Strawberry Final Project

Ruijian Maggie Lin

2024-10-28

Strawberries Data Cleaning

```
strawberry <- read_csv("strawberries.csv", col_names = TRUE)
```

```
## Rows: 12669 Columns: 21
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (15): Program, Period, Geo Level, State, State ANSI, Ag District, County...
## dbl  (2): Year, Ag District Code
## lgl  (4): Week Ending, Zip Code, Region, Watershed
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

I have 12699 rows and 21 columns.

All I can see from the glimpse is I have date, location, values and coefficients of variation.

Remove columns with a single value in all rows

```
#|label: function def - drop 1-item columns

drop_one_value_col <- function(df){   ## takes whole dataframe
  drop <- NULL

  ## test each column for a single value
  for(i in 1:dim(df)[2]){
    if((df |> distinct(df[,i]) |> count()) == 1){
        drop = c(drop, i)
      }
  }

  ## report the result -- names of columns dropped
  ## consider using the column content for labels or headers

  if(is.null(drop)){return("none")}else{
       print("Columns dropped:")
```

```
      print(colnames(df)[drop])
      strawberry <- df[, -1*drop]
  }
}

## use the function
strawberry <- drop_one_value_col(strawberry)
```

```
## [1] "Columns dropped:"
## [1] "Week Ending"    "Zip Code"       "Region"         "watershed_code"
## [5] "Watershed"      "Commodity"
```

Separate composite columns

Data Item

Data Item into (fruit, category, item, metric)

```
#|label: Split Data Item into 4 new columns: Fruits, Category, Item, Metric.

strawberry_cleaned <- strawberry %>%
  separate_wider_delim(
    cols = 'Data Item',
    delim = " - ",
    names = c("Fruit", "Category_Metric"),
    too_many = "merge",
    too_few = "align_start"
  ) %>%
  separate_wider_delim(
    cols = Category_Metric,
    delim = ", ",
    names = c("Category", "Item", "Metric"),
    too_many = "merge",
    too_few = "align_start"
  )
```

```
#|label: Briefly clean `Data Item` to the correspond columns.

# Check the common term for each column after split
# Identify the next step of cleaning
#   unique(strawberry_cleaned$Fruit)
#   unique(strawberry_cleaned$Category)
#   unique(strawberry_cleaned$Item)
#   unique(strawberry_cleaned$Metric)

# Combine the Item to Metric, leaving Item column blank.
strawberry_cleaned_i_m <- strawberry_cleaned %>%
  mutate(
    # Move non-NA Item values to the Metric column, prepending them
    Metric = ifelse(!is.na(Item), paste(Item, Metric, sep = ", "), Metric),
```

```
    # Remove everything in the Item column, keeping it blank
    Item = NA_character_,
    # Remove 'NA' text from Metric (if exists)
    Metric = gsub("NA, |, NA", "", Metric)
  )

# Combine the Category to Item, leaving Category column blank.
strawberry_cleaned_c_i <- strawberry_cleaned_i_m %>%
  mutate(
    # Move contents from Category to Item
    Item = ifelse(!is.na(Category), Category, Item),
    # Clear the Category column
    Category = NA_character_
  )

strawberry <- strawberry_cleaned_c_i %>%
  mutate(
    # Move everything after the comma from Fruit to Category
    Category = ifelse(grepl(",", Fruit),
                      paste(sub(".*?, ", "", Fruit), Category, sep = ", "),
                      Category),
    # Keep only "STRAWBERRIES" in the Fruit column
    Fruit = "STRAWBERRIES",
    # Remove NA or empty characters from Category if they exist
    Category = gsub("NA, |, NA", "", Category),
    Category = trimws(Category, which = "both")  # Trim leading/trailing spaces
  )
```

```
#|label: Clean Category column.

# Check the common term for each column
#   unique(strawberry$Fruit)
#   unique(strawberry$Category)

# Replace commas with hyphens in the Category column
strawberry <- strawberry %>%
  mutate(
    Category = gsub(", ", "-", Category),
  )
```

```
#|label: Clean Item column.

# unique(strawberry$Item)

strawberry <- strawberry %>%
  mutate(
    # Replace "AREA" with "ACRES" in the Item column
    Item = gsub("AREA", "ACRES", Item),
    # Replace 'WITH' with ':' in the Item column
    Item = gsub(" WITH ", ":", Item)
  )
```

```
#|label: Clean Metric column.

# unique(strawberry$Metric)

strawberry <- strawberry %>%
  mutate(
    # Remove 'MEASURED IN' from the Metric column
    Metric = gsub("MEASURED IN ", "", Metric),
    # Remove extra spaces around '/' and after ','
    Metric = gsub("\\s*/\\s*", "/", Metric),  # Trim spaces around '/'
    Metric = gsub(",\\s*", ",", Metric)       # Remove spaces after commas
  )
```

Remove redundant parts in `Domain Category`

```
# When Domain has TOTAL, keeps "NOT SPECIFIED" in Domain Category.
strawberry <- strawberry %>%
  mutate(`Domain Category` = case_when(
    Domain == "TOTAL" & `Domain Category` == "NOT SPECIFIED" ~ "NOT SPECIFIED",
    TRUE ~ `Domain Category`
  ))

# Remove redundant parts in `Domain Category` that already exist in `Domain`
strawberry <- strawberry %>%
  mutate(
    `Domain Category` = case_when(
      # Check if the text before the colon (:) in the Domain Category column
      # matches the text in the Domain column
      str_detect(`Domain Category`, ":") &
      str_trim(str_extract(`Domain Category`, "^[^:]+")) == Domain ~
        # If matches, remove text before : and brackets
        str_replace_all(str_remove(`Domain Category`, "^[^:]+: "), "\\((.*?)\\)", "\\1"),
      TRUE ~ `Domain Category`  # Keep the original value if not matched
    ),
    # Remove brackets and trim spaces
    `Domain Category` = str_replace_all(`Domain Category`, "\\s*\\(\\s*|\\s*\\)", "")
  )

# unique(strawberry$`Domain Category`)
```

`Domain Category`

`Domain Category` into (Domain Category, Acres, Code)

```
strawberry <- strawberry %>%
  mutate(
    # Extract Acreage information where 'ACRES' is present
    Acres = case_when(
      str_detect(`Domain Category`, "ACRES") ~ str_trim(`Domain Category`),
```

```
      TRUE ~ NA_character_
    ),
    # Extract the part after the '=' symbol for Code
    Code = case_when(
      str_detect(`Domain Category`, "=") ~ str_trim(str_extract(`Domain Category`, "(?<=\\=).*")),
      TRUE ~ NA_character_
    ),
    # Clean Domain_Category by removing code part if it exists
    `Domain Category` = str_trim(case_when(
      str_detect(`Domain Category`, "=") ~ str_trim(str_replace(`Domain Category`, " =.*", "")), # Remo
      TRUE ~ `Domain Category`
    )),
    # Replace "ACRES" with NA in the Domain Category column
  `Domain Category` = ifelse(grepl("ACRES", `Domain Category`), NA, `Domain Category`)
  ) %>%
  # Relocate the new Acres and Code columns next to Domain Category
  relocate(Acres, Code, .after = `Domain Category`)

# unique(strawberry$`Domain Category`)
# unique(strawberry$Acres)
```

Clean `Value` & `CV` columns

`Value`

Identify footnotes in `Value`

```
footnotes_v <- strawberry %>%
  # Filter out numeric values including decimals and commas
  filter(!is.na(Value) & !grepl("^[0-9]+(\\.[0-9]+)?(,[0-9]{1,3})*$", Value)) %>%
  distinct(Value)
```

The `Value` column contains the footnote (D), (Z), and (NA).

(D): Withheld to avoid disclosing data for individual operations. (Z): Less than half the rounding unit. (NA): Not available.

```
# Replace the string "(NA)" with actual NA values
# strawberry <- strawberry %>% mutate(Value = na_if(Value, "(NA)"))
# strawberry$Value <- strawberry$Value[is.na(as.numeric(str_replace(strawberry$Value, ",", "")))]
```

1. Clean `Value` for Florida State

```
florida <- strawberry |>  filter(State=="FLORIDA")

florida_census <- florida |> filter(Program=="CENSUS")
florida_survey  <- florida |>  filter(Program=="SURVEY")
```

```r
# 1.
unique(florida_census$Item)
```

```
##  [1] "ACRES BEARING"                "ACRES GROWN"
##  [3] "ACRES NON-BEARING"            "OPERATIONS:ACRES BEARING"
##  [5] "OPERATIONS:ACRES GROWN"       "OPERATIONS:ACRES NON-BEARING"
##  [7] "ACRES HARVESTED"              "OPERATIONS:ACRES HARVESTED"
##  [9] "OPERATIONS:SALES"             "PRODUCTION"
## [11] "SALES"
```

```r
# Filter for all unique items in florida_census and assign to different variables
#   acres_bearing <- florida_census |> filter(Item == "ACRES BEARING")
#   acres_grown <- florida_census |> filter(Item == "ACRES GROWN")
#   acres_non_bearing <- florida_census |> filter(Item == "ACRES NON-BEARING")
#   operations_acres_bearing <- florida_census |> filter(Item == "OPERATIONS:ACRES BEARING")
#   operations_acres_grown <- florida_census |> filter(Item == "OPERATIONS:ACRES GROWN")
#   operations_acres_non_bearing <- florida_census |> filter(Item == "OPERATIONS:ACRES NON-BEARING")
#   acres_harvested <- florida_census |> filter(Item == "ACRES HARVESTED")
#   operations_acres_harvested <- florida_census |> filter(Item == "OPERATIONS:ACRES HARVESTED")
#   operations_sales <- florida_census |> filter(Item == "OPERATIONS:SALES")
#   production <- florida_census |> filter(Item == "PRODUCTION")
#   sales <- florida_census |> filter(Item == "SALES")

unique(florida_census$Domain)
```

```
## [1] "TOTAL"          "AREA GROWN"     "ORGANIC STATUS"
```

```r
# The unique items in Domain contains Total, Area Grown, and Organic Status.
```

After checking the unique items in the `Domain`, I noticed that each calif_census contains several columns for Area Grown and that for Organic Status. The total Value of Area Grown and that of Organic Status should sum up to the Value in the `Total` within the `Domain` column, which reflects the total after considering these columns.

Additionally, there are several footnotes in the `Value` for Area Grown and Organic Status. These footnotes should be replaced with reasonable numbers into, based on the correspond range in `Acres` column, ensure that they sum up to the correct Value in the `Total` within the `Domain` column.

```r
# 2.
unique(florida_survey$Item)
```

```
## [1] "PRICE RECEIVED"  "ACRES HARVESTED" "ACRES PLANTED"   "APPLICATIONS"
## [5] "PRODUCTION"      "TREATED"         "YIELD"
```

```r
# Filter for all unique items in calif_survey and assign to different variables
#   price_received <- florida_survey |> filter(Item == "PRICE RECEIVED")
#   acres_harvested <- florida_survey |> filter(Item == "ACRES HARVESTED")
#   acres_planted <- florida_survey |> filter(Item == "ACRES PLANTED")
#   applications <- florida_survey |> filter(Item == "APPLICATIONS")
#   production <- florida_survey |> filter(Item == "PRODUCTION")
#   treated <- florida_survey |> filter(Item == "TREATED")
```

```
#   yield <- florida_survey |> filter(Item == "YIELD")

unique(florida_survey$Domain)
```

```
## [1] "TOTAL"                 "CHEMICAL, FUNGICIDE"   "CHEMICAL, HERBICIDE"
## [4] "CHEMICAL, INSECTICIDE" "CHEMICAL, OTHER"       "FERTILIZER"
```

After check the unique items in Domain, no additional action to clean the Value column.

2. Clean `Value` for California State

```
calif <- strawberry |>  filter(State=="CALIFORNIA")

calif_census <- calif |> filter(Program=="CENSUS")
calif_survey  <- calif |>  filter(Program=="SURVEY")
```

```
# 1.
unique(calif_census$Item)
```

```
##  [1] "ACRES BEARING"                 "ACRES GROWN"
##  [3] "OPERATIONS:ACRES BEARING"      "OPERATIONS:ACRES GROWN"
##  [5] "ACRES NON-BEARING"             "OPERATIONS:ACRES NON-BEARING"
##  [7] "ACRES HARVESTED"               "OPERATIONS:ACRES HARVESTED"
##  [9] "OPERATIONS:SALES"              "PRODUCTION"
## [11] "SALES"
```

```
# Filter for all unique items in calif_census_c and assign to different variables
#   acres_bearing <- calif_census |> filter(Item == "ACRES BEARING")
#   acres_grown <- calif_census |> filter(Item == "ACRES GROWN")
#   acres_non_bearing <- calif_census |> filter(Item == "ACRES NON-BEARING")
#   operations_acres_bearing <- calif_census |> filter(Item == "OPERATIONS:ACRES BEARING")
#   operations_acres_grown <- calif_census |> filter(Item == "OPERATIONS:ACRES GROWN")
#   operations_acres_non_bearing <- calif_census |> filter(Item == "OPERATIONS:ACRES NON-BEARING")
#   acres_harvested <- calif_census |> filter(Item == "ACRES HARVESTED")
#   operations_acres_harvested <- calif_census |> filter(Item == "OPERATIONS:ACRES HARVESTED")
#   operations_sales <- calif_census |> filter(Item == "OPERATIONS:SALES")
#   production <- calif_census |> filter(Item == "PRODUCTION")
#   sales <- calif_census |> filter(Item == "SALES")

unique(calif_census$Domain)
```

```
## [1] "TOTAL"          "AREA GROWN"     "ORGANIC STATUS"
```

```
# The unique items in Domain contains Total, Area Grown, and Organic Status.
```

Same process and same result and solution as clean `Value` for Florida State.

```r
# 2.
unique(calif_survey$Item)
```

```
## [1] "PRICE RECEIVED"  "ACRES HARVESTED" "ACRES PLANTED"   "APPLICATIONS"
## [5] "PRODUCTION"      "TREATED"         "YIELD"
```

```r
# Filter for all unique items in calif_survey and assign to different variables
#   price_received <- calif_survey |> filter(Item == "PRICE RECEIVED")
#   acres_harvested <- calif_survey |> filter(Item == "ACRES HARVESTED")
#   acres_planted <- calif_survey |> filter(Item == "ACRES PLANTED")
#   applications <- calif_survey |> filter(Item == "APPLICATIONS")
#   production <- calif_survey |> filter(Item == "PRODUCTION")
#   treated <- calif_survey |> filter(Item == "TREATED")
#   yield <- calif_survey |> filter(Item == "YIELD")

unique(calif_survey$Domain)
```

```
## [1] "TOTAL"                "CHEMICAL, FUNGICIDE"   "CHEMICAL, INSECTICIDE"
## [4] "CHEMICAL, OTHER"      "CHEMICAL, HERBICIDE"   "FERTILIZER"
```

Same process as clean `Value` for Florida State. After check the unique items in Domain, no additional action to clean the Value column.

CV (%)

Identify footnotes in CV (%)

```r
footnotes_cv <- strawberry %>%
  # Filter out numeric values including decimals and commas
  filter(!is.na(`CV (%)`) & !grepl("^[0-9]+(\\.[0-9]+)?(,[0-9]{1,3})*$", `CV (%)`)) %>%
  distinct(`CV (%)`)
```

The `CV` column contains the footnote (D), (L), and (H).

(D): Withheld to avoid disclosing data for individual operations. (L): Coefficient of variation or generalized coefficient of variation is less than 0.05% or the standard error is less than 0.05% of the mean. (H): Coefficient of variation or generalized coefficient of variation is greater than or equal to 99.95% or the standard error is greater than or equal to 99.95% of the mean.

After checking the `CV` column, no additional action.

Export csv file of cleaning data

```r
write_csv(strawberry, "strawberry_cleaned.csv")
```

Strawberry Data Cleaning - Chemicals EDA

Read Strawberry Data I cleaned

```
strawberry <- read_csv("strawberry_cleaned.csv", col_names = TRUE)
```

```
## Rows: 12669 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): Program, Period, Geo Level, State, State ANSI, Ag District, County...
## dbl  (3): Year, Ag District Code, Code
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

1. Create new data table when `Domain` == Chemical & `Metric` == LB for California state

```
calif <- strawberry %>%  filter(State=="CALIFORNIA")
calif_chem_lb_o <- calif %>% filter(str_detect(Domain, "CHEMICAL"), Metric == "LB")

# Remove rows where "TOTAL" is detected in Domain Category
calif_chem_lb <- calif_chem_lb_o %>%
  filter(!str_detect(`Domain Category`, "TOTAL"))
```

2. Create new data tables for each different kind of Chemical in `Domain`

```
unique(calif_chem_lb$Domain)
```

```
## [1] "CHEMICAL, FUNGICIDE"   "CHEMICAL, INSECTICIDE" "CHEMICAL, OTHER"
## [4] "CHEMICAL, HERBICIDE"
```

```
fungicide <- calif_chem_lb %>% filter(str_detect(Domain, "FUNGICIDE"))
insecticide <- calif_chem_lb %>% filter(str_detect(Domain, "INSECTICIDE"))
other <- calif_chem_lb %>% filter(str_detect(Domain, "OTHER"))
herbicide <- calif_chem_lb %>% filter(str_detect(Domain, "HERBICIDE"))
```

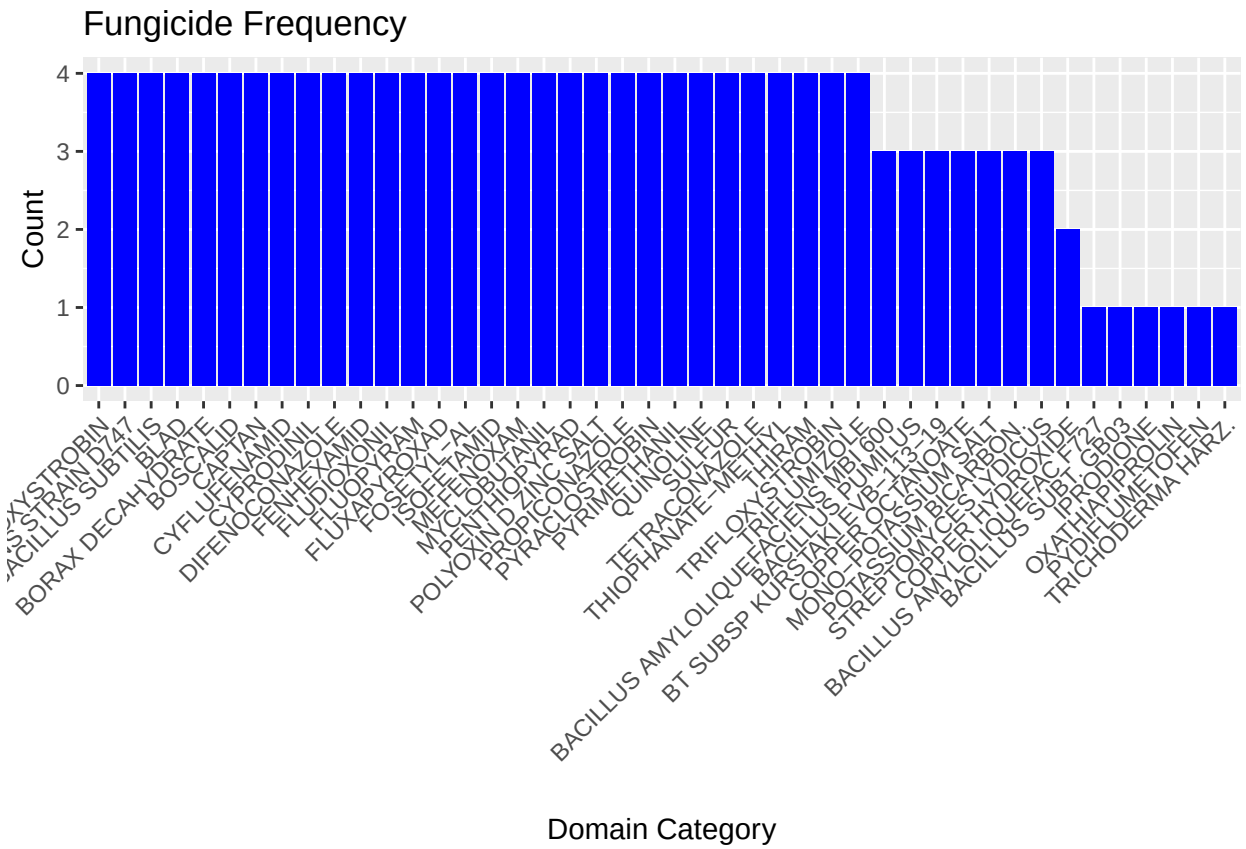
```
# Count frequencies for each chemical type
fungicide_counts <- fungicide %>%
  group_by(`Domain Category`) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count))

insecticide_counts <- insecticide %>%
  group_by(`Domain Category`) %>%
```

```
  summarise(Count = n()) %>%
  arrange(desc(Count))

other_counts <- other %>%
  group_by(`Domain Category`) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count))

herbicide_counts <- herbicide %>%
  group_by(`Domain Category`) %>%
  summarise(Count = n()) %>%
  arrange(desc(Count))
```

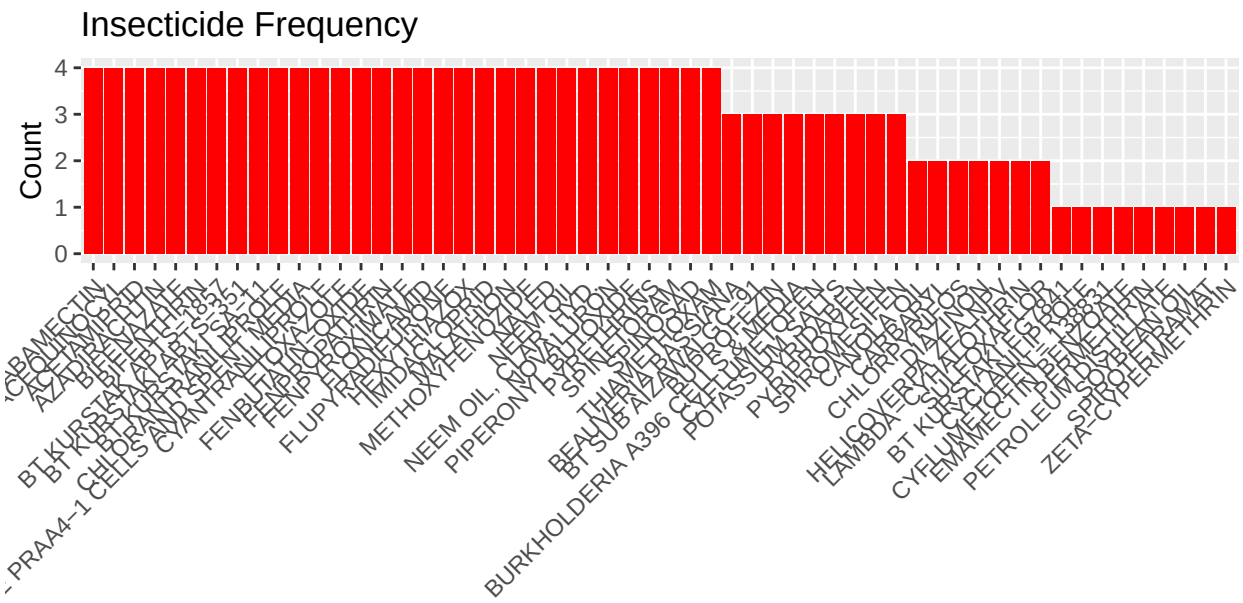
Count the frequency of each chemical name in `Domain Category` for each specific chemical type (from Step 2)

```
# Fungicides
ggplot(fungicide_counts, aes(x = reorder(`Domain Category`, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Fungicide Frequency", x = "Domain Category", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

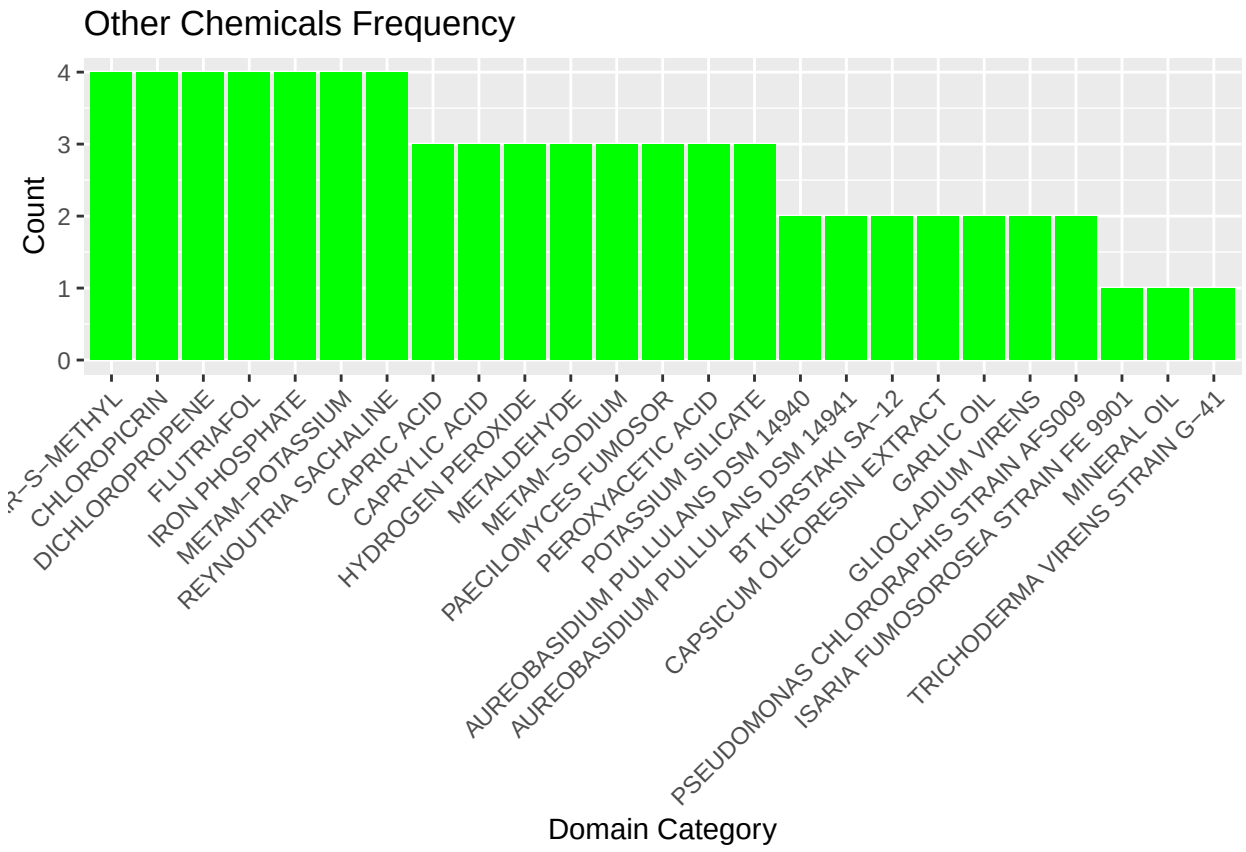
Graph histograms for each specific chemical type to show the frequency of each chemical name visually



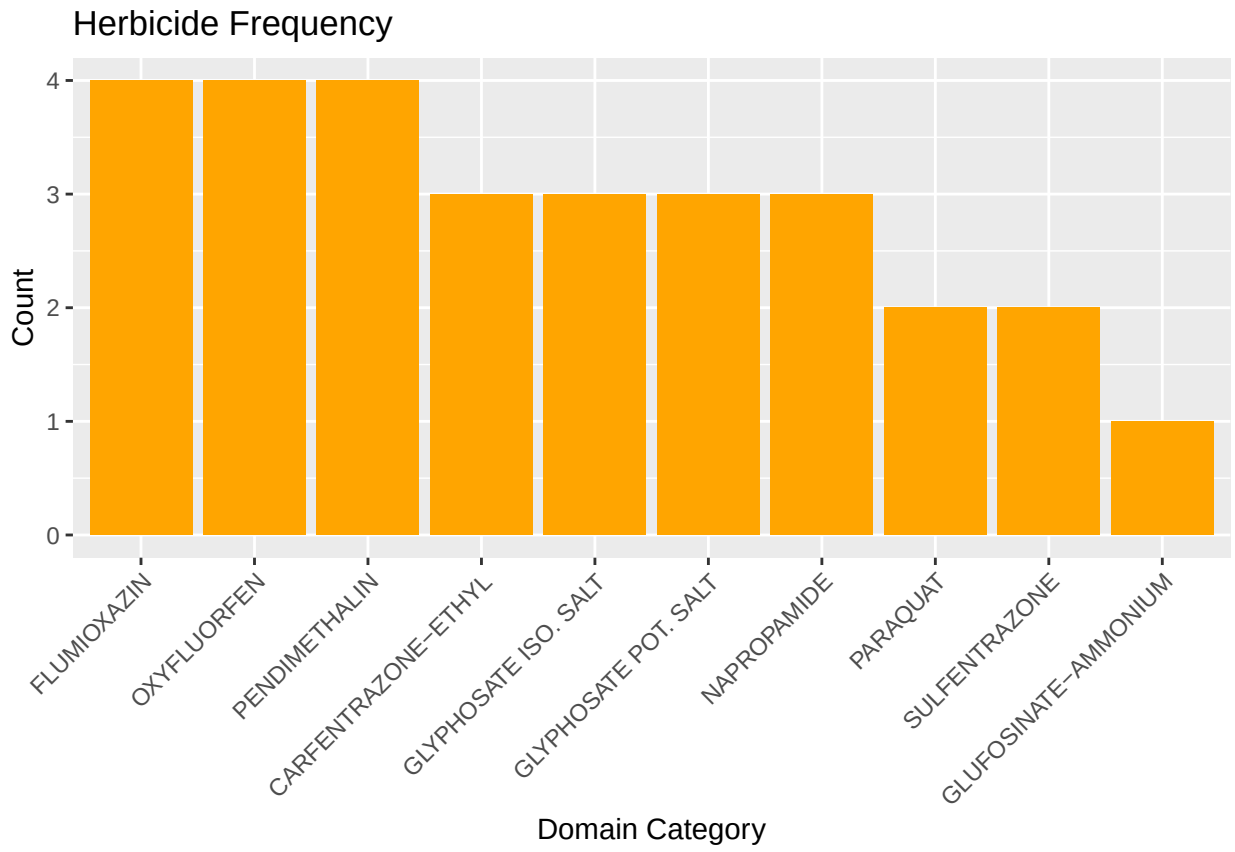
```
# Insecticides
ggplot(insecticide_counts, aes(x = reorder(`Domain Category`, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "red") +
  labs(title = "Insecticide Frequency", x = "Domain Category", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Others
ggplot(other_counts, aes(x = reorder(`Domain Category`, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "green") +
  labs(title = "Other Chemicals Frequency", x = "Domain Category", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Herbicides
ggplot(herbicide_counts, aes(x = reorder(`Domain Category`, -Count), y = Count)) +
  geom_bar(stat = "identity", fill = "orange") +
  labs(title = "Herbicide Frequency", x = "Domain Category", y = "Count") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# Count the number of different chemicals for each chemical type
num_fungicides <- fungicide %>%
  summarise(Unique_Chemicals = n_distinct(`Domain Category`))

num_insecticides <- insecticide %>%
  summarise(Unique_Chemicals = n_distinct(`Domain Category`))

num_others <- other %>%
  summarise(Unique_Chemicals = n_distinct(`Domain Category`))

num_herbicides <- herbicide %>%
  summarise(Unique_Chemicals = n_distinct(`Domain Category`))

list(
  Fungicide = num_fungicides$Unique_Chemicals,
  Insecticide = num_insecticides$Unique_Chemicals,
  Other = num_others$Unique_Chemicals,
  Herbicide = num_herbicides$Unique_Chemicals
)
```

```
## $Fungicide
## [1] 44
##
## $Insecticide
## [1] 56
##
```

```
## $Other
## [1] 25
##
## $Herbicide
## [1] 10
```

Based on the result above, I could observe that in California, there are more different chemicals using for insecticide than other and less chemicals are use for herbicide.

Question: Based on the natural concept, why there are more different chemicals using for insecticide than other, and less chemicals are used for herbicide?

- The analysis I could get based on the result above: The number of chemicals used for insecticides is higher because of the diversity of insect pests, their life cycle complexity, and the development of resistance. In contrast, herbicides are used against a narrower range of targets (weeds) and are constrained by the need to selectively kill weeds without harming crops, resulting in fewer distinct chemicals in use.

3. The chemical names with highest and lowest frequency for Insecticides and Herbicide respectively

```r
# Insecticide: Find the chemical with the highest and lowest frequency
insecticide_freq <- insecticide %>%
  count(`Domain Category`, sort = TRUE)

insecticide_max <- insecticide_freq %>%
  slice_max(n, n = 1)

insecticide_min <- insecticide_freq %>%
  slice_min(n, n = 1)

# Herbicide: Find the chemical with the highest and lowest frequency
herbicide_freq <- herbicide %>%
  count(`Domain Category`, sort = TRUE)

herbicide_max <- herbicide_freq %>%
  slice_max(n, n = 1)

herbicide_min <- herbicide_freq %>%
  slice_min(n, n = 1)
```

Based on the result above, highest frequency = 4; lowest frequency = 1:

Insecticide:

- 10 chemicals with highest frequency: ABAMECTIN, ACEQUINOCYL, ACETAMIPRID, AZADIRACHTIN, BIFENAZATE, BIFENTHRIN, BT KURSTAK ABTS-1857, BT KURSTAKI ABTS-351, BT KURSTAKI SA-11, CHLORANTRANILIPROLE.
- 9 chemicals with lowest frequency: BT KURSTAKI EG7841, CYCLANILIPROLE, CYFLUMETOFEN = 138831, EMAMECTIN BENZOATE, PERMETHRIN, PETROLEUM DISTILLATE, SOYBEAN OIL, SPIROTETRAMAT, ZETA-CYPERMETHRIN.

Herbicide:

- 3 chemicals with highest frequency: FLUMIOXAZIN, OXYFLUORFEN, PENDIMETHALIN.
- 1 chemicals with lowest frequency: GLUFOSINATE-AMMONIUM.

```
library(PubChemR)

# List all functions in PubChemR
# ls("package:PubChemR")
```

```
# Define the GHS_searcher function
GHS_searcher <- function(result_json_object) {
  # Check if the necessary parts of the result exist
  if (!is.null(result_json_object[["result"]][["Hierarchies"]])) {
    hierarchies <- result_json_object[["result"]][["Hierarchies"]][["Hierarchy"]]
    if (length(hierarchies) > 0) {
      for (i in 1:length(hierarchies)) {
        if (hierarchies[[i]][["SourceName"]] == "GHS Classification (UNECE)") {
          return(hierarchies[[i]])
        }
      }
    }
  }
  return(NULL)  # Return NULL if GHS Classification is not found
}

# Placeholder for hazards_retriever function
hazards_retriever <- function(ghs_data, full_data) {
  if (is.null(ghs_data)) {
    print("No GHS data found.")
    return(NULL)
  }

  # Assuming you want to extract specific hazard statements or information
  # Here's a simple example of how you might structure this
  hazard_statements <- ghs_data[["GHS Hazard Statements"]]

  if (!is.null(hazard_statements)) {
    print(hazard_statements)
  } else {
    print("No hazard statements available.")
  }
}
```

```
# Retrieve GHS classification for Acequinocyl
result_f <- get_pug_rest(identifier = "acequinocyl", namespace = "name", domain = "compound",
                         operation = "classification", output = "JSON")
```

```r
# Use the GHS_searcher function to extract GHS data
ghs_data <- GHS_searcher(result_f)

# Retrieve and print the hazards information
hazards_retriever(ghs_data, result_f)
```

(a) For example, choose Acequinocyl from high frequency list and Cyclaniliprole from low frequency list. Assess the chemical's safety and hazards, and analyze why we use those chemicals in high frequency and others in low frequency in Insecticide broadly.

```
## [1] "No hazard statements available."
```

```r
# Retrieve GHS classification for Cyclaniliprole
result_f <- get_pug_rest(identifier = "cyclaniliprole", namespace = "name",
                         domain = "compound", operation = "classification", output = "JSON")

# Use the GHS_searcher function to extract GHS data
ghs_data <- GHS_searcher(result_f)

# Retrieve and print the hazards information
hazards_retriever(ghs_data, result_f)
```

```
## [1] "No hazard statements available."
```

Conclusion: While Acequinocyl and Cyclaniliprole are effective insecticides that play a crucial role in pest management, their high frequency of use must be accompanied by careful consideration of their safety and environmental impacts. Adopting integrated pest management practices that emphasize responsible use and environmental stewardship can help ensure that these chemicals contribute to effective pest control while minimizing adverse effects on human health and ecosystems.

```r
# Retrieve GHS classification for Flumioxazin
result_f <- get_pug_rest(identifier = "flumioxazin", namespace = "name", domain = "compound",
                         operation = "classification", output = "JSON")

# Use the GHS_searcher function to extract GHS data
ghs_data <- GHS_searcher(result_f)

# Retrieve and print the hazards information
hazards_retriever(ghs_data, result_f)
```

(b) For example, choose Flumioxazin from high frequency list and Glufosinate-Ammonium from low frequency list. Assess the chemical's safety and hazards, and analyze why we use those chemicals in high frequency and others in low frequency in Herbicide broadly.

```
## [1] "No hazard statements available."
```

```r
# Retrieve GHS classification for Glufosinate-Ammonium
result_f <- get_pug_rest(identifier = "glufosinate-ammonium ", namespace = "name",
                         domain = "compound", operation = "classification", output = "JSON")
```

```r
# Use the GHS_searcher function to extract GHS data
ghs_data <- GHS_searcher(result_f)

# Retrieve and print the hazards information
hazards_retriever(ghs_data, result_f)
```

```
## [1] "No hazard statements available."
```

Conclusion: The high frequency of Flumioxazin in herbicide applications can be attributed to its broad-spectrum efficacy and effectiveness in controlling resistant weeds, despite its associated health and environmental risks. Conversely, Glufosinate-Ammonium is used less frequently due to its broader acute toxicity concerns and limited target spectrum, along with potential reproductive and organ damage risks. The choice between these herbicides reflects a balance between effectiveness, safety, and environmental stewardship, emphasizing the need for integrated pest management practices that consider the specific requirements of the agricultural context.