To get **the raw data**, I asked my client, a coffee shop, to generate 448 \*.csv files via a cloud-based cash register system they use.[[1]](#footnote-1) Each file contains all orders made on one calendar day. All files follow the same format.

**My goal** was to put the data from all the 448 files into a single data frame. Since my analysis concerns drinks, I would only keep the data on drinks ordered at the café, discarding the data on food and services.

To write **the initial wrangling procedure**, I used readLines() to extract lines from one file. I would feed one of the \*.csv files to my script, making a character vector with a sequence of order numbers and menu items ordered.

After examining printouts of this vector for different files, I realized **I was facing four big problems**:

1. **Double-booked identification numbers**. Most menu items at the café can be ordered with different options, or modifiers. For example, a large latte can be made with regular or soy milk. When exporting to \*.csv, the cash register does not differentiate between menu items (4 refers to “large latte") and their modifiers (4 also refers to “soy milk"). This means that *the same number can refer to two objects, each on a different level of the item-modifier hierarchy*. Further, the raw data *does not tell me at which level of hierarchy each object is*.
2. **Several number-descriptor combinations referring to the same item**. Some menu items in the data are typos―a result of human error when maintaining the menu in the cash register system.[[2]](#footnote-2) Also, promotional versions of regular items were recorded as separate items.[[3]](#footnote-3)
3. **An order can contain any number of items more than zero.** The vector of lines extracted from a file would contain a sequence of order numbers and items ordered. Without a point of reference, it wouldn’t be possible to tell which vector element is a menu item and which is a modifier to a menu item.
4. **Most drinks could be modified in many ways**. For instance, a large latte could be “iced” or made with “specialty milk,” but usually it’s only made with “espresso” and “regular milk.”

**I solved these problems in four steps**, each undertaken in a separate R script:

1. [Learn what’s on the menu](https://github.com/friendelectric/Coffee-Weather/blob/master/1%20extract%20menu%20items%20and%20modifiers.R).I first extracted all unique pairs of identification numbers and descriptors from all files. Then, I manually coded each ID-descriptor pair according to its type (drink, food, or service), position in the menu-modifier hierarchy (noting whether it’s a standalone menu item, like a latte, or an option to a menu item, like soy milk), and correct spelling (if it was a typo or a promotional item). I would use this *reference table* later to differentiate among elements within a vector extracted from a raw \*.csv file.
2. [Extract data on all orders completed at the café into a data frame](https://github.com/friendelectric/Coffee-Weather/blob/master/2%20extract%20orders.R). The *data frame of orders* includes ID number for the order, day, HH:MM, and the contents of the order (stored as a string).
3. Using the reference table,[discard unnecessary items (food and services)](https://github.com/friendelectric/Coffee-Weather/blob/master/3%20discard%20foods%20and%20services.R) from the data frame of orders.
4. [Build a table of hourly counts of sales, by ingredient/trait of drink](https://github.com/friendelectric/Coffee-Weather/blob/master/4%20hourly%20counts.R).I achieved this over several stages:
   1. I first expanded my reference table into a *coding book*. The coding book contains codes for every variable that I could potentially use in my statistical analysis. Most variables code TRUE or FALSE on the presence of a given ingredient or trait in a drink. For instance, a latte is coded TRUE on espresso, milk, and froth.
   2. Then, I parsed my *data frame of orders* made at the café (one observation is an order that may contain several drinks), making it into a *data frame of drinks* (one observation is one drink: the base and modifiers).
   3. I made sure all items were recorded correctly and replaced typos and promos with their original names.
   4. Then, I joined the data frame of drinks with the coding book, adding the variables for coding the drinks on traits and ingredients.
   5. Having the data (in string form) on what modifiers were used and the default codes of base drinks, I overwrote the default codes with relevant modifiers’ codes. For instance, having your latte “decaf” would cancel out the “espresso,” or having it “iced” would cancel out “frothed.”
   6. Finally, I cleaned the resulting data frame, counted tallies for hourly sales of each ingredient and trait, and joined this final data frame with a data frame on hourly weather data in the client’s location.

In **the final dataset**, each observation is *one hour of the café’s operation*. Dependent variables are hourly counts of sales of drinks that adhered to the relevant traits and ingredients. Independent variables are hourly weather condition variables. The dataset is available [here](https://github.com/friendelectric/Coffee-Weather/blob/master/CafeHourly.csv).

1. I worked with proprietary data, so I cannot share any of the raw files or those used during the wrangling stage, as they still contain pieces of information that could unmistakably identify my client. [↑](#footnote-ref-1)
2. For example, "CappucciNNo" is a typo: it should be recorded as "CappucciNo." Still, because of the typo, it appears that a separate item called "CappucciNNo" was sold for a few days. [↑](#footnote-ref-2)
3. For instance, "ESPRESSO PROMO!!" isn’t a separate item and would need to be recorded as the rest of "Espresso". [↑](#footnote-ref-3)