# CSC 465

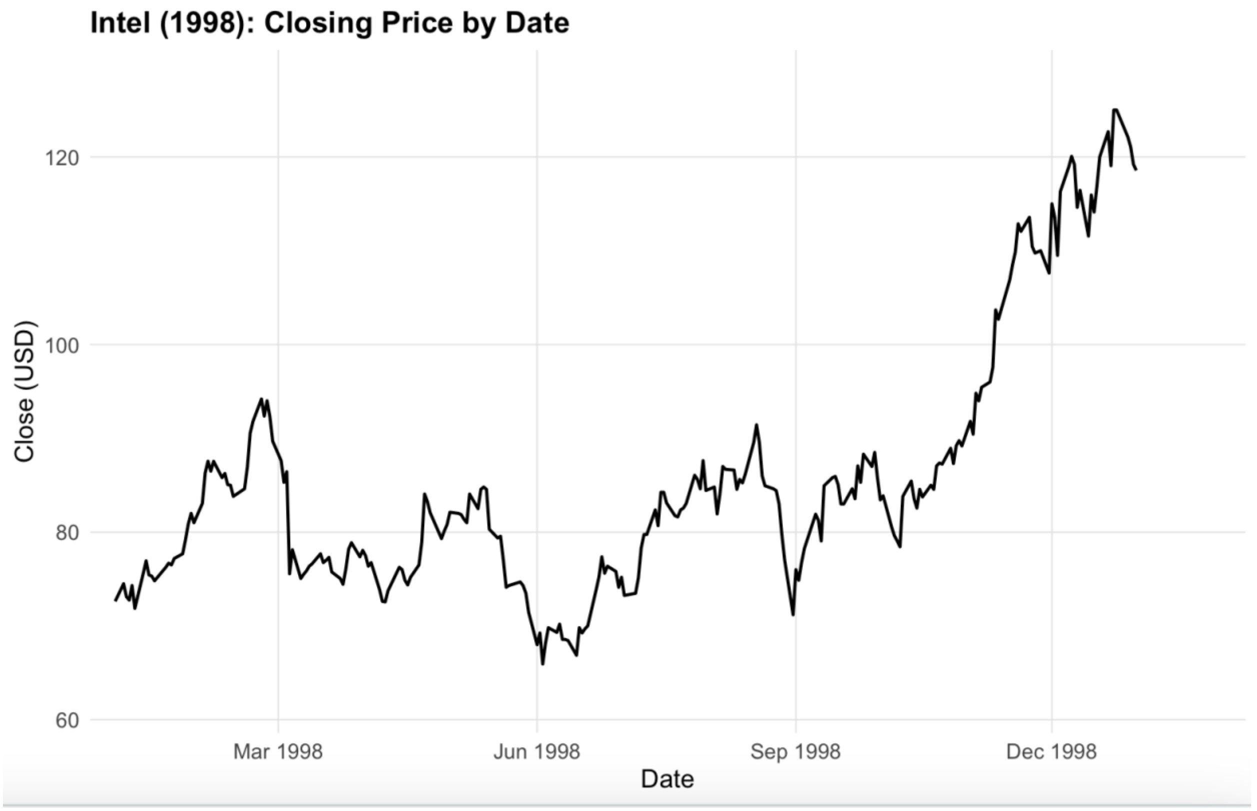
# Homework 1 (Ankita Kshirsagar )

1) For this problem, we'll look at data bout Intel stock(Intel-1998datasetfrom the website). The data covers stock market trading for the Intel corporation in 1998. Each row is a day, with the following columns: Date, Trading Day (integer day number, including skips), Open (price at market open), High (highest price of day), Low (lowest price of day), Close (price at market close), Volume (shares traded), and Adj. Close (adjusted closing price, meaning accounting for stock splits, which are not a problem in this data).

Make the specified graphs in either R or Tableau:

1. Graph the closing price vs. the date with an ordinary line graph. If you use Tableau, you need to right-click on the Date and choose Exact Date from the dropdown menu so that it uses the full date with "day".

**Closing Price vs. Date**



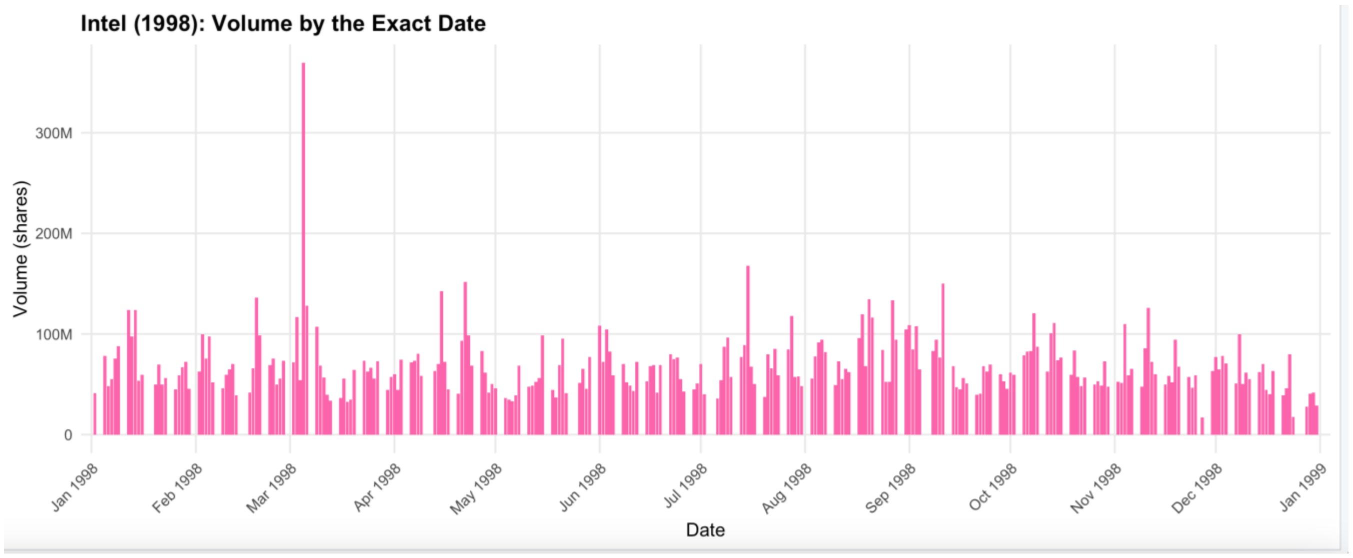
R Code :

```
# Load required libraries
library(lubridate) library(ggplot2) library(scales)

#load the data
Intel.1998$Date <- as.Date(Intel.1998$Date)

# Create the stock price line chart

stock_chart <- ggplot(Intel.1998, aes(x = Date, y = Close)) +

 geom_line(color = "black", size = 0.7) +

 labs(

  title = "Intel (1998): Closing Price by Date",

  x = "Date" ,   y = "Close (USD)  ) +

 theme_minimal() +

 theme(

  plot.title = element_text(size = 14, face = "bold", hjust = 0),

  axis.title.x = element_text(size = 12),

  axis.title.y = element_text(size = 12),

  axis.text = element_text(size = 10),

  panel.grid.minor = element_blank(),

  panel.grid.major = element_line(color = "gray90", size = 0.3)   ) +

 scale_x_date(

  date_breaks = "3 months",

  date_labels = "%b %Y",

  limits = c(as.Date("1998-01-01"), as.Date("1999-01-31")),

  expand = expansion(mult = 0.02)  ) +

 scale_y_continuous(

  breaks = seq(60, 140, 20),
  limits = c(60, 130),
  expand = expansion(mult = c(0.02, 0.02))  )

# Display the chart
print(stock_chart)
```

2. Graph the Volume vs. the exact Date as in the last part with a bar graph.

   Volume vs. the exact Date

   **Visualization :**



**R Code :**

```
# If intel_daily_1998 isn't already defined, point it to your loaded data frame:
if (!exists("intel_daily_1998")) intel_daily_1998 <- Intel.1998
# Ensure Date is a Date
if (!inherits(intel_daily_1998$Date, "Date")) {
  intel_daily_1998 <- intel_daily_1998 %>%
    mutate(Date = parse_date_time(Date, orders = c("mdy","ymd","dmy")) |> as_date())
}

intel_daily_1998 <- intel_daily_1998 %>% arrange(Date)

volume_by_day_plot_design <- ggplot(intel_daily_1998, aes(Date, Volume)) +
  geom_col(width = 0.9, fill = "hotpink") +
  scale_y_continuous(labels = label_number(scale_cut = cut_short_scale())) +
  scale_x_date(date_breaks = "1 month", date_labels = "%b %Y",
               expand = expansion(mult = c(0.01, 0.01))) +
  labs(title = "Intel (1998): Volume by Exact Date",
       x = "Date", y = "Volume (shares)") +
  theme_minimal(base_size = 13) +
  theme(panel.grid.minor = element_blank(),
        axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(face = "bold"))

print(volume_by_day_plot_design)

# Save the image (use the correct object)
ggsave("Intel_1998_Volume_by_Date.png",
       plot = volume_by_day_plot_design, width = 12, height = 5, dpi = 300)
```

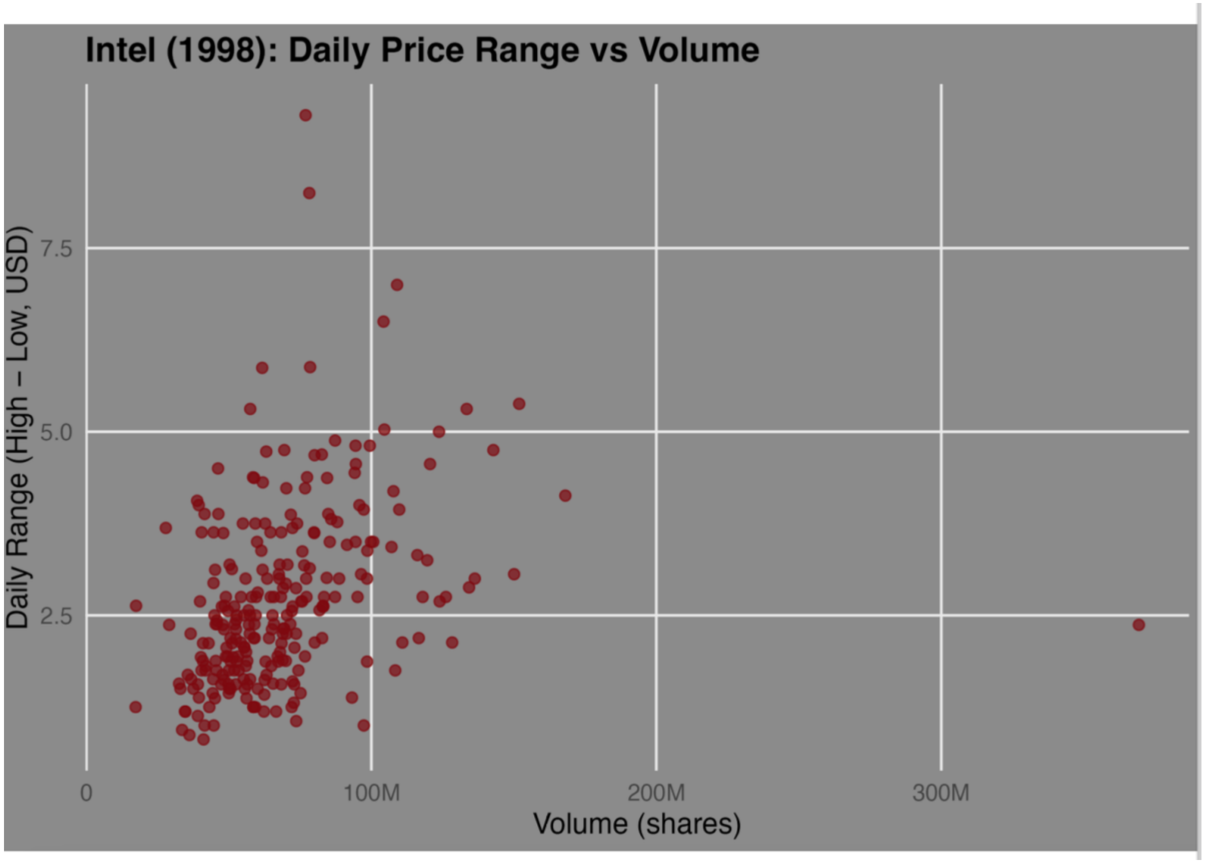
3. Create a scatterplot that graphs the Volume on the x-axis and the daily price

   range on the y-axis. You will need to create an additional column that contains the "range" of the prices for the day as the difference between the fields High and Low.

   Range = High – Low

   Tableau can do it with a Calculated Field. In R you can do it by making a new column equal to the result from subtracting the two columns. In Tableau, to get a scatter plot, you will need to right click on both the Range and Volume entries in graph and change them to "Dimensions".

Scatter Plot : Daily price Range vs Volume



RCode :

```r
# --- Packages ---

pkgs <- c("dplyr","ggplot2","lubridate","scales")

new <- setdiff(pkgs, rownames(installed.packages()))

if (length(new)) install.packages(new, repos = "https://cloud.r-project.org")

library(dplyr); library(ggplot2); library(lubridate); library(scales)

stock_daily_1998 <- if (exists("intel_daily_1998")) intel_daily_1998 else Intel.1998

if (!inherits(stock_daily_1998$Date, "Date")) {

  stock_daily_1998 <- stock_daily_1998 %>%

    mutate(Date = parse_date_time(Date, orders = c("mdy","ymd","dmy")) |> as_date())

}

stock_daily_1998 <- stock_daily_1998 %>%

  arrange(Date) %>%

  mutate(price_range_usd = High - Low)

range_vs_volume_plot <- ggplot(stock_daily_1998, aes(x = Volume, y = price_range_usd)) +

  geom_point(color = "#8B0000", alpha = 0.8, size = 2) +  # dark red points

  scale_x_continuous(labels = label_number(scale_cut = cut_short_scale())) +

  labs(

    title = "Intel (1998): Daily Price Range vs Volume",

    x = "Volume (shares)",

    y = "Daily Range (High − Low, USD)") +

  theme_gray(base_size = 13) +

  theme(

    panel.background = element_rect(fill = "grey30", color = NA),

    plot.background  = element_rect(fill = "grey30", color = NA),

    panel.grid.major = element_line(color = "white", linewidth = 0.6),
    panel.grid.minor = element_blank(),
    axis.text       = element_text(color = "white"),
    axis.title      = element_text(color = "white"),
    plot.title      = element_text(color = "white", face = "bold")
  )
# Save image
ggsave("Q1c_volume_vs_range_image.png",
      plot = range_vs_volume_plot, width = 9, height = 6, dpi = 300)
```

2) Use Tableau for this question. Open the GM cars dataset included with this assignment (gmcar_price.txt). Each row represents a different car that was sold and includes information about features like the mileage and the price of sale. Hint: use the "Show Me" menu.

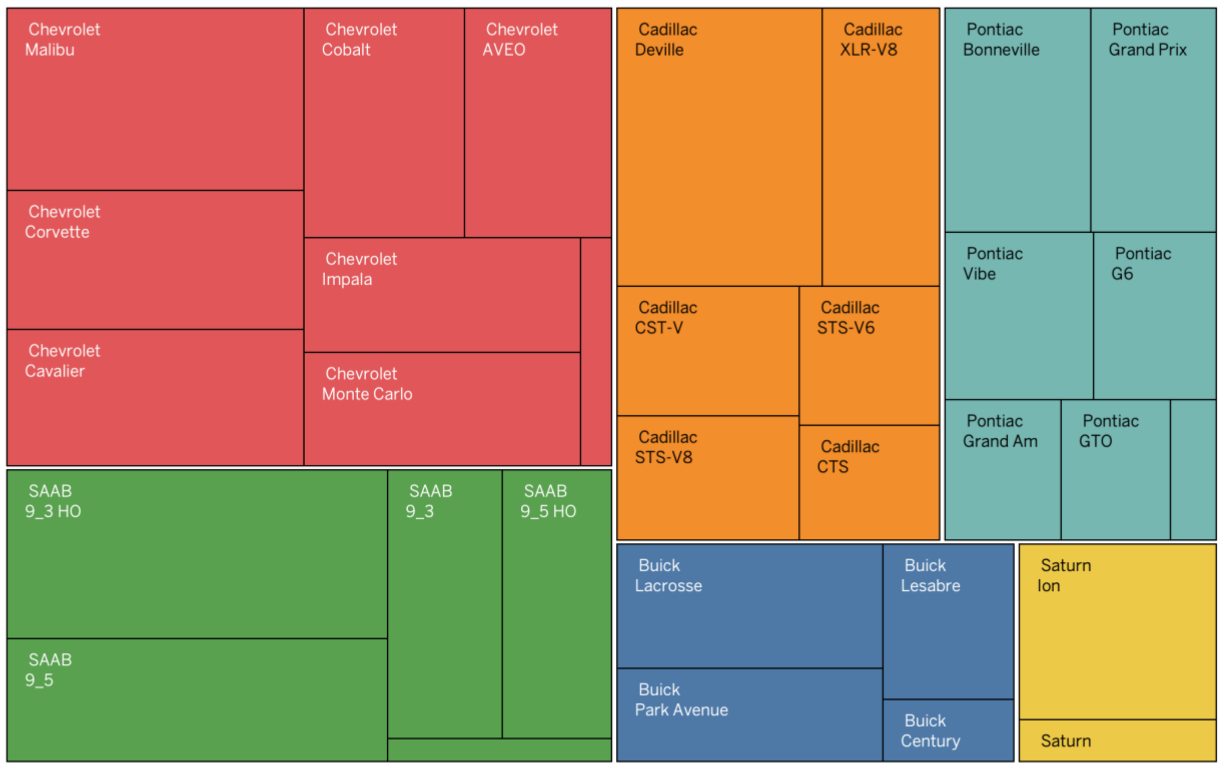
a. A treemap based on Price with a main subdivision for the Make of the car and a minor subdivision based on the Model. Because each row of the data file represents a single car but each box in the treemap represents all the cars with a given make and model, pay very close attention to what kind of aggregation is being used.

   Tablaue Steps:

   1.Open Tableau and connect to **gmcar_price.txt** (Text file) → which I downloaded from assignment folder
   2. Go to a new worksheet (**Sheet 1**).
   3. In the left **Data** pane, **selected** fields: **Make**, **Model**, **Price**.
   4. click on show me icon and choose Treemap
   5. Added AVG(Price) or SUM(Price) as per visualization requirement
   6.set up color and nested color from multiple options
   7. checked its correct or not output
   8.Chnaged sheet name and Exported that as PDF

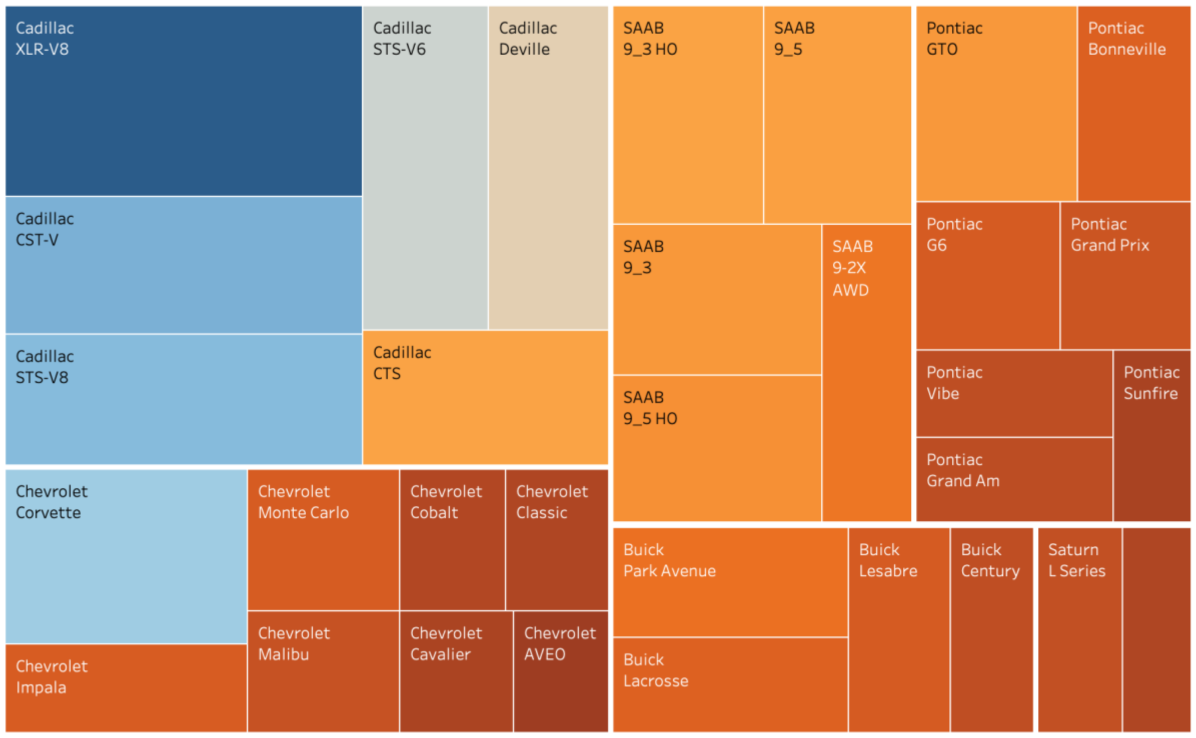
Using sum() :

TreeMap

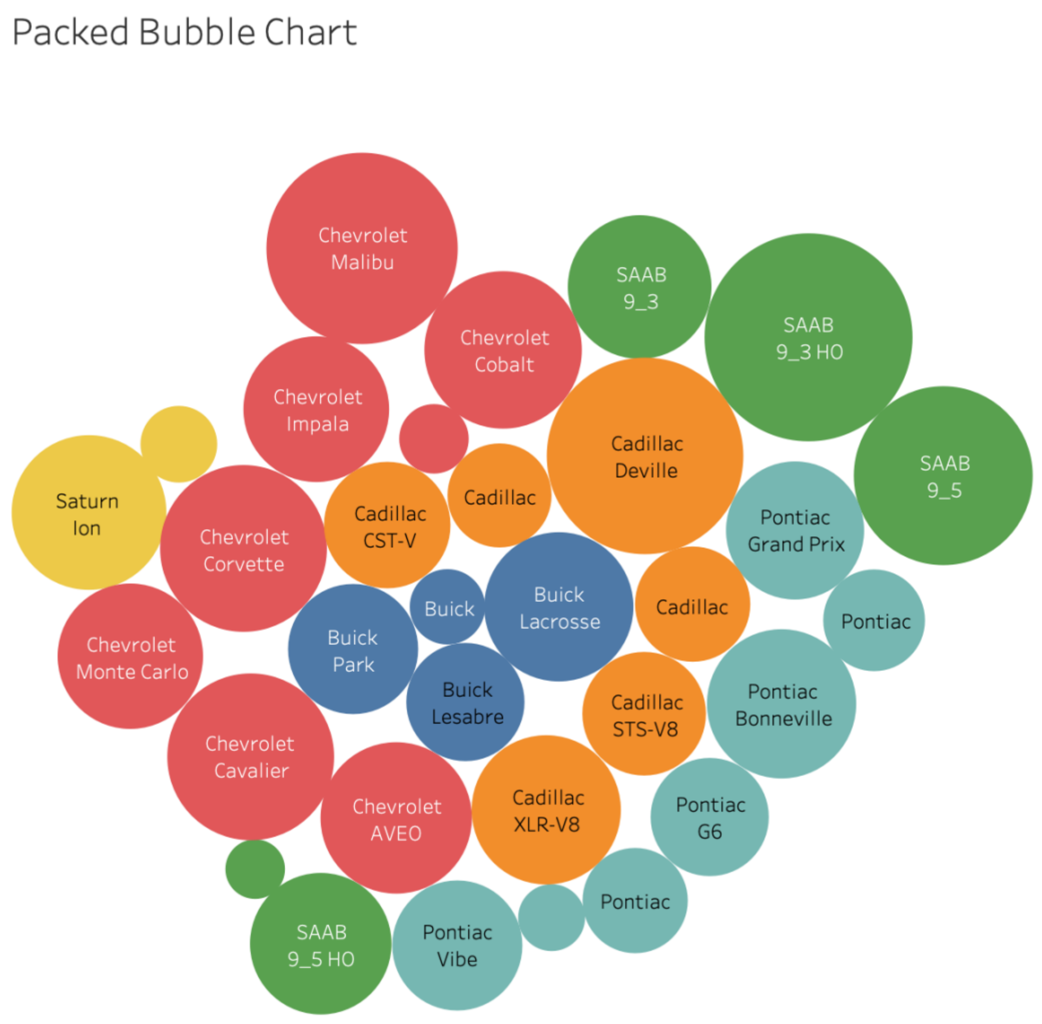

Chevrolet Malibu
Chevrolet Cobalt
Chevrolet AVEO
Chevrolet Corvette
Chevrolet Impala
Chevrolet Cavalier
Chevrolet Monte Carlo
Cadillac Deville
Cadillac XLR-V8
Cadillac CST-V
Cadillac STS-V6
Cadillac STS-V8
Cadillac CTS
Pontiac Bonneville
Pontiac Grand Prix
Pontiac Vibe
Pontiac G6
Pontiac Grand Am
Pontiac GTO
SAAB 9_3 HO
SAAB 9_3
SAAB 9_5 HO
SAAB 9_5
Buick Lacrosse
Buick Lesabre
Buick Park Avenue
Buick Century
Saturn Ion
Saturn

Using Average avg() method :

Packed Bubble Chart



b. A packed bubble chart of the same type.



Steps :

1. Already loaded file in previous question used it as it is
2. Take New worksheet.
3. Select Model, Make, Price - Show Me - Packed Bubbles.
4. Set Size = SUM(Price)
5. Put Make on Color and Model on Label (format Price as currency if labeled).
6. Clean formatting (palette, label size) and verify one bubble per model.
7. Rename the sheet and exported the PDF

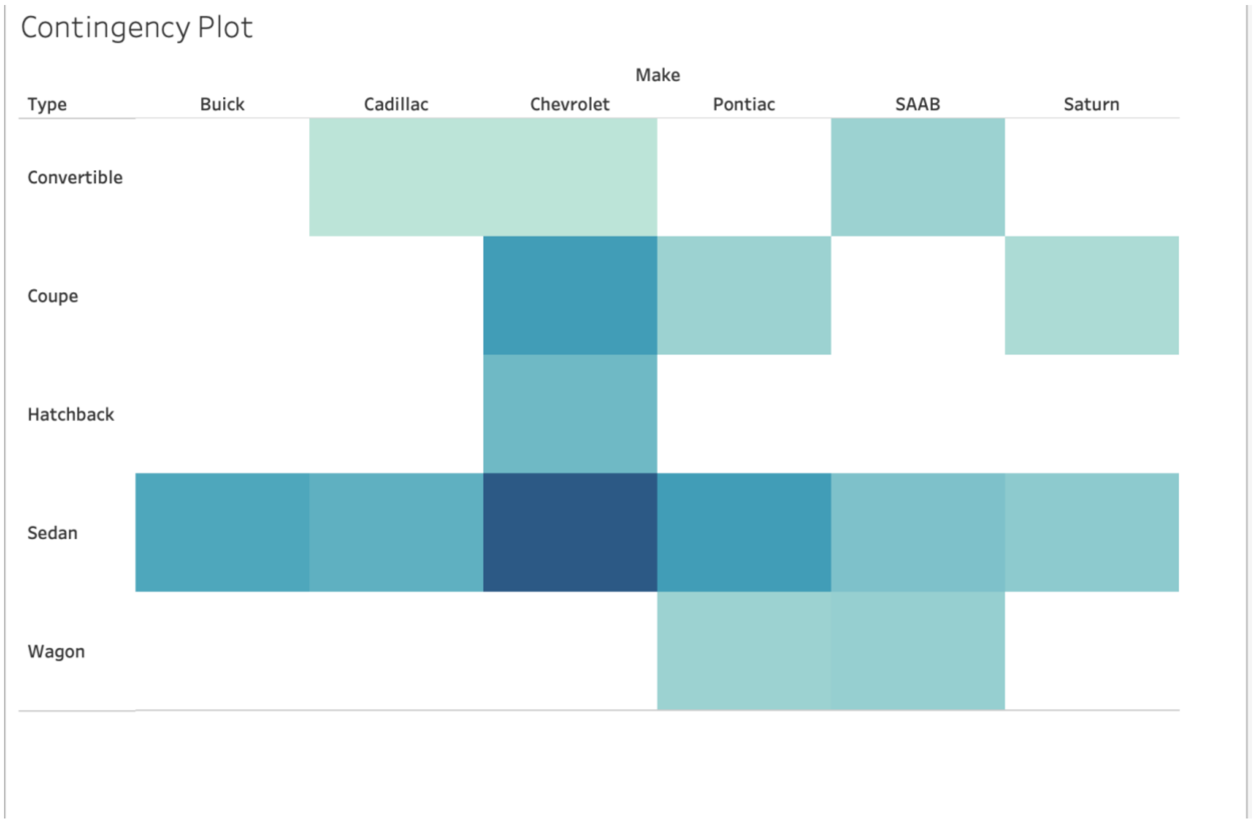
c. Write a short paragraph discussing the differences between the two plots. Describe for each something that displayed more clearly than with the other.

Both Treemap and Packed Bubbles visualize the same gmcar_price.txt data - SUM(Price) by Make – Model ( Rectangle blocks replaced by circles in packed bubble ) but they highlight different things. The tree map excels at precise size comparison: each rectangle's area is proportional to total price, and the nested layout makes the hierarchy (Make as the outer group, Model inside) immediately clear and space efficient. This makes it easier to judge 'market share' and compare makes and models accurately. Packed bubbles, on the other hand, are less precise but more attention-grabbing: very large or very small models pop out quickly, and loose clustering can hint at patterns or outliers that may be less noticeable in the tree map. In short, use the tree map for accurate, hierarchical comparison, and packed bubbles for salient extremes and quick pattern spotting.

d. Create a contingency plot (Tableau calls it a heat map under Show Me) showing with color the number of cars (Number of Records) of each Type sold by each Make. Explain at least one observation about that data that this chart makes it easy to see.

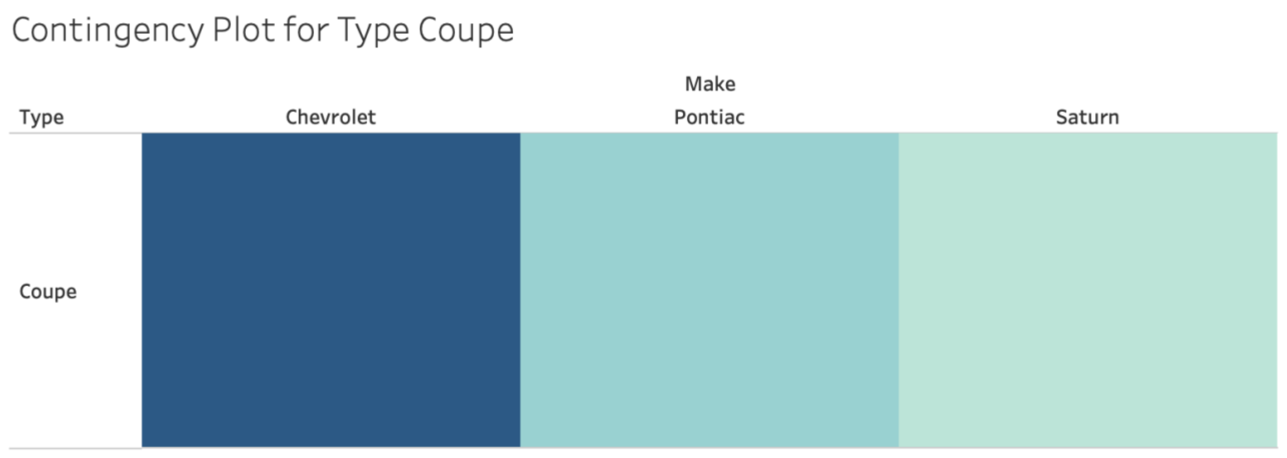
Steps:

1. Used already loaded dataset or file gmcar_price.txt
2. Openthe new worksheet.
3. Drag Make to Columns and Type to Rows.
4. Click Show Me then click on Heat Map
5. Drag Number of Records column in tableu which it generate automatically (or the table's (Count) field) to Color and set it to SUM.
6. Drag SUM(Number of Records) to Label to show counts.
7. Color - Edit Colors... selected a single-hue palette
8. add a title and export.



**One Observation:**

The coupe row shows a clear pecking order. Chevrolet stands out with the deepest shading they have the most coupes in this dataset. Pontiac participates but at a smaller scale, and Saturn's presence is minimal. Buick, Cadillac, and SAAB are completely absent from coupes here.
That hints at different strategies: Chevy treats coupes as volume, Pontiac as performance, Saturn as an entry-sport option. Luxury or conservative brands (Cadillac, Buick, SAAB) seem to skip traditional coupes, likely betting on other body styles.
You would not see this as clearly in the tree map or bubbles - the heat map isolates the coupe category and makes the brand gaps obvious.



3)

This problem works with a dataset containing the population of Montana and of each of the 7 Native American reservations within it (reservation70-00.xlsx). There is a measurement for each decade between 1970 and 2000. Sheet1 has the original data.

We will use Tableau for this question, but Sheet1 has a header that confuses Tableau. If you're interested, check out the "Data Interpreter" feature in Tableau to learn how to deal with this. Otherwise, use Sheet2, where I've removed the header. We need a few transformations to get the data ready to work with:

1. Renaming the 1970* field so it has no * and can be converted to a number

2. "Pivoting" the year fields in a similar manner to how it was demonstrated in the tutorial.

3. Changing the name of the pivot fields to Year and Population, and changing the type of the year field to "whole number".

4. You can also hide the "Percent Change" field as it only contains information for change over the entire period, not per decade.

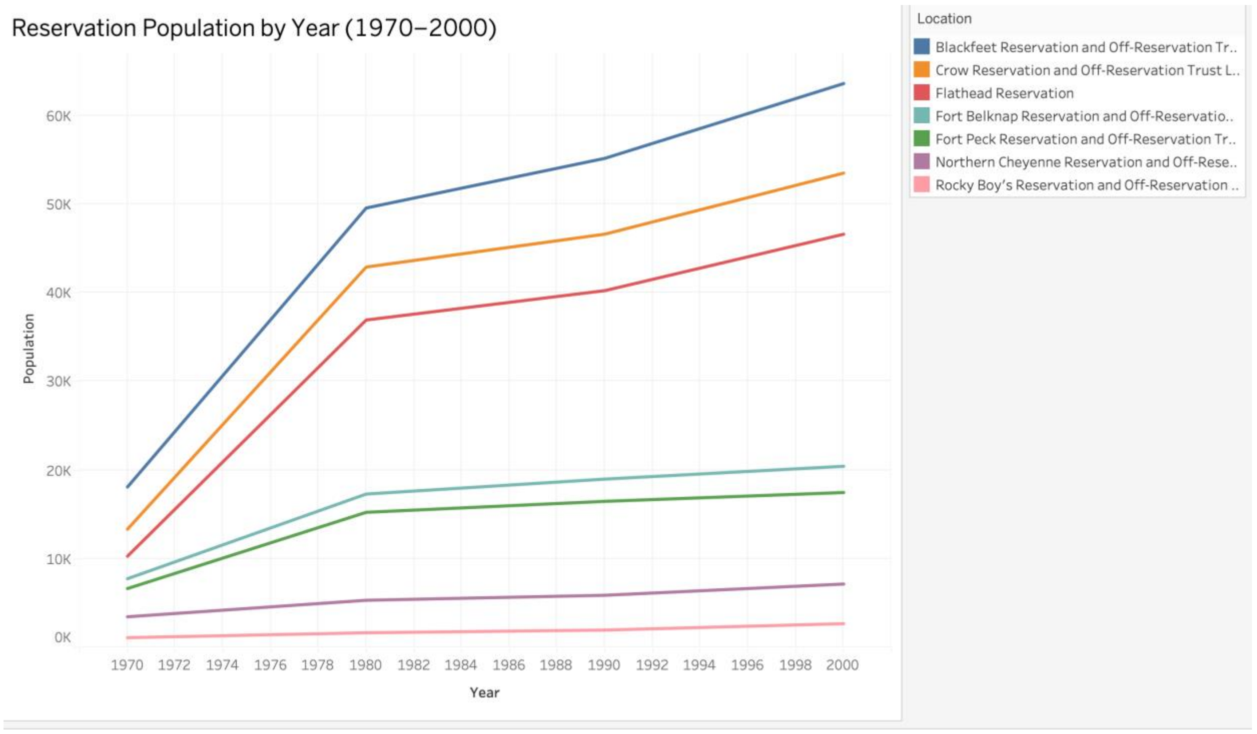
5. If you would like to have an actual Date field for the Year, so that it is treated by Tableau as a time instead of just a number, you need to create a "Calculated Field". It should construct a Date using the Year, i.e. make a Date field that is on January 1 of the specified year:

makedate([Year], 1, 1)

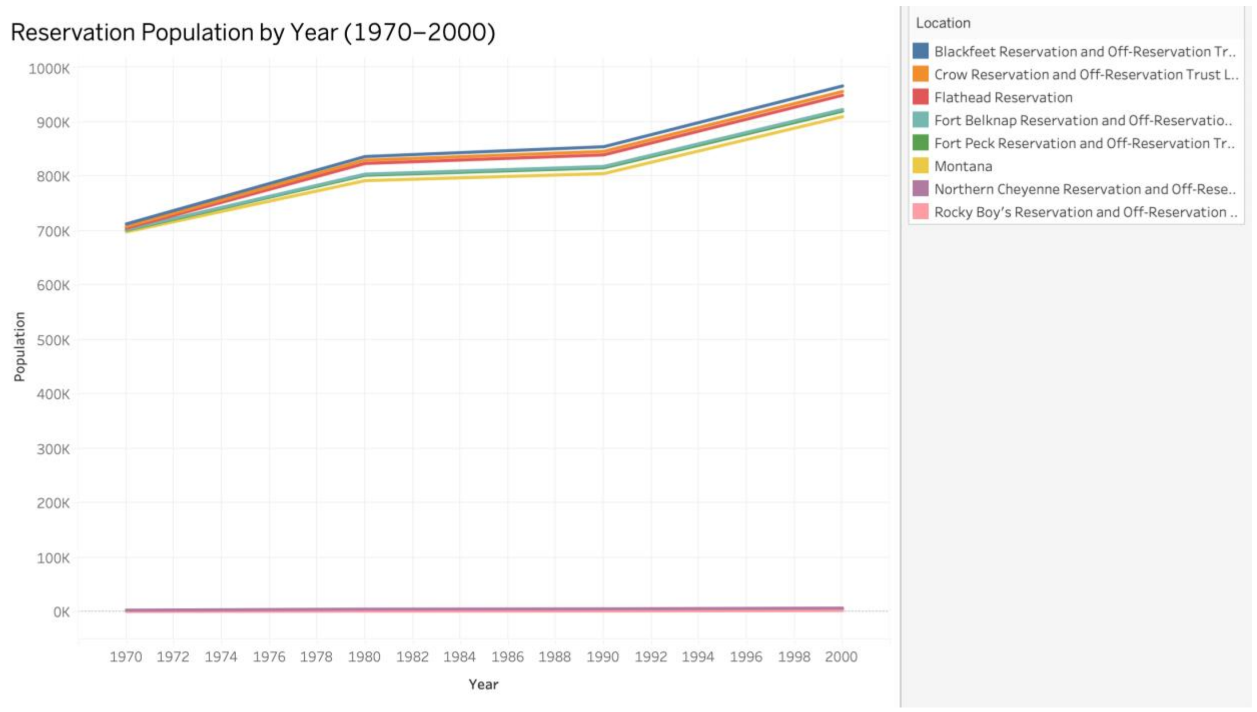
6. We are not interested in the Montana population, only the reservation populations. When you have used Location on your graph, you can right mouse click (or click the down arrow within it) to apply filters. You can also use "Exclude" from the right click menu on the legend just below the "Marks" configuration.

Create graphs to show the following information, using appropriate graph types. Make sure that the graphs are properly labeled and that the axis scales properly reflect the type of data represented.

a. One chart that graphs the population growth over the years for the individual reservations. - Applied Filter : Excluded Location : Montana



without applying any filter :

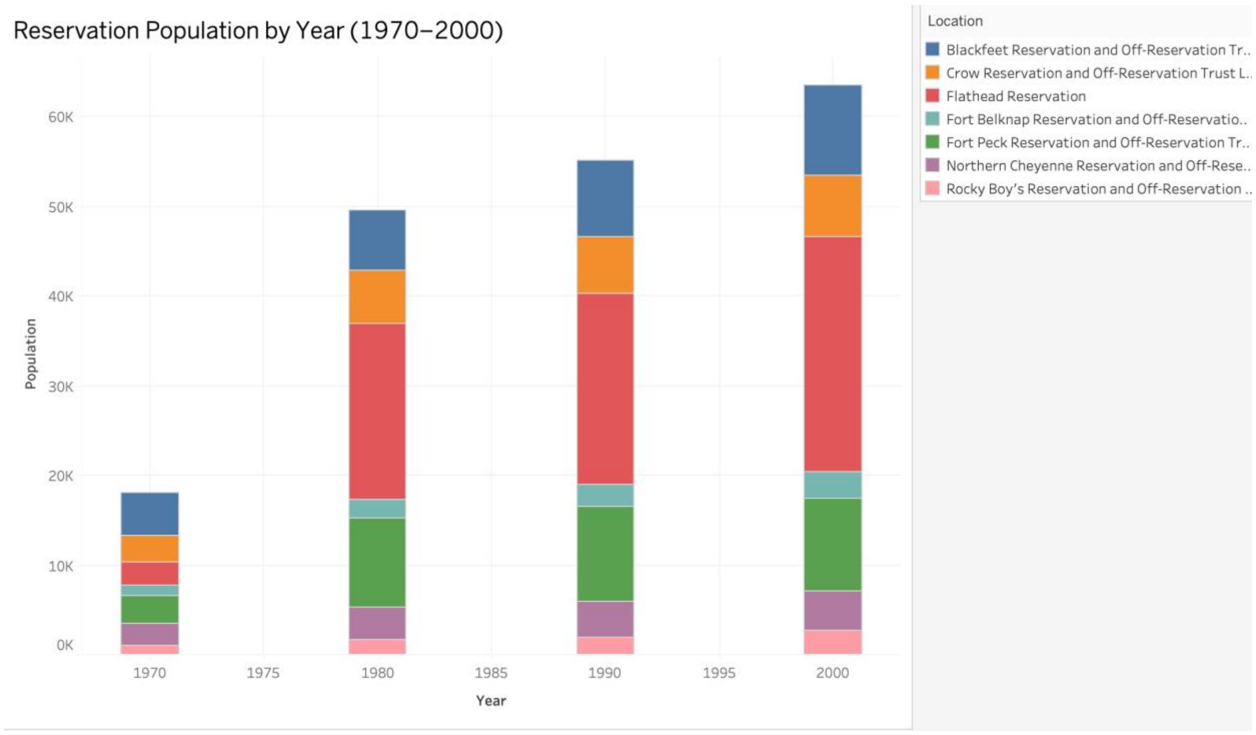


Steps :

1. Connect to reservation70-00.xlsx and opened Sheet2 (without header issue).
2. In the Data Source grid: rename 1970* - 1970; select columns 1970, 1980, 1990, 2000 - Pivot . Rename Pivot Field Names - Year, Pivot Field Values → Population; set Year = Number (Whole) and Population = Number (Whole).
3. Data - Create Calculated Field… - Year Date = MAKEDATE([Year],1,1) (type: Date).
4. Then new Worksheet: drag Year Date to Columns - drag SUM(Population) to Rows and set Marks = Line.
5. Drag Location to Color
6. drag Location to Filters and uncheck or exclude 'Montana' and one I did without adding any filter as well .
7. change the title of sheet
8. Export: Worksheet -Export - Image or pdf

b. One that graphs the total reservation population for each year, subdivided among the different reservations. The difference between this and (a) is that in (b) we are not looking only at each population individually but at the growth of the total population of all of them together, then subdivided by the reservations.

Applied Filter : Excluded Location : Montana



Steps :
Followed same above steps only changed type as bar chart and kept location filter as it is .

**4)** For this question,answeronlywithtext.Youmayincludeanillustrationifyou would like, but you do not need to visualize data for this question.

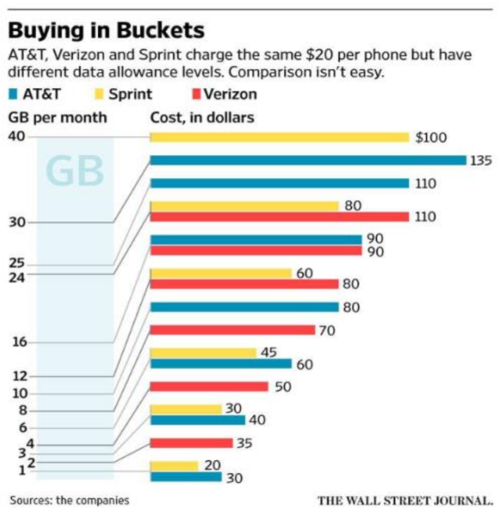
a. Explain what we mean by 'pre-attentive' attributes. Are these as effectively recognized by human perception when they are used in combinations?

Pre-attentive attributes are the visual cues we notice in a split second, before we consciously focus. Think position on a scale, length, size, tilt, color (hue/brightness), shape, borders, motion, or even dot vs. line. They are most effective when a single, clear cue carries the message - quick to read and hard to misinterpret.
Define a target by a combination of cues (for example, red and vertical) and that instant 'pop-out' weakens, forcing slower, step-by-step searching. Pile on too many cues and they start competing, so less actually stands out.
A smarter approach is redundant coding: use two cues to reinforce the same category (like color plus shape), which also helps people with color-vision differences.
When combining, choose separable pairs (hue with shape or position) rather than tightly linked ones (two kinds of size) to keep perception effortless.

b. Use Weber's Law to explain why it is important to include 0 in the numerical axis of a bar chart.

Weber's Law says we notice differences relative to a starting level ($\Delta I$ / I), not as absolute amounts. In a bar chart, length stands for magnitude.  if the value axis does not begin at zero, the baseline is smaller and the same numeric change takes up a larger share of the bar, visually overstating small differences. Starting at zero preserves the true proportion between lengths and values, so what you see matches the data and typical just-noticeable difference limits. Bottom line: use a zero baseline for bars; if you need to emphasize small fluctuations around a nonzero level, switch to a line, slope, or dot plot with error bars.

5) This graph of cell phone pricing plans is not very easy to use. Use R for this question and recreate this graph in two different ways of your choice. For each one, explain what you are trying to help the user see. For example, one might be to compare the cell phone companies to see what kind of plans they have. Another might be best for examining the trend of the relationship between price and data bandwidth. That relationship may hold overall, or you could look to see if it is different per company. You can decide what to visualize, i.e. what question to answer with your visualization, but make sure to explain what this visualization should be showing. To get full credit, you must produce a graph which makes the answer to your question immediately clear. It must also be well implemented, i.e. following the guidelines at the top for a clean graph.



You do not need to type in all the values by hand. Here is R code that makes a dataframe with these values in it:

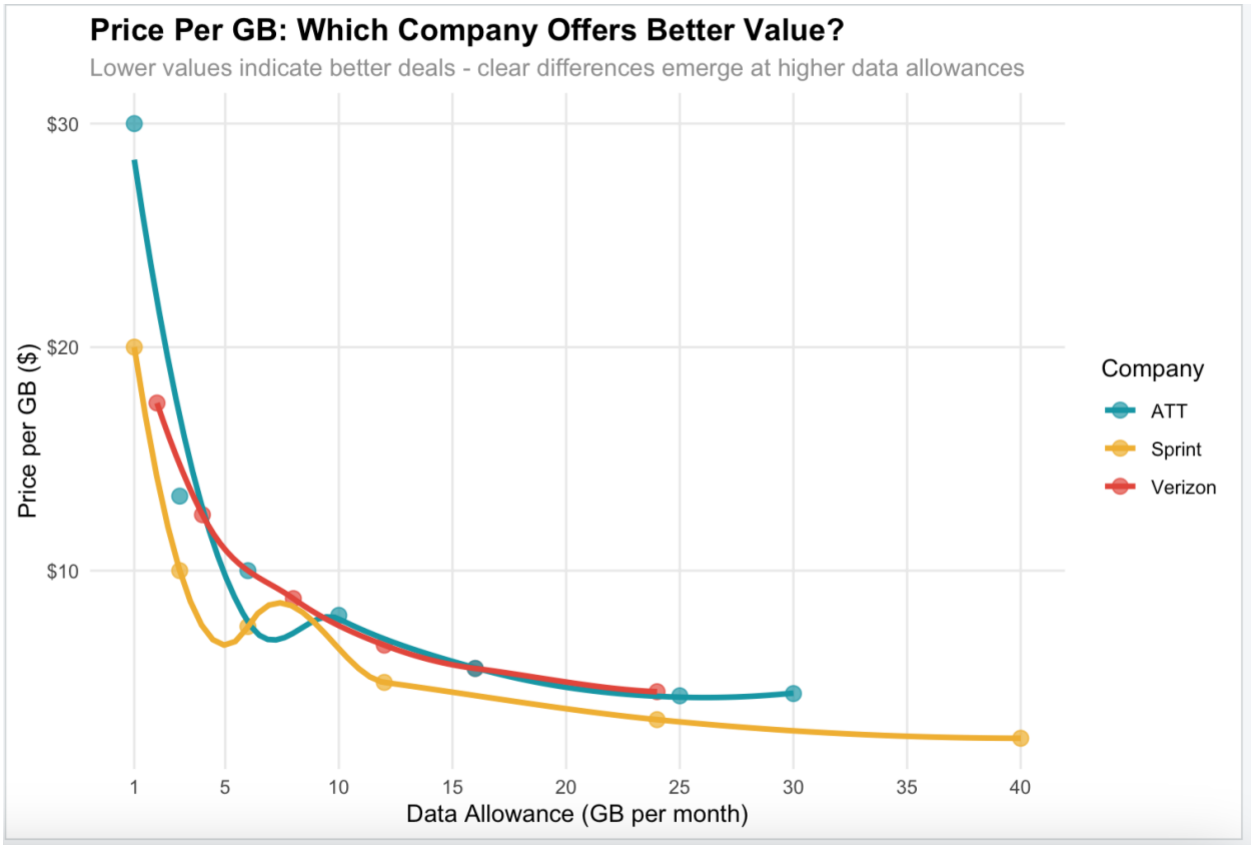
```
cellPlans = data.frame(
    c("ATT", "Sprint", "Verizon", "ATT", "Sprint",
      "Verizon", "ATT", "Sprint", "Verizon", "ATT",
      "Verizon", "Sprint", "Verizon", "ATT",
      "Verizon", "Sprint", "ATT", "ATT", "Sprint"),
    c(1, 1, 2, 3, 3, 4, 6, 6, 8, 10, 12, 12, 16, 16,
      24, 24, 25, 30, 40),
    c(30, 20, 35, 40, 30, 50, 60, 45, 70, 80, 80, 60,
      90, 90, 110, 80, 110, 135, 100))

names(cellPlans) = c("Company", "DataGB", "Price")
```

Question Answered : Which company offers the best value (lowest cost per GB) at each data tier

## Visualization 1 : Visualization 1: Price-per-GB Analysis

This scatter plot with smooth trend lines shows how cost per GB changes as you move to bigger data plans.
Sprint usually comes out cheapest across most sizes, AT&T tends to get less competitive at the high end, and Verizon sits in the middle with fairly steady pricing.
Because the lines are continuous, you can see who actually rewards bigger plans with lower $/GB and who does not.
It's a big-picture read, not a tier-by-tier checklist: you're looking at each company's pricing trajectory as usage grows.

Bottom line: it helps a budget-minded buyer spot which carrier delivers better value across the whole range, not just at one data bucket.

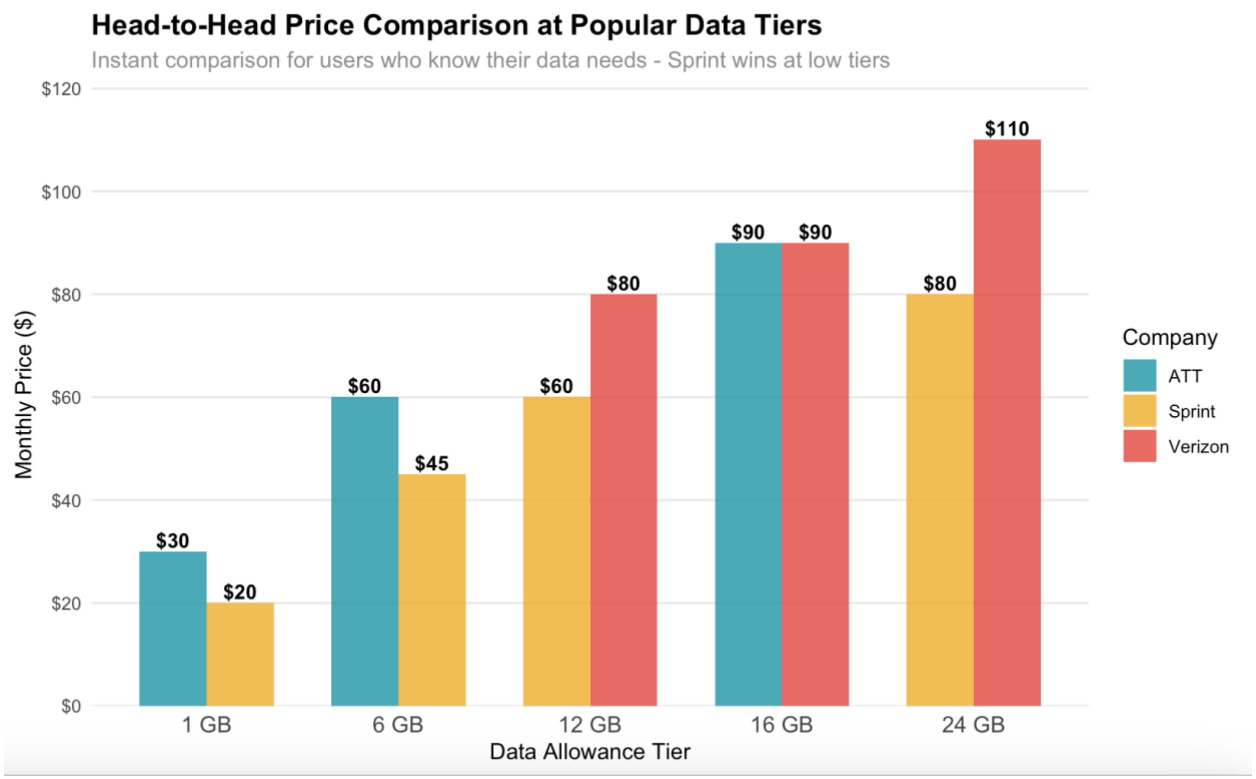


R Code :

```
> # Load necessary libraries

> library(ggplot2) library(dplyr).  library(scales)

> # Create the cell phone dataset

> cellphone_data = data.frame(

+    c("ATT", "Sprint", "Verizon", "ATT", "Sprint",

+      "Verizon", "ATT", "Sprint", "Verizon", "ATT",

+      "Verizon", "Sprint", "Verizon", "ATT",
+     "Verizon", "Sprint", "ATT", "ATT", "Sprint"),
+   c(1, 1, 2, 3, 3, 4, 6, 6, 8, 10, 12, 12, 16, 16,
+     24, 24, 25, 30, 40),
+   c(30, 20, 35, 40, 30, 50, 60, 45, 70, 80, 80, 60,
+     90, 90, 110, 80, 110, 135, 100))

> names(cellphone_data) = c("Company", "DataGB", "Price")
> # Calculate price per GB for better value analysis
> value_analysis <- cellphone_data %>%
+   mutate(PricePerGB = Price / DataGB)
> price_per_gb_plot <- ggplot(value_analysis, aes(x = DataGB, y = PricePerGB, color =
Company)) +
+   geom_point(size = 3, alpha = 0.7) +
+   geom_smooth(method = "loess", se = FALSE, size = 1.2) +
+   scale_color_manual(values = c("ATT" = "#00A0B0", "Sprint" = "#F0B90B", "Verizon" =
"#E74C3C")) +
+   labs(
+     title = "Price Per GB: Which Company Offers Better Value?",
+     subtitle = "Lower values indicate better deals - clear differences emerge at higher
data allowances",
+     x = "Data Allowance (GB per month)",
+     y = "Price per GB ($)",
+     color = "Company"
+   ) +
+   theme_minimal() +
+   theme(
+     plot.title = element_text(size = 14, face = "bold"),
+       plot.subtitle = element_text(size = 11, color = "gray60"),
+       legend.position = "right",
+       panel.grid.minor = element_blank(),
+       panel.grid.major = element_line(color = "gray90", size = 0.3)
+    ) +
+    scale_x_continuous(breaks = c(1, 5, 10, 15, 20, 25, 30, 35, 40)) +
+    scale_y_continuous(labels = scales::dollar_format())
> # print the chart
> print(price_per_gb_plot)
```

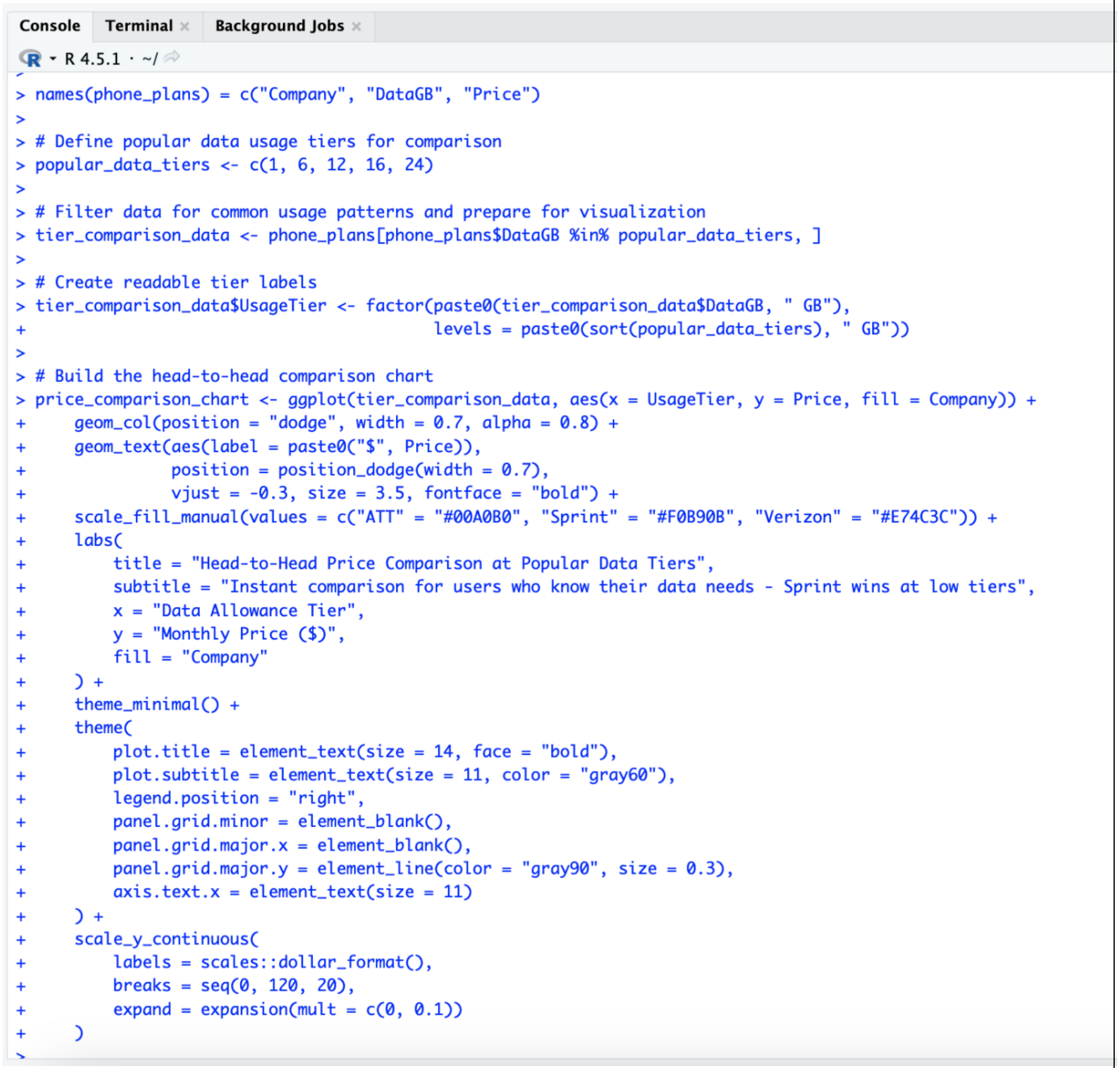
## Visualization 2: Price per GB Visualization

This grouped bar chart shows cost per gigabyte for each carrier at every data tier, so you're judging value rather than raw prices. Shorter bars mean better value, making the best option at any tier immediately obvious.
If you are shopping for, say, 12 GB, you just find that group and pick the shortest bar.
The chart replaces the confusing bucket graphic with a simple head-to-head value comparison.
It highlights the 'winner' at each tier without any mental math.
Bottom line: it helps to choose the most cost-effective plan for your specific data need.



Head-to-Head Price Comparison at Popular Data Tiers
Instant comparison for users who know their data needs - Sprint wins at low tiers
Monthly Price ($)
$120
$100
$80
$60
$40
$20
$0
$30
$20
$60
$45
$60
$80
$90
$90
$80
$110
1 GB
6 GB
12 GB
16 GB
24 GB
Data Allowance Tier
Company
ATT
Sprint
Verizon

R code :



```
> names(phone_plans) = c("Company", "DataGB", "Price")
>
> # Define popular data usage tiers for comparison
> popular_data_tiers <- c(1, 6, 12, 16, 24)
>
> # Filter data for common usage patterns and prepare for visualization
> tier_comparison_data <- phone_plans[phone_plans$DataGB %in% popular_data_tiers, ]
>
> # Create readable tier labels
> tier_comparison_data$UsageTier <- factor(paste0(tier_comparison_data$DataGB, " GB"),
+                                          levels = paste0(sort(popular_data_tiers), " GB"))
>
> # Build the head-to-head comparison chart
> price_comparison_chart <- ggplot(tier_comparison_data, aes(x = UsageTier, y = Price, fill = Company)) +
+     geom_col(position = "dodge", width = 0.7, alpha = 0.8) +
+     geom_text(aes(label = paste0("$", Price)),
+               position = position_dodge(width = 0.7),
+               vjust = -0.3, size = 3.5, fontface = "bold") +
+     scale_fill_manual(values = c("ATT" = "#00A0B0", "Sprint" = "#F0B90B", "Verizon" = "#E74C3C")) +
+     labs(
+         title = "Head-to-Head Price Comparison at Popular Data Tiers",
+         subtitle = "Instant comparison for users who know their data needs - Sprint wins at low tiers",
+         x = "Data Allowance Tier",
+         y = "Monthly Price ($)",
+         fill = "Company"
+     ) +
+     theme_minimal() +
+     theme(
+         plot.title = element_text(size = 14, face = "bold"),
+         plot.subtitle = element_text(size = 11, color = "gray60"),
+         legend.position = "right",
+         panel.grid.minor = element_blank(),
+         panel.grid.major.x = element_blank(),
+         panel.grid.major.y = element_line(color = "gray90", size = 0.3),
+         axis.text.x = element_text(size = 11)
+     ) +
+     scale_y_continuous(
+         labels = scales::dollar_format(),
+         breaks = seq(0, 120, 20),
+         expand = expansion(mult = c(0, 0.1))
+     )
```