THE MELBOURNE DECONSTRUCTION

Capstone Project Data Story

February 21, 2018

INTRODUCING THE PROBLEM

In my capstone project, I wanted to answer the following **research question**:

> At my local coffee shop, are **sales of drinks** influenced by **weather**?

The shop's owner was excited to see if that is the case and provided me with daily sales reports in **.csv* format—448 of them.[1]

These reports were generated by a cloud-based cash register system.

The system records sales of items like "latte," "chai latte," or "dirty chai"—names that the manager chose.

Third wave coffee aficionados aside,[2] most readers won't be able to explain what makes these drinks different. (And we're not even going in too deep yet.)

Well, the data I received couldn't do that either.

> The client supplied me with daily sales reports counting unclear categories.
>
> *How could I reinterpret the data to make it useful?*

I wanted to know if there's a relationship between what drinks the café sells and the weather conditions outside.

But what's the worth in knowing that, say, increased *pressure* and *temperature* are associated with *more sales of "lattes"* or *"mochas"*?

Knowing the answer wouldn't deliver much value to the client in terms of logistics or insight into consumer behavior.

Further, interpreting the results would be a nightmare. If we see a trend for "double-shot lattes," but not regular "lattes," is it the extra shot of espresso that matters? What if we don't see this same trend for "double-shot espressos," which also have the extra shot but nothing else?

The problem is that **by default, a drink is just a name**.

It's an "americano," a "latte," a "cappuccino," or a "macchiato."

It has to be made meaningful, and the way to do it is to learn *what it's made of* and *how*.

> To produce actionable insights, I'm not thinking in terms of *drinks*.
>
> I'm thinking in terms of their *ingredients and traits*.

[1] To respect the conditions of my non-disclosure agreement with the owner, I will not name the business. This means that I will not be able to share or publish the raw files, as they contain identifying information.

[2] The client is part of the third wave coffee movement in the coffee industry: an independent-minded, artisanal reaction to the corporate second wave of the likes of Starbucks. The third wave coffee market treats coffee as a premium product defined by attention to detail in all stages of the process, from sourcing fair-trade beans to roasting to preparing quality espresso. By definition, these small businesses do not have access to big data that second-wave chains like Starbucks have. However, lately, the third wave industry has attracted large-scale investments. Putting aside the question of whether an overgrown third-wave coffee company can still be considered third wave, any third-wave coffee business will benefit from analyzing data available to it to thwart competition from both big chains and other small businesses.

Then, **a drink becomes a unique combination of ingredients**—espresso shots, portions of regular or specialty (almond, coconut, soy) milk, tea concentrates (chai or matcha), or smidges of chocolate. And don't forget **preparation techniques** (frothing to varying degrees) and **serving methods** (chilled or on ice).

Immediately, there's more precision to our thinking.

> This approach fundamentally changes how sales should be recorded.

Let's take the example of recording a sale for a drink called "*dirty chai*."

In my client's cash register system, it's recorded as a menu item of the same name.

On the menu, there's also an item called "*chai latte*."

By just looking at the names, we wouldn't be able to tell that a "*dirty chai*" is simply a "*chai latte*" with a shot of espresso.

> But if we were to forget *names* and think of *ingredients and traits* instead, a recorded sale in the abstract category "*dirty chai*" would become a sale of *an espresso shot, a portion of chai syrup, and a portion of frothed milk*, all taken separately.

Remember "deconstructed coffee" from Australia that took the Internet by storm a couple of years ago?

A Melbourne coffee shop became infamous for serving lattes like this:

In the world of "deconstructed coffee," the only way to say "thanks a latte" is passive-aggressive.

It doesn't appear so at first, but **this "deconstructive" approach is very useful**.

> While this is no way to serve a product to a client, this is a perfect illustration of what the product looks like to the café's manager.

Logistically speaking, every *ingredient* should be stocked, and machines should be operational to make sure the drink has proper *traits*.

For example, take ~~what was supposed to be a~~ the long macchiato above.

Normally, it consists of *an espresso shot*, made longer with *hot water*, and *milk—the ingredients*.

The milk is also *frothed*—that's *a trait*.

The manager will stock coffee beans, which become espresso after passing through the grinder and the espresso machine.

They will also restock the milk and make sure the frothing nozzle is always operational.

Aesthetics aside, the Melbourne barista's deconstructive approach was the approach of a responsible manager.

It took into account both *the logistics of the business* and *the highly customizable nature of the product*.

I took the same approach to my dataset.

I was *not* concerned with how many *"long macchiatos"* the café sold.

I wanted to know how many *espresso shots and milk portions* went into the drinks sold.

I'll call this approach ***The Melbourne Deconstruction***.

This approach, where a drink is understood not as an entity representative of *one* category, but a unique product coded according to *several* defining features—ingredients and traits—would

- **help my data speak the language of my client's logistics** and deliver value beyond what is offered by their cash register system, which already tracks sales of drinks, but not the ingredients,
- **streamline the interpretation of results at the modeling stage**, and
- **make it easier to discuss my data with those unfamiliar with third wave coffee culture**.

It's good to know that something that was made fun of on the Internet can be used to our advantage in a data science project.

It was the perfect target until it became the perfect weapon.

DATA WRANGLING

PLANNING THE DATASET

To get **the raw sales data**, I asked my client to generate *.csv files via the cloud-based cash register system they use.

The client exported 448 files.

Each of the files detailed all orders made at the coffee shop in one calendar day.

My goal was to pull the sales data from all 448 files into a single data frame.

Then, I would combine it with hourly weather data.

Originally, I proposed that I would use *daily* weather data, combined with *daily* sales.

However, after giving it some more thought, I decided to create a dataset of *hourly* sales combined with *hourly* weather data.

My reason for this was threefold:

- **Unlike the weather, the café does not operate 24/7.** It's closed for at least ten hours every day, but the weather is still out there. The gaps in time would make the daily resolution weather data unrepresentative of the daily sales.

- **Hourly data is better suited for an immediate snapshot of the situation.** When I think of getting a coffee, I'm not thinking "what is the weather going to be like today, on average?" I'm thinking, "is it nice enough out to go grab a coffee, right now?" The hourly resolution is more representative of the consumer's decision making.
- **It would make for more than ten times as many observations.**

WRANGLING CAFÉ DATA

Since I cannot share the files I obtained from the client, I made a file illustrating the formatting of the raw data.[3]

Looking at **the raw data's formatting**, we see that

- the first lines always contain the name of the business and the calendar day for which the file details recorded orders,
- a section with reference information follows, explaining the fields that detail order information, and
- orders follow until the end of the file, separated by a series of dashes.

And **for every recorded order, there is**

- *General information*: order number and time
- *Price and payment information*: order sub-total before tax, calculated sales tax charges, taxable order total, order total for tax exempt orders, order total, payment type—cash or debit, device used to record the order
- *Information on items ordered*:
  - ID of menu item, name of the menu item, and price (e.g., "4, Large Latte, 2.83")
  - ID of menu item modifier, name of the modifier, and (not in all cases) price (e.g., "3, Extra shot, 0.48" or "1, Milk option: regular")

  Since my analysis would concern drinks only, I would need to discard the data on food and services.

  I would also discard the data on prices and payment.

To write the initial wrangling procedure, I used *readLines()* to extract lines from one file.[4]

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 three big problems**:

1. **Most drinks could be modified in many ways. When looking at a record of the order, there was no way to tell what's a menu item, and what is an option to a menu item.** Most menu items at the café can be ordered with different options, or modifiers. For example, a large latte can be made with an extra shot of espresso, or with specialty milk to replace regular milk. There could be any number of options recorded. Other than knowing that the first item in an order would not be a modifier, a script could not gauge by looking at the data which level of hierarchy each object is. The only unique thing about modifiers is that some of them don't have prices, but most do. When I extracted all the ID-name pairs from several files, I concluded that there was also no regularity to the identification numbers. They appear to have been in the order in which the café's manager recorded them into the system.

[3] The raw files I worked with were in French. The names of items were changed. "$$$" replaces price information. The file is abridged to show what the beginning and end of a raw file would look like.

[4] A later version of this procedure can be found in a function here.

2. **Identification numbers were double-booked, referring to both menu items and options.** For example, 4 would refer to "large latte" and also to "soy milk." This means that the same number can refer to two objects, each on a different level of the item-modifier hierarchy.

3. **At least several number-descriptor combinations—typos and promos—referred to the same menu item.** Some menu items in the data are typos—a result of human error when maintaining the menu in the cash register system. For example, "CappucciNNo" is a typo: it should be recorded as "CappucciNo." Still, because of the typo, it appears like a separate item called "CappucciNNo" was sold for a few days. Also, promotional versions of regular items were recorded as separate items. For instance, "ESPRESSO PROMO!!" isn't a separate item and would need to be recorded as the rest of "espressos."

My solution to the problem of discerning between menu items and modifiers, as well as typos and promos, was to create *a reference table* where every ID-descriptor pair is coded according to its type and position in the menu-modifier hierarchy.

I would then expand the reference table into *a coding book*.

The coding book would be the key to performing *the Melbourne Deconstruction*: recording each drink sale as a unique combination of ingredients and traits.

I wrangled the client-supplied data in four steps, each implemented in a separate R script:

1. Learn what's on the menu and put it into a reference table. I first extracted all unique pairs of identification numbers and descriptors from all 448 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. 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) from the data frame of orders. After this was done, I expanded my reference table into a coding book. The coding book contains codes for every feature 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.

4. Build a table of hourly counts of sales by drink feature.

The final script was particularly complex and included several stages:

1. 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 menu item and modifiers).

2. I replaced typos and promos with the menu items' original names.

3. Then, I joined the data frame of drinks with the coding book, adding the variables for coding the drinks on traits and ingredients.

4. 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."

5. Finally, I cleaned the resulting data frame, counted tallies for hourly sales of each ingredient and trait, wrangled the weather data for the client's location, and joined it with the dataset of drink sales.

WRANGLING WEATHER DATA

To obtain the weather data, I initially tried using the weatherData package, which pulls data from wunderground.com. However, the website changed the URL directory structure that the package relied on. When I was trying to pull the data, wunderground.com responded with "bad request," and the issue wasn't fixed yet.

I considered other R packages, but most of them seemed to require getting API keys from the weather data providers they connected to.

This seemed unnecessarily time-consuming, compared to the option I proposed initially—using government-supplied data from Climate Canada.

The agency had *.csv files available for download with no special arrangements needed.

For the weather station in my client's location, reports were available at daily and hourly resolution.

For my purposes, downloading government data proved to be faster and more straightforward than using packages.

The wrangling procedures I performed on the weather data were less ~~nightmare-inducing~~ exciting than what I had to do for the sales data, but they can be found in my last wrangling script.

DIVING INTO THE DATASET

OVERVIEW

The dataset can be downloaded here.

Before going deeper into the dataset, let's look at its structure:

```
str(dataset)
```

```
## 'data.frame':    5667 obs. of  29 variables:
##  $ Day                  : Date, format: "2016-09-01" "2016-09-01" ...
##  $ Hour                 : int  10 11 12 13 14 15 17 18 19 7 ...
##  $ Season               : Factor w/ 4 levels "Fall","Spring",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ ExamPeriod           : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ LowSalesPeriod       : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ DrinksSold           : int  8 7 4 7 13 6 3 3 4 4 ...
##  $ Size.Mean            : int  281 253 296 233 260 154 434 394 384 296 ...
##  $ Content.Espresso     : int  6 5 2 6 9 3 0 1 1 4 ...
##  $ Content.Drip         : int  0 0 1 1 4 1 0 1 1 0 ...
##  $ Content.Water        : int  2 1 1 1 0 0 2 1 1 0 ...
##  $ Content.Tea          : int  3 1 1 0 0 0 3 1 2 0 ...
##  $ Content.Milk         : int  6 5 3 3 5 2 0 1 1 3 ...
##  $ Content.SpecialtyMilk: int  0 1 0 1 2 0 1 0 1 1 ...
##  $ Content.Chocolate    : int  1 0 0 0 1 1 0 0 0 0 ...
##  $ Content.Seasonal     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Content.Juice        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Trait.Cold           : int  0 0 1 0 1 0 1 0 1 0 ...
##  $ Trait.HighInSugar    : int  1 0 0 0 0 1 1 0 1 0 ...
##  $ Trait.HighInCaffeine : int  0 0 1 1 4 1 0 1 1 0 ...
##  $ Trait.Froth          : int  7 6 4 6 13 5 3 3 3 4 ...
##  $ Temperature          : num  20.5 20.4 21.5 21 21.1 21.9 21.2 21.5 20.7 16.7 ...
##  $ DewPoint             : num  12.6 12.2 12.2 14 13.2 13.1 13.2 13.3 14.5 11.9 ...
##  $ Humidity             : int  60 59 55 64 60 57 60 59 67 73 ...
```

```
##  $ WindSpeed            : int  16 24 16 29 27 7 14 12 25 20 ...
##  $ Pressure             : num  101 101 101 101 101 ...
##  $ Clear                : logi  TRUE TRUE TRUE FALSE FALSE FALSE ...
##  $ Fog                  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Rain                 : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ Snow                 : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
```

Every observation is *one hour of the cafe's operations.*

Each of the *Content.** and *Trait.** variables represents the number of drinks sold that had the noted ingredient or trait.

For example, an observation can be partially spelled out like this:

```
On September 1st, 2016, from 10:00 to 10:59AM,
the cafe sold 6 espresso-based drinks,
6 drinks containing milk, 1 drink containing chocolate,
0 drinks that were chilled or iced, and 7 frothed drinks.
During that time, the sky was clear, there were no fog, rain, or snow,
and the temperature was at 20.5 degrees C.
```

The next two sections describe the dependent and independent variables in detail.

DEPENDENT VARIABLES

All dependent variables other than *DrinksSold* and *Size.Mean* describe the number of drinks adhering to the noted feature that were sold during the hour of observation.

The following table provides context on the coding process, noting examples of drinks that were coded as TRUE on the relevant features (ingredients and traits).

Variable	Description
DrinksSold	Total number of drinks sold.
Size.Mean	Mean volume of drinks sold, in milliliters.
Content.Water	Water understood as a separate ingredient added to the drink (e.g., americanos are made by pouring an espresso shot in a cup of hot water). This excludes water going through an espresso machine, filter pot, chemex or other pour over devices.
Content.Tea	Tea leaves, chai, and matcha, as well as kombucha (fermented tea).
Content.RegularMilk	Regular milk is the default option for making espresso-based beverages (e.g., lattes, cappuccinos). In a hot drink, the milk is frothed. In a cold drink, it's used as is.
Content.SpecialtyMilk	For an additional price, clients have the option to substitute regular milk for specialty milks (coconut, almond, or soy). Appeals particularly to the lactose intolerant. However, many others prefer the taste of a certain specialty milk to the regular milk.
Content.Chocolate	Found in hot chocolates (which have quite a large portion of chocolate) and mochas (lattes with a smidge of chocolate on the bottom to offset the bitterness of espresso).
Content.Seasonal	Seasonal ingredients added to standard drinks (e.g., latte), like pumpkin spice (fall) or mint (winter).
Content.Juice	Bottled juices.
Trait.HighInSugar	Drinks high in sugar, compared to other drinks offered at the shop. Primarily, drinks with chai syrup.

Variable	Description
Trait.HighInCaffeine	Drinks with extra espresso shots or filtered/drip/slow bar coffee. For a discussion on the concentration of caffeine in espresso vs. drip coffee, see this post.
Trait.Cold	Drinks served on ice, as well as cold brew coffee or any bottled or canned drinks served from the fridge.
Trait.Froth	Hot drinks with milk, prepared using a frothing nozzle on an espresso machine.

INDEPENDENT VARIABLES

The dataset contains the following weather variables:

- Temperature, degrees Celsius
- Dew Point temperature, degrees Celsius
- Relative humidity, percent
- Wind speed, km/h
- Pressure, kilopascals
- Binaries for atmospheric phenomena and sky conditions—rain, snow, fog, and cloudy

IMPLICATIONS

The dataset can have implications to several areas of the client's day-to-day operations:

- **Stock management.** Having insight into the effects of weather variables on product demand, the client can adjust stock management. For example, they could order less milk if they expect less demand for milk-based beverages given tomorrow's weather forecast.
- **Product development.** Knowing how weather conditions influence beverage preferences, the client may decide to adjust the current product selection or create new products. To illustrate, knowing that there is a better chance of selling a chocolaty beverage when it rains may lead to developing a special drink containing chocolate only sold on rainy days.
- **Marketing strategies.** Aware of just which products customers want to buy in certain weather conditions, the client may adjust what—and how—they market. Weather-based marketing won't just involve deciding what witty message will work well on that sidewalk sign outside the door. For instance, the client may want to collect their customers' email addresses to produce weather-trigged marketing campaigns or send out a timely call to action on the business' social media accounts.

Importantly, **the dataset can only be used to answer research questions about my client's business, and not the third wave café market or the coffee industry in general**.

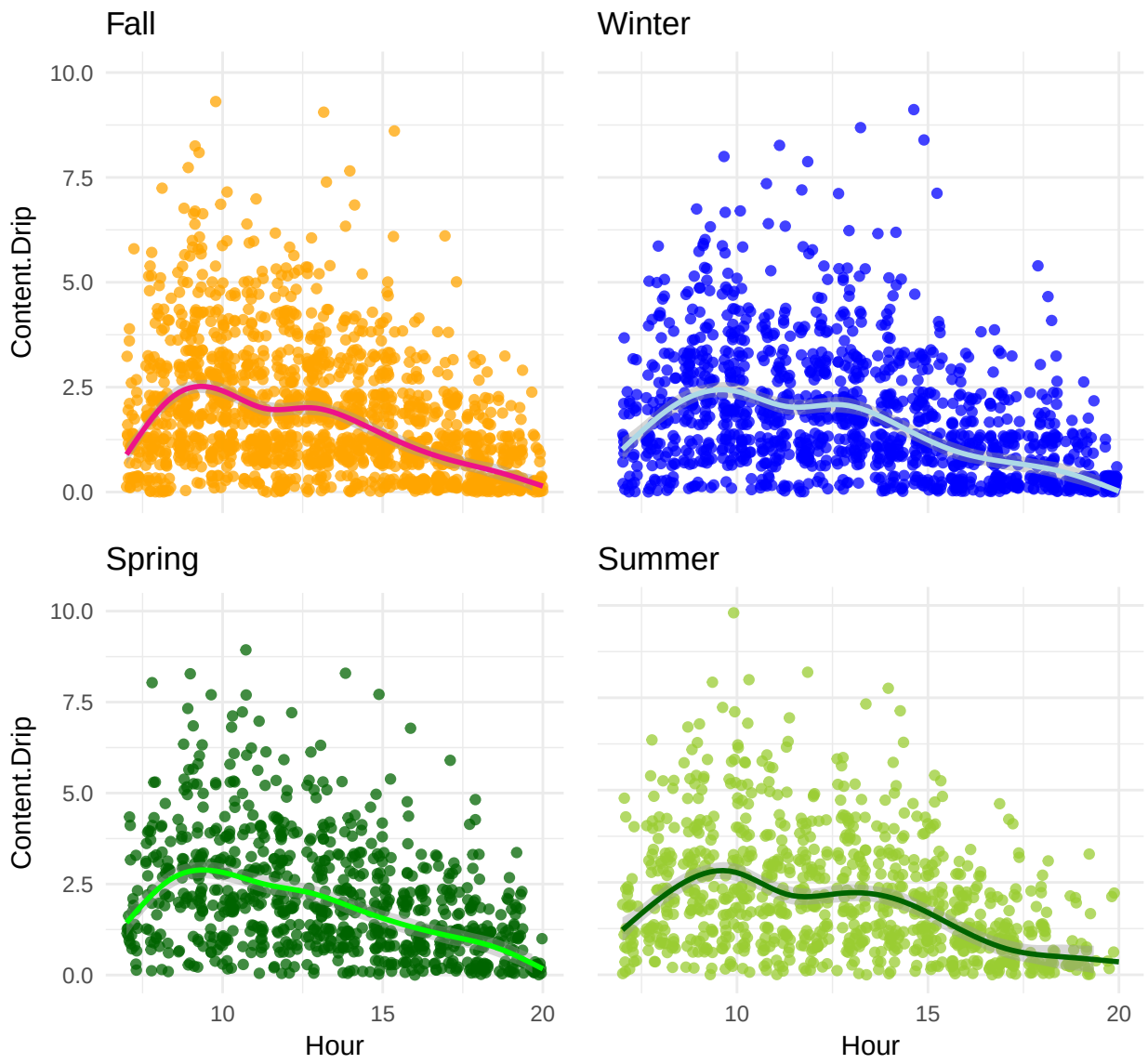
PRELIMINARY EXPLORATION

While my inquiry will be focused on the relationship between weather and sales counts, the dataset can be useful to my client in other ways.

My dataset is cleaner and more nuanced than what the client has access to via their cash register system.

It may let us spot trends that the client may not have been aware of.

For instance, this plot shows the number of filtered coffee drinks sold each hour, by season:

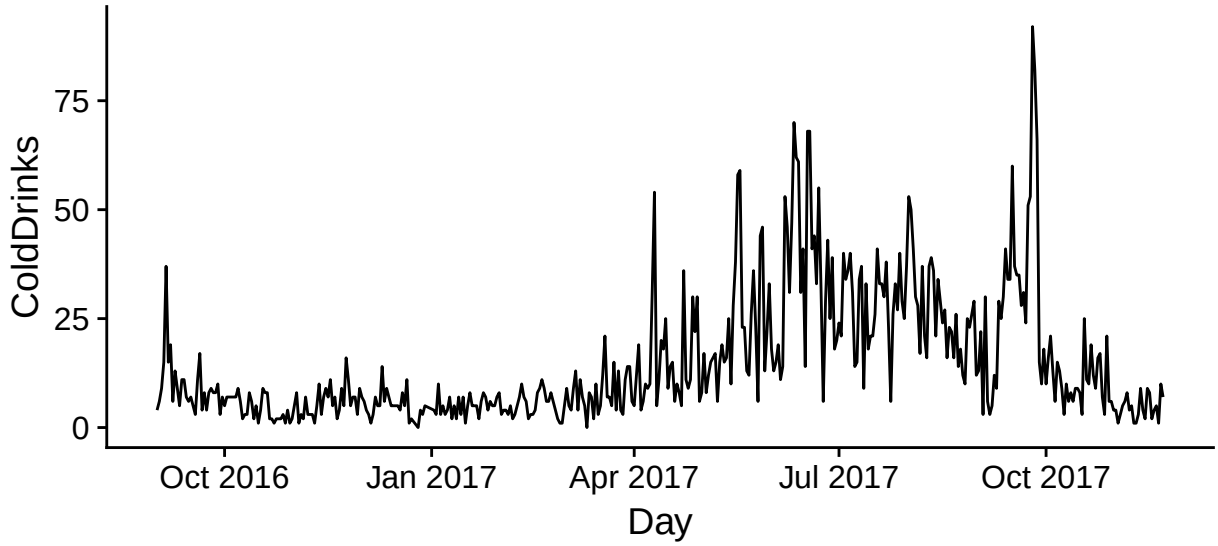


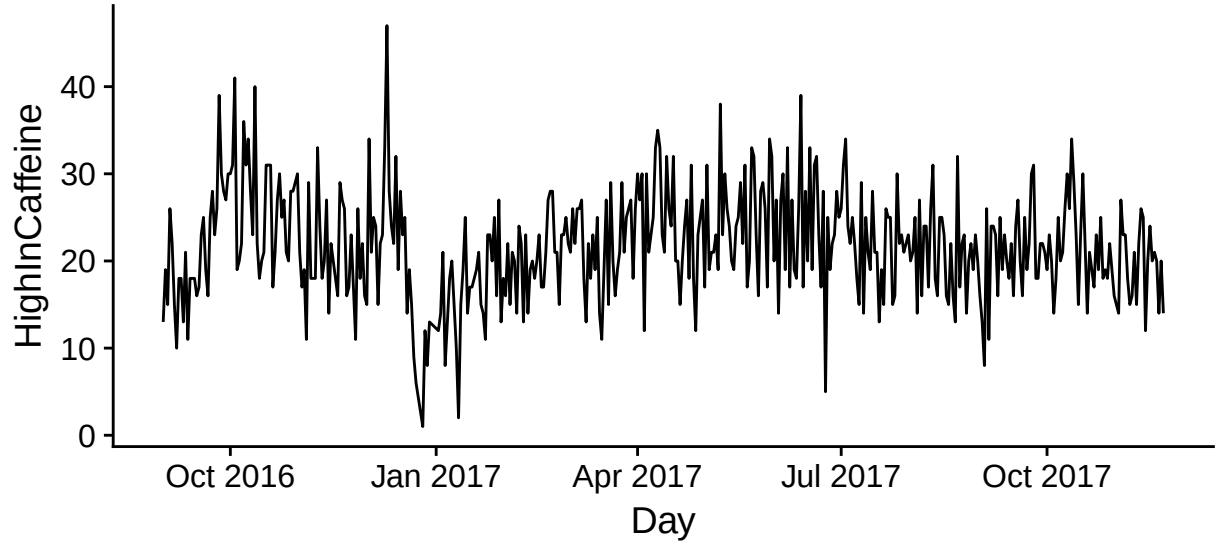
Judging by this, it's probably not a good idea to start brewing a pot of filtered coffee after 5PM, unless there are orders coming in—particularly in the summertime.

Finally, **if needed, the dataset can be rescaled to the daily resolution**, providing a further dimension of insight for the client.

```
daily <- dataset %>% group_by(Day) %>%
  summarise(sum(Trait.Cold), sum(Trait.HighInCaffeine), mean(Temperature))
```

For example, we could plot daily **time series** for drinks with specific features, like *cold* or *high in caffeine*:





These are just some possibilities of what can be done with the dataset besides analyzing the impact of weather variables!