Midterm Assignment: Strawberry data analysis: Bader Alhodithi

Strawberry data analysis

We make a separate R script and connect it to our quarto document so as to keep things organized and clean

```
source("hw1_functions.R")
```

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

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.2.3


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
```

```
library(tidyr)
```

```
Warning: package 'tidyr' was built under R version 4.2.3
```

```
library(stringr)
library(ggplot2)
library(DT) #Got from website + documentation
```

```
Warning: package 'DT' was built under R version 4.2.3
```

```
library(plotly)
```

```
Warning: package 'plotly' was built under R version 4.2.3


Attaching package: 'plotly'

The following object is masked from 'package:ggplot2':

    last_plot

The following object is masked from 'package:stats':

    filter

The following object is masked from 'package:graphics':

    layout
```

```
library(tibble)
```

```
Warning: package 'tibble' was built under R version 4.2.3
```

You can add options to executable code like this

```
strawberry <- read.csv("strawb_mar6.csv")
```

First we start by cleaning the data. The first step here is to split our data into two tables, census and survey

```
census_data <- strawberry %>%
  filter(Program == "CENSUS")

survey_data <- strawberry %>%
  filter(Program == "SURVEY")
```

After making the function to drop columns that have less than one unique value, we implement it here:

```
census_data <- census_data %>%
  drop_single_value_col()

survey_data <- survey_data %>%
  drop_single_value_col()
```

Census Data cleaning

Reviewing the number of unique values for each column, we notice a few issues. For starters, the “Commodity” column has only two unique values, INCOME, NET CASH FARM and STRAWBERRIES. This feels really redundant; the entire data set is regarding strawberries, there is no confusion on that, and the “INCOME, NET CASH FARM” is messy jargon that is already referenced in other columns. hence, we can drop that column.

```
sapply(census_data, function(col) length(unique(col)))  #checking for reference
```

```
           Year           State      State.ANSI       Commodity       Data.Item 
              2               9               9               2              13 
         Domain Domain.Category           Value           CV.... 
              8              63             959             498 
```

Next, we notice a few redundancies in domain and domain category in the census data, they are often repetitive. lets mutate by removing the repetition. Luckily, they all, while different, have the same format of taking the domain + “:” + code.

```
census_data <- census_data %>%
  mutate(
    Domain.Category = gsub("^.+: \\((.*)\\)$", "\\1", `Domain.Category`)
  )  #remove redundent data, this took a bit to figure out, played around until it worked.
```

The census data has two different commodities, income, and strawberry data. while the strawberry data itself doesn’t have a lot of data points, we can further organize by splitting them into two different datasets as we did with survey and census.

```
income_data <- census_data %>%
  filter(Commodity == "INCOME, NET CASH FARM")  #filtering and splitting census into income and strawberry data

straw_data <- census_data %>%
  filter(Commodity == "STRAWBERRIES")
```

Lets continue more cleaning, we really dont need the state ANSI, and because our project focus is regarding California and Florida, we remove rows for all other states as well.

```
straw_data <- straw_data %>%
  select(-`State.ANSI`)

income_data <- income_data %>%
  select(-`State.ANSI`)

#not needed, gives no new information
```

lets also remove any state other than California and Florida, as they are our focal point of study

```
straw_data <- straw_data %>%
  filter(State %in% c("CALIFORNIA", "FLORIDA"))

income_data <- income_data %>%
  filter(State %in% c("CALIFORNIA", "FLORIDA")) #only using california and florida
```

now that the tables are split, lets use our function again to drop single value columns.

```
income_data <- income_data %>%
  drop_single_value_col()  #using my function to remove single val columns

straw_data <- straw_data %>%
  drop_single_value_col()
```

Now our table looks much cleaner, the years were both removed, as the strawberry data was only for 2021 and income was for 2022. commodity was removed due to its redundancy as well. Data.item carries a good amount of information that already makes up for the loss of the columns.

```
datatable(straw_data, options = list(pageLength = 5, scrollX = TRUE, autoWidth = TRUE)) #ok i really like this visual,, self
        note: use again.
```

Note that for both tables, I made use of the DT package for R to make the tables interactive, I personally find that design more appealing that knitr tables.

```
datatable(income_data, options = list(pageLength = 5, scrollX = TRUE, autoWidth = TRUE))
```

```
#took me a bit to figure scaling out, made a weird stupid mistake of using head?????
```

lets fix the data.item column for strawberry data, we can split it into multiple columns for brevity so it looks nicer.

```
straw_data <- straw_data %>% #basically splitting data.item with the dash delimeter, pre dash takes everyone prior to it,
        post takes everything after it. we do that again but by commas.  then merge and fill on right.
  separate(`Data.Item`, c("pre_dash","post_dash"), " - ", extra = "merge", fill = "right") %>%
  separate(pre_dash, c("commodity","organic_status","market_type"), ",", extra = "merge", fill = "right") %>%
  mutate(
    commodity = str_trim(commodity),
    organic_status = str_trim(organic_status),
    market_type = str_trim(market_type)
  ) %>%
  separate(post_dash, c("metric","unit_info"), ",", extra = "merge", fill = "right") %>%
  mutate(
    metric = str_trim(metric),
    unit_info = str_trim(unit_info),
    unit_info = str_remove(unit_info, "MEASURED IN ")
  )
#used chatgpt to get base form,, I dont really like using AI for answers, so i always ask for explanation, and at most to get
        a base form on what i should do.
```

The code above does a few things, it splits the data.item column, we use the dash as help in this case. I decided to create new columns, naming them commodity, organic_status (yes to all), and market_type. ultimately, there will be NA values as not all of them have same type of information in the data.item column, but now, to me at least, it is much cleaner looking regardless and is more understandable.

Now, lets do something similar to income_data.

```
library(dplyr)
library(tidyr)
library(stringr)

income_data <- income_data %>% #pretty much same process as above, split by -, organize, split comma, merge, etc..
  separate(`Data.Item`, c("pre_dash", "post_dash"), " - ", extra = "merge", fill = "right") %>%
  separate(post_dash, c("income_type", "unit_info"), ", MEASURED IN ", extra = "merge", fill = "right") %>%
  mutate(
    pre_dash = str_trim(pre_dash),
    income_type = str_trim(income_type),
    unit_info = str_trim(unit_info),
    unit_info = str_remove(unit_info, "\\$")
  ) %>%
  select(-pre_dash, -unit_info)
#pretty much same as above, although slgihtly different as i ran into some issues: note: do NOT copy paste code used for
        other table :)
```

note that two columns were removed, the pre_dash one was only used as part of the code, while the unit info was empty and not needed anyways as the dollar symbol itself is already present in income values

Survey data cleaning

Now lets move on to the survey data for cleaning. lets start off by using our function to remove columns with only one unique value.

```
survey_data <- survey_data %>%
  drop_single_value_col() #using function
```

As with the census data, lets remove columns that we do not need. in this case, as before, lets remove state.ANSI, along with every state except California and Florida.

```
survey_data <- survey_data %>%
  select(-`State.ANSI`) #again same cleaning, not needed
```

```
survey_data <- survey_data %>%
  filter(State %in% c("CALIFORNIA", "FLORIDA")) #only need those two states
```

Lets also go ahead and remove the period and week.ending, as for this table, week.ending is empty and period only has two values which are practically the same.

```
survey_data <- survey_data %>%
  select(- "Week.Ending")  #this would have been REALLY usefull had it been available for Florida and California :(
survey_data <- survey_data %>% #not needed
  select(- "Period")
```

We could, in theory, split the table based on the domain, but we really don't have enough data to do so, meaning we will work with this table as is.

Moving on, the first outliequarto render report.qmdr I see is the data.item field. lets list all of its unique values so we can identify how to split.

```
unique(survey_data$"Data.Item") #checking
```

```
 [1] "STRAWBERRIES - PRICE RECEIVED, MEASURED IN $ / CWT"
 [2] "STRAWBERRIES, FRESH MARKET - PRICE RECEIVED, MEASURED IN $ / CWT"
 [3] "STRAWBERRIES, PROCESSING - PRICE RECEIVED, MEASURED IN $ / CWT"
 [4] "STRAWBERRIES - ACRES HARVESTED"
 [5] "STRAWBERRIES - APPLICATIONS, MEASURED IN LB"
 [6] "STRAWBERRIES - APPLICATIONS, MEASURED IN LB / ACRE / APPLICATION, AVG"
 [7] "STRAWBERRIES - APPLICATIONS, MEASURED IN LB / ACRE / YEAR, AVG"
 [8] "STRAWBERRIES - APPLICATIONS, MEASURED IN NUMBER, AVG"
 [9] "STRAWBERRIES - TREATED, MEASURED IN PCT OF AREA BEARING, AVG"
[10] "STRAWBERRIES - YIELD, MEASURED IN CWT / ACRE"
[11] "STRAWBERRIES, BEARING - APPLICATIONS, MEASURED IN LB"
[12] "STRAWBERRIES, BEARING - APPLICATIONS, MEASURED IN LB / ACRE / APPLICATION, AVG"
[13] "STRAWBERRIES, BEARING - APPLICATIONS, MEASURED IN LB / ACRE / YEAR, AVG"
[14] "STRAWBERRIES, BEARING - APPLICATIONS, MEASURED IN NUMBER, AVG"
[15] "STRAWBERRIES, BEARING - TREATED, MEASURED IN PCT OF AREA BEARING, AVG"
```

given the unique values in the data.item column, lets go for a similar approach as before in splitting:

```
survey_data <- survey_data %>% #decided to try this mostly on my own without taking from previous one, same process, probably
        did not need to change names but post and pre dash isnt as "neat" in my head.
  separate(`Data.Item`, c("left_side", "right_side"), " - ", fill = "right", extra = "merge") %>%
  separate(left_side, c("commodity", "market_type"), ",", fill = "right", extra = "merge") %>%
  mutate(
    commodity = str_trim(commodity),
    market_type = str_trim(market_type)
  ) %>%
  separate(right_side, c("measure_type", "unit_info"), ", MEASURED IN ", fill = "right", extra = "merge") %>%
  mutate(
    measure_type = str_trim(measure_type),
    unit_info = str_trim(unit_info)
  )
#also took me a weirdly longer time than i feel it should have? need to practice more on splitting/merging columns from
        strings..
```

Again, as this is a strawberry data table, we really don't need a whole column (commodity) to mention that. We can remove it as it is the only value in that column. We can use our function again.

```
survey_data <- survey_data %>%
  drop_single_value_col()
```

Finally, lets try to keep the number of columns consistent, We can easily cut down some columns that are unnecessary:

```
survey_data <- survey_data %>%
  select(- "Domain")

straw_data <- straw_data %>%
  select(- "market_type")
```

In this case, Domain does not tell us anything that the category doesn't say, and market_type has no meaning and is mostly missing anyways. Seems like we are pretty much done with cleaning, anything more than this and we run the risk of "over cleaning" our data, making it too simple for any interesting visualization. Here is a display of all three tables I made:

```
datatable(income_data, options = list(pageLength = 5, scrollX = TRUE, autoWidth = TRUE)) #interactive, feels very intuitive.
        i like the search feature.
```

```
datatable(straw_data, options = list(pageLength = 5, scrollX = TRUE, autoWidth = TRUE))
```

```
datatable(survey_data, options = list(pageLength = 5, scrollX = TRUE, autoWidth = TRUE))
```

After checking data types. We notice that columns like Value is interpreted as a character, not a number. the final step in cleaning now is to normalize and fix data types for processing. In our case, I use gsub which will help remove commas from value and save it as a numeric value. As this will be repeated, I decided to code a function that:

1. Changes letter valued (D,H, etc...) to NA, then converts column to numeric

```
straw_data <- straw_data %>%
  rename(CV = `CV....`)  #for some reason the CV column has 4 dots after it??

income_data <- income_data %>%
  rename(CV = `CV....`)

straw_data <- convert_numeric(straw_data, cols_to_convert = c("Value", "CV"))
```

```
Warning: There were 2 warnings in `mutate()`.
The first warning was:
ℹ In argument: `across(...)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
ℹ Run `dplyr::last_dplyr_warnings()` to see the 1 remaining warning.
```

```
income_data <- convert_numeric(income_data, cols_to_convert = c("Value", "CV"))
```

```
Warning: There was 1 warning in `mutate()`.
ℹ In argument: `across(...)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
```

```
survey_data <- convert_numeric(survey_data, cols_to_convert = "Value")
```

```
Warning: There was 1 warning in `mutate()`.
ℹ In argument: `across(...)`.
Caused by warning in `ifelse()`:
! NAs introduced by coercion
```

```
#survey has no CV
```

Visualizations: Survey data

Lets explore the survey data to start with, as it has the information we want on chemical pesticides. First, lets try to filter rows where application (application of pesticides), then find the top 5 chemicals.

```
top_chemicals <- survey_data %>%  #getting top chemicals, i did this by taking CHEMICAL from domain category and taking the
        highest, avoided rows which are not number (d,h,etc..)
  filter(
    grepl("APPLICATIONS|TREATED", measure_type, ignore.case = TRUE),
    grepl("CHEMICAL", `Domain.Category`),
    grepl("^[0-9,.]+$", Value)  # keep only rows that are numeric-like
  ) %>%
  mutate(Value = as.numeric(gsub(",", "", Value)))
```

```
top_chemicals_lst <- top_chemicals %>%
  group_by(State, `Domain.Category`) %>%
  summarise(total_value = sum(Value, na.rm = TRUE), .groups = "drop") %>%
  arrange(State, desc(total_value)) %>%
  group_by(State) %>%
  slice_head(n = 5) %>%  #listing them, probably should have excluded total,,,
  ungroup()
#prior to this, i had it top 5 all together, but california uses so much more chemicals that its impossible to visualize
        well, hence i decided to take top 3 of each state
```

With the code above, i filtered the chemicals (made sure to group values as numeric while also excluding the letter values, some of which we can’t use as they are withheld for privacy reasons I assume. Next, i listed the top 10 chemicals used. With this information, my objective is to pick the top 5 chemicals, i picked a slide head of 10 so as to avoid using the “total” values. based on the data, I will use these chemicals:

California:

1. CHLOROPICRIN
2. DICHLOROPROPENE
3. SULFUR

Florida:

1. CAPTAN
2. THIRAM
3. CYPRODINIL

now lets adjust naming conventions and some small cleaning before moving to visualizing:

```
chem_top5 <- top_chemicals %>%
  filter(`Domain.Category` %in% top_chemicals_lst$`Domain.Category`) %>%
  group_by(State, `Domain.Category`, measure_type) %>%
  summarise(avg_value = mean(Value, na.rm = TRUE), .groups = "drop")

chem_top5 <- chem_top5 %>%
  mutate(chemical = str_extract(`Domain.Category`, "(?<=\\().+?(?=\\s?=)"))
#initially i manually did for all of the names, asked chatgpt if there is a better way to do it and it recomended i extract
        the characters from domain.category. not sure why i "brute forced" it.
```

Now lets fix the names (probably could have done the steps before this more efficiently, I do acknowledge some redundancy with my code.

```
chem_top5 <- chem_top5 %>%
  mutate(chemical = case_when(
    grepl("CHLOROPICRIN", `Domain.Category`) ~ "CHLOROPICRIN",
    grepl("DICHLOROPROPENE", `Domain.Category`) ~ "DICHLOROPROPENE",
    grepl("SULFUR", `Domain.Category`) ~ "SULFUR",
    grepl("CAPTAN", `Domain.Category`) ~ "CAPTAN",
    grepl("METAM-POTASSIUM", `Domain.Category`) ~ "METAM-POTASSIUM"
  ))

p <- ggplot(chem_top5, aes(x = chemical, y = avg_value, fill = State)) +
  geom_col(position = "dodge") +
  facet_wrap(~measure_type, scales = "free_y") +
  labs(
    title = "Interactive chart of chemicals used on Strawberries for california and florida",
    x = "Chemical",  #r documentation and the ? feature in r studio is my SAVING GRACE,,, its very intuitive.
    y = "Average Value",
    fill = "State"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 30, hjust = 1),
        plot.title = element_text(size = 10, face = "bold")) #had to change text size, would not fit otherwise, weirdly still
        moves text to the right a bit, doesnt matter hopefully, still visible.

ggplotly(p) #kept bugging at first, doesnt let me hover for extra information if i add legends or annotation??
```

```
#note: a bit different to use than plotly in python, which I am more familliar with. i LOVE the hover for extra info feature
```

This looks interesting. In prior iterations, I made it such that it picked the top 5 used chemicals in all states, which was problematic as it had heavily biased data towards California, mainly due to Florida having more NA and redacted values. So picking the top 3 from each state was the right call in my view. I used plotly here to create an interactive bar chart, you are able to specifically select either state to highlight, and hovering over the lines gives some more information on the exact value.

Looking at this visual, we see very minimal data in the applications section, so we can mostly disregard it for now. Lets start off with CHLOROPICRIN, which has a fairly high average value of over 50; According to the CDS, this substance causes eye irritation and is often used in tear gas and as a chemical warfare agent, which I see as concerning. the CDS also has studies on Dichloropropenes, which have been found to cause chest pain irritation, coughing, and erosion of stomach lining. finally, Sulfur seems to be, according to my research and the Environmental protection agency (EPA), the safest of the three in use. It has a low toxicity and has little risk to human health. Moreover, the EPA also studied its environmental and ecological effects and made the conclusion that it does not cause too much adverse effects.

Moving on to Florida, the clear winner here is CAPTAN, a fungicide, which can cause illness by inhalation, skin absorption, or consumption. It can cause irritation of eyes, skin, upper respiratory system, and vomiting. Thiram is a fungicide, it has moderate toxicity by ingestion, but highly toxic if inhaled; chronic or repeated exposure may have negative effects on liver and thyroid. finally, Cyprodinil, a fungicide, generally has low toxicity, but it is important to note the limited research on it, which is a common factor with most chemicals in this list.

Analyzing the difference in chemicals:

Looking at the data and top 3, it is evident that California relies mostly on Fumigants, while Florida on Fungicides, highlighting the possible need for each in both states. Below is a table I created for the top 3 chemicals in both states along with their type.

```
chemical_table <- tibble(
  Chemical = c("Chloropicrin", "Dichloropropene", "Sulfur",
               "Captan", "Thiram", "Cyprodinil"),
  Type = c("Fumigant", "Fumigant", "Fungicide",
           "Fungicide", "Fungicide", "Fungicide"),
  Purpose = c("Soil disinfection, pest control",
              "Controls nematodes in the soil",
              "Controls fungal diseases, powdery mildew",
              "Controls mold, rots, and fungal infections",
              "Seed protection, prevents rot and damping-off",
              "Targets gray mold and Botrytis"),
  State = c("California", "California", "California",
            "Florida", "Florida", "Florida")
)
datatable(chemical_table, options = list(
  pageLength = 6,
  autoWidth = TRUE
), extensions = 'Buttons')
```

```
#this took a weirdly long time, I gave chatgpt the information on the type of chemicals, and asked it to write the tibble for
        me so i dont have to repeat the same name. I then manually wrote the DT
```

California has more of a dry climate, which promotes the growth of pests in the soil, which is why fumigation is used more often. Florida, on the other hand, is humid and wet, which gives rise to fungal diseases, which is they more often use fungicides.

2: Production and sales of conventional vs organic strawberries:

Here i created a pyramid chart to show acres in which conventional vs organic strawberries are farmed

```
pyramid_data <- bind_rows( #for some reason bar charts would only display ONE item, even if the legend has both, and they
        would be identible, probably did something wrong. pyramid seems to work.
  straw_data %>% filter(metric == "ACRES HARVESTED") %>% mutate(Type = "Organic"),
  survey_data %>% filter(measure_type == "ACRES HARVESTED") %>% mutate(Type = "Conventional") #i could have kept making
        tables for each visual but I thought it was wayyyy too convoluted :(
)

ggplot(pyramid_data, aes(x = State, y = Value, fill = Type)) +
  geom_bar(data = pyramid_data %>% filter(Type == "Organic"),
           aes(y = -Value), stat = "identity") +
  geom_bar(data = pyramid_data %>% filter(Type == "Conventional"),
           stat = "identity") +
  coord_flip() +
  scale_y_continuous(labels = abs) +
  labs(
    title = "pyramid Chart: Organic vs Conventional acres harvested",
    x = "State",
    y = "Acres Harvested",
    fill = "Type"
  ) +
  theme_minimal()
```

```
ggsave("pyramid_organic_conv_acres.png", width = 8, height = 6, dpi = 300)
```

The results are not surprising at all. When you are able to use chemicals, you do not need to worry about having constant maintenance to stop bugs and disease, whereas with organic strawberries, you need a more hands-on approach in cultivation, meaning less acres. It is also worth noting that the organic strawberries data is VERY limited, only having 13 rows of data.

Sales comparison:

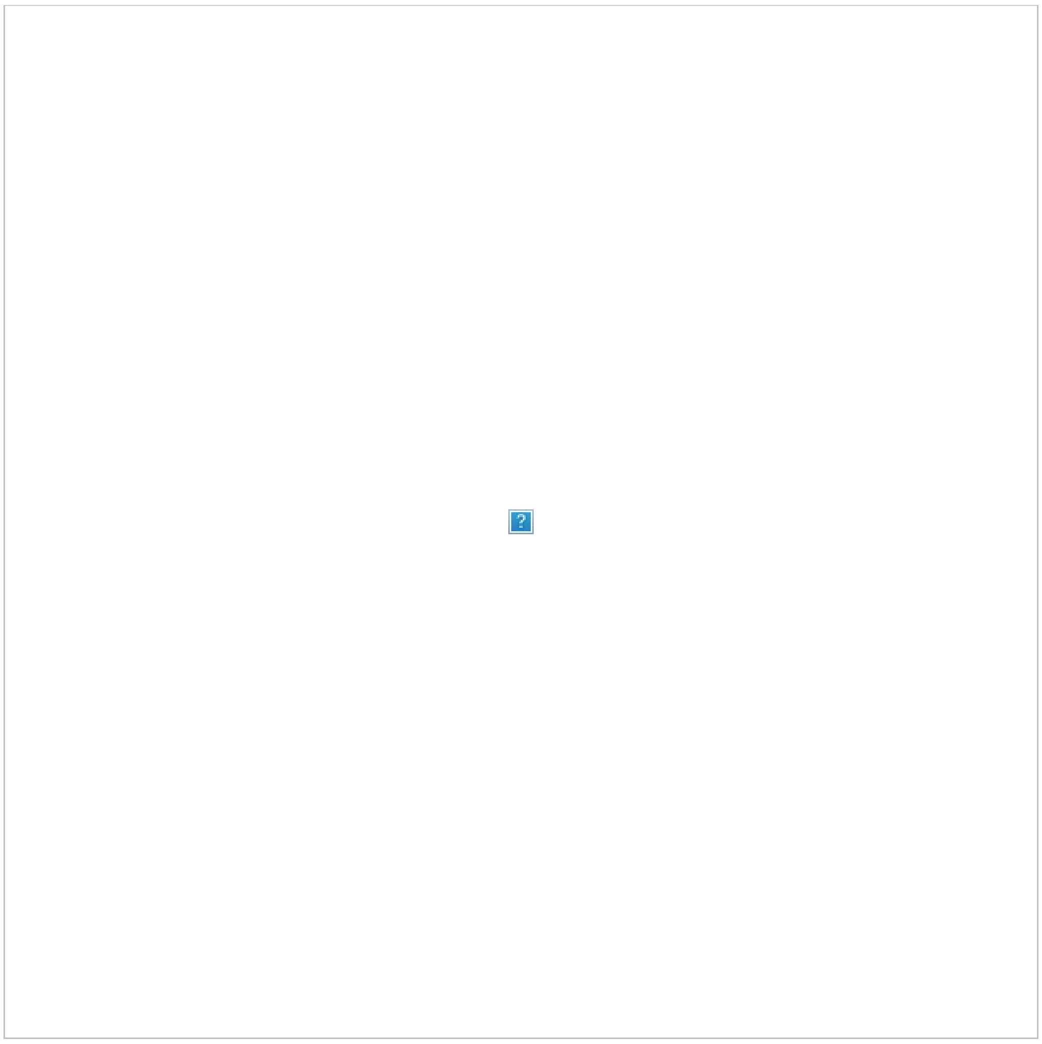
Lets first analyze the coefficient of variation for both states

```
ggplot(straw_data, aes(x = Value, y = CV, color = State)) +
  geom_point(alpha = 0.6, size = 3) +
  geom_smooth(method = "lm", se = FALSE, linetype = "dashed", color = "black") +
  labs(
    title = "sales vs coefficient variation (CV) by State",
    x = "sales $",
    y = "coefficient variation (%)", #again the R ? is an amazing feature.
    color = "State"
  ) +
  theme_minimal() #i really wanted to visualize the CV, but the data is not helping me with this. not much correlation given
      the line.
```

```
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: Removed 7 rows containing non-finite values (`stat_smooth()`).
```

```
Warning: Removed 7 rows containing missing values (`geom_point()`).
```



```
ggsave("cv_point.png", width = 8, height = 6, dpi = 300)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: Removed 7 rows containing non-finite values (`stat_smooth()`).
Removed 7 rows containing missing values (`geom_point()`).
```

This is hard to read to be frank, but this is mainly due to the limited data. what this shows is two things:

- Florida has a much higher Coefficient of variation (CV)
- California has a downward slope, meaning that as sales increase, the variation decreases, leading to more predictable results. although, we can also argue that it is an outlier as the line is flat, meaning we have little to no relationship. Again, its really difficult to analyze with the limited data.

Trends over time: income data

the week ending column does NOT have data for California or Florida, Lets go back to the census_data and aggregate it to get the trends over time.

```
census_data$Value <- as.numeric(gsub("[^0-9.]", "", census_data$Value))
#had to go back to this,,,,
```

```
strawberry$Value <- as.numeric(strawberry$Value)
```

```
Warning: NAs introduced by coercion
```

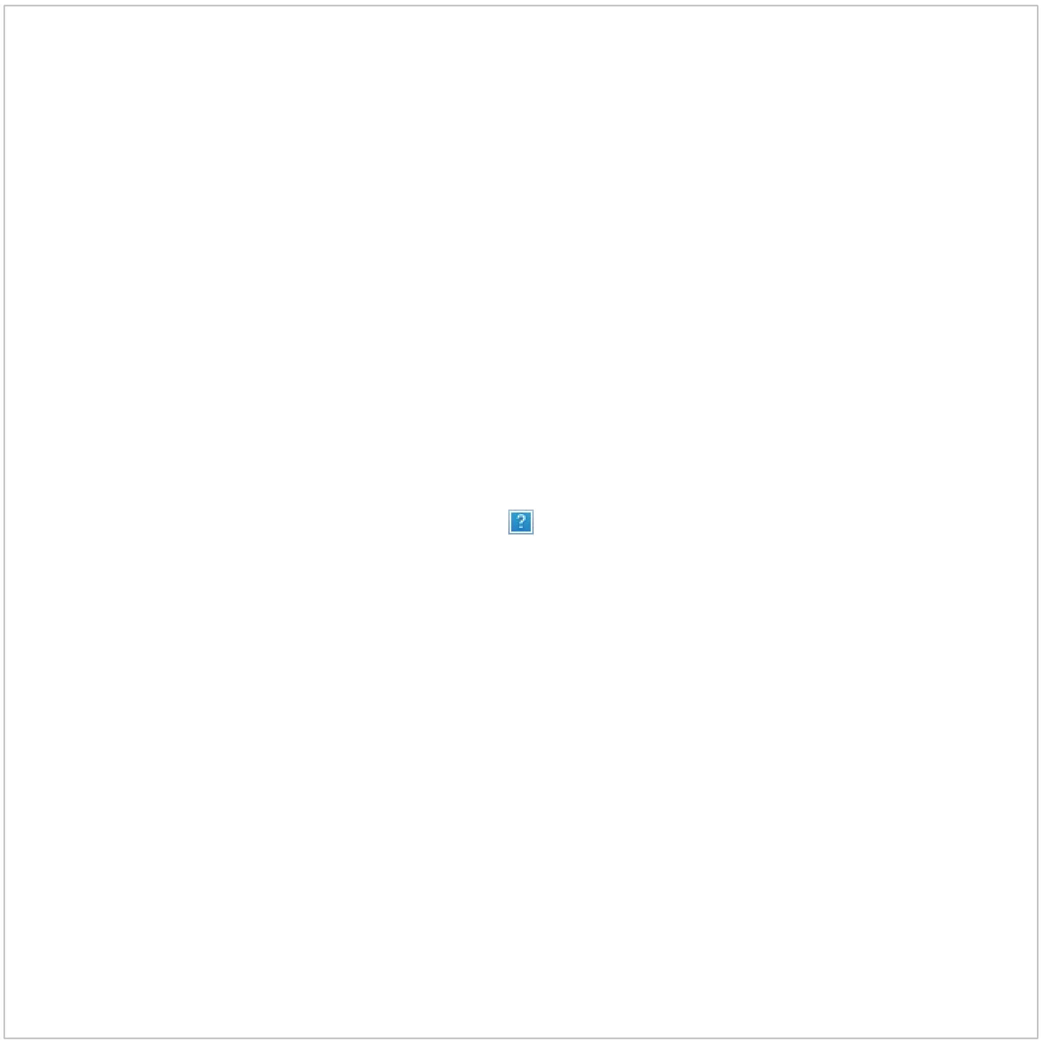
```
strawberry_filtered <- subset(strawberry, State %in% c("CALIFORNIA", "FLORIDA"))

agg_data <- aggregate(Value ~ Year + State, data = strawberry_filtered, sum, na.rm = TRUE) #aggregating by year/state. this
        was breaking too many times, crashed R studio for some reason???? should work now

plot(NA, xlim = range(agg_data$Year), ylim = range(agg_data$Value, na.rm = TRUE),
     xlab = "Year", ylab = "Total Income (USD)",
     main = "Total Strawberry Income Over Time")  #using line chart. this one took me hours to do :(

colors <- c("blue", "red")
states <- unique(agg_data$State) #by state, bugged without
for (s in states) {
  subset_data <- subset(agg_data, State == s)
  lines(subset_data$Year, subset_data$Value, col = colors[which(states == s)], lwd = 2, type = "o", pch = 16) #did not need
        to do this probably but really copied from the template in documentations and ?help then filled in for some, this for
        loop took me a bit
}

#ok, this one was so bad it took me way too long, income_data didnt have year, so i had to compromise
legend("topright", legend = states, col = colors, lwd = 2)
```



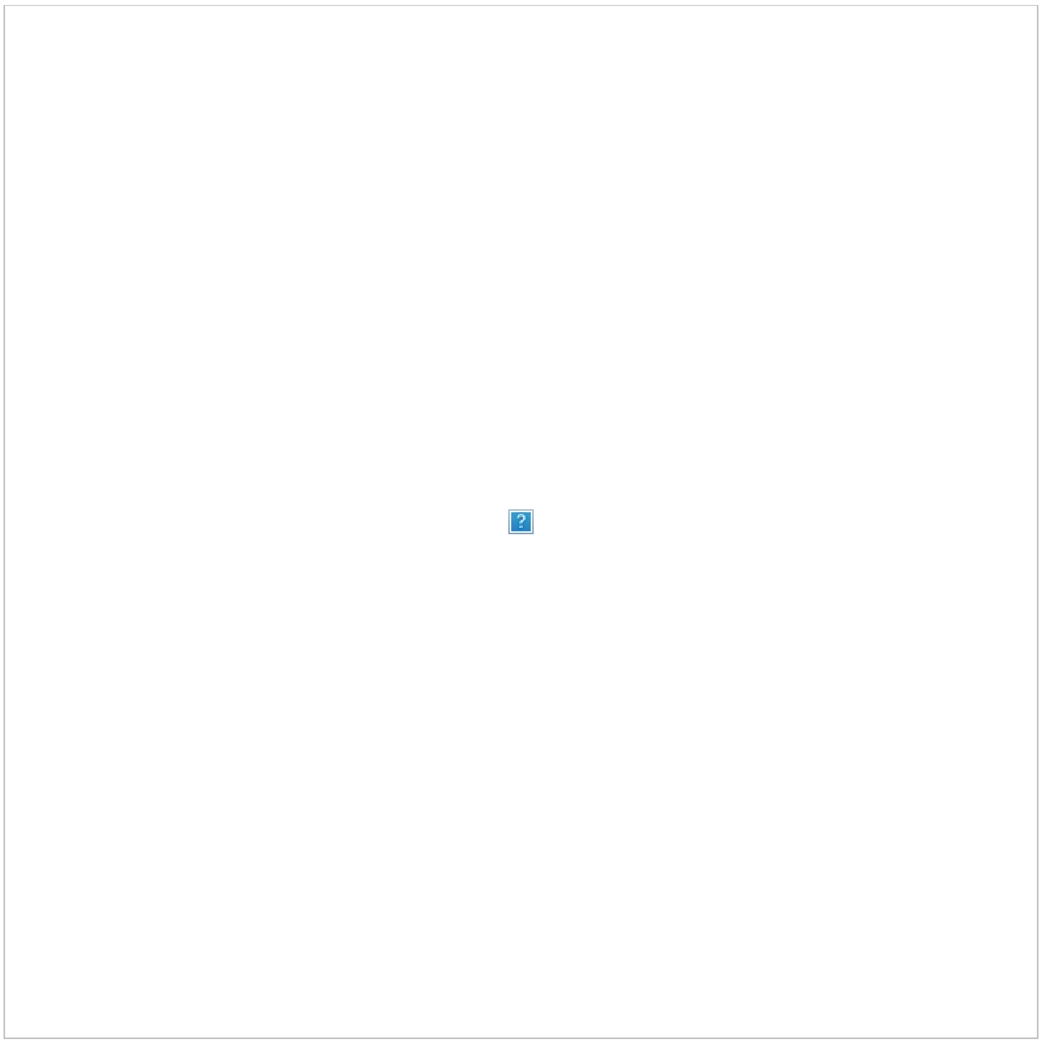
This visual is interesting, it clearly shows California is dominant when it comes to income over time. What I find most interesting is that the trends are nearly identical. I would have expected slightly differing trends, due to the nature of the different climates, but clearly This shows otherwise. Here are possible reasons in my opinion:

- 2020: This is really low likely due to the Covid-19 pandemic, which heavily disrupted supply chains and caused a reduction in labor and a fluctuation in demand.
- 2021: This to me seems like a result of the post-pandemic recovery period, which led to stronger consumer demands and the opening of food industries and increased exports after the regulations eased.
- 2022: This is a tricky one to analyze. it could be due to lack of data, but also could be due to inflation and economic instability. I have little information about weather patterns to judge if that is relevant, meaning this needs further research.
- 2023: This might suggest that the markets are strengthening, possibly better weather for farming, or even more investments. Again, like 2022, its hard to answer this without more data.

Lets now move to analyzing the income data:

```
ggplot(income_data, aes(x = State, y = Value, fill = income_type)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "comparison of income types",
    x = "State",
    y = "Total Income (USD)",
    fill = "Income Type"
  ) +
  scale_fill_manual(values = c("GAIN" = "blue", "LOSS" = "red", "NET INCOME" = "green")) +
  theme_minimal() #reminder to change colour, why on earth would gain be red.
```

```
Warning: Removed 68 rows containing missing values (`geom_bar()`).
```



```
ggsave("income_type.png", width = 8, height = 6, dpi = 300)
```

```
Warning: Removed 68 rows containing missing values (`geom_bar()`).
```

Trends here make sense, in both states, gain is higher than loss. Florida has less data and/or production, so the data shows it to be less, but the trends are similar.

```
ggplot(income_data, aes(x = income_type, y = CV, fill = State)) + #box plot,, variability is very likely just outliers?
  geom_boxplot() +
  labs(
    title = "income variability by state",
    x = "income",
    y = "coefficient variation (%)",
    fill = "State"
  ) +
  theme_minimal()
```

```
Warning: Removed 77 rows containing non-finite values (`stat_boxplot()`).
```

```
ggsave("income_cv.png", width = 8, height = 6, dpi = 300)
```

```
Warning: Removed 77 rows containing non-finite values (`stat_boxplot()`).
```

While slightly looking convoluted, this does tell us that Florida has a higher coefficient of variation (CV) in this case. The higher variability could also be an outlier.

Conclusion

Based on the analysis of the strawberry data, we got a comprehensive view of the data regarding California and Florida. The findings show a good amount of interesting information regarding the trends. Through the use of visuals, My main objective was to not simply analyze a specific research question, but to showcase the many interpretations one can achieve from looking at the data. From this project, It is clear to me that merely having data may not be enough, and making over-generalizations can hinder our understanding of the information; Which is why I added a lot of my personal opinion on what the data means and the reasoning, because we cannot know for certain.

Suggestions/improvements for future analysis

In the future, I hope to gather more data that can be related to farming and crop analysis. I found myself wanting to know weather pattern data, farmland numbers, disease emergence in crops, etc…. This would further help me find interesting features and correlations (if any) within the data.

Sources used:

https://en.wikipedia.org/wiki/Thiram

https://pubchem.ncbi.nlm.nih.gov/compound/Captan#section=Toxicity

https://www3.epa.gov/pesticides/chem_search/reg_actions/reregistration/fs_PC-077501_1-May-91.pdf

https://www.andrewheiss.com/blog/2022/12/08/log10-natural-log-scales-ggplot/

https://rstudio.github.io/DT/

https://www.njtierney.com/post/2022/08/09/ggplot-pyramid/

https://dplyr.tidyverse.org/

https://dplyr.tidyverse.org/reference/mutate.html

chatGPT: asked for some help on splitting, i explained my issue and it gave me this:

data <- data %>% separate(“column_name”, c(“pre_dash”, “post_dash”), ” - “, extra =”merge”, fill = “right”) %>% separate(“pre_dash”, c(“col1”, “col2”, “col3”, etc…), “,”, extra = “merge”, fill = “right”) %>% mutate(col1 = str_trim(col1), col2 = str_trim(col2), col3 = str_trim(col3)) %>% separate(“post_dash”, c(“col4”, “col5”), “,”, extra = “merge”, fill = “right”) %>% mutate(col4 = str_trim(col4), col5 = str_trim(col5), col5 = str_remove(col5, “MEASURED IN”))

I avoid giving it much information as i don’t want a direct answer.

also, I did use chatgpt do debug errors i got, never gave my code, just gave the error and asked why and what can cause it.