ABSTRACT
Today’s recipe search websites, which simply present a list of recipes to click through, do not provide efficient ways to compare recipes and understand the differences between them. Our visualization captures the variation in ingredient composition by graphing the amount of the most common ingredients in related recipes, allowing users to sort and explore their options. We created an ingredient database by pulling and parsing data from allrecipes.com. Using public code libraries for data visualization, we created ingredient distribution graphs that illustrate the amount of each ingredient in each recipe and implemented data brushing to allows users to filter their search results. Links to the actual recipes allows users to compare how different ratios of ingredients can lead to different variations on a dish. The feedback from test users was that the visual comparison showed surprising variance in recipes and that the visualization would be useful for finding recipes to make with a limited amount of ingredients. Future work would include making the displayed ratios more meaningful through data normalization and the attachment of ratings and review keywords to the distributions to show how the taste of a dish can change as its composition changes.

INTRODUCTION
Cooking is a creative endeavor, based on principles of flavor matching, chemistry, and technique. It is also a flexible art, with infinite variation covering a vast range of ingredients and taste.

However, cooking is dominated by recipes, which fail to capture the richness of this landscape. In their standard form, recipes do not tell the reader anything about which measurements are critical and which can be adjusted, or how changes in ingredient or technique will affect the final result. In fact, English speakers occasionally use the term “cookbook” as an adjective to describe something presented in an ordered sequence with little room for nuance or creativity.

Recipes are like navigation directions from one address to another, which provide a clear sequence of steps but do not help us if we get off track or want to take a slight detour. What we really need is a map.

The Internet has added another layer to the problem. There are dozens of recipe-sharing websites, plus countless food blogs and cooking columns. When a would-be cook searches for a recipe, they’re confronted with a paralyzing number of choices, and are left to pick one based on what title sounds most appetizing or whichever page has the most appealing picture.

Yet the vast troves of recipes online might also provide a solution. If individual recipes are points in a large food-landscape, then by aggregating thousands of recipes we may be able to uncover the shape of the underlying terrain.

This project aims to create a new tool to help novice and intermediate cooks explore the recipe space, both to find recipes they are interested in and to give them an intuition for the key ratios and dimensions of variance.

Popular books such as Ratio\(^1\) describe how mastery of a handful of fundamental ratios enables a cook to make a great variety of meals and experiment with confidence. Our objective is to enable users to explore and discover such ratios on their own.

RELATED WORK
Online tools
There are many online recipe sites, most of which innovate beyond cookbooks by providing enhanced search functionality.

Allrecipes [1] and Epicurious [7] use a text search that can be refined with filters, such as the inclusion or exclusion of certain ingredients, dietary concerns, meal and course, and type of cuisine. MyFridgeFood [6] allows users to enter the ingredients in their kitchens and find recipes that can be made primarily with the listed ingredients. Yummly [14] aggregates recipes from many other sites and parses their ingredient lists, allowing users to evaluate and search based on health information, such as dietary constraints, allergies, cooking techniques, nutritional content, and tastes.

The user interfaces of these sites resemble giant photo albums of foods (often with clickbait titles), and are meant to entice users to scroll through the recipes, which are sorted by relevance to the text search, rating, or popularity. Clicking

\(^1\)This snippet from the back cover of Ratio perfectly describes the motivation behind our approach: “Cooking with ratios will unchain you from recipes and set you free. With thirty-three ratios and suggestions for enticing variations, Ratio is the truth of cooking: basic preparations that teach us how the fundamental ingredients of the kitchen – water, flour, butter and oils, milk and cream, and eggs – work. Change the ratio and bread dough becomes pasta dough, cakes become muffins become popovers become crepes.”
on a recipe yields a maximized view of said recipe, a photo of it, and related information, such as reviews and nutritional content.

These mainstream approaches, which focus primarily on multifaceted search mechanisms, present recipes to users as precomputed solutions as opposed to data points to compare, analyze, and extrapolate from. Our focus is instead on visualizing recipes in a way that provides users with the ability to analyze what the recipe content means.

**Past research**
For nearly twenty years, researchers have explored additional ways of searching, parsing, and analyzing recipes shared on the Internet.

The CMU CURD project [5] introduced MILK, a “Minimal Instruction Language for the Kitchen”, and published a small database of MILK-annotated recipes for testing NLP parsing of recipes. While MILK breaks down recipes into ingredients, tools, and actions for easy, computer-driven recipe analysis, their hand-annotated database is not large enough for our application. We chose to focus primarily on ingredients, as the way they are combined is often standard across different recipes of the same type (e.g., beat butter and sugar to make cookies).

Ahn et al. [13] mined recipes to show what ingredients have similar flavor makeups. To visualize their results they created a graph that mapped foods based on their flavor compounds, revealing clusters of ingredients with similar flavors. The analysis of the graph was focused on using this flavor network to compare how regional cuisines across the globe combined the various flavor clusters. In contrast to their exploration of global trends, we focus on the relationships between recipes for individual dishes, or dish categories.

Teng et al. [12] mined the comment sections of recipe sites to create networks for both substitution and complement ingredients, which allowed them to predict which recipes would receive higher ratings, given which ingredients pair well and which ingredients had more preferred substitutes. The substitution network could recommend what ingredients could be swapped out, but the ingredient substitution simply provides a new recipe for the same dish. By allowing users to perform their own analysis of ingredient composition, we hope to allow interested users to infer the relationships between ingredient and taste changes.

**DATABASE CREATION**
**Web scraping**
We began by pulling over 200,000 pages from one popular website, allrecipes.com. Pages on AllRecipes have the URL format http://allrecipes.com/recipe/<NUMBER>/<NAME>. The NAME portion is redundant, so we scrape pages by simply querying recipe numbers in sequence. For each page, we usexpath queries to select the HTML DOM items of importance and extract the text. The values (name, ID number, category, and ingredient list) are stored in a flat JSON file.

At this point, it is interesting to take a brief detour to comment on the organization of Allrecipes. Figure 1 shows the categorization of all the pages we downloaded, and brings out a few unexpected quirks.

While Allrecipes has an extensive categorization system, only a small percentage of the recipes are labeled. For the first few thousand pages, recipes are grouped roughly by category: cookies, cakes, salads, and soups, and so on. After a while, the categorization becomes seemingly random, and there are large swathes of recipes with no categorization. Perhaps this initial organization is due to a database reorganization early in the site’s history.

The biggest surprise was that there were tens of thousands of instances of the same recipe: “Johnsonville Three Cheese Italian Style Chicken Sausage Skillet Pizza”. This single recipe makes up about half of the total number of pages on the site. Visiting these pages in a browser confirms that it isn’t a scraping error, but searching for this recipe does not yield any results. It is unclear whether this repetition is due to a database error, a test case, or someone’s partially-removed attempt at SEO.

**Parsing**
For each recipe, our parsing tool consumes the raw scraped data and produces a new list with standardized ingredients measured in metric units. We use a set of regular expressions to pull out the amounts (e.g., 3/4, 1 1/2, 3.5). Then we do simple matching to find the unit (e.g., oz, cup, teaspoon, kg) and convert it to the metric equivalent (ml or grams).
The more difficult task is to match the ingredient as listed in the recipe to a known ingredient which we can compare across recipes. To do this, we remove adjectives (e.g., minced, crumbled), strip off plurals (eggs → egg), and use a handwritten substitution list to handle the many corner cases (e.g., sugar → white sugar, orange peel → orange zest).

With this complete, we check whether the ingredient is in our database. Known ingredients are written to the output file; unknown ingredients are written to an error log.

**VISUALIZATION DESIGN**

**Design of visualization**

Our visualization relies upon the open-source Javascript data visualization libraries D3[2], Crossfilter[11], dc.js[3], and Bootstrap[8] for styling.

As shown in Figure 2, we display distribution graphs for the top-eight ingredients by number of appearances for a set of recipes. The name of the ingredients and their standardized unit is displayed in the title of the distribution graph.

We primarily wanted to show the variance and distribution of amount of ingredient used throughout the recipes, which made a distribution graph an appropriate choice. These distribution graphs were made by creating one-dimensional scatter plots using the scatter plot from dc.js [3].

Each point on the scatter plot represents the amount of an ingredient in a particular recipe. The points are given slight opacity to show bubble overlap. While this may occlude the number of recipes with a certain amount of ingredient, most users were focused on the different amount values available versus the number of recipes per amount. Future work could change the radius or color of the dots to better indicate how many recipes have a particular amount of an ingredient.

We wanted users to be able to get a general sense of the distribution of ingredients, so we chose to use horizontal scatter plots stacked on top of one another with minimal spacing between each chart. Users can see a general trend in ratios - or shape across the plots - for the recipe and view changes simultaneously as filters are applied.

The horizontal orientation also allowed the distribution graphs to be longer, allowing for space between the data dots, without being clipped by the viewing window. To support multiple browser sizes, we implemented dynamic width that expands or shrinks the charts when the window is resized, which allows the full ingredient distribution graphs to always be viewed. Each ingredient axis is scaled to span from the lowest (typically zero) amount of ingredient used to the highest.

To connect the ingredient data with real recipes so users may see the context to their data brushing, we include a name chart. The name chart was made with dc.js bubble chart [3].

Each recipe is displayed by name in the bubble chart, which sorts them alphabetically and colors them by the first letter of their name to visually break up a possibly dense row of bubbles. The color scheme was generated by ColorBrewer 2.0 [4] with a qualitative scheme and 12 data classes to make the color of the recipe bubbles aesthetically pleasing. Future work could be to color them by subcategory of recipe type (e.g., Bar Cookies, No-Bake Cookies, Drop Cookies).

While the visualization can provide a handy sense of where the average recipe for a dish might fall, we believe it to be more important for users to interact with the data. Instructions at the top of the visualization let users know what types of interactions are supported and can be hidden for a better view of the data.

Users can explore different ratios of ingredients by filtering the data by dragging and clicking to select recipes with ingredients in certain ranges. Figures 3 and 4 shows the effects of filtering on the ingredient distribution graphs and name charts.

Animated transitions help visually follow the cross-chart filtering, as dc.js [3] removes data points from other distribution graphs that dont fit the filters from other charts. Data points that do not fit the filter on the current chart but fit the filters on other charts are shown smaller and more opaque.

Filtering on an ingredient distribution graph also filters on the name chart by lifting bubbles of recipes that fulfill the filters. Users can also click on a recipe bubble in order to filter by recipe, which raises the bubble further and displays the name of the recipe. The separate level of recipes that still fulfill the ingredient filters is maintained, so that users may select more recipes with the same ingredient requirements.

During early user testing, users commented that they would like to navigate to the original recipe once they filtered through their options. To enable this behavior, we used D3 [2] to display and update links for the recipes selected in the name bubble chart by plugging the Allrecipes ID number of the recipe into the standard form of http://allrecipes.com/<NUMBER>. Figure 5 illustrates the link displays.

Reset buttons for each chart allows for fast filter removal from the corresponding chart.

**Search implementation**

The visual analysis using our visualization is most interesting and effective when recipes with similar ingredients are compared - otherwise the majority of the data points would be zero as the most common ingredients would not be very common.

To provide user flexibility in the subject matter, we implemented a simple text search that loaded the recipe data for only the recipes whose name included the string inputted by the user. We further narrowed the visualization by only gathering the ingredient data for the eight most occurring ingredients across the searched subsection. If a top-eight ingredient did not appear in a recipe, its amount was set to zero.

When a new search query is entered, we show a simple “load-
Figure 2. Implemented visualization for cookie recipes. Ingredient distribution graphs are displayed for each of the eight most common ingredients across the recipes. A bubble chart with the names of each recipe allows for filtering by name.

Figure 3. Unfiltered data using a database of cookie recipes.

Figure 4. Filtered data using a database of cookie recipes. A selection window has been dragged around recipes requiring 4–7 eggs, and the colored circles corresponding to those recipes are pulled up above the rest.
ing” image for the couple seconds while the search is performed and the data is reloaded. All links and ingredient charts are erased with their dimensions in the crossfilter data, and then the crossfilter is regenerated with new data, updating the name chart and displaying new ingredients.

Figure 5 shows the complete visualization with search and filters applied to ingredient and name charts.

**APPLICATION IMPLEMENTATION**

Our live visualization is hosted by Heroku at [http://tinyurl.com/recipeasy](http://tinyurl.com/recipeasy) and the source code is on GitHub at [http://github.com/stevenbell/recipeviz](http://github.com/stevenbell/recipeviz).

Our database is a simple 30MB JSON file which contains the parsed and simplified recipes from prior data scraping of Allrecipes.com. Each recipe data point contains its name, recipe ID, designated recipe category, and ingredients list containing the ingredient name, amount, and measurement type.

Our backend is a Python Flask [9] server, which serves our visualization code and hosts a single REST API endpoint: `/search/<QUERY>`. Our frontend makes a HTTP GET request to the search endpoint each time a user searches for a recipe. When our backend receives the HTTP GET request, it searches our JSON database and returns a JSON array of recipe objects consisting of all recipes matching QUERY, along with the ingredient names and amounts, and the corresponding URL to Allrecipes.com. Our frontend uses the queried URL to visualize the name and ingredient charts.

**USER STUDY**

**First Iteration**

Our first prototype of our recipe visualization had only a selectable bubble chart representing recipes, and four vertically oriented ingredient distribution graphs. The name chart was colored randomly, and ingredient charts took up lots of space. Searching for recipes was not yet implemented, so we relied on a hardcoded data set of cookie recipes to be visualized. Recipes did not have links to their pages at Allrecipes.com yet.

We intentionally provided no instructions to our users in order to observe their initial reactions and interactions to our main prototype features of brushing and linking. We did allow them to ask clarifying questions if they were stuck or confused, and let them know about the hardcoded dataset.

Users generally enjoyed interacting with our visualization. Most found the visualization to be aesthetically pleasing and praised the minimal design. We found that the filtering animations by brushing the ingredients data seemed to delight and fascinate users. In fact, most users really enjoyed clicking the recipe name bubble chart and seemed to spend most time clicking on different bubbles that were filtered.

Users’ saw our prototypes’ main value proposition as being able to discover viable recipes with only a limited number of ingredients. To use the cookie recipe dataset for example, several users mentioned that filtering recipes that only required a low amount of eggs to be very useful to them. One user said, “it’s useful to see what I can make with a limited amount of ingredients.” Another user said that it “was easy to see the ingredient ratios and to be able to add my own ingredients just by looking at the distribution.”

Many of the current features came from initial user feedback. We added an instructions panel since we found that many users didn’t know how to interact with the ingredient distribution graphs. Users often stuck to selecting the multi-colored bubbles of the name chart instead of brushing ingredients, unless otherwise guided towards it. We also implemented a color scheme that maps alphabetical order to color to help differentiate recipe bubbles since users had associated meaning to an otherwise meaningless visual encoding (recipe bubbles were colored randomly).

We also added more visual indicators to represent potential actions and clearly identify sections of the visualization. When selecting a recipe bubble, the name of the recipe would appear, and we made it a hyperlink so users can click to see the recipe details on Allrecipes.com. We added section headers to represent recipes that were selected by the recipe bubble chart, and designated the recipe bubble chart as Filtered Recipes (A-Z) to provide more context on what the chart represented and its organization. Furthermore, we made the reset button on the charts to be bold and red so that users wouldn’t miss it, as they often did when exploring and brushing the visualization.

**Performance**

During our initial testing, our prototype visualized a rather small, hardcoded dataset of 54 KB. When searching our recipe database of 30MB was implemented, we started to see a noticeable performance hit when interacting with our visualization due to the larger sizes of data to visualize.

On average, querying the database for recipes would take at least 5 seconds, with queries to more popular recipes like cookies or cake would take longer. With a much larger database and with general queries to common recipes, our frontend visualization had to parse much more data. For example, a query to cake recipes returns almost 1MB of data to our client. From GET request to initial load of cake recipes took 15 seconds, with an average of 1 second delay for interactions.

While it is understandable that more data requires more computing power, we must optimize for the user experience and lag definitely detracts from the overall user experience. To help alleviate this problem, it would be useful to constrain the amount of data sent to the client to about 500 KB to strike a balance between performance and recipe variance. Our search and data loading methods could also be further refined.

**FUTURE WORK**
Several small technical improvements are necessary for our tool to be as effective as we envision.

First, we need to expand our known ingredient database. While we cover most common ingredients, there are still many missing ingredients or ingredient mappings. A particularly embarrassing example is that a search for “chicken parmesan” turns up a number of ingredients, but omits “chicken.”

Because recipes produce different quantities of food, it is important to normalize the ingredient amounts by the total quantity of the recipe (e.g., one industrial-sized cookie recipe calls for 40 eggs, completely skewing the ranges for ingredients). We plan to convert all quantities to mass, and normalizing the amounts to the median recipe mass. On the front end, we will need to convert back to units that the user is more familiar with, such as teaspoons rather than grams for small measurements.

After observing several users interacting with our visualization, we would make our brushing operation easier and more visible. The user currently drags a 2D box across our 1D data, which is inefficient and error prone.

Users also expressed interest in seeing more recipe information tied to the data distribution without leaving the visualization page. We plan to expand the visualization to include indications of how ratings and/or the use of adjectives in reviews (e.g., chewy, crispy, savory) changes as with changes in the amount of each ingredient.

Looking further ahead, we would like to provide higher-level insight into the recipe space as a whole, without the user needing to perform a search first. We began experimenting with clustering and dimensionality reduction as ways to examine the relationships between clusters of related recipes, but did not get to the point of having useful results.

Perhaps most importantly, we’re looking forward to sharing our work in its current state with others and watching what interesting and delicious trends they uncover.

REFERENCES

   http://flask.pocoo.org/.


12. ChunYuen Teng, Yu-Ru Lin, and Lada A. Adamic.  
    Recipe recommendation using ingredient networks.  
    *CoRR*, abs/1111.3919, 2011.

13. James P. Bagrow Albert-László Barabási  
    Yong-Yeol Ahn, Sebastian E. Ahnert. Flavor network and the principles of food pairing.  
    *Scientific Reports*, 1(196), December 2011.