StrawberryAnalysis

Chang Lu

2024-10-05

Strawberries: Data

This is a project about acquiring strawberry data from the USDA-NASS system and several tasks involving data cleaning, missing data analysis and imputation, data organization, and exploratory data analysis.

Strawberry data cleaning

```
# --- Data Cleaning ---
# Load the strawberry dataset and take a glimpse at the data
strawberry_data <- 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_data)
```

```
## 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)",~
```

Explanation

Dataset Overview: The `strawberry_data` contains 12,699 rows and 21 columns. Columns include information like date, location, values, and coefficients of variation.

Remove Columns with Single Value in All Rows

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

# Function to remove columns with only a single unique value
drop_one_value_col <- function(df) {
  drop <- NULL

  # Test each column for a single value
  for (i in 1:ncol(df)) {
    if (n_distinct(df[[i]]) == 1) {
      drop <- c(drop, i)
    }
  }

  # Report the result - names of columns dropped
  if (is.null(drop)) {
    return(df)
  } else {
    print("Columns dropped:")
    print(colnames(df)[drop])
    df <- df[, -drop]
  }
}

# Use the function to remove single-value columns
strawberry_data <- drop_one_value_col(strawberry_data)
```

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

```
glimpse(strawberry_data)
```

```
## Rows: 12,669
## Columns: 15
```

```
## $ 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~
## $ `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~
## $ `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)",~
```

Separate Composite Columns

```
#|label: split Data Item

# Split 'Data Item' column into two parts using delimiter '-'
strawberry_data <- strawberry_data |>
  separate_wider_delim(  cols = `Data Item`,
                         delim = "-",
                         names = c("column1", "column2"),
                         too_many = "merge",
                         too_few = "align_start")

# Further split 'column1' into 'Fruit' and 'Category' using delimiter ','
strawberry_data <- strawberry_data |>
  separate_wider_delim(  cols = `column1`,
                         delim = ",",
                         names = c("Fruit", "Category"),
                         too_many = "merge",
                         too_few = "align_start")

# Trim white spaces from the split columns
strawberry_data$Fruit <- str_trim(strawberry_data$Fruit, side = "both")
strawberry_data$Category <- str_trim(strawberry_data$Category, side = "both")
strawberry_data$column2 <- str_trim(strawberry_data$column2, side = "both")

# Remove columns with single-value again after modifications
strawberry_data <- drop_one_value_col(strawberry_data)
```

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

```
unique(strawberry_data$Category)
```

```
##  [1] NA                       "ORGANIC"                "ORGANIC, FRESH MARKET"
##  [4] "ORGANIC, PROCESSING"    "FRESH MARKET"           "PROCESSING"
```

```
##  [7] "FRESH MARKET, UTILIZED" "NOT SOLD"               "PROCESSING, UTILIZED"
## [10] "UTILIZED"               "BEARING"
```

Clean Data in the 'Category' Column

```
#|label: Clean data in the Category

# Extract specific attributes from 'Category' and assign them to new columns
strawberry_data <- strawberry_data %>%
  mutate(Marketing_channels = ifelse(str_detect(Category, "FRESH MARKET|PROCESSING"),
                                     str_extract(Category, "FRESH MARKET|PROCESSING"), NA)) %>%
  mutate(Utilizations = ifelse(str_detect(Category, "UTILIZED"),
                               str_extract(Category, "UTILIZED"), NA)) %>%
  mutate(Method = ifelse(str_detect(Category, "ORGANIC"),
                         str_extract(Category, "ORGANIC"), NA)) %>%
  mutate(Class = ifelse(str_detect(Category, "BEARING"),
                        str_extract(Category, "BEARING"), NA)) %>%
  mutate(Measurement = ifelse(str_detect(Category, "NOT SOLD"),
                              str_extract(Category, "NOT SOLD"), NA)) %>%
  select(1:match("County ANSI", names(.)), Class, Method, Marketing_channels, Utilizations, Measurement
  select(-Category)
```

Clean Data in 'column2'

```
#|label: Clean column2

# Split 'column2' into 'Measurement1' and 'Metric1'
strawberry_data <- strawberry_data %>%
  mutate(
    Measurement1 = ifelse(str_detect(column2, ","),
                          str_split_fixed(column2, ",", 2)[, 1],
                          column2),
    Metric1 = ifelse(str_detect(column2, ","),
                     str_split_fixed(column2, ",", 2)[, 2],
                     NA)
  ) %>%
  select(1:match("Measurement", names(.)), Measurement1, Metric1, everything()) %>%
  select(-column2)
```

Classify Strings in Their Positions

```
#|label: classify strings

# Extract specific strings from 'Metric1' to 'Metric' and 'Remark'
strawberry_data <- strawberry_data %>%
  mutate(Metric = str_extract(Metric1, "(?<=,|^)[^,]*MEASURED IN[^,]*(?=,|$)")) %>%
  mutate(Remark = str_remove_all(Metric1, "(?<=,|^)[^,]*MEASURED IN[^,]*(,)?")) %>%
  mutate(Remark = str_trim(str_replace_all(Remark, "^,|,$|,,", ""))) %>%
```

```
  select(1:match("Measurement1", names(.)), Metric, Remark, everything())  %>%
  select(-Metric1)

# Create 'Category' column by combining 'Measurement' and 'Measurement1'
strawberry_data <- strawberry_data %>%
  mutate(Category = ifelse(is.na(Measurement),
                           Measurement1,
                           paste(Measurement, Measurement1, sep = " "))) %>%
  select(1:match("Utilizations", names(.)), Category, everything()) %>%
  select(-Measurement, -Measurement1)

# Trim whitespace
strawberry_data$Metric <- str_trim(strawberry_data$Metric, side = "both")
strawberry_data$Category <- str_trim(strawberry_data$Category, side = "both")

# Replace empty strings in 'Remark' with NA
strawberry_data <- strawberry_data %>% mutate(across(Remark, ~ na_if(., "")))
glimpse(strawberry_data)
```

```
## 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~
## $ `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~
## $ Class              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Method             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Marketing_channels <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Utilizations       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Category           <chr> "ACRES BEARING", "ACRES GROWN", "ACRES NON-BEARING"~
## $ Metric             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Remark             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ 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)",~
```

Separate Domain and Domain Category

```
#|label: separate domain category

# Clean up 'Domain Category' by removing redundant text and characters
strawberry_data <- strawberry_data %>%
  mutate(`Domain Category` = ifelse(str_detect(`Domain Category`, ":"),
                                    str_split_fixed(`Domain Category`, ":", 2)[, 2],
```

```
                              `Domain Category`)) %>%
  mutate(`Domain Category` = str_replace_all(`Domain Category`, "[\\(\\)]", ""))

strawberry_data$`Domain Category` <- str_trim(strawberry_data$`Domain Category`, side = "both")

# Separate 'Domain Category' into specific chemical and numbers
strawberry_data <- strawberry_data %>%
  mutate(Chemical_Number = ifelse(str_detect(`Domain Category`, "="),
                                  str_split_fixed(`Domain Category`, "=", 2)[, 2],
                                  NA)) %>%
  mutate(`Domain Category` = ifelse(str_detect(`Domain Category`, "="),
                                str_split_fixed(`Domain Category`, "=", 2)[, 1],
                                `Domain Category`)) %>%
  select(1:match("Domain Category", names(.)), Chemical_Number, everything())

# Trim whitespace
strawberry_data$`Domain Category` <- str_trim(strawberry_data$`Domain Category`, side = "both")
strawberry_data$Chemical_Number <- str_trim(strawberry_data$Chemical_Number, side = "both")

# Remove 'CHEMICAL' from the 'Domain' column
strawberry_data <- strawberry_data %>%
  mutate(Domain = str_replace(Domain, "CHEMICAL, ", ""))
glimpse(strawberry_data)
```

```
## Rows: 12,669
## Columns: 22
## $ 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~
## $ `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~
## $ Class              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Method             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Marketing_channels <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Utilizations       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Category           <chr> "ACRES BEARING", "ACRES GROWN", "ACRES NON-BEARING"~
## $ Metric             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Remark             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Domain             <chr> "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL~
## $ `Domain Category`  <chr> "NOT SPECIFIED", "NOT SPECIFIED", "NOT SPECIFIED", ~
## $ Chemical_Number    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Value              <chr> "(D)", "3", "(D)", "1", "6", "5", "(D)", "(D)", "2"~
## $ `CV (%)`           <chr> "(D)", "15.7", "(D)", "(L)", "52.7", "47.6", "(D)",~
```

Transfer Data Types to Correct Types

```
#|label: transfer data types

# Replace specific strings in 'Value' and 'CV (%)' columns and convert to numeric
strawberry_data <- strawberry_data %>%
  mutate(Value = ifelse(Value %in% c("(D)", "(NA)"), NA, Value)) %>%
  mutate(Value = ifelse(Value == "(Z)", "0.0005", Value)) %>%
  mutate(Value = str_replace_all(Value,",","")) %>%
  mutate(Value = as.numeric(Value))

strawberry_data <- strawberry_data %>%
  mutate(`CV (%)` = ifelse(`CV (%)` %in% c("(D)", "(NA)"), NA, `CV (%)`)) %>%
  mutate(`CV (%)` = ifelse(`CV (%)` == "(L)", "0.05", `CV (%)`)) %>%
  mutate(`CV (%)` = ifelse(`CV (%)` == "(H)", "99.95", `CV (%)`)) %>%
  mutate(`CV (%)` = str_replace_all(`CV (%)`,",","")) %>%
  mutate(`CV (%)` = as.numeric(`CV (%)`))
glimpse(strawberry_data)
```

```
## Rows: 12,669
## Columns: 22
## $ 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~
## $ `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~
## $ Class              <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Method             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Marketing_channels <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Utilizations       <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Category           <chr> "ACRES BEARING", "ACRES GROWN", "ACRES NON-BEARING"~
## $ Metric             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Remark             <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Domain             <chr> "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL", "TOTAL~
## $ `Domain Category`  <chr> "NOT SPECIFIED", "NOT SPECIFIED", "NOT SPECIFIED", ~
## $ Chemical_Number    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ Value              <dbl> NA, 3, NA, 1, 6, 5, NA, NA, 2, 2, NA, NA, 2, 2, 1, ~
## $ `CV (%)`           <dbl> NA, 15.70, NA, 0.05, 52.70, 47.60, NA, NA, 55.70, 5~
```

Separate the Data into Categories

```
#|label: separate data categories

# Split the table into 'census' and 'survey'
strawberry_census <- strawberry_data %>% filter(Program == "CENSUS")
```

```
strawberry_survey <- strawberry_data %>% filter(Program == "SURVEY")
strawberry_census <- drop_one_value_col(strawberry_census)
```

```
## [1] "Columns dropped:"
## [1] "Program"         "Period"          "Class"           "Utilizations"
## [5] "Remark"          "Chemical_Number"
```

```
strawberry_survey <- drop_one_value_col(strawberry_survey)
```

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

```
# Split the 'census' table into 'organic' and 'non-organic'
strawberry_organic <- strawberry_census %>% filter(Domain == "ORGANIC STATUS")
strawberry_organic <- drop_one_value_col(strawberry_organic)
```

```
## [1] "Columns dropped:"
## [1] "Ag District"      "Ag District Code" "County"           "County ANSI"
## [5] "Method"           "Domain"           "Domain Category"
```

```
strawberry_non_organic <- strawberry_census %>% filter(!Domain == "ORGANIC STATUS")
strawberry_non_organic <- drop_one_value_col(strawberry_non_organic)
```

```
## [1] "Columns dropped:"
## [1] "Year"               "Method"             "Marketing_channels"
## [4] "Metric"
```

```
# Split the 'survey' table into 'chemical' and 'non-chemical'
strawberry_chemical <- strawberry_survey %>% filter(!Domain == "TOTAL")
strawberry_chemical <- drop_one_value_col(strawberry_chemical)
```

```
## [1] "Columns dropped:"
## [1] "Period"             "Geo Level"          "Marketing_channels"
## [4] "Utilizations"
```

```
strawberry_non_chemical <- strawberry_survey %>% filter(Domain == "TOTAL")
strawberry_non_chemical <- drop_one_value_col(strawberry_non_chemical)
```

```
## [1] "Columns dropped:"
## [1] "Class"           "Domain"          "Domain Category" "Chemical_Number"
```

Missing Data Analysis and Imputation

Estimate the NA in Value and CV(%)

```r
#|label: estimate missing values

# Function to estimate missing values in a given dataset using linear regression
estimate_na_values <- function(df, value_model_formula, cv_model_formula = NULL) {
  df <- df %>% mutate(original_order = row_number())
  # Value estimation
  with_value <- df %>% filter(!is.na(Value))
  missing_value <- df %>% filter(is.na(Value))
  value_model <- lm(value_model_formula, data = with_value)
  missing_value <- missing_value %>%
    mutate(Value = round(exp(predict(value_model, newdata = missing_value)), 1))
  df_filled <- bind_rows(with_value, missing_value) %>% arrange(original_order)
  # CV estimation
  if (!is.null(cv_model_formula)) {
    with_cv <- df_filled %>% filter(!is.na(`CV (%)`))
    missing_cv <- df_filled %>% filter(is.na(`CV (%)`))
    cv_model <- lm(cv_model_formula, data = with_cv)
    missing_cv <- missing_cv %>%
      mutate(`CV (%)` = round(predict(cv_model, newdata = missing_cv), 2))
    df_filled <- bind_rows(with_cv, missing_cv) %>% arrange(original_order)
  }
  return(df_filled %>% select(-original_order) %>% mutate(Value = round(Value, 1)))
}

# Estimate missing values for 'strawberry_organic'
strawberry_organic <- estimate_na_values(strawberry_organic, log(Value) ~ factor(Year) + State + Categor

# Estimate missing values for 'strawberry_non_organic'
strawberry_non_organic <- estimate_na_values(strawberry_non_organic, Value ~ State + Category + Domain +

# Estimate missing values for 'strawberry_chemical'
strawberry_chemical <- estimate_na_values(strawberry_chemical, log(Value) ~ factor(Year) + State + Categ

# Estimate missing values for 'strawberry_non_chemical'
strawberry_non_chemical <- estimate_na_values(strawberry_non_chemical, log(Value + 1) ~ factor(Year) + S
```

Data Organization

Write Cleaned Data to CSV Files

```r
#|label: write csv

# Write cleaned datasets to new CSV files
write.csv(strawberry_organic, "strawberry_organic.csv", row.names = FALSE)
write.csv(strawberry_non_organic, "strawberry_non_organic.csv", row.names = FALSE)
write.csv(strawberry_chemical, "strawberry_chemical.csv", row.names = FALSE)
write.csv(strawberry_non_chemical, "strawberry_non_chemical.csv", row.names = FALSE)
```

Exporatory Data Analysis

```
strawberry <- read_csv("strawberries25_v3.csv", col_names = TRUE, show_col_types = FALSE )

## glimpse(strawberry)
```

The data was originally collected at the county, state, and national levels, but the degree of missingness at the state level was too high, so I dropped the county-level data.

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

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

There are 5,359 rows and 21 column in the initial data set. The only complete year is 2022, although there is data for years 2018 through 2024.

To work with the data, define a function to remove columns with only single value in all its rows.

```
drop_one_value_col <- function(df, prt_val = FALSE){
 # browser()
  df_id <- ensym(df)
  if(prt_val){
  msg = paste("Looking for single value columns in data frame: ",as.character(df_id) )
  print(msg)}
  ## takes whole dataframe
dropc <- NULL
val <- NULL
## test each column for a single value
for(i in 1:dim(df)[2]){
  if(dim(distinct(df[,i]))[1] == 1){
    dropc <- c(dropc, i)
    val <- c(val, df[1,i])
  }
}

if(prt_val){
if(is.null(dropc)){
  print("No columns dropped")
  return(df)}else{
   print("Columns dropped:")
   # print(colnames(df)[drop])
   print(unlist(val))
   df <- df[, -1*dropc]
   return(df)
  }
}
 df <- df[, -1*dropc]
   return(df)
}


## use the function
```

```
strawberry <- strawberry |> drop_one_value_col(prt_val = FALSE)
```

To work with this data, split the Census data from the Survey data.

Census data cleaning and organizing

we're examining census data because it's different from survey data

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

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

straw_cen <- straw_cen |> drop_one_value_col()

straw_sur <- straw_sur |> drop_one_value_col()

rm(strawberry)
```

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

we're isolating organic data

```
# unique(straw_cen$strawberries)
# straw_cen$strawberries |> str_which("STRAWBERRIES") |> length()
# straw_cen$strawberries |> str_which("STRAWBERRIES, ORGANIC") |> length()
# straw_cen$strawberries |> str_which("STRAWBERRIES, ORGANIC, FRESH MARKET") |> length()
# straw_cen$strawberries |> str_which("STRAWBERRIES, ORGANIC, PROCESSING" ) |> length()
#
# ## count the cases
# straw_cen$strawberries |> str_which("ORGANIC") |> length()
# straw_cen$strawberries |> str_which("FRESH MARKET") |> length()
# straw_cen$strawberries |> str_which("PROCESSING") |> length()

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

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

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

## how many organic rows?

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

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

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


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

Note that straw_cen has only one year: 2022

Current stats Census date has been isolated and split between Organic and Conventional strawberries

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



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

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

```r
## remove AREA GROWN and parens
## change NOT SPECIFIEC TO TOTAL

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

imputed values

```r
#Casting value to numeric and replacing commas with blanks first

straw_cen$Value <- as.numeric(str_replace(straw_cen$Value, ",", ""))

organic_cen$Value <- as.numeric(str_replace(organic_cen$Value, ",", ""))
```

```r
#Looking at what's missing in our strawberry_census dataframes

aa <- dim(straw_cen |> filter(is.na(Value)))
bb <- dim(organic_cen |> filter(is.na(Value)))
```

We'll start with missing value interpolation for census data:

```r
state_vec <- straw_cen |> distinct(State)

#Let's write a function that does the percentage imputation

#First retrieve the indices of the rows with NA Values:
bearing_na_indices <- which(is.na(straw_cen$Value) &
                              straw_cen$Bearing_type == "BEARING")
non_bearing_na_indices <- which(is.na(straw_cen$Value) &
                                  straw_cen$Bearing_type == "NON-BEARING")

#We will also need indices of the rows with NA Values that have bearing 'Grown'

grown_na_indices <- which(is.na(straw_cen$Value) &
                            straw_cen$Bearing_type == "GROWN")

rm(state_vec)
```

We will first interpolate the Bearing "Grown" values. Then we can interpolate Bearing and Non-bearing from those values as we discussed in lecture.

```r
bearing_non_bearing_prop_avg <- function(state, measure, bearing) {
  state_cen <- straw_cen |> filter(State == state & Measure == measure)
  bearing_vec <- state_cen |> filter(Bearing_type == bearing)
  bearing_vec <- bearing_vec$Value
  bearing_indices <- which(!is.na(bearing_vec))

  grown_vec <- state_cen |> filter(Bearing_type == "GROWN")
  grown_vec <- grown_vec[bearing_indices, ]$Value

  bearing_no_na <- which(!is.na(bearing_vec))
  grown_no_na <- which(!is.na(grown_vec))

  not_na_indices <- intersect(bearing_no_na, grown_no_na)

  bearing_vec_noNA <- bearing_vec[not_na_indices]
  grown_vec_noNA <- grown_vec[not_na_indices]

  return(mean(bearing_vec_noNA / grown_vec_noNA))
}
```

Now if GROWN is NA, there are three possibilities:

1 Bearing and Non-bearing are both NA
2 Neither Bearing nor Non-bearing are NA
3 One of Bearing nor Non-bearing are NA

The solution for each of these cases is different:

1 Interpolate from the total for area grown(a check of the dataframe shows you that the total is never NA). We'll try and calculate the average ratio of the missing parts across the non-missing data and split the difference from the total across those
2 Sum Bearing and non-bearing. Easy.

3 Calculate the average proportion of grown that bearing or non bearing(as the case may be) is for the state in question. Then we use that to figure out the value.

Applying our functions now to strawberry census:

```
grown_imputed <- sapply(grown_na_indices, census_grown_imputer)
straw_cen$Value[grown_na_indices] <- grown_imputed
```

Now that we have imputed the values, we still have an adjustment to make. Because we are using proportional averages, the values might no longer add up to our totals. So we need to write a function that will scale our imputed values so that the totals add up.

The happy news is because we are doing this by index, we still know which are the values we have imputed and therefore which values we need to scale.

```
state_index_range_retriever <- function(state, bearing, measure) {
  return(
    which(
      straw_cen$State == state &
        straw_cen$Measure == measure & straw_cen$Bearing_type == bearing
    )
  )
}
#We're going to
imputed_val_scaler <- function(index_vec, bearing) {
  scaled_imputed_vals <- rep(0, length(index_vec))
  #First, we group the indices which have the same state, bearing
  #Keep track of indices we've already grouped
  index_tracker <- c()
  for (i in index_vec) {
    if (!(i %in% index_tracker)) {
      grouped_indices <- index_vec[which(straw_cen$State[index_vec] == straw_cen$State[i])]
      #Then we scale those indices to the total:
      state <- straw_cen$State[i]
      measure <- straw_cen$Measure[i]
      state_total_cen <- straw_cen |> filter(State == state &
                                               Measure == measure & Bearing_type == bearing)

      #This is so we can sum over non-totals
      state_total_cen_not_total <- state_total_cen |> filter(Domain != "TOTAL")

      #Extract the total we want everything to sum to
      overall_total <- state_total_cen |> filter(Domain == "TOTAL")
      overall_total <- overall_total$Value

      #What is the correct sum the imputed values should add to?
      grouped_indices_adj <- grouped_indices - which(
        straw_cen$State == state &
          straw_cen$Measure == measure & straw_cen$Bearing_type == bearing
      )[1] + 1
```

```
      correct_total <- overall_total - sum(state_total_cen_not_total$Value[-grouped_indices_adj])
      incorrect_total <- sum(state_total_cen_not_total$Value[grouped_indices_adj])
      #Now we scale these:
      scaled_vals <- (correct_total / incorrect_total) * state_total_cen_not_total$Value[grouped_indices
      scaled_imputed_vals[which(index_vec %in% grouped_indices)] <- scaled_vals
      index_tracker <- c(grouped_indices, index_tracker)
    }
  }

  return(scaled_imputed_vals)

}

grown_imputed_scaled <- imputed_val_scaler(grown_na_indices, bearing = "GROWN")
straw_cen$Value[grown_na_indices] <- grown_imputed_scaled
```

Imputing acres bearing and non-bearing

```
bearing_non_bearing_prop_avg<-function(i){
  state<-as.character(straw_cen[i,"State"])
  measure<-as.character(straw_cen[i,"Measure"])
  bearing<-as.character(straw_cen[i,"Bearing_type"])

  #Filter and retrieve the
  state_cen<-straw_cen |> filter(State==state& Measure==measure)
  bearing_vec<-state_cen |> filter(Bearing_type==bearing)
  bearing_vec<-bearing_vec$Value
  bearing_indices<-which(!is.na(bearing_vec))

  grown_vec<-state_cen |> filter(Bearing_type==bearing)
  grown_vec<-grown_vec[bearing_indices,]$Value

  bearing_no_na<-which(!is.na(bearing_vec))
  grown_no_na<-which(!is.na(grown_vec))

  not_na_indices<-intersect(bearing_no_na,grown_no_na)

  bearing_vec_noNA<-bearing_vec[not_na_indices]
  grown_vec_noNA<-grown_vec[not_na_indices]

  return(mean(bearing_vec_noNA/grown_vec_noNA))
}
```

```
bearing<-"BEARING"
bearing_imputed <-sapply(bearing_na_indices,bearing_non_bearing_prop_avg)
straw_cen$Value[bearing_na_indices]<-bearing_imputed


bearing<-"NON-BEARING"
non_bearing_imputed <-sapply(non_bearing_na_indices,bearing_non_bearing_prop_avg)
straw_cen$Value[non_bearing_na_indices]<-non_bearing_imputed
```

We can go one better with imputation where we impute case 1 for the "GROWN" categories together with other case 1s for each. A slight modification to the function above plus some other changes achieves this.

Survey data cleaning and organizing

```
## Data Item

## unique(straw_sur$`Data Item`)

straw_sur1 <- straw_sur |>  separate_wider_delim(cols = `Data Item`,
                                                 delim = ", ",
                                                 names = c("straw",
                                                           "mkt",
                                                           "measure",
                                                           "other"
                                                           ),
                                                 too_many = "merge",
                                                 too_few = "align_start")


straw_sur2 <- straw_sur1 |> separate_wider_delim(cols = "straw",
                                                  delim = " - ",
                                                  names = c("straw",
                                                            "more"),
                                                  too_many = "merge",
                                                  too_few = "align_start"
                                                  )


rm(straw_sur, straw_sur1)
```

Shift data into alignment function

```
## function shift_loc
## Moves adjacent data cells in a data.frame on a single row
## Use this function to fix alignment problems after separating
## columns containing multiple columns of data.

## Of course the working assumption is that there is room in the
## data frame for the data you're shifting.
##
## The data cells that are empty after the data shift are NA.
##
## Input paramaters
##
## df -- data frame
## col_name -- name of colume where the left-most data item is located
## dat_name -- name of data item in the column
## num_col -- the number of columns is the same as the number of
##            adjacent data to be moved.
## num_shift -- the number of rows to move the data
##
```

```
shift_loc <- function(df, col_name, dat_name, num_col, num_shift){
 # browser()
  col_num = which(colnames(df) == col_name)
  row_num = which(df[,col_num] == dat_name)  ## calcs a vector of rows

  for(k in 1:length(row_num)){
  d = rep(0,num_col) ## storage for items to be moved
  for(i in 1:num_col){
    d[i] = df[row_num[k], col_num + i - 1]
  }
  for(i in 1:num_col){
    ra = row_num[k]
    cb = col_num + i - 1
    df[ra, cb] <-  NA
  }
  for(j in 1:num_col){
    rc = row_num[k]
    cd = col_num + j - 1 + num_shift
    df[rc, cd] = d[j]
  }
  }
 # sprintf("Rows adjusted:")
  # print("%d",row_num)
  return(df)
}
```

```
straw_sur2 %<>% shift_loc("more", "PRICE RECEIVED", 2, 1 )

straw_sur2 %<>% shift_loc("more", "ACRES HARVESTED", 1, 1 )

straw_sur2 %<>% shift_loc("more", "ACRES PLANTED", 1, 1 )

straw_sur2 %<>% shift_loc("more", "PRODUCTION", 2, 1 )

straw_sur2 %<>% shift_loc("more", "YIELD", 2, 1 )

straw_sur2 %<>% shift_loc("more", "APPLICATIONS", 3, 1 )

straw_sur2 %<>% shift_loc("more", "TREATED", 3, 1 )

straw_sur2 %<>% drop_one_value_col()
```

Examine Domain

```
# unique(straw_sur2$Domain)
#    The Domain column (2965 rows) contains data about
#    Chemicals (3359 rows)
#    Fertilizers (115 rows)
#    Production and Yield data ("TOTAL") (491 rows)
#

#    The Chemical data is in categories for
```

```
#         Insecticides
#         Fungicides
#         Herbicdes, and
#         Other
#

#    The Domain is split into three dataframes for
#     "Total", "Chemical", and "Fertilizer"


straw_sur2 <- straw_sur2 |>
  separate_wider_delim(cols = Domain,
                       delim = ", ",
                       names = c("col1",
                                 "col2"),

                       too_many = "merge",
                       too_few = "align_start")


# unique(straw_sur2$col1)

survey_d_total <- straw_sur2 |>  filter(col1 == "TOTAL")

survey_d_chem <- straw_sur2 |>  filter(col1 == "CHEMICAL")

survey_d_fert <- straw_sur2 |>  filter(col1 == "FERTILIZER")
```

now look at totals

```
survey_d_total %<>% drop_one_value_col()

### align terms

survey_d_total %<>% shift_loc("measure", "MEASURED IN $ / CWT", 1, 1 )


survey_d_total %<>% shift_loc("measure", "MEASURED IN $", 1, 1 )


survey_d_total %<>% shift_loc("measure", "MEASURED IN CWT", 1, 1 )

survey_d_total %<>% shift_loc("measure", "MEASURED IN TONS", 1, 1 )


survey_d_total %<>% shift_loc("measure", "MEASURED IN CWT / ACRE", 1, 1 )

survey_d_total %<>% shift_loc("measure", "MEASURED IN TONS / ACRE", 1, 1 )


#### split the mkt column
```

```
survey_d_total <- survey_d_total |>
  separate_wider_delim(cols = mkt,
                       delim = " - ",
                       names = c("col3",
                                 "col4"),
                    too_many = "merge",
                     too_few = "align_start")
```

there are two markets for Strawberries – Fresh Marketing and Processing

make a table for each

from the Survey Totals

we have reports for

Markets: Fresh and Processing Operations: Growing and Production

```
survey_d_total %<>%
  select(-`State ANSI`)

survey_d_total <- survey_d_total |>
  group_by(Year) |>
  group_by(State) |>
  group_by(Period) |>
  group_by(col3)



# unique(survey_d_total$col3)
# unique(survey_d_total$col4)

# mv <- survey_d_total |> filter(col3=="PRODUCTION") |> count()
# mv1 <- which(survey_d_total$col3 == "PRODUCTION")
# mv2 <- is.na(survey_d_total$col4[mv1])
# sum(mv2) == length(mv1)

survey_d_total <-  survey_d_total |>
  shift_loc(col_name = "col3", dat_name = "PRODUCTION", 2, 1)

# mv1 <- which(survey_d_total$col3 == "PRICE RECEIVED")
# mv2 <- is.na(survey_d_total$col4[mv1])
# sum(mv2) == length(mv1)


survey_d_total <-  survey_d_total |>
  shift_loc(col_name = "col3",
            dat_name = "PRICE RECEIVED", 2, 1)

survey_d_total <-  survey_d_total |>
  rename(market = col3, product_price = col4, summ = measure, measure = other)


## fix ACRES HARVESTED
## the category "HARVESTED" and its measure "acRES" are in reverse
```

```r
## order

h_index <- which(str_detect(survey_d_total$market, "ACRES HARVESTED") == TRUE)
survey_d_total$product_price[h_index] <- "HARVESTED"
survey_d_total$measure[h_index] <- "acres"
survey_d_total$market[h_index] <- NA

## fix ACRES PLANTED
p_index <- which(str_detect(survey_d_total$market, "ACRES PLANTED") == TRUE)
survey_d_total$product_price[p_index] <- "PLANTED"
survey_d_total$measure[p_index] <- "Acres"
survey_d_total$market[p_index] <- NA

## fixed up measure column
survey_d_total$measure <- str_replace(survey_d_total$measure, "MEASURED IN ", "")

## Other table fix-ups
## move Yield

y_index <- which(str_detect(survey_d_total$market, "YIELD") == TRUE)
survey_d_total$product_price[y_index] <- "YIELD"
survey_d_total$market[y_index] <- NA

ns_index <- which(str_detect(survey_d_total$market, "NOT SOLD") == TRUE)
survey_d_total$product_price[ns_index] <- "NOT SOLD"
survey_d_total$market[ns_index] <- NA

u_index <- which(str_detect(survey_d_total$market, "UTILIZED") == TRUE)
survey_d_total$product_price[u_index] <- "UTILIZED"
survey_d_total$market[u_index] <- NA

rm(ns_index, p_index, u_index, y_index, h_index, straw_sur2)
```

```r
# unique(survey_d_total$State)
# [1] "US TOTAL"       "CALIFORNIA"
# [3] "FLORIDA"        "OTHER STATES"
# [5] "NEW YORK"       "NORTH CAROLINA"
# [7] "OREGON"         "WASHINGTON"


sur_tot_ca <- survey_d_total |>
  filter(State == "CALIFORNIA")

sur_tot_fl <- survey_d_total |>
  filter(State == "FLORIDA")

sur_tot_ny <- survey_d_total |>
  filter(State == "NEW YORK")

sur_tot_or <- survey_d_total |>
  filter(State == "OREGON")

sur_tot_nc <- survey_d_total |>
```

```
  filter(State == "NORTH CAROLINA")

sur_tot_wa <- survey_d_total |>
  filter(State == "WASHINGTON")

sur_tot_other <- survey_d_total |>
  filter(State == "OTHER STATES")

sur_tot_US <- survey_d_total |>
  filter(State == "US TOTAL")
```

Florida - California - 2018 -2023

```
sur_CA_23 <- sur_tot_ca |> filter(Year == "2023") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")


sur_fl_23 <- sur_tot_fl |> filter(Year == "2023") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

sur_CA_22 <- sur_tot_ca |> filter(Year == "2022") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

sur_fl_22 <- sur_tot_fl |> filter(Year == "2022") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

sur_CA_21 <- sur_tot_ca |> filter(Year == "2021") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

sur_fl_21 <- sur_tot_fl |> filter(Year == "2021") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

sur_CA_20 <- sur_tot_ca |> filter(Year == "2020") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")


sur_fl_20 <- sur_tot_fl |> filter(Year == "2020") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

sur_CA_19 <- sur_tot_ca |> filter(Year == "2019") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")
```

```r
sur_fl_19 <- sur_tot_fl |> filter(Year == "2019") |>
  select(market, product_price, summ, measure, Value) |>
   filter(Value != "(D)")

# sur_CA_18 <- sur_tot_ca |> filter(Year == "2018") |>
#   select(market, product_price, summ, measure, Value) |>
#    filter(Value != "(D)")
#
#
# sur_fl_18 <- sur_tot_fl |> filter(Year == "2018") |>
#   select(market, product_price, summ, measure, Value) |>
#    filter(Value != "(D)")

sur_fl_19 <- sur_fl_19 |> drop_one_value_col()
sur_fl_20 <- sur_fl_20 |> drop_one_value_col()
sur_fl_21 <- sur_fl_21 |> drop_one_value_col()
sur_fl_22 <- sur_fl_22 |> drop_one_value_col()
sur_fl_23 <- sur_fl_23 |> drop_one_value_col()

sur_CA_23 <- sur_CA_23 |> drop_one_value_col()
sur_CA_22 <- sur_CA_22 |> drop_one_value_col()
sur_CA_21 <- sur_CA_21 |> drop_one_value_col()
sur_CA_20 <- sur_CA_20 |> drop_one_value_col()
sur_CA_19 <- sur_CA_19 |> drop_one_value_col()

sur_CA_19 <- sur_CA_19 |> rename(value_19 = Value)
sur_CA_20 <- sur_CA_20 |> rename(value_20 = Value)
sur_CA_21 <- sur_CA_21 |> rename(value_21 = Value)
sur_CA_22 <- sur_CA_22 |> rename(value_22 = Value)
sur_CA_23 <- sur_CA_23 |> rename(value_23 = Value)

sur_CA_19_23 <- cbind(sur_CA_19,sur_CA_20[,3],sur_CA_21[,3],sur_CA_22[,3],sur_CA_23[,3])

sur_fl_19 <- sur_fl_19 |> rename(value_19 = Value)
sur_fl_20 <- sur_fl_20 |> rename(value_20 = Value)
sur_fl_21 <- sur_fl_21 |> rename(value_21 = Value)
sur_fl_22 <- sur_fl_22 |> rename(value_22 = Value)
sur_fl_23 <- sur_fl_23 |> rename(value_23 = Value)

sur_fl_19_23 <- cbind(sur_fl_19,sur_fl_20[,3],sur_fl_21[,3],sur_fl_22[,3],sur_fl_23[,3])
```

```r
library(knitr)
library(kableExtra)

fl_19_23 <- as.data.frame(sur_fl_19_23)

colnames(fl_19_23) <- c("Measure", "Units", "2019", "2020", "2021", "2022", "2023")



fl_19_23$Measure <- c("Price", "Havested", "Planted", "Production", "Production", "Yield", "Not Sold",

fl_19_23 <- fl_19_23 |> select(-Units)
```

```
Units <- c("$-cwt", "ac", "ac", "$", "cwt", "cwt-ac", "cwt", "cwt")

fl_19_23 <- fl_19_23 |> mutate(Units, .after=Measure)



#fl |> kable()
```

California and Florida chemicals

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

survey_d_chem <- survey_d_chem |> select(-`State ANSI`)

## California Chemicals

# ca_chem <- survey_d_chem |> filter(State=="CALIFORNIA") |>
#   select()

survey_d_chem <- survey_d_chem |>
  separate_wider_delim(cols = mkt,
                       delim = " - ",
                       names = c("mk1",
                                 "mk2"),
                    too_many = "merge",
                     too_few = "align_start")


survey_d_chem$measure <- str_replace(survey_d_chem$measure, "MEASURED IN ", "")

# unique(survey_d_chem$`Domain Category`)

survey_d_chem$`Domain Category` <- str_replace(survey_d_chem$`Domain Category`, "CHEMICAL, ", "")

survey_d_chem <- survey_d_chem |> rename(chem = `Domain Category`)

survey_d_chem <- survey_d_chem |>
          separate_wider_delim(cols = chem,
                               delim = ": ",
                               names = c("type",
                                 "chem_type"),
                             too_many = "merge",
                              too_few = "align_start")

  # s1 <- survey_d_chem$col2 == survey_d_chem$type
  # sum(s1)

  survey_d_chem <- survey_d_chem |> select(-col2)

  survey_d_chem <- survey_d_chem |>
    rename(chem_name = chem_type)
```

```
  survey_d_chem$chem_name <- str_replace(survey_d_chem$chem_name, "^\\(", "")

   survey_d_chem$chem_name <- str_replace(survey_d_chem$chem_name, "\\)$", "")

survey_d_chem <- survey_d_chem |>
  separate_wider_delim(cols = chem_name,
                       delim = " = ",
                       names = c("chem_name",
                                 "chem_index"),
                    too_many = "error",
                     too_few = "align_start")

chemicals_Used_cA <- survey_d_chem |>
  filter(State == "CALIFORNIA") |>
  select(type, chem_name, chem_index)

cA_chem_fung <- chemicals_Used_cA |>
  filter(type == "FUNGICIDE") |>
  distinct()


cA_chem_herb <- chemicals_Used_cA |>
  filter(type == "HERBICIDE") |>
  distinct()


cA_chem_insect <- chemicals_Used_cA |>
  filter(type == "INSECTICIDE") |>
  distinct()

cA_chem_other <- chemicals_Used_cA |>
  filter(type == "OTHER") |>
  distinct()
```

California and Florida fertilizers

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

survey_d_fert <- survey_d_fert |> select(-`State ANSI`)

survey_d_fert <- survey_d_fert |>
  separate_wider_delim(cols = mkt,
                       delim = " - ",
                       names = c("mk1",
                                 "mk2"),
                    too_many = "merge",
                     too_few = "align_start")

survey_d_fert$measure <- str_replace(survey_d_fert$measure, "MEASURED IN ", "")

# unique(survey_d_chem$`Domain Category`)
```

```
survey_d_fert$`Domain Category` <- str_replace(survey_d_fert$`Domain Category`, "CHEMICAL, ", "")

survey_d_fert <- survey_d_fert |> rename(chem = `Domain Category`)

survey_d_fert <- survey_d_fert |>
          separate_wider_delim(cols = chem,
                               delim = ": ",
                               names = c("type",
                                 "chem_type"),
                             too_many = "merge",
                              too_few = "align_start")

  survey_d_fert <- survey_d_fert |>
    rename(chem_name = chem_type)

  survey_d_fert$chem_name <- str_replace(survey_d_fert$chem_name, "^\\(", "")

   survey_d_fert$chem_name <- str_replace(survey_d_fert$chem_name, "\\)$", "")

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

```
survey_d_total_ca <- survey_d_total |>
  filter(State == "CALIFORNIA")

ca_tab <- survey_d_total_ca |> group_by(Year, Period
                                        )
ca_tab_22 <- survey_d_total_ca |> filter(Year == 2022)

ca_tab_22 <- ca_tab_22 |> drop_one_value_col()

ca_tab_22 <- ca_tab_22 |>
  filter(Period == "YEAR")

ca_tab_22 <- ca_tab_22 |>
  filter(Value != "(D)")

ca_tab_22 <- ca_tab_22 |> drop_one_value_col()
```

Fungicide used in strawberry cultivation

An example: AZOXYSTROBIN

```
survey_d_fung <- survey_d_chem %>% filter(type == "FUNGICIDE")
survey_d_fung <- survey_d_fung %>% filter(Value != "(D)") %>% filter(Value != "(NA)")
```

Among the chemicals listed, several pose significant risks to aquatic ecosystems. One such example is AZOXYSTROBIN. Research shows that as little as 0.55 ml/L of AZOXYSTROBIN can cause severe harm to aquatic crustaceans, as indicated in the data.

```
survey_d_AZOXYSTROBIN <- survey_d_fung %>% filter(chem_name == "AZOXYSTROBIN")
```

Both Florida and California use approximately 0.3 lb/acre/year of AZOXYSTROBIN. Although this is a relatively small quantity, it can still be damaging to aquatic organisms when runoff from rain carries the chemical into nearby water bodies.

On the other hand, AZOXYSTROBIN poses minimal to moderate risks to mammals and other animals, making it a relatively safe fungicide for farmers to use in strawberry cultivation.

Top three fungicide used on strawberry in California

```
survey_ca_fung_order <- survey_d_fung %>% filter(State == "CALIFORNIA", measure == "LB / ACRE / YEAR") %
```

From this table we can see that in California, sulfur, captan, and thiram are among the most commonly used fungicides. However, two other chemicals also feature prominently.

Unexpected Result The first unexpected result is FOSETYL-AL, a fungicide primarily used to control diseases caused by oomycetes such as downy mildew and Phytophthora species. It's possible that the high usage in 2019 indicates a significant disease outbreak in strawberries during that period, prompting farmers to use fosetyl-al.

The second is POTASSIUM BICARBONATE, commonly used in organic farming as an eco-friendly fungicide. It also acts as a soil amendment by raising the pH of acidic soils. Two possible explanations for its usage in 2021 are either experiments in organic growth or attempts to correct overly acidic soils in California's strawberry fields.

Common three fungicide(sulfur to be discussed seperately) Firstly, captan is widely used to control a range of fungal pathogens, such as powdery mildew and fruit rot. According to safety guidelines, captan poses little risk to humans and animals when handled properly, which is why it's commonly used.

Secondly, thiram often complements captan's function. Unlike captan, however, thiram can cause serious harm to aquatic life, including fish and crustaceans. Care must be taken to prevent this chemical from contaminating water sources.

Sulfur plays different roles in fungicide and fertilizer Interestingly, sulfur appears in both the fungicide and fertilizer categories. In fertilizers, sulfur helps improve the absorption of nitrogen and phosphate. However, it's important to adjust the dosage appropriately, as long-term sulfur use can lower soil pH and lead to "leaf burn" in plants, which is caused by deficiencies in manganese and iron.

```
survey_d_sulfur <- survey_d_fert %>% filter(chem_name == "SULFUR")
```

As a fungicide, sulfur-based pesticides pose risks to human health, plant damage, and the environment. These chemicals can irritate the skin, eyes, and respiratory system, potentially worsening conditions like asthma. Prolonged exposure may lead to chronic respiratory issues and allergic reactions.

```
survey_f_sulfur <- survey_d_fung %>% filter(chem_name == "SULFUR")
```

According to the data, about 30 lb/acre/year of sulfur is used in fungicides. While the exact amount of sulfur or its compounds isn't clear, the quantity is substantial and warrants concern for its effects on both humans and animals. Sulfur is often used to keep fruits like strawberries fresh, but its use should be carefully monitored to avoid excessive exposure.

Chemicals usd in strawberry cultivaion

Six deadly carcinogens from WHO list

captafol

ethylene dibromide also

glyphosate See also 1
2
3
4

malathion 1 2

diazinon 1 2 3

dichlorophenyltrichloroethane (DDT) 1 2
3

(https://www.epa.gov/ingredients-used-pesticide-products/ddt-brief-history-and-status#:/~:text=DDT%20(dichloro%2Ddi

For contrast

Azadirachtin 1 2 3

Sources of agricultural chemical information

for EPA number lookup epa numbers

Active Pesticide Product Registration Informational Listing

CAS for Methyl Bromide

pesticide chemical search

toxic chemical dashboard

pubChem

The EPA PC (Pesticide Chemical) Code is a unique chemical code number assigned by the EPA to a particular pesticide active ingredient, inert ingredient or mixture of active ingredients.

Investigating toxic pesticides

start here with chem PC code

step 2 to get label (with warnings) for products using the chemical

Pesticide Product and Label System

Search by Chemical

CompTox Chemicals Dashboard

Active Pesticide Product Registration Informational Listing

OSHA chemical database

Pesticide Ingredients

NPIC Product Research Online (NPRO)

Databases for Chemical Information

Pesticide Active Ingredients

TSCA Chemical Substance Inventory

glyphosate