MA615-EDA-Strawberry

Mingrui Du

Overview

Objective

The objective of this assignment is to practice data cleaning then exploratory data analysis(EDA) on data set "Strawberry"

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v forcats   1.0.0     v readr     2.1.4
v ggplot2   3.4.3     v stringr   1.5.0
v lubridate 1.9.3     v tibble    3.2.1
v purrr     1.0.2     v tidyr     1.3.0
-- 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 beco
```

Data acquisition

"Strawberry" selected from USDA NASS , original data stored here

The Data

The original data set consists of 21 columns, among which "Week.Ending","Ag.District", "Ag.District.Code","County","County.ANSI","Zip.Code","Region","Watershed" have only NA elements, and "Geo.Level"(=STATE), "watershed_code"(=0),"Commodity"(=STRAWBERRIES) are single-value columns.

Column "Program" contains two sources of data: CENSUS and SURVEY.

"Data.Item" is a concatenation of six mixed columns:

- Commodity = STRAWBERRIES in our case
- class_desc: recording a physical attribute (not recorded in this strawberry case)
- prodn_practice_desc: a method of production or action taken on the commodity(e.g., IRRIGATED, ORGANIC, ON FEED)
- util_practice_desc: Utilizations (e.g., GRAIN, FROZEN, SLAUGHTER) or marketing channels (e.g., FRESH MARKET, PROCESSING, RETAIL)
- statisticcat_desc: The aspect of a commodity being measured (e.g., AREA HARVESTED, PRICE RECEIVED, INVENTORY, SALES).
- unit.

"Domain":

- domain = ORGANIC STATUS for organic commodity;
- for chemical usage data, the domain describes the type of chemical applied to the commodity(e.g., FUNGICIDE, HERBICIDE, INSECTICIDE, FERTILIZER, OTHER);
- domain = TOTAL will have no further breakouts.

"Domain.Category" records categories or partitions within a domain. For instance, the specific chemical taken for a commodity, along with PC code.

"Value" contains published data value or suppression reason code.

Data assessment

"Data.Item" compresses multiple columns into one; "Domain.Category" contains various chemicals(and PC codes) that require extraction; "Value" involves numbers in different units, as well as abbreviation codes, so does "CV".

Here I first delete single value columns then split dataset into CENSUS and SURVEY, since organic commodity lies in CENSUS and chemical usage stored in the latter.

Initial questions

- Initial questions about strawberries, the data, and about the work you are undertaking. Write these before you begin working.

Data cleaning and organization

```
## split CENSUS and SURVEY
strwb_census <- strawberry |> filter(Program == "CENSUS")
strwb_survey <- strawberry |> filter(Program == "SURVEY")
#nrow(strawberry) == (nrow(strwb_census) + nrow(strwb_survey))
```

```
## No need to deal with Domain.Category in CENSUS as it's single value.
#unique(strwb_census$Domain.Category)
#unique(strwb_census$Domain)

## Seperate Data.Item into 4 columns at most, the first set to be commodity(STRAWBERRIES)
#unique(strwb_census$Data.Item)
strwb_census <- strwb_census |>
                separate_wider_delim(cols = `Data.Item`,
                                     delim = ",",
                                     names = c("Commodity", ## STRAWBERRIES
                                               "temp1",
                                               "temp2",
                                               "temp3"),
                                     too_many = "error",
                                     too_few = "align_start")
strwb_census$temp1 <- trimws(strwb_census$temp1)
strwb_census$temp2 <- trimws(strwb_census$temp2)
strwb_census$temp3 <- trimws(strwb_census$temp3)

## Separate temp1 into Production Practice(prodn_practice_desc) and prop_acct
#unique(strwb_census$temp1)
#strwb_census |> distinct(temp1)
strwb_census <- strwb_census |>
  separate_wider_delim(  cols = temp1,
                         delim = " - ",
                         names = c("prodn_practice_desc",
                                   "prop_acct"),
                         too_many = "error",
                         too_few = "align_start"
                       )
#unique(strwb_census$prodn_practice_desc)
#unique(strwb_census$prop_acct)
### We can see now NA appears while cleaning data.
```

```r
## Separate temp2 into marketing channels(Fresh Market,Processing) and units(unit_desc)
strwb_census <- strwb_census |> mutate(`Fresh Market` = temp2, .after = temp2)
strwb_census$`Fresh Market` <- strwb_census$`Fresh Market` |>
                               str_replace( "^MEA.*", "")|>
                               str_replace( "^P.*", "")
strwb_census$`Fresh Market`[is.na(strwb_census$`Fresh Market`)] <- ""
strwb_census$temp2 <- strwb_census$temp2 |> str_replace("^F.*", "")
strwb_census$`Fresh Market` <- strwb_census$`Fresh Market` |> str_replace("^FRESH MARKET -
#unique(strwb_census$temp2)
strwb_census <- strwb_census |> mutate(`Processing` = temp2, .after = temp2)
strwb_census$`Processing` <- strwb_census$`Processing` |>
                               str_replace( "^MEA.*", "")
strwb_census$`Processing`[is.na(strwb_census$`Processing`)] <- ""
strwb_census$temp2 <- strwb_census$temp2 |> str_replace("^P.*", "")
strwb_census$`Processing` <- strwb_census$`Processing` |> str_replace("^PROCESSING - ", ""
#unique(strwb_census$temp2)
#unique(strwb_census$Processing)
strwb_census$prop_acct[is.na(strwb_census$prop_acct)] <- ""
strwb_census$temp2[is.na(strwb_census$temp2)] <- ""
strwb_census$temp3[is.na(strwb_census$temp3)] <- ""

## Now combine temp2 and temp3 into a column for units
strwb_census <- strwb_census |> unite(temp2, temp3, col="unit_desc", sep="")
#unique(strwb_census$unit_desc)
strwb_census$unit_desc <- strwb_census$unit_desc |> str_replace("^MEASURED IN ", "")
## These separated column have no NA inside
#unique(!is.na(strwb_census$unit_desc))
#unique(!is.na(strwb_census$prop_acct))
#unique(!is.na(strwb_census$`Fresh Market`))
#unique(!is.na(strwb_census$Processing))

## Remove single-value columns:
# Program(CENSUS), Commodity(STRAWBERRIES), prodn_practice_desc(ORGANIC), Domain(ORGANIC S
strwb_census <- strwb_census %>% select_if(~length(unique(.)) > 1)
```

```r
## function to remove comma in numbers
dcomma <- function(c){
  suppressWarnings({
  xnew = as.numeric(gsub(",", "", c))
  fns = unique(c[is.na(xnew)])
  vtran = list("new_vec" = xnew, "footnotes" = fns)
```

```
  return(vtran)
  })
}
c <- data.frame(dcomma(strwb_census$Value))
strwb_census$Value <- c[,1]
strwb_census$fn <- c[,2]

## Separate values by units($, CWT)
strwb_census$Value_USD <- ifelse(strwb_census$unit_desc == "$", strwb_census$Value, "")
strwb_census$Value_CWT <- ifelse(strwb_census$unit_desc == "CWT", strwb_census$Value, "")

# data_USD <- strwb_census |>
#   select(c(Year, State, `Fresh Market`, unit_desc, Value)) |>
#   filter((unit_desc == '$') & (`Fresh Market` == 'SALES') & (Value != "(D)"))
```

```
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("Commodity",
                                   "temp1b"),
                         too_many = "error",
                         too_few = "align_start"
                       )
### delete head & tail spaces
strwb_survey$temp4 <- trimws(strwb_survey$temp4)
strwb_survey$temp2 <- trimws(strwb_survey$temp2)
strwb_survey$temp3 <- trimws(strwb_survey$temp3)

## Divide Data.Item into "Fresh Market"  "Processing" "Bearing" "Utilized"
### Fresh Market
```

```
strwb_survey <- strwb_survey |> mutate(`Fresh Market` = temp2, .after = temp2)
strwb_survey$`Fresh Market` <- strwb_survey$`Fresh Market` |>
                                str_replace( "^MEA.*", "") |>
                                str_replace( "^P.*", "") |>
                                str_replace( "^N.*", "") |>
                                str_replace( "^U.*", "") |>
                                str_replace( "^B.*", "")

strwb_survey$`Fresh Market`[is.na(strwb_survey$`Fresh Market`)] <- ""
strwb_survey$temp2 <- strwb_survey$temp2 |> str_replace("^F.*", "")
strwb_survey$`Fresh Market` <- strwb_survey$`Fresh Market` |> str_replace("^FRESH MARKET -
### Processing
strwb_survey <- strwb_survey |> mutate(Processing = temp2, .after = temp2)
strwb_survey$Processing <- strwb_survey$Processing |>
                                str_replace( "^MEA.*", "") |>
                                str_replace( "^N.*", "") |>
                                str_replace( "^U.*", "") |>
                                str_replace( "^B.*", "")
strwb_survey$Processing[is.na(strwb_survey$Processing)] <- ""
strwb_survey$temp2 <- strwb_survey$temp2 |> str_replace("^P.*", "")
strwb_survey$Processing <- strwb_survey$Processing |> str_replace("^PROCESSING - ", "")
### UTILIZED
strwb_survey <- strwb_survey |> mutate(Utilized = temp2, .after = temp2)
strwb_survey$Utilized <- strwb_survey$Utilized |>
                                str_replace( "^MEA.*", "") |>
                                str_replace( "^N.*", "") |>
                                str_replace( "^B.*", "")
strwb_survey$Utilized[is.na(strwb_survey$Utilized)] <- ""
strwb_survey$temp2 <- strwb_survey$temp2 |> str_replace("^U.*", "")
strwb_survey$Utilized <- strwb_survey$Utilized |> str_replace("^UTILIZED - ", "")
### NOT SOLD
strwb_survey <- strwb_survey |> mutate(`Not Sold` = temp2, .after = temp2)
strwb_survey$`Not Sold` <- strwb_survey$`Not Sold` |>
                                str_replace( "^MEA.*", "") |>
                                str_replace( "^B.*", "")
strwb_survey$`Not Sold`[is.na(strwb_survey$`Not Sold`)] <- ""
strwb_survey$temp2 <- strwb_survey$temp2 |> str_replace("^N.*", "")
strwb_survey$`Not Sold` <- strwb_survey$`Not Sold` |> str_replace("^NOT SOLD - ", "")
### Bearing
strwb_survey <- strwb_survey |> mutate(Bearing = temp2, .after = temp2)
strwb_survey$Bearing <- strwb_survey$Bearing |>
```

```
                                str_replace( "^MEA.*", "")
strwb_survey$Bearing[is.na(strwb_survey$Bearing)] <- ""
strwb_survey$temp2 <- strwb_survey$temp2 |> str_replace("^B.*", "")
strwb_survey$Bearing <- strwb_survey$Bearing |> str_replace("^BEARING - ", "")
#strwb_survey |> distinct(temp2)
## temp2 clear, except MEASURED IN
## temp3
#strwb_survey |> distinct(temp3)
## temp3 contains "Utilized" and "Measured in xx" to separate
strwb_survey$temp3[is.na(strwb_survey$temp3)] <- ""  ## prepare for str_detect
strwb_survey$Utilized[str_detect(strwb_survey$temp3, "^U.*")] <- strwb_survey$temp3[str_de
strwb_survey$temp3 <- strwb_survey$temp3 |> str_replace("^U.*", "")
#strwb_survey |> distinct(temp3)
## temp3 clear, except MEASURED IN
## combine temp2 and 3
strwb_survey <- strwb_survey |> unite(temp2, temp3, col="unit_desc", sep="")
## add "MEASURED IN xx" from temp4
strwb_survey$temp4[is.na(strwb_survey$temp4)] <- ""
strwb_survey$unit_desc[str_detect(strwb_survey$temp4, "^MEA.*")] <- strwb_survey$temp4[str
strwb_survey$temp4 <- strwb_survey$temp4 |> str_replace("^MEA.*", "")
strwb_survey$Utilized <- strwb_survey$Utilized |> str_replace("^UTILIZED - ", "")
strwb_survey$unit_desc <- strwb_survey$unit_desc |> str_replace("^MEASURED IN ", "")
#strwb_survey <- strwb_survey %>% select(-temp3)
```

```
d <- data.frame(dcomma(strwb_survey$Value))
strwb_survey$Value <- d[,1]
strwb_survey$fn <- d[,2]
```

```
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$temp23[is.na(strwb_survey$temp23)] <- ""
strwb_survey$temp23 <- trimws(strwb_survey$temp23)
```

```r
strwb_survey$temp22 <- trimws(strwb_survey$temp22)
```

```r
strwb_survey$PC <- str_extract_all(strwb_survey$Domain.Category, "= \\d+", simplify = T)
strwb_survey$PC <- gsub("= ", "", strwb_survey$PC)
strwb_survey$PC <- strwb_survey$PC[,1]
```

```r
library(httr)
library(jsonlite)
```

```
Attaching package: 'jsonlite'
```

```
The following object is masked from 'package:purrr':

    flatten
```

```r
get_cas <- function(PC){
    PC <- sprintf("%06d", as.numeric(PC))
    path <- paste0("https://ordspub.epa.gov/ords/pesticides/apprilapi/?q=%7b%22ais%22:%7b%
                   PC,"%22%7d%7d")
    r <- GET(url = path)
    r_text <- content(r, as = "text", encoding = "UTF-8")
    df <- fromJSON(r_text, flatten = TRUE)
    df_strwb <- df$items[grepl("Strawberries", df$items$sites, fixed=T),]
    ais <- df_strwb$ais[1]
    pattern <- "\\(([^A-Za-z]+)\\/([0-9-]+)\\)"
    text <- ais
    matches <- regmatches(text, gregexpr(pattern, text))
    cas <- sapply(matches, function(x) gsub(".*\\/([0-9-]+)\\)", "\\1", x))
    if (is.character(cas)) {
        return(cas[1])
}
    else {
        return("can't find")
}
}
## Create a dictionary for PC code to CAS number
PC <- unique(strwb_survey$PC)[-1]
n = length(PC)
```

```r
dic <- data.frame(PC, CAS = rep(NA,n))

for(i in 1:n){
   dic$CAS[i] <- get_cas(PC[i])
}
## Dictionary Fix
dic$CAS[130] <- "can't find"
dic$CAS[79] <- "8002-65-1"
dic$CAS[85] <- "8003-34-7"
dic$CAS[109] <- "can't find"
dic$CAS[98] <- "76674-21-0"
dic$CAS[69] <- "39515-41-8"
dic$CAS[9] <- "188425-85-6"
dic$CAS[163] <- "57754-85-5"
dic$CAS[141] <- "32341-80-3"
dic$CAS[94] <- "124-07-2"
dic$CAS[121] <- "133-32-4"
dic$CAS[96] <- "76-06-2"
dic$CAS[35] <- "23564-05-8"
dic$CAS[136] <- "can't find"
dic$CAS[83] <- "51-03-6"
dic$CAS[100] <- "7722-84-1"
dic$CAS[95] <- "8023-77-6"
dic$CAS[82] <- "64742-89-8"
## The fix above is just a rough check for function get_cas();
## As shown, the function cannot always return correct results,
## further modification required;
## Since little time left, I will leave this part unfinished,
## transfer no CAS num into toxicity rate, and discuss later on.
## This procedure reminds us how important double check is
strwb_survey <- left_join(strwb_survey, dic, by= "PC")
```

I will complete this part after getting all PC-CAS conversion correctly.

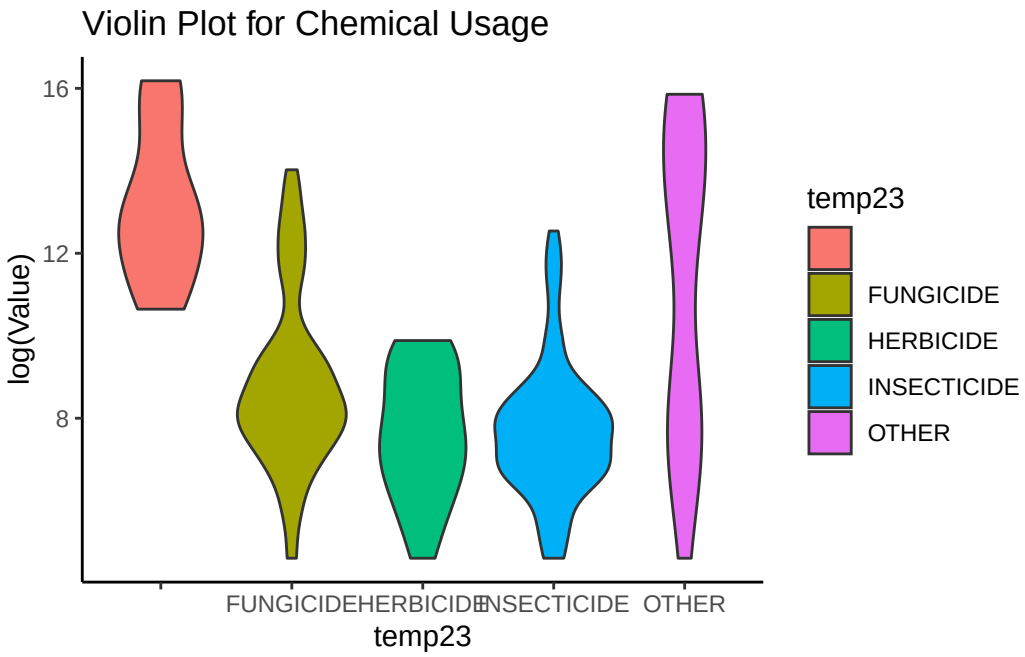
Exploratory Data Analysis

```r
# state_all <- strawberry |> group_by(State) |> count()
# state_all <- subset(state_all, State != "OTHER STATES")
## Strawberry production in California and Florida are exceptionally high since they appea
## Percentile for organic production in each state
```

```r
# state_all_census <- strwb_census |> group_by(State) |> count()
# organic <- data.frame(state_all$State, c(state_all_census[,2] / state_all[,2]))

LB <- strwb_survey %>%
  filter(unit_desc == 'LB' & Value>0 )

LB %>%
  ggplot(aes(x=temp23, y=log(Value), fill=temp23)) +
    geom_violin() +
    theme_classic() +
    theme()+
  labs(title = "Violin Plot for Chemical Usage", xlab = "Chemical Types", ylab = "Chemical
```



References

USDA NASS

EPA Pesticide Product and Label System

PPLS API

Abbreviation

Column Definitions