Strawberry Analysis - MA615

Jordan Stout

2024-10-10

In this document I will be explaining the process of cleaning open-source USDA data on Strawberry production in the US from 2018 to 2024. The cleaned data is exported as 4 separate CSV files: survey_data and census_data includes data on strawberry production that was collected through surveys and the census respectively, chemicals includes all the data regarding chemical use in strawberry production, and census_organic_data includes all data involving organic strawberry sales as per the census.

First we will load the data

```
strawberry <- read_csv("strawberries25_v3.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.
```

Now we check to make sure that every row is associated with a state.

```
state_all <- strawberry |> distinct(State)

state_all1 <- strawberry |> group_by(State) |> count()
```

Some columns of the data only have one value which is useless to us. To assist us in data cleaning we create a function 'drop_one_value_col()' which will remove such columns.

```
drop_one_value_col <- function(df){
  drop <- NULL
  for(i in 1:dim(df)[2]){
    if((df |> distinct(df[,i]) |> count()) == 1){
    drop = c(drop, i)
    }
  }
  if(is.null(drop)){
    return("none")}
  else{
     strawberry <- df[, -1*drop]
  }
```

```
}
strawberry <- drop_one_value_col(strawberry)
drop_one_value_col(strawberry)
```

```
## [1] "none"
```

Start by removing bad values from 'Value' and 'CV (%)' columns and make 'Value' column numeric type

```
strawberry$Value[strawberry$Value == "(D)"] <- NA
strawberry$Value[strawberry$Value == "(Z)"] <- NA

strawberry$`CV (%)`[strawberry$`CV (%)` == "(D)"] <- NA
strawberry$`CV (%)`[strawberry$`CV (%)` == "(L)"] <- NA
strawberry$`CV (%)`[strawberry$`CV (%)` == "(H)"] <- NA
```

We will only keep values of 'Geo Level' that include NATIONAL AND STATE becasue the rest is useless to us.

```
strawberry <- strawberry |>
  filter(`Geo Level`== "NATIONAL" | `Geo Level`== "STATE")
```

We will create two new data frames. 'census' will contain all the data where the program was the census, likewise 'survey' will include those where the program was survey.

```
census <- strawberry |>
  filter(Program=="CENSUS")

census <- census |>
  drop_one_value_col()

survey <- strawberry |>
  filter(Program == "SURVEY")

survey <- survey |>
  drop_one_value_col()

nrow(strawberry) == (nrow(survey) + nrow(census))
```

```
## [1] TRUE
```

Clean and isolate the chemicals used by creating two new columns: 'chem_name' and 'chem_num'

```
survey <- survey |>
  mutate(`Domain Category` = gsub(".*: \\(([^=]+) = ([0-9]+)\\)", "\\1,\\2", `Domain Category`)) |>
  separate(`Domain Category`, into = c("chem_name", "chem_num"), sep = ",") |>
  mutate(chem_name = trimws(chem_name), chem_num = as.numeric(trimws(chem_num)))
```

Extract 'use' from 'domain' column

```
survey <- survey |>
  separate(Domain, into = c("Domain", "use"), sep = ",", extra = "merge")
```

Use stringr + regexs to sort out the 'Data Item' column

```
survey <- survey |>
  mutate(measurement = str_extract(`Data Item`, "(?<=MEASURED\\s).*")) |>
  mutate(`Data Item` = str_remove(`Data Item`, "MEASURED.*")) |>
  separate(`Data Item`, into = c("Data Item", "category"), sep = "[,-]", extra = "merge", fill = "right
  mutate(measurement = gsub("\\bIN\\b", "", measurement)) |>
  mutate(measurement = trimws(measurement))
```

Remove 'Data Item' column because it has no more data that we need as well as remove unwanted commas

```
survey <- survey |>
  select(-`Data Item`)
survey$category <- gsub(",", "", survey$category)
```

Separate chemicals and total

```
total <- survey |>
  filter(Domain == "TOTAL")

chemicals <- survey |>
  filter(Domain == "CHEMICAL")

total <- drop_one_value_col(total)
chemicals <- drop_one_value_col(chemicals)
```

Organize 'Data Item' column in Census data

```
census <- census |>
  separate_wider_delim(cols = `Data Item`, delim = " - ", names = c("strawberries", "Category"), too_ma
```

Make a new dataframe out of the census specific data that includes only organic strawberry sales

```
census <- census |>
  separate_wider_delim(cols = strawberries, delim = ", ", names = c("strawberries", "ORGANIC", "organic
                       too_few = "align_start") |>
  drop_one_value_col()

census_organic <- census |>
  filter(ORGANIC == "ORGANIC")

census <- census[(is.na(census$ORGANIC)),]

census <- census |>
  drop_one_value_col()
```

Now we can split the category column in census into two columns. Measure and bearing to organize the bearing type.

```
census <- census |>
  separate_wider_delim(cols = `Category`, delim = " ", names = c("COL1", "COL2"), too_many = "merge", to

census$COL2 <- str_replace(census$COL2,"WITH ","")

census <- census |>
  rename(Measure = COL1, Bearing_type= COL2)
```

Split up 'Domain Category' column in the census dataframe

```
census <- census |>
  rename(size_bracket = `Domain Category`)

census$size_bracket <- str_replace(census$size_bracket, "NOT SPECIFIED", "TOTAL")

census$size_bracket <- str_replace(census$size_bracket, "AREA GROWN: ", "")
```

Now onto the organic census data. First lets drop all the one value columns

```
census_organic <- census_organic |>
  drop_one_value_col()
```

Now we can clean "Category" column similar to how we cleaned it with the survey data. By splitting the columns at the word measure.

```
census_organic <- census_organic |>
  separate(Category, into = c("Category", "measurement"), sep = " MEASURED ", extra = "merge", fill = "
```

Get rid of commas and 'IN' from the new column it is just units again.

```
census_organic$Category <- gsub(",", "", census_organic$Category)
census_organic$measurement <- gsub("IN", "", census_organic$measurement, ignore.case = TRUE)
```

Change values in 'Value' column to numeric type

```
census <- census |>
  mutate(Value = as.numeric(gsub(",", "", Value)))
```

This is a function that calculates ratios for each state by dividing total acres grown by total bearing acres, then applies those ratios to the total acres grown in the same size bracket where the value is marked as NA.

```
fix_bearing_na <- function(df) {
  df_ratio <- df
  states <- unique(df$State)
  for (state in states) {
    state_data <- df |>
      filter(State == state)

    grown_value <- state_data |>
      filter(Bearing_type == "GROWN", Domain == "TOTAL", size_bracket == "TOTAL") |>
```

```
      pull(Value)

    bearing_value <- state_data |>
      filter(Bearing_type == "BEARING", Domain == "TOTAL", size_bracket == "TOTAL") |>
      pull(Value)

    if (length(grown_value) > 0 && length(bearing_value) > 0 && grown_value > 0) {
      percentage <- bearing_value / grown_value}
    else {
      next
    }

    df_ratio <- df_ratio |>
      mutate(Value = ifelse(is.na(Value) & Bearing_type == "BEARING" & State == state, percentage * Valu
  }
  return(df_ratio)
}
```

Same function but for non-bearing acres

```
fix_nonbearing_na <- function(df) {
  df_ratio <- df

  states <- unique(df$State)

  for (state in states) {
    state_data <- df |>
      filter(State == state)

    grown_value <- state_data |>
      filter(Bearing_type == "GROWN", Domain == "TOTAL", size_bracket == "TOTAL") |>
      pull(Value)

    non_bearing_value <- state_data |>
      filter(Bearing_type == "NON-BEARING", Domain == "TOTAL", size_bracket == "TOTAL") |>
      pull(Value)

    if (length(grown_value) > 0 && length(non_bearing_value) > 0 && grown_value > 0) {
      percentage <- non_bearing_value / grown_value}
    else{
      next
    }

    df_ratio <- df_ratio |>
      mutate(Value = ifelse(is.na(Value) & Bearing_type == "NON-BEARING" & State == state, percentage *
  }
  return(df_ratio)
}
```

Apply both these functions

```
census <- fix_bearing_na(census)
census <- fix_nonbearing_na(census)
```

Write CSVs

```
write.csv(survey, "survey_data.csv", row.names = FALSE)
write.csv(census, "census_data.csv", row.names = FALSE)
write.csv(chemicals, "chemicals.csv", row.names = FALSE)
write.csv(census_organic, "census_organic_data.csv", row.names = FALSE)
```

Jon Neimann helped me with the final functions