount          : num [1:27] 3.01 3.48 3.66 3.07 3.74 2.86 2.25 3.81 3.29 2.43 ...
##  $ purchase_timestamp: chr [1:27] "2023-01-01 08:33:37 +0000" "2023-01-01 00:41:34 +0000" "2023-01-01 18:22:27 +0000" "2023-01-01 12:55:49 +0000" ...
##  $ type            : chr [1:27] "puffs" "cakes" "tomato" "beans" ...
##  $ location        : chr [1:27] "New York" "New York" "New York" "New York" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_double(),
##   ..   barcode = col_character(),
##   ..   amount = col_double(),
##   ..   purchase_timestamp = col_character(),
##   ..   type = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
# NY timestamp is not in date/time (POSIX) format, let's change that
# Detroit purchases are automatically coded as UTC when read in, let's change this to EST to be consistent with N
Y.

detroit$purchase_timestamp <- as.POSIXct(as.character(detroit$purchase_timestamp), tz = "EST")
ny$purchase_timestamp <- as.POSIXct(ny$purchase_timestamp, tz = "EST")

# make amount numeric for detroit

detroit$amount <- parse_number(detroit$amount)

# merging the datasets by appending them to each other

full_data <-
  rbind(detroit, ny) %>%

  # dropping "id" column because it's redundant to R's index and is misaligned after merge

  select(-id)

# previewing data

head(full_data)
```

```
## # A tibble: 6 × 5
##   barcode       amount purchase_timestamp  type      location
##   <chr>          <dbl> <dttm>              <chr>     <chr>
## 1 1835566385273   1.61 2023-01-01 02:58:07 vegetable Detroit
## 2 7750948061357   3.86 2023-01-01 20:34:02 vegetable Detroit
## 3 7410144862762   2.56 2023-01-01 14:01:41 dairy     Detroit
## 4 3470282535786   2.3  2023-01-01 01:50:27 vegetable Detroit
## 5 5583880784608   3.67 2023-01-01 21:49:34 dairy     Detroit
## 6 6986147139275   2.8  2023-01-01 17:23:27 dairy     Detroit
```

Normalizing type field to a product line:

```
# selecting rows with specific values and replacing "type" according to mapping

full_data$type[full_data$type %in% c("cakes", "pizzas", "puffs")] <- "bakery"
full_data$type[full_data$type %in% c("milk", "cheese")] <- "dairy"
full_data$type[full_data$type %in% c("tomato", "carrot", "beans")] <- "vegetable"

# checking that only bakery, dairy, and vegetable remain in the data

table(full_data$type)
```

```
##
##    bakery     dairy vegetable
##        21        15        18
```

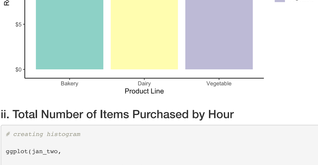
Problem 2 (Filter)

```
# only selecting rows with timestamp on 1/2/2023

jan_two <- full_data %>%
  filter(as.Date(as.character(purchase_timestamp)) == as.Date("2023-01-02"))
```

Problem 3 (Graph)

i. Total Revenue by Product Line

```
# creating barplot

ggplot(

  # aggregating data to sum revenue by product line

  jan_two %>%
    group_by(type) %>%
    summarise(amount = sum(amount)),

  # setting x-axis and color to product line, y to revenue

  aes(x=type, y=amount, fill = type)) +
  geom_bar(stat="identity") +

  # relabeling categories and setting color palette

  scale_fill_brewer(palette = "Set3",
                    labels = c('Bakery','Dairy','Vegetable')) +
  scale_x_discrete(labels = c('Bakery','Dairy','Vegetable')) +

  # putting dollar labels on y-axis

  scale_y_continuous(labels=scales::dollar_format()) +

  # setting legend/x-axis/y-axis/title labels

  labs(fill = "Product Line",
       x = "Product Line",
       y = "Revenue (USD)",
       title = "Total Revenue by Product Line on 1/2/2023") +

  # setting theme

  theme_classic()
```



ii. Total Number of Items Purchased by Hour

```
# creating histogram

ggplot(jan_two,

       # setting x-axis to purchase time

       aes(x = purchase_timestamp)) +

  # making the bin size 1 hour and setting colors

  geom_histogram(binwidth = 3600,
                 boundary = 0,
                 fill = "lightblue",
                 color = "black") +

  # creating labels for each hour

  scale_x_datetime(date_breaks = "1 hour",
                   date_labels = "%H:%M",
                   expand = c(.05, 0, .01, 0)) +

  # setting x-axis/y-axis/title labels

  labs(x = "Time",
       y = "Number of Purchases",
       title = "Purchases by Hour on 1/2/2023") +

  # setting theme

  theme_bw() +

  # rotating x-axis labels so they don't overlap

  theme(axis.text.x = element_text(angle = 60, hjust=1))
```