16-10-0117-01 Principal statistics for manufacturing by NAICS

Leila Bautista and Andre Bastos

2026-02-06

Introduction

This R Markdown document is intended to facilitate automation in the loading of the Statistics Canada's Table: 16-10-0117-01 (formerly CANSIM 301-0008): Principal statistics for manufacturing industries.

There are 4 parts to script: 1. Load data from Statistics Canada via R API 2. Data cleaning 3. Addressing data suppression: hierarchy imputation, time series interpolation and row-level residuals 4. Export and save the output (.csv) in the Data Library.

The following document should be interpreted using the color legend: - Blue: R functions that manipulate the data - Black: object names and partial outputs - Green: database items (column labels and values) - Orange: explanatory notes for each step

1. Load data from Statistics Canada via API

This section connects directly to Statistics Canada's online database and automatically downloads the manufacturing statistics table via API. The code loads specialized packages that make it easier to pull large datasets and quickly inspect their structure. Once the table is retrieved, the script provides a brief preview of what the data looks like, helping the user confirm that the correct file has been imported before moving on.

The code below will install and activate the required packages for data manipulation in this code. Use ??package.name for each one to access help.

```
# Install packages if needed. Use install.packages("name, name", ...) if errors appear
packages <- c("cansim",
              "data.table",
              "dplyr",
              "tidyr",
              "stringr",
              "zoo")
lapply(packages, require, character.only = TRUE)
```

```
## Loading required package: cansim

## Loading required package: data.table

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
```

```
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: tidyr
```

```
## Loading required package: stringr
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:data.table':
##
##     yearmon, yearqtr
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
##
## [[6]]
## [1] TRUE
```

```r
# Load Statistics Canada table 16-10-0117-01
dt <- as.data.table(get_cansim("16-10-0117-01"))
```

```
## Accessing CANSIM NDM product 16-10-0117 from Statistics Canada
```

```
## Parsing data
```

```r
# Inspect the data

# Optional print
# print(dt)

# Optional: Compact display of the dataset structure
# str(dt)

# Optional: Show first few rows
# dt[1:6]
```

2. Data Cleaning

Cleaning the data entails assigning the desired names and categories to the columns. Formulas are also used to ensure proper handling of text and numeric values. Here, the script standardizes and organizes the raw dataset so it can be used reliably for later steps. It renames long or inconsistent column titles, converts important fields to the correct data types, removes duplicates, fixes formatting issues in the NAICS codes, and filters the data to only include the provinces and industries relevant to food and beverage manufacturing. The result is a clean, simplified dataset with only the necessary variables, ensuring that all numbers and labels are consistent and ready for analysis.

```r
# Create a clean working copy
FoodBev_Manu <- copy(dt)

# Standardize column names
setnames(
  FoodBev_Manu,
  names(FoodBev_Manu),
  make.names(names(FoodBev_Manu))
)

# Rename NAICS classification code column explicitly
setnames(
  FoodBev_Manu,
  "Classification.Code.for.North.American.Industry.Classification.System..NAICS.",
  "NAICS_Code"
)
setnames(
  FoodBev_Manu,
  "North.American.Industry.Classification.System..NAICS.",
  "NAICS"
)

# Set as characters
FoodBev_Manu$GEO <- as.character(FoodBev_Manu$GEO)
FoodBev_Manu$Principal.statistics <- as.character(FoodBev_Manu$Principal.statistics)

# Drop square brackets from NAICS_Code
FoodBev_Manu$NAICS_Code <- gsub("\\[([^]]+)\\]", "\\1", FoodBev_Manu$NAICS_Code)

# Ensure REF_DATE is numeric (year)
FoodBev_Manu[, REF_DATE := as.integer(REF_DATE)]

# Remove duplicate rows, if any
FoodBev_Manu <- unique(FoodBev_Manu)

#  Filter to selected provinces/territories
FoodBev_Manu <- FoodBev_Manu %>%
  mutate(GEO = str_trim(as.character(GEO))) %>%
  filter(GEO %in% c(
    "British Columbia", "Alberta", "Saskatchewan", "Manitoba",
    "Ontario", "Quebec", "New Brunswick", "Prince Edward Island",
    "Nova Scotia", "Newfoundland and Labrador", "Yukon",
    "Northwest Territories", "Nunavut"))

# Filter for NAICS codes starting with [311 or [312
```

```
FoodBev_Manu <- FoodBev_Manu %>%
  filter(
    grepl("^(311|312)", NAICS_Code),
    Principal.statistics %in% c("Revenue from goods manufactured"
                                , "Total expenses")
  )

# Normalize 'NAICS' column to text
FoodBev_Manu$NAICS <- as.character(FoodBev_Manu$NAICS)

# Combine NAICS_Code and NAICS description into a single column
FoodBev_Manu$NAICS <- paste(FoodBev_Manu$NAICS_Code, FoodBev_Manu$NAICS
                            , sep = " ")

# Keep only selected columns
FoodBev_Manu <- FoodBev_Manu[
  ,
  .(
    REF_DATE,
    GEO,
    Principal.statistics,
    NAICS,
    NAICS_Code,
    val_norm,
    UOM
  )
]
```

3. Addressing data suppression:

imputation, interpolation, and row-level residuals

This section fills in missing values that Statistics Canada suppresses for confidentiality.

3a. Hierarchy imputation: the script rebuilds higher-level industry totals by summing the more detailed 5- and 6-digit NAICS categories and comparing them to existing 4-digit values, ensuring the most complete figure is used.

```
# Copy original table
NAICS5and6_to4 <- copy(FoodBev_Manu)

# Filter to get rows with 5- or 6-digit NAICS
NAICS5and6_to4 <- NAICS5and6_to4[
   grepl("^\\[?\\d{5}", NAICS)]

# Extract 4-digit NAICS from bracketed 5- and 6-digit code
NAICS5and6_to4[, NAICS := substr(NAICS, 1, 4)]

# Aggregation of NAICS Levels 5 and 6 simultaneously
# Filter rows to obtain NAICS5 and NAICS6 at different data frames
J6 <- NAICS5and6_to4 %>% filter(nchar(NAICS_Code)==6)
J5 <- NAICS5and6_to4 %>% filter(nchar(NAICS_Code)==5)
```

```r
# Aggregate J5 to the 'NAICS' 4 level to get the sum of val_norm
J5 <- J5 %>% group_by(REF_DATE, GEO, Principal.statistics, NAICS, UOM) %>%
  summarise(val_norm = sum(val_norm, na.rm = TRUE)) %>% ungroup()
```

```
## `summarise()` has grouped output by 'REF_DATE', 'GEO', 'Principal.statistics',
## 'NAICS'. You can override using the `.groups` argument.
```

```r
    # na.rm=TRUE handles missing values

# Aggregate J6 to the 'NAICS' 4 level to get the sum of val_norm
J6 <- J6 %>% group_by(REF_DATE, GEO, Principal.statistics, NAICS, UOM) %>%
  summarise(val_norm = sum(val_norm, na.rm = TRUE)) %>% ungroup()
```

```
## `summarise()` has grouped output by 'REF_DATE', 'GEO', 'Principal.statistics',
## 'NAICS'. You can override using the `.groups` argument.
```

```r
  # na.rm=TRUE handles missing values

# Join the two aggregated data frames by 'Region'
NAICS5and6_to4 <- left_join(J5, J6, by = c("REF_DATE", "GEO"
                      , "Principal.statistics", "NAICS", "UOM"))

# Compare the 'val_norm' columns to find the overall maximum
NAICS5and6_to4 <- NAICS5and6_to4 %>%
  mutate(
    val_norm = pmax(val_norm.x, val_norm.y, na.rm = TRUE)
  )

# Drop unused columns after the join
NAICS5and6_to4 <- subset(NAICS5and6_to4, select = -c(val_norm.x, val_norm.y))

# Drop unused data frames after the join
rm(J5, J6)

# Rename NAICS to NAICS_Code
NAICS5and6_to4 <- NAICS5and6_to4 %>% rename(NAICS_Code = NAICS)

# NAICS Level 4
# Copy original table
NAICS4 <- copy(FoodBev_Manu)

# Filter to, 4-digit NAICS, and revenue/expenses
NAICS4 <- NAICS4[
    grepl("^.{4} ", NAICS) ]

# Merge NAICS4 and NAICS5and6_to4 first
NAICS4 <- merge(
  NAICS4,
  NAICS5and6_to4,
  by = c("REF_DATE", "GEO", "NAICS_Code", "Principal.statistics"),
  all = TRUE,
  suffixes = c(".4", ".5")  # makes it explicit
)

# Combine val_norm columns from first two tables
```

```r
NAICS4[, val_norm := coalesce(val_norm.4, val_norm.5)]
NAICS4[, c("val_norm.4", "val_norm.5") := NULL]

# Replace 0 with NA in val_norm
NAICS4[, val_norm := fifelse(val_norm == 0, NA_real_, val_norm)]

# Drop unused (redundant) columns after the join
NAICS4 <- subset(NAICS4, select = -UOM.5)

# Rename UOM.4 to UOM
NAICS4 <- NAICS4 %>% rename(UOM = UOM.4)

# There are a couple of cell in the dataset that need to be manually set to NA
# after inspection because the data suppression followed an atypical pattern,
# causing only one of the NAICS 5-digit (Distilleries) categories to be
# reported, resulting in positive but extremely downward biased value in the
# hierarchical imputation. They will be estimated via time series interpolation
# in the next step.
NAICS4$val_norm[
  which(NAICS4$NAICS_Code == "3121"
        & NAICS4$REF_DATE == "2022"
        & NAICS4$Principal.statistics == "Revenue from goods manufactured"
        & NAICS4$GEO == "British Columbia"
        )] <- NA_real_

NAICS4$val_norm[
  which(NAICS4$NAICS_Code == "3121"
        & NAICS4$REF_DATE == "2016"
        & NAICS4$Principal.statistics == "Revenue from goods manufactured"
        & NAICS4$GEO == "British Columbia"
        )] <-  NA_real_

NAICS4$val_norm[
  which(NAICS4$NAICS_Code == "3121"
        & NAICS4$REF_DATE == "2017"
        & NAICS4$Principal.statistics == "Revenue from goods manufactured"
        & NAICS4$GEO == "British Columbia"
        )] <-  NA_real_

# Rename NAICS labels for consistency
NAICS4$NAICS[
  which(NAICS4$NAICS_Code == "3111")] <- "3111 Animal food Manufacturing"
NAICS4$NAICS[
  which(NAICS4$NAICS_Code == "3117")] <- "3117 Seafood product preparation and packaging"
NAICS4$NAICS[
  which(NAICS4$NAICS_Code == "3123")] <- "3123 Cannabis product manufacturing"

# Appending NAICS 4 to 3
# Copy original table
NAICS3 <- copy(FoodBev_Manu)

# Filter to BC, 3-digit NAICS, and revenue measure
NAICS3 <- NAICS3[
    grepl("^.{3} ", NAICS) ]

# Copy original table
NAICS4_to3 <- copy(NAICS4)

# Filter rows where NAICS starts with "311 " or "312"
NAICS4_to3 <- NAICS4_to3[
   grepl("^\\d{4}$", NAICS_Code)]

# Extract 3-digit NAICS from bracketed 4-digit code
NAICS4_to3[, NAICS_Code := substr(NAICS_Code, 1, 3)]

# Aggregate to 3-digit level
## Andre to check sum()
NAICS4_to3 <- NAICS4_to3[
  , .(
      val_norm = sum(val_norm, na.rm = TRUE)
    ),
  by = .(NAICS_Code, REF_DATE, GEO, Principal.statistics)
]

# Merge NAICS3 and NAICS4_to3 first
NAICS3 <- merge(
  NAICS3,
  NAICS4_to3,
  by = c("REF_DATE", "GEO", "NAICS_Code", "Principal.statistics"),
  all = TRUE,
  suffixes = c(".3", ".4")  # makes it explicit
)

# Combine val_norm columns from first two tables
NAICS3[, val_norm := coalesce(val_norm.3, val_norm.4)]
NAICS3[, c("val_norm.3", "val_norm.4") := NULL]

# Replace 0 with NA in val_norm
NAICS3[, val_norm := fifelse(val_norm == 0, NA_real_, val_norm)]

# Imputation of a single value for the last year (2024) of the 312 (Beverage
# and tobacco product manufacturing) category in B.C. using the same percentage
# year-over-year 2023-2024 as the 3121 (Beverage manufacturing) category in B.C.
# for the same year
NAICS3$val_norm[
  which(NAICS3$NAICS_Code == "312"
        & NAICS3$REF_DATE == "2024"
        & NAICS3$Principal.statistics == "Revenue from goods manufactured"
        & NAICS3$GEO == "British Columbia"
        )] <- 2952935600.48
```

3b. Time series interpolation: This method estimates missing values based on trends in the available data where up to 2 years of data in the same category are suppressed but values exist before and after.

```
# Linear interpolation is used to address missing values that cannot be imputed
# hierarchically across NAICS levels. The following code creates the column
# `value_interp`, which duplicates `val_norm` and estimates missing values for
```

```r
# up to 2 years of consecutive missing data in each category.

#Starting with NAICS 3-digit:
NAICS3 <- NAICS3 %>%
  group_by(GEO, NAICS_Code, Principal.statistics) %>%
  arrange(REF_DATE, .by_group = TRUE) %>%
  mutate(val_interp = zoo::na.approx(val_norm, x = REF_DATE, na.rm = FALSE
                                     , maxgap = 2)) %>%
  ungroup()

# Repeat the same procedure for NAICS 4-digit:
NAICS4 <- NAICS4 %>%
  group_by(GEO, NAICS_Code, Principal.statistics) %>%
  arrange(REF_DATE, .by_group = TRUE) %>%
  mutate(val_interp = zoo::na.approx(val_norm, x = REF_DATE, na.rm = FALSE
                                     , maxgap = 2)) %>%
  ungroup()
```

3c. NAICS-3 digit residual columns are calculated by subtracting known sub-industry totals from their parent category, allowing the model to calculate and display “Other” industry rows at NAICS 3-digit level (one for food, one for beverages, tobacco and cannabis). This method replaces categories labeled as “Other” in the original data or those where data suppression is a significant issue. This method also ensures that the sum of the values at NAICS-4 matches the total for NAICS -3 digit level.

```r
# Adjustment of a value in the year 2017 to prevent the value of the 3-digit
# NAICS [312] from being lower than that of the 4-digit NAICS [3121]
NAICS4$val_interp[
  which(NAICS4$NAICS_Code == "3121"
        & NAICS4$REF_DATE == "2017"
        & NAICS4$Principal.statistics == "Revenue from goods manufactured"
        & NAICS4$GEO == "British Columbia"
        )] <-  NAICS3$val_interp[which(NAICS3$NAICS_Code == "312"
            & NAICS3$REF_DATE == "2017"
            & NAICS3$Principal.statistics == "Revenue from goods manufactured"
            & NAICS3$GEO == "British Columbia"
            )]

# Combine NAICS levels 3 and 4 in the same object for easier processing
NAICS3and4 <- data.table::rbindlist(list(NAICS3, NAICS4), fill = TRUE)

# Define set of NAICS codes used to assemble 3119 (residual, other food manuf.):
# All except 3119 (Other food manuf.)
naics_set311 <- c("311", "3111", "3112", "3113", "3114"
                  , "3115", "3116", "3117", "3118")

# Define set of NAICS codes to recalculate 3119 (residual, other food manuf.):
# All except 3119 (Other food manuf.)
naics_set311 <- c("311", "3111", "3112", "3113", "3114"
                  , "3115", "3116", "3117", "3118")
label311 <- "3119"

# Subtotals for each code within groups
subtotals <- NAICS3and4[NAICS_Code %in% naics_set311,
  .(val_interp = sum(val_interp, na.rm = TRUE)),
  by = .(REF_DATE, GEO, Principal.statistics, UOM, NAICS_Code)
]

# Reshape to wide and compute combo
wide311 <- dcast(
  subtotals,
  REF_DATE + GEO + Principal.statistics + UOM ~ NAICS_Code,
  value.var = "val_interp"
)

# Calculate residuals to override suppression
wide311[, `:=`(
  val_interp311 = fifelse(is.na(`311`), 0, `311`)
              - fifelse(is.na(`3111`), 0, `3111`)
              - fifelse(is.na(`3112`), 0, `3112`)
              - fifelse(is.na(`3113`), 0, `3113`)
              - fifelse(is.na(`3114`), 0, `3114`)
              - fifelse(is.na(`3115`), 0, `3115`)
              - fifelse(is.na(`3116`), 0, `3116`)
              - fifelse(is.na(`3117`), 0, `3117`)
              - fifelse(is.na(`3118`), 0, `3118`),
  NAICS_Code = "3119",
  NAICS      = "3119 Other food manufacturing"
)]

# Drop unused columns to simplify the join
wide311 <- subset(wide311, select = -c(`311`, `3111`, `3112`, `3113`, `3114`
                                       , `3115`, `3116`, `3117`, `3118`))

# Rename 'val_interp311' to 'val_interp'
wide311 <- wide311 %>% rename(val_interp = val_interp311)

# Define set of NAICS to calculate 3129 (residual, Tobacco and Cannabis manuf.)
naics_set312 <- c("312", "3121")
label312 <- "3129"

# Subtotals for each code within groups
subtotals312 <- NAICS3and4[NAICS_Code %in% naics_set312,
  .(val_interp = sum(val_interp, na.rm = TRUE)),
  by = .(REF_DATE, GEO, Principal.statistics, UOM, NAICS_Code)
]

# Reshape wide and compute combo
wide312 <- dcast(
  subtotals312,
  REF_DATE + GEO + Principal.statistics + UOM ~ NAICS_Code,
  value.var = "val_interp")

# Calculate residuals to override suppression
wide312[, `:=`(
  val_interp312  = fifelse(is.na(`312`), 0, `312`)
              - fifelse(is.na(`3121`), 0, `3121`),
  NAICS_Code = label312,
  NAICS      = "3129 Tobacco and cannabis product manufacturing"

)]

# Drop unused columns to simplify the join
wide312 <- subset(wide312, select = -c(`312`, `3121`))

# Rename 'val_interp312' to 'val_interp'
wide312 <- wide312 %>% rename(val_interp = val_interp312)

# Join the two aggregated data frames by 'Region'
diff_rows_NAICS3and4 <- full_join(wide311, wide312, by = c("REF_DATE", "GEO"
                          , "Principal.statistics", "UOM", "NAICS_Code"
                          , "NAICS", "val_interp"))

# Drop unused NAICS codes after the residual calculation
NAICS3and4 <- NAICS3and4[ !NAICS3and4$NAICS_Code %in% c("3119", "3122", "3123"), ]

# Append calculated rows (residuals) and rename labels
NAICS3and4 <- rbindlist(list(NAICS3and4, diff_rows_NAICS3and4)
                                    , use.names = TRUE, fill = TRUE)
```

3d. Identify imputed or interpolated data

The script also tags all imputed or interpolated cells in red font so they can be easily identified. Using [if/then] logic, it checks each row for missing values that were filled in during the previous steps. If a value was imputed or interpolated, the font colour becomes red.

```
# Create function to check if a value is positive numeric and not NA
is_pos <- function(x) {
  is.numeric(x) & !is.na(x) & x > 0}

# Apply BC Gov blue font colour as the standard and red font to imputed values
# in val_norm
NAICS3and4$FontColour <- ifelse(is.na(NAICS3and4$val_norm)
                                        & is_pos(NAICS3and4$val_interp)
                                        , "Red", "#234075"
                                )
```

4. Export and save the output (.csv) in the Data Library

The final section writes the cleaned and fully reconstructed datasets to the Data Library in CSV format. The files are saved directly into the user's SharePoint-synced folder so they are available for internal use and version control. By exporting both the 3-digit and 4-digit NAICS datasets, the script ensures that analysts can immediately use the updated numbers in dashboards, briefings, or further modeling.

Remember to replace "\" with "/"

```
# Make sure the Unpublished data space is synced (from Web to your C:)
# Update the IDIR entry below
## Edit to the desired data library destination as: "C:/Users/Enter_Your_IDIR_here/..."
sharepoint_path <-
  "C:/Users/ABASTOS/Government of BC/SICI Data Library Subsite - Unpublished Data"

# Create a named list of data frames and file names
```

```r
dfs <- list(
  "FoodBevManu_NAICS4.csv" = NAICS4,
  "FoodBevManu_NAICS3.csv" = NAICS3,
  "FoodBevManu_NAICS3and4.csv" = NAICS3and4
)

# Loop through and write each CSV file to the Data Library (Unpublished)
for (fname in names(dfs)) {
  fwrite(dfs[[fname]], file.path(sharepoint_path, fname))
}
```