Strawberries EDA

Amie Thomas

2023-10-16

Introduction

How do you choose your food? Do you go for the cheapest option? The food that is the quickest to prepare? Have you ever thought about the process your food goes through before it gets to your table? According to the World Health Organization, you should. The following includes an analysis of states that produced the most organic strawberries in the year 2021. Additionally, there is an analysis of the amount of pesticides used on the strawberries produced in California, the largest Strawberry producer in the United States.

```
strawberry <- read.csv("strawberry(1).csv")

#looking at data before
glimpse(strawberry)
```

```
## Rows: 4,314
## Columns: 21
## $ Program          <chr> "CENSUS", "CENSUS", "CENSUS", "CENSUS", "CENSUS", "CE~
## $ Year             <int> 2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021,~
## $ Period           <chr> "YEAR", "YEAR", "YEAR", "YEAR", "YEAR", "YEAR", "YEAR~
## $ Week.Ending      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Geo.Level        <chr> "STATE", "STATE", "STATE", "STATE", "STATE", "STATE",~
## $ State            <chr> "ALASKA", "ALASKA", "ALASKA", "ALASKA", "ALASKA", "AL~
## $ State.ANSI       <int> 2, 2, 2, 2, 2, 2, 2, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 9,~
## $ Ag.District      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Ag.District.Code <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ County           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ County.ANSI      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Zip.Code         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Region           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ watershed_code   <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
## $ Watershed        <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ Commodity        <chr> "STRAWBERRIES", "STRAWBERRIES", "STRAWBERRIES", "STRA~
## $ Data.Item        <chr> "STRAWBERRIES, ORGANIC - OPERATIONS WITH SALES", "STR~
## $ Domain           <chr> "ORGANIC STATUS", "ORGANIC STATUS", "ORGANIC STATUS",~
## $ Domain.Category  <chr> "ORGANIC STATUS: (NOP USDA CERTIFIED)", "ORGANIC STAT~
## $ Value            <chr> "2", " (D)", " (D)", " (D)", "2", " (D)", " (D)", "14~
## $ CV....           <chr> "(H)", "(D)", "(D)", "(D)", "(H)", "(D)", "(D)", "19.~
```

< Get rid of single value columns. Haviland's code >

```
drop_single_val <- function(df){
  col_val <- NULL
  col_name <- NULL
suppressWarnings({
  for(i in 1:dim(df)[2]){
if((df |> distinct(df[,i]) |> count()) == 1){
```

```
  col_name = c(col_name, colnames(df[i]))
  col_val = c(col_val, df[1,i])
} }
})

if(is.null(col_name)){return("No Columns to drop")}else{
   col_val = unlist(col_val)
   attributes(col_val) = NULL
   drp = data.frame(col_name, col_val)
   return(drp)
   }
}

str <- drop_single_val(strawberry)

str <- str$col_name

strawberry <- strawberry |> select(!all_of(str))
```

```
#take a gander at the data after
glimpse(strawberry)
```

```
## Rows: 4,314
## Columns: 10
## $ Program         <chr> "CENSUS", "CENSUS", "CENSUS", "CENSUS", "CENSUS", "CEN~
## $ Year            <int> 2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021, 2021, ~
## $ Period          <chr> "YEAR", "YEAR", "YEAR", "YEAR", "YEAR", "YEAR", "YEAR"~
## $ State           <chr> "ALASKA", "ALASKA", "ALASKA", "ALASKA", "ALASKA", "ALA~
## $ State.ANSI      <int> 2, 2, 2, 2, 2, 2, 2, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 9, ~
## $ Data.Item       <chr> "STRAWBERRIES, ORGANIC - OPERATIONS WITH SALES", "STRA~
## $ Domain          <chr> "ORGANIC STATUS", "ORGANIC STATUS", "ORGANIC STATUS", ~
## $ Domain.Category <chr> "ORGANIC STATUS: (NOP USDA CERTIFIED)", "ORGANIC STATU~
## $ Value           <chr> "2", " (D)", " (D)", " (D)", "2", " (D)", " (D)", "142~
## $ CV....          <chr> "(H)", "(D)", "(D)", "(D)", "(H)", "(D)", "(D)", "19.2~
```

```
#check if every row has is associated with a state
###question to self: why do we need to do this
state_all <- strawberry |> group_by(State) |> count()

### test if every row is associated with a state by summing the
## counts and testing for equality with the total rows in the
## data frame

if(sum(state_all$n) == dim(strawberry)[1]){print("Every row has value in the State column.")}
```

```
## [1] "Every row has value in the State column."
```

So far, Haviland has removed missing values from the columns. And I think checked for missing values in the rows. Why did Haviland find the state with the most rows. Ask about the difference in his notations. Is this a common way of notation

```
#which state has the most rows?
#how does this information help us?
state_max <- state_all$State[which(state_all$n ==  max(state_all$n)  )]
```

```
strwb_survey <- strawberry |> filter(Program == "SURVEY")
strwb_census <- strawberry |> filter(Program == "CENSUS")
```

Need to separate the columns that have more than one type of information. Domain item and Domain category

```
strwb_census <- strwb_census |>
  separate_wider_delim(  cols = `Data.Item`,
                         delim = ",",
                         names = c("Fruit",
                                 "temp1",
                                 "temp2",
                                 "temp3"),
                         too_many = "error",
                         too_few = "align_start"
                       )
#what does the too many and too few do in this

strwb_census <- strwb_census |>
  separate_wider_delim(  cols = temp1,
                         delim = " - ",
                         names = c("crop_type",
                                 "prop_acct"),
                         too_many = "error",
                         too_few = "align_start"
                       )

## trim the strings
## you can see which columns contain string values that need
## to have leading or trailing spaces that need to be trimmed.


# glimpse(strwb_census)

strwb_census$crop_type <- str_trim(strwb_census$crop_type, side = "both")

strwb_census$temp2 <- str_trim(strwb_census$temp2, side = "both")

strwb_census$temp3 <- str_trim(strwb_census$temp3, side = "both")
```

< Fresh market column >

```
## substitute a space for NA in prop_acct column
strwb_census$prop_acct[is.na(strwb_census$prop_acct)] <- ""

## substitute a space for NA in temp2 column
strwb_census$temp2[is.na(strwb_census$temp2)] <- ""

## substitute a space for NA in temp2 column
strwb_census$temp3[is.na(strwb_census$temp3)] <- ""
```

```
strwb_census <- strwb_census |> unite(temp2, temp3, col="Metric", sep="")

## Now fix the entries in the Metric column
##   Remove "MEASURED IN " from the cells
```

```r
strwb_census$Metric <- strwb_census$Metric |> str_replace("MEASURED IN ", "")

## move Metric to the end
strwb_census <- strwb_census |> relocate(Metric, .before = Domain)

strwb_census <- strwb_census |> relocate(`Process Market`, .before = Metric)

strwb_census <- strwb_census |> rename(Totals = prop_acct)

#drop_one_value_col(strwb_census)
```

```r
## remove commas from numbers
## fix footnotes

## basic tools

## start by getting the Values column so you can work on it

vals <- strwb_census$Value

## note where vals goes in the environment.

## tools -- 2 choices  base R, and stringr package

## BaseR -- Piping??


g1 <- sub(",", "", vals)
# vals[1:20]
# g1[1:20]


g2 <- gsub(",", "", vals)
# vals[1:20]
# g2[1:20]


## stringr - str_replace(), str_replace_all()

## LOOK -- see ref for stingr pkg
a <- vals |> str_detect(",")

# vals[1:20]
# a[1:20]

## Still strings!!

b <- vals |> str_replace(",", "")
# vals[1:20]
# b[1:20]

c <- vals |> str_replace_all(",", "")
# vals[1:20]
# c[1:20]
```

```
## Now notice what happens when the
## the strings of digits are cast to numerics.

## for example
c <- as.numeric(c)
```

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

```
# c[1:20]


### remove commas from Value entries
dcomma <- function(c){
  x_new <- as.numeric(gsub(",", "", c))
  return(x_new)
}



########################################  footnotes

## finds single uppor case Character in parens in s2
## e.g. "(D)"

## To fine the location and value of the footnotes

v <- strwb_census$Value


## find the footnote locations
## fn_i: locations
fn_i <- v |> str_detect("^\\([:upper:]\\)$") ## returns


## dcomma returns numbers and NA's
v1 <- dcomma(v)
```

```
## Warning in dcomma(v): NAs introduced by coercion
```

```
## locations of NA's
na_i <- is.na(v1)

## Demonstration that the locations of the footnotes
## are the same as the locations of the NA's

# length(v) == sum(na_i == fn_i)

## update dcomma()
## Integrate transformation of the values column and
## reporting the footnote values.


dcomma <- function(c){
  suppressWarnings({
  xnew = as.numeric(gsub(",", "", c))
```

```
  fns = unique(c[is.na(xnew)])
  vtran = list("new_vec" = xnew, "footnotes" = fns)
  return(vtran)
  })
}


v_trns <- dcomma(v)


 a <- v_trns$new_vec
 # a[1:20]

 # v_trns$footnotes
```

```
per_c <- strwb_survey |> select(Period) |> distinct()
per_c <- unlist(per_c)


## the Period column denotes
## three periods for data collection
##    marketing year
##    year
##    year - Aug Forecast
```

```
## remove commas from numbers
## fix footnotes

## basic tools

## start by getting the Values column so you can work on it

vals <- strwb_survey$Value

## note where vals goes in the environment.

## tools -- 2 choices  base R, and stringr package

## BaseR -- Piping??


g1 <- sub(",", "", vals)
# vals[1:20]
# g1[1:20]


g2 <- gsub(",", "", vals)
# vals[1:20]
# g2[1:20]


## stringr - str_replace(), str_replace_all()

## LOOK -- see ref for stingr pkg
```

```
a <- vals |> str_detect(",")

# vals[1:20]
# a[1:20]

## Still strings!!

b <- vals |> str_replace(",", "")
# vals[1:20]
# b[1:20]

c <- vals |> str_replace_all(",", "")
# vals[1:20]
# c[1:20]

## Now notice what happens when the
## the strings of digits are cast to numerics.

## for example
c <- as.numeric(c)
```

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

```
# c[1:20]


### remove commas from Value entries
dcomma <- function(c){
  x_new <- as.numeric(gsub(",", "", c))
  return(x_new)
}



##########################################  footnotes

## finds single uppor case Character in parens in s2
## e.g. "(D)"

## To fine the location and value of the footnotes

v <- strwb_survey$Value


## find the footnote locations
## fn_i: locations
fn_i <- v |> str_detect("^\\([:upper:]\\)$") ## returns


## dcomma returns numbers and NA's
v1 <- dcomma(v)
```

```
## Warning in dcomma(v): NAs introduced by coercion
```

```
## locations of NA's
na_i <- is.na(v1)

## Demonstration that the locations of the footnotes
## are the same as the locations of the NA's

# length(v) == sum(na_i == fn_i)

## update dcomma()
## Integrate transformation of the values column and
## reporting the footnote values.


dcomma <- function(c){
  suppressWarnings({
  xnew = as.numeric(gsub(",", "", c))
  fns = unique(c[is.na(xnew)])
  vtran = list("new_vec" = xnew, "footnotes" = fns)
  return(vtran)
  })
}


v_trns <- dcomma(v)


 a <- v_trns$new_vec
 # a[1:20]

 # v_trns$footnotes
```

```
strwb_survey <- strwb_survey |>
  separate_wider_delim(  cols = `Data.Item`,
                         delim = ",",
                         names = c("temp1",
                                   "temp2",
                                   "temp3",
                                   "temp4"),
                         too_many = "error",
                         too_few = "align_start"
                       )

strwb_survey <- strwb_survey |>
  separate_wider_delim(  cols = temp1,
                         delim = " - ",
                         names = c("temp1a",
                                   "temp1b"),
                         too_many = "error",
                         too_few = "align_start"
                       )
```

```
strwb_survey <- strwb_survey |>
  separate_wider_delim(  cols = Domain,
                         delim = ",",
```

```
                          names = c("temp22",
                                    "temp23"),
                          too_many = "error",
                          too_few = "align_start"
                        )


t22 <- unique(strwb_survey$temp22)

t23 <- unique(strwb_survey$temp23)
```

```
strwb_survey <- strwb_survey |>
  separate_wider_delim(  cols = `Domain.Category`,
                         delim = ",",
                         names = c("temp42",
                                   "temp43",
                                   "temp44",
                                   "temp45"),
                         too_many = "error",
                         too_few = "align_start"
                        )


## temp22 or temp42 or both == CHEMICAL
##  else the row contains market data



strwb_survey_chem <- strwb_survey |> filter((temp22 == "CHEMICAL") | (temp42 == "CHEMICAL"))

strwb_survey_mkt <- strwb_survey |> filter(!((temp22 == "CHEMICAL") | (temp42 == "CHEMICAL")))
```

End of Haviland code

```
chem1 <-  drop_single_val(strwb_survey_chem)

# chem1 |> kable(caption = "1-value columns dropped")

chem1 <- setdiff(colnames(strwb_survey_chem), chem1$col_name)

strwb_survey_chem <- strwb_survey_chem |> select(all_of(chem1))
```

```
mkt1 <- drop_single_val(strwb_survey_mkt)

strwb_survey_mkt <- strwb_survey_mkt |>
  separate_wider_delim(  cols = `temp1a`,
                         delim = ",",
                         names = c("crop","type"),
                         too_many = "merge",
                         too_few = "align_start"
                        )
strwb_survey_mkt <- strwb_survey_mkt |>
  separate_wider_delim(  cols = `type`,
```

```
                          delim = ",",
                          names = c("typ","temp1"),
                          too_many = "merge",
                          too_few = "align_start"
                        )

# mkt1 |> kable(caption = "droping 1-value cols - mkt")
```

```
strwb_survey_mkt <- strwb_survey_mkt %>%
  select(-typ, -temp1, -temp23, -temp43, -temp44, -temp45)
```

```
## substitute a space for NA in prop_acct column
strwb_survey_mkt$temp1b[is.na(strwb_survey_mkt$temp1b)] <- ""

## substitute a space for NA in temp2 column
strwb_survey_mkt$temp2[is.na(strwb_survey_mkt$temp2)] <- ""

## substitute a space for NA in temp2 column
strwb_survey_mkt$temp3[is.na(strwb_survey_mkt$temp3)] <- ""

strwb_survey_mkt$temp4[is.na(strwb_survey_mkt$temp4)] <- ""

strwb_survey_mkt$State.ANSI[is.na(strwb_survey_mkt$State.ANSI)] <- ""

colnames(strwb_survey_mkt)[colnames(strwb_survey_mkt) == "temp1b"] <- "Pest"

colnames(strwb_survey_mkt)[colnames(strwb_survey_mkt) == "temp2"] <- "Metric_1"

colnames(strwb_survey_mkt)[colnames(strwb_survey_mkt) == "temp3"] <- "Metric_2"

colnames(strwb_survey_mkt)[colnames(strwb_survey_mkt) == "temp4"] <- "AVG"

strwb_survey_mkt$Metric_1 <- str_trim(strwb_survey_mkt$Metric_1, side = "both")

strwb_survey_mkt$Metric_2 <- str_trim(strwb_survey_mkt$Metric_2, side = "both")

strwb_survey_mkt$Value <- str_trim(strwb_survey_mkt$Value, side = "both")

strwb_survey_mkt$State <- str_trim(strwb_survey_mkt$State, side = "both")

strwb_census$Value <- str_trim(strwb_census$Value, side = "both")

strwb_census$Year <- str_trim(strwb_census$Year, side = "both")
```

```
strwb_survey_mkt <- subset(strwb_survey_mkt, Value!= " (D)")
```

```
colnames(strwb_survey_mkt)[colnames(strwb_survey_mkt) == "temp1b"] ="prop_acct"
```

```
strwb_survey_chem$temp4[is.na(strwb_survey_chem$temp4)] <- ""

strwb_survey_chem$temp44[is.na(strwb_survey_chem$temp44)] <- ""

strwb_survey_chem$temp45[is.na(strwb_survey_chem$temp45)] <- ""
```

```
strwb_survey_chem <- separate(strwb_survey_chem, temp43, into = c("temp23", "tempchem"), sep = ":")
```

```
strwb_survey_chem <- subset(strwb_survey_chem, Value!= " (Z)")

strwb_survey_chem <- subset(strwb_survey_chem, Value!= " (D)")

strwb_survey_chem <- subset(strwb_survey_chem, Value!= " (NA)")
```

```
strwb_survey_chem$tempchem <- str_trim(strwb_survey_chem$tempchem, side = "both")
strwb_survey_chem <- separate(strwb_survey_chem, tempchem, into = c("chem_name", "chem_total"), sep = "
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 49 rows [24, 30, 50, 57,
## 65, 113, 174, 241, 259, 260, 263, 264, 312, 319, 340, 350, 358, 411, 480, 558,
## ...].
```

```
strwb_survey_chem$chem_name <- substr(strwb_survey_chem$chem_name, 2, nchar(strwb_survey_chem$chem_name
strwb_survey_chem$chem_total <- substr(strwb_survey_chem$chem_total, 1, nchar(strwb_survey_chem$chem_to

strwb_survey_chem <- strwb_survey_chem %>%
  select(-temp44)

strwb_survey_chem <- strwb_survey_chem %>%
  select(-temp45)
```

```
library(readr)
chemical <- read_csv("chemical.csv")
```

```
## Rows: 1174 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (8): State, temp3, temp4, temp23, temp43, temp44, chem_name, harm_level
## dbl (3): Year, State.ANSI, chem_column
## num (1): Value
## lgl (1): temp45
##
## 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.
```

```
harm_level <- chemical$harm_level

strwb_survey_chem$harm_level <- harm_level

value_mapping <- c("Ia" = "Extremely hazardous", "Ib" = "Highly hazardous",
                   "II" = "Moderately hazardous", "III" = "Slightly hazardous",
                   "U" = "Unlikely to present acute hazard in normal use",
                   "FM" = "Fumigant", "O" = "Obsolete as pesticide")

strwb_survey_chem$lvl_descript <- value_mapping[strwb_survey_chem$harm_level]
```

Questions for EDA Market data what state produces the most for the fresh market

```
# Market <- strwb_survey_mkt %>%
#   filter(Metric_1 == "PROCESSING") |>
#   group_by(State)
```

Chemical data What pesticides are used and how does it vary by state? What state uses the most harmful pesticides? -what does each harm level mean -need to by state -eventually group by harm level

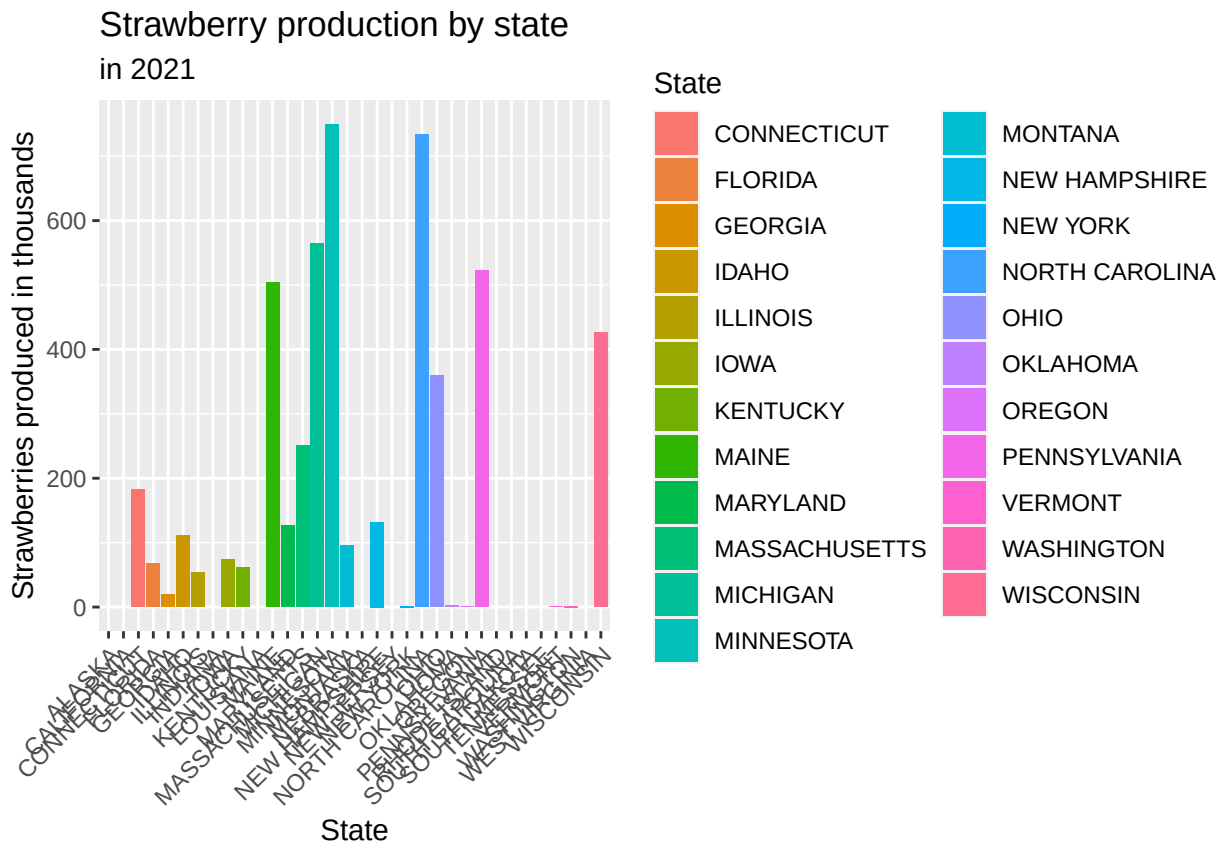
```
# Replace commas with periods and convert to numeric
strwb_census$Value <- as.numeric(gsub(",", "\\.", strwb_census$Value))
```

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

```
#top 10 states with highest production of strawberries
 production_2021 <- strwb_census |>
  filter(Year == 2021) |>
  filter(Totals == "PRODUCTION") |>
  arrange(desc(Value))


ggplot(production_2021, aes(x = State, y = Value, fill = State))+
  geom_bar(stat = "identity")+
  labs(x = "State", y = "Strawberries produced in thousands", subtitle = "in 2021") +
  ggtitle("Strawberry production by state")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Removed 11 rows containing missing values (`position_stack()`).
```

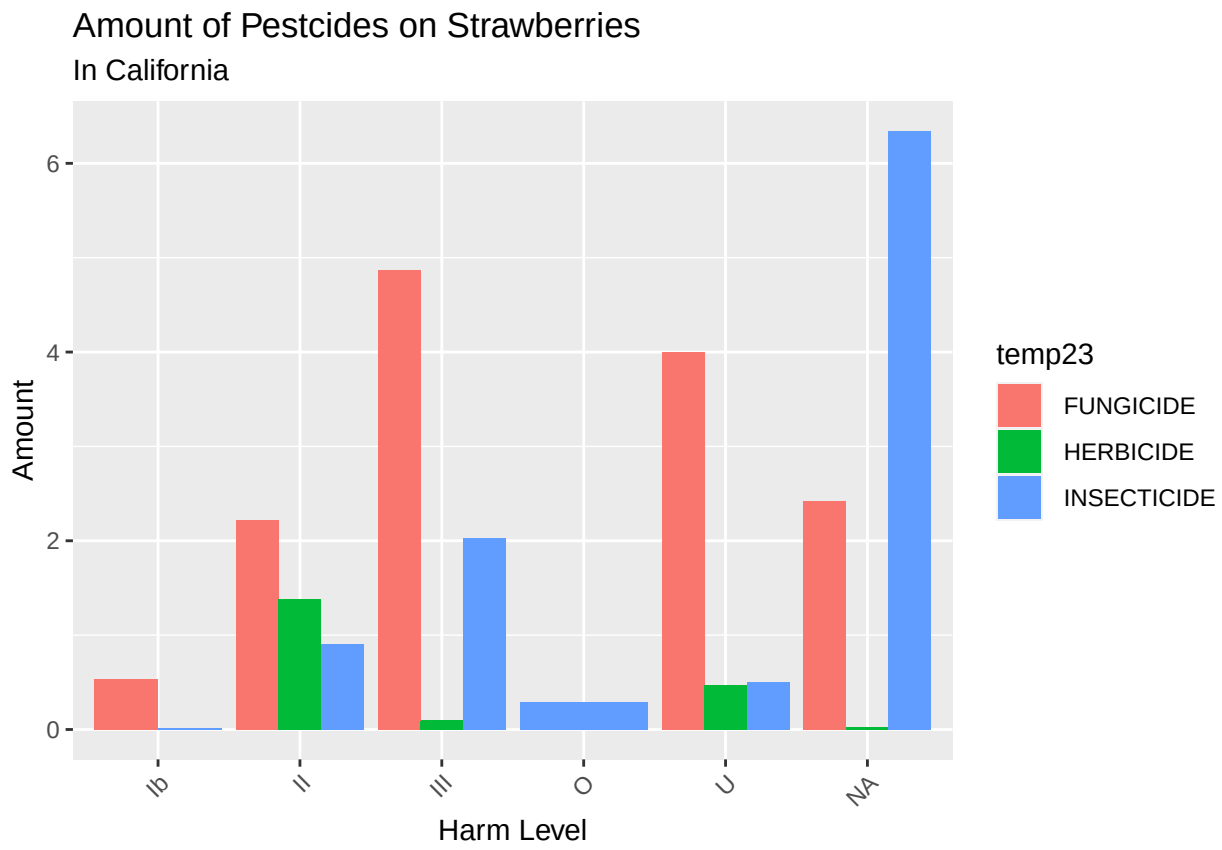



```
strwb_survey_chem$Value <- as.numeric(gsub(",", "\\.", strwb_survey_chem$Value))
```

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

```
California <- strwb_survey_chem |>
  filter(temp3 == " MEASURED IN LB / ACRE / APPLICATION")|>
  filter(temp23 != " OTHER")|>
  filter(State == "CALIFORNIA")
```

```
ggplot(California, aes(x = harm_level, y = Value, fill = temp23))+
  geom_bar(stat = "identity", position = "dodge")+
  labs(x = "Harm Level", y = "Amount", subtitle = "In California") +
  ggtitle("Amount of Pestcides on Strawberries")+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



#Conclusion

Ia" = "Extremely hazardous", "Ib" = "Highly hazardous", "II" = "Moderately hazardous", "III" = "Slightly hazardous", "U" = "Unlikely to present acute hazard in normal use", "FM" = "Fumigant", "O" = "Obsolete as pesticide"

Here is the key for the interpretation of the harm levels in the graph above.