Strawberry Data Cleaning HW

Ruijian Maggie Lin

2024-10-02

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)"))
```

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")
```