Strawberries EDA

Taha Ababou

2024-10-01

Introduction

This document performs an exploratory data analysis (EDA) on the strawberries dataset. We clean the dataset, split the chemical data, and visualize key trends, such as strawberry production and pesticide use.

Step 1: Data Cleaning

We begin by loading the dataset and exploring its structure.

```
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.
```

```
glimpse(strawberry)
```

```
Rows: 12,669
Columns: 21
$ Program            <chr> "CENSUS", "CENSUS", "CENSUS", "CENSUS", "CENSUS", "~
$ Year               <dbl> 2022, 2022, 2022, 2022, 2022, 2022, 2022, 2022, 202~
```

```
$ Period             <chr> "YEAR", "YEAR", "YEAR", "YEAR", "YEAR", "YEAR", "YE~
$ `Week Ending`      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ `Geo Level`        <chr> "COUNTY", "COUNTY", "COUNTY", "COUNTY", "COUNTY", "~
$ State              <chr> "ALABAMA", "ALABAMA", "ALABAMA", "ALABAMA", "ALABAM~
$ `State ANSI`       <chr> "01", "01", "01", "01", "01", "01", "01", "01", "01~
$ `Ag District`      <chr> "BLACK BELT", "BLACK BELT", "BLACK BELT", "BLACK BE~
$ `Ag District Code` <dbl> 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40,~
$ County             <chr> "BULLOCK", "BULLOCK", "BULLOCK", "BULLOCK", "BULLOC~
$ `County ANSI`      <chr> "011", "011", "011", "011", "011", "011", "101", "1~
$ `Zip Code`         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Region             <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ watershed_code     <chr> "00000000", "00000000", "00000000", "00000000", "00~
$ Watershed          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
$ Commodity          <chr> "STRAWBERRIES", "STRAWBERRIES", "STRAWBERRIES", "ST~
$ `Data Item`        <chr> "STRAWBERRIES - ACRES BEARING", "STRAWBERRIES - ACR~
$ Domain             <chr> "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL~
$ `Domain Category`  <chr> "NOT SPECIFIED", "NOT SPECIFIED", "NOT SPECIFIED", ~
$ Value              <chr> "(D)", "3", "(D)", "1", "6", "5", "(D)", "(D)", "2"~
$ `CV (%)`           <chr> "(D)", "15.7", "(D)", "(L)", "52.7", "47.6", "(D)",~
```

```
#structure(strawberry)
```

Step 2: Remove Single-Value Columns

We now define a function to drop columns with a single unique value across all rows, as they don't add useful information to the analysis.

```
# Function to drop columns with a single unique value
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)
    }
  }

  ## 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]
  }
}

# Apply the function to drop unnecessary columns
strawberry <- drop_one_value_col(strawberry)
```

```
[1] "Columns dropped:"
[1] "Week Ending"    "Zip Code"       "Region"         "watershed_code"
[5] "Watershed"      "Commodity"
```

Step 3: Explore Data Organization

At this point, we examine whether every row is associated with a specific state.

```
# Check if every row is associated with a state
state_all <- strawberry |> distinct(State)
state_all1 <- strawberry |> group_by(State) |> count()

if(sum(state_all1$n) == dim(strawberry)[1]){print("Yes, every row in the data is associated w
```

```
[1] "Yes, every row in the data is associated with a state."
```

Step 4: Data Cleaning

Explore California Data

To gain better insight into the data, we explore the entries for California and see the differences between `CENSUS` and `SURVEY` programs.

```
# Filter California data
calif <- strawberry |> filter(State == "CALIFORNIA")

# Split data by 'Program' (CENSUS and SURVEY)
calif_census <- calif |> filter(Program == "CENSUS")
calif_survey <- calif |> filter(Program == "SURVEY")
```

```r
# Drop single-value columns in California data
drop_one_value_col(calif_census)
```

```
[1] "Columns dropped:"
[1] "Program"    "Period"     "State"      "State ANSI"
```

```r
drop_one_value_col(calif_survey)
```

```
[1] "Columns dropped:"
[1] "Program"          "Geo Level"        "State"            "State ANSI"
[5] "Ag District"      "Ag District Code" "County"           "County ANSI"
[9] "CV (%)"
```

Split the `Data Item` Column

We now clean and split composite columns for better clarity. We start with the `Data Item` column, separating it into `Fruit`, `Category`, `Item`, and `Metric`.

```r
# Split the 'Data Item' column into multiple columns
strawberry <- strawberry |>
  separate_wider_delim(cols = `Data Item`,
                       delim = ",",
                       names = c("Fruit", "Category", "Item", "Metric"),
                       too_many = "error",
                       too_few = "align_start")

# Trim white spaces from the newly created columns
strawberry$Category <- str_trim(strawberry$Category, side = "both")
strawberry$Item <- str_trim(strawberry$Item, side = "both")
strawberry$Metric <- str_trim(strawberry$Metric, side = "both")
```

Exploring Fruit Column (finding hidden sub-columns)

```r
unique(strawberry$Fruit)
```

```
 [1] "STRAWBERRIES - ACRES BEARING"
 [2] "STRAWBERRIES - ACRES GROWN"
```

```
 [3] "STRAWBERRIES - ACRES NON-BEARING"
 [4] "STRAWBERRIES - OPERATIONS WITH AREA BEARING"
 [5] "STRAWBERRIES - OPERATIONS WITH AREA GROWN"
 [6] "STRAWBERRIES - OPERATIONS WITH AREA NON-BEARING"
 [7] "STRAWBERRIES"
 [8] "STRAWBERRIES - PRICE RECEIVED"
 [9] "STRAWBERRIES - ACRES HARVESTED"
[10] "STRAWBERRIES - ACRES PLANTED"
[11] "STRAWBERRIES - PRODUCTION"
[12] "STRAWBERRIES - YIELD"
[13] "STRAWBERRIES - APPLICATIONS"
[14] "STRAWBERRIES - TREATED"
```

```r
## generate a list of rows with the production and price information

spr <- which((strawberry$Fruit=="STRAWBERRIES - PRODUCTION") | (strawberry$Fruit=="STRAWBERRI


strw_prod_price <- strawberry |> slice(spr)


## this has the census data, too

strw_chem <- strawberry |> slice(-1*spr)  ## too soon
```

Step 5: Splitting Chemical Data

In this step, our goal is to extract the chemical subtype, name, and code from the `Domain Category` column where applicable. The format of interest is:

Example:

- Input: `"CHEMICAL, FUNGICIDE: (BACILLUS SUBTILIS = 6479)"`
- Output:
  - `use` = `"FUNGICIDE"`
  - `name` = `"BACILLUS SUBTILIS"`
  - `code` = `"6479"`

After cleaning and organizing, this data should appear in three renamed columns as:

FUNGICIDE, BACILLUS SUBTILIS, 6479.

```r
# Step 1: Save the original Domain Category in another column
strawberry <- strawberry %>%
  mutate(original_domain_category = `Domain Category`)

# Step 2: Filter for rows where the Domain Category contains CHEMICAL
chemical_data <- strawberry %>%
  filter(str_detect(`Domain Category`, "CHEMICAL"))

# Step 3: Separate the Domain Category into 'type' and 'remaining'
chemical_data <- chemical_data %>%
  separate(col = `Domain Category`, into = c("type", "remaining"), sep = ", ", extra = "merg

# Step 4: Further split the 'remaining' part into 'use' and 'chemical_info'
chemical_data <- chemical_data %>%
  separate(col = remaining, into = c("use", "chemical_info"), sep = ": ", extra = "merge", fi

# Step 5: Split the 'chemical_info' into 'name' and 'code'
chemical_data <- chemical_data %>%
  separate(col = chemical_info, into = c("name", "code"), sep = " = ", extra = "merge", fill

# Step 6: Clean up the 'chemical_name' and 'chemical_code' columns
chemical_data <- chemical_data %>%
  mutate(
    name = str_remove_all(name, "\\("),  # Remove '(' before chemical name
    code = str_replace_all(code, "\\(|\\)", ""),  # Remove parentheses around chemical code
    code = as.numeric(code)  # Convert chemical code to numeric
  )

# Step 7: Select relevant columns and bring back the original Domain Category
final_chemical_data <- chemical_data %>%
  select(State, Year, original_domain_category, type, use, name, code, Value)

# Step 8: View the final cleaned chemical data
glimpse(final_chemical_data)
```

```
Rows: 3,359
Columns: 8
$ State                   <chr> "CALIFORNIA", "CALIFORNIA", "CALIFORNIA", "CA~
$ Year                    <dbl> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 202~
$ original_domain_category <chr> "CHEMICAL, FUNGICIDE: (OXATHIAPIPROLIN = 1281~
$ type                    <chr> "CHEMICAL", "CHEMICAL", "CHEMICAL", "CHEMICAL~
$ use                     <chr> "FUNGICIDE", "INSECTICIDE", "INSECTICIDE", "O~
```

```
$ name                    <chr> "OXATHIAPIPROLIN", "CYCLANILIPROLE", "PERMETH~
$ code                    <dbl> 128111, 26202, 109701, 115003, 128111, 26202,~
$ Value                   <chr> "(D)", "(D)", "(D)", "(NA)", "(D)", "(D)", "(~
```

```
# Saving the cleaned chemical data for future analysis
write_csv(final_chemical_data, "cleaned_chemical_data.csv")
```