Problem set 4

Kai Mao

Table of contents

1 Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

2 Problem 1 - Tidyverse

Install and load the package nycflights13.

```
library(nycflights13)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- 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 beco
```

2.1 a.Generate a table

```
# Calculate the mean and median departure delays for each airport
departure_delays <- flights %>%
  group_by(origin) %>%
  summarise(
    mean_departure_delay = mean(dep_delay, na.rm = TRUE),
    median_departure_delay = median(dep_delay, na.rm = TRUE),
    flights_count = n()
  ) %>%
  ungroup() %>%
  filter(flights_count >= 10) %>%
  arrange(desc(mean_departure_delay)) %>%
  left_join(airports, by = c("origin" = "faa")) %>%
  select(name, mean_departure_delay, median_departure_delay)
```

```r
# Output
print(departure_delays)
```

```
# A tibble: 3 x 3
  name                mean_departure_delay median_departure_delay
  <chr>                              <dbl>                  <dbl>
1 Newark Liberty Intl                 15.1                     -1
2 John F Kennedy Intl                 12.1                     -1
3 La Guardia                          10.3                     -3
```

```r
# Calculate the mean and median arrival delays for each airport
arrival_delays <- flights %>%
  group_by(dest) %>%
  summarise(
    mean_arrival_delay = mean(arr_delay, na.rm = TRUE),
    median_arrival_delay = median(arr_delay, na.rm = TRUE),
    flights_count = n()
  ) %>%
  ungroup() %>%
  filter(flights_count >= 10) %>%
  arrange(desc(mean_arrival_delay)) %>%
  left_join(airports, by = c("dest" = "faa")) %>%
  select(name, mean_arrival_delay, median_arrival_delay)

# Output
print(arrival_delays)
```

```
# A tibble: 102 x 3
   name                          mean_arrival_delay median_arrival_delay
   <chr>                                      <dbl>                  <dbl>
 1 Columbia Metropolitan                       41.8                     28
 2 Tulsa Intl                                  33.7                     14
 3 Will Rogers World                           30.6                     16
 4 Jackson Hole Airport                        28.1                     15
 5 Mc Ghee Tyson                               24.1                      2
 6 Dane Co Rgnl Truax Fld                      20.2                      1
 7 Richmond Intl                               20.1                      1
 8 Akron Canton Regional Airport               19.7                      3
 9 Des Moines Intl                             19.0                      0
10 Gerald R Ford Intl                          18.2                      1
# i 92 more rows
```

2.2 b. How many flights did the aircraft model with the fastest average speed take?

```
# Find the aircraft model with the fastest average speed
fastest_aircraft <- flights %>%
  left_join(planes, by = "tailnum") %>%
  group_by(model) %>%
  summarise(
    average_speed = mean(distance / air_time * 60, na.rm = TRUE),  #
    flights_count = n()
  ) %>%
  arrange(desc(average_speed)) %>%
  slice(1) %>%
  select(model, average_speed, flights_count)

# Output
print(fastest_aircraft)
```

```
# A tibble: 1 x 3
  model   average_speed flights_count
  <chr>           <dbl>         <int>
1 777-222          483.             4
```

3 Problem 2 - get_temp()

3.1 Code

```
#' Calculate the average temperature for a given month and year
#'
#' This function calculates the average temperature for a specified month and year.
#' Users can specify if the output should be in Celsius and can provide their own
#' function to compute the mean.
#'
#' @param month The month as a numeric (1-12) or as a full month name (e.g., "January").
#' @param year The year as a numeric value, specifically between 1997 and 2000 for the datase
#' @param data The data frame containing temperature data with columns named 'temp', 'month_n
#' @param celsius Logical, indicating whether to convert the temperature to Celsius. Default
#' @param average_fn The function to use for computing the average temperature. Default is \(
```

```r
#'
#' @return Numeric, the average temperature for the given month and year, in the specified u
#' @export
#'
#' @examples
#' get_temp("Apr", 1999, data = nnmaps, celsius = TRUE)

# load the library
library(tidyverse)

# Define the get_temp function
get_temp <- function(month, year, data, celsius = FALSE, average_fn = mean){
  tryCatch({
    # Normalize month input to numeric if it is a string
    if (month %>% is.character){
      month_name <- c("January", "February", "March", "April", "May", "June",
                      "July","August", "September", "October", "November", "December")
      month %>%
      match.arg(month_name) %>%
      '=='(month_name) %>%
      which -> month
      }
    # Validate the month input
    else if (month %>% is.numeric){
      if (!month %in% 1:12){
        stop("Month value is not 1 to 12.")
      }
    }
    else {
      stop("Month is not character or numeric.")
    }
    # Validate the year input
    if (year %>% is.numeric){
      if (year < 1997 | year > 2000){
        stop("Year value is not 1997 to 2000.")
      }
    }
    else {
      stop("Year is not numeric.")
    }
    # Ensure average_fn is a function
    if(!(average_fn %>% is.function)){
```

```
      stop("average_fn is not a function.")
    }

  # Process the data to calculate the average temperature
  result <- data %>%
    select(temp, month_numeric, year) %>%
    rename(data_year = year) %>%
    filter(data_year == year, month_numeric == month) %>%
    summarise(average_temperature = average_fn(temp), na.rm = TRUE) %>%
    mutate(average_temperature = if (celsius) (average_temperature - 32) * 5/9 else average_t
    pull(average_temperature)

  # Return the average temperature
  return(result)
  }, error = function(e) {
    warning(e$message)
    NA
  })
}
```

3.2 Prove code works by evaluating the following.

```
nnmaps <- read.csv("C:\\Users\\Feixing\\Desktop\\stats 506\\data\\Problem Set 4\\chicago-nmma
get_temp("Apr", 1999, data = nnmaps)
```

```
[1] 49.8
```

```
get_temp("Apr", 1999, data = nnmaps, celsius = TRUE)
```

```
[1] 9.888889
```

```
get_temp(10, 1998, data = nnmaps, average_fn = median)
```

```
[1] 55
```

```
get_temp(13, 1998, data = nnmaps)
```

```
Warning in value[[3L]](cond): Month value is not 1 to 12.
```

```
[1] NA
```

```
get_temp(2, 2005, data = nnmaps)
```

```
Warning in value[[3L]](cond): Year value is not 1997 to 2000.
```

```
[1] NA
```

```
get_temp("November", 1999, data =nnmaps, celsius = TRUE,
         average_fn = function(x) {
           x %>% sort -> x
           x[2:(length(x) - 1)] %>% mean %>% return
         })
```

```
[1] 7.301587
```

4 Problem 3 - Visualization

4.1 a. Is there a change in the sales price in USD over time?

4.1.1 Code

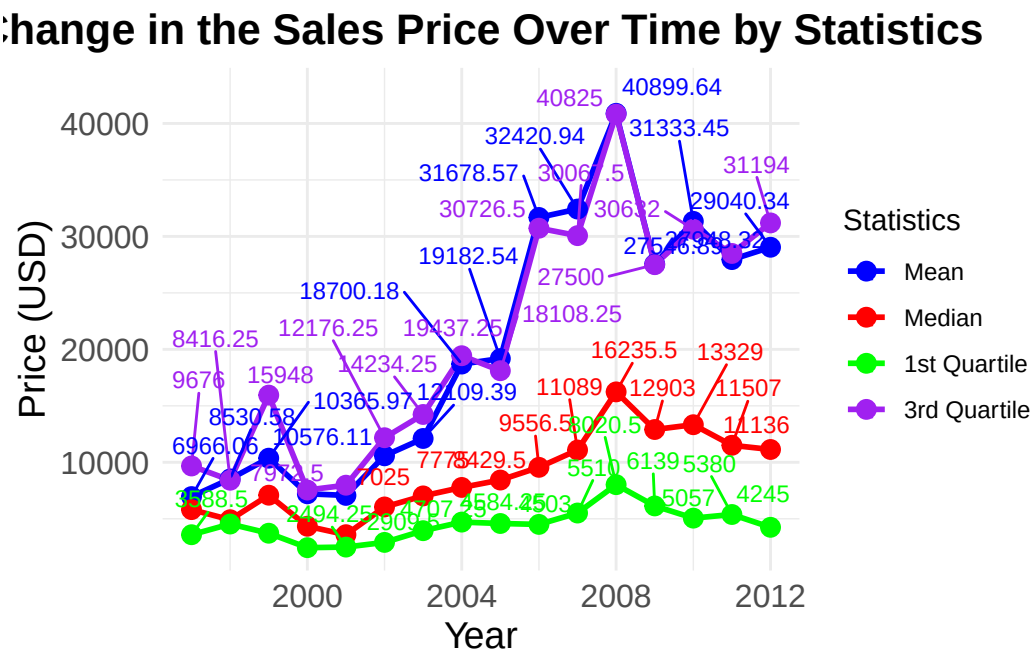
```
# Load the necessary libraries
library(ggplot2)
library(dplyr)
library(ggrepel)  # For better label placement

# Load the data
art_sales <- read.csv("C:\\Users\\Feixing\\Desktop\\stats 506\\data\\Problem Set 4\\df_for_ml

# Prepare the data for plotting
yearly_prices <- art_sales %>%
  group_by(year) %>%
  summarise(
    mean_price = mean(price_usd, na.rm = TRUE),
    median_price = median(price_usd, na.rm = TRUE),
```

```
    q1_price = quantile(price_usd, 0.25, na.rm = TRUE),
    q3_price = quantile(price_usd, 0.75, na.rm = TRUE)
  ) %>%
  pivot_longer(
    cols = c(mean_price, median_price, q1_price, q3_price),
    names_to = "statistic",
    values_to = "price"
  )

# Plot the data with enhanced visualization features
plot <- ggplot(yearly_prices, aes(x = year, y = price, color = statistic)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  geom_text_repel(
    aes(label = round(price, 2)),
    nudge_y = 2000,  # Adjust based on your data scale
    size = 3
  ) +
  labs(
    title = "Change in the Sales Price Over Time by Statistics",
    x = "Year",
    y = "Price (USD)",
    color = "Statistics"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 16),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    plot.caption = element_text(hjust = 0, face = "italic"),
    legend.position = "right"
  ) +
  scale_color_manual(values = c("blue", "red", "green", "purple"),
                     labels = c("Mean", "Median", "1st Quartile", "3rd Quartile"),
                     name = "Statistics")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
# Print the plot
print(plot)
```

```
Warning: ggrepel: 14 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



4.1.2 Conclusion

Absolutely, there is a change in the sales price in USD over time.

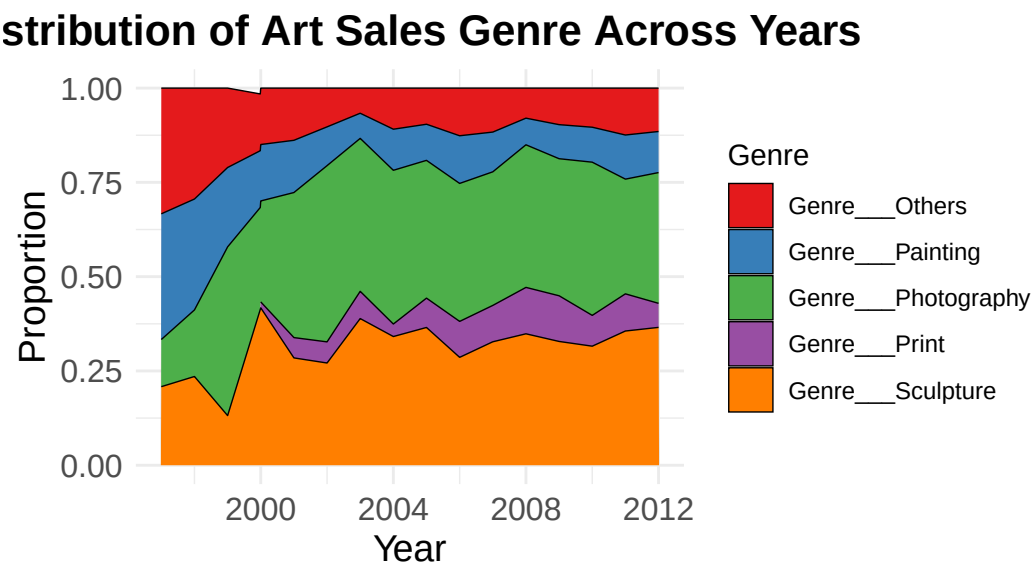
The chart shows the trend of the sale price of art from 1997 to 2012. As can be clearly seen from the chart, selling prices have experienced significant fluctuations, especially between 2004 and 2008, when prices rose significantly and reached their peak. Prices have since fallen back, but the overall trend remains upward through 2012.

4.2 b. Does the distribution of genre of sales across years appear to change?

4.2.1 Code

```
# Load the necessary libraries
library(ggplot2)
library(dplyr)
library(tidyr)
```

```r
# Load the data
art_sales <- read.csv("C:\\Users\\Feixing\\Desktop\\stats 506\\data\\Problem Set 4\\df_for_ml

# Gather genre data for easier plotting
art_sales_genre <- art_sales %>%
  gather(key = "genre", value = "value", starts_with("Genre___")) %>%
  filter(value == 1) %>%
  group_by(year, genre) %>%
  summarise(count = n(), .groups = 'drop')

# Calculate proportions
genre_distribution <- art_sales_genre %>%
  group_by(year) %>%
  mutate(total = sum(count)) %>%
  ungroup() %>%
  mutate(frequency = count / total)

# Plotting the distribution of genre sales over years
plot <- ggplot(genre_distribution, aes(x = year, y = frequency, fill = genre)) +
  geom_area(position = "stack", color = "black", size = 0.25) +
  # Color-blind friendly palette
  scale_fill_brewer(palette = "Set1", name = "Genre") +
  labs(
    title = "Distribution of Art Sales Genre Across Years",
    x = "Year",
    y = "Proportion",
    fill = "Genre",
    caption = "Data source: df_for_ml_improved_new_market.csv\nNote: Each color represents a
    illustrating changes in genre popularity over time."
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, face = "bold", size = 16),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    plot.caption = element_text(hjust = 0, face = "italic"),
    legend.position = "right"
  )

# Print the plot
print(plot)
```



Data source: df_for_ml_improved_new_market.csv
Note: Each color represents a different genre of art sales, illustrating changes in genre popularity over time.

4.2.2 Conclusion

The provided stacked area chart illustrates changes in the distribution of art sales genres over the years from 2000 to 2012. It clearly shows that the popularity of different art genres has fluctuated over time.

Painting (Genre_Painting): This category maintained a significant proportion throughout the period, although there was a decline after 2008.

Photography (Genre_Photography): The popularity of photography increased from 2000 to 2004 and then stabilized, representing a significant share of the art sales.

Print (Genre_Print): The proportion of prints was zero before 2004 but showed a noticeable upward trend afterward.

Sculpture (Genre_Sculpture): The presence of sculptures in the market has been relatively stable across the observed years.

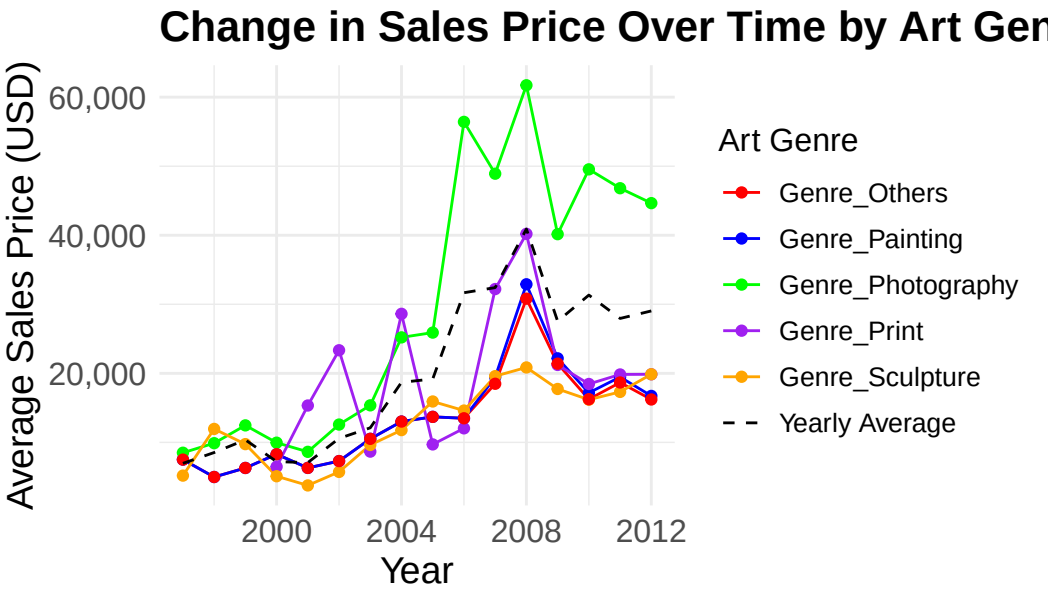
Others (Genre_Others): Initially, this category maintained a significant proportion but began to decrease significantly starting in 2004, stabilizing post-2008.

4.3 c. How does the genre affect the change in sales price over time?

4.3.1 Code

```
# Load necessary libraries
library(tidyverse)

# Load data
art_sales <- read.csv("C:\\Users\\Feixing\\Desktop\\stats 506\\data\\Problem Set 4\\df_for_ml

# Prepare the data for plotting specific genre averages
genre_data <- art_sales %>%
  select(year, price_usd, Genre___Painting, Genre___Photography, Genre___Print, Genre___Sculp
  mutate(
    Genre___Painting = if_else(Genre___Painting == 1, price_usd, NA_real_),
    Genre___Photography = if_else(Genre___Photography == 1, price_usd, NA_real_),
    Genre___Print = if_else(Genre___Print == 1, price_usd, NA_real_),
    Genre___Sculpture = if_else(Genre___Sculpture == 1, price_usd, NA_real_),
    Genre___Others = if_else(Genre___Others == 1, price_usd, NA_real_)
  ) %>%
  group_by(year) %>%
  summarise(across(starts_with("Genre"), ~mean(., na.rm = TRUE), .names = "mean_{.col}"))

# Prepare the yearly average prices data
yearly_prices <- art_sales %>%
  group_by(year) %>%
  summarise(mean_price = mean(price_usd, na.rm = TRUE))

# Filter out years where all genre values are NA
long_genre_data <- genre_data %>%
  filter(rowSums(is.na(across(starts_with("Genre")))) < ncol(.) - 1) %>%
  # Pivot the data longer for plotting
  pivot_longer(
    cols = starts_with("mean_"),
    names_to = "genre",
    values_to = "mean_price",
    names_prefix = "mean_"
  ) %>%
  drop_na(mean_price)

# Plot the changes in sales price over time by art genre
```

```r
plot <- ggplot(long_genre_data, aes(x = year, y = mean_price, color = genre, group = genre))
  geom_line() +
  geom_point() +
  # Add the average yearly prices line
  geom_line(data = yearly_prices, aes(x = year, y = mean_price, group = 1, color = "Yearly Av
  scale_color_manual(
    values = c("red", "blue", "green", "purple", "orange", "black"),
    labels = c("Genre_Others", "Genre_Painting", "Genre_Photography", "Genre_Print", "Genre_S
    name = "Art Genre"
  ) +
  labs(
    title = "Change in Sales Price Over Time by Art Genre",
    x = "Year",
    y = "Average Sales Price (USD)",
    color = "Art Genre",
    caption = "Data source: df_for_ml_improved_new_market.csv\nNote: Each color represents a
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 14),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 12),
    legend.position = "right",
    legend.title = element_text(size = 12),
    legend.text = element_text(size = 10),
    plot.caption = element_text(hjust = 0, face = "italic"),
  ) +
  scale_y_continuous(labels = scales::comma)

# Print the plot
print(plot)
```



Data source: df_for_ml_improved_new_market.csv
Note: Each color represents a different genre of art sales, illustrating changes in sale

4.3.2 Conclusion

From **Question a:** Overall, the average sale price of art peaked between 2004 and 2008 and has since declined significantly.

For the different genres

Painting (Genre_Painting): Prices were relatively low until 2004, but rose after 2008 and stabilized thereafter. Painting prices are usually below the annual average.

Photography (Genre_Photography): Sales prices are volatile, reaching a significant peak in 2008 in particular. Prices for Photography are often higher than the annual average.

Print (Genre_Print): Sales price volatility rose, reaching a peak in 2008, after which it declined sharply. Occasionally above the annual average.

Sculpture (Genre_Sculpture): It is relatively stable, with small price fluctuations and small increases after 2002, showing that the market value of this art type is relatively stable, and most of the time slightly below the annual average price.

Others (Genre_Others): Prices rose significantly between 2006 and 2008, but have since fallen. It is always below the annual average.

5 References and Data Sources

The datasets used in this project are obtained from the following sources:

- **nycflights13 Data**: Provides airline on-time data for all flights departing NYC in 2013. This comprehensive dataset includes metadata on airlines, airports, weather, and planes. Available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.org/package=nycflights13. Maintained by Hadley Wickham.
- **Chicago NMMAPS Data**: The National Morbidity and Mortality Air Pollution Study (NMMAPS) data used in this project focus on the Chicago area from 1997 to 2000. This time-series dataset is employed to examine the relationship between air pollutants and mortality rates. The dataset is used for illustrative purposes to plot the ozone levels over time. Data source link.
- **df_for_ml_improved_new_market**: This dataset enumerates characteristics of art sales, including the sale prices and details about the artworks sold. It is intended for creating publication-ready plots that elucidate market trends and valuations in art sales. More information and dataset download are available at Repository Link.

6 Methods and Implementation

All data analyses were performed in the R environment, employing a range of techniques including data import, data cleaning, statistical analysis, and result visualization.

7 Code and Documentation Repository

This document and related code are hosted on GitHub for review and sharing purposes. Access link: GitHub Repository Link

8 Acknowledgements

Thanks to the course instructors and teaching assistants for their guidance on this assignment. Thanks to all data providers for supporting open data.

9 Notes

The analyses in this document are for academic purposes only, intended to fulfill the requirements of a statistics course. While every reasonable effort has been made to ensure the accuracy of the analysis results, the content of this document represents only the views of the author.