Strawberry EDA

Zhenwei Weng

2024-10-21

```r
library(dplyr)
```

```
##
## 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
```

```r
straw<-read.csv("straw_eda.csv")

straw <-straw %>%
  mutate(Value = na_if(Value, "(D)")) %>%
  mutate(Value = na_if(Value, "(NA)")) %>%
  mutate(Value = as.numeric(Value)) %>%
  filter(!is.na(Value) & is.finite(Value))
```

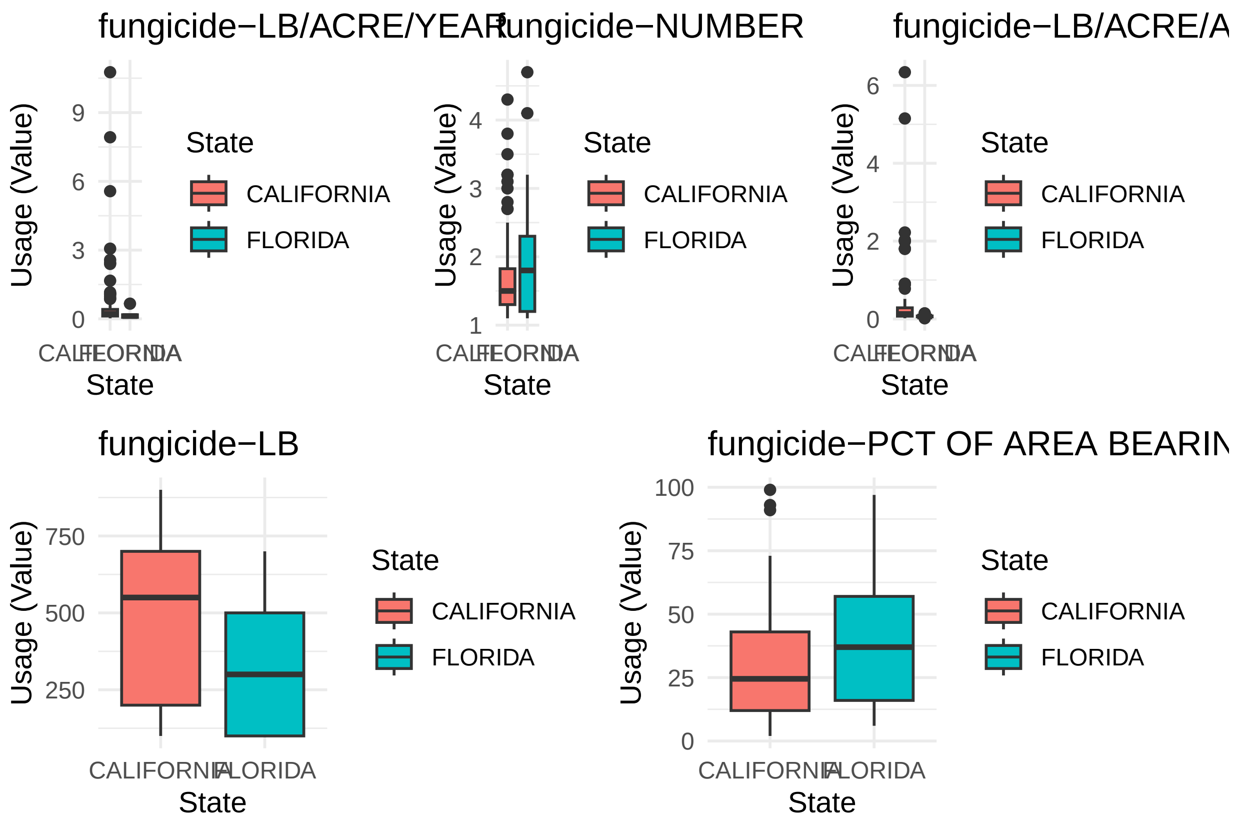
```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Value = as.numeric(Value)`.
## Caused by warning:
## ! NAs introduced by coercion
```

```r
data_ins <- straw %>%
  filter(type == "INSECTICIDE")

data_fun <- straw %>%
  filter(type == "FUNGICIDE")

data_her <- straw %>%
  filter(type == "HERBICIDE")

data_other <- straw %>%
  filter(type == "OTHER")

ins_1 <- data_ins %>%
  filter(measure == "LB / ACRE / YEAR")
```

```r
ins_2 <- data_ins %>%
  filter(measure == "NUMBER")

ins_3 <- data_ins %>%
  filter(measure == "LB / ACRE / APPLICATION")

ins_4 <- data_ins %>%
  filter(measure == "LB")

ins_5 <- data_ins %>%
  filter(measure == "PCT OF AREA BEARING")

fun_1 <- data_fun %>%
  filter(measure == "LB / ACRE / YEAR")

fun_2 <- data_fun %>%
  filter(measure == "NUMBER")

fun_3 <- data_fun %>%
  filter(measure == "LB / ACRE / APPLICATION")

fun_4 <- data_fun %>%
  filter(measure == "LB")

fun_5 <- data_fun %>%
  filter(measure == "PCT OF AREA BEARING")


herb_1 <- data_her %>%
  filter(measure == "LB / ACRE / YEAR")

herb_2 <- data_her %>%
  filter(measure == "NUMBER")

herb_3 <- data_her %>%
  filter(measure == "LB / ACRE / APPLICATION")

herb_4 <- data_her %>%
  filter(measure == "LB")

herb_5 <- data_her %>%
  filter(measure == "PCT OF AREA BEARING")

other_1 <- data_other %>%
  filter(measure == "LB / ACRE / YEAR")

other_2 <- data_other %>%
  filter(measure == "NUMBER")

other_3 <- data_other %>%
  filter(measure == "LB / ACRE / APPLICATION")

other_4 <- data_other %>%
  filter(measure == "LB")

other_5 <- data_other %>%
  filter(measure == "PCT OF AREA BEARING")


library(ggplot2)
library(patchwork)

plot_metric <- function(data, title) {
  if (nrow(data) == 0 || !("Value" %in% names(data)) || sum(!is.na(data$Value)) == 0) {
    return(ggplot() +
             labs(title = paste(title, "(No Data Available)")) +
             theme_void())
  }

  ggplot(data, aes(x = State, y = as.numeric(Value), fill = State)) +
    geom_boxplot() +
    labs(title = title, x = "State", y = "Usage (Value)") +
    theme_minimal()
}
```

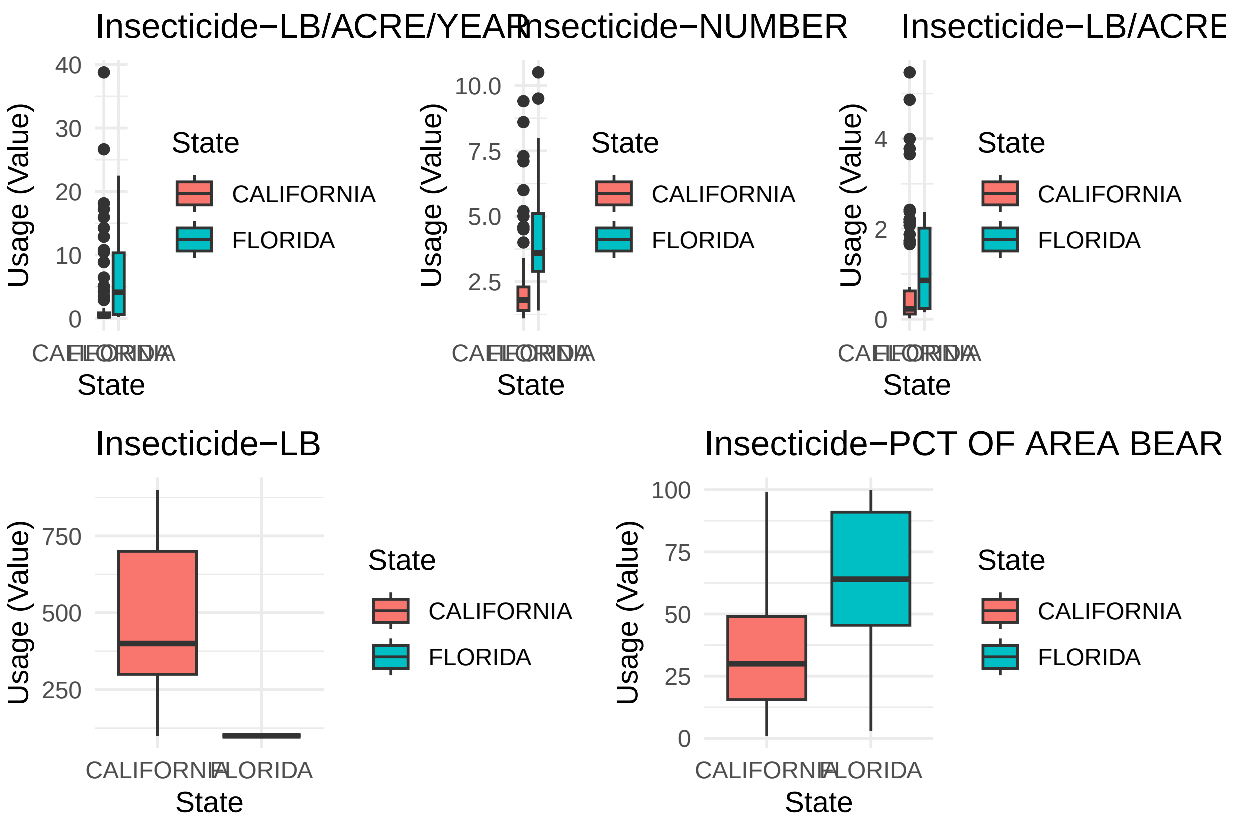
Plots for FUNGICIDE

```r
ins_1_plot <- plot_metric(ins_1, "fungicide-LB/ACRE/YEAR")
ins_2_plot <- plot_metric(ins_2, "fungicide-NUMBER")
ins_3_plot <- plot_metric(ins_3, "fungicide-LB/ACRE/APPLICATION")
ins_4_plot <- plot_metric(ins_4, "fungicide-LB")
ins_5_plot <- plot_metric(ins_5, "fungicide-PCT OF AREA BEARING")
insecticide_plots <- (ins_1_plot | ins_2_plot | ins_3_plot) /
                     (ins_4_plot | ins_5_plot)
print(insecticide_plots)
```



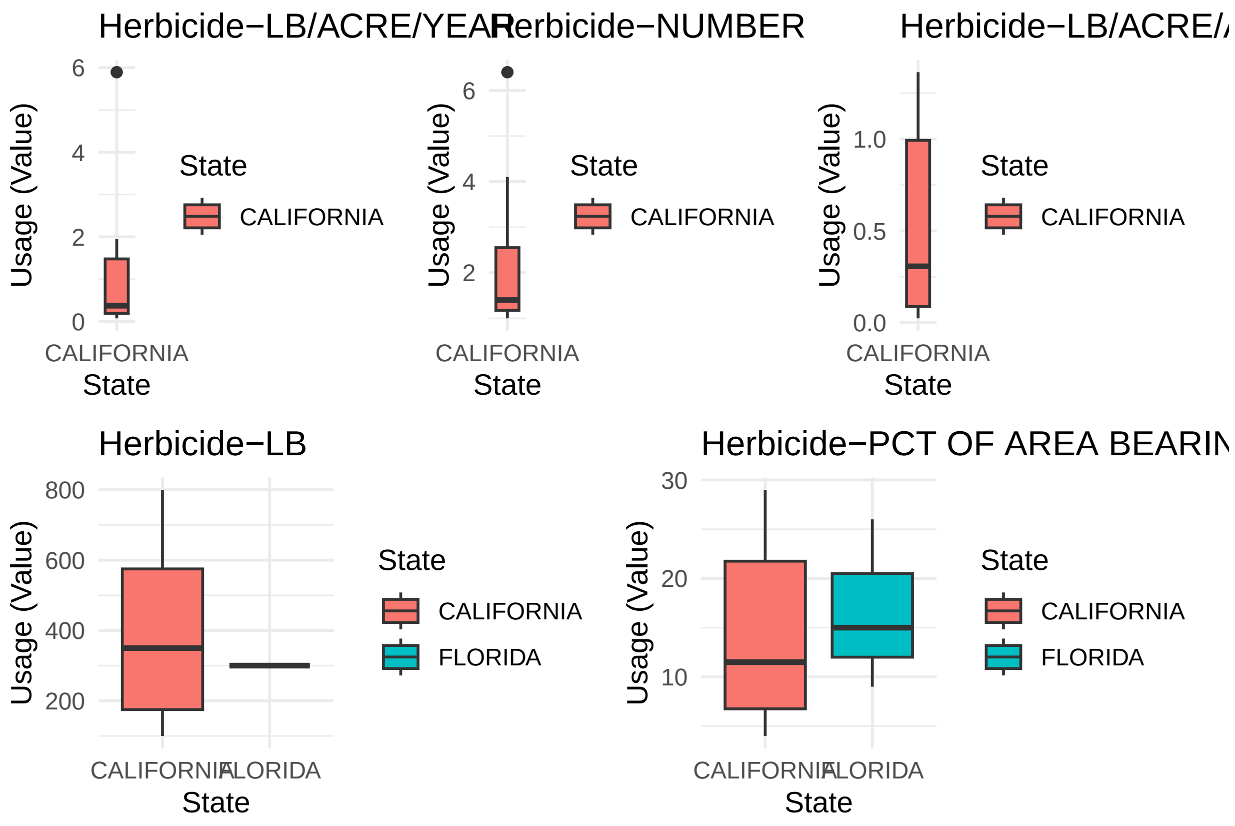
Plots for Insecticide

```r
fun_1_plot <- plot_metric(fun_1, "Insecticide-LB/ACRE/YEAR")
fun_2_plot <- plot_metric(fun_2, "Insecticide-NUMBER")
fun_3_plot <- plot_metric(fun_3, "Insecticide-LB/ACRE/APPLICATION")
fun_4_plot <- plot_metric(fun_4, "Insecticide-LB")
fun_5_plot <- plot_metric(fun_5, "Insecticide-PCT OF AREA BEARING")
fungicide <- (fun_1_plot | fun_2_plot | fun_3_plot) /
                     (fun_4_plot | fun_5_plot)

print(fungicide)
```



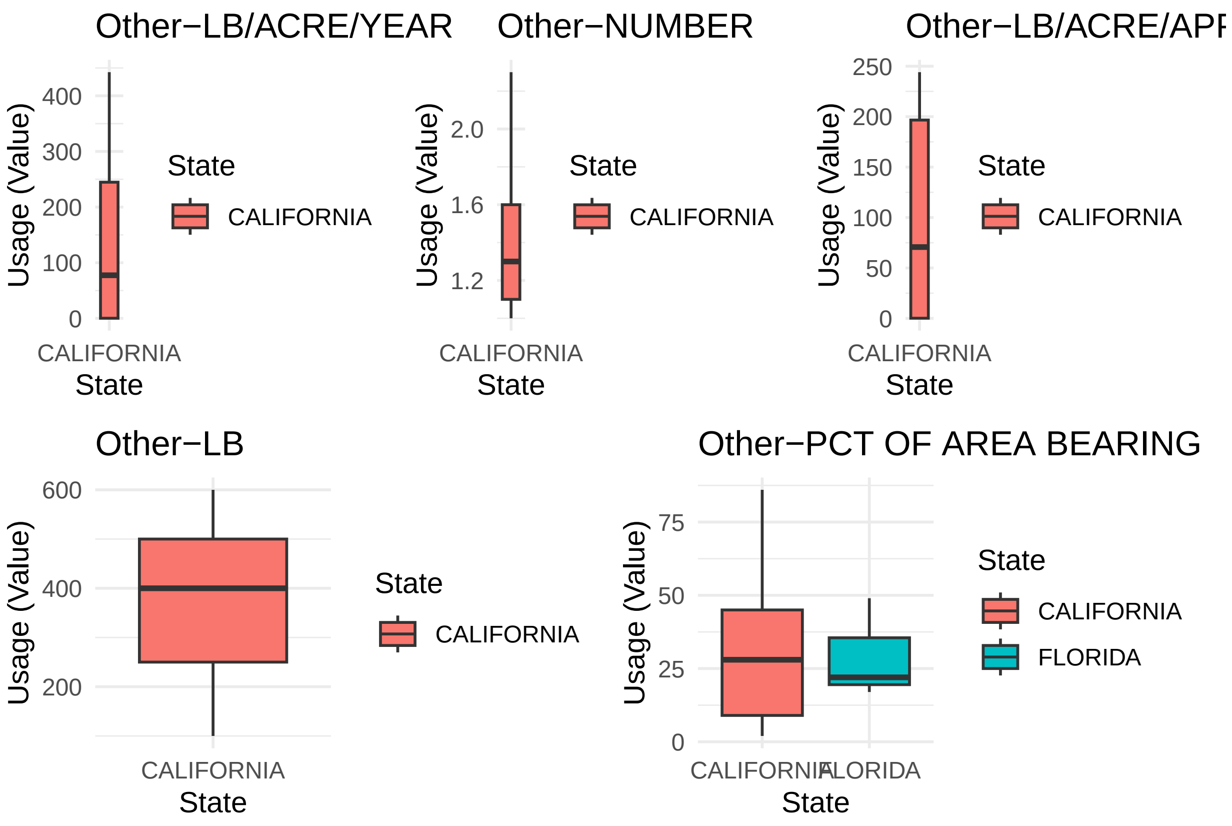
Plots for Heribicide

```r
herb_1_plot <- plot_metric(herb_1, "Herbicide-LB/ACRE/YEAR")
herb_2_plot <- plot_metric(herb_2, "Herbicide-NUMBER")
herb_3_plot <- plot_metric(herb_3, "Herbicide-LB/ACRE/APPLICATION")
herb_4_plot <- plot_metric(herb_4, "Herbicide-LB")
herb_5_plot <- plot_metric(herb_5, "Herbicide-PCT OF AREA BEARING")

herbicide_plots <- (herb_1_plot | herb_2_plot | herb_3_plot) /
                   (herb_4_plot | herb_5_plot)
print(herbicide_plots)
```



Plots for Other chemical type

```
other_1_plot <- plot_metric(other_1, "Other-LB/ACRE/YEAR")
other_2_plot <- plot_metric(other_2, "Other-NUMBER")
other_3_plot <- plot_metric(other_3, "Other-LB/ACRE/APPLICATION")
other_4_plot <- plot_metric(other_4, "Other-LB")
other_5_plot <- plot_metric(other_5, "Other-PCT OF AREA BEARING")

other_plots <- (other_1_plot | other_2_plot | other_3_plot) /
               (other_4_plot | other_5_plot)
print(other_plots)
```



Setup for GHS searcher and harzard retriever

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## v readr     2.1.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(PubChemR)
GHS_searcher<-function(result_json_object){
  result<-result_json_object
  for (i in 1:length(result[["result"]][["Hierarchies"]][["Hierarchy"]])){
    if(result[["result"]][["Hierarchies"]][["Hierarchy"]][[i]]
       [["SourceName"]]=="GHS Classification (UNECE)"){
      return(i)
    }

  }
}
```

```r
hazards_retriever<-function(index,result_json_object){
  result<-result_json_object
  hierarchy<-result[["result"]][["Hierarchies"]][["Hierarchy"]][[index]]
  i<-1
  output_list<-rep(NA,length(hierarchy[["Node"]]))
  while(str_detect(hierarchy[["Node"]][[i]][["Information"]][["Name"]],"H") &
        i<length(hierarchy[["Node"]])){
    output_list[i]<-hierarchy[["Node"]][[i]][["Information"]][["Name"]]
    i<-i+1
  }
  return(output_list[!is.na(output_list)])
}
```

Find the most common chemical

```r
most_frequent_chem <- straw %>%
  group_by(type, chem_name) %>%
  summarise(count = n(), .groups = 'drop') %>%
  group_by(type) %>%
  filter(count == max(count)) %>%
  arrange(desc(count))
```

```r
chemical_vec<-c("ABAMECTIN", "CYPRODINIL","FLUDIOXONIL")

result_f<-get_pug_rest(identifier = "FLUDIOXONIL", namespace = "name", domain = "compound",
                       operation="classification", output = "JSON")

hazards_retriever(GHS_searcher(result_f),result_f)
```

```
## [1] "H320: Causes eye irritation [Warning Serious eye damage/eye irritation]"
## [2] "H300: Health Hazards"
## [3] "Hazard Statement Codes"
## [4] "H351: Suspected of causing cancer [Warning Carcinogenicity]"
## [5] "H400: Very toxic to aquatic life [Warning Hazardous to the aquatic environment, acute hazard]"
## [6] "H400: Environmental Hazards"
## [7] "H410: Very toxic to aquatic life with long lasting effects [Warning Hazardous to the aquatic en
```

```r
result_c<-get_pug_rest(identifier = "CYPRODINIL", namespace = "name", domain = "compound",
                       operation="classification", output = "JSON")

hazards_retriever(GHS_searcher(result_c),result_c)
```

```
##  [1] "H315: Causes skin irritation [Warning Skin corrosion/irritation]"
##  [2] "H300: Health Hazards"
##  [3] "Hazard Statement Codes"
##  [4] "H317: May cause an allergic skin reaction [Warning Sensitization, Skin]"
##  [5] "H319: Causes serious eye irritation [Warning Serious eye damage/eye irritation]"
##  [6] "H372: Causes damage to organs through prolonged or repeated exposure [Danger Specific target o
##  [7] "H400: Very toxic to aquatic life [Warning Hazardous to the aquatic environment, acute hazard]"
##  [8] "H400: Environmental Hazards"
##  [9] "H401: Toxic to aquatic life [Hazardous to the aquatic environment, acute hazard]"
## [10] "H410: Very toxic to aquatic life with long lasting effects [Warning Hazardous to the aquatic e
## [11] "H411: Toxic to aquatic life with long lasting effects [Hazardous to the aquatic environment, lo
## [12] "Hazardous to the aquatic environment, acute hazard"
## [13] "Environmental Hazards"
## [14] "Hazard Classes"
## [15] "Hazardous to the aquatic environment, long-term hazard"
```

```
result_a<-get_pug_rest(identifier = "ABAMECTIN", namespace = "name", domain = "compound",
                       operation="classification", output = "JSON")

hazards_retriever(GHS_searcher(result_a),result_a)
```

```
## [1] "H260: In contact with water releases flammable gases which may ignite spontaneously [Danger Subs
## [2] "H200: Physical Hazards"
## [3] "Hazard Statement Codes"
## [4] "H314: Causes severe skin burns and eye damage [Danger Skin corrosion/irritation]"
## [5] "H300: Health Hazards"
## [6] "H317: May cause an allergic skin reaction [Warning Sensitization, Skin]"
## [7] "H334: May cause allergy or asthma symptoms or breathing difficulties if inhaled [Danger Sensiti
## [8] "H373: May causes damage to organs through prolonged or repeated exposure [Warning Specific targe
```

I get the hazard data for the most common chemical

```
library(PubChemR)
chemical_vec <- c("ABAMECTIN", "CYPRODINIL", "FLUDIOXONIL")
get_ghs_info <- function(chemical_name) {
  result <- get_pug_rest(identifier = chemical_name, namespace = "name", domain = "compound",
                         operation = "classification", output = "JSON")
  ghs_index <- GHS_searcher(result)

  if (!is.null(ghs_index)) {
    hazards <- hazards_retriever(ghs_index, result)
    return(paste(hazards, collapse = ", "))
  } else {
    return("No GHS Information Found")
  }
}
hazards_data <- data.frame(
  Chemical = character(),
  Hazards = character(),
  stringsAsFactors = FALSE
)
for (chemical in chemical_vec) {
  ghs_info <- get_ghs_info(chemical)
  hazards_data <- rbind(hazards_data, data.frame(Chemical = chemical, Hazards = ghs_info, stringsAsFacto
}
print(hazards_data)
```

```
##      Chemical
## 1   ABAMECTIN
## 2  CYPRODINIL
## 3 FLUDIOXONIL
##
```

```
## 1
## 2 H315: Causes skin irritation [Warning Skin corrosion/irritation], H300: Health Hazards, Hazard Sta
## 3
```

Now I count the hazard.

```r
hazard_counts <- hazards_data %>%
  separate_rows(Hazards, sep = ",\\s*") %>%
  mutate(Hazard_Code = sub(":.*", "", Hazards)) %>%
  filter(grepl("^H\\d+", Hazard_Code)) %>%
  count(Hazard_Code, sort = TRUE)
print(hazard_counts)
```

```
## # A tibble: 16 x 2
##    Hazard_Code     n
##    <chr>       <int>
##  1 H400            4
##  2 H300            3
##  3 H317            2
##  4 H410            2
##  5 H200            1
##  6 H260            1
##  7 H314            1
##  8 H315            1
##  9 H319            1
## 10 H320            1
## 11 H334            1
## 12 H351            1
## 13 H372            1
## 14 H373            1
## 15 H401            1
## 16 H411            1
```

According to the information I get from GHS, H300 and H400 are high hazard codes: H300: Acute toxicity, highly lethal. H400: Very toxic to aquatic life.

H410 and H317 also belong to a less hazard level: H410: Very toxic to aquatic life with long-lasting effects. H317: May cause an allergic skin reaction.

ABAMECTIN contains H300 and H400, and Abamectin is most common in INSECTICIDE, so from the graph for INSECTICIDE we can see it's used more in Florida than California, which means strawberry from California may healther than strawberry from Florida.

For CYPRODINIL and FLUDIOXONIL, harzard in Cyprodinil contains H317 which be considered as less harzard level than H300 and H400, and Fludioxonil contains H410 wich is very toxic to aquatic life but strawberry is not aquatic life.

In this paper I ask some help from Chatgpt for unerstanding how to use GHS_searcher.