University of British Columbia

Social-Ecological Economic Development Studies (SEEDS) Sustainability Program

Student Research Report

CLIMATE-FRIENDLY FOOD SYSTEMS (CFFS) LABELLING PROJECT

Development of the Evaluation Framework for UBC's first Climate-Friendly Food Label (Pilot Phase 1, 2, 3)

Prepared by: Silvia Huang, Undergraduate Student and Climate-Friendly Food Systems Data Analyst

Supervised by: Juan Diego Martinez, Ph.D. Candidate at the Institute for Resources, Environment and Sustainability (IRES)

Prepared for: Campus & Community Planning (Sustainability & Engineering), UBC Food Services, UBC Botanical Garden

University of British Columbia

May 2022

Disclaimer: “UBC SEEDS Sustainability Program provides students with the opportunity to share the findings of their studies, as well as their opinions, conclusions, and recommendations with the UBC community. The reader should bear in mind that this is a student research project and is not an official document of UBC. Furthermore, readers should bear in mind that these reports may not reflect the current status of activities at UBC. We urge you to contact the research persons mentioned in a report or the SEEDS Sustainability Program representative about the current status of the subject matter of a report.”
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>3</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>3</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS</td>
<td>3</td>
</tr>
<tr>
<td>EXECUTIVE SUMMARY</td>
<td>4</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>6</td>
</tr>
<tr>
<td>1.1 RESEARCH CONTEXT &amp; TOPIC</td>
<td>6</td>
</tr>
<tr>
<td>1.2 RESEARCH RELEVANCE</td>
<td>7</td>
</tr>
<tr>
<td>1.3 PROJECT PURPOSE, GOALS, AND OBJECTIVES</td>
<td>7</td>
</tr>
<tr>
<td>2. METHODOLOGY</td>
<td>8</td>
</tr>
<tr>
<td>2.1 RESEARCH METHODOLOGY AND METHODS</td>
<td>8</td>
</tr>
<tr>
<td>2.2 DATA COLLECTION</td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Primary Data Collection</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 Secondary Data Collection</td>
<td>9</td>
</tr>
<tr>
<td>2.3 ASSUMPTIONS</td>
<td>10</td>
</tr>
<tr>
<td>2.4 EVALUATION OF MENU ITEMS</td>
<td>11</td>
</tr>
<tr>
<td>2.5 EVALUATION FRAMEWORK</td>
<td>12</td>
</tr>
<tr>
<td>2.6 BASELINE AND LABEL CUT-OFFS</td>
<td>14</td>
</tr>
<tr>
<td>2.7 SENSITIVITY ANALYSIS AND DAILY ALLOWANCE VALUE (DV)</td>
<td>17</td>
</tr>
<tr>
<td>2.8 ADDITIONAL ATTRIBUTES</td>
<td>17</td>
</tr>
<tr>
<td>3. RESULTS</td>
<td>19</td>
</tr>
<tr>
<td>3.1 SUMMER PILOT (PHASE 1)</td>
<td>19</td>
</tr>
<tr>
<td>3.2 FALL PILOT (PHASE 2)</td>
<td>22</td>
</tr>
<tr>
<td>3.3 SPRING PILOT (PHASE 3)</td>
<td>23</td>
</tr>
<tr>
<td>4. DISCUSSION</td>
<td>27</td>
</tr>
<tr>
<td>5. RECOMMENDATIONS</td>
<td>29</td>
</tr>
<tr>
<td>5.1 SHORT-TERM RECOMMENDATIONS (&lt; 3 MONTHS)</td>
<td>29</td>
</tr>
<tr>
<td>5.2 MID-TERM RECOMMENDATIONS (&lt; 6 MONTHS)</td>
<td>29</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

- Figure 1: Flowchart for Calculating the GHG Emissions of a Bacon Sandwich
- Figure 2: Evaluation Framework Flowchart
- Figure 3: Phase 1 Traffic Light Labelling System
- Figure 4: Phase 2 Traffic Light Labelling System
- Figure 5: Phase 3 Single Icon Labelling System
- Figure 6: GHG Emissions (Kg) Per Serving (Phase 1)
- Figure 7: CFFS Label on Menu Board (Phase 1)
- Figure 8: GHG Emissions (Kg) Per Serving vs. Per 100g (Phase 1)
- Figure 9: Phase 1 Label Cut-offs
- Figure 10: Phase 2 Label Cut-offs
- Figure 11: GHG Emissions (g) Per Serving vs. Per 100g (Phase 2)

LIST OF TABLES

- Table 1: UBC 19-20 GHG Emissions Baseline (Phase 2)
- Table 2: Sensitivity Analysis 1-3 Results
- Table 3: Label Counts in GHG-only and Composite Metric
- Table 4: Label Change in GHG-only and Composite Metric
- Table 5: UBC 19-20 GHG, Nitrogen, Water Footprints Baselines (Phase 3)
- Table 6: Standard Healthy Diet (2200 Calories Per Day)
- Table 7: Daily Allowance Value (DV)

LIST OF ABBREVIATIONS

- CAP: Climate Action Plan
- CFFS: Climate-Friendly Food Systems
- CO2eq: Carbon Dioxide equivalents
- DV: Daily allowance Value
- GHG: Greenhouse Gas
- OC: Optimum Control
- UBCFS: UBC Food Services
- UBCFSP: UBC Food System Project
EXECUTIVE SUMMARY

How is the University of British Columbia (UBC) able to offer a system that helps consumers choose more Climate-Friendly menu items? From a global perspective, food systems are an enormous driver of climate change and contribute to more than one-third (34%) of global greenhouse gas (GHG) emissions, which represent 17.9 billion tonnes of carbon dioxide equivalents (CO2eq) (Crippa et al., 2021). Other estimates suggest that the food system is responsible for one-quarter (26%) of global GHG emissions, representing 13.6 billion tonnes of CO2eq (Poore & Nemecek, 2018). This brings a range of opportunities for actions to mitigate the effect of food systems on the climate.

The Climate-Friendly Food Systems (CFFS) labelling project at UBC takes action to inform the UBC community of the climate impact information of menu items they purchase every day at UBC Food Services (UBCFS). The label provides an opportunity for the campus community to make informed purchasing decisions that can promote a Climate-Friendly Food System. This research report was prepared by the CFFS data analyst, supervised by a member of the CFFS Action Team. This report is focused on the data analysis and the back-end implementation of the CFFS Labelling pilot and is complementary to the report on the communication and definition side prepared by the CFFS communication and engagement coordinator.

The CFFS Labelling pilot is part of the bold actions taken by UBC in response to the Climate Action Plan (CAP) 2030 scope 3 emission reduction goal. The CFFS Action Team has been formed to accelerate transitions toward a Climate-Friendly Food System and advance the CAP 2030 food-related actions and priorities. This project is part of the SEEDS Sustainability Program research collaboration to develop, pilot, and evaluate UBC’s first Climate-Friendly Food Label that aims to evaluate the climate impact of menu items sold at UBCFS outlets and operationalize the CFFS food label to inform Climate-Friendly menu choices. The research includes developing a methodology and framework that assesses GHG emissions and other CFFS attributes for menu items at UBCFS. It also evaluates perceptions and the impacts of the Climate-Friendly Food Label on awareness, knowledge, and purchasing decisions.

This project utilized a combination of literature review, discussion with peer institutions, and assessment of the feasibility in the UBC’s context to decide the methodology. The primary data sources (recipes and sales data) were extracted from the UBCFS inventory management system,
Optimum Control (OC). The data on carbon, nitrogen, and water footprint factors came from external secondary data sources.

The main deliverable of the project is the evaluation framework that conducts the evaluation process of recipes automatically once GHG emission factors have been assigned to each ingredient, and is updated to incorporate additional attributes and adapt to the expansion of the CFFS Label. The evaluation framework is able to read the primary data automatically and output the total GHG emissions, nitrogen footprint, and water footprint of each menu item. To determine the cut-offs for the levels of the label according to GHG emissions, we established a 2019 UBCFS GHG emission baseline and set cut-offs in accordance with the CAP 2030 GHG scope 3 50% reduction goal for food systems.

To help the transition to a Climate-Friendly Food System, we suggest that one way to mitigate the total food system emissions is to reduce the amount of meat and dairy consumption and replace them with plant-based protein products without compromising nutritional value. In addition, to improve the accuracy and specificity of current labels, we recommend UBC lead the engagement process and the establishment of a Pacific Northwest/Canadian-specific footprint factors database by conducting research collaboratively with peer institutions.

**Keywords:** climate label, Climate-Friendly, reproducible data analysis, GHG, nitrogen, water use, food systems

**References**


1. INTRODUCTION

1.1 RESEARCH CONTEXT & TOPIC

Roughly 26% of global total GHG emissions (13.7 billion tons of CO2eq) generated by human activities were contributed by the food supply chain (Poore & Newecek, 2018). A range from 10.8 and 19.1 billion tonnes of CO2-equivalent (CO2e) emissions per year representing between 21% to 37% of global total emissions has been reported by The Intergovernmental Panel on Climate Change (IPCC) Special Report on Climate Change and Land report.¹ According to Hannah Ritchie²: “Food emissions are around 25% to 30% from food. Around one-third, if we include all agricultural products.”

This brings a range of opportunities for climate action to mitigate the effect of food systems on the environment. In December 2019, UBC joined organizations and governments around the world to declare a climate emergency and renewed its commitment to sustainability, including a commitment to the CAP 2030 (an update from a 2020 plan) to accelerate UBC’s climate actions. As part of the CAP 2030, food was identified as an area of opportunity under scope 3 (indirect) emissions.

The purpose of the CFFS Action Team is to serve as engaged experts from the existing UBC Food System Project (UBCFSP) Steering Committee. The CFFS Action Team is responsible for the ideation, coordination, and development of student-led research, initiatives, and interdisciplinary collaborations that can accelerate transitions towards a climate-friendly food system and advance UBCFSP’s mission and priorities. In response to UBC’s CAP 2030, the CFFS Action Team aims to achieve a 50% GHG emission reduction associated with food systems by 2030 compared to 2019, starting with the development of a Food System Resilience & Climate Action Strategy, with support for campus-wide climate food labelling, and a toolkit to encourage more sustainable dietary choices and habits.

This project researches how to implement and operationalize the CFFS Label across campus by developing a back-end evaluation framework for the climate impact of menu items and implementing a label that indicates the impact of food sold at UBCFS. The main objective is

constructing an evaluation framework for analyzing the recipes and ingredients to provide a composite metric that informs customers about the food’s climate impact. The evaluation framework will incorporate a range of attributes that indicate aspects of the definition of CFFS for food products. The definition work and the additional attributes can be found in the complementary report developed by the CFFS Communications and Engagement Coordinator. Along with other education and engagement materials, the label will indicate and incorporate a range of CFFS attributes to give a comprehensive view of the food’s climate impact that students purchase at UBCFS.

1.2 RESEARCH RELEVANCE

In order to mitigate GHG emissions and other climate impacts of the food system, various actions from the food production and consumption side are necessary. As a major food provider at the UBC campus, UBCFS contributed to a large proportion of the total GHG emissions from food systems through students’ daily meals. The action of providing students with the GHG emission information of the menu items they purchase every day could help to educate and influence their purchasing behavior in a more climate-friendly way (Brunner et al., 2018). The CFFS Label is a clear and efficient presentation to indicate the climate impact information of menu items, thus helping students make purchasing choices that take the climate impacts into consideration.

1.3 PROJECT PURPOSE, GOALS, AND OBJECTIVES

This project aims to operationalize the CFFS Label by constructing an evaluation framework for analyzing the climate impact of menu items sold at UBCFS outlets. This includes creating a reproducible data analysis framework for calculating recipes’ GHG emissions; establishing a food GHG emissions baseline for the UBC Vancouver campus; deciding cut-offs for the CFFS Label; and further integrating additional CFFS attributes into the framework and establishing corresponding UBCFS baselines to decide label cut-offs.
2. METHODOLOGY

2.1 RESEARCH METHODOLOGY AND METHODS

This project utilized a combination of literature review, discussion with peer institutions, and assessment of the feasibility in the UBC’s context, such as available data and department support, to decide the methodology that best met the goals and objectives of this research. Methods were also determined through discussion with researchers from the University of Michigan, Université Laval, and the University of Victoria who are working on similar climate food labelling projects.

The research methods include primary and secondary data collection; evaluating recipes’ GHG emissions; developing an external data analysis framework; constructing a UBC GHG emission baseline; deciding label cut-offs; and incorporating additional attributes. Detailed explanations are provided below.

2.2 DATA COLLECTION

2.2.1 PRIMARY DATA COLLECTION

The raw recipe data for menu items sold at UBC food venues was extracted from the inventory management system Optimum Control (OC) of UBC by the FMIS Administrator of UBCFS. Due to system and administration restrictions, the data extraction was conducted manually instead of using database queries. Recipe data was extracted in XML file format, and each file contained one aspect of the recipe information, such as raw ingredients, preprocessed recipes used, and unit conversion information. The evaluation framework was designed in accordance with this data structure.

In Phase 1 at Mercante, in order to establish a 2019 GHG emissions baseline, the sales and recipes data for all products sold at Mercante between January 1 and December 31, 2019 were extracted from OC. Then, in the next two iterations of the CFFS Labelling pilot which expands to Open Kitchen, the sales and recipes data for all products sold at UBC’s three major student residence dining (i.e., Open Kitchen, Feast, and Gather) between September 1, 2019 and February 28, 2020 were extracted to calculate the UBCFS GHG emission, nitrogen, and water footprint baseline.
2.2.2 Secondary Data Collection

The GHG emission factor data comes from three main sources, in the following order of preference:

- First, we used the World Resources Institute (WRI)'s Cool Food Calculator emission factors for most of the food groups. It provides GHG emission data based on life-cycle assessments for major food categories in the North American region from research conducted from January 2015 to December 2018. These represented the factors used in the large majority of our ingredients\(^3\) (94.76%).

- Second, we used the GHG emission data from The Big Climate Database, published by CONCITO (Denmark’s green think tank), as a supplementary data source for food categories that are not in the Cool Food Calculator such as salt and vinegar. It provides GHG emission data based on life-cycle assessments for major food categories in Denmark. These represented 1.68% of total ingredients.

- Last, for some items that don’t have emission factors available, we calculated their emission factors manually by approximating their ingredients using recipes stored in OC or recipes found online. These represented 3.56% of total ingredients.

The GHG emission factors for the rest of the ingredients extracted from OC such as water, beverage, sauce, packaged food, kitchen supplies etc. was assumed to be zero either by the research assumption or the unavailability to estimate. See Appendix B for detailed source of data on GHG emission factors for each food category.

Note that the food groups were slightly adjusted from the Cool Food Calculator for better assignment of GHG emission factors on ingredients procured by UBCFS. For example, the GHG emission factors for more general-level food groups (i.e., fruits) were used for assigning ingredients that were not specified as less general food groups (i.e., apples, bananas, berries) and were renamed as "other" (i.e., other fruits) in the GHG emission factors list. See Appendix B for detailed food categories and emission factors.

\(^3\) Edible ingredients that have been evaluated only, excluding water, beverages, sauces whose emission factors are assumed zero and packaged food whose emission factors are unable to be estimated. Also excluding non-edible items that show up in the ingredients list extraction, such as human labor and kitchen supplies.
During Phase 3 of the CFFS Labeling implementation at Open Kitchen between March and April 2022, the label also incorporates the nitrogen and water footprint for each menu item to produce a composite metric. The nitrogen footprint factor data was provided by the Food Label Toolkit from Leach et al. (2016) and the water footprint factors (including freshwater and stress-weighed water use) were sourced from Poore and Newceck (2018). The footprint factors for nitrogen and water were also slightly adjusted and calculated for each food category present in the GHG emission factors list to avoid assigning footprint factors multiple times, thus improving the efficiency of the evaluation framework and label implementation. See Appendix B and C for detailed nitrogen and water footprint factors.

2.3 ASSUMPTIONS

To make the process of recipe evaluation consistent, accurate, and structured, several assumptions were made when evaluating the environmental footprints.

- The same GHG emission factor was assigned to different forms (puree, sliced, chopped, etc.) of the same raw ingredient.

- The same GHG emission factors were applied to different varieties of the same ingredient (i.e., red and yellow onions).

- GHG emissions are for the raw ingredients, and the final weight of serving takes into account loss or addition of weight during the cooking process (e.g., water evaporation in beef vs. water absorption in pasta) or loss from cutting out inedible parts (prepping stage).

- GHG emissions from the cooking process were ignored, because we had no knowledge of a standard, feasible method to calculate them.

- The GHG emission factor for water is zero.

- GHG emissions from the cooking process were ignored, because they are captured under Scope 1 (natural gas) & Scope 2 (electricity) emissions - i.e. for building energy supply. So distinct from the food (Scope 3) item

- The GHG emission factor for water is assumed zero.
● We ignored the water use in the prepping and cooking process for the water footprint calculation because we have no reliable way of estimating it for different dishes.

● We excluded the GHG emissions from sauces and dressings with unknown recipes (bought in bulk) that have no dominant ingredients.

● For prep recipes that have no standard unit information, the weight of the preps are the sum of the weight of all ingredients that they used.

● The nitrogen footprint and water footprint factors of some dairy categories (i.e. butter, yogurt, and cream) that are not specified in the source footprint data, are estimated as the nitrogen/water footprint factor of milk multiplied by the corresponding ratio of their GHG factor to milk. For example, the GHG emission factor ratio of butter to milk is 5.12. Then the nitrogen footprint factor of butter is estimated by multiplying 5.12 by the nitrogen footprint factor of milk.

● For vegan food categories whose nitrogen and water footprint factors are not specified in the source footprint data, their nitrogen and water footprint factors are the average of the footprint factors of all available vegan food categories.

● For animal food categories whose nitrogen and water footprint factors are not specified in the source footprint data, their nitrogen and water footprint factors are the average of the footprint factors of all available animal food categories.

2.4 EVALUATION OF MENU ITEMS

The GHG emissions of each menu item are calculated by summing up the weight of every raw ingredient multiplied by their respective emission factors. Ingredients' emission factors are assigned according to their category in the Cool Food Calculator, which provides data about the amount of GHGs emitted to the environment during the entire life cycle of a menu item.

For example, the process flowchart for calculating the GHG emissions of a bacon sandwich is shown in Figure 1 below. First, we get the raw ingredient (item) information and then categorize each item into the food categories in the GHG Emission Factors List. See Appendix B for all food categories and associated GHG emission factors. Next, we assign the GHG emission factors based on the food category for each item and calculate the amount of GHG emissions in grams for each
item used in this recipe. For recipes that use pre-processed recipes (preps), such as the garlic butter made of garlic and butter in this example, we calculate a GHG emission factor for this prep based on the items used and then calculate the total amount of GHG emissions in grams for this prep.

![Figure 1: Flowchart for Calculating the GHG Emissions of a Bacon Sandwich](image)

Lastly, we sum up all the GHG emissions of each item or prep and use this sum and the food group (i.e., lunch/dinner, breakfast, or desserts/snacks) to determine the label color for the Summer Pilot (Phase 1). For Phase 2 and 3 of the CFFS Labelling pilot which evaluated menu items based on per 100g standard, the weight and total GHG emissions for each dish were also calculated by summing up the weight and GHG emissions for each ingredient and then calculated the GHG emissions per 100g of food. The calculation of the nitrogen footprint and water footprint per 100g for each menu item follows the same process with the respective footprints.

### 2.5 EVALUATION FRAMEWORK

The evaluation of menu items is an automatic process that is conducted by an evaluation framework, a workflow documented in Python on Jupyter notebooks that calculates the GHG emission of menu items in an efficient and structured way. It reads the .xml files exported from OC and does most of the calculating process. See Appendix A for the code that constructed the...
evaluation framework. The process flowchart for the whole evaluation process is shown in Figure 2:

![Figure 2: Evaluation Framework Flowchart](image)

This flowchart presents the main steps and components that make up the whole evaluation process. And the color of each box indicates where this step takes place, or which system or software is associated with it. For a box that has two colors, it means it is associated with two systems or can happen in either place.

The first step is extracting raw ingredients and recipe data from the UBCFS inventory management system. Before feeding these raw data into the automated calculation process, it requires preprocessing and cleaning these data by listing and adjusting units for all ingredients and assigning them with associated GHG emission factors, which are from several external data sources such as the Cool Food Calculator. Data extraction from OC and preprocessing represent the largest time requirements every time new recipes need evaluation. Besides GHG emissions, the framework is also able to calculate the total nitrogen and water footprint and per 100g of food for each menu item. After these data gets processed in the automated calculation step/evaluation framework, it will output the environmental footprint of each menu item, and then we weigh these results with other qualitative attributes to have a weighted metric of the overall climate impact of each menu item. Lastly, we use the baseline data to decide the cut-offs for the three levels of labels, and the results can be shown on the Nutrislice, which is the online...
platform where students can see nutrition facts and also the climate label of the food they buy at UBCFS.

2.6 BASELINE AND LABEL CUT-OFFS

For the Summer Pilot (Phase 1) of the CFFS Label launched at Mercante and the following Fall Pilot (Phase 2) launched at the Open Kitchen, we decided to use the traffic light system to categorize foods by their climate impact into high, medium, and low levels, corresponding to the colors of red, yellow, and green. It would allow easy interpretation for customers to see the food’s emission level by looking at the colors. See Figure 3 for the design and meaning of the labels implemented during the Summer Pilot (Phase 1).

To determine the cut-off levels of the label according to the GHG emissions of menu items, we decided to establish a UBCFS GHG emission baseline that represents the average GHG emissions per dish before the label is launched. In this way, we can set cut-offs in accordance with the 50% UBC CAP 2030 GHG reduction goals for food systems. This requires utilizing the sales and recipe data during a period and then calculating the average GHG emissions per dish. For Phase 1, we decided to have separate sets of cut-offs for different meal groups (i.e., lunch/dinner, breakfast, desserts/snacks) due to the disparity in serving size and main ingredients inspired by the methodology by WRI.

The methods for determining cut-offs for the three levels of the label for the Summer Pilot (Phase 1) are shown below:

- **Green**: These food items have below-average GHG emissions compared to other food items sold within the same meal category (i.e., lunch/dinner, breakfast, or desserts/snacks) and have low enough emissions to achieve UBC’s 50% reduction target in food-related GHG emissions.

- **Yellow**: These food items have below-average GHG emissions compared to other food items sold within the same meal category (i.e., lunch/dinner, breakfast, or desserts/snacks) but higher emissions than what is necessary to achieve UBC’s 50% reduction target in food-related GHG emissions.

- **Red**: These food items have above-average GHG emissions compared to other food items sold within the same meal category (i.e., lunch/dinner, breakfast, or desserts/snacks). Food with
red labels would drive the average GHG emissions higher, thus impeding the process for UBC in achieving the 50% reduction target in food-related GHG emissions.

![Traffic Light Labelling System](image)

**Figure 3: Phase 1 Traffic Light Labelling System**

For Phases 2 and 3, due to the large variety and quantity of meal categories offered by the Open Kitchen, we calculated the GHG emissions per 100g of food for each menu item so that their environmental impacts are comparable between different meal categories no matter the serving size. Correspondingly, the UBCFS GHG emission baseline was also calculated based on per 100g of food to decide label cut-offs for Phase 2 and 3. The methods for determining cut-offs for the three levels of the label for the Fall Pilot (Phase 2) are shown below:

- **Green**: These food items have below-average GHG emissions per 100g and have low enough emissions to achieve UBC’s 50% reduction target in food-related GHG emissions.

- **Yellow**: These food items have below-average GHG emissions per 100g but have higher emissions than what is necessary to achieve UBC’s 50% reduction target in food-related GHG emissions.

- **Red**: These food items have above-average GHG emissions per 100g of food. Food with red labels would drive the average GHG emissions per 100g higher, thus impeding the process for UBC in achieving the 50% reduction target in food-related GHG emissions.

See **Figure 4** below for the design and meaning of the labels during the Fall Pilot (Phase 2).
For Phase 3 of the CFFS Label, we decided to shift from the traffic-light system to a single label indicating foods that are Climate-Friendly based on a composite metric including GHG emissions, nitrogen footprint, and stress-weighted water use and weigh each attribute equally. See Figure 5 below for the design of the label during the Spring Pilot (Phase 3).

The new methodology includes the calculation of a baseline for nitrogen and water footprint per 100g for UBCFS using 2019 recipes and sales data similar to the GHG emissions baseline. To get a composite metric that measures and weighs the three attributes equally, we divided the GHG emissions, nitrogen footprint, and stress-weighed water use per 100g of food by their corresponding baseline for each menu item. Then we average the three ratios of footprint to baseline to get the standardized metric. For the cut-off deciding which menu item gets the CFFS Label, we chose the threshold of 0.5 that is in accordance with the 50% UBC CAP 2030 GHG
reduction goals for food systems. A menu item with the CFFS Label means that this menu item has at least a 50% lower environmental footprint per 100 grams than other items.

### 2.7 SENSITIVITY ANALYSIS AND DAILY ALLOWANCE VALUE (DV)

To decide between the GHG-only metric and the composite metric on evaluating menu items, we conducted a sensitivity analysis by comparing the labelling results based on the GHG-only metric versus combining GHG, nitrogen, and water use (freshwater or stress-weighed water use) per 100g of food. It allowed us to see whether the incorporation of nitrogen and water footprints substantially changed which items got the label. Specifically, we compared the labeling results in the following steps:

1. The number of items that all 4 attributes (GHG, Nitrogen, Freshwater, Stress-weighed water) result in the same "color" of impact.
2. The number of items that all 3 attributes (GHG, Nitrogen, and freshwater) result in the same "color" of impact.
3. The number of items that all 3 attributes (GHG, Nitrogen, and stress-weighed water) result in the same "color" of impact.
4. The number of green, yellow, and red items using the GHG-only metric and the composite metric (GHG, nitrogen, and stress-weighed water use).
5. The number of items that changed label color after incorporating nitrogen and water footprints (i.e., red in GHG-only -> yellow in composite metric).

In addition, to provide students with more CFFS attribute information behind the composite label and the food they purchase at UBCFS for suggesting climate-friendly menu choices, we estimated a Daily Allowance Value (DV) of GHG emissions, nitrogen, and water footprints based on a "climate-friendly" healthy diet for a day (Leach et al., 2016). Then we can show students the percentage of footprints compared to the DV by consuming 100g of food they purchase. This information can be added to the Nutrislice menu item descriptions in the future.

### 2.8 ADDITIONAL ATTRIBUTES

Besides GHG emissions, the evaluation framework also considers the incorporation of new additional attributes for Phase 2 and 3 to produce a more comprehensive CFFS Label. The
additional attributes were the metrics to define a Climate-Friendly Food System by the CFFS Action Team, which were developed based on aspects of climate change mitigation and adaptation.

The potential additional attributes are land use, nitrogen pollution, water use, and local, which were developed from the CFFS definition research conducted by the CFFS Communication and Engagement Coordinator. To decide which additional attribute should be incorporated, we evaluated these attributes based on the availability of data, UBCFS's tracking ability/capacity for qualitative attributes, their impact on climate change mitigation and adaptation strategies, and evaluation survey results.
3. RESULTS

3.1 SUMMER PILOT (PHASE 1)

The Summer Pilot (Phase 1) for the operationalization of the CFFS Label took place at the Mercante, one of the UBCFS retail venues that remained open during the summer of 2021. The evaluation only focused on the GHG emissions of the menu items, most of which are pizzas that have almost the same serving size. The total GHG emission for each menu item, calculated by the evaluation framework, is shown in Figure 6:

![Figure 6: GHG Emissions (Kg) Per Serving (Phase 1)](image)

Note: the GHG emission results are based on 2021 data

The corresponding CFFS Label is available to students on the menu boards and also on Nutrislice. See Figure 7 for the actual look of labels on display.
The evaluation framework also calculated the GHG emissions per 100g of the product for each item. This gives another point of view for comparing the climate impact of the recipes. Although there are a few products that have high per 100 gram GHG emissions, which indicates that they may use a lot of high-emission ingredients, their respective total emissions are low due to the small serving size. For example, the GHG emission per 100g of Colazione is 538g, which is higher than the Salsiccia pizza (529g/100g), but the total GHG emission per serving of Colazione is about only ⅙ of the GHG emissions of a Salsiccia pizza. To make the label easier for interpretation by the customer and align with the goal of reducing total GHG emissions, we chose to assign labels based on total GHG emissions per serving of the products instead of per 100g, see Figure 8.
The label cut-offs for the Summer Pilot (Phase 1) are shown in Figure 9. GHGs are evaluated based on meal categories (lunch/dinner, breakfast, or desserts/snacks). Menu items are categorized as green, yellow, or red, depending on whether they have below or above average GHG emissions compared to other food items sold at Mercante within the same meal category. The categories also consider if food items have low enough emissions to achieve UBC’s food emissions targets.

Figure 8: GHG Emissions (Kg) Per Serving vs. Per 100g (Phase 1)

The label cut-offs for the Summer Pilot (Phase 1) are shown in Figure 9. GHGs are evaluated based on meal categories (lunch/dinner, breakfast, or desserts/snacks). Menu items are categorized as green, yellow, or red, depending on whether they have below or above average GHG emissions compared to other food items sold at Mercante within the same meal category. The categories also consider if food items have low enough emissions to achieve UBC’s food emissions targets.

Figure 9: Phase 1 Label Cut-offs
3.2 FALL PILOT (PHASE 2)

The Fall Pilot (Phase 2) for the operationalization of the CFFS Label continued at Mercante and also expanded to Open Kitchen, which is one of the three major UBCFS residence dining halls open during the 2021-2022 academic year. Due to the increased variety and quantity of food groups offered by Open Kitchen, we decided to assign the CFFS Label based on the amount of GHG emissions per 100 grams of food. This allows us to compare the GHG emissions of foods with different serving sizes and ingredient components using a single set of thresholds. The weight, GHG emissions per serving of food, and GHG emissions per 100 grams of food were calculated by the evaluation framework for assigning labels.

To determine the label cut-offs, we updated the UBCFS GHG emissions baseline using 2019 sales and recipes data. See Table 1 for the estimated baseline for each student residence dining room and UBCFS in total:

<table>
<thead>
<tr>
<th>Location</th>
<th>19-20 Baseline GHG Emissions</th>
<th>UBC 19-20 GHG Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Kitchen</td>
<td>462.81g/per 100g</td>
<td></td>
</tr>
<tr>
<td>Totem</td>
<td>311.25g/per 100g</td>
<td>360.25g/per 100g</td>
</tr>
<tr>
<td>Gather</td>
<td>306.96g/per 100g</td>
<td></td>
</tr>
<tr>
<td>Mercante</td>
<td>416.71g/per 100g</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: UBC 19-20 GHG Emissions Baseline (Phase 2)

The label cut-offs for the Fall Pilot (Phase 2) are shown in Figure 10 below. By comparing the mean and medium GHG emissions per 100g of menu items offered at Open Kitchen during the 2021-2022 academic year, we can see that the mean GHG emissions per 100g (369.3g CO2eq) is above the baseline GHG emissions. Therefore, applying the CFFS Label on menu items would likely reduce the average GHG emissions if the label is effective in reducing the purchase of red label menu items. The median GHG emissions per 100g is 232.1g CO2eq, roughly halfway in between the 2 cut-off points that lie in the yellow range. Thus, the label cut-offs would make the quantity of menu items that are labeled as green, yellow, and red approximately the same.
The plots in Figure 11 show the distribution of total GHG emissions and GHG emissions per 100g of menu items at Open Kitchen during the 2021-2022 academic year. The plots also indicate the mean and median GHG emissions per dish and per 100g of food.

Figure 11: GHG Emissions (g) Per Serving vs. Per 100g (Phase 2)

3.3 SPRING PILOT (PHASE 3)

The Spring Pilot (Phase 3) for the operationalization of the CFFS Label continued at Open Kitchen after evaluating the effects of the CFFS Label on students’ behavior during the Fall Pilot (Phase 2). This time, the CFFS research teams decided to simplify the label system by only applying the CFFS Label to menu items that fit the criteria of climate-friendliness. While we keep evaluating the footprints of menu items on a per 100g basis, we incorporated the nitrogen and water footprints into consideration after the sensitivity analysis and produced a composite label metric to decide which menu item gets the CFFS Label assigned. The steps 1-3 of the sensitivity analysis allow us to see which attribute should be incorporated by checking the number of items that the
combination of attributes (GHG, Nitrogen, and Water) results in the same "color" of impact. See Table 2 for the results of the sensitivity analysis steps 1-3:

<table>
<thead>
<tr>
<th>Attributes Combination</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHG, Nitrogen, Freshwater, Stress-weighed Water</td>
<td>175</td>
<td>28.69%</td>
</tr>
<tr>
<td>GHG, Nitrogen, Freshwater</td>
<td>215</td>
<td>35.25%</td>
</tr>
<tr>
<td>GHG, Nitrogen, Stress-weighed Water</td>
<td>214</td>
<td>35.08%</td>
</tr>
</tbody>
</table>

Table 2: Sensitivity Analysis 1-3 Results

The above results suggest that choosing between freshwater and stress-weighed water use results in more consistency in the "color" of impact. And we chose the stress-weighed water use because it takes water scarcity into account. Therefore, the composite metric includes the three attributes: GHG, nitrogen, and stress-weighed water use. The results of the sensitivity analysis steps 4-5 are shown in Table 3 below:

<table>
<thead>
<tr>
<th>Metric-Color</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHG-Red</td>
<td>158</td>
</tr>
<tr>
<td>GHG-Yellow</td>
<td>204</td>
</tr>
<tr>
<td>GHG-Green</td>
<td>248</td>
</tr>
<tr>
<td>Composite-Red</td>
<td>189</td>
</tr>
<tr>
<td>Composite-Yellow</td>
<td>175</td>
</tr>
<tr>
<td>Composite-Green</td>
<td>246</td>
</tr>
</tbody>
</table>

Table 3: Label Counts in GHG-only and Composite Metric

According to Table 3, the distribution of label colors is balanced under both the GHG-only and the composite metric. And Table 4 indicates that the total number of menu items that changed color after incorporating nitrogen and water footprints is small (13.93%).

<table>
<thead>
<tr>
<th>GHG -&gt; Composite</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red -&gt; Yellow</td>
<td>7</td>
</tr>
</tbody>
</table>
Table 4: Label Change in GHG-only and Composite Metric

<table>
<thead>
<tr>
<th>Change</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red -&gt; Green</td>
<td>2</td>
</tr>
<tr>
<td>Yellow -&gt; Green</td>
<td>16</td>
</tr>
<tr>
<td>Green -&gt; Yellow</td>
<td>20</td>
</tr>
<tr>
<td>Green -&gt; Red</td>
<td>0</td>
</tr>
<tr>
<td>Yellow -&gt; Red</td>
<td>40</td>
</tr>
<tr>
<td><strong>Total Changed</strong></td>
<td><strong>85 (13.93%)</strong></td>
</tr>
</tbody>
</table>

The estimated UBC 19-20 baseline for GHG emissions, nitrogen footprint, and water footprints per 100g of food for the composite metric is shown in Table 5 below:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>UBC 19-20 Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHG Emissions</td>
<td>381.13 g/per 100g</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>4.21 g/per 100g</td>
</tr>
<tr>
<td>Fresh Water</td>
<td>47.16 L/per 100g</td>
</tr>
<tr>
<td>Stress-weighed Water</td>
<td>1501.2 L/per 100g</td>
</tr>
</tbody>
</table>

Table 5: UBC 19-20 GHG, Nitrogen, Water Footprints Baselines (Phase 3)

Using the estimated baseline above and calculating a composite metric for each menu item, there are about one-third (246) of the menu items classified as Climate-Friendly (green in the composite label). Almost the same proportion of green labels as in the Fall Pilot (Phase 2).

The estimation of the Daily Allowance Value (DV) for each attribute in the composite metric is calculated using the 50% of UBC 19-20 Baseline to define as Climate-Friendly. The estimated DV is based on a standard healthy diet from the Food Label Toolkit by Leach et al. (2016), which is set as 2200 calories per day, see Table 6 for detailed food components of the healthy diet and Table 7 for corresponding DV for each CFFS attribute calculated:

---

4 The estimated UBC 19-20 GHG emissions baseline in Phase 3 is updated as many preps get units estimated after analyzing the newly-extracted open kitchen data for phase 3, thus the units information database expanded. Many old recipes in 2019 also used these preps so they were included in the Phase 3 baseline estimation but excluded in Phase 2 (Phase 2’s baseline is a rough estimate that only included menu items that didn’t use preps with odd units).
<table>
<thead>
<tr>
<th>Food Component</th>
<th>Grams Per Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicken</td>
<td>40</td>
</tr>
<tr>
<td>Pork</td>
<td>20</td>
</tr>
<tr>
<td>Beef</td>
<td>20</td>
</tr>
<tr>
<td>Milk</td>
<td>280</td>
</tr>
<tr>
<td>Cheese</td>
<td>30</td>
</tr>
<tr>
<td>Eggs</td>
<td>30</td>
</tr>
<tr>
<td>Fish</td>
<td>30</td>
</tr>
<tr>
<td>Grains</td>
<td>120</td>
</tr>
<tr>
<td>Rice</td>
<td>40</td>
</tr>
<tr>
<td>Fruits</td>
<td>220</td>
</tr>
<tr>
<td>Beans</td>
<td>40</td>
</tr>
<tr>
<td>Potatoes</td>
<td>100</td>
</tr>
<tr>
<td>Vegetables</td>
<td>200</td>
</tr>
<tr>
<td>Nuts</td>
<td>10</td>
</tr>
<tr>
<td>Oils</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1200</strong></td>
</tr>
</tbody>
</table>

Table 6: Standard Healthy Diet (2200 Calories Per Day). Source: Food Label Toolkit by Leach et al. (2016)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Daily Allowance Value (DV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHG Emissions</td>
<td>3037.98g</td>
</tr>
<tr>
<td>Nitrogen</td>
<td>41.73g</td>
</tr>
<tr>
<td>Fresh Water</td>
<td>434.11L</td>
</tr>
<tr>
<td>Stress-weighed Water</td>
<td>11730.57L</td>
</tr>
</tbody>
</table>

Table 7: Daily Allowance Value (DV)
4. DISCUSSION

From the above analysis, we can see that foods that contain ruminant meat and dairy products (i.e. beef, lamb, cheese, etc.) tend to have high GHG emissions, nitrogen and also water footprints per serving, and in particular per 100g. This suggests one way to lower the climate impact of the food system is by reducing the amount of ruminant meat and dairy consumption and switching to plant-based protein products (i.e., beans, tofu, etc.). For example, the difference between the Salsiccia Pizza (the pizza with the highest GHG emissions at Mercante with chorizo, tomato, basil, oregano, and mozzarella) and the Beyond BBQ Pizza (the pizza with the lowest GHG emissions at Mercante with beyond meat crumble, chipotle BBQ sauce, arugula, and mushrooms) is 2,463 grams of CO2eq, which is equivalent to the emissions from an 11.96-kilometer drive in an average passenger vehicle (average of 206g CO2 emissions per km driven, Canada Energy Regulator, 2019).

The evaluation framework is tailored for what UBC has in place, such as the inventory management system OC and Nutrislice. It maximized the efficiency of implementing and evaluating the CFFS Label at UBCFS outlets and various associated displaying platforms to utilize the opportunity of reaching thousands of customers daily with reliable information collected for analysis. The systemized workflow from implementation to evaluation allows the rapid adjustment of the label for better influence on the Climate-Friendly purchasing behavior of customers. On the other hand, the specificity of the evaluation framework to UBCFS’s organization and systems may limit the expansion of the CFFS Label to other institutions and places whose recipe management systems are different or there’s no systemized way of storing the recipes’ data. While the basic methodologies and supporting data can be transferred, the efficiency of expanding the CFFS Label to those places is limited and adjustments to the framework should be considered based on the amount and data structure of the recipes for evaluation.

In addition, the manual process involved in the evaluation process allows for better accuracy and flexibility in adjusting the labelling methodologies and incorporating more comprehensive considerations in evaluating the climate impact of some menu items. While it sacrifices the efficiency and automaticity of the framework and may limit the expansion to larger scale implementation of the CFFS Label.
There are additional limitations of the evaluation framework. First, there are several processed products and packaged foods that are directly purchased from external suppliers, such as sauces, dressings, snacks, etc. Therefore, the evaluation can only approximate their GHG emission factors by estimating the proportions of the ingredients contained in these products.

Secondly, emissions from bucket items such as "parfait," "salad bar," and "build your own" represent an average with a lot of variances since they are customized by the client at the retail venue. The recipes for these products recorded in the system use the estimated average amount for each composition that customers may choose.

Thirdly, there is human dependence on matching items with associated emission factors. Although manually matching takes less time and is more accurate, this may raise some problems if the label is expanded to more food venues and thus human work will take more time. Besides, the information for ingredients stored in OC is incomplete for some items, such as the unit information and conversion data, which needs to be adjusted and inserted manually.

Lastly, the relevance of this methodology and effort rests on the effectiveness of the label in educating and ultimately changing consumer choice among the UBC population eating food sold on campus. Establishing these methods has been the result of countless meetings and engagement with stakeholders and experts from the UBC community and beyond. Implementation and fine-tuning of the methodology has taken more than 1000 hours of work and does not take into account the time committed by the stakeholders who gave feedback and provided valuable data. These aspects need to be taken into account and compared to the effectiveness of improving the climate friendliness of food eaten at UBC primarily through supply-side solutions, i.e., the food procured and offered at UBC—which includes UBCFS and multiple other venues.
5. RECOMMENDATIONS

The steps outlined below could be taken for future development and expansion of the CFFS Labelling pilot to improve the evaluation framework and make it more resilient and suitable for expanded operations.

5.1 SHORT-TERM RECOMMENDATIONS (< 3 MONTHS)

Data Analysis Aspect:

- Integrate the CFFS attribute weighing process into the evaluation framework to automate the process.
- Conduct research on introducing biodiversity as a new CFFS attribute for evaluating menu items.

Operational Work Aspect:

- Display the carbon, nitrogen, and water footprint DV and/or percentile per dish and/or per 100g of food on Nutrislice menu item descriptions.
- Evaluate the benefit of displaying the CFFS Label on the Nutrislice Platform with the all-inclusive dining services.
- Evaluate the possibility and interest of adding new attributes to the CFFS Label calculation.

5.2 MID-TERM RECOMMENDATIONS (< 6 MONTHS)

Data Analysis Aspect:

- Improve the recording and tracking of food information stored in the inventory management system and reduce the amount of missing data for ingredients and recipes.
- Incorporate the climate footprint data for ingredients into the inventory management system if feasible to embed the calculation process within the system.
- Streamline the process of data extraction from UBCFS and data output to display CFFS Label information on Nutrislice.
- Conduct research on the GHG emissions from the cooking process for improved reporting and evaluation of menu items.
Conduct research on the water use footprint from the prepping and cooking process for improved reporting and evaluation of menu items.

**Operational Work Aspect:**

- Implement the CFFS Label at all student residence dining halls managed by UBCFS.
- Have other research to support the expansion of the CFFS Label to other food outlets at UBC.

### 5.3 LONG-TERM RECOMMENDATIONS (< 1 YEAR)

**Data Analysis Aspect:**

- UBC can lead the engagement process to build a Pacific Northwest/Canadian specific GHGe, nitrogen, and water footprint factors database by conducting research together with peer institutions. This can also help to improve the accuracy and specificity of current labels.
- Collaborate closely with peer institutions to have a standardized method of menu item evaluation and labeling criteria across different universities and institutions in North America (e.g., Oxford’s Health Behaviors team at the Livestock, Environment and People (LEAP) project).

**Operational Work Aspect:**

- Evaluate the possibility of adding the CFFS Label to UBC delivery menus.
- Expand the CFFS Label implementation to other UBC campuses (i.e., UBC Okanagan).
In conclusion, the CFFS Label evaluation framework is a resilient approach to conducting the evaluation process in an efficient and structured way that meets the needs for the future expansion of the CFFS Label across UBCFS. However, there are a few limitations in the current framework due to the tailored design of the framework to UBCFS and the manual reliance on cleaning, assigning, and extracting data. The recommendation for the next steps is to streamline the extraction process and improve the tracking of ingredient information in the systems. It will require more time, resourcing, and close coordination between associated departments to produce a comprehensive CFFS Label that indicates all-around information on the climate impact of menu items sold by UBCFS.
REFERENCES


https://doi.org/10.1038/s43016-021-00225-9


https://doi.org/10.1016/j.foodpol.2016.03.006


APPENDICES

APPENDIX A [CODE FOR EVALUATION FRAMEWORK]

Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang, CFFS Data Analyst

Part I: Data Preprocessing

Set up and Import Libraries

```python
#pip install -r requirements.txt

In [2]: import numpy as np
import pandas as pd
import pg8000 as ppg
import matplotlib.pyplot as plt
import glob
import os
import cv2
from itertools import islice
from decimal import Decimal
import xml.etree.ElementTree as et
from xml.etree.ElementTree import parse
import openpyxl
import json

In [3]: # Set the root path, change the current working directory into the project folder
path = "C:\Users\silvia\cffs-label"
os.chdir(path)

In [4]: # Enable reading data table in the scrolling window if you prefer
#pd.set_option('display.max_rows', None, 'display.max_columns', None)

Load Data Files

Set Data File Path

```python
In [5]: # Select data file path for the chosen venue and time range where the recipes data stored
filepath_list = [glob.glob(os.path.join(os.getcwd(), "data", "raw", "21-22", "*.csv"))]

Out[5]:
['C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04182021_9938.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_9918.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_1141.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_1301.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_1555.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_1007.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_1207.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_1451.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04232021_1555.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_0933.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_1150.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_1351.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_1503.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_1946.csv',
 'C:\Users\silvia\cffs-label\data\raw\21-22\IPR_Export_04192021_1946.csv']

Import Items List

```python
In [6]: # Read items.xml files in the filepath_list and construct a dataframe
ItemDict = {}
Description = []
CaseQty = []
CaseCOK = []
PkgQty = []
PkgCOK = []
InventoryGroup = []

for filepath in filepath_list:
    ...:
```

```
```python
import pandas as pd
import xml.etree.ElementTree as ET

# XML path
path = '/items.xml'

# Read XML file
xml_tree = ET.parse(path)
root = xml_tree.getroot()

# Extract data
items = pd.DataFrame({
    'ItemId': [item.attrib.get('id') for item in root.findall('item')],
    'Description': [item.findtext('Description') for item in root.findall('item')],
    'CaseQty': [int(item.findtext('CaseQty')) for item in root.findall('item')],
    'CaseUOM': [item.findtext('CaseUOM') for item in root.findall('item')],
    'PakQty': [item.findtext('PakQty') for item in root.findall('item')],
    'PakUOM': [item.findtext('PakUOM') for item in root.findall('item')],
    'InventoryGroup': [item.findtext('InventoryGroup') for item in root.findall('item')]
}).drop_duplicates()

# Reset index
items.reset_index(drop=True, inplace=True)

# Show data frame
items
```

```python
import pandas as pd
import xml.etree.ElementTree as ET

# XML path
path = '/items.xml'

# Read XML file
xml_tree = ET.parse(path)
root = xml_tree.getroot()

# Extract data
items = pd.DataFrame({
    'ItemId': [item.attrib.get('id') for item in root.findall('item')],
    'Description': [item.findtext('Description') for item in root.findall('item')],
    'CaseQty': [int(item.findtext('CaseQty')) for item in root.findall('item')],
    'CaseUOM': [item.findtext('CaseUOM') for item in root.findall('item')],
    'PakQty': [item.findtext('PakQty') for item in root.findall('item')],
    'PakUOM': [item.findtext('PakUOM') for item in root.findall('item')],
    'InventoryGroup': [item.findtext('InventoryGroup') for item in root.findall('item')]
}).drop_duplicates()

# Reset index
items.reset_index(drop=True, inplace=True)

# Show data frame
items
```

```python
# Save the dataframe to CSV
path = os.path.join(os.getcwd(), 'data', 'preprocessed', 'items_list.csv')
items.to_csv(path, index=False, header=True)
```

```python
# Read ingredients.xml file in the filepath_list and construct a dataframe
ingredients_list = []
Conversion = []
InvFactor = []
Qty = []
Recipe = []
Usn = []

for filepath in filepath_list:
    path = filepath + '/ingredients.xml'
    if os.path.isfile(path):
        xml_tree = ET.parse(path)
        root = xml_tree.getroot()

        for x in root.iterfind('Ingredient'):
            IngredientId = x.attrib.get('ingredientId')
            Conversion = x.attrib.get('conversion')
            InvFactor = x.attrib.get('invFactor')
            Qty = x.attrib.get('qty')

            ingredients_list.append((IngredientId, Conversion, InvFactor, Qty))

ingredients_df = pd.DataFrame(ingredients_list, columns=['IngredientId', 'Conversion', 'InvFactor', 'Qty'])
```

```python
import pandas as pd
import xml.etree.ElementTree as ET

# XML path
path = '/ingredients.xml'

# Read XML file
xml_tree = ET.parse(path)
root = xml_tree.getroot()

# Extract data
ingredients = [(x.attrib.get('ingredientId'), x.attrib.get('conversion'), x.attrib.get('invFactor'), x.attrib.get('qty')) for x in root.iterfind('Ingredient')]

# Create DataFrame
ingredients_df = pd.DataFrame(ingredients, columns=['IngredientId', 'Conversion', 'InvFactor', 'Qty'])
```

Recipe.append(x.attrib['recipe'])
Uom.append(x.attrib['UOM'])

Ingredients = pd.DataFrame({'IngredientId': IngredientId, 'Qty': Qty, 'Uom': Uom, 'Conversion': Conversion, 'InvFactor': InvFactor, 'Recipe': Recipe}).drop_duplicates()

Ingredients.reset_index(drop=True, inplace=True)

In [12]:

<table>
<thead>
<tr>
<th>IngredientId</th>
<th>Qty</th>
<th>Uom</th>
<th>Conversion</th>
<th>InvFactor</th>
<th>Recipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.000 Kg</td>
<td>1.000000000</td>
<td>1.000000</td>
<td>P-10241</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1.000 Kg</td>
<td>1.000000000</td>
<td>0.3058</td>
<td>P-10496</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.370 Kg</td>
<td>2.204632000</td>
<td>0.6942</td>
<td>P-10496</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.500 Kg</td>
<td>1.000000000</td>
<td>1.2800</td>
<td>P-13933</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.000 BUNCH</td>
<td>1.000000000</td>
<td>0.0063</td>
<td>P-18318</td>
<td></td>
</tr>
</tbody>
</table>
| ...          | ... | ...     | ...       | ...      | ...
| 5377         | 170.000 g   | 0.001000000 | 1.000000 | R-62022 |
| 5378         | 140.000 g   | 0.09200462  | 1.000000 | R-62022 |
| 5379         | 3.000 g     | 1.000000000 | 1.000000 | R-62022 |
| 5380         | 180.000 g   | 0.001000000 | 1.000000 | R-62022 |
| 5381         | 90.000 g    | 0.001000000 | 1.000000 | R-62022 |

5382 rows x 6 columns

In [13]:

Ingredients.shape

Out[13]:

(5382, 6)

In [14]:

Ingredients.dtypes

Out[14]:

<table>
<thead>
<tr>
<th>IngredientId</th>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qty</td>
<td>object</td>
</tr>
<tr>
<td>Uom</td>
<td>object</td>
</tr>
<tr>
<td>Conversion</td>
<td>object</td>
</tr>
<tr>
<td>InvFactor</td>
<td>object</td>
</tr>
<tr>
<td>Recipe</td>
<td>object</td>
</tr>
<tr>
<td>dtype</td>
<td>object</td>
</tr>
</tbody>
</table>

In [15]:

# Save the dataframe to csv
path = os.path.join(os.getcwd(), 'data', 'preprocessed', 'ingredients_list.csv')
Ingredients.to_csv(path, index = False, header = True)

Import Preps List

In [16]:

# Read preps.xml files in the filepath_list and construct a dataframe
Preps = []
Description = []
PakQty = []
PakUOM = []
InventoryGroup = []

for filepath in filepath_list:
    path = filepath = '/Preps.xml'
    if os.path.isfile(path):
        tree = et.parse(path)
        root = tree.getroot()
        for Prep in root.findall('Prep'):
            PrepId.append(x.attrib['id'])
            Description.append(x.findtext('Description'))
            PakQty.append(x.findtext('PakQty'))
            PakUOM.append(x.findtext('PakUOM'))
            InventoryGroup.append(x.findtext('InventoryGroup'))

Preps = pd.DataFrame({'PrepId': PrepId, 'PrepName': Description, 'PakQty': PakQty, 'PakUOM': PakUOM, 'InventoryGroup': InventoryGroup}).drop_duplicates()

Preps.reset_index(drop=True, inplace=True)

In [17]:

Preps

Out[17]:

<table>
<thead>
<tr>
<th>PrepId</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 P-5059</td>
<td>BAKEDI løs og Spin Mushroom</td>
<td>5.550</td>
<td>Kg</td>
<td></td>
</tr>
<tr>
<td>1 P-54666</td>
<td>BAKEDPastaChickn Alfredo</td>
<td>6.176</td>
<td>Kg</td>
<td></td>
</tr>
<tr>
<td>Prepid</td>
<td>Description</td>
<td>PakQty</td>
<td>PakUOM</td>
<td>InventoryGroup</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------</td>
<td>--------</td>
<td>--------</td>
<td>----------------</td>
</tr>
<tr>
<td>2</td>
<td>BAKED PastaChorizo Penne</td>
<td>7.360</td>
<td>Kg</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>BAKED PastaShrimp Pesto</td>
<td>8.760</td>
<td>Kg</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BATCH Shrimp Remoulade</td>
<td>1.600</td>
<td>Kg</td>
<td></td>
</tr>
<tr>
<td>748</td>
<td>MOCheddar</td>
<td>2.000</td>
<td>Kg</td>
<td>PREP</td>
</tr>
<tr>
<td>749</td>
<td>ROASTED</td>
<td>spaghetti Squash</td>
<td>1.400</td>
<td>Kg</td>
</tr>
<tr>
<td>750</td>
<td>SAUTÉE</td>
<td>Cauliflower Rice</td>
<td>1.000</td>
<td>Kg</td>
</tr>
<tr>
<td>751</td>
<td>YIELD</td>
<td>Drated Pear</td>
<td>800.000</td>
<td>g</td>
</tr>
<tr>
<td>752</td>
<td>YIELD</td>
<td>Lettuce bun</td>
<td>3.000</td>
<td>PTN</td>
</tr>
</tbody>
</table>

753 rows x 5 columns

```r
In [18]: Preps.shape
Out[18]: (753, 5)
```

```r
In [19]: Preps.dtypes
Out[19]:
Prepid    object
Description    object
PakQty        object
PakUOM        object
InventoryGroup object
dtypes: object
```

```r
In [20]: # Save the dataframe to csv
    path = os.path.join(os.getcwd(), "data", "preprocessed", "Preps_list.csv")
    Preps.to_csv(path, index = False, header = True)
```

Import Products List

```r
In [21]: # Read products.xml files in the filepath_list and construct a dataframe
    Products = []
    for file in filepath_list:
        tree = ET.parse(file)
        root = tree.getroot()
        for x in root.iter('Prod'):
            Prodid = x.attrib['Id']
            Description = x.findtext('Description')
            SalesGroup = x.findtext('SalesGroup')
            Products.append([Prodid, Description, SalesGroup])
    Products = pd.DataFrame(Products, columns=['Prodid', 'Description', 'SalesGroup']).drop_duplicates()
    Products.reset_index(drop=True, inplace=True)
```

```r
In [22]: Products
Out[22]:
<table>
<thead>
<tr>
<th>Prodid</th>
<th>Description</th>
<th>SalesGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ALFFlatbread4 Cheese</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>1</td>
<td>ALFFlatbreadApple &amp; Pancotto</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>2</td>
<td>ALFFlatbread BBQ Chicken</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>3</td>
<td>ALFFlatbread Bruschetta</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>4</td>
<td>ALFFlatbread Caprese</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>453</td>
<td>SQR Tofu Sofrito Quesadilla +1</td>
<td>OK - SQUARE MEAL</td>
</tr>
<tr>
<td>454</td>
<td>SQR Tofu Sofrito Quesadilla +2</td>
<td>OK - SQUARE MEAL</td>
</tr>
<tr>
<td>455</td>
<td>SQR Vegan Lettuce Wrap</td>
<td>OK - SQUARE MEAL</td>
</tr>
<tr>
<td>456</td>
<td>SQR Vegan Lettuce Wrap +1</td>
<td>OK - SQUARE MEAL</td>
</tr>
<tr>
<td>487</td>
<td>SQR Vegan Lettuce Wrap +2</td>
<td>OK - SQUARE MEAL</td>
</tr>
</tbody>
</table>
```

458 rows x 3 columns
In [33]: Products.shape

Out[33]: (458, 3)

In [34]: Products.dtypes

Out[34]:

<table>
<thead>
<tr>
<th></th>
<th>dtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProdId</td>
<td>object</td>
</tr>
<tr>
<td>Description</td>
<td>object</td>
</tr>
<tr>
<td>SaleDesc</td>
<td>object</td>
</tr>
</tbody>
</table>

In [35]: # Save the dataframe to csv
path = os.path.join(os.getcwd(), 'data', 'preprocessed', 'products_list.csv')
products.to_csv(path, index = False, header = True)

Import Conversions List

In [36]: # Read conventions.xml file in the filepath_list and construct a dataframe
ConversionsId = []
Multiplier = []
ConvertFromQty = []
ConvertToUom = []
ConvertToQty = []

for filepath in filepath_list:
    path = filepath + '/conversions.xml'
    if os.path.isfile(path):
        tree = ET.parse(path)
        root = tree.getroot()
        for x in root.iterfind('Conversion'):
            ConversionsId.append(x.attrib['id'])
            Multiplier.append(x.attrib['multiplier'])
            ConvertFromQty.append(x.find('ConvertFrom').attrib['qty'])
            ConvertToUom.append(x.find('ConvertTo').attrib['uom'])
            ConvertToQty.append(x.find('ConvertTo').attrib['qty'])
            Conversions = pd.DataFrame({'ConversionId': ConversionsId, 'Multiplier': Multiplier, 'ConvertFromQty': ConvertFromQty, 'ConvertFromUom': ConvertFromUom, 'ConvertToQty': ConvertToQty, 'ConvertToUom': ConvertToUom})
            Conversions.drop_duplicates()
Conversions.reset_index(drop=True, inplace=True)

In [37]: Conversions

<table>
<thead>
<tr>
<th>ConversionId</th>
<th>Multiplier</th>
<th>ConvertFromQty</th>
<th>ConvertFromUom</th>
<th>ConvertToQty</th>
<th>ConvertToUom</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00010000</td>
<td>1.0000</td>
<td>XXX</td>
<td>1.0000</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>0.87719298</td>
<td>1.0000</td>
<td>1.14L</td>
<td>1.1400</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>0.86666667</td>
<td>1.0000</td>
<td>1.5L</td>
<td>1.5000</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>0.87428571</td>
<td>1.0000</td>
<td>1.75L</td>
<td>1.7500</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>0.80000000</td>
<td>1.0000</td>
<td>2L</td>
<td>2.0000</td>
<td>L</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>291</td>
<td>1.3334</td>
<td>0.32250865</td>
<td>1.0000</td>
<td>Tbsp</td>
<td>3.1000</td>
</tr>
<tr>
<td>292</td>
<td>1.3390</td>
<td>0.22222222</td>
<td>1.0000</td>
<td>tsp</td>
<td>4.5000</td>
</tr>
<tr>
<td>293</td>
<td>1.3390</td>
<td>0.07407407</td>
<td>1.0000</td>
<td>Tbsp</td>
<td>13.5000</td>
</tr>
<tr>
<td>294</td>
<td>1.3390</td>
<td>0.00402903</td>
<td>1.0000</td>
<td>cup</td>
<td>216.0000</td>
</tr>
<tr>
<td>295</td>
<td>1.3390</td>
<td>0.00409060</td>
<td>1.0000</td>
<td>es</td>
<td>202.0000</td>
</tr>
</tbody>
</table>

296 rows x 6 columns

In [38]: Conversions.shape

Out[38]: (296, 6)

In [39]: Conversions.dtypes

Out[39]:

<table>
<thead>
<tr>
<th>ConversionId</th>
<th>Multiplier</th>
<th>ConvertFromQty</th>
<th>ConvertFromUom</th>
<th>ConvertToQty</th>
<th>ConvertToUom</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>object</td>
<td>object</td>
<td>object</td>
<td>object</td>
<td>object</td>
</tr>
</tbody>
</table>

dtype: object
Data Summary

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>598</td>
<td>7</td>
</tr>
<tr>
<td>Preps</td>
<td>753</td>
<td>5</td>
</tr>
<tr>
<td>Ingredients</td>
<td>5382</td>
<td>6</td>
</tr>
<tr>
<td>Products</td>
<td>458</td>
<td>3</td>
</tr>
<tr>
<td>Conversions</td>
<td>296</td>
<td>6</td>
</tr>
</tbody>
</table>
Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang, CFFS Data Analyst

Part II: Data Cleaning

Set up and Import Libraries

```python
In [1]: # pip install -r requirements.txt

In [3]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob
import os
import cv2
from itertools import iallce
from decimal import Decimal
import xml.etree.ElementTree as et
from xml.etree.ElementTree import parse
import os.path
import pywt
from datetime import datetime

In [3]: # Set the root path, change the current working directory into the project folder
   path = '/Users/silvia/cffs-label'
   os.chdir(path)

In [4]: # Enable reading data table in the scrolling window if you prefer
   #pd.set_option('display.max_rows', None, 'display.max_columns', None)
```

Import Preprocessed Datasets

```python
In [5]: # Read Items_List.csv
   Items = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Items_List.csv"))
   Items.head()

Out[5]:
   ItemId   Description  CaseQty  CaseUOM  PakQty  PakUOM  InventoryGroup
0  1-6271   APPLES GRANNY SMITH   113.0   ea  1.0   CT   PRODUCE
1  1-6971  ARTICHOKE 1/4 SALAD CUT TFC   6.0   LG CAN  2.5   Kg   PRODUCE
2  1-2305  BACON PANCETTA  1.0   Kg  1.0   Kg   MEAT
3  1-1207  BAGUETTE FRENCH  24.0   each  1.0   CT   BREAD
4  1-17203  BALSAMIC GLAZE  2.0   bottle  2.0   L   FOOD - GROCERY

In [6]: Items.describe()

Out[6]:
```

In [7]: Items.shape

Out[7]: (508, 7)
```

# Read Ingredients_List.csv

```python
In [8]: Ingredients = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Ingredients_List.csv"))
   Ingredients.head()

Out[8]:
   IngredientId  Qty
0        float64
```

In [8]: IngredientId  Qty
0        float64
Dum      object
Conversion float64
In [9]: Ingredients.head()

<table>
<thead>
<tr>
<th>IngredientId</th>
<th>Qty</th>
<th>Uom</th>
<th>Conversion</th>
<th>InvFactor</th>
<th>Recipe</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1-18746</td>
<td>1.00</td>
<td>Kg</td>
<td>1.000000</td>
<td>1.0000</td>
<td>P-18241</td>
</tr>
<tr>
<td>1 1-3388</td>
<td>1.00</td>
<td>L</td>
<td>1.000000</td>
<td>0.3058</td>
<td>P-10496</td>
</tr>
<tr>
<td>2 1-4660</td>
<td>2.27</td>
<td>Kg</td>
<td>2.20462</td>
<td>0.6942</td>
<td>P-10496</td>
</tr>
<tr>
<td>3 1-3451</td>
<td>2.56</td>
<td>L</td>
<td>1.000000</td>
<td>1.2800</td>
<td>P-13933</td>
</tr>
<tr>
<td>4 1-4679</td>
<td>1.00</td>
<td>BUNCH</td>
<td>1.000000</td>
<td>0.0083</td>
<td>P-19318</td>
</tr>
</tbody>
</table>

Out[9]:

In [10]: Ingredients.shape

Out[10]:

In [11]: # Read Preps_List.csv
    
    Preps = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Preps_list.csv"))

Out[11]: Preps.head()

<table>
<thead>
<tr>
<th>Prepid</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BAKEDLasagnaSpinMushroom</td>
<td>5.550</td>
<td>Kg</td>
<td>NaN</td>
</tr>
<tr>
<td>1</td>
<td>BAKEDPostaChickenAlfredo</td>
<td>6.176</td>
<td>Kg</td>
<td>NaN</td>
</tr>
<tr>
<td>2</td>
<td>BAKEDPastaChORIZOPenne</td>
<td>7.360</td>
<td>Kg</td>
<td>NaN</td>
</tr>
<tr>
<td>3</td>
<td>BAKEDPastaShrimpPesto</td>
<td>5.700</td>
<td>Kg</td>
<td>NaN</td>
</tr>
<tr>
<td>4</td>
<td>BACTSHrimpRemoulade</td>
<td>1.900</td>
<td>Kg</td>
<td>NaN</td>
</tr>
</tbody>
</table>

In [13]: Preps.shape

Out[13]:

In [14]: # Read Products_List.csv
    
    Products = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_list.csv"))

Out[14]: Products.head()

<table>
<thead>
<tr>
<th>ProdId</th>
<th>Description</th>
<th>SalesGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ALFLatbread1 Cheese</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>1</td>
<td>ALFLatbreadApple &amp; Pancetta</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>2</td>
<td>ALFLatbreadBBQ Chicken</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>3</td>
<td>ALFLatbreadBruschetta</td>
<td>OK - AL FORNO</td>
</tr>
<tr>
<td>4</td>
<td>ALFLatbreadCaprese</td>
<td>OK - AL FORNO</td>
</tr>
</tbody>
</table>

In [16]: Products.shape

Out[16]:

In [17]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Conversions_list.csv"))

Out[17]: Conversions.head()

<table>
<thead>
<tr>
<th>ConversionId</th>
<th>Multiplier</th>
<th>ConvertFromQty</th>
<th>ConvertToQty</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>float64</td>
<td>float64</td>
<td>float64</td>
</tr>
</tbody>
</table>
Update Conversion List

Out[20]:

<table>
<thead>
<tr>
<th>ConversionId</th>
<th>Multiplier</th>
<th>ConvertFromQty</th>
<th>ConvertFromUm</th>
<th>ConvertToQty</th>
<th>ConvertToUom</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NaN</td>
<td>1.0</td>
<td>XXX</td>
<td>1.00</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>NaN</td>
<td>1.0</td>
<td>1.4L</td>
<td>1.50</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>NaN</td>
<td>1.0</td>
<td>1.5L</td>
<td>2.00</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>NaN</td>
<td>1.0</td>
<td>1.75 L</td>
<td>2.00</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>NaN</td>
<td>1.0</td>
<td>2L</td>
<td>3.00</td>
<td>L</td>
</tr>
</tbody>
</table>

315 rows x 6 columns

In [21]:

for index, row in Update_Conv.iterrows():
    Conversions.drop(Conversions[Conversions['ConversionId'] == index].index, inplace = True)

In [22]:

frames = [Conversions, Update_Conv]
Conversions = pd.concat(frames).reset_index(drop=True, inplace=False).drop_duplicates()

In [23]:

Conversions

Out[23]:

<table>
<thead>
<tr>
<th>ConversionId</th>
<th>Multiplier</th>
<th>ConvertFromQty</th>
<th>ConvertFromUm</th>
<th>ConvertToQty</th>
<th>ConvertToUom</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NaN</td>
<td>1.000000</td>
<td>XXX</td>
<td>1.00</td>
<td>L</td>
</tr>
<tr>
<td>1</td>
<td>NaN</td>
<td>1.0</td>
<td>1.14L</td>
<td>1.50</td>
<td>L</td>
</tr>
<tr>
<td>2</td>
<td>NaN</td>
<td>1.0</td>
<td>1.5L</td>
<td>2.00</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>NaN</td>
<td>1.0</td>
<td>1.75 L</td>
<td>3.00</td>
<td>L</td>
</tr>
<tr>
<td>4</td>
<td>NaN</td>
<td>1.0</td>
<td>2L</td>
<td>4.00</td>
<td>L</td>
</tr>
<tr>
<td>585</td>
<td>P-32664</td>
<td>1.0</td>
<td>each</td>
<td>61.00</td>
<td>g</td>
</tr>
<tr>
<td>586</td>
<td>P-55707</td>
<td>0.065405</td>
<td>PTN</td>
<td>185.00</td>
<td>g</td>
</tr>
<tr>
<td>587</td>
<td>P-62933</td>
<td>0.065405</td>
<td>PTN</td>
<td>185.00</td>
<td>g</td>
</tr>
<tr>
<td>588</td>
<td>P-62023</td>
<td>0.065452</td>
<td>ROLL</td>
<td>155.00</td>
<td>g</td>
</tr>
</tbody>
</table>

550 rows x 6 columns
Create Unit Converter

In [24]:
```bash
path = os.path.join(os.getcwd(), "data", "cleaning", "Conversions_Added.csv")
Conversions_to_csv(path, index = False, header = True)
```

Create Unit Converter

In [25]:
```python
# Import standard unit conversion information and construct a dataframe
std_units = pd.read_csv(os.path.join(os.getcwd(), "data", "external", "standard_conversions.csv"))
std_units.head()
```

Out[25]:
```
   Multiplier ConvertFromQty ConvertFromUnit ConvertToQty ConvertToUnit
0        1.000000     4.928990       mi
1        1.000000    14.787000      m
2        2.204620    36.875000      m
3        1.000000    473.176250     mL
4        1.000000    28.349500     g
```

In [26]:
```python
# Separate units that converted to 'ml' or 'g'
liquid_units = std_units.loc[std_units["ConvertToUnit"] == 'ml', "ConvertFromUnit"].tolist()
solid_units = std_units.loc[std_units["ConvertToUnit"] == 'g', "ConvertFromUnit"].tolist()
```

In [27]:
```python
# Construct a standard unit converter

def std_converter(qty, uom):
    if uom in std_units["ConvertFromUnit"].tolist():
        multiplier = std_units.loc[std_units["ConvertToUnit"] == uom, "Multiplier"]
        qty = float(qty) * multiplier
    else:
        qty = qty
    return qty, uom
```

In [28]:
```python
# Test the std_converter
assert std_converter(6.25, 'lb') == (113.398, 'g')
```

In [29]:
```python
# Construct a unit converter for specific ingredients

scc_conv = list(filter(None, Conversions["ConversionId"].tolist()))

def scc_converter(ingre, qty, uom):
    if uom in liquid_units + solid_units:
        return std_converter(qty, uom)
    elif ingre in scc_conv:
        conversion = Conversions.loc[Conversions["ConversionId"] == ingre, & (Conversions["ConvertFromUnit"] == uom) & (Conversions["ConvertToUnit"] == 'g')]
        multiplier = conversion["Multiplier"]
        if multiplier.empty:
            return std_converter(qty, uom)
        else:
            qty = float(qty) / float(multiplier)
            return qty, uom
    else:
        return std_converter(qty, uom)
```

In [30]:
```python
# Test the scc_converter
assert scc_converter('I-1120', 1, 'CT') == (50, 'g')
```

Items with Non-standard Units

In [31]:
```python
# Filter out the items whose unit information is unknown

col_names = list(ingredients.columns.values)
Items_Nonstd = []

for index, row in ingredients.iterrows():
    ingre = ingredients.loc[index, "Ingredient"]
    uom = ingredients.loc[index, "Uom"]
    if ingre not in liquid_units + solid_units and ingre not in Conversions["ConversionId"] and uom not in Conversions["ConvertToUnit"]:
        Items_Nonstd.append((index, row))
```

Items_Nonstd = pd.DataFrame(Items_Nonstd, columns = col_names)
Items_Nonstd.drop_duplicates(subset=['IngredientId'], inplace=True)
Items_Nonstd
```

Out[31]:
```
   IngredientId Qty Uom Conversion InvFactor Recipe
```

42
Clean Preps Units

```python
In [33]:
preps['StdUom'] = np.nan
preps['Units'] = np.nan

In [34]:
# Convert uom to 'g' or 'ml' for each prep using the unit converter
for index in preps['Index']:
    PrepId = preps['Index'][index]['PrepId']
    Qty = preps['Index'][index]['Qty']
    uom = preps['Index'][index]['Uom']
    preps.loc[index, 'StdQty'] = np.min([float(q)*1000 if uom == 'g' else float(q) for q in Qty])
    preps.loc[index, 'StdUom'] = uom
```

```python
In [35]:
pd.DataFrame(preps)
```

```
Out[35]:
  PrepId  Description          PakQty PakUOM  InventoryGroup  StdQty  StdUom
0     0    BAKED Lasagna Spin M 5.550    Kg       NaN  5550.000000   g
1     1  BAKED Pasta Chicken A 6.178    Kg       NaN  6178.000000   g
2     2  BAKED Pasta Chorizo P 7.360    Kg       NaN  7360.000000   g
3     3  BAKED Pasta Shrimp P 6.790    Kg       NaN  6790.000000   g
4     4  BATH Shrimp Remoulade 1.000    Kg       NaN  1000.000000   g
...   ...            ...       ...    ...        ...        ...  ...
745  745    MIX Cheese           2.000    Kg       NaN  2000.000000   g
746  746  ROASTED Spaghetti Squ 1.400    Kg       NaN  1400.000000   g
747  747    SAUTE Cauliflower R 1.000    Kg       NaN  1000.000000   g
748  748   YIELD Grated Pear    800.000    g       NaN  800.000000    g
749  749  YIELD Lettuce bun      3.000    PTN      NaN  3000.000000   g
750  750
```

```python
In [36]:
# Save cleaned preps list to file
path = os.path.join(os.getcwd()), 'data', 'cleaning', 'Preps_Unit_Cleaned.csv')
preps.to_csv(path, index=False, header=True)
```

Get Preps with Nonstandard Units

```python
In [37]:
col_names = list(preps.columns.values)
preps_nonstd = []

for index, row in preps.iterrows():
    StdBox = preps['Box'][index]['StdBox']
    if StdBox not in ('g', 'ml'):
        dict = {}
        dict.update(dfrow)
        PrepNonstd.append(dict)

Preps_nonstd = pd.DataFrame([Preps_nonstd, columns = col_names])
```

```python
In [38]:
pd.DataFrame(preps_nonstd)
```

```
Out[38]:
  PrepId  Description          PakQty PakUOM  InventoryGroup  StdQty  StdUom
0     0    BAKED Lasagna Spin M 5.550    Kg       NaN  5550.000000   g
1     1  BAKED Pasta Chicken A 6.178    Kg       NaN  6178.000000   g
2     2  BAKED Pasta Chorizo P 7.360    Kg       NaN  7360.000000   g
3     3  BAKED Pasta Shrimp P 6.790    Kg       NaN  6790.000000   g
4     4  BATH Shrimp Remoulade 1.000    Kg       NaN  1000.000000   g
...   ...            ...       ...    ...        ...        ...  ...
745  745    MIX Cheese           2.000    Kg       NaN  2000.000000   g
746  746  ROASTED Spaghetti Squ 1.400    Kg       NaN  1400.000000   g
747  747    SAUTE Cauliflower R 1.000    Kg       NaN  1000.000000   g
748  748   YIELD Grated Pear    800.000    g       NaN  800.000000    g
749  749  YIELD Lettuce bun      3.000    PTN      NaN  3000.000000   g
750  750
```

```python
In [39]:
# Filter out prep with nonstandard uom but have information already
Manual_Prep = pd.read_csv(os.path.join(os.getcwd()), 'data', 'cleaning', 'update', 'Preps_UpdateUom.csv'))
col_names = list(Preps_nonstd.columns.values)
Preps_nonstd_na = []

for index, row in Preps_nonstd.iterrows():
    PrepId = Preps_nonstd['Index'][index]['PrepId']
    if PrepId not in Manual_Prep['PrepId'], values:
        dict = {}
        dict.update(dfrow)
        Preps_nonstd_na.append(dict)
```

43
```python
Preps_Nonstd = pd.DataFrame(Preps_Nonstd_na, columns = col_names
Preps_Nonstd

path = os.path.join(os.getcwd()), "data", "cleaning", "Preps_NonstdDay.csv"
Preps_Nonstd.to_csv(path, index = False, header = True)

New Items

# Load current Items List with assigned Emission Factors Category ID
Items_Assigned = pd.read_csv(os.path.join(os.getcwd()), "data", "mapping", "Items_List_Assigned.csv"
Items_Assigned.head()

Out[41]:

<table>
<thead>
<tr>
<th>ItemId</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1-57645</td>
<td>CHUCK FLAT BONELESS FZN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
</tr>
<tr>
<td>1</td>
<td>1-10869</td>
<td>BEEF STIRFRY COV FR</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
</tr>
<tr>
<td>2</td>
<td>1-7064</td>
<td>BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
</tr>
<tr>
<td>3</td>
<td>1-37005</td>
<td>BEEF MEATBALLS</td>
<td>4.54</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
</tr>
<tr>
<td>4</td>
<td>1-37002</td>
<td>BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
</tr>
</tbody>
</table>

Out[42]:

Items_Assigned.shape

Out[42]:

(1993, 8)

Get the List of New Items

# Filter new items by itemId that not in the database and output then in a dataframe
col_names = list(Items.columns.values)
New_Items_List = []

for index, row in Items.iterrows():
    ItemId = row.loc["itemId"]
    if ItemId not in Items_Assigned["itemId"].values:
        Dict = {}
        Dict.update(dict(row))
        New_Items_List.append(Dict)

New_Items = pd.DataFrame(New_Items_List, columns = col_names)

Out[44]:

New_Items.shape

Out[44]:

(6, 8)

# Store the list of new items into .csv file
if not New_Items.empty:
    path = os.path.join(os.getcwd()), "data", "mapping", "new items", str(datetime.datetime.now())
    New_Items.to_csv(path, index = False, header = True)

Data Summary

In [47]:

dataset = pd.DataFrame([New_Items.shape, Preps_Nonstd.shape, Items_nonstd.shape], columns = ['count', 'columns'],
                     index = ['New_Items', 'Preps_Nonstd', 'Items_nonstd'])

dataset

Out[47]:

<table>
<thead>
<tr>
<th></th>
<th>count</th>
<th>columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>New_Items</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Preps_Nonstd</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>Items_nonstd</td>
<td>0</td>
<td>6</td>
</tr>
</tbody>
</table>
Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang, CFFS Data Analyst

Part III: Update Information and Mapping

Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang, CFFS Data Analyst

Part III: Update Information and Mapping

Set up and Import Libraries

```python
# pip install -r requirements.txt

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import glob
import os
import cv2

from itertools import islice
from decimal import Decimal
import xml.etree.ElementTree as ET
from xml.etree.ElementTree import parse
import openpyxl
import pickle
from datetime import datetime
```

# Set the root path, change the current working directory into the project folder
path = '~/users/silvia/cffs-label'
os.chdir(path)

# Enable reading data table in the scrolling window if you prefer
pd.set_option('display.max_rows', None, 'display.max_columns', None)

Import Preprocessed Datasets

```python
Preps = pd.read_csv(os.path.join(os.getcwd(), 'data', 'cleaning', 'Preps_Unit_Cleaned.csv'))
Preps.head()

Out[5]:

<table>
<thead>
<tr>
<th>Prepd</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>StkQty</th>
<th>StkUOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BAKEDLlasagnaSpin Mushroom</td>
<td>6.550</td>
<td>Kg</td>
<td>NaN</td>
<td>5550.0</td>
<td>g</td>
</tr>
<tr>
<td>1</td>
<td>BAKEDPastaChicken Alfredo</td>
<td>6.176</td>
<td>Kg</td>
<td>NaN</td>
<td>6176.0</td>
<td>g</td>
</tr>
<tr>
<td>2</td>
<td>BAKEDPastaChorizo Penne</td>
<td>7.360</td>
<td>Kg</td>
<td>NaN</td>
<td>7360.0</td>
<td>g</td>
</tr>
<tr>
<td>3</td>
<td>BAKEDPastaShrimp Pesto</td>
<td>5.760</td>
<td>Kg</td>
<td>NaN</td>
<td>5760.0</td>
<td>g</td>
</tr>
<tr>
<td>4</td>
<td>BATCHShrimp Remoulade</td>
<td>1.000</td>
<td>Kg</td>
<td>NaN</td>
<td>1500.0</td>
<td>g</td>
</tr>
</tbody>
</table>
```

```python
gbpe_factors = pd.read_csv(os.path.join(os.getcwd(), 'data', 'external', 'gbpe_factors.csv'))
gbpe_factors.head()

Out[6]:

<table>
<thead>
<tr>
<th>Category ID</th>
<th>Food Category</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>beef &amp; buffalo meat</td>
<td>413463</td>
</tr>
<tr>
<td>1</td>
<td>lamb &amp; mutton &amp; goat meat</td>
<td>416111</td>
</tr>
<tr>
<td>2</td>
<td>pork (pig meat)</td>
<td>9.8315</td>
</tr>
<tr>
<td>3</td>
<td>poultry (chicken, turkey)</td>
<td>4.3906</td>
</tr>
</tbody>
</table>
```
### Import Update Info

#### Import list of prep that need convert uom to standard uom manually

```r
Manual_Prep <- pd.read_csv(os.path.join(os.getcwd(), "data", "cleaning", "update", "Preps_UpdateUom.csv"))
Manual_Prep
```

#### # Select the file path for new items list with category id

```r
New_Items_Added <- pd.read_csv(os.path.join(os.getcwd(), "data", "mapping", "new_items_added", "New_Items_Added_10.csv"))
New_Items_Added
```
<table>
<thead>
<tr>
<th>ItemId</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1-1392</td>
<td>COOKIE GF GINGER VEGAN</td>
<td>12</td>
<td>CT</td>
<td>1.00</td>
<td>CT</td>
<td>BAKED GOODS</td>
</tr>
<tr>
<td>7</td>
<td>1-63860</td>
<td>POPPYHUMS</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>BAKED GOODS</td>
</tr>
<tr>
<td>8</td>
<td>1-1250</td>
<td>PRETZEL PIZZA EACH</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>CT</td>
<td>BAKED GOODS</td>
</tr>
<tr>
<td>9</td>
<td>1-3771</td>
<td>YERBA MATE TEA ORG TRADITIONAL</td>
<td>6</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>BEVERAGE</td>
</tr>
<tr>
<td>10</td>
<td>1-1790</td>
<td>JUICE APPLE 100% UNSWT TETRA</td>
<td>40</td>
<td>P1N</td>
<td>1.00</td>
<td>P1N</td>
<td>BEVERAGE</td>
</tr>
<tr>
<td>11</td>
<td>1-9634</td>
<td>LOAF FRENCH BREAD</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>LOAF</td>
<td>BREAD</td>
</tr>
<tr>
<td>12</td>
<td>1-2309</td>
<td>HAM FESTIVE</td>
<td>1</td>
<td>Kg</td>
<td>1.00</td>
<td>Kg</td>
<td>DELI &amp; PREPARED MEAT</td>
</tr>
<tr>
<td>13</td>
<td>1-3125</td>
<td>CEREAL RICE KIRSPIE SQ</td>
<td>6</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>14</td>
<td>1-37470</td>
<td>CORN HOMINY</td>
<td>6</td>
<td>LG CAN</td>
<td>2835.00</td>
<td>g</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>15</td>
<td>1-19640</td>
<td>FENUGREEK LEAVES DRY</td>
<td>454</td>
<td>g</td>
<td>1.00</td>
<td>g</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>16</td>
<td>1-3397</td>
<td>OLIVE RIPE BLACK SLCD</td>
<td>6</td>
<td>LG CAN</td>
<td>2.84</td>
<td>L</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>17</td>
<td>1-3506</td>
<td>SAUCE BBQ SMOKEY TFC</td>
<td>2</td>
<td>JUG</td>
<td>3.78</td>
<td>L</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>18</td>
<td>1-3540</td>
<td>SAUCE SAMBAL DELEK</td>
<td>3</td>
<td>jar</td>
<td>3500.00</td>
<td>L</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>19</td>
<td>1-3110</td>
<td>TOFU EXTRA FIRM GREEN</td>
<td>12</td>
<td>pak</td>
<td>350.00</td>
<td>g</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>20</td>
<td>1-2409</td>
<td>TOMATO SAN MARZANO LA BREGNA</td>
<td>6</td>
<td>LG CAN</td>
<td>100.00</td>
<td>fl oz</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>21</td>
<td>1-37475</td>
<td>PASTE ACHORTE</td>
<td>110</td>
<td>g</td>
<td>1.00</td>
<td>g</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>22</td>
<td>1-36906</td>
<td>SANDWICH BACON, CHEDD &amp; TOMATO</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>23</td>
<td>1-36999</td>
<td>SANDWICH CASATA FALAFEL</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>24</td>
<td>1-36989</td>
<td>SANDWICH CHICKEN CRUNCHBERRY</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>25</td>
<td>1-36995</td>
<td>SANDWICH EGG SALAD</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>26</td>
<td>1-37000</td>
<td>SANDWICH HAM &amp; CHEESE</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>27</td>
<td>1-36997</td>
<td>SANDWICH SALMON SALAD</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>28</td>
<td>1-38987</td>
<td>WRAP BREAKFAST HAM &amp; CHEESE</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>29</td>
<td>1-38988</td>
<td>WRAP BREAKFAST MEDITERRANEAN</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>FOOD - GROCERY</td>
</tr>
<tr>
<td>30</td>
<td>1-3912</td>
<td>BURGER VEGGIE 44CT F2N</td>
<td>44</td>
<td>CT</td>
<td>1.00</td>
<td>CT</td>
<td>MEAT</td>
</tr>
<tr>
<td>31</td>
<td>1-3852</td>
<td>PORK BUTT SHLOR BNS N/MR</td>
<td>1</td>
<td>Kg</td>
<td>1.00</td>
<td>Kg</td>
<td>MEAT</td>
</tr>
<tr>
<td>32</td>
<td>1-22915</td>
<td>PORK FEET CUT</td>
<td>1</td>
<td>Kg</td>
<td>1.00</td>
<td>Kg</td>
<td>MEAT</td>
</tr>
<tr>
<td>33</td>
<td>1-64876</td>
<td>TMRIW BURGER PATTIES VEGAN</td>
<td>40</td>
<td>each</td>
<td>1.00</td>
<td>each</td>
<td>MEAT</td>
</tr>
<tr>
<td>34</td>
<td>1-64877</td>
<td>TMRIW SAUSAGE BREAKFAST PATTY</td>
<td>100</td>
<td>each</td>
<td>1.00</td>
<td>ea</td>
<td>MEAT</td>
</tr>
<tr>
<td>35</td>
<td>1-55331</td>
<td>CHICK BREAST BLISO HAL TENDOUT</td>
<td>1</td>
<td>Kg</td>
<td>1.00</td>
<td>Kg</td>
<td>POULTRY</td>
</tr>
<tr>
<td>36</td>
<td>1-3999</td>
<td>CHICK DRUMSTICK HALAL</td>
<td>1</td>
<td>Kg</td>
<td>1.00</td>
<td>Kg</td>
<td>POULTRY</td>
</tr>
<tr>
<td>37</td>
<td>1-4465</td>
<td>ASPARAGUS (LARGE) MX</td>
<td>11</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>38</td>
<td>1-22443</td>
<td>BAMBOO SHOOTS STRIP</td>
<td>6</td>
<td>LG CAN</td>
<td>2.84</td>
<td>L</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>39</td>
<td>1-10616</td>
<td>BEANS ROMANO</td>
<td>1</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>40</td>
<td>1-4582</td>
<td>CARROTS BABY BUNCHE'D BC</td>
<td>1</td>
<td>each</td>
<td>1.00</td>
<td>CT</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>41</td>
<td>1-11670</td>
<td>COLESLAW MIX CAEBEBAGE&amp;CARROT</td>
<td>5</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>42</td>
<td>1-10265</td>
<td>TOMATOES HEIRLOOM BC</td>
<td>1</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>43</td>
<td>1-44849</td>
<td>SALAD MIX ARTISAN</td>
<td>3</td>
<td>bag</td>
<td>2.00</td>
<td>lb</td>
<td>PRODUCE</td>
</tr>
<tr>
<td>44</td>
<td>1-62603</td>
<td>CKGSGGHHRMMeat Bell &amp; Patta.</td>
<td>1</td>
<td>ea</td>
<td>1.00</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
</tr>
<tr>
<td>45</td>
<td>1-19923</td>
<td>HALIBUT STEAK 40Z,OW</td>
<td>1</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>SEAFOOD</td>
</tr>
<tr>
<td>46</td>
<td>1-10512</td>
<td>SAL LOX SMK SLG OW</td>
<td>1</td>
<td>lb</td>
<td>1.00</td>
<td>lb</td>
<td>SEAFOOD</td>
</tr>
<tr>
<td>47</td>
<td>1-3094</td>
<td>PEPPERCORN BLACK WHOLE</td>
<td>3</td>
<td>Kg</td>
<td>1.00</td>
<td>Kg</td>
<td>SPICES</td>
</tr>
<tr>
<td>48</td>
<td>1-95606</td>
<td>SUMAC GROUND</td>
<td>1</td>
<td>each</td>
<td>45.00</td>
<td>g</td>
<td>SPICES</td>
</tr>
</tbody>
</table>
Update Correct Uom for Preps

In [13]:
# Update prep list with manually adjusted uom
for index, row in Manual_Preps.iterrows():
    Preps['PrepId'] = Manual_Preps.loc[index, 'PrepId']
    qty = Manual_Preps.loc[index, 'StdQty']
    uom = Manual_Preps.loc[index, 'StdUom']
    Preps.loc[Preps['PrepId'] == Preps['PrepId'], 'stdQty'] = qty
    Preps.loc[Preps['PrepId'] == Preps['PrepId'], 'stdUom'] = uom

In [14]:
Preps.drop_duplicates(subset=['PrepId'], inplace=True)

In [15]:
Preps.head()

Out[15]:
<table>
<thead>
<tr>
<th>Prepid</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>StdQty</th>
<th>StdUom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-55536 BAKED Lasagna/SO Mushroom</td>
<td>5.550</td>
<td>Kg</td>
<td>NaN</td>
<td>5550.0</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>1-58866 BAKED/Pasta/Chicken Alfredo</td>
<td>6.176</td>
<td>Kg</td>
<td>NaN</td>
<td>8176.0</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>2-54064 BAKED/Pasta/Choco Ruffle</td>
<td>7.360</td>
<td>Kg</td>
<td>NaN</td>
<td>7360.0</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>2-58502 BAKED/Pasta/Shrimp Pasta</td>
<td>5.760</td>
<td>Kg</td>
<td>NaN</td>
<td>5760.0</td>
<td>g</td>
<td></td>
</tr>
<tr>
<td>4-56433 BAC/Pasta/Shrimp Remoulade</td>
<td>1.800</td>
<td>Kg</td>
<td>NaN</td>
<td>1500.0</td>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>

In [16]:
Preps.shape

Out[16]:
(792, 7)

In [17]:
path = os.path.join(os.getcwd()), 'data', 'cleaning', 'Preps_List_Cleaned.csv')
Preps.to_csv(path, index = False, header = True)

Import List of New Items with Emission Factors Category ID Assigned

In [18]:
frames = [Items_Assigned, New_Items_Added]
Items_Assigned_Updated = pd.concat(frames).reset_index(drop=True, inplace=False).drop_duplicates() Items_Assigned_Updated.head()

Out[18]:
<table>
<thead>
<tr>
<th>ItemId</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2/kg food)</th>
<th>g N lost/kg product</th>
<th>Freshwater Withdrawals (L/JU)</th>
<th>Stress-Weighted Water Use (L/JU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-70646 1 CHUCK MAY BONELESS FZN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-10869 1 BEEF STEMITY COV FR</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-7064 1 BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-37000 1 BEEF MEATBALLS</td>
<td>4.56</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-37002 1 BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In [19]:
Items_Assigned_Updated.shape

Out[19]:
(1993, 8)

In [20]:
Items_Assigned_Updated[['CategoryID']] = Items_Assigned_Updated[['CategoryID']].apply(pd.to_numeric)

In [21]:
path = os.path.join(os.getcwd()), 'data', 'mapping', 'Items_List_Assigned.csv')
Items_Assigned_Updated.to_csv(path, index = False, header = True)
## Mapping Items to Footprint Factors

```python
In [22]:
mapping = pd.merge(mapping, item_desc, on=['Category ID', 'Food Category'], how='left',
                 left_on='CategoryID',
                 right_on='Category ID')
```

<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>57545</td>
<td>CHUCK FLAT BONELESS FZN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
</tr>
<tr>
<td>1</td>
<td>10869</td>
<td>BEEF STIRRY COV FR</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
</tr>
<tr>
<td>2</td>
<td>7064</td>
<td>BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
</tr>
<tr>
<td>3</td>
<td>37005</td>
<td>BEEF MEATBALLS</td>
<td>4.54</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
</tr>
<tr>
<td>4</td>
<td>37002</td>
<td>BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>629863</td>
<td>CH00GHMRNMeat Ball &amp; Pasta.</td>
<td>1.00</td>
<td>ea</td>
<td>1.0</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
<td>0.0000</td>
</tr>
<tr>
<td>1989</td>
<td>19923</td>
<td>HALIBUT STEAK 4OZ OW</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9798</td>
</tr>
<tr>
<td>1990</td>
<td>8106</td>
<td>SAL LOX SMK SLICE OW</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9798</td>
</tr>
<tr>
<td>1991</td>
<td>3094</td>
<td>PEPPERCORN BLACK WHOLE</td>
<td>3.00</td>
<td>Kg</td>
<td>10</td>
<td>Kg</td>
<td>SPICES</td>
<td>9.3703</td>
</tr>
<tr>
<td>1992</td>
<td>16060</td>
<td>SUMAC GROUND</td>
<td>1.00</td>
<td>ea</td>
<td>454.0</td>
<td>g</td>
<td>SPICES</td>
<td>9.3703</td>
</tr>
</tbody>
</table>

## Mapping Nitrogen Footprint Factors

```python
In [23]:
mapping = pd.merge(mapping, nitrogen_factors, on=['Category ID', 'Food Category'], how='left',
                   left_on='CategoryID',
                   right_on='Category ID')
```

<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>g N lost/kg product</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>57545</td>
<td>CHUCK FLAT BONELESS FZN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
</tr>
<tr>
<td>1</td>
<td>10869</td>
<td>BEEF STIRRY COV FR</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
</tr>
<tr>
<td>2</td>
<td>7064</td>
<td>BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
</tr>
<tr>
<td>3</td>
<td>37005</td>
<td>BEEF MEATBALLS</td>
<td>4.54</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
</tr>
<tr>
<td>4</td>
<td>37002</td>
<td>BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>g N lost/kg product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1988</td>
<td>629863</td>
<td>CH00GHMRNMeat Ball &amp; Pasta.</td>
<td>1.00</td>
<td>ea</td>
<td>1.0</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
<td>0.0000</td>
<td>0.00</td>
</tr>
<tr>
<td>1989</td>
<td>19923</td>
<td>HALIBUT STEAK 4OZ OW</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9798</td>
<td>76.30</td>
</tr>
<tr>
<td>1990</td>
<td>8106</td>
<td>SAL LOX SMK SLICE OW</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9798</td>
<td>76.30</td>
</tr>
</tbody>
</table>
### Manually Adjust Footprint Factor for Specific Items

```r
for index, row in Manual_Factor.iterrows():
    itemID = Manual_Factor.loc[index, 'ItemID']
    ghe = Manual_Factor.loc[index, 'Active Total Supply Chain Emissions (kg CO2 / kg food)']
    water = Manual_Factor.loc[index, 'Freshwater Withdrawals (L/PU)']
    ghe = itemID, 'Active Total Supply Chain Emissions (kg CO2 / kg food)' = ghe
    water = itemID, 'Freshwater Withdrawals (L/PU)' = water
    mapping.loc[mapping['ItemID'] == itemID, 'Active Total Supply Chain Emissions (kg CO2 / kg food)'] = ghe
    mapping.loc[mapping['ItemID'] == itemID, 'Freshwater Withdrawals (L/PU)'] = water
```

```r
mapping = mapping.drop_duplicates(subset=["ItemID"], inplace=True)
```

### Table: Water Footprint Factors

<table>
<thead>
<tr>
<th>ItemID</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>g N lost/kg product</th>
<th>Freshwater Withdrawals (L/PU)</th>
<th>Stress-Weighted Water Use (L/PU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10869</td>
<td>54</td>
<td>PEPPERCORN BLACK WHOLE</td>
<td>3.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>SPICES</td>
<td>9.3703</td>
<td>6.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10869</td>
<td>54</td>
<td>SUMAC GROUND</td>
<td>1.00</td>
<td>each</td>
<td>454.0</td>
<td>g</td>
<td>SPICES</td>
<td>9.3703</td>
<td>6.75</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In [24]:
# Map water footprint factors

```
mapping = pd.merge(mapping, water_factors.loc[:, ['Category ID', 'Food Category', 'Freshwater Withdrawals (L/PU)', 'Stress-Weighted Water Use (L/PU)']], left_on = 'CategoryID', right_on = 'Category ID')
```

Out [24]:

```
<table>
<thead>
<tr>
<th>ItemID</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>g N lost/kg product</th>
<th>Freshwater Withdrawals (L/PU)</th>
<th>Stress-Weighted Water Use (L/PU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>57645</td>
<td>10869</td>
<td>DUCK FLAT BONELESS CHICKEN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>57645</td>
<td>10869</td>
<td>BEEF STRIPPER COW</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>57645</td>
<td>10869</td>
<td>BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>37005</td>
<td>10869</td>
<td>BEEF MEATBALLS</td>
<td>4.54</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>37002</td>
<td>10869</td>
<td>BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>62863</td>
<td>19923</td>
<td>DUCK G&amp;G HMR Meat Ball A Pasta</td>
<td>1.00</td>
<td>ea</td>
<td>1.0</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
<td>0.0000</td>
<td>0.000</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>62863</td>
<td>19923</td>
<td>HAUBUT STEAK 36.0OZ</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9768</td>
<td>70.30</td>
<td>1580.5</td>
<td>8483.4</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>3094</td>
<td>10869</td>
<td>PEPPERCORN BLACK WHOLE</td>
<td>3.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>SPICES</td>
<td>9.3703</td>
<td>6.75</td>
<td>24.9</td>
<td>220.3</td>
</tr>
<tr>
<td>10600</td>
<td>10869</td>
<td>SUMAC GROUND</td>
<td>1.00</td>
<td>each</td>
<td>454.0</td>
<td>g</td>
<td>SPICES</td>
<td>9.3703</td>
<td>6.75</td>
<td>24.9</td>
<td>220.3</td>
</tr>
</tbody>
</table>
```

Manually Adjust Footprint Factor for Specific Items

```r
for index, row in Manual_Factor.iterrows():
    itemID = Manual_Factor.loc[index, 'ItemID']
    ...```

50
In [27]:

    mapping.shape

Out[27]:

    (1993, 12)

In [28]:

    mapping

Out[28]:

<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUOM</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>gN lost/kg product</th>
<th>Freshwater Withdrawals (L/LFU)</th>
<th>Stress-Weighted Water Use (L/LFU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1-57545</td>
<td>CHUCK FLAT BONELESS FZ FN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>1</td>
<td>1-10869</td>
<td>BEEF STIRRY COV FR</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>2</td>
<td>1-7064</td>
<td>BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>3</td>
<td>1-37005</td>
<td>BEEF MEATBALLS</td>
<td>4.54</td>
<td>Kg</td>
<td>100.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>4</td>
<td>1-37002</td>
<td>BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>100.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463</td>
<td>329.50</td>
<td>1677.2</td>
<td>61309.0</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>DRIED GARLIC</td>
<td>1.00</td>
<td>ea</td>
<td>1.0</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
<td>0.0000</td>
<td>0.000</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>6</td>
<td>1-62863</td>
<td>DRIED GARLIC</td>
<td>1.00</td>
<td>ea</td>
<td>1.0</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
<td>0.0000</td>
<td>0.000</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>7</td>
<td>1990</td>
<td>NUTS</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9706</td>
<td>70.30</td>
<td>1580.5</td>
<td>8483.4</td>
</tr>
<tr>
<td>8</td>
<td>1-19023</td>
<td>PORK</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9706</td>
<td>70.30</td>
<td>1580.5</td>
<td>8483.4</td>
</tr>
<tr>
<td>9</td>
<td>1-8105</td>
<td>SAL LDX SMK SLICED</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9706</td>
<td>70.30</td>
<td>1580.5</td>
<td>8483.4</td>
</tr>
<tr>
<td>10</td>
<td>1-3094</td>
<td>SPICE</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>SPICES</td>
<td>9.3703</td>
<td>6.75</td>
<td>24.9</td>
<td>220.3</td>
</tr>
<tr>
<td>11</td>
<td>1-16060</td>
<td>SUMAC GROUND</td>
<td>1.00</td>
<td>ea</td>
<td>454.0</td>
<td>g</td>
<td>SPICES</td>
<td>9.3703</td>
<td>6.75</td>
<td>24.9</td>
<td>220.3</td>
</tr>
</tbody>
</table>

1993 rows x 12 columns

In [29]:

    path = os.path.join(os.getcwd(), "data", "mapping", "Mapping.csv")
    mapping.to_csv(path, index = False, header = True)
Climate-Friendly Food Systems (CFFS) Labelling Project

The University of British Columbia

Created by Silvia Huang, CFFS Data Analyst

Part IV: Data Analysis

Set up and Import Libraries

```python
#pip install -- requirements.txt

import numpy as np
import pandas as pd
import pandaspipe as pdp
import matplotlib.pyplot as plt
import glob
import os
import cv2
from itertools import islice
from decimal import Decimal
import xml.etree.ElementTree as et
from xml.etree.ElementTree import parse
import aepyzt
import psutil
pd.set_option('mode.chained_assignment', None)
```

```
# Set the root path, change the the current working directory into the project folder
path = '/Users/silvia/cffs-label'
os.chdir(path)
```

```
# Enable reading data table in the scrolling window if you prefer
pd.set_option('display.max_rows', None, "display.max_columns", None)
```

Import Cleaned Datasets

```python
In [5]:
Items = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Items_list.csv"))
Item.dtypes

Out[5]:
ItemID   object
Description   object
CaseQty   float64
CaseUOM   object
PakQty   float64
PakUOM   object
InventoryGroup   object
dtype   object
```

```
In [6]:
Items.head()
```

```
0   ItemID: 4277  APPLES GRANNY SMITH  113.0  ea  1.0  CT PRODUCE
1   ItemID: 4977  ARTICHOKE 1/2 SALAD CUT TFC  6.0  LG CAN  2.5  Kg PRODUCE
2   ItemID: 2305  BACON PANSETTA  1.0  Kg  1.0  Kg MEAT
3   ItemID: 1207  BAGUETTE FRENCH  24.0  each  1.0  Ct BREAD
4   ItemID: 12703  BALSAMO GLAZE  2.0  bottle  2.0  L FOOD - GROCERY
```

```
In [7]:
Ingredients = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Ingredients_list.csv"))
Ingredients.dtypes

Out[7]:
IngredientId   object
Qty   float64
Uom   object
Conversion   float64
Recipe   object
dtype   object
```

```
In [8]:
Ingredients.head()
```

```
IngredientId  Qty  Uom  Conversion  Recipe
```

52
```python
In [9]: Preps = pd.read_csv(os.path.join(os.getcwd(), "data", "cleaning", "Preps_list_Cleaned.csv"));
Preps.dtypes

Out[9]:
PrepId          object
Description     object
PackQty        float64
PackUOM        object
InventoryGroup  object
StdQty         float64
StdUOM         object
dtype: object

In [10]: Preps.head()
Preps.shape

Out[10]:
(752, 7)

In [11]: Products = pd.read_csv(os.path.join(os.getcwd(), "data", "preprocessed", "Products_list.csv"));
Products.dtypes

Out[11]:
ProdtId         object
Description     object
SalesGroup      object
dtype: object

In [12]: Products.head()

Out[12]:
ProdtId  Description
0  R-61778  ALFFlatbread4 Cheese  OK - AL FORNO
1  R-61790  ALFFlatbreadApple & Pancetta  OK - AL FORNO
2  R-61749  ALFFlatbreadBread  OK - AL FORNO
3  R-60816  ALFFlatbreadCaprese  OK - AL FORNO
4  R-50788  ALFFlatbreadCaprese  OK - AL FORNO

In [13]: Conversions = pd.read_csv(os.path.join(os.getcwd(), "data", "cleaning", "Conversions_added.csv"));
Conversions.dtypes

Out[13]:
ConversionId    object
Multiplier      float64
ConvertQty      float64
ConvertFromUOM  object
ConvertToUOM    object
dtype: object

In [14]: Conversions

Out[14]:
ConversionId  Multiplier  ConvertQty  ConvertFromUOM  ConvertToUOM
0  NaN         1.000000   XXX         1.00           L
1  NaN         0.877193   1.14L       1.14           L
2  NaN         0.666667   1.5L        1.50           L
3  NaN         0.571429   1.75L       1.75           L
4  NaN         0.600000   2L          2.00           L
...          ...         ...         ...            ...
585  P-32864  0.016383   each        61.00          g
586  P-50707  0.005405   PTN         185.00         g
587  P-50706  0.005405   PTN         185.00         g
588  P-62983  0.005405   PTN         185.00         g
589  P-62223  0.006452   ROLL        155.00         g

590 rows x 6 columns
In [15]:
mapping = pd.read_csv(os.path.join(os.getcwd()), "data", "mapping", "Mapping.csv")
mapping.dtypes

Out[15]:
ItemID  CategoryID  Description  CaseQty  CaseUGM  PakQty  PakUGM  InventoryGroup  Active Total Supply Chain Emissions (kg CO2 / kg food)  g N lost/kg product  Freshwater Withdrawals (L/FU)  Stress-Weighted Water Use (L/FU)
object  int64  object  float64  object  float64  object  object  float64  float64  float64  float64

In [16]:
mapping

Out[16]:

<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUGM</th>
<th>PakQty</th>
<th>PakUGM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>g N lost/kg product</th>
<th>Freshwater Withdrawals (L/FU)</th>
<th>Stress-Weighted Water Use (L/FU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>57545</td>
<td>CHUCK FLAT BONELESS FZN</td>
<td>3.30</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463 329.50 1677.2 61309.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10869</td>
<td>BEEF STRIPFY COV PR</td>
<td>5.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463 329.50 1677.2 61309.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>17064</td>
<td>BEEF OUTSIDE FLAT AAA</td>
<td>1.00</td>
<td>Kg</td>
<td>1.0</td>
<td>Kg</td>
<td>MEAT</td>
<td>41.3463 329.50 1677.2 61309.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>37005</td>
<td>BEEF MEATBALLS</td>
<td>4.54</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463 329.50 1677.2 61309.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>37002</td>
<td>BEEF INSIDE ROUND SHAVED</td>
<td>9.00</td>
<td>Kg</td>
<td>1000.0</td>
<td>g</td>
<td>MEAT</td>
<td>41.3463 329.50 1677.2 61309.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1988
<table>
<thead>
<tr>
<th>Item</th>
<th>CategoryID</th>
<th>Description</th>
<th>CaseQty</th>
<th>CaseUGM</th>
<th>PakQty</th>
<th>PakUGM</th>
<th>InventoryGroup</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>g N lost/kg product</th>
<th>Freshwater Withdrawals (L/FU)</th>
<th>Stress-Weighted Water Use (L/FU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>62863</td>
<td>DKG&amp;GHMRMeat Ball &amp; Pasta</td>
<td>1.00</td>
<td>ea</td>
<td>1.0</td>
<td>ea</td>
<td>PRODUCTION FOOD</td>
<td>0.0000 0.00 0.0 0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19023</td>
<td>HAIRUT STEAK 40Z CHW</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9798 70.30 1580.5 8483.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8705</td>
<td>SALLOX SMK SLC OW</td>
<td>1.00</td>
<td>lb</td>
<td>1.0</td>
<td>lb</td>
<td>SEAFOOD</td>
<td>4.9798 70.30 1580.5 8483.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1993 rows x 12 columns

Unit Converter

In [17]:
# Import standard unit conversion information for items
Std_Unit = pd.read_csv(os.path.join(os.getcwd()), "data", "external", "standard_conversions.csv")

Out[17]:

<table>
<thead>
<tr>
<th>Multiplier</th>
<th>ConvertFromQty</th>
<th>ConvertFromUm</th>
<th>ConvertToQty</th>
<th>ConvertToUm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tsp</td>
<td>ml</td>
<td>1</td>
<td>tsp</td>
</tr>
<tr>
<td>1</td>
<td>lb</td>
<td>mi</td>
<td>1</td>
<td>lb</td>
</tr>
<tr>
<td>2</td>
<td>qt</td>
<td>mi</td>
<td>946.350000</td>
<td>qt</td>
</tr>
<tr>
<td>3</td>
<td>pt</td>
<td>mi</td>
<td>473.17625</td>
<td>pt</td>
</tr>
<tr>
<td>4</td>
<td>oz</td>
<td>g</td>
<td>28.34950</td>
<td>oz</td>
</tr>
</tbody>
</table>

In [18]:
# Import list of prep that need convert um to standard um manually
Manual_PrepU = pd.read_csv(os.path.join(os.getcwd()), "data", "cleaning", "update", "Preps_UpdateUm.csv")

Manual_PrepU.head()

Out[18]:

<table>
<thead>
<tr>
<th>PrepID</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUGM</th>
<th>InventoryGroup</th>
<th>StdQty</th>
<th>StdUm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LEMONWedge 1/8</td>
<td>1.0</td>
<td>each</td>
<td>PREP</td>
<td>84.0</td>
<td>g</td>
</tr>
<tr>
<td>1</td>
<td>MARINATEDLemon &amp; Herb Ctx</td>
<td>185.0</td>
<td>ea</td>
<td>PREP</td>
<td>24050.0</td>
<td>g</td>
</tr>
<tr>
<td>2</td>
<td>YIELDBreadSandwich 6/8</td>
<td>36.0</td>
<td>slice</td>
<td>NaN</td>
<td>1620.0</td>
<td>g</td>
</tr>
</tbody>
</table>
In [19]:
# Add unit conversion info for props into converter
Prep_cv = Manual_PrepP([PrepId, 'PakQty', 'PakUOM', 'InventoryGroup', 'StdQty', 'StdUOM'])
Prep_cv.insert(1, 'Multiplier', '')
Prep_cv.columns = Conversions.columns
Prep_cv.loc['Multiplier'] = Prep_cv['Conversions']['Multiplier']
Prep_cv.head()

Out[19]:
ConversionId  Multiplier  ConvertFromQty  ConvertFromUOM  ConvertToQty  ConvertToUom
0      P-54897      8.00      each       84.00          g
1      P-35132     185.00      ea       2400.00        g
2      P-51992      30.00    slice      1620.00        g
3      P-26234     10.00      ea       1280.00        g
4      P-26170      1.00      ea       125.00         g

In [20]:
frames = [Conversions, Prep_cv]
Conversions = pd.concat(frames).reset_index(drop=True, inplace=False).drop_duplicates()
Conversions

Out[20]:
ConversionId  Multiplier  ConvertFromQty  ConvertFromUOM  ConvertToQty  ConvertToUom
0         NaN           1           XXX            1.00          L
1       NaN           0.877193   1.00          1.14L         L
2       NaN           0.696667   1.00          1.52L         L
3       NaN           0.571429   1.00          1.75L         L
4       NaN           0.5           1.0            2L          2.00         L
...        ...          ...          ...            ...          ...
798      P-55707      1.00      PTN            185.00        g
799      P-55709      1.00      PTN            185.00        g
800      P-62929      1.00      PTN            185.00        g
801      P-62082      1.00      ROL           155.00        g
802      NaN           NaN         NaN            NaN          NaN
803 rows x 6 columns

In [21]:
# Separate units that converted to 'ml' or 'g'
lst_unit = Std_Unit.loc[Std_Unit['ConvertToUom'], 'ConvertFromUom'].tolist()
solid_unit = Std_Unit.loc[Std_Unit['ConvertToUom'] == 'g', 'ConvertFromUom'].tolist()

In [22]:
# Construct a standard unit converter
std_converter(qty, uom):
    if uom in Std_Unit['ConvertFromUom'].tolist():
        multiplier = Std_Unit.loc[Std_Unit['ConvertToUom'] == 'uom', 'Multiplier']
        Qty = float(qty)*float(multiplier)
        Uom = Std_Unit.loc[Std_Unit['ConvertToUom'] == 'uom', 'ConvertFromUom'].value[0]
    else:
        Qty = qty
        Uom = uom
    return (Qty, Uom)

In [23]:
# Test the std converter
std_converter(6.25, 'lb')

Out[23]:
(113.398, 'g')

In [24]:
# Construct a unit converter for specific items
spc_converter = list(filter(lambda x: x in Conversions['ConversionId'].tolist(),))

def spc_converter(qty, uom, unit):
    if uom in liquid_unit + solid_unit: # Convert to std uom for ingredients has no specific conversion instruction
        return std_converter(qty, uom)
    else:
        Ingr in spc_conv # Convert to std uom for ingredients has specific conversion instruction
        conversion = Conversions[(ConversionId == Ingr) & (Conversions['ConvertToUom'] == uom) & (Conversions['ConvertFromUom'] == uom) & (Conversions['Multiplier'] == multiplier)]
        multiplier = conversion['Multiplier']
        if multiplier.empty:
library(ggplot2)

return std_converter(qty, uom)
else:
    #print(conversion)
    Qty = float(qty)/float(multiplier)
    Uom = conversion['ConvertToUom'].values[0]
    return Qty, Uom

else:
    return std_converter(qty, uom)

# Text the spc_converter
#spc_converter('T-112', 1, 'CT')

spc_converter('P-35132', 1, 'ma')

(129.9999496000002, 'g')

GHG Factors Calculation for Preps

Preps['GHG Emission (g)'] = 0
Preps['GHG Emission(g)/StdDm'] = 0
Preps['N lost (g)'] = 0
Preps['N lost (g)/StdDm'] = 0
Preps['Freshwater Withdrawals (ml)'] = 0
Preps['Freshwater Withdrawals (ml)/StdDm'] = 0
Preps['Stress-Weighted Water Use (ml)'] = 0
Preps['Stress-Weighted Water Use (ml)/StdDm'] = 0

# Calculate GHG, nitro, water footprints per gram/ml of each prep for items as ingredients only
def get_items_gbps_prep(index, row):
    ingres = Ingredients.loc[Ingredients['Recipe'] == Preps.loc[Preps['PrepId']]]
    ghg = Preps.loc[Preps['index'], 'GHG Emission (g)']
    nitro = Preps.loc[Preps['index'], 'N lost (g)']
    water = Preps.loc[Preps['index'], 'Freshwater Withdrawals (ml)']
    str_water = Preps.loc[Preps['index'], 'Stress-Weighted Water Use (ml)']
    weight = Preps.loc[Preps['index'], 'StdDm']
    #print('index', 'index', 'ingredients', 'ingres')
    for idx, row in ingres.iterrows():
        ingres = ingres.loc[ingres['index'], 'Quantity']
        if ingres.iloc[0]['index']:
            ghg = mapping.loc[mapping['index'] == ingres, 'Active Total Supply Chain Emissions (kg CO2 / kg product)']
            nitro = mapping.loc[mapping['index'] == ingres, 'N lost/kg product']
            water = mapping.loc[mapping['index'] == ingres, 'Freshwater Withdrawals (L/FU)']
            str_water = mapping.loc[mapping['index'] == ingres, 'Stress-Weighted Water Use (L/FU)']
            #print(index)
            Qty = float(ingres.loc[idx, 'Quantity'])
            Uom = ingres.loc[ingres['index'], 'Uom']
            if ingres in spc_gov:
                qty = spc_converter(qty, Qty, Uom)
                ghg = qty/float(ghg)
                nitro = qty/float(nitro)
                water = qty/float(water)
                str_water = qty/float(str_water)
            else:
                qty = std_converter(qty, Qty, Uom)
                ghg = qty/float(ghg)
                nitro = qty/float(nitro)
                water = qty/float(water)
                str_water = qty/float(str_water)
                #print(index, Qty, Uom, qty, float(ghg), float(nitro), float(water), float(str_water))
                #print(ghg, nitro, water, str_water)
                Preps.loc[Preps['index'], 'GHG Emission (g)'] = float(ghg)
                Preps.loc[Preps['index'], 'N lost (g)'] = float(nitro)
                Preps.loc[Preps['index'], 'Freshwater Withdrawals (ml)'] = float(water)
                Preps.loc[Preps['index'], 'Stress-Weighted Water Use (ml)'] = float(str_water)

# Calculate GHG, nitro, water footprints per gram/ml of each prep for other preps as ingredients

def get_preps_gbps_prep(index, row):
    ingres = Ingredients.loc[Ingredients['Recipe'] == Preps.loc[Preps['PrepId']]]
    ghg = Preps.loc[Preps['index'], 'GHG Emission(g)']
    water = Preps.loc[Preps['index'], 'Freshwater Withdrawals (ml)']
    str_water = Preps.loc[Preps['index'], 'Stress-Weighted Water Use (ml)']
    weight = Preps.loc[Preps['index'], 'StdDm']
    #print('index', 'ingredients', 'ingres')
    for idx, row in ingres.iterrows():
        ingres = ingres.loc[ingres['index'], 'ingredients']
        if ingres.iloc[0]['index'] and len(ingres) > 1:
            ghg = Preps.loc[Prep[Prep['PrepId']]] == ingres, 'GHG Emission(g)/StdDm'
            nitro = Preps.loc[Prep[Prep['PrepId']]] == ingres, 'N lost (g)/StdDm'
```python
water_fac = Preps.loc[Preps['Prepid'] == ingre, 'Freshwater Withdrawals (ml)/StdHm']
str_water_fac = Preps.loc[Preps['Prepid'] == ingre, 'Stress-Weighted Water Use (ml)/StdHm']

print(ghg)
Qty = float(ingres.loc[idx, 'Qty'])

Uom = ingres.loc[idx, 'Uom']
if Uom in snc_cols:
    qty = snc_converter(ingres, Qty, Uom)[0]
ghg = qty/float(ghg)

str_water = qty*float(str_water_fac)

else:
    qty = std_converter(qty, Uom)[0]
ghg = qty*float(ghg)

str_water = qty*float(str_water_fac)

Preps.loc[index, ' GHG Emission (g) ' ] = ghg
Preps.loc[index, ' GHG Emission (g)/StdHm ' ] = ghg/float(weight)
Preps.loc[index, ' N lost (g) ' ] = float(nitro)
Preps.loc[index, ' N lost (g)/StdHm ' ] = float(nitro)/float(weight)
Preps.loc[index, ' Freshwater Withdrawals (ml) ' ] = float(water)
Preps.loc[index, ' Freshwater Withdrawals (ml)/StdHm ' ] = float(water)/float(weight)
Preps.loc[index, ' Stress-Weighted Water Use (ml) ' ] = float(str_water)
Preps.loc[index, ' Stress-Weighted Water Use (ml)/StdHm ' ] = float(str_water)/float(weight)

In [30]:
def link_preps(index, row):
   
    ingres = ingrediemn.loc[ingrediemn['Recipe'] == Preps.loc[index, 'Prepid']].iloc[0]

    ghg = Preps.loc[index, 'GHG Emission (g)']
    nitro = Preps.loc[index, 'N lost (g)']
    water = Preps.loc[index, 'Freshwater Withdrawals (ml)']
    str_water = Preps.loc[index, 'Stress-Weighted Water Use (ml)']
   
    weight = Preps.loc[index, 'stOqm']
    if len(ingres) == 1:
        ingres = ingres.iloc[0]['Ingrediemn']
    if ingres.anazawit["B"]:
        #print(ingres)

    ghg = Preps.loc[Preps['Prepid'] == ingres, 'GHG Emission (g)']/stdHm
    nitro = Preps.loc[Preps['Prepid'] == ingres, 'N lost (g)']/stdHm
    water = Preps.loc[Preps['Prepid'] == ingres, 'Freshwater Withdrawals (ml)/stdHm']
    str_water = Preps.loc[Preps['Prepid'] == ingres, 'Stress-Weighted Water Use (ml)/stdHm']

    Qty = float(ingres.loc[0]['Qty'])
    Uom = ingres.loc[0]['Uom']

    if Uom in snc_cols:
        qty = snc_converter(ingres, Qty, Uom)[0]
        ghg = qty/float(ghg)
        nitro = qty/float(nitro)
        water = qty/float(water)
        str_water = qty/float(str_water)

        else:
            qty = std_converter(qty, Uom)[0]
            ghg = qty/float(ghg)
            nitro = float(nitro)
            water = float(water)
            str_water = float(str_water)

        #print(ingres, ghg, qty, Uom, qty, weight)
        #print(ghg, nitro, water, str_water)

        Preps.loc[index, 'GHG Emission (g)'] = float(ghg)
        Preps.loc[index, 'GHG Emission (g)/stdHm'] = ghg/float(weight)
        Preps.loc[index, 'N lost (g)'] = float(nitro)
        Preps.loc[index, 'N lost (g)/stdHm'] = nitro/float(weight)
        Preps.loc[index, 'Freshwater Withdrawals (ml)'] = float(water)
        Preps.loc[index, 'Freshwater Withdrawals (ml)/stdHm'] = water/float(weight)
        Preps.loc[index, 'Stress-Weighted Water Use (ml)'] = float(str_water)
        Preps.loc[index, 'Stress-Weighted Water Use (ml)/stdHm'] = float(str_water)/float(weight)
```

In [31]:
for index, row in Preps.iterrows():
    get_itema_gqpe_preps(index, row)

In [32]:
for index, row in Preps.iterrows():
    link_preps(index, row)

In [33]:
for index, row in Preps.iterrows():
    get_preps_gqpe_preps(index, row)

In [34]:
Preps
```
<table>
<thead>
<tr>
<th>Prepid</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>StdQty</th>
<th>StdUom</th>
<th>GHG Emission (g)</th>
<th>GHG Emission(g)/stdHm</th>
<th>N lost (g)</th>
<th>N lost (g)/stdHm</th>
</tr>
</thead>
</table>
```
<table>
<thead>
<tr>
<th>Prepid</th>
<th>Description</th>
<th>PakQty</th>
<th>PakUOM</th>
<th>InventoryGroup</th>
<th>StdQty</th>
<th>StdUom</th>
<th>GHG Emission (g)</th>
<th>Emission(g)/StdUom</th>
<th>GHG</th>
<th>N lost (g)</th>
<th>N Lc (g)/StdUom</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BAKEDLasagnaSpin Mushroom</td>
<td>5.550</td>
<td>Kg</td>
<td>NaN</td>
<td>5550.0</td>
<td>g</td>
<td>19928.597446</td>
<td>3.590738</td>
<td>g</td>
<td>203.304444</td>
<td>0.9360</td>
</tr>
<tr>
<td>1</td>
<td>BAKEDPastryChickn Alfredo</td>
<td>0.178</td>
<td>Kg</td>
<td>NaN</td>
<td>9176.0</td>
<td>g</td>
<td>17657.71270</td>
<td>2.859092</td>
<td>g</td>
<td>220.725071</td>
<td>0.0357</td>
</tr>
<tr>
<td>2</td>
<td>BAKEDPastryCholte Pepper</td>
<td>7.360</td>
<td>Kg</td>
<td>NaN</td>
<td>7360.0</td>
<td>g</td>
<td>22170.04985</td>
<td>3.013186</td>
<td>g</td>
<td>263.815524</td>
<td>0.0258</td>
</tr>
<tr>
<td>3</td>
<td>BAKEDPastryShrimp Pesto</td>
<td>5.760</td>
<td>Kg</td>
<td>NaN</td>
<td>5760.0</td>
<td>g</td>
<td>29040.084386</td>
<td>5.041681</td>
<td>g</td>
<td>178.743124</td>
<td>0.0310</td>
</tr>
<tr>
<td>4</td>
<td>BACHEShrimp Remoulade</td>
<td>1.000</td>
<td>Kg</td>
<td>NaN</td>
<td>1600.0</td>
<td>g</td>
<td>12750.771772</td>
<td>7.096232</td>
<td>g</td>
<td>45.416647</td>
<td>0.0283</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>747</td>
<td>MIXChoose</td>
<td>2.000</td>
<td>Kg</td>
<td>PreP</td>
<td>2000.0</td>
<td>g</td>
<td>17920.800000</td>
<td>8.910400</td>
<td>g</td>
<td>185.020000</td>
<td>0.0933</td>
</tr>
<tr>
<td>748</td>
<td>ROASTEDS(Baghetti Squash)</td>
<td>1.400</td>
<td>Kg</td>
<td>NaN</td>
<td>1400.0</td>
<td>g</td>
<td>1411.767296</td>
<td>1.038465</td>
<td>g</td>
<td>18.878999</td>
<td>0.0135</td>
</tr>
<tr>
<td>749</td>
<td>SAUTE(California Rice)</td>
<td>1.000</td>
<td>Kg</td>
<td>NaN</td>
<td>1000.0</td>
<td>g</td>
<td>809.481296</td>
<td>0.809481</td>
<td>g</td>
<td>8.311999</td>
<td>0.0821</td>
</tr>
<tr>
<td>760</td>
<td>YEILD(Context Peer)</td>
<td>800.000</td>
<td>g</td>
<td>NaN</td>
<td>800.000</td>
<td>g</td>
<td>430.600000</td>
<td>0.538250</td>
<td>g</td>
<td>2.700000</td>
<td>0.0033</td>
</tr>
<tr>
<td>751</td>
<td>YEILDLetuce Kun</td>
<td>3.000</td>
<td>PTN</td>
<td>NaN</td>
<td>450.000</td>
<td>g</td>
<td>469.611174</td>
<td>1.043580</td>
<td>g</td>
<td>5.946516</td>
<td>0.0132</td>
</tr>
</tbody>
</table>

752 rows x 15 columns

In [35]:
path = os.path.join(os.getcwd(), 'data', 'final', 'PrepFootprints.csv')
Prep.to_csv(path, index = False, header = True)

GHGe Calculation for Products

In [36]:
Products['Weight (g)'] = 0
Products['GHG Emission (g)'] = 0
Products['N lost (g)'] = 0
Products['Freshwater Withdrawals (mL)'] = 0
Products['Stress-Weighted Water Use (mL)'] = 0

In [37]:
# Calculate GHG, nitro, water footprints per gram/ml of each product for items ingredients only
def gas_items_gphe(index, row):
    lngres = Ingredients.loc[Ingredients['Recipe'] == Products.loc[index, 'Product']]
    ghp = Products.loc[index, 'GHG Emission (g)']
    nitro = Products.loc[index, 'N lost (g)']
    water = Products.loc[index, 'Freshwater Withdrawals (mL)']
    wtrater = Products.loc[index, 'Stress-Weighted Water Use (mL)']
    weight = Products.loc[index, 'Weight (g)']
    #print index, lngres, lngres['lngres']
    for idx, row in lngres.iterrows():
        lngres = lngres.loc[idx, ['IngredientName']]
        if lngres.size > 0:
            ghp = mapping.loc[mapping['ItemID'] == lngres, 'Active Total Supply Chain Emissions (kg CO2 / kg food)']
            nitro = mapping.loc[mapping['ItemID'] == lngres, 'N lost/kg product']
            water = mapping.loc[mapping['ItemID'] == lngres, 'Freshwater Withdrawals (L/FU)']
            wtrater = mapping.loc[mapping['ItemID'] == lngres, 'Stress-Weighted Water Use (L/FU)']
            qty = float(lngres.loc[idx, 'Qty'])
            weight = float(lngres.loc[idx, 'Weight'])
            ghp = qty*float(ghp)
            nitro = qty*float(nitro)/1000
            water = qty*float(water)/1000
            wtrater = qty*float(wtrater)/1000
            #print idx, qty, ghp, qty*float(ghp), qty*float(nitro), qty*float(water), qty*float(wtrater)
        else:
In [38]:
# Calculate GHG, nitro, water footprints per gram/ml of each product for prep ingredients only

def get_prep_ggw(index, row):
    ingre = IngredLoc.loc[IngredLoc['Recipe'] == ProductsLoc['Product ID']]
    ggw = ProductsLoc.loc[ProductsLoc['GHG Emission (g)']]
    nitro = ProductsLoc.loc[ProductsLoc['Weight (g)']]
    water = ProductsLoc.loc[ProductsLoc['Stress-Weighted Water Use (ml)']]
    str_water = ProductsLoc.loc[ProductsLoc['Stress-Weighted Water Use (ml)']]
    weight = ProductsLoc.loc[ProductsLoc['Weight (g)']]
    
    for idx, row in ingres.iterrows():
        ggw = ggw.loc[ggw['Product ID'] == ingre, 'GHG Emission(g)']/stdUom
        nitro = nitro.loc[nitro['Product ID'] == ingre, 'Weight (g)']/stdUom
        water = water.loc[water['Product ID'] == ingre, 'Freshwater Withdrawals (ml)/stdUom
        str_water = str_water.loc[str_water['Product ID'] == ingre, 'Stress-Weighted Water Use (ml)/stdUom
        
    
In [39]:
# Calculate GHG, nitro, water footprints per gram/ml of each product for other products ingredients

def get_ggw_ggw(index, row):
    ingre = IngredLoc.loc[IngredLoc['Recipe'] == ProductsLoc['Product ID']]
    ggw = ProductsLoc.loc[ProductsLoc['GHG Emission (g)']]
    nitro = ProductsLoc.loc[ProductsLoc['Weight (g)']]
    water = ProductsLoc.loc[ProductsLoc['Freshwater Withdrawals (ml)']]
    str_water = ProductsLoc.loc[ProductsLoc['Stress-Weighted Water Use (ml)']]
    
    for idx, row in ingres.iterrows():
        ggw = ggw.loc[ggw['Product ID'] == ingre, 'GHG Emission (g)']
        nitro = nitro.loc[nitro['Product ID'] == ingre, 'Weight (g)']
        water = water.loc[water['Product ID'] == ingre, 'Freshwater Withdrawals (ml)']
        str_water = str_water.loc[str_water['Product ID'] == ingre, 'Stress-Weighted Water Use (ml)']

In [40]:
for index, row in Products.iterrows():
    get_ggw_ggw(index, row)

In [41]:
for index, row in Products.iterrows():
    get_prep_ggw(index, row)

In [42]:
for index, row in Products.iterrows():
    get_products_ggw(index, row)

In [43]:
# Filter our products using preps with unknown units
Preps_Unknown = pd.read_csv('prep_20180802.csv', 'data', 'cleaning', 'Prep_Unknown.csv')
Preps_Unknown
```python
In [44]:
def filter_products(index, row):
    ingreas = Ingredients.loc[Ingredients['Recipe'] == Products.loc[index, 'ProdId']]  
    print(ingreas)
    for idx, row in ingreas.iterrows():
        ingr = ingr.loc[idx, 'IngredientId']
        if ingr in Products['Product'].tolist():
            print(ingr, index, Products.loc[index, 'ProdId'])  
            Products.drop(index, inplace=True)
    break

In [45]:
for index, row in Products.iterrows():
    filter_products(index, row)

In [46]:
Products['Freshwater Withdrawals (L)'] = round(Products['Freshwater Withdrawals (ml)'] / 1000, 2)
Products['Stress-Weighted Water Use (L)'] = round(Products['Stress-Weighted Water Use (ml)'] / 1000, 2)
Products = Products.drop(columns=['Freshwater Withdrawals (ml)', 'Stress-Weighted Water Use (ml)'])

In [47]:
Products['GHG Emission (g) / 100g'] = round(100*Products['GHG Emission (g)'] / Products['Weight (g)'], 2)
Products['N lost (g) / 100g'] = round(100*Products['N lost (g)'] / Products['Weight (g)'], 2)
Products['Freshwater Withdrawals (L) / 100g'] = round(100*Products['Freshwater Withdrawals (L)'] / Products['Weight (g)'], 2)
Products['Stress-Weighted Water Use (L) / 100g'] = round(100*Products['Stress-Weighted Water Use (L)'] / Products['Weight (g)'])

In [48]:
Products

Out[48]:
<table>
<thead>
<tr>
<th>ProdId</th>
<th>Description</th>
<th>SalesGroup</th>
<th>Weight (g)</th>
<th>GHG Emission (g)</th>
<th>N lost (g)</th>
<th>Freshwater Withdrawals (L)</th>
<th>Stress-Weighted Water Use (L)</th>
<th>GHG Emission (g) / 100g</th>
<th>N lost (g) / 100g</th>
<th>Freshwater Withdrawals (L) / 100g</th>
<th>Stress-Weighted Water Use (L) / 100g</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>R-61778 ALFFlatbread4 Cheese</td>
<td>OK - AL FORNO</td>
<td>165.000000</td>
<td>1229.947650</td>
<td>119.030000</td>
<td>212.84</td>
<td>10662.59</td>
<td>664.84</td>
<td>6.45</td>
<td>115.05</td>
<td>91.06</td>
</tr>
<tr>
<td>1</td>
<td>R-61780 ALFFlatbread.Apple &amp; Pork</td>
<td>OK - AL FORNO</td>
<td>140.000000</td>
<td>756.345760</td>
<td>9.031250</td>
<td>144.99</td>
<td>5287.95</td>
<td>540.25</td>
<td>6.45</td>
<td>103.56</td>
<td>95.67</td>
</tr>
<tr>
<td>2</td>
<td>R-61769 ALFFlatbread.BBQ Chicken</td>
<td>OK - AL FORNO</td>
<td>245.000000</td>
<td>1011.766023</td>
<td>18.958191</td>
<td>118.17</td>
<td>3196.50</td>
<td>412.96</td>
<td>7.74</td>
<td>48.23</td>
<td>40.34</td>
</tr>
<tr>
<td>3</td>
<td>R-50659 ALFFlatbread.Braschetta</td>
<td>OK - AL FORNO</td>
<td>215.000000</td>
<td>454.128829</td>
<td>4.589045</td>
<td>75.98</td>
<td>3520.38</td>
<td>213.22</td>
<td>2.13</td>
<td>35.34</td>
<td>27.34</td>
</tr>
<tr>
<td>4</td>
<td>R-50788 ALFFlatbread.Capprese</td>
<td>OK - AL FORNO</td>
<td>233.000000</td>
<td>1010.276452</td>
<td>10.809234</td>
<td>160.95</td>
<td>7071.61</td>
<td>433.60</td>
<td>4.64</td>
<td>64.79</td>
<td>54.19</td>
</tr>
<tr>
<td>453</td>
<td>R-57915 SQRItofu Sohrito Quessed</td>
<td>OK - SQUARE MEAL</td>
<td>600.000002</td>
<td>1542.748135</td>
<td>11.964991</td>
<td>171.88</td>
<td>8616.10</td>
<td>237.35</td>
<td>1.84</td>
<td>26.44</td>
<td>21.26</td>
</tr>
<tr>
<td>454</td>
<td>R-61679 SQRItofu Sohrito Guasalo</td>
<td>OK - SQUARE MEAL</td>
<td>1065.000002</td>
<td>2275.218575</td>
<td>16.560511</td>
<td>214.09</td>
<td>11264.83</td>
<td>213.64</td>
<td>1.55</td>
<td>26.19</td>
<td>20.95</td>
</tr>
<tr>
<td>455</td>
<td>R-56902 SQRWegan Lettuce Wrap</td>
<td>OK - SQUARE MEAL</td>
<td>398.999993</td>
<td>640.285633</td>
<td>3.967896</td>
<td>81.04</td>
<td>4652.22</td>
<td>169.07</td>
<td>1.00</td>
<td>20.26</td>
<td>17.13</td>
</tr>
<tr>
<td>456</td>
<td>R-57810 SQRWegan Lettuce Wrap</td>
<td>OK - SQUARE MEAL</td>
<td>544.999993</td>
<td>806.712203</td>
<td>4.920096</td>
<td>157.61</td>
<td>5364.54</td>
<td>148.02</td>
<td>0.93</td>
<td>28.92</td>
<td>25.02</td>
</tr>
<tr>
<td>457</td>
<td>R-57811 SQRWegan Lettuce Wrap</td>
<td>OK - SQUARE MEAL</td>
<td>688.909993</td>
<td>973.128774</td>
<td>5.852376</td>
<td>234.18</td>
<td>6026.86</td>
<td>141.03</td>
<td>0.85</td>
<td>33.94</td>
<td>29.04</td>
</tr>
</tbody>
</table>

458 rows x 12 columns

In [49]:
Products.shape

Out[49]:
(458, 12)

In [50]:
path = os.path.join(os.getcwd(), "data", "final", "Recipex Footprints.csv")
Products.to_csv(path, index = False, header = True)

Data Visualization

In [51]:
path = os.path.join(os.getcwd(), "reports", "figures")

In [52]:
Products.boxplot(column=['GHG Emission (g)'], return_type='axes')
```
In [53]: Products.boxplot(column=['N lost (g)'], return_type='axes')

Out[53]:<AxesSubplot:>

In [54]: Products.boxplot(column=['Freshwater Withdrawals (L)'], return_type='axes')

Out[54]:<AxesSubplot:>

In [55]: Products.boxplot(column=['Stress-Weighted Water Use (L)'], return_type='axes')

Out[55]:<AxesSubplot:>

In [56]: Products.hist(column='GHS Emission (g)'; bins=40, alpha=0.7
plot.axvline(Products['GHS Emission (g)'].mean(), color='r', linestyle='dashed', linewidth=2, label='mean')
plot.axvline(Products['GHS Emission (g)'].median(), color='k', linewidth=1, label='median')
plot.ylabel('GHS Emission (g) / Dish')
plot.ylim Palest 'GHSd_dish.png')
plot.savefig(path + 'GHSd_dish.png')
```python
import matplotlib.pyplot as plt
import pandas as pd

# Read data from a CSV file
data = pd.read_csv('example_data.csv')

# Plot GHG Emission distribution
plt.hist(data['GHG Emission (g) / 100g'], bins=40, alpha=0.7)
plt.axvline(data['GHG Emission (g) / 100g'].mean(), color='r', linestyle='dashed', linewidth=2, label='Mean')
plt.xlabel('GHG Emission (g) / 100g')
plt.ylabel('Frequency')
plt.legend()
plt.savefig('GHG_100g.png')

# Plot N lost distribution
plt.hist(data['N lost (g)'], bins=40, alpha=0.7)
plt.axvline(data['N lost (g)'].mean(), color='r', linestyle='dashed', linewidth=2, label='Mean')
plt.xlabel('N lost (g) / Dish')
plt.ylabel('Frequency')
plt.legend()
plt.savefig('N_lost.png')
```

This code reads data from a CSV file, plots the distribution of GHG Emission and N lost, and saves the plots to files named `GHG_100g.png` and `N_lost.png`. The plots show the frequency of GHG Emission and N lost along with the mean value.
In [53]:
```
Products.hist(column=['Stress-Weighted Water Use (L) / 100g'], bins=40, alpha = 0.7)
plt.axvline(Products['Stress-Weighted Water Use (L) / 100g'].mean(), color='r', linestyle='dashed', linewidth=2, label = Products['Stress-Weighted Water Use (L) / 100g'].median(), color='k', linewidth=1, label = 'median')
plt.ylabel('Stress-Weighted Water Use (L) / 100g')
plt.xlim(left=0)
plt.savefig(path + 'Stress_water_100g.png')
```
<table>
<thead>
<tr>
<th>Category ID</th>
<th>Food Category</th>
<th>Active Total Supply Chain Emissions (kg CO2 / kg food)</th>
<th>Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beef &amp; buffalo meat</td>
<td>41.3463</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>2</td>
<td>lamb/mutton &amp; goat meat</td>
<td>41.6211</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>3</td>
<td>pork (pig meat)</td>
<td>9.8315</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>4</td>
<td>poultry (chicken, turkey)</td>
<td>4.3996</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>5</td>
<td>butter</td>
<td>11.4316</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>6</td>
<td>cheese</td>
<td>8.9104</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>7</td>
<td>ice cream</td>
<td>4.0163</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>8</td>
<td>cream</td>
<td>6.9824</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>9</td>
<td>milk (cow’s milk)</td>
<td>2.2325</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>10</td>
<td>yogurt</td>
<td>2.9782</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>11</td>
<td>eggs</td>
<td>3.6615</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>12</td>
<td>fish (finfish)</td>
<td>4.9798</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>13</td>
<td>crustaceans (shrimp/prawns)</td>
<td>21.1274</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>14</td>
<td>mollusks</td>
<td>2.4351</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>15</td>
<td>animal fats</td>
<td>6.9693</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>16</td>
<td>other legumes</td>
<td>1.6042</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>17</td>
<td>beans and pulses (dried)</td>
<td>1.6776</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>18</td>
<td>peas</td>
<td>0.6995</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>19</td>
<td>peanuts/groundnuts</td>
<td>1.692</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>20</td>
<td>soybeans/tofu</td>
<td>1.7542</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>21</td>
<td>other grains/cereals</td>
<td>1.4785</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>22</td>
<td>corn (maize)</td>
<td>0.9734</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>23</td>
<td>oats (oatmeal)</td>
<td>2.3017</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>24</td>
<td>wheat/rye (bread, pasta, baked goods)</td>
<td>1.5225</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>25</td>
<td>rice</td>
<td>2.5345</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>26</td>
<td>tree nuts and seeds</td>
<td>4.2854</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>27</td>
<td>almond milk</td>
<td>0.7021</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>28</td>
<td>oat milk</td>
<td>0.9943</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>29</td>
<td>rice milk</td>
<td>0.6972</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>30</td>
<td>soy milk</td>
<td>0.489</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>31</td>
<td>other fruits</td>
<td>0.4306</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>32</td>
<td>apples</td>
<td>0.3581</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td></td>
<td>Food Item</td>
<td>Emissions</td>
<td>Source</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>33</td>
<td>bananas</td>
<td>0.7115</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>34</td>
<td>berries</td>
<td>1.6547</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>35</td>
<td>citrus fruit</td>
<td>0.3942</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>36</td>
<td>cabbages and other brassicas (broccoli)</td>
<td>0.622</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>37</td>
<td>tomatoes</td>
<td>0.6932</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>38</td>
<td>root vegetables</td>
<td>0.3062</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>39</td>
<td>onions and leeks</td>
<td>0.3015</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>40</td>
<td>other vegetables</td>
<td>0.5029</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>41</td>
<td>potatoes</td>
<td>0.397</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>42</td>
<td>cassava and other roots</td>
<td>0.397</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>43</td>
<td>sugars and sweeteners</td>
<td>1.6414</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>44</td>
<td>other vegetable oils</td>
<td>3.1509</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>45</td>
<td>soybeans (oil)</td>
<td>3.0336</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>46</td>
<td>palm (oil)</td>
<td>4.2483</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>47</td>
<td>sunflower (oil)</td>
<td>3.0231</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>48</td>
<td>rapeseed/canola (oil)</td>
<td>3.2401</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>49</td>
<td>olives (oil)</td>
<td>5.6383</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>50</td>
<td>barley (beer)</td>
<td>0.9535</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>51</td>
<td>wine grapes (wine)</td>
<td>1.3776</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>52</td>
<td>cocoa</td>
<td>10.456</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>53</td>
<td>coffee</td>
<td>16.6995</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>54</td>
<td>stimulants &amp; spices misc.</td>
<td>9.3703</td>
<td>Cool Food Calculator</td>
</tr>
<tr>
<td>55</td>
<td>water &amp; beverages</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>56</td>
<td>salt</td>
<td>0.44</td>
<td>The Big Climate Database</td>
</tr>
<tr>
<td>57</td>
<td>vinegar</td>
<td>1.93</td>
<td>The Big Climate Database</td>
</tr>
<tr>
<td>58</td>
<td>sauces &amp; paste</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>59</td>
<td>manually adjusted</td>
<td>0</td>
<td>Estimated Individually</td>
</tr>
<tr>
<td>60</td>
<td>human labor</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>61</td>
<td>kitchen supplies</td>
<td>0</td>
<td>By Assumption</td>
</tr>
</tbody>
</table>
# Appendix C [Nitrogen Footprint Factors List]

<table>
<thead>
<tr>
<th>Category ID</th>
<th>Food Category</th>
<th>g N lost/kg product</th>
<th>Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beef &amp; buffalo meat</td>
<td>329.5</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>2</td>
<td>lamb/mutton &amp; goat meat</td>
<td>231.15</td>
<td>Average of beef (1) and pork (3)</td>
</tr>
<tr>
<td>3</td>
<td>pork (pig meat)</td>
<td>132.8</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>4</td>
<td>poultry (chicken, turkey)</td>
<td>116.8</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>5</td>
<td>butter</td>
<td>100.35</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>6</td>
<td>cheese</td>
<td>93.3</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>7</td>
<td>ice cream</td>
<td>16.2</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>8</td>
<td>cream</td>
<td>28.08</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>9</td>
<td>milk (cow's milk)</td>
<td>19.6</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>10</td>
<td>yogurt</td>
<td>26.07</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>11</td>
<td>eggs</td>
<td>61.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>12</td>
<td>fish (finfish)</td>
<td>70.3</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>13</td>
<td>crustaceans (shrimp/prawns)</td>
<td>70.3</td>
<td>Same as fish (12)</td>
</tr>
<tr>
<td>14</td>
<td>mollusks</td>
<td>70.3</td>
<td>Same as fish (12)</td>
</tr>
<tr>
<td>15</td>
<td>animal fats</td>
<td>0.2</td>
<td>Same as oil (44)</td>
</tr>
<tr>
<td>16</td>
<td>other legumes</td>
<td>5.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>17</td>
<td>beans and pulses (dried)</td>
<td>5.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>18</td>
<td>peas</td>
<td>5.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>19</td>
<td>peanuts/groundnuts</td>
<td>12.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>20</td>
<td>soybeans/tofu</td>
<td>5.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>21</td>
<td>other grains/cereals</td>
<td>5.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>22</td>
<td>corn (maize)</td>
<td>6.75</td>
<td>Average of all veg products</td>
</tr>
<tr>
<td>23</td>
<td>oats (oatmeal)</td>
<td>6.75</td>
<td>Average of all veg products</td>
</tr>
<tr>
<td>24</td>
<td>wheat/rye (bread, pasta, baked goods)</td>
<td>14.8</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>25</td>
<td>rice</td>
<td>5.3</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>26</td>
<td>tree nuts and seeds</td>
<td>12.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>27</td>
<td>almond milk</td>
<td>3.05</td>
<td>1/4 of nuts (19)</td>
</tr>
<tr>
<td>28</td>
<td>oat milk</td>
<td>0.68</td>
<td>1/10 of oats (23)</td>
</tr>
<tr>
<td>29</td>
<td>rice milk</td>
<td>1.06</td>
<td>1/5 of rice (25)</td>
</tr>
<tr>
<td>30</td>
<td>soy milk</td>
<td>2.37</td>
<td>2/5 of soybean (20)</td>
</tr>
<tr>
<td>31</td>
<td>other fruits</td>
<td>2.7</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>32</td>
<td>apples</td>
<td>2.7</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>33</td>
<td>bananas</td>
<td>2.7</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>GHG (kg CO2e/100g)</td>
<td>Source</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------</td>
<td>--------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>34</td>
<td>berries</td>
<td>2.7</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>35</td>
<td>citrus fruit</td>
<td>2.7</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>36</td>
<td>cabbages and other brassicas (broccoli)</td>
<td>7.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>37</td>
<td>tomatoes</td>
<td>7.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>38</td>
<td>root vegetables</td>
<td>7.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>39</td>
<td>onions and leeks</td>
<td>7.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>40</td>
<td>other vegetables</td>
<td>7.9</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>41</td>
<td>potatoes</td>
<td>5</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>42</td>
<td>cassava and other roots</td>
<td>5</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>43</td>
<td>sugars and sweeteners</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>44</td>
<td>other vegetable oils</td>
<td>0.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>45</td>
<td>soybeans (oil)</td>
<td>0.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>46</td>
<td>palm (oil)</td>
<td>0.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>47</td>
<td>sunflower (oil)</td>
<td>0.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>48</td>
<td>rapeseed/canola (oil)</td>
<td>0.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>49</td>
<td>olives (oil)</td>
<td>0.2</td>
<td>Food Label Toolkit</td>
</tr>
<tr>
<td>50</td>
<td>barley (beer)</td>
<td>9.32</td>
<td>GHG ratio to wheat (24)</td>
</tr>
<tr>
<td>51</td>
<td>wine grapes (wine)</td>
<td>8.64</td>
<td>GHG ratio to fruits (31)</td>
</tr>
<tr>
<td>52</td>
<td>cocoa</td>
<td>2.7</td>
<td>Same as fruits (31)</td>
</tr>
<tr>
<td>53</td>
<td>coffee</td>
<td>2.7</td>
<td>Same as fruits (31)</td>
</tr>
<tr>
<td>54</td>
<td>stimulants &amp; spices misc.</td>
<td>6.75</td>
<td>Average of all veg products</td>
</tr>
<tr>
<td>55</td>
<td>water &amp; beverages</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>56</td>
<td>salt</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>57</td>
<td>vinegar</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>58</td>
<td>sauces &amp; paste</td>
<td>6.75</td>
<td>Average of all veg products</td>
</tr>
<tr>
<td>59</td>
<td>manually adjusted</td>
<td>0</td>
<td>Estimated Individually</td>
</tr>
<tr>
<td>60</td>
<td>human labor</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>61</td>
<td>kitchen supplies</td>
<td>0</td>
<td>By Assumption</td>
</tr>
</tbody>
</table>
## APPENDIX D [WATER FOOTPRINT FACTORS LIST]

<table>
<thead>
<tr>
<th>Category ID</th>
<th>Food Category</th>
<th>FreshwaterWithdrawals (L/FU)</th>
<th>Stress-Weighted Water Use (L/FU)</th>
<th>Data Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beef &amp; buffalo meat</td>
<td>1677.2</td>
<td>61309</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>2</td>
<td>lamb/mutton &amp; goat meat</td>
<td>461.2</td>
<td>258.9</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>3</td>
<td>pork (pig meat)</td>
<td>1810.3</td>
<td>54242.7</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>4</td>
<td>poultry (chicken, turkey)</td>
<td>370.3</td>
<td>333.5</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>5</td>
<td>butter</td>
<td>1010.176</td>
<td>50055.168</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>6</td>
<td>cheese</td>
<td>1559.3</td>
<td>80463.1</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>7</td>
<td>ice cream</td>
<td>16.2</td>
<td>17597.52</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>8</td>
<td>cream</td>
<td>28.08</td>
<td>30502.368</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>9</td>
<td>milk (cow's milk)</td>
<td>197.3</td>
<td>9776.4</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>10</td>
<td>yogurt</td>
<td>262.409</td>
<td>13002.612</td>
<td>GHG ratio to milk (9)</td>
</tr>
<tr>
<td>11</td>
<td>eggs</td>
<td>632.9</td>
<td>18621</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>12</td>
<td>fish (finfish)</td>
<td>1580.5</td>
<td>8483.4</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>13</td>
<td>crustaceans (shrimp/prawns)</td>
<td>1207.8</td>
<td>48737.6</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>14</td>
<td>mollusks</td>
<td>0</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>15</td>
<td>animal fats</td>
<td>1810.3</td>
<td>54242.7</td>
<td>Same as pork (3)</td>
</tr>
<tr>
<td>16</td>
<td>other legumes</td>
<td>0</td>
<td>0</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>17</td>
<td>beans and pulses (dried)</td>
<td>0</td>
<td>0</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>18</td>
<td>peas</td>
<td>0</td>
<td>0</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>19</td>
<td>peanuts/groundnuts</td>
<td>900.2</td>
<td>44352.1</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>20</td>
<td>soybeans/tofu</td>
<td>6.6</td>
<td>32.4</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>21</td>
<td>other grains/cereals</td>
<td>677.075</td>
<td>10563.3</td>
<td>Average of all grains</td>
</tr>
<tr>
<td>22</td>
<td>corn (maize)</td>
<td>43.9</td>
<td>349.6</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>23</td>
<td>oats (oatmeal)</td>
<td>670.3</td>
<td>24456.3</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>24</td>
<td>wheat/rye (bread, pasta, baked goods)</td>
<td>419.2</td>
<td>12821.7</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>25</td>
<td>rice</td>
<td>1574.9</td>
<td>4625.6</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>26</td>
<td>tree nuts and seeds</td>
<td>1823.3</td>
<td>129364.3</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>27</td>
<td>almond milk</td>
<td>455.825</td>
<td>32341.075</td>
<td>1/4 of nuts (19)</td>
</tr>
<tr>
<td>28</td>
<td>oat milk</td>
<td>67.03</td>
<td>2445.63</td>
<td>1/10 of oats (23)</td>
</tr>
<tr>
<td>29</td>
<td>rice milk</td>
<td>314.98</td>
<td>925.12</td>
<td>1/5 of rice (25)</td>
</tr>
<tr>
<td>30</td>
<td>soy milk</td>
<td>1.3</td>
<td>6.2</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td>31</td>
<td>other fruits</td>
<td>3.5</td>
<td>4.7</td>
<td>Poore &amp; Newecek</td>
</tr>
<tr>
<td></td>
<td>Item</td>
<td>114.5</td>
<td>1024.7</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>---</td>
<td>------------------------------------------</td>
<td>-------</td>
<td>--------</td>
<td>-----------------</td>
</tr>
<tr>
<td>32</td>
<td>apples</td>
<td>114.5</td>
<td>1024.7</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>33</td>
<td>bananas</td>
<td>1</td>
<td>31.3</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>34</td>
<td>berries</td>
<td>403.5</td>
<td>16245.1</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>35</td>
<td>citrus fruit</td>
<td>37.4</td>
<td>1345.5</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>36</td>
<td>cabbages and other brassicas (broccoli)</td>
<td>54.5</td>
<td>2483.4</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>37</td>
<td>tomatoes</td>
<td>77</td>
<td>4480.7</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>38</td>
<td>root vegetables</td>
<td>9.9</td>
<td>37.9</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>39</td>
<td>onions and leeks</td>
<td>1.9</td>
<td>57</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>40</td>
<td>other vegetables</td>
<td>81.3</td>
<td>2939.5</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>41</td>
<td>potatoes</td>
<td>2.6</td>
<td>78.3</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>42</td>
<td>cassava and other roots</td>
<td>9.9</td>
<td>37.9</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>43</td>
<td>sugars and sweeteners</td>
<td>10.1</td>
<td>65.2</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>44</td>
<td>other vegetable oils</td>
<td>67.5</td>
<td>4937.72</td>
<td>Average of all veg oils</td>
</tr>
<tr>
<td>45</td>
<td>soybeans (oil)</td>
<td>1.6</td>
<td>7.8</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>46</td>
<td>palm (oil)</td>
<td>6.4</td>
<td>34.8</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>47</td>
<td>sunflower (oil)</td>
<td>10.2</td>
<td>236.7</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>48</td>
<td>rapeseed/canola (oil)</td>
<td>1.4</td>
<td>13.6</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>49</td>
<td>olives (oil)</td>
<td>317.9</td>
<td>24395.7</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>50</td>
<td>barley (beer)</td>
<td>7</td>
<td>27.3</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>51</td>
<td>wine grapes (wine)</td>
<td>4.5</td>
<td>60.4</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>52</td>
<td>cocoa</td>
<td>24.9</td>
<td>220.3</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>53</td>
<td>coffee</td>
<td>33.3</td>
<td>340.7</td>
<td>Poore &amp; Newceck</td>
</tr>
<tr>
<td>54</td>
<td>stimulants &amp; spices misc.</td>
<td>24.9</td>
<td>220.3</td>
<td>Same as cocoa (52)</td>
</tr>
<tr>
<td>55</td>
<td>water &amp; beverages</td>
<td>1</td>
<td>1</td>
<td>By Assumption</td>
</tr>
<tr>
<td>56</td>
<td>salt</td>
<td>0</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>57</td>
<td>vinegar</td>
<td>1</td>
<td>1</td>
<td>By Assumption</td>
</tr>
<tr>
<td>58</td>
<td>sauces &amp; paste</td>
<td>20.225</td>
<td>1134.925</td>
<td>½ water + ¼ tomato + ¼ onion</td>
</tr>
<tr>
<td>59</td>
<td>manually adjusted</td>
<td>0</td>
<td>0</td>
<td>Estimated Individually</td>
</tr>
<tr>
<td>60</td>
<td>human labor</td>
<td>0</td>
<td>0</td>
<td>By Assumption</td>
</tr>
<tr>
<td>61</td>
<td>kitchen supplies</td>
<td>0</td>
<td>0</td>
<td>By Assumption</td>
</tr>
</tbody>
</table>