Beef Analysis - In Class

Load Data and Packages

```
# load packages
pacman::p_load(dplyr, tidyverse, ggplot2, here, readxl, janitor, modelsummary, readr, gridExtra)

# read beef_consumerdata.csv as beef_raw using read.csv() and here() functions
beef_raw <- read.csv(here("data", "beef_consumerdata.csv"))

# create new df called beef that cleans up variable names using clean_names() function
  # specify as upper_camel option meaning first letter after any separator (space, period, underscore)
  # will be capitalized, e.g., if the variable name initially was "Input.per.capita"
  # this function will rename it to InputPerCapita
  # we will also rename ConocAmbien to Knoweldge, and BeefConsumptionXTimesPerWeek
  # to BeefConsumption

# beef <- beef_raw %>%
#   clean_names(., "upper_camel") %>%
#   rename(insert_code_here)
```

Data Cleaning and Descriptives

- Tabulate every single variable

```
# insert_code_here
```

- Use `mutate()` and `ifelse()` to recode variables where values do not match dictionary
  - recode `Knowledge` = 2.5 to 3, recode `Knowledge` = 0 to 1
  - recode `Preference1` = 6 to 5
  - recode `ChickenConsumption` = 8 to 7
- Use `mutate()` and `factor(var, labels = c("label1", "label2"))` to convert categorical variables to factor variables for use in analysis later on
  - `City` and `Gen`
- Use `mutate()` and `parse_number()` to extract numbers from dollar amounts
  - `HowMuchMorePerPound` (call this var as `WtpAmount`)
- Use `mutate()` and `ifelse()` to recode `WtpAmount = 0` if `PayMore = 0`

```r
# beef <- beef %>%
#   mutate(Knowledge = insert_code_here,
#          Preference1 = insert_code_here,
#          ChickenConsumption = insert_code_here,
#          City = insert_code_here,
#          Gen = insert_code_here,
#          WtpAmount = insert_code_here,
#          WtpAmount = insert_code_here,
#          Study = factor(Study, labels = c("No education", "Incomplete primary",
#                                           "Full primary", "Incomplete secondary",
#                                           "Full secondary", "Technical",
#                                           "Incomplete university", "University",
#                                           "Postgraduate")),
#          Strata = factor(Strata, labels = c("Strata 1 (lowest)", "Strata 2",
#                                             "Strata 3", "Strata 4", "Strata 5",
#                                             "Strata 6 (highest)")),
#          Min12 = factor(Min12, labels = c("No children under 12", "With children under 12")),
#          Preference1 = factor(Preference1, labels = c("Beef", "Chicken", "Fish",
#                                                       "Pork", "Other")),
#          PayMore = factor(PayMore, labels = c("No", "Yes")),
#          Knowledge = factor(Knowledge, labels = c("No knowledge", "Very little knowledge",
#                                                   "Some knowledge", "Good knowledge", "Much knowledge
#          Location = factor(WhereBeef, labels = c("Home", "Out of Home",
#                                                  "Both Home and Outside")),
#          IncomePerCapita = parse_number(InputPerCapita),
#          BeefExpenditurePerCapita = parse_number(BeefExpenditurePerCapita))
```

- Create a new dataframe called `beef_expenditure_outlier` that contains rows where `BeefExpenditurePerCapita` $> 99^{th}$ perc
- Create an object called `income_exp` that contains a scatterplot of `BeefExpenditurePerCapita` and `IncomePerCapita`
  - Plot observations where `BeefExpenditurePerCapita` $> 99^{th}$ percentile as red dots
- Using `subset()`, remove observations from `beef` if `BeefExpenditurePerCapita` $> 99^{th}$ perc
- Create a scatterplot of `BeefExpenditurePerCapita` and `IncomePerCapita` without the outliers
- Use `grid.arrange()` to plot the two scatterplots side by side

```r
# create a dataframe called beef_expenditure_outlier that contains rows where
  # BeefExpenditurePerCapita > 99th percentile

# beef_expenditure_outlier <- insert_code_here

# create an object called income_exp that contains that scatterplot of the whole dataset
  # indicate that the outliers should be color red

# income_exp <- insert_code_here +
#   theme_classic() +
#   labs(title = "Whole Sample", y = "Beef expenditure per capita",
#        x = "Income per capita")

# using the subset() function, only keep obvs if BeefExpenditurePerCapita < 99th perc or if NA

# beef <- subset(insert_code_here)
```

```r
# create an object called income_exp_removed that contains that scatterplot of the whole dataset

# income_exp_removed <- insert_code_here +
#   theme_classic() +
#   labs(title = "Removed >99%", y = "Beef expenditure per capita",
#        x = "Income per capita")

# use grid.arrange() to plot the two scatterplot objects side by side

# grid.arrange(income_exp, income_exp_removed, ncol = 2)
```

Note: The codes are the same to remove observations if `IncomePerCapita` $> 99^{th}$ percentile and if `WtpAmount` $> 99^{th}$ percentile. Use `geom_histogram()` to replicate the graphs

- Remove outliers based on `IncomePerCapita` and `WtpAmount` values and overwrite the `beef` dataframe

```r
# only keep obvs if IncomePerCapita and WtpAmount < 99th perc or if NA

# beef <- beef %>%
#   insert_code_here
```

- Use `datasummary_skim()` from the `{datasummary}` package to generate descriptive stats

```r
# descriptive stats for categorical

# beef %>%
#   select(City, Gen, Study, Strata, Min12, Preference1, Knowledge, PayMore) %>%
#   insert_code_here

# descriptive stats for numerical

# beef %>%
#   select(Age, HhSize, BeefConsumption, ChickenConsumption, PorkConsumption, FishConsumption, WtpAmoun
#    insert_code_here
```

- Use `datasummary_balance()` to stratify by a certain variable

```r
# stratifying by gender

# datasummary_balance(insert_code_here,
#                     data = beef %>% select(City, Gen, Study, Strata, Min12,
#                                            Preference1, Knowledge, PayMore),
#                     dinm = F,
#                     title = "Descriptive Statistics of Categorical Variables, Stratified by Gender")
#
# datasummary_balance(insert_code_here,
#                     data = beef %>%
#                       select(Gen, Age, HhSize, BeefConsumption, ChickenConsumption,
#                              PorkConsumption, FishConsumption, WtpAmount,
#                              IncomePerCapita, BeefExpenditurePerCapita,
#                              BeefComsumptionPerCapita),
#                     dinm = F,
#          title = "Descriptive Statistics of Continuous Variables, Stratified by Gender")
```

Statistical Tests

- Use `cor()` and `cor.test()` to calculate correlations

```
# calc correlation between WtpAmount and Age

# insert_code_here
```

- Use `t.test()` to run t-tests - recode some variables to numeric first

```
# create a new var called GenNumeric that codes 0 if female and 1 if male
# insert_code_here

# t.test if WTP is different for gender (Male = 1)
# insert_code_here
```

- Create a boxplot to look distribution of `WtpAmount` by `Study` groups
- Use `lm()` and `anova()` to run the ANOVA test, and `TukeyHSD(aov())` to determine contrasts
  - Read here for more info

```
# create a boxplot
# insert_code_here +
#   theme_classic() +
#   theme(axis.text.x = element_text(angle = 45, hjust=1)) +
#   labs(x = "Level of Education", y = "WTP Amount")

# run a regression and call it lm_study
# insert_code_here

# print the anova results
# insert_code_here

# print the Tukey Honest Significant Differences test
# insert_code_here
```

Qualtrics API

Instead of downloading your survey every time from the Qualtrics website, you can connect directly to the server with an API using the `{qualtRics}` package.

You will need an **API key**.

- In your UBC Qualtrics Account, go to **Account Settings** -> **Qualtrics IDs** tab -> on the left under the API table click **Generate Token**. Copy the **Token** and paste it in the code below under `api_key`.

You will need the **survey id** of your survey.

- In your UBC Qualtrics Account, go to **Account Settings** -> **Qualtrics IDs** tab -> on the right under the *Surveys* table copy the code beside the name of your survey and paste it in the code below under `SurveyID`. On my end, my surveys all start with `SV_`, but I'm not sure if that's true for all users.

```r
pacman::p_load(qualtRics)

# Establish API
# qualtrics_api_credentials(api_key = "insert_code_here",
#                            base_url = "ubc.ca1.qualtrics.com",
#                            install = TRUE)
#
# Load your survey data to a dataframe called rawsurveydata
# rawsurveydata <- fetch_survey(surveyID = "insert_code_here")
```