Strawberries1

Chloe Singer

2024-10-09

```r
## Step 1: Import data set ##

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.
```

```r
## Step 2: Check and make sure each row is associated with a specific state. ##

#| label: explore organization 1


state_all <- strawberry |> distinct(State)

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

# Each row is associated with a specific state #

if(sum(state_all1$n) == dim(strawberry)[1]){print("Yes, each row in the data set is associated with a sp
```

```
## [1] "Yes, each row in the data set is associated with a specific state."
```

```r
## Step 3: Because the dataset contains some columns filled entirely with missing values (NA), I can cre

#|label: function def - drop 1-item columns

drop_one_value_col <- function(df){   # Takes the entire dataframe #
drop <- NULL

# Check each column to see if it contains only one unique value. #

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{

   print("Columns dropped:")
   print(colnames(df)[drop])
   strawberry <- df[, -1*drop]
   }
}

# Now I will use the function: #

strawberry <- drop_one_value_col(strawberry)
```

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

```
drop_one_value_col(strawberry)
```

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

```
## Step 4: By examining the data, I noticed that the 'Geo Level' column at the county level doesn't pro

#| label: ditch the counties

unique(strawberry$`Geo Level`)
```

```
## [1] "COUNTY"   "NATIONAL" "STATE"
```

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

Now I will examine the rest of the columns

```
## Step 5: The 'Program' column divides the data into two primary categories: CENSUS data, which focuse

#|label: split srawberry into census and survey pieces

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

```
## [1] "Columns dropped:"
## [1] "Program"          "Period"           "Ag District"      "Ag District Code"
## [5] "County"           "County ANSI"
```

```r
straw_sur <- strawberry |> filter(Program == "SURVEY")
straw_sur <- straw_sur %>%  drop_one_value_col()
```

```
## [1] "Columns dropped:"
## [1] "Program"          "Ag District"      "Ag District Code" "County"
## [5] "County ANSI"      "CV (%)"
```

```r
nrow(strawberry) == (nrow(straw_sur) + nrow(straw_cen))
```

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

```r
## Step 6: I will start by exploring the survey data. I observe that the 'Domain Category' column holds

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

```
## Warning: Expected 2 pieces. Additional pieces discarded in 35 rows [126, 219, 310, 401,
## 494, 631, 703, 772, 841, 911, 1139, 1249, 1357, 1465, 1575, 2100, 2222, 2342,
## 2462, 2584, ...].

## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 606 rows [1, 2, 3, 4, 5,
## 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].

## Warning: There was 1 warning in `mutate()`.
## i In argument: `chemical_num = as.numeric(trimws(chemical_num))`.
## Caused by warning:
## ! NAs introduced by coercion
```

```r
## Step 7: This looks a little bit better, but the "Domain" column still holds multiple pieces of infor

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

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 606 rows [1, 2, 3, 4, 5,
## 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```r
## Step 8: With "Domain Category" sorted, I can now turn to another information-heavy column, "Data Ite

straw_sur <- straw_sur %>%
  # I extract the measurement portion first and remove it from Data Item #
  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
## Step 9: I noticed that the measurement column was moved to the end of the data frame after using the

straw_sur <- straw_sur %>%
  select(1:7, 13, 8:12)
```

```
## Step 10: I also noticed that the measurement column includes some unnecessary words, such as "IN." I

straw_sur <- straw_sur %>%
  mutate(measurement = gsub("\\bIN\\b", "", measurement)) %>%
  mutate(measurement = trimws(measurement))
```

```
## Step 11: I noticed that the value column is currently at the end of the data frame. It would be more

straw_sur <- straw_sur %>%
  select(1:8, 13, 9:12)
```

```
## Step 12: My data item column now contains only one value: "strawberries." However, using the unique

straw_sur <- straw_sur %>%
  select(-`Data Item`)
```

```
## Step 13: While reviewing the data, I noticed that some values in the "Category" column have unnecess

straw_sur$category <- gsub(",", "", straw_sur$category)
```

```
## Step 14: Finally, I will split the data into two tables based on the values in the domain column-one

sur_total <- straw_sur %>% filter(Domain == "TOTAL")
sur_chem <- straw_sur %>% filter(Domain == "CHEMICAL")
sur_total = drop_one_value_col(sur_total)
```

```
## [1] "Columns dropped:"
## [1] "Domain"        "use"           "chemical_name" "chemical_num"
```

```
sur_chem = drop_one_value_col(sur_chem)
```

```
## [1] "Columns dropped:"
## [1] "Period"    "Geo Level" "Domain"
```

```
## Step 15: Next, I'm going to shift my focus to the Census data. Here, I again notice that the Data It

#| label: straw_cen split cols

straw_cen <- straw_cen |>
  separate_wider_delim(  cols = `Data Item`,
                         delim = " - ",
                         names = c("strawberries",
                                   "Category"),
                         too_many = "error",
                         too_few = "align_start"
                       )
```

```
## Step 16: Next, I want to create a separate table specifically for organic strawberry sales. To do th

#| label: isolate organic


straw_cen <- straw_cen |>
  separate_wider_delim(  cols = strawberries,
                         delim = ", ",
                         names = c("strawberries",
                                 "ORGANIC",
                                 "organic_detail"),

                         too_many = "error",
                         too_few = "align_start"
                       )


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

```
## [1] "Columns dropped:"
## [1] "strawberries"
```

```
# How many organic rows are there? #

organic_cen <- straw_cen |> filter(ORGANIC == "ORGANIC")

sum(is.na(straw_cen$ORGANIC))
```

```
## [1] 662
```

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


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

```
## [1] "Columns dropped:"
## [1] "Year"           "ORGANIC"        "organic_detail"
```

```
## Step 17: Now, I can split the Category column in the straw_cen data frame into two new columns: meas

#| label: explore straw_cen$Category

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

```r
straw_cen$COL2 <- str_replace(straw_cen$COL2,"WITH ","")

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

```r
## Step 18: Next, I noticed that the Domain Category column, similar to the one in the straw_sur data fi

#| label: explore straw_cen$Domain & Domain Category


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

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

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

```r
## Step 19: Moving on to the organic census data, I start by removing all the columns that contain only

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

```
## [1] "Columns dropped:"
## [1] "ORGANIC"         "Domain"          "Domain Category"
```

```r
## Step 20: Next, I will clean the Category column, just like I did in the straw_sur data. I will split

organic_cen <- organic_cen %>%
  separate(Category, into = c("Category", "measurement"), sep = " MEASURED ", extra = "merge", fill = "r
```

```r
## Step 21: I now need to get rid of the occurrences of commas and "IN" from the new column. ##

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

```r
## Step 22: I now need to get rid of the occurrences of (D) and (Z) in the value column and replace them

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

```r
## Step 23: I now have four cleaned tables: sur_chem containing chemical data, sur_total with the total

straw_cen <- straw_cen %>%
  mutate(Value = as.numeric(gsub(",", "", Value)))
```

```r
## Step 24: This process took some time, and I used ChatGPT to help with parts of it. However, I success

fill_na_bearing_values <- function(df) {
  # Creating a copy of the data frame so I do not have to edit the original version #
  df_filled <- df

  states <- unique(df$State)
  for (state in states) {
    # Filter the data for the current state #
    state_data <- df %>% filter(State == state)

    # Taking out the GROWN and BEARING values #
    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)

    # Now calculating the total percentage of bearing strawberries #
    if (length(grown_value) > 0 && length(bearing_value) > 0 && grown_value > 0) {
      percentage <- bearing_value / grown_value
    } else {
      next
    }

    # Now filling in NA values for the BEARING rows where the size_bracket is the same #
    df_filled <- df_filled %>%
      mutate(Value = ifelse(is.na(Value) & Bearing_type == "BEARING" & State == state,
                            percentage * Value[which(Bearing_type == "GROWN" &
                                                     size_bracket == size_bracket &
                                                     State == state)],
                            Value))
  }

  return(df_filled)
}
```

```r
## Step 25: And now I will now the same thing for the non-bearing strawberries ##

fill_na_non_bearing_values <- function(df) {
  # Creating a copy of the data frame so I do not have to edit the original version ##
  df_filled <- df

  states <- unique(df$State)

  for (state in states) {
    # Filter the data for the current state #
    state_data <- df %>% filter(State == state)

    # Taking out the GROWN and NON-BEARING values #
    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)
    # Now calculating the total percentage of non-bearing strawberries #
    if (length(grown_value) > 0 && length(non_bearing_value) > 0 && grown_value > 0) {
      percentage <- non_bearing_value / grown_value
    } else {
      next
    }

    # Now filling in NA values for the NONBEARING rows where the size_bracket is the same #
    df_filled <- df_filled %>%
      mutate(Value = ifelse(is.na(Value) & Bearing_type == "NON-BEARING" & State == state,
                            percentage * Value[which(Bearing_type == "GROWN" &
                                                       size_bracket == size_bracket &
                                                       State == state)],
                            Value))
  }

  return(df_filled)
}
```

```r
## Step 26: Finally, I will apply both functions to the straw_cen data to fill in some of the NA values

straw_cen = fill_na_bearing_values(straw_cen)
straw_cen = fill_na_non_bearing_values(straw_cen)
```

```r
## Step 27: Write the data sets into csv files. ##

write.csv(sur_chem, "cleaned_survey_chemical_data.csv")
write.csv(sur_total, "cleaned_survey_total_data.csv")

write.csv(straw_cen, "cleaned_census_nonorganic_data.csv")
write.csv(organic_cen, "cleaned_census_organic_data.csv")
```