Sales Analysis for Dibs

Zoel Huynh

14/09/2025

Contents

1 Load Libraries

```
library(tidyverse)
library(stringr)
library(tis)        # isHoliday()
library(lubridate)
library(scales)
library(ggplot2)
```

2 Import Datasets

```
sales_01 = read.csv("Dataset/01_Sales_Jan.csv")
sales_02 = read.csv("Dataset/02_Sales_Feb.csv")
sales_03 = read.csv("Dataset/03_Sales_Mar.csv")
sales_04 = read.csv("Dataset/04_Sales_Apr.csv")
sales_05 = read.csv("Dataset/05_Sales_May.csv")
sales_06 = read.csv("Dataset/06_Sales_Jun.csv")
sales_07 = read.csv("Dataset/07_Sales_Jul.csv")
sales_08 = read.csv("Dataset/08_Sales_Aug.csv")
sales_09 = read.csv("Dataset/09_Sales_Sep.csv")
sales_10 = read.csv("Dataset/10_Sales_Oct.csv")
sales_11 = read.csv("Dataset/11_Sales_Nov.csv")
sales_12 = read.csv("Dataset/12_Sales_Dec.csv")
```

3 Clean Data

```
#Append these data sets into one consolidated dataset for further cleaning
combined_sales = rbind(sales_01,sales_02,sales_03,sales_04,sales_05,sales_06,
                       sales_07,sales_08,sales_09,sales_10,sales_11,sales_12,
                       deparse.level = 0)

#Have a glimpse at the data
head(combined_sales)
```

```
##   Order.ID                 Product Quantity.Ordered Price.Each      Order.Date
## 1   141234                  iPhone                1        700  01/22/19 21:25
## 2   141235 Lightning Charging Cable                1      14.95  01/28/19 14:15
## 3   141236         Wired Headphones                2      11.99  01/17/19 13:33
## 4   141237         27in FHD Monitor                1     149.99 1/05/2019 20:33
## 5   141238         Wired Headphones                1      11.99  01/25/19 11:59
## 6   141239   AAA Batteries (4-pack)                1       2.99  01/29/19 20:22
##                        Purchase.Address
## 1        944 Walnut St, Boston, MA 02215
## 2       185 Maple St, Portland, OR 97035
## 3  538 Adams St, San Francisco, CA 94016
## 4     738 10th St, Los Angeles, CA 90001
## 5          387 10th St, Austin, TX 73301
## 6 775 Willow St, San Francisco, CA 94016
```

```
str(combined_sales)
```

```
## 'data.frame':    186894 obs. of  6 variables:
##  $ Order.ID        : chr  "141234" "141235" "141236" "141237" ...
##  $ Product         : chr  "iPhone" "Lightning Charging Cable" "Wired Headphones" "27in FHD Monitor"
##  $ Quantity.Ordered: chr  "1" "1" "2" "1" ...
##  $ Price.Each      : chr  "700" "14.95" "11.99" "149.99" ...
##  $ Order.Date      : chr  "01/22/19 21:25" "01/28/19 14:15" "01/17/19 13:33" "1/05/2019 20:33" ...
##  $ Purchase.Address: chr  "944 Walnut St, Boston, MA 02215" "185 Maple St, Portland, OR 97035" "538 A
```

```
summary(combined_sales)
```

```
##    Order.ID           Product          Quantity.Ordered    Price.Each
##  Length:186894      Length:186894      Length:186894      Length:186894
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##   Order.Date        Purchase.Address
##  Length:186894      Length:186894
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

```
#Rename the columns in a conventional way to make lines of codes more readable
combined_sales <- combined_sales %>% rename(
  order_id = Order.ID,
  product = Product,
```

```
  quantity_ordered = Quantity.Ordered,
  price_each = Price.Each,
  order_date = Order.Date,
  purchase_address = Purchase.Address
)

#Drop duplicated rows within 'combined_sales'
any(duplicated(combined_sales)) #check there are duplications
```

```
## [1] TRUE
```

```
combined_sales <- distinct(combined_sales)

#Convert 'quantity_order' and 'price_each' to numeric type and drop rows with NA values

combined_sales[grepl("\\$", combined_sales$price_each), ] #check if the "$" exists in the "price_each"
```

```
##       order_id              product quantity_ordered price_each      order_date
## 22      141255 USB-C Charging Cable                1     $11.95 1/09/2019 20:55
## 36841   176623     27in FHD Monitor                1    $149.99  04/20/19 23:51
##                        purchase_address
## 22    764 11th St, Los Angeles, CA 90001
## 36841     807 12th St, Atlanta, GA 30301
```

```
combined_sales <- combined_sales %>%
  mutate(price_each = str_replace(price_each, "\\$", "")) %>%
  mutate(across(c(order_id, quantity_ordered, price_each), as.numeric)) %>%
  filter(!is.na(quantity_ordered) & !is.na(price_each) & !is.na(order_id))


#Deal with 'order_date' column

##Check if there is any noise within the date column

###Split 'order_date' into 3 parts
split_dates <- strsplit(combined_sales$order_date, split = "/")

###First column should be the month, check if any strange value
length(which(sapply(split_dates, function(x) nchar(x[[1]]) <= 2))) - nrow(combined_sales) #All columns
```

```
## [1] 0
```

```
order_month <- sapply(split_dates, function(x) as.numeric(x[[1]])) #convert to numeric type

length(which (order_month > 0 & order_month <13)) - nrow(combined_sales)
```

```
## [1] 0
```

```
unique(order_month) #no row has the value out of the range (from 1 to 12)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```r
###Second column should be the day, check if any strange value
order_day <- sapply(split_dates, function(x) as.numeric(x[[2]]))

length(which (order_day > 0 & order_day <32)) - nrow(combined_sales)
```

```
## [1] 0
```

```r
unique(order_day) #no row has the value out of the range (from 1 to 31)
```

```
##  [1] 22 28 17  5 25 29 26  1  7 31  9  3 10 24 30  8 12 19 20 16 11  2 14 27 21
## [26] 13 18  6  4 23 15
```

```r
###Third column of 'split_dates' would be the year and time, further splitting it into 2 corresponding
split_years <- strsplit(sapply(split_dates, "[[", 3), split = " ")

order_year <- sapply(split_years, function(x) as.numeric(x[[1]])) #convert to numeric for the year

unique(order_year)
```

```
## [1]   19 2019   21 2021   28    1 2020   20
```

```r
#1 outlier existed in numeric years with the value of '28', and 1 with '1' -> remove these rows in cons
#some inconsistent format: '19' and '2019','20' and '2020', '21' and '2021', replace these values in th

###Second column of 'split_years' would be the hour and minute, further splitting it into 2 correspondi
split_time <- strsplit(sapply(split_years, "[[", 2), split = ":")

order_hour <- sapply(split_time, function(x) as.numeric(x[[1]])) #split into the hours column

length(which (order_hour >= 0 & order_hour < 24)) - nrow(combined_sales)
```

```
## [1] 0
```

```r
unique(order_hour) ##no row has the value out of the range (from 0 to 23)
```

```
##  [1] 21 14 13 20 11 12 10 18 19 17  8  9  0 22 23 16  7 15  1  2  5  6  3  4
```

```r
order_minute <- sapply(split_time, function(x) as.numeric(x[[2]])) #split into the minutes column

length(which (order_minute >= 0 & order_minute <= 59)) - nrow(combined_sales)
```

```
## [1] 0
```

```r
unique(order_minute) ##no row has the value out of the range (from 0 to 59)
```

```
##  [1] 25 15 33 59 22 16  4 30 20 29 12 57 19 54 13 28  9 51 55 40 47 26  0 14 35
## [26] 46 52 21  5 10  6 23  7 38 17 11 24 58 18  2 32 45  3 43 37 56 36 49  1 31
## [51] 39 34 53  8 50 27 41 48 44 42
```

```r
##Incorporate these datetime columns to consolidated dataset 'combined_sales'
combined_sales <- cbind(combined_sales, order_year, order_month, order_day, order_hour, order_minute)

##Remove noise in datetime-related columns
combined_sales <- combined_sales %>%
  filter(!(order_year == 1 | order_year == 28)) %>%
  mutate(order_year = case_when(
    order_year == 19 ~ 2019,
    order_year == 20 ~ 2020,
    order_year == 21 ~ 2021,
    TRUE ~ order_year
  ))

## Replace the existing 'order_date' with consistent values using datetime-relevant previously generate
combined_sales <- combined_sales %>%
  mutate(order_date = as.Date(paste(order_year, order_month, order_day, sep = "-"),format = "%Y-%m-%d"))

##Drop unnecessary variables or df in R environment to clear up the memory space
rm(order_day, order_hour, order_minute, order_month, order_year, split_dates,split_time,split_years)
rm(sales_01,sales_02,sales_03,sales_04,sales_05,sales_06,sales_07,sales_08,sales_09,sales_10,sales_11,sa


#Deal with 'purchase_address' column

##Filter the outlier row with "N/A" value
combined_sales <- combined_sales %>%
  filter(purchase_address != "N/A")

##Split 'purchass_address' into 4 parts: 'residential_address', 'purchase_city', 'purchase_state_abbrev
split_address <- strsplit(combined_sales$purchase_address, split = ", ")

purchase_city <- sapply(split_address, function(x) (x[[2]]))

split_state_postcode <- strsplit(sapply(split_address, function(x) x[[3]]), split = " ") #third element

purchase_state_abbreviation <- sapply(split_state_postcode, function(x) (x[[1]]))

purchase_postcode <- sapply(split_state_postcode, function(x) (x[[2]]))

##Incorporate these columns: 'purchase_city', 'purchase_state_abbreviation', and 'purchase_postcode' to
combined_sales <- cbind(combined_sales, purchase_city, purchase_state_abbreviation, purchase_postcode)

##Check if there are any noisy data within these new columns

unique(combined_sales$purchase_postcode) #no noise
```

```
##  [1] "02215" "97035" "94016" "90001" "73301" "30301" "98101" "10001" "75001"
## [10] "04101"
```

```r
unique(combined_sales$purchase_state_abbreviation) #no noise
```

```
## [1] "MA" "OR" "CA" "TX" "GA" "WA" "NY" "ME"
```

```r
unique(combined_sales$purchase_city) #have noises
```

```
##  [1] "Boston"        "Portland"      "San Francisco" "Los Angeles"
##  [5] "Austin"        "Atlanta"       "Seattle"       "New York City"
##  [9] "Dallas"        "Las Angeles"   "SanFrancisco"
```

```r
combined_sales <- combined_sales %>%
  mutate(purchase_city = case_when(
    purchase_city == "Las Angeles" ~ "Los Angeles",
    purchase_city == "SanFrancisco" ~ "San Francisco",
    TRUE ~ purchase_city
  ))

##Drop unnecessary variables or df in R environment to clear up the memory space
rm(purchase_city, purchase_postcode, purchase_state_abbreviation, split_address, split_state_postcode)

#Check the noisy data within the 'product' data and replace them
unique(combined_sales$product) #have noises
```

```
##  [1] "iPhone"                    "Lightning Charging Cable"
##  [3] "Wired Headphones"          "27in FHD Monitor"
##  [5] "AAA Batteries (4-pack)"    "27in 4K Gaming Monitor"
##  [7] "USB-C Charging Cable"      "Bose SoundSport Headphones"
##  [9] "Apple Airpods Headphones"  "Macbook Pro Laptop"
## [11] "Flatscreen TV"             "Vareebadd Phone"
## [13] "AA Batteries (4-pack)"     "Google Phone"
## [15] "AAA Batteries (4pack)"     "20in Monitor"
## [17] "34in Ultrawide Monitor"    "ThinkPad Laptop"
## [19] "LG Dryer"                  "Wired Headphoness"
## [21] "GooOgle Phone"             "LG Washing Machine"
## [23] "IPhone"                    "LightCharging Cable"
## [25] "USBC Charging Cable"       "##system error##"
```

```r
combined_sales <- combined_sales %>%
  mutate(product = case_when(
    product == "AAA Batteries (4pack)" ~ "AAA Batteries (4-pack)",
    product == "Wired Headphoness" ~ "Wired Headphones",
    product == "GooOgle Phone" ~ "Google Phone",
    product == "IPhone" ~ "iPhone",
    product == "LightCharging Cable" ~ "Lightning Charging Cable",
    product == "USBC Charging Cable" ~ "USB-C Charging Cable",
    TRUE ~ product
  )) %>%
  filter(product != "##system error##")

#Add extra features that are necessary for further parts
## Create a new column to calculate sales
combined_sales <- combined_sales %>%
  mutate(sales = quantity_ordered*price_each)

## Create a new column to flag if it is the weekend or not
combined_sales <- combined_sales %>%
```

```
  mutate(is_weekend = weekdays(order_date) %in% c("Saturday", "Sunday")) %>%
  mutate(is_holiday = isHoliday(order_date))
```

4 Ad-Hoc Business Questions

```
# a. What is the worst year of sales and how much sales was earned?

# Create a tibble that has the total sales of each year.
yearly_sales <- combined_sales %>%
  group_by(order_year) %>%
  summarise(total_sales = sum(sales))

# Identify the best year of sales
best_year <- yearly_sales$order_year[which.max(yearly_sales$total_sales)]
worst_year <- yearly_sales$order_year[which.min(yearly_sales$total_sales)]

# Print the statement.
paste0("The worst year of sales is ", worst_year, " and the sales earned were $", min(yearly_sales$total
```

```
## [1] "The worst year of sales is 2021 and the sales earned were $3926.82"
```

```
# b. How much was earned in the best Year of sales?
paste0("The best year of sales is ", best_year, " and the sales earned were $", max(yearly_sales$total_s
```

```
## [1] "The best year of sales is 2019 and the sales earned were $34456867.65"
```

```
# c. In the best year of sales which was the best month for sales?
# Identify the best year of sales
best_year_sales <- combined_sales %>%
  filter(order_year == best_year)

# Create a tibble that has the total sales per month in the best year of sales.
monthly_sales <- best_year_sales %>%
  group_by(order_month) %>%
  summarise(total_sales = sum(sales))

# Identify the best month of sales in the best year of sales.
best_month <- month.name[which.max(monthly_sales$total_sales)]

# Print the statement.
paste("The best month of sales in the best year of sales was", best_month)
```

```
## [1] "The best month of sales in the best year of sales was December"
```

```
# d. In the best year of sales how much was earned in the best month?
paste0("The sales earned in the best month of sales (", best_month, ") in the best year of sales (", bes
```

```
## [1] "The sales earned in the best month of sales (December) in the best year of sales (2019) is $4608
```

```r
# e. Which City had the most sales in the best year of sales?
# Create a tibble that has the total sales categorized by cities.
city_most_sales <- best_year_sales %>%
  group_by(purchase_city) %>%
  summarise(total_sales = sum(sales))

# Print the statement.
paste0(city_most_sales$purchase_city[which.max(city_most_sales$total_sales)], " is the city that had th
```

```
## [1] "San Francisco is the city that had the most sales in the best year of sales (2019)"
```

```r
# f. To maximise the likelihood of customers buying a product, what time should Dibs business be displa
# Create a tibble that has total sales per hour.
hourly_sales <- best_year_sales %>%
  group_by(order_hour) %>%
  summarise(total_sales = sum(sales))

# Rank the total sales per hour in a descending order.
hourly_sales_sorted <- hourly_sales %>%
  arrange(desc(total_sales))

# List the top 5 hours that have the highest total sales.
top_5_hours <- head(hourly_sales_sorted, 5)
top_5_hours
```

```
## # A tibble: 5 x 2
##   order_hour total_sales
##        <dbl>       <dbl>
## 1         19    2411971.
## 2         12    2314360.
## 3         11    2296620.
## 4         20    2280784.
## 5         18    2218374.
```

```r
# Print the statement.
paste0("The best time for displaying advertisements is at ", hourly_sales$order_hour[which.max(hourly_sa
```

```
## [1] "The best time for displaying advertisements is at 19:00"
```

```r
paste0("The optimal time slots for displaying advertisement are from 6 PM to 8 PM and from 11 AM to 12 P
```

```
## [1] "The optimal time slots for displaying advertisement are from 6 PM to 8 PM and from 11 AM to 12 P
```

```r
# g. Which products are most often sold together?

# Select duplicated order_id
duplicated_ids <- combined_sales$order_id[duplicated(combined_sales$order_id)]
d_ids <- combined_sales[combined_sales$order_id %in% duplicated_ids, ]

# Concatenate the products within each group of order_id.
```

```r
product_sold_tgt <- d_ids %>%
  group_by(order_id) %>%
  summarize(products = paste(product, collapse = ", "))

# Count the frequency of products that were sold together.
product_sold_tgt_counts <- product_sold_tgt %>%
  count(products, sort = TRUE)

# Print the statement
paste(product_sold_tgt_counts[1, 1], "were most often sold together.")
```

```
## [1] "iPhone, Lightning Charging Cable were most often sold together."
```

```r
# h Overall which product sold the most and why do you think it has sold the most?
# Calculate the total of quantity_ordered within each products group.
product_sold <- best_year_sales %>%
  group_by(product) %>%
  summarise(good_sold = sum(quantity_ordered))

# Identify which product has sold the most.
best_product <- product_sold$product[which.max(product_sold$good_sold)]

# Print the statement
paste0(best_product, " is the product that was sold the most as it had the highest number of quantities
```

```
## [1] "AAA Batteries (4-pack) is the product that was sold the most as it had the highest number of qua
```

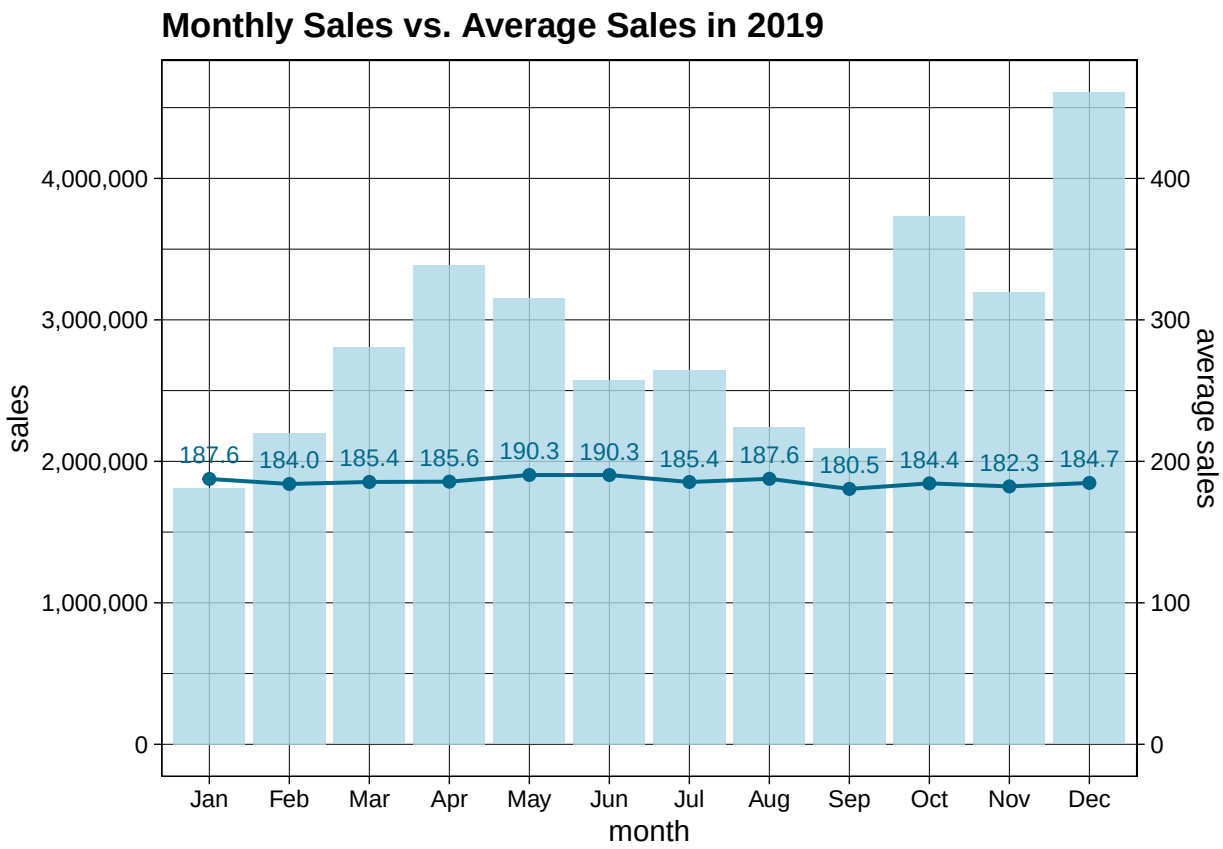
```r
# i. What is the least sold product in the best year of sales?

# Identify which product has sold the least.
worst_product <- product_sold$product[which.min(product_sold$good_sold)]

# Print the statement.
paste0(worst_product, " was the product that was sold the least in the best year of sales.")
```

```
## [1] "LG Dryer was the product that was sold the least in the best year of sales."
```

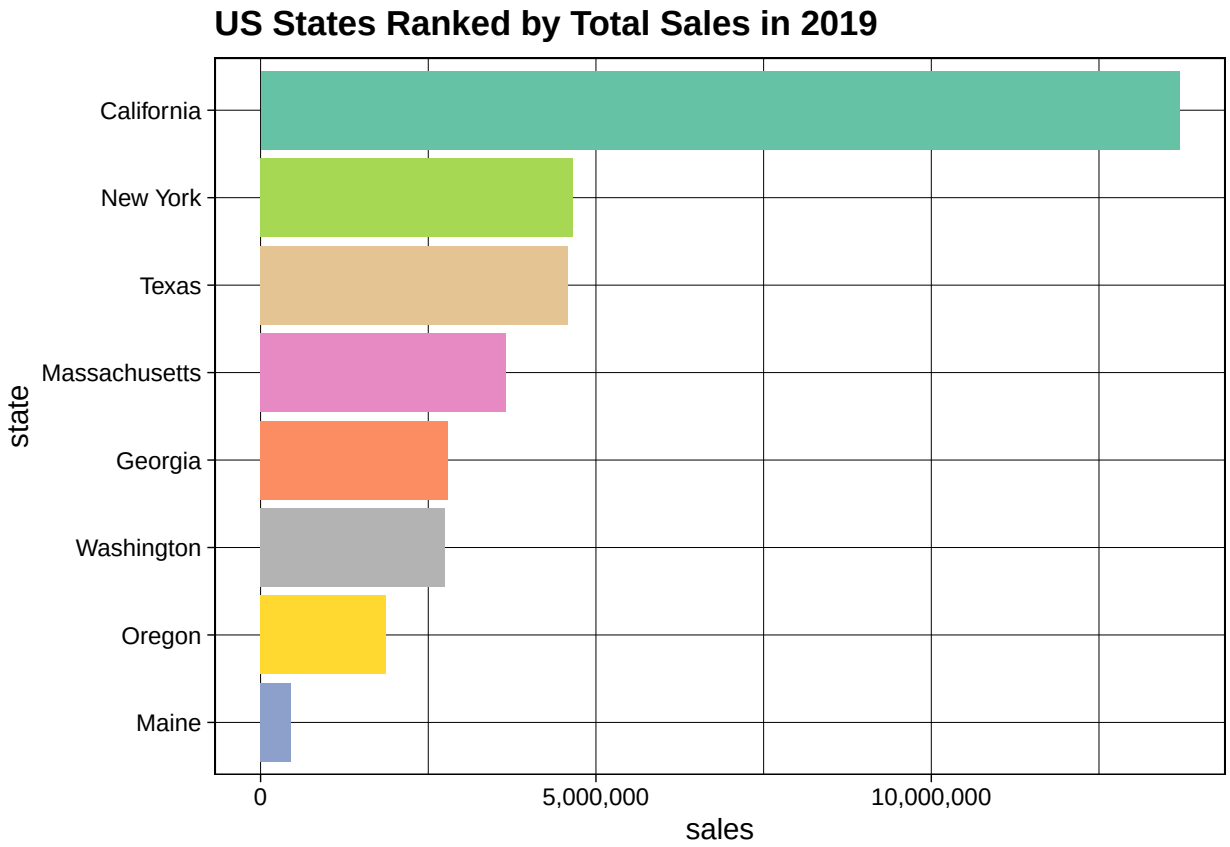
5 Visualisations

```r
### a. Monthly sales trend vs monthly average sales

## Graph 1: Monthly sales trend vs monthly average sales in 2019 using combine line and bar chart

# Load necessary libraries to change the data type of order_month
library(ggplot2)
library(scales)
# recoding state abbreviations to full state names
# converting numeric month values to their corresponding abbreviated month names
# add column date
```

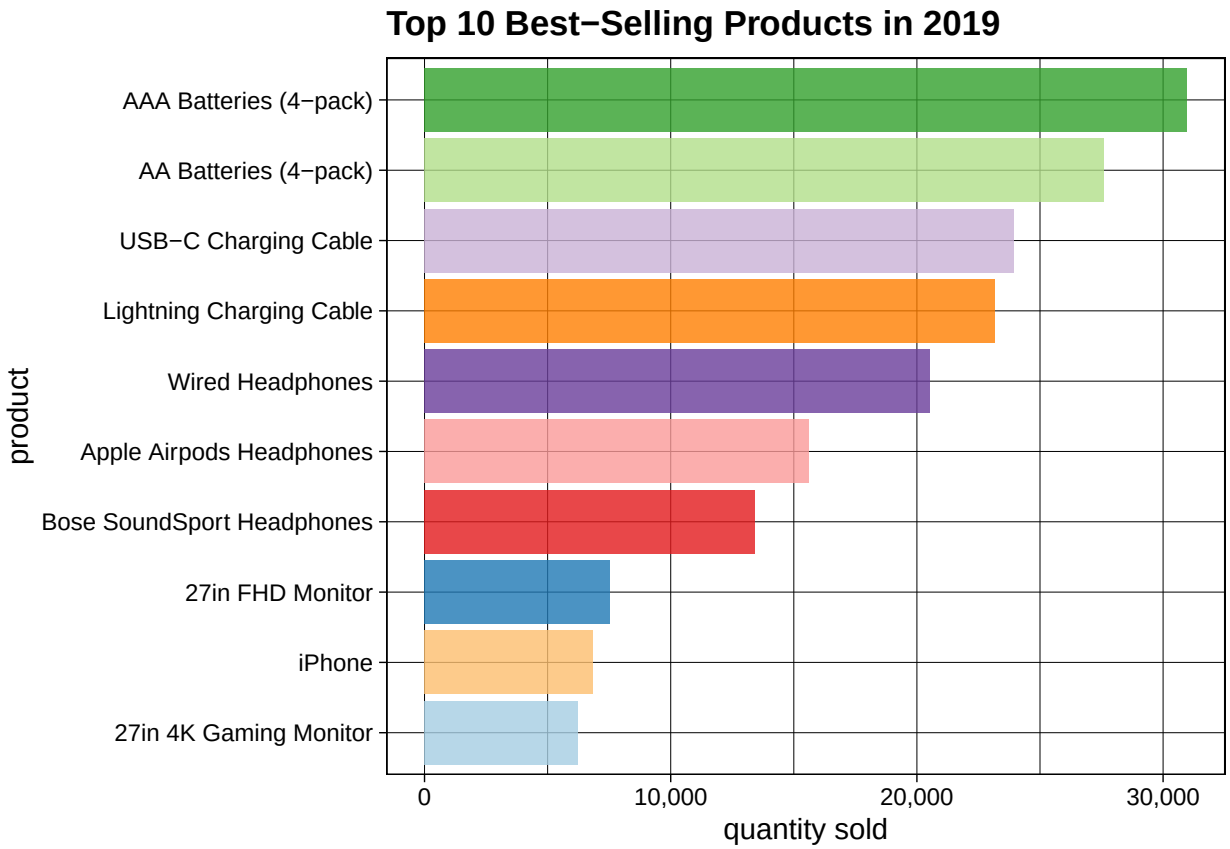
```r
sales <- best_year_sales %>%
  mutate(state = recode(purchase_state_abbreviation,
                        'CA' = 'California',
                        'GA' = 'Georgia',
                        'MA' = 'Massachusetts',
                        'ME' = 'Maine',
                        'NY' = 'New York',
                        'OR' = 'Oregon',
                        'TX' = 'Texas',
                        'WA' = 'Washington')) %>%
  mutate(month = factor(order_month, levels = 1:12, labels=month.abb),
         wday = wday(order_date, label = TRUE))

# groups the sales data by month, then calculates and summarizes the total and average sales for each m
monthly_sales <- sales %>%
  group_by(month) %>%
  summarise(total_sales = sum(sales),
            average_sales = mean(sales))
# create the plot using bar and line graph combined
monthly_sales$formatted_average_sales <- sprintf("%.1f", monthly_sales$average_sales) #change the forma
ggplot(monthly_sales) +
  geom_bar(aes(x = month, y = total_sales), stat = 'identity', fill ="lightblue", alpha = 0.8) +
  geom_line(aes(x = month, y = average_sales * 10000, group = 1), linewidth = 0.7,color="deepskyblue4")
  geom_point(aes(x = month, y = average_sales * 10000), color = "deepskyblue4",size=1.8) +
  geom_text(aes(x = month, y = average_sales * 10000, label = formatted_average_sales), vjust = -0.9, c
  scale_y_continuous(label = comma, sec.axis = sec_axis(~./10000, name = "average sales")) +
  theme_linedraw()+
  labs(title = "Monthly Sales vs. Average Sales in 2019", x = "month", y = "sales")+
  theme(plot.title = element_text(face = "bold"))
```



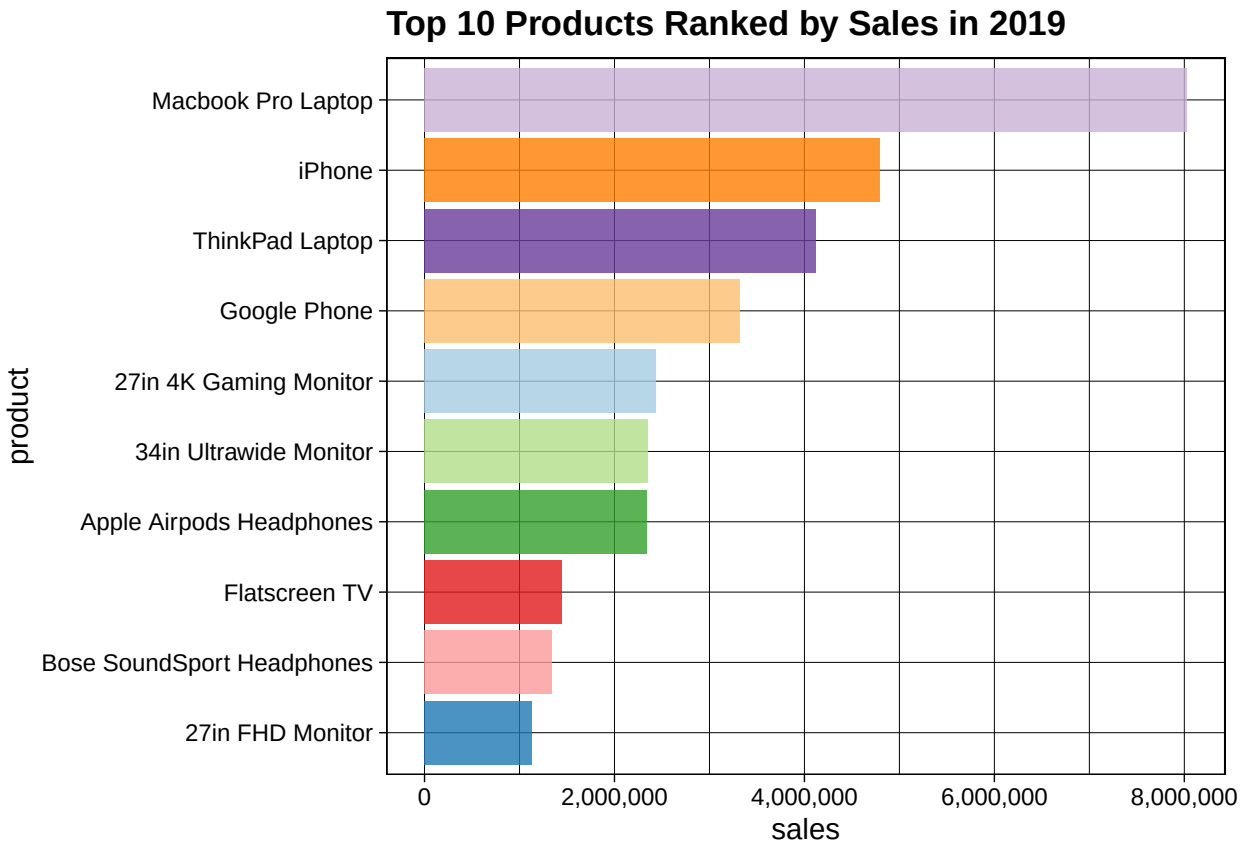

```
### b. Sales by state

## Graph 1:  SALES COMPARISON AMONG STATES

# grouping the sales dataset by state and summarizing the sales
sales_state <- sales %>%
  group_by(state) %>%
  summarise(total_sales = sum(sales))
# create bar chart
ggplot(sales_state) +
  geom_bar(aes(x=reorder(state,total_sales),y=total_sales,fill=state),
           stat='identity',alpha=1.1) +
  labs(x="state", y="sales", title="US States Ranked by Total Sales in 2019") +
  coord_flip() +
  scale_y_continuous(label = comma) +
  theme_linedraw() +
  guides(fill = "none") +
  scale_fill_brewer(palette="Set2")+
  theme(plot.title = element_text(face = "bold"))
```



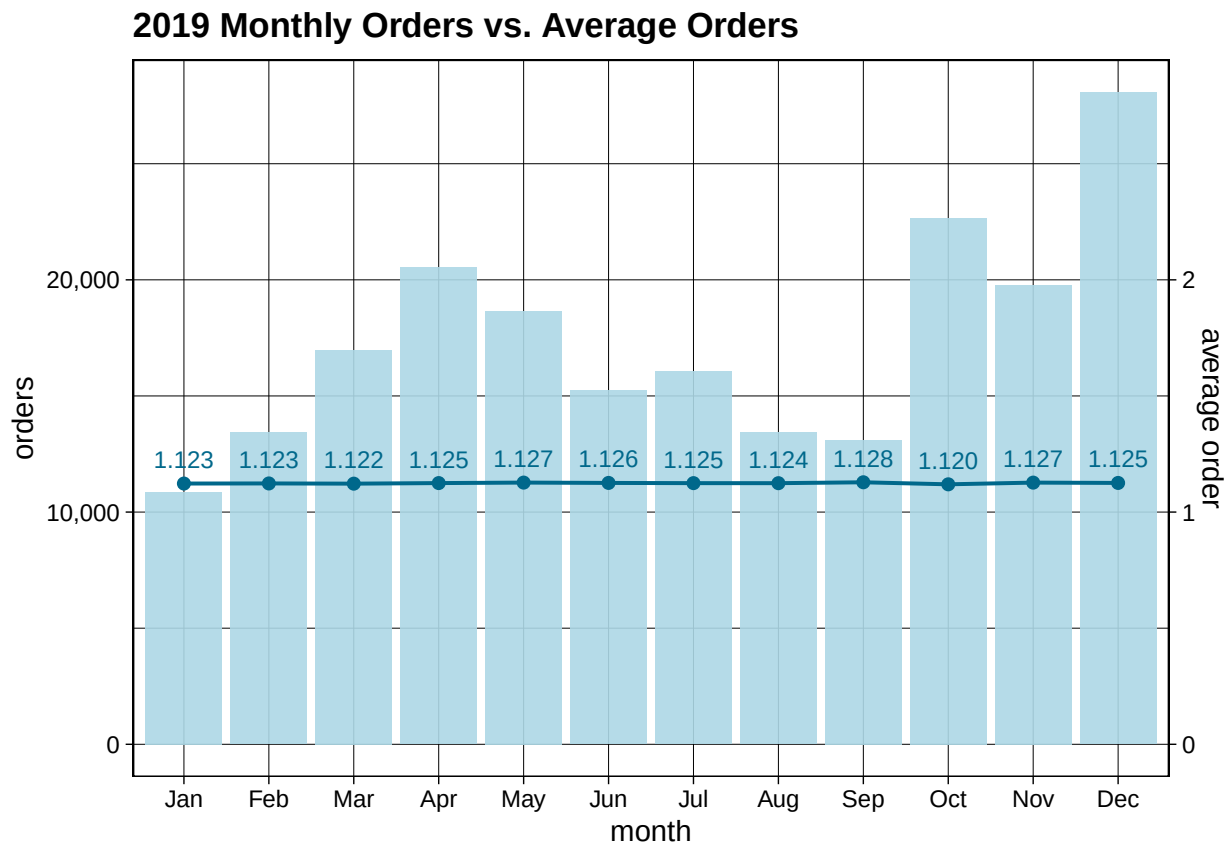

```
### c. Top 10 products sold in the best year of sales

## Graph 1: TOP 10 PRODUCTS ORDERED IN 2019

# identify the top 10 best-selling products by total quantity ordered -> arranging them by rank
top10_product_order <- sales %>%
  group_by(product) %>%
  summarise(good_sold = sum(quantity_ordered)) %>%
  mutate(rank = min_rank(desc(good_sold))) %>%
  filter(rank<=10) %>%
  arrange(rank)
# create the bar chart
ggplot(top10_product_order) +
  geom_bar(aes(x=reorder(product,good_sold),y=good_sold,fill=product),
           stat='identity',alpha=0.8) +
  labs(x="product", y="quantity sold", title="Top 10 Best-Selling Products in 2019") +
  coord_flip() +
  scale_y_continuous(label = comma) +
  theme_linedraw() +
  guides(fill = "none") +
  scale_fill_brewer(palette="Paired")+
  theme(plot.title = element_text(face = "bold"))
```




```
## Graph 2: TOP 10 PRODUCTS IN SALES REVENUE IN 2019

# identify the top 10 highest-selling products by total sales -> arranging them by rank
top_product_sales <- sales %>%
  group_by(product) %>%
  summarise(total_sales = sum(sales)) %>%
  mutate(rank = min_rank(desc(total_sales))) %>%
  filter(rank<=10) %>%
  arrange(rank)
# create the bar chart
ggplot(top_product_sales) +
  geom_bar(aes(x=reorder(product,total_sales),y=total_sales,fill=product),
           stat='identity',alpha=0.8) +
  labs(x="product", y="sales", title="Top 10 Products Ranked by Sales in 2019") +
  coord_flip() +
  scale_y_continuous(label = comma) +
  theme_linedraw() +
  guides(fill = "none") +
  scale_fill_brewer(palette="Paired")+
  theme(plot.title = element_text(face = "bold"))
```

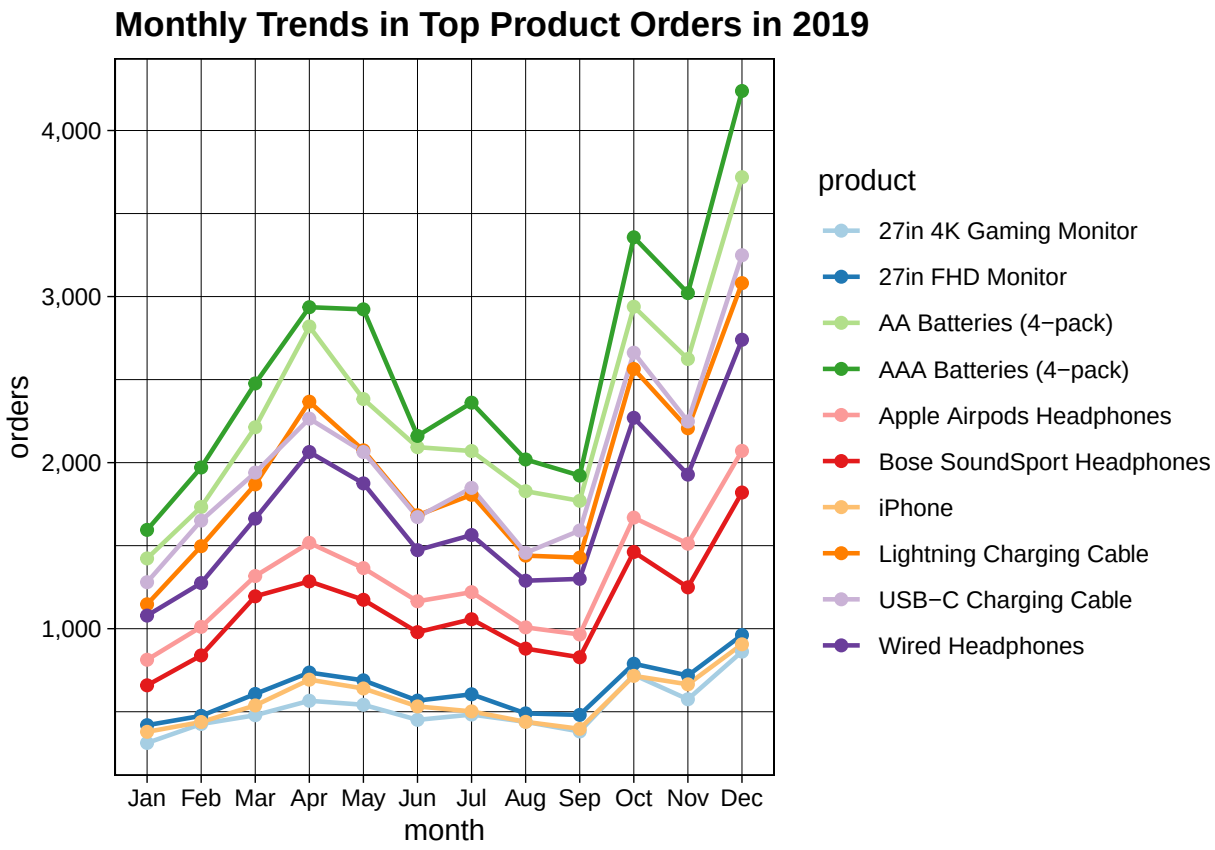



```
### d. Monthly order trend vs monthly average order

## Graph 1: MONTHLY ORDER AND MONTHLY AVERAGE ORDER

# computes the total and average quantity ordered per month from the 'sales', grouped by order_month
monthly_order <- sales %>%
  group_by(month) %>%
  summarise(total_order = sum(quantity_ordered),
            average_order = mean(quantity_ordered))
# create the plot
monthly_order$formatted_average_order <- sprintf("%.3f", monthly_order$average_order) #change the forma
ggplot(monthly_order)+
  geom_bar(aes(x= month,y= total_order),stat='identity',fill="lightblue",alpha=0.9)+
  geom_line(aes(x= month,y= average_order*10000, group=1),linewidth =0.7, color = "deepskyblue4")+
  geom_point(aes(x= month,y= average_order*10000),color = "deepskyblue4",size=1.8)+
  geom_text(aes(x = month, y = average_order * 10000, label = formatted_average_order), vjust = -0.9, c
  scale_y_continuous(sec.axis=sec_axis(~./10000,name="average order"),label = comma)+
  theme_linedraw()+
  labs(title="2019 Monthly Orders vs. Average Orders",x="month",y="orders") +
  theme(plot.title = element_text(face = "bold"))
```

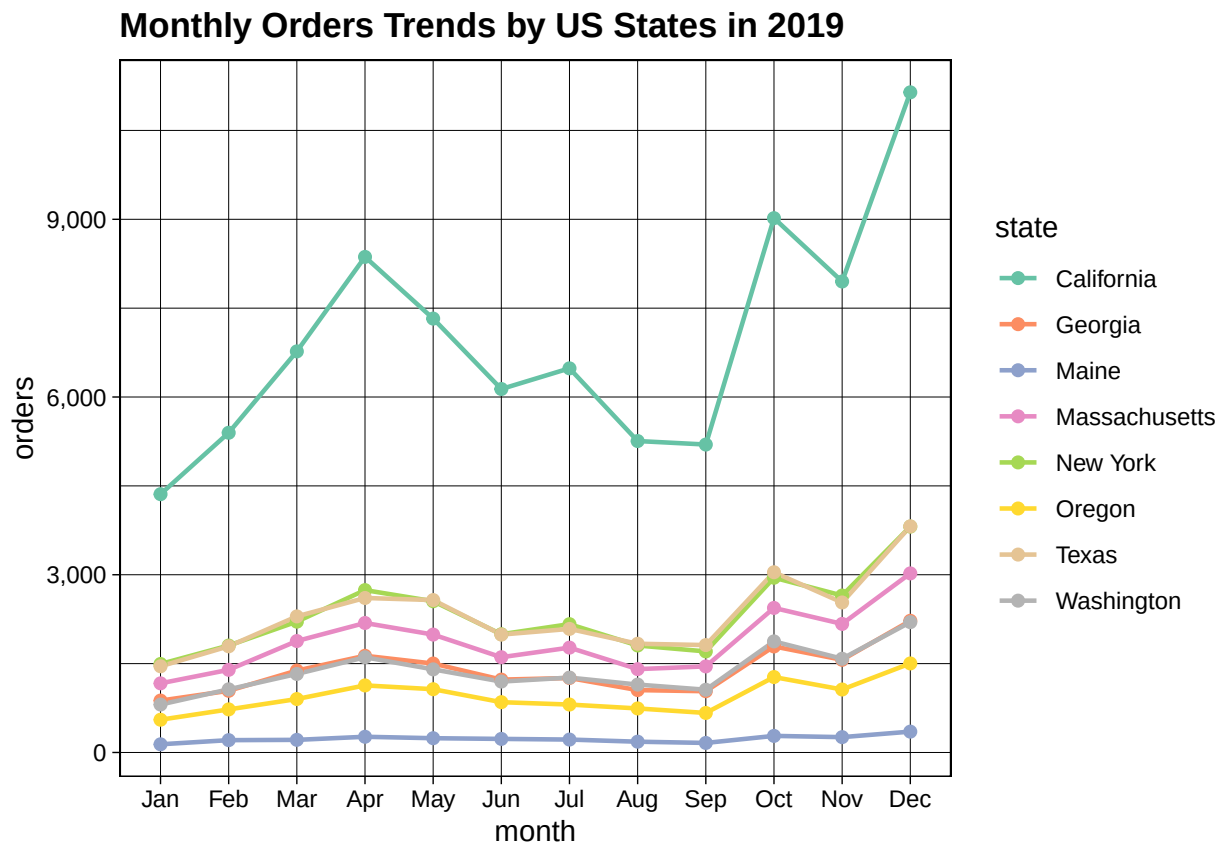



```
## Graph 2: MONTHLY AVERAGE ORDER BY TOP PRODUCTS

# filters the product column matches one of the specified top products
top_product <- sales %>%
  filter(product == "AAA Batteries (4-pack)"|
           product == "AA Batteries (4-pack)"|
           product == "USB-C Charging Cable"|
           product == "Lightning Charging Cable"|
           product == "Wired Headphones"|
           product == "Apple Airpods Headphones"|
           product == "Bose SoundSport Headphones"|
           product == "27in FHD Monitor"|
           product == "iPhone"|
           product == "27in 4K Gaming Monitor")

monthly_order_prod <- top_product %>%
  group_by(month,product) %>%
  summarise(total_order = sum(quantity_ordered),
            average_order = mean(quantity_ordered))
# create a line plot showing the monthly average quantity ordered for top products in 2019
ggplot(monthly_order_prod,aes(x= month,y= total_order, color = product, group=product))+
  geom_line(linewidth = 0.8,alpha=1)+
  geom_point(size=1.8)+
  theme_linedraw()+
  labs(title="Monthly Trends in Top Product Orders in 2019",x="month",y="orders",color = "product")+
  scale_color_brewer(palette="Paired")+
```

```
  scale_y_continuous(label = comma)+
  theme(plot.title = element_text(face = "bold"))
```

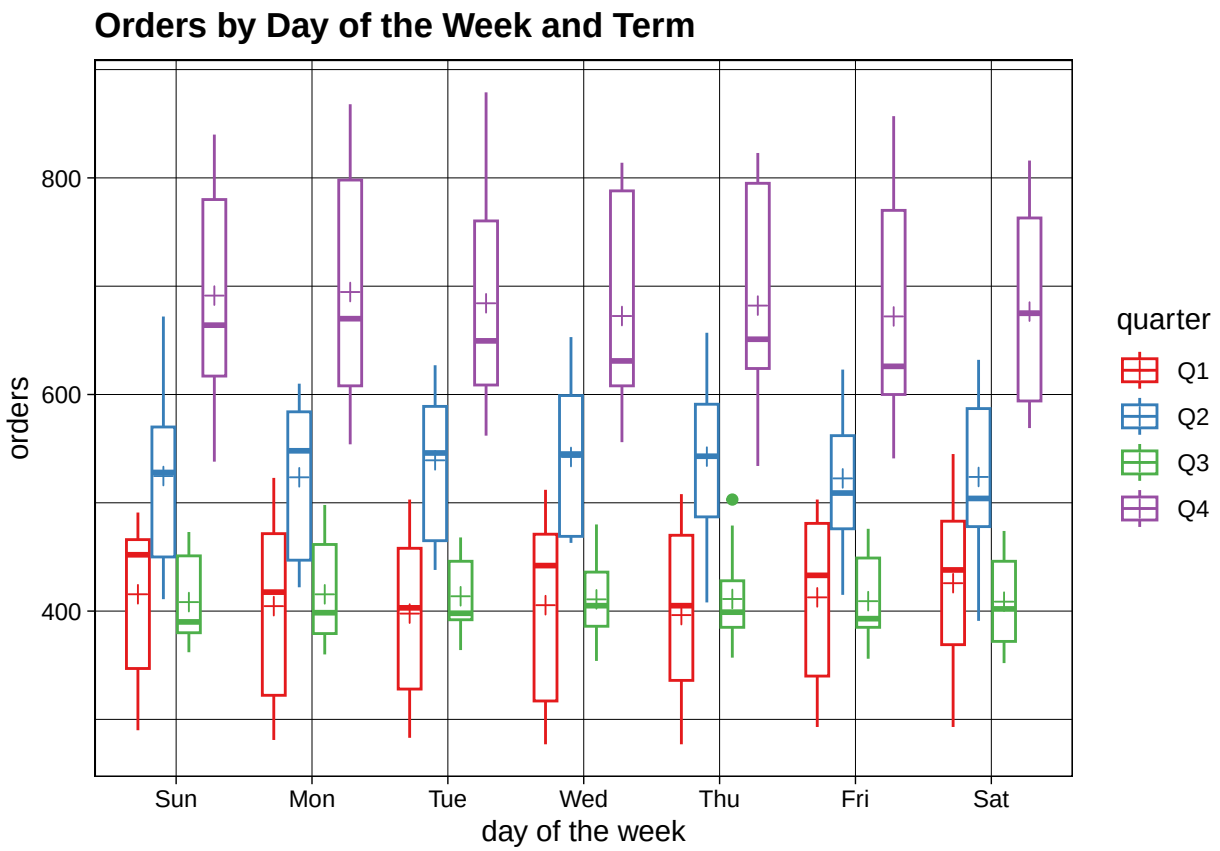



```
## Graph 3: MONTHLY AVERAGE ORDER BY TOP STATES

monthly_order_state <- sales %>%
  group_by(month,state) %>%
  summarise(total_order = sum(quantity_ordered),
            average_order = mean(quantity_ordered))
# create a line plot showing the monthly average quantity ordered for top products in 2019
ggplot(monthly_order_state,aes(x= month,y= total_order, color = state, group=state))+
  geom_line(linewidth = 0.8)+
  geom_point(size=1.8)+
  theme_linedraw()+
  labs(title="Monthly Orders Trends by US States in 2019",x="month",y="orders",color="state") +
  scale_color_brewer(palette="Set2")+
  scale_y_continuous(label = comma)+
  theme(plot.title = element_text(face = "bold"))
```

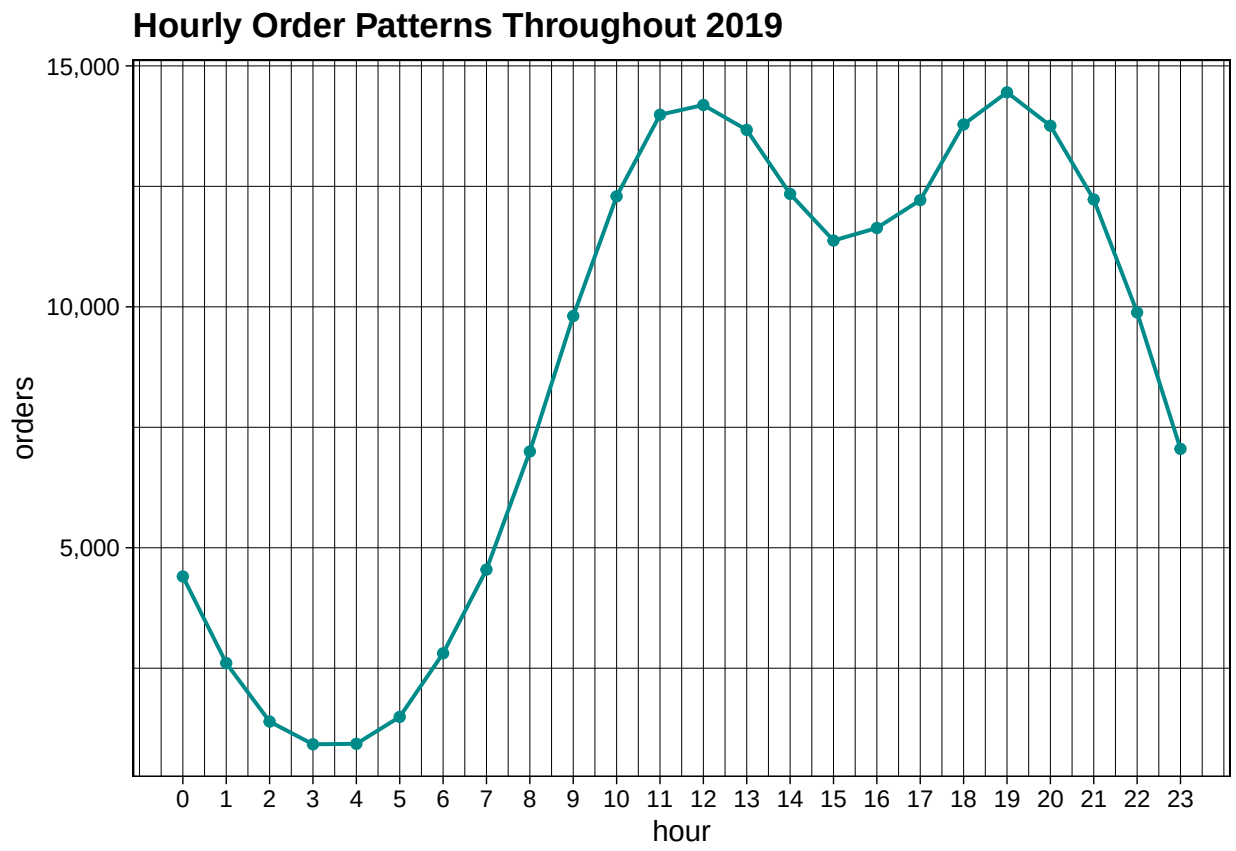



```
### e. Daily order trend vs daily average

## Graph 1: DAILY ORDER IN 2019

daily_order <- sales %>%
  group_by(order_date) %>%
  summarise(total_order = n())

# Create the plot
ggplot(daily_order) +
  geom_line(aes(x = order_date, y = total_order), linewidth = 0.5, color = "darkblue") +
  theme_linedraw() +
  labs(title = "2019 Daily Orders Trend", x = "date", y = "orders") +
  scale_x_date(date_labels = "%b", date_breaks = "1 month") +
  theme(plot.title = element_text(face = "bold"))
```

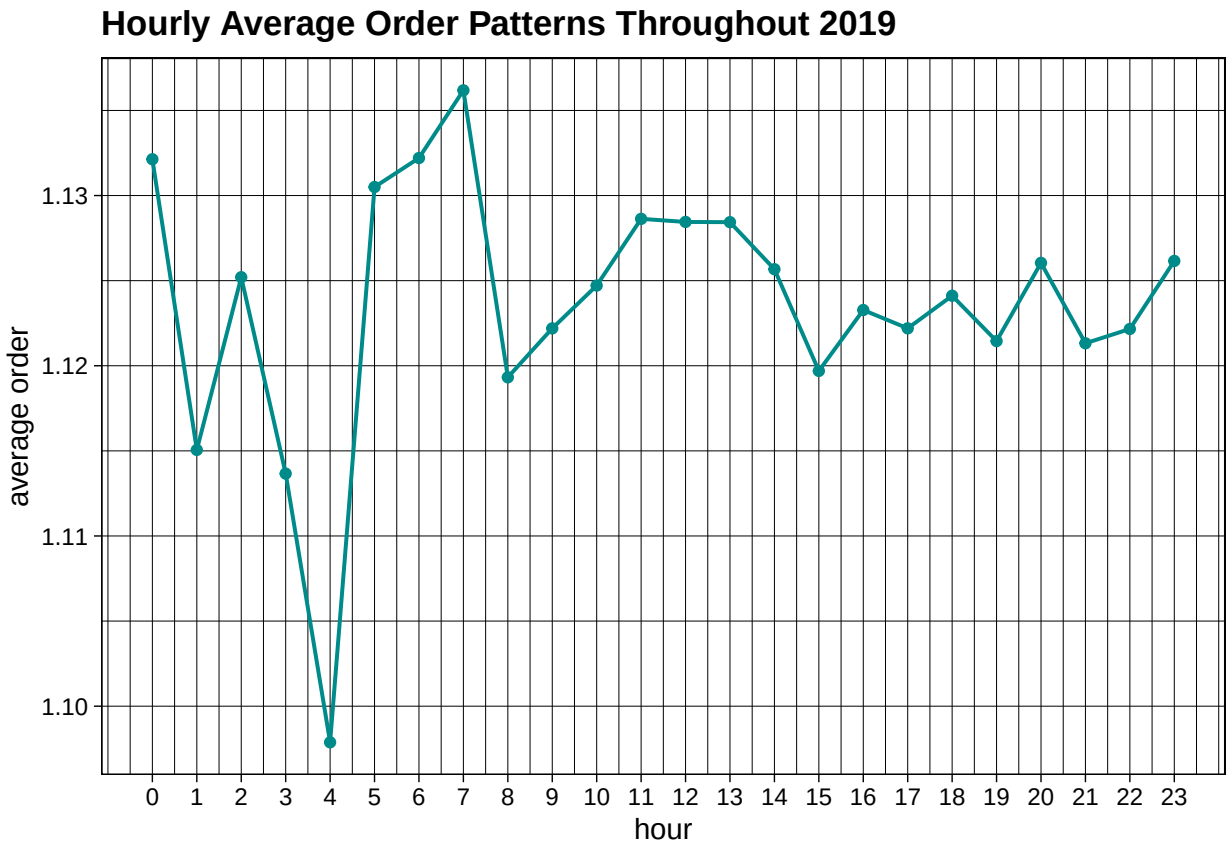

```
##### Week day effect

## Graph 2: AVERAGE WEEK DAY ORDER IN 2019

library(dplyr)
library(lubridate)
quarter <- function(date) {
  cut(date,
      breaks = as.Date(c("2019-01-01", "2019-04-01", "2019-07-01", "2019-10-01", "2020-01-01")),  # Ens
      labels = c("Q1", "Q2", "Q3", "Q4"),
      right = FALSE)  # Ensure intervals are inclusive of the start date
}
wday_order <- daily_order %>%
  mutate(quarter = quarter(order_date),
         wday = wday(order_date, label = TRUE))

ggplot(wday_order, aes(x = wday, y = total_order, color = quarter)) +
  geom_boxplot() +
  stat_summary(fun = mean, geom = "point", shape = 3, size = 2, fill = "white",position = position_dodg
  theme_linedraw() +
  labs(title = "Orders by Day of the Week and Term",
       x = "day of the week",
       y = "orders",
       color = "quarter")+
  theme(plot.title = element_text(face = "bold")) +
  scale_color_brewer(palette = "Set1")
```



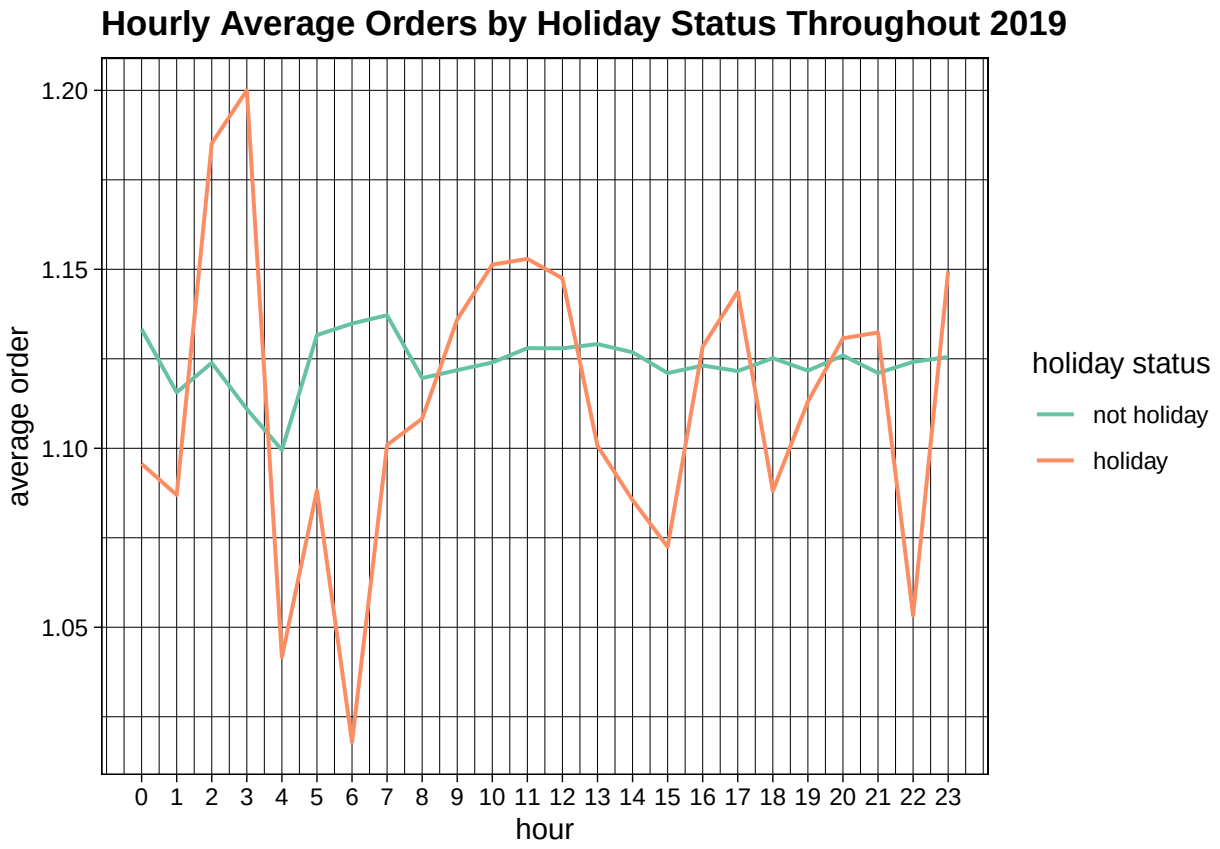

```
### f. Hourly order trend vs hourly average order

#
hourly_order <- best_year_sales %>%
  group_by(order_hour) %>%
  summarise(total_order = sum(quantity_ordered),
            average_order = mean(quantity_ordered))

ggplot(hourly_order)+
  geom_line(aes(x= order_hour,y= total_order),linewidth =0.7, color="darkcyan")+
  geom_point(aes(x= order_hour,y= total_order),size=1.5, color="darkcyan")+
  theme_linedraw()+
  scale_x_continuous(breaks = seq(0, 23, by = 1)) +
  scale_y_continuous(label = comma) +
  labs(title="Hourly Order Patterns Throughout 2019",x="hour",y="orders")+
  theme(plot.title = element_text(face = "bold"))
```



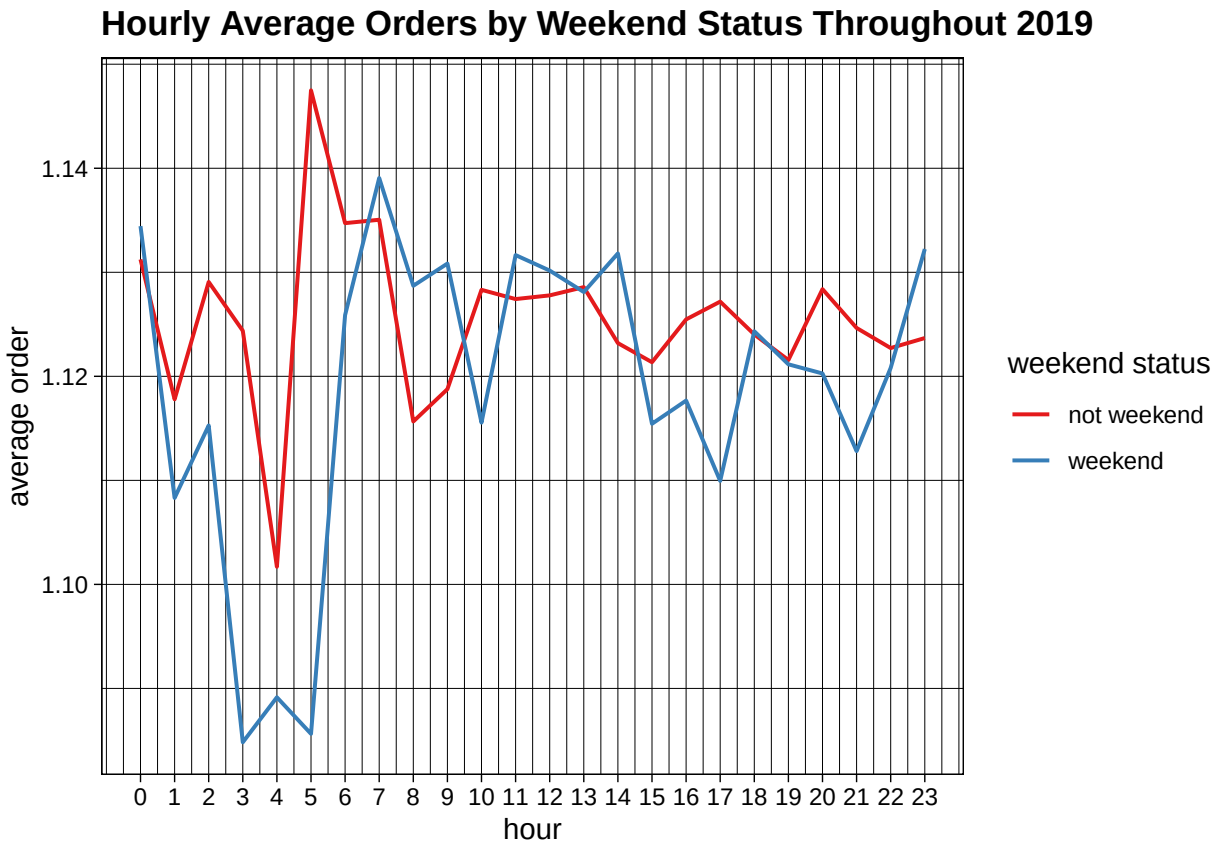

```
ggplot(hourly_order)+
  geom_line(aes(x= order_hour,y= average_order),linewidth =0.7, color="darkcyan")+
  geom_point(aes(x= order_hour,y= average_order),size=1.5, color="darkcyan")+
  theme_linedraw()+
  scale_x_continuous(breaks = seq(0, 23, by = 1)) +
  labs(title="Hourly Average Order Patterns Throughout 2019",x="hour",y="average order")+
  theme(plot.title = element_text(face = "bold"))
```




```
### Holiday effect
hourly_order_holiday <- sales %>%
  group_by(order_hour,is_holiday) %>%
  summarise(total_order = sum(quantity_ordered),
            average_order = mean(quantity_ordered)) %>%
  mutate(is_holiday = factor(is_holiday)) %>%
  mutate(is_holiday = recode(is_holiday,
                             "TRUE" = "holiday",
                             "FALSE" = "not holiday"))

ggplot(hourly_order_holiday)+
  geom_line(aes(x= order_hour,y= average_order,color=is_holiday),linewidth =0.7)+
  theme_linedraw()+
  scale_x_continuous(breaks = seq(0, 23, by = 1)) +
  labs(title="Hourly Average Orders by Holiday Status Throughout 2019",x="hour",y="average order",color=
  theme(plot.title = element_text(face = "bold")) +
  scale_color_brewer(palette = "Set2")
```




```
### Weekend effect

hourly_order_wday <- sales %>%
  group_by(order_hour,is_weekend) %>%
  summarise(total_order = sum(quantity_ordered),
            average_order = mean(quantity_ordered))%>%
  mutate(is_weekend = factor(is_weekend)) %>%
  mutate(is_weekend = recode(is_weekend,
                             "TRUE" = "weekend",
                             "FALSE" = "not weekend"))

ggplot(hourly_order_wday)+
  geom_line(aes(x= order_hour,y= average_order,color=is_weekend),linewidth =0.7)+
  theme_linedraw()+
  scale_x_continuous(breaks = seq(0, 23, by = 1)) +
  labs(title="Hourly Average Orders by Weekend Status Throughout 2019",x="hour",y="average order",color=
  theme(plot.title = element_text(face = "bold")) +
  scale_color_brewer(palette = "Set1")
```


Hourly Average Orders by Weekend Status Throughout 2019
average order
1.14
1.12
1.10
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
hour
weekend status
not weekend
weekend