The Inspiration

It all started, predictably, over some drinks. My good old buddy Noah Bloom, just your average compost farmer with a Philosophy degree, told me he was considering learning about coding. I told him he should learn Python because it was fairly easy and you can do almost anything with it. When I relayed this story to a DevOps engineer at my work later, he laughed and said "such a Data Scientist thing to recommend!" I stand by it. For one, I *am* a Data Scientist. And for two, *I'm not wrong!* Python *is* pretty easy and it *is* used for almost anything. But I digress...

In an effort to condense roughly 18 months of emails and foggy, semi-medicated conversations into several sentences, he learned Python. And also some JavaScript. And some SQL. And way more about the difference between Linux distro's than I ever cared to know. And somewhere along the line he decided to apply his newfound skill set to a project that would truly be a benefit to his fellow man: a cocktail app.

And so drinkBase was conceived. The idea was simple: a database for hip and classic cocktails that could be searched like any other database. "I have the following three liquors, but I don't have these two. What can I make?" Or more specifically, "What can I make with gin and Green Chartreuse that *doesn't* have lime juice in it?" (You'll have to click the link above to find out. Spoiler alert: there's only two drinks. And they're both delicious.)

You might be shocked to hear this doesn't already exist on the internet, but if it does, neither of us could find it. So like any good digital tinkerer would, he decided to build it.

The Vision

I don't remember when (I think it was in the spring of 2018, somewhere between my second Boulevardier and my first Lucien Gaudin) I decided this database needed some data viz. The true inspiration was Dave Arnold's *Liquid Intelligence*, which I was gifted for Christmas by both my wife *and* my brother the preceding year. In many ways, Dave's approach to drinking and drink-making differs from own, but you gotta respect his devotion to his craft. And above all, his attention to detail.

[Liquid Intelligence graph]

Somewhere in the middle of the book there is a full two-page spread mapping the acid, sugar, and alcohol content of a few dozen cocktails (reprinted above without permission* above), complete with recipe and calculations for all of them. Using that--and the ingredient list on the preceding pages--as a starting point, I painstakingly recorded various statistics (the most important being sugar and alcohol content, as you'll later see) of every ingredient in every cocktail we'd collected. From there I was off and running.

The Viz

And so we had some data. Now for the viz.

Being trained in the Data Arts & Sciences, I instinctually reached in my toolbox. A simple bubble plot seemed the obvious choice, but we still had some number of features (between 3 and 256, depending

on how you cut them) and we needed to reduce them to two dimensions. We had a few other dimensions to mess with, but I decided to limit my options by assigning size of the bubble to the size of the drink (in oz.) and color to drink style (shaken, stirred, etc.). I tried a few other options for those two and couldn't find any contenders so those two were fixed for the rest of the experiments.

My former boss, Doug Bryan, instilled in me the precept that the most important thing is also *the data*, more so than the algorithm you feed it to. And so, when deciding how to encode my features, I sampled several paths. The amply fill out two dimensions, I would need some larger number of ingredient with which with to begin the distillation. I settled on three different recipe to examine.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from scipy.spatial.distance import cosine, pdist, squareform
from sklearn.manifold import MDS

import pandas as pd
import numpy as np


def combine(row):
    """
    Selection function, applied to each row below.
    """
    if X == 'all_text':
        picks = ['name', 'style', 'glass', 'ingredients']
    elif X == 'ingredients':
        picks = ['ingredients']
    elif X == 'prep':
        picks = ['name', 'style', 'glass']
    else:
        raise ValueError("must be 'all_text', 'ingredients' or 'prep'")
    return'|'.join([str(x.lower()).replace(' ', '|') for x in row[picks]])

def cluster_XY(df, X, mode):
    """
    X {str} - must be 'all_text', 'ingredients' or 'prep'
    mode {str} - must be 'pca' or 'tsne' or 'mds'
    """
    df['features'] = df.apply(lambda row: combine(row), axis=1)

    try:
        docs = [c.replace('|', ' ') for c in list(df['features'])]
    except AttributeError:
        print("FAILURE: df['ingredients'] doesn't exist. Or something else went wrong.")
        return None
    #
    vectorizer = CountVectorizer(max_df=0.5, min_df=0)
    vectors = vectorizer.fit_transform(docs)
    print("vocab shape: {}".format(vectors.shape))
    X = vectors.toarray()

        ...
```

The Prep: The simplest option seemed to the combination of the drink style (shaken, stirred, etc.-as mentioned above) and the glass in which it's served. This is how restaurants typically display their choices. One-hot encoding these two columns got me 16 features. If I added in the words from the drink name I got an additional 139, which got me to 155.

The Ingredients: My other obvious data points were the ingredients in the drink. One-hot encoding these got me to 122 features.
All Text: Combining everything above into one big bag-o'-words got us all the way to 256 features.

It crossed my mind into the considerably deeper and murkier waters of word embeddings. But that would've gotten me something on the order of 100 to 300 times as many features. This would

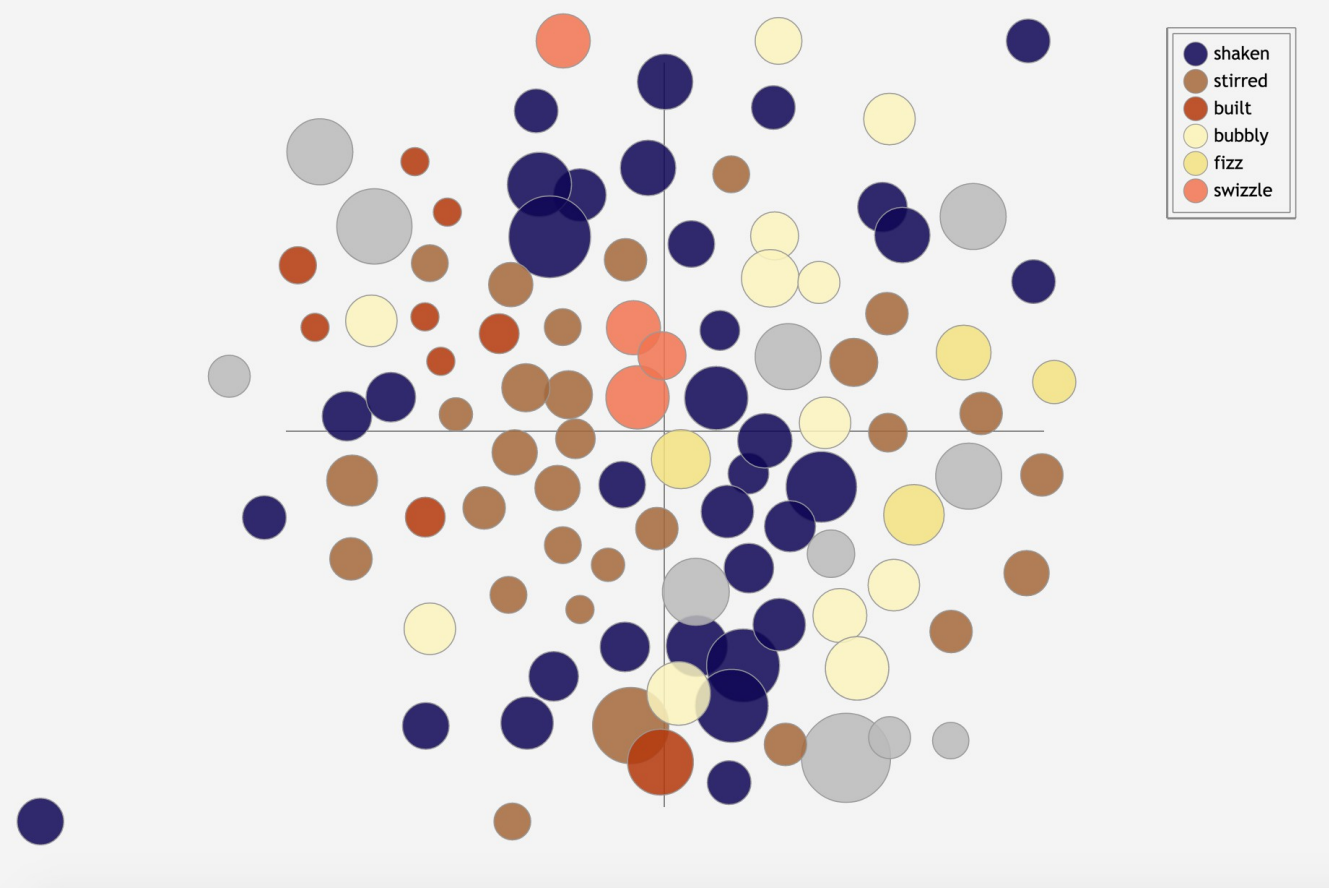
necessitate something a bit more nuanced, likely some family of neural nets, and now we were getting in deep waters indeed.

tSNE

Nonetheless, the first tool I reached for on the bench is frequent bedfellow of embeddings and neural nets. The t-distributed Stochastic Nearest Neighbor Embedding, known to its friends as t-SNE, has become widely popular in the past half decade or so.

```
        if mode == 'tsne':
        X_pca = PCA(n_components=50).fit(X).transform(X)
             X_tsne = TSNE().fit(X_pca)
             XY = pd.DataFrame(X_tsne.embedding_, columns=['x', 'y'])
             print("loaded t-sne data")

        ...
```

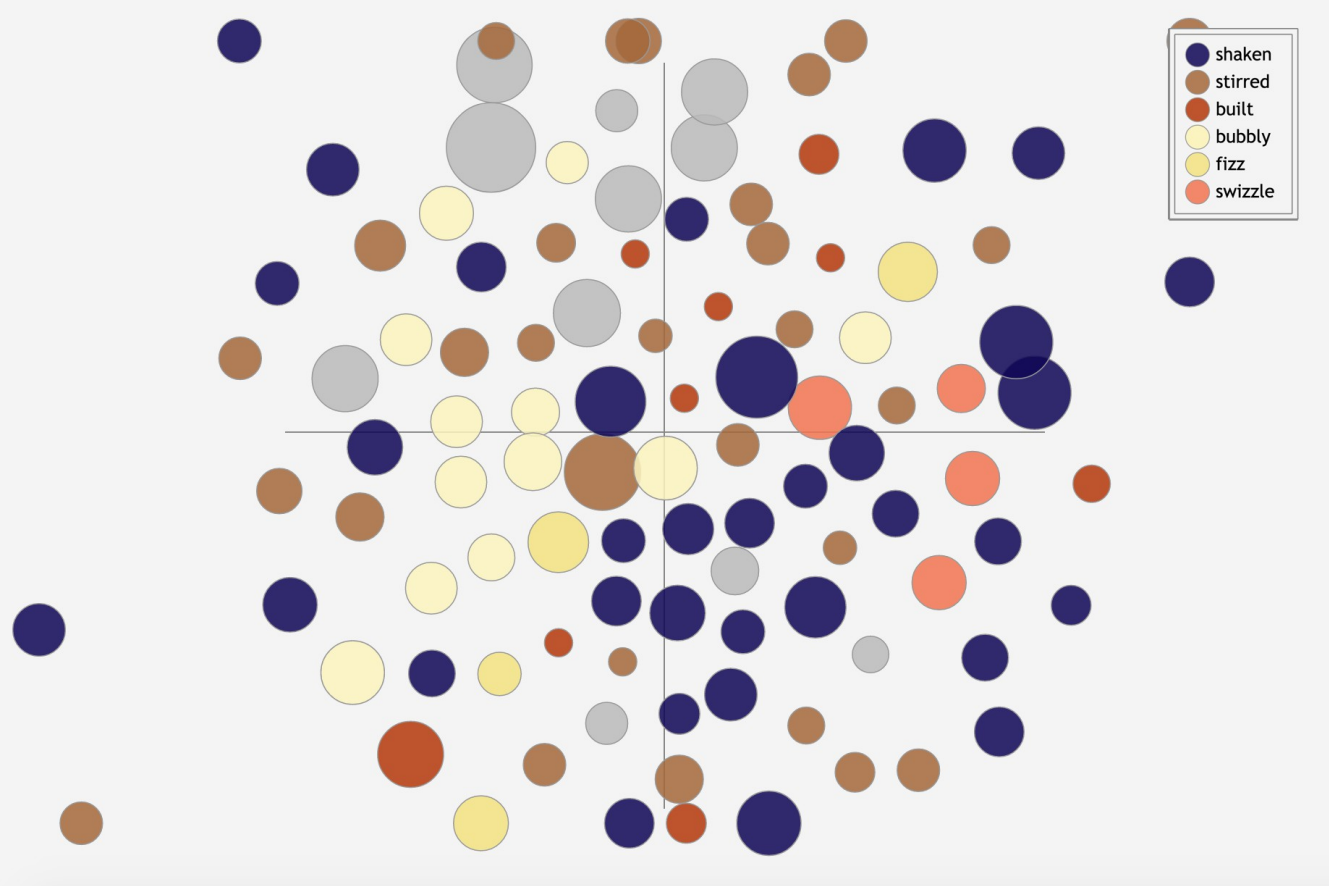
The knock against it is that, as my current boss Ania Alton pointed out, it's a whole 'nother algorithm in and off itself. Her meaning is that you have to spend time tuning it, on top of the tuning you're already doing on your features and anything else you're doing. This can make it unpredictable, but as shown below, it can also be a bit, shall we say, uniform.



With all 256 features we see, generally speaking, a large blob. Scanning about, it's hard to make much sense of it. We do see two curious outliers separated from the pack: in the bottom left sits the Brandy Alexander (cognac, heavy cream, simple syrup, grated nutmeg – shaken) while the dangerously

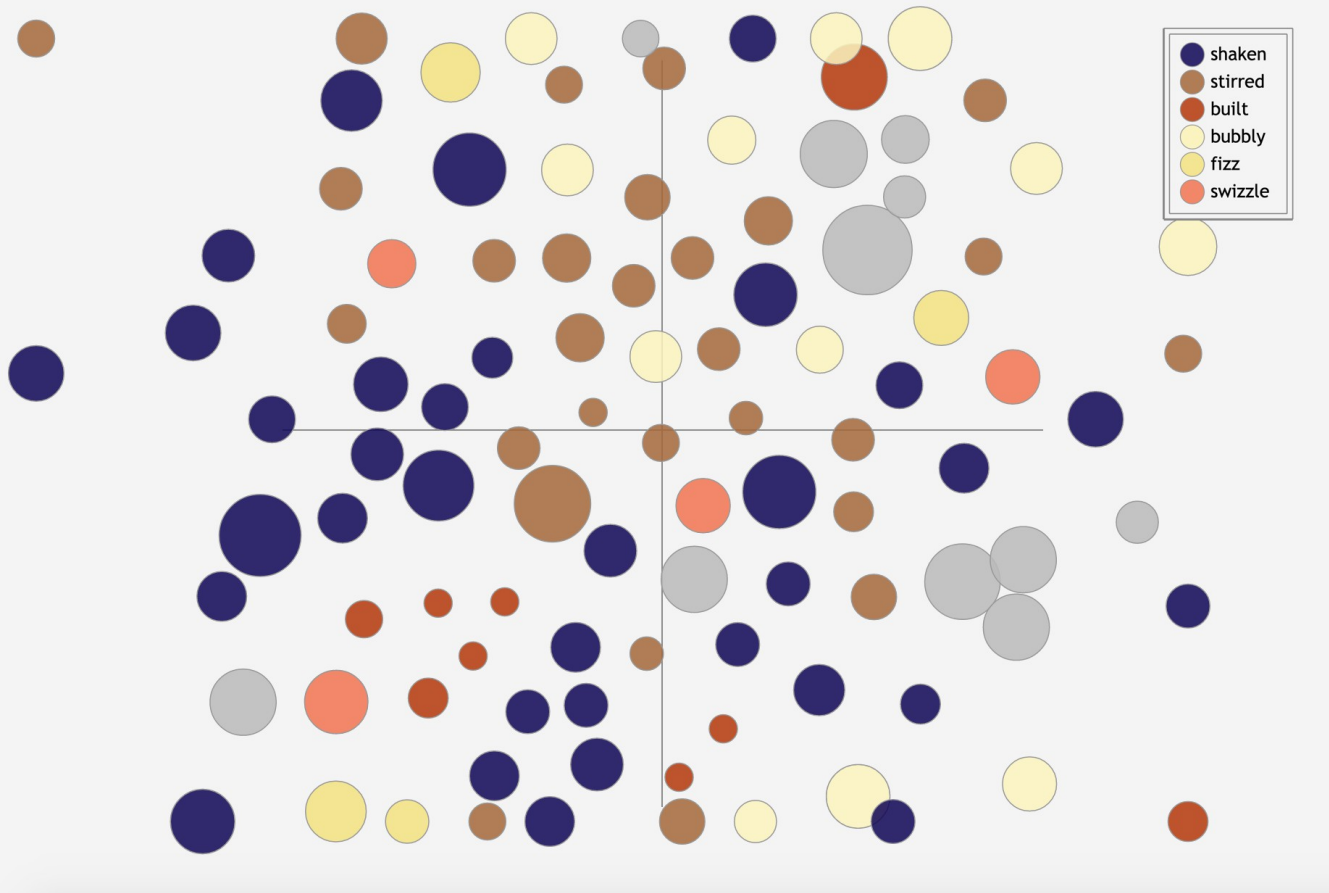
refreshing Corpse Reviver #2 (Gin, lemon juice, Cointreau, Lillet Blanc, absinthe, orange twist – shaken) takes up the opposite top right corner.

When we limit it to only the cocktail ingredients, things don't get much clearer.



But again we see a few distinguish themselves from the pack. The idiosyncratic Bitter Handshake (Fernet Branca, blood orange reduction, rye syrup, orange spiral – stirred) holds down the bottom left, with your classic but inventive Planter's Punch (dark rum, pineapple juice, lime juice, orange juice, simple syrup, exotic tiki garnish – shaken) just above it. In the top right is the Rob Roy (scotch, sweet vermouth, Angostura, lemon twist – stirred) a twist on an old standby.

If we restrict ourselves to only the “prep” features (style, glass, and drink name) things are even more scattered.



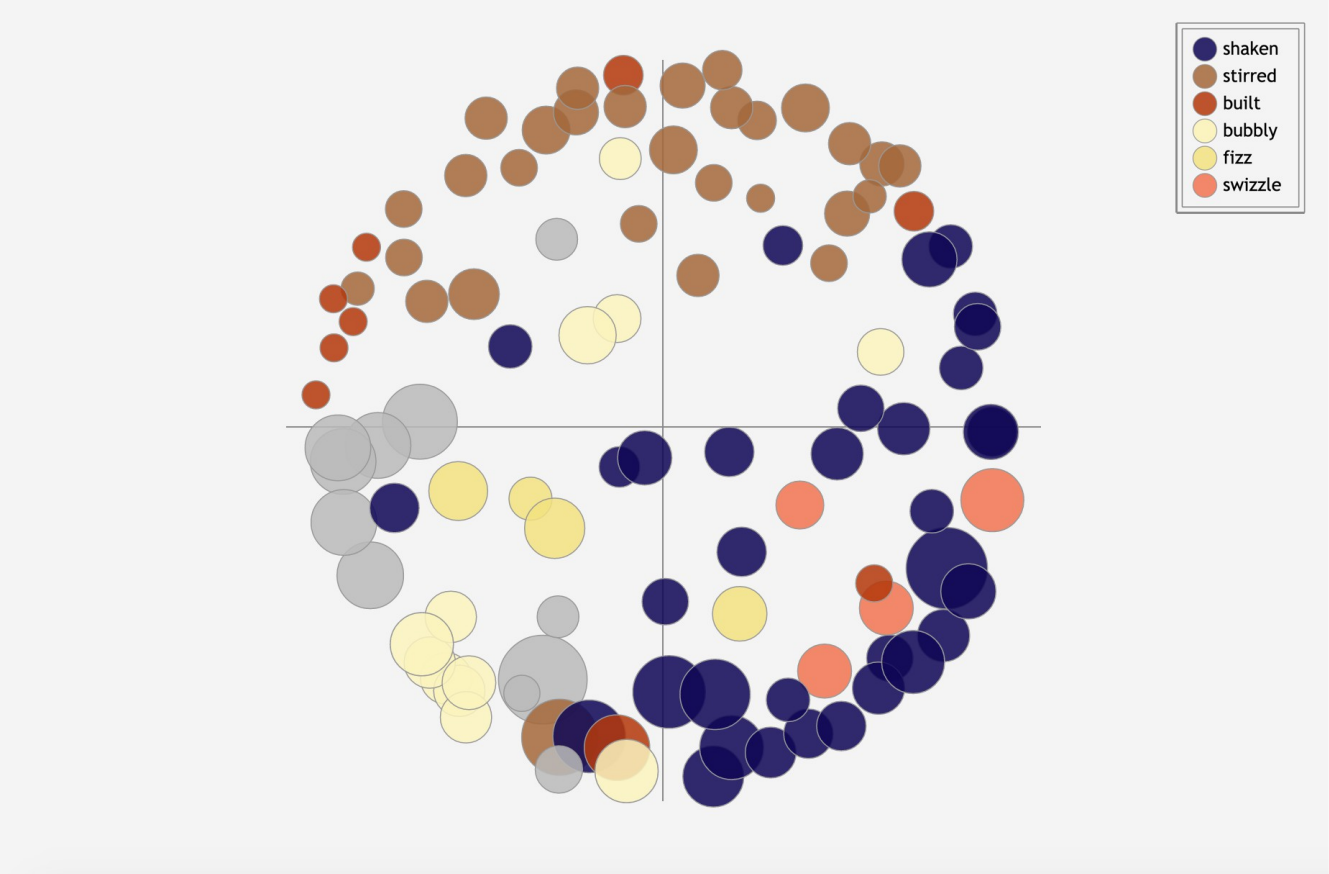
In the top left we find the rugged but elegant Rusty Nail (scotch & Drambuie)
and the bottom right is reserved for everybody's new best friend: the Negroni (gin, Campari, sweet vermouth). Still, the t-SNE plots show no real discernible pattern at all.

MDS

Classical Multidimensional Scaling (MDS) was the next to get a swing at it. As the name suggests, it a bit of a classic. Sort of the Manhattan for dimensionality reduction.

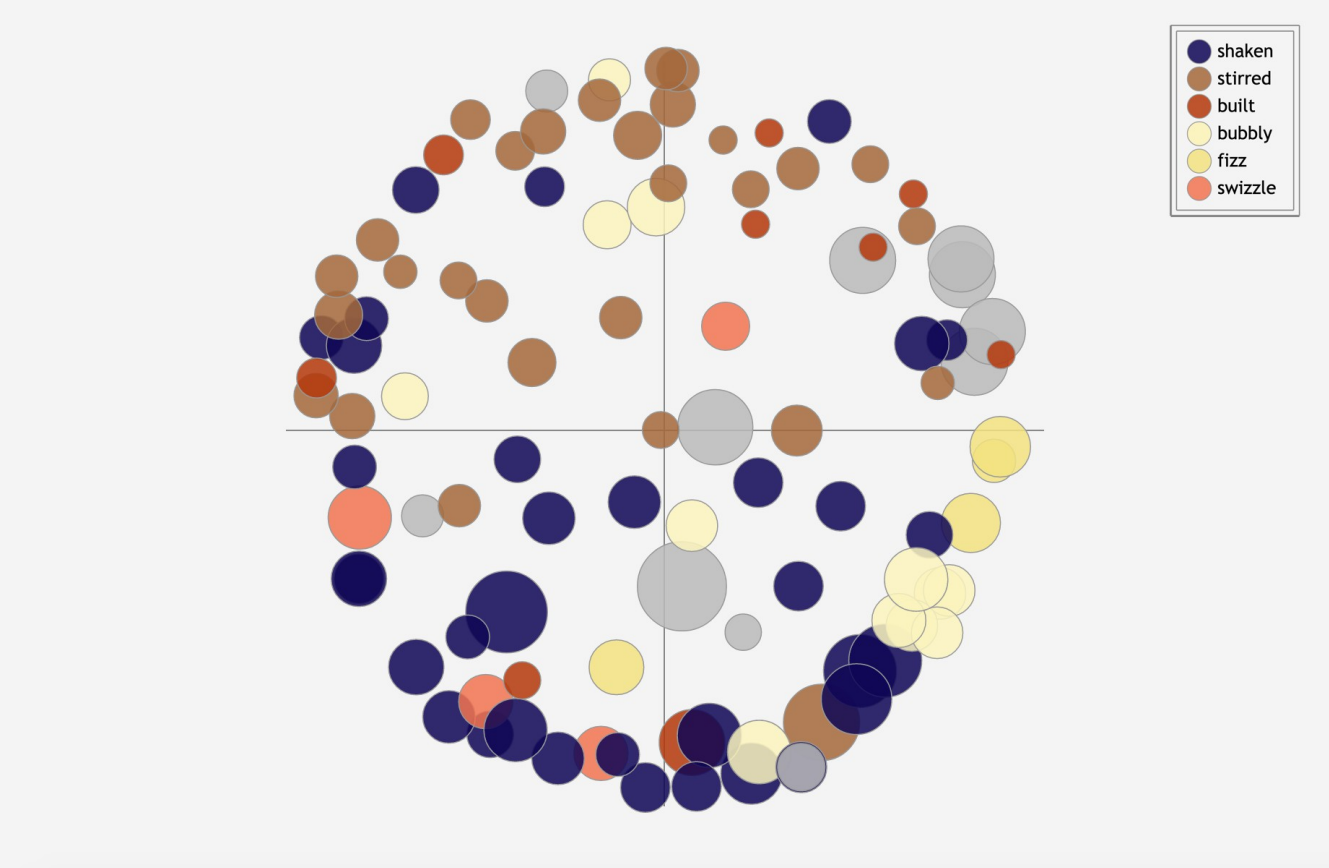
```
        ...

        elif mode == 'mds':
            X_pca = PCA(n_components=50).fit(X).transform(X)
            dists = pdist(X_pca, cosine)
            dist_matrix = pd.DataFrame(squareform(dists),
                                       columns=df['name'],
                                       index=df['name'])

            scaler = MDS(dissimilarity='precomputed', random_state=123)
            XY = pd.DataFrame(scaler.fit_transform(dist_matrix))
            XY.rename(index=str, columns={0:'x', 1:'y'}, inplace=True)
            print("loaded MDS data")

        ...
```

With all the features available, we see patterns right away.



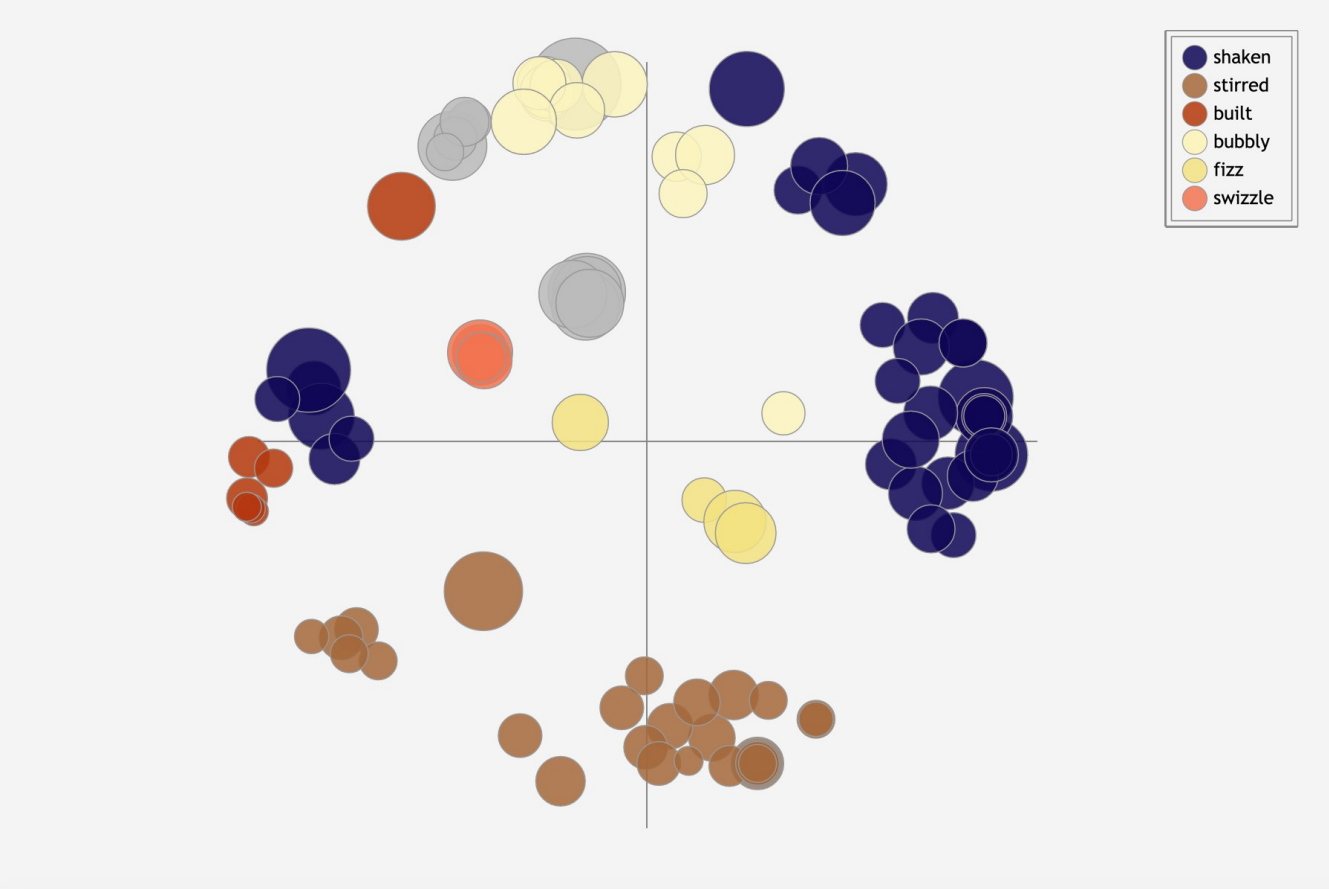
The plot becomes a neverending arc. We see a yin and yang of sophisticated sipping. The drinks are skillfully split by style (shaken, stirred, built, etc.) and then within that by primary ingredient (whisky, gin, rum, etc.). Things begin to get a bit more interesting.

If only ingredients are used for the clustering, it is predictably split up by ingredient, though generally *not* the primary liquor.



For example, the bottom crescent is all drinks with juice of some kind in them (some rum, some gin, a few whisky), while the top left all feature vermouth, fading to only sweet vermouth as you move closer to the top. All the beige and yellow on the right features bubbles of some kind, champagne above and egg whites below. A palette of tastes is laid before us. This is definitely my favorite so far.

Feeding only the prep features to MDS is fascinating for it's pure precision. Given only three categorical features--drink name, style, and the glass in which it's served--the MDS sensibly collapses them down to two dimensions.



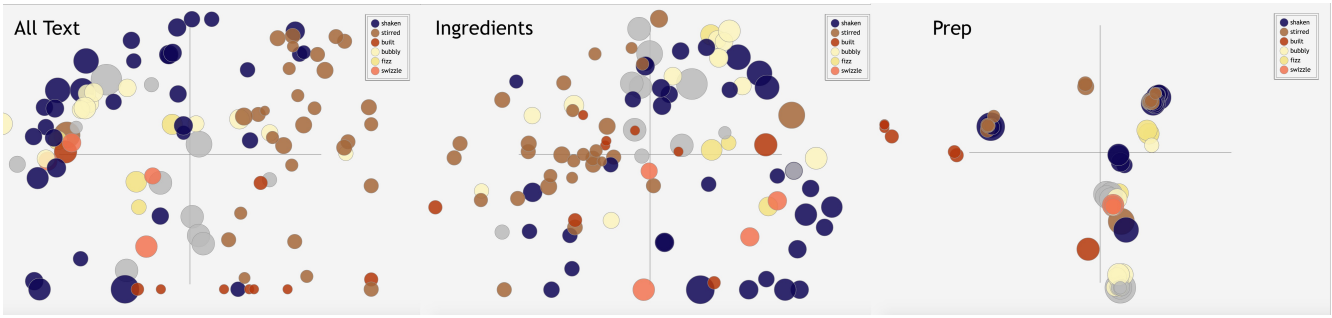
All the stirred drinks are on the bottom. The small grey and pink clusters on the inside represent toddies and swizzles respectively. Shaken drinks are split into a large clump on the right and a smaller clump--which are served in an old-fashioned glass instead of the traditional coupe--on the left. The upper peninsula of the rightmost clump is actually shaken drinks served in a highball glass, which serves as a bridge to the other highball drinks at the top. And smack in the middle we find the Ramos Gin Fizz; a drink so singular that it could only be in another orbit altogether... or perhaps alone at the center of gravity.

PCA

Finally, I went back to the a real classic, Principal Component Analysis (PCA). This might be the only algorithm we've touched that you could actually do with pen and paper. Well, maybe a large chalkboard. Or a lot of sheets of paper.

```
        ...

        elif mode == 'pca':
            X_pca = PCA(n_components=2).fit(X).transform(X)
            XY = pd.DataFrame(X_pca, columns = ['x', 'y'])
            print("loaded PCA data")

    # assign to dataframe and return
    df['x'] = list(XY['x'])
    df['y'] = list(XY['y'])
    return df
```

Unfortunately, it wasn't up to the task.



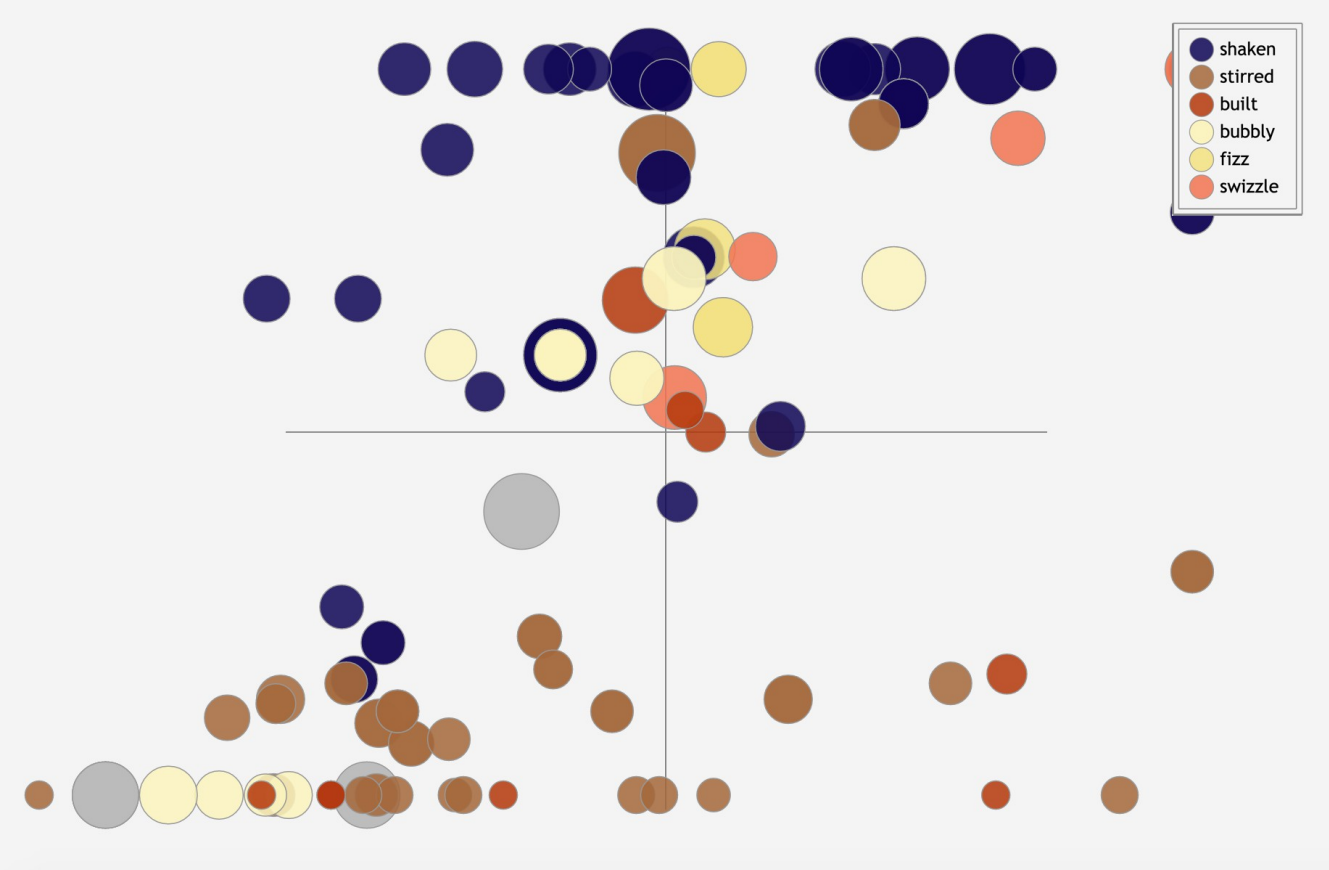
Initially I liked these a lot. Complicated patterns emerged from the cluster. Psychedelic waves of vermouths and amaros spun in front of my eyes. But sadly, as is often the case, the psychedelia dissipated upon deeper inspection. As I browsed these maps, I realized I was reasoning with goat entrails. There was nothing really there. I grew bored.

Chemistry

The punchline, unfortunately for those of us who make are living in this Data Science business, is that the simplest solution (which involves no Machine Learning at all) is often the best. Or maybe by "best" I just mean "easiest to make serviceable in a reasonable amount of time." In other words, the Data Science didn't fit on the skateboard. Or maybe I just mean, the most useful.

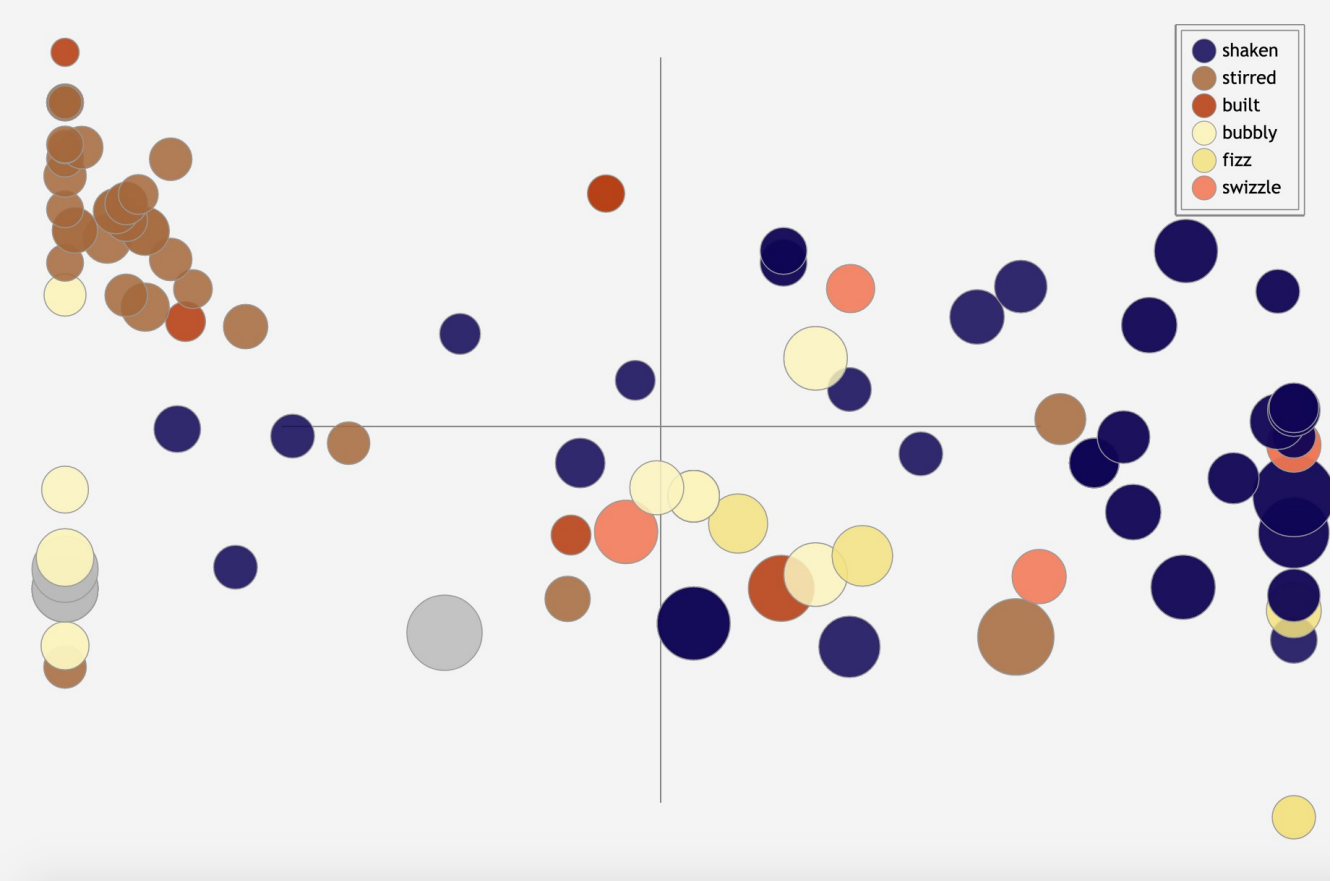
After staring a these for a few weeks, and trying them out on some only marginally interested innocent bystanders, a dark horse fourth way took the lead. The elements of the drink that most connected with people's visual imagination was the same one that interacts with their taste buds most directly: the chemistry is real.

By only the sugar and acidity measures mentioned earlier, I saw a downright sensible map of the drinkspace.



The main issue was the clutter, not in the middle, but in the top and the bottom. This is the result of drinks (at the top) with significant lime or lemon in them, and a lot of drinks (on the bottom) with nothing acidic at all.

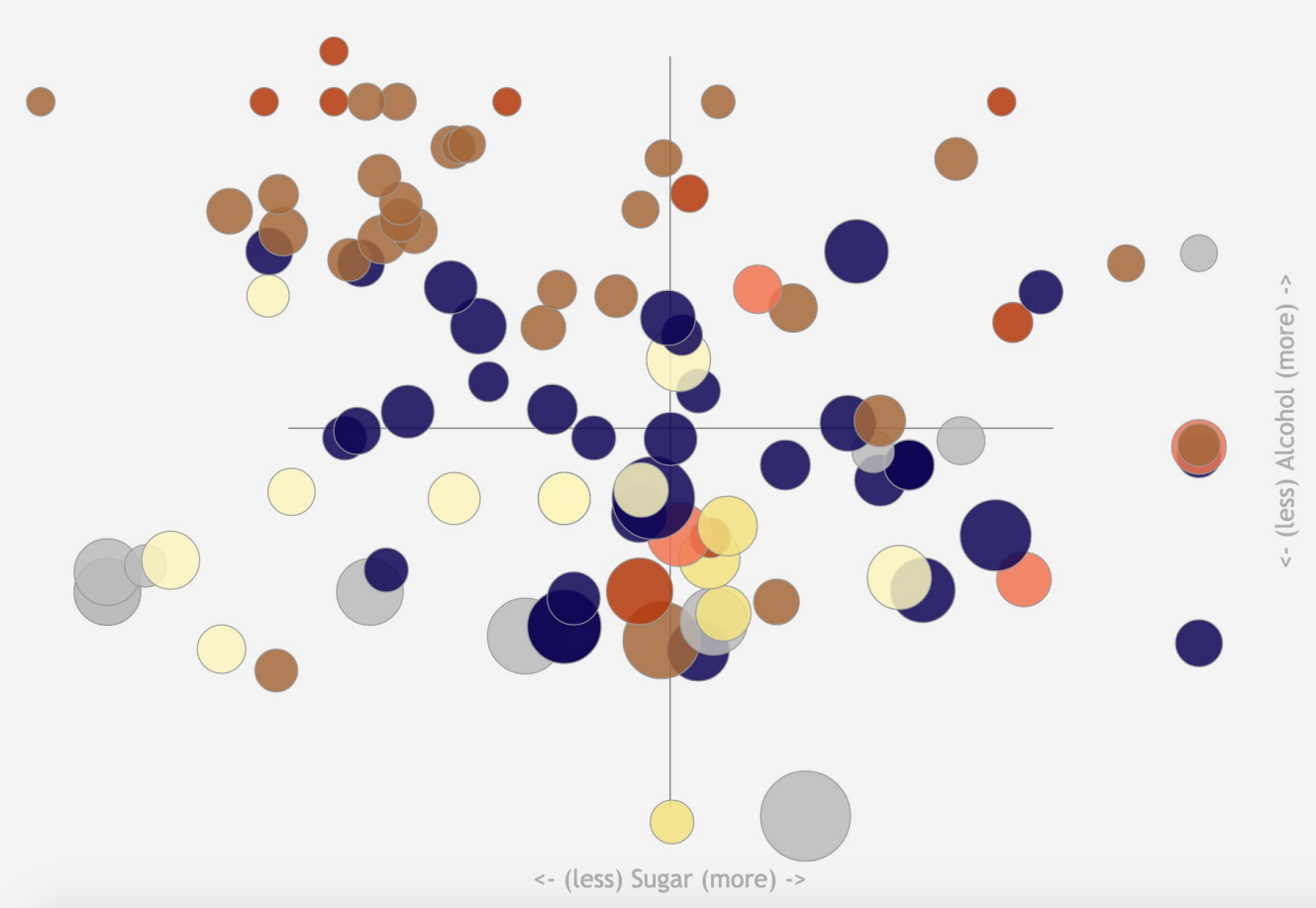
Flipping the axes to put acidity on the left-to-right (X-axis) and Alcohol By Volume on the Y-axis makes things a little more interesting.



But the cluttered edges remain. I do often want to choose my drink based want to choose my drink based on how strong I want it and how much citrus I want to squeeze into it. But my (fake*) acidity measures just weren't behaving well on the page.

Sweet and Strong

We're told sugar and spice make everything nice. That's especially true when by “spice” we actually mean “booze.” When we spread the drinks out by their percent sugar and percent alcohol, the universe takes shape.



Your bottom right quadrant is for warm-weather outdoor drinking, all full of Aperol and grapefruit juice and Punt e Mes. Refreshing, not too strong, and nice lifting sweetness like a perfect 7 & 7. The top right gets stronger, but no less sweet. It is dominated by the mysterious Chartreuse-drenched cousins: The Bijou and The Last Word. These are drinks you only order in cities full of artists and elegant street lighting. And only after 10pm.

The bottom left is strange territory. Drinks that are neither strong or sweet... so what are they? Bubbly for one. We see some old friends like the original Champagned Cocktail and the elegant French 75. A few egg white fizzes also find their way towards the bottom, transitioning the dry bubbles back towards the fertile and adventurous endeavor of combining hard liquor and dairy products.

If we're being honest though, the top left is where I tend to do most of my work. Call me a purist, but to me a cocktail should be what it originally was: liquor, slightly sweetened, and then spiced and embittered by something interesting.

The original, which we now simply as The Old-Fashioned, used Angostura Bitters as that “something interesting.” The Old-Fashioned itself, and several of it's closest relatives, can be found in the upper left corner. Arrayed below them and to the right are their second cousins who a dash of simple sugar for the luscious embrace of sweet vermouth. This is where we find the Manhattan, the Vieux Carre, and the Bobby Burns, among other favorites. The dry martini, of course, also lives in this quadrant, as do it's slightly funkier brethren (or forbearers) the Hanky Panky and the Martinez. And I'll make one last shout out to the Corn 'n' Oil, near the top and directly in the middle. A wicked brew quickly became my favorite summer evening sipper when I learned to homemake falernum a few years ago.

In Closing

And there we had it. At least we had a good enough grip on it to feel alright about putting it on the internet. The drinkViz is live on the drinkBase website (ntbloom.com), although it currently has an "experimental" tag on it, which I'm offended by *only* because I believe it's meant as a warning more than an invitation. I guess I'd say drinkViz is still experimental. I'd be a lot *more* experimental if we embedded it in three dimensions and contracted with some freelance "augmented reality" shop for a more immersive experience. But that might be just a pour or two too far. At least for now.

Until we next raise a glass together, thanks for reading.
S

Postscript notes

- I wrote the visual in d3.js because it's the best. For my money, it gives you the most control and the most flexibility, while still being a pretty simple and intuitive interface for manipulating raw data into a colorful, interactive visual. The code, improved and more properly CCS'd by Noah, is in the repo. *Note:* **all the other code**, for those of you who care about things like React and/or postgreSQL, is also the repo linked in the previous sentence. Noah wrote all that.
- All of the calculations for each drink were written in Python (actually, I originally did some of them in R. I can't remember why. I translated it to Python before passing it off to Noah. :shrug: emoji) although some of that has since been translated into SQL.
- Noah and I had a lengthy debate about my measure of acidity. It boiled down to his assertion that you can't do math with acidity (still disputed, though I haven't taken or read any chemistry in close to two decades) versus my assertion that, even if that's technically true, the devil is not in the details it's in the allure of false precision. I maintain that my arithmetic acidity calculations were correct *enough* to serve the purpose of showing the relative acidity of different cocktails at a level of precision that was discernible to the human eye in a web browser. All that said, I couldn't find a way to represent acidity that didn't clutter the more important aspects of the visualization. So, at this point, it remains in the database but not in the data viz.
- An apology for the *Liquid Intelligence* graph. Dave, if you ever read this, I'm sorry for using your graphic without permission, and even more so for the shotty reproduction of it. With your blessing I will gladly accept a better digital image of the plot. Or take it down altogether, if you'd prefer.