Valentina Colorado
Dr. Rei Sanchez-Arias
COP 5090 Scientific Computation and Programming
06 November 2023

My Fitness Friend

Introduction

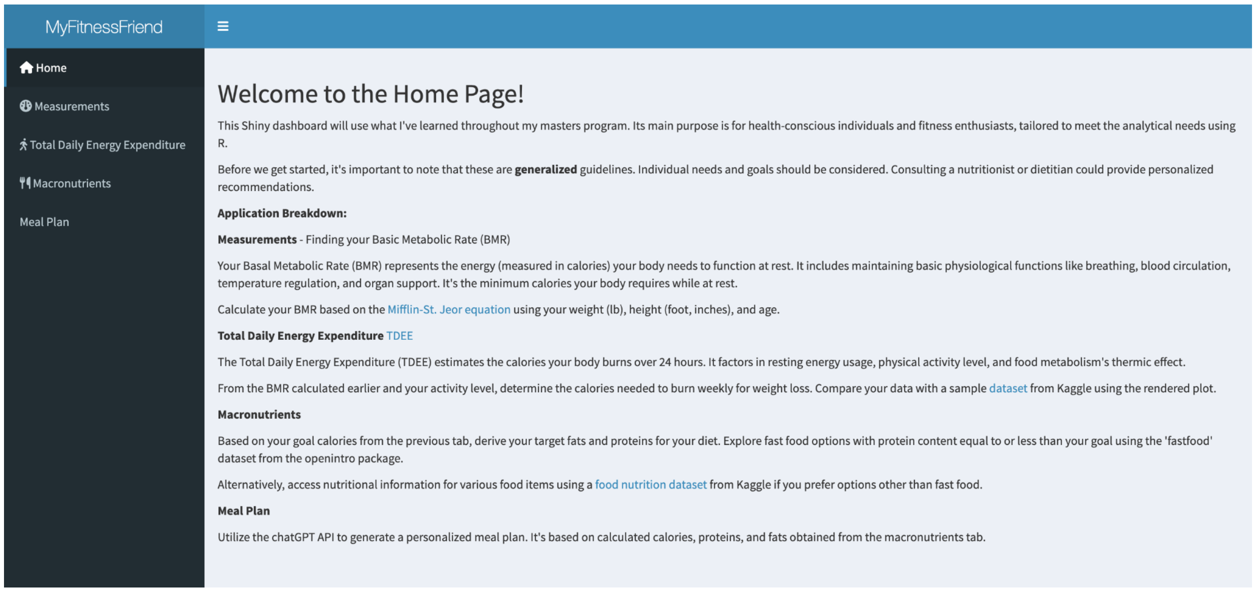
This application aims to use what I learned this semester and apply it to a comprehensive shiny application. This shiny app makes it easy for someone to input metrics like age, current weight, goal weight, and height, calculate their basic metabolic rate and total daily energy expenditure, and find their suggested macronutrients for the day. A more in-depth explanation for each tab is broken down in the paper's methods section. Each tab name is bolded and italicized. I included screenshots of the app working at the end of every section to get a good view of everything working as it should. Finally, at the bottom is a conclusion section where I reference the link for the shiny app the GitHub repo, and review future work.

Methods

Home

When a user loads the application, the first thing they see is the *home* tab. This is where a user can learn more about what the application does. I set up my shiny application as a dashboard by importing the library(shinydashboard). This made having a clear structure for my application easy since the dashboard comprises of three parts: a header, a sidebar, and a body. Since this was my first time making a shiny application in R, I needed some help figuring out how to structure my page. Luckily, an open-source website, SUE ShinyUiEditor, helps you build Shiny application UIs by dragging and dropping. Generates clean and proper code as you build. This webpage is maintained by POSIT, formally known as R-studio. I couldn't have done as much without this page[1].

Once I laid out my structure, I started to craft the home tab. The primary purpose of this page was so that when a user loaded up the application, they understood the general purpose of it and could understand some of the terminology used to make the other tabs. I am by no means a nutritionist. I got my undergraduate degree in computer science and, at one point, really enjoyed fitness. Everything within this application is a general guideline, and the user should not take my measurements as something concrete to go by. To ensure the user understands what my app wants to measure, I defined basic metabolic rate as the energy (measured in calories) your body needs to function at rest, maintaining essential physiological functions like breathing, circulating blood, regulating body temperature, and supporting organ functions. In other words, it's the minimum calories your body requires to sustain itself while at rest [2]. This is useful to know because, with this, we can calculate how many calories to cut out each day [3]. The Harris-Benedict equation and the Mifflin-St Jeor equation are two well-known ways to calculate a person's basic metabolic rate. For this application, we will only be using the Mifflin-St Jeor equation. Recent studies have shown that this equation is more useful when calculating a person's basic metabolic rate [4]. The second important keyword for this application is Total Daily Energy Expenditure. This estimates the amount of energy (or number of calories) your body burns over 24 hours, factoring in how much energy it uses while at rest, your typical level of physical activity, and the thermic effect of food metabolism[5]. This is useful because it lets people know how many calories they need to burn daily. This is different from the basic metabolic rate because the total daily energy expenditure also considers a user's level of daily physical activity.



Welcome to the Home Page!

This Shiny dashboard will use what I've learned throughout my masters program. Its main purpose is for health-conscious individuals and fitness enthusiasts, tailored to meet the analytical needs using R.

Before we get started, it's important to note that these are **generalized** guidelines. Individual needs and goals should be considered. Consulting a nutritionist or dietitian could provide personalized recommendations.

Application Breakdown:

Measurements - Finding your Basic Metabolic Rate (BMR)

Your Basal Metabolic Rate (BMR) represents the energy (measured in calories) your body needs to function at rest. It includes maintaining basic physiological functions like breathing, blood circulation, temperature regulation, and organ support. It's the minimum calories your body requires while at rest.

Calculate your BMR based on the Mifflin-St. Jeor equation using your weight (lb), height (foot, inches), and age.

Total Daily Energy Expenditure TDEE

The Total Daily Energy Expenditure (TDEE) estimates the calories your body burns over 24 hours. It factors in resting energy usage, physical activity level, and food metabolism's thermic effect.

From the BMR calculated earlier and your activity level, determine the calories needed to burn weekly for weight loss. Compare your data with a sample dataset from Kaggle using the rendered plot.

Macronutrients

Based on your goal calories from the previous tab, derive your target fats and proteins for your diet. Explore fast food options with protein content equal to or less than your goal using the 'fastfood' dataset from the openintro package.

Alternatively, access nutritional information for various food items using a food nutrition dataset from Kaggle if you prefer options other than fast food.

Meal Plan

Utilize the chatGPT API to generate a personalized meal plan. It's based on calculated calories, proteins, and fats obtained from the macronutrients tab.

Basic Metabolic Rate

Once the user understands what this application does, they can move on to the basic metabolic rate tab. To create this, I created a new `tabItem()` and filled it in with the `tabName,` so the server knows what to load when the application is selected, and then created a `fluidRow()` that tells R/Shiny that you want to put multiple elements into the same row of the page. This was useful because it allowed me to put input boxes for each input. You need weight, height, and age to calculate a user's basic metabolic rate. The user's weight and age are calculated using Shiny's card_body() and textInput() functions. Since everyone is different, I did not include any preset values for these. Between weight and age is a numericInput() function where a user can enter their height in feet and inches. Below all three text boxes are two action buttons: "Calculate BMR Male" and "Calculate BMR Female." The Mifflin-St Jeor requires that males and females have different equations. Once the user clicks the "Calculate BMR" button, the server function kicks off, and an observeEvent() function runs.

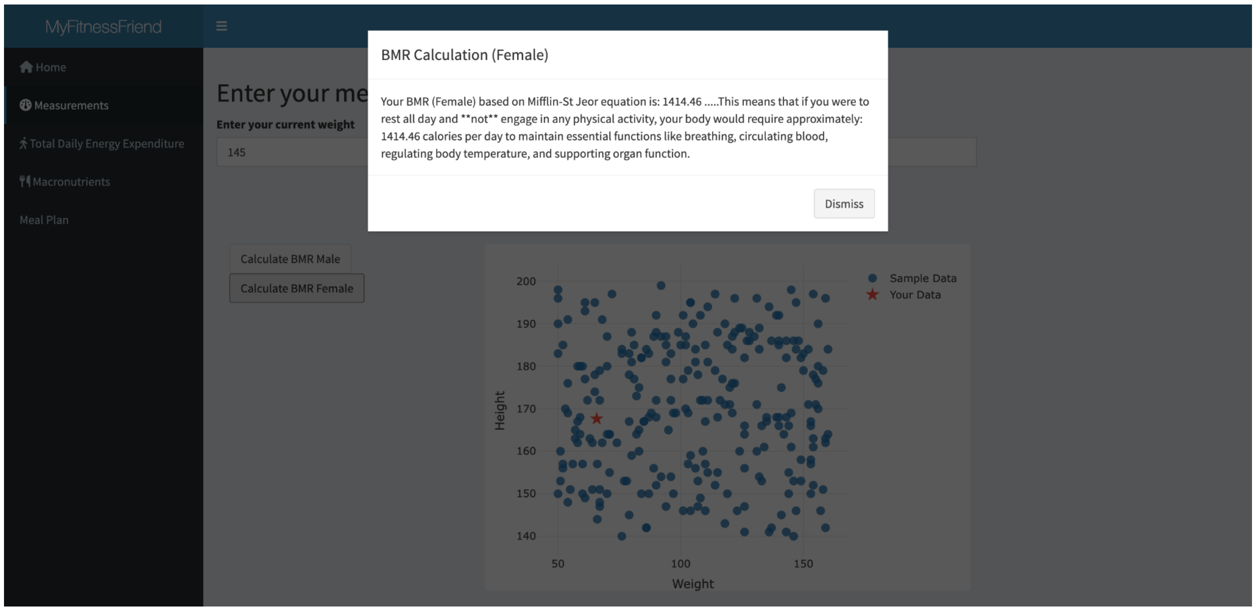
I live in the United States, where, unlike the rest of the world, we follow the imperial system. The application must convert everything into the metric system when users enter their information. Instead of calculating everything inside my app.R file, I made an R package, "MyFitnessFriend"! The process of making a package was actually super easy. R-studio has a built-in option where you can select the create new R package when making a new project. Once that ran, I started making R scripts with all my functions. My first built-in function is the weight-conversion.R. This file comprises two functions. `weight.kg` and `weight.lb`.

The weight.kg function takes one input, the weight in pounds from the user's weight, and then converts that number into kilograms by dividing the entered weight by 2.20463. The weight.lb function does the opposite. It takes the user's weight in kilograms and then returns the weight in kilograms. The file height-conversion.R has the function `height.cm`. This function takes two integer inputs, feet and inches. These two are then converted and added to get the user's height in cm. Another file I made is BMR.R. This file consists of four different functions. All three functions take inputs: x: weight in kilograms, y: height in cm, and age. The first two functions are BMR_male.Mifflin and BMR_female.Mifflin, the difference between the two equations is shown below. The other two functions are BMR_male.Harris BMR_female.Harris these two functions use the Harris-Benedict equation to calculate a user's basic metabolic rate.

Men: $(10 \times \text{weight in kg}) + (6.25 \times \text{height in cm}) - (5 \times \text{age in years}) + 5$

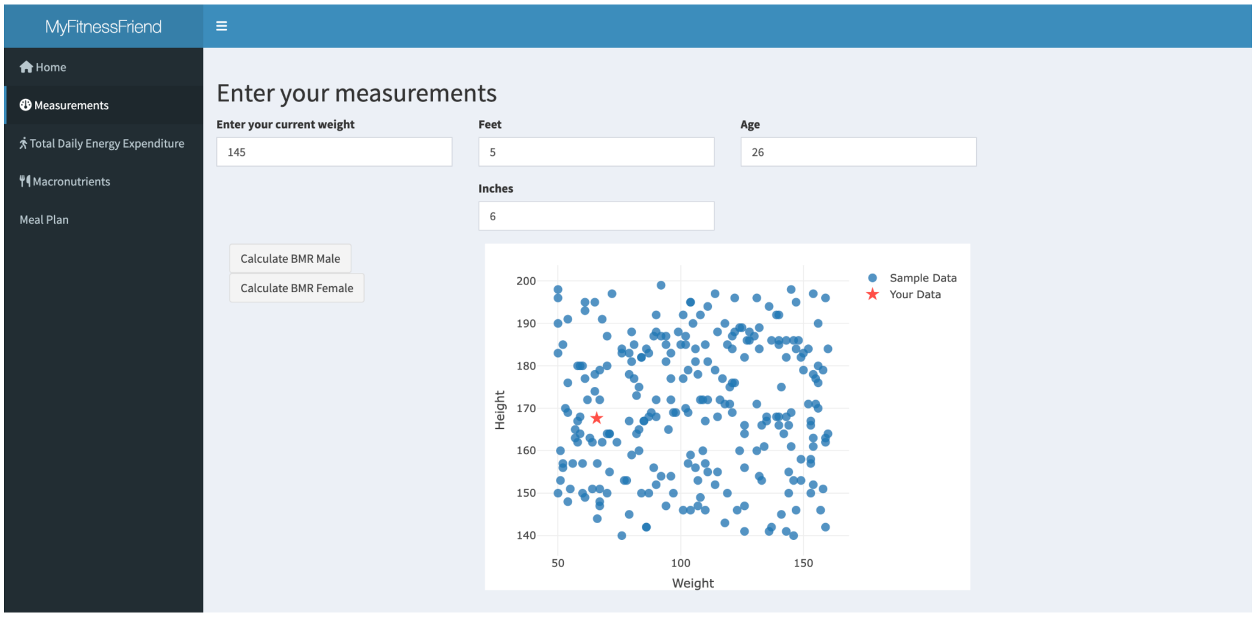
Women: $(10 \times \text{weight in kg}) + (6.25 \times \text{height in cm}) - (5 \times \text{age in years}) - 161$.

After the necessary calculations are performed on the back end, the GUI screen displays a modal with the prompt "Your BMR (gender) based on Mifflin-St Jeor equation is: x .....This means that if you were to rest all day and **not** engage in any physical activity, your body would require approximately x calories per day to maintain essential functions like breathing, circulating blood, regulating body temperature, and supporting organ function." the x in the prompt is dynamic and changes every time a user enters their information. After you dismiss the model, a scatterplot built by Plotly appears. It consists of two data points: a blue circle showing the sample data and a red star representing your inputted data. I searched for multiple different datasets with more than 100 observations, with gender, weight, and height. It was hard to find an open-source dataset with everything I was looking for. I found one on Kaggle, the 500 Person Gender-Height-Weight-Body Mass Index dataset by Yasin Ersever[6]. Although it was last updated five years ago, it had everything I was looking for, with information on 500 individuals' gender, height, weight, and calculated Body Mass Index (BMI). The dataset was constructed through the generation of synthetic data to simulate the physical attributes of individuals. The randomly generated dataset includes information on individuals' heights (measured in centimeters) and weights (measured in kilograms). This dataset also included BMI values that were calculated using height and weight data points according to the standard BMI formula. The BMI values are categorized into six groups, ranging from extremely weak to extreme obesity, providing insights into the health status derived from BMI calculations. I did not feel comfortable including this feature when comparing a real user data point, so I filtered it out. I think it's important to point out that with synthetic data, it's important to acknowledge the limitations. While a helpful tool, synthetic data may not precisely mirror the complexity and diversity of real-world information. It's generated using assumptions or models, potentially oversimplifying or omitting certain intricacies present in actual data. Moreover, synthetic data might not fully capture the variability and biases inherent in genuine datasets. This discrepancy could lead to a lack of subtle patterns, outliers, or unexpected correlations often present in authentic datasets. Even though it's synthetic data, I thought it would be useful for a user to visualize their information clearly compared to other similar data points.



BMR Calculation (Female)

Your BMR (Female) based on Mifflin-St Jeor equation is: 1414.46 .....This means that if you were to rest all day and **not** engage in any physical activity, your body would require approximately: 1414.46 calories per day to maintain essential functions like breathing, circulating blood, regulating body temperature, and supporting organ function.



Enter your measurements

Enter your current weight: 145

Feet: 5

Age: 26

Inches: 6

Total Daily Energy Expenditure

The second tab calculates a user's Total Daily Energy Expenditure, which calculates how many calories your body burns daily based on your activity level. Similar to the basic metabolic rate tab in the UI function, I added another `tabItem()` and filled it in with the `tabName,` so the server knows what to load when the application is selected, and then created another `fluidRow()` to hold the grid_cards that hold the UI functions. At the top, there is a textInput()

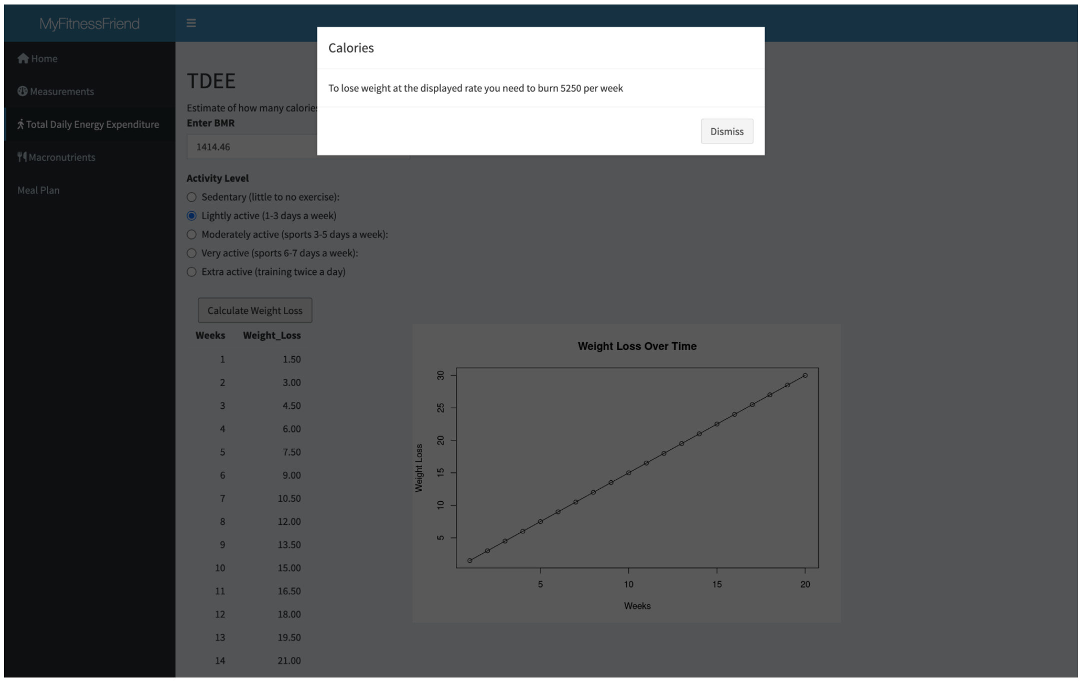
function where a user inputs their Basal Metabolic Rate (BMR) from the first tab. I wanted the app to feel cohesive. If users goes through the tabs consecutively, their basic metabolic rate will automatically populate from the first calculation. Right below the text input there is a radioButtons() function that allows a user to choose their activity level from options such as sedentary (little to no exercise), lightly active (1-3 days a week), moderately active (sports 3-5 days a week), very active (sports 6-7 days a week), or extra active (training twice a day). Once they've entered this information, they can click the action button at the bottom to calculate their weight loss.

Once a user clicks that action button the observeEvent(input$calculateButtonTDEE) function is kicked off. To calculate a user's total daily energy expenditure, I used a function from MyFitnessFriend package, "calculate_weight_loss." This function takes three inputs: basic metabolic rate, activity multiplier, and number of weeks. The basic metabolic rate comes from the last tab, or users can input it themselves if they already know it. The activity multiplier comes from the fitness level selected by the user. The number of weeks is set to 20. Inside the calculate_weight_loss function to determine the daily calorie deficit based on various activity levels. I used a switch function where different activity multipliers are associated with specific calorie deficit values. For instance, a sedentary activity level (activity_multiplier = 1.2) corresponds to a calorie deficit of 500 per day, while an extra-active level (activity_multiplier = 1.9) indicates a deficit of 1500 calories daily. This setup allows for easy adjustment of calorie deficits depending on different activity intensities.

Then, the total daily energy expenditure is calculated by multiplying the basic metabolic rate by the activity multiplier. Then, it computes the weekly calorie deficit and weight loss based on this total daily energy expenditure, assuming 3500 calories result in 1 pound of weight loss. Then, it generates a data frame to store the projected weight loss over a specified number of weeks, displaying the cumulative weight loss progression week by week. Using that dataframe, it generates a line plot. The plot stores the weight loss progression over time by creating a line plot using the generated data frame 'weight_loss_data.' The plotted line graphically represents the cumulative weight loss against the number of weeks, offering a visual understanding of the projected weight loss trajectory. For this total daily energy expenditure tab, the calculated weight loss function returns in a list as a table with the user weight_loss data, a plot with the plot_data, and calories that have the calorie deficit per week.

Once the user clicks the "calculate_weight_loss" button, I used another show showModal() modalDialog() function. The heading shows "Calories" to let the user know what the pop-up is for. Inside the body of the modal, it has the prompt, "To lose weight at the displayed rate, you need to burn x amount of calories per week." The x is dynamic and changes with the BMR and activity level selected. For sedentary, its about 3500 calories. After this message is dismissed, a table and graph are populated at the bottom. The table is the weight_loss_data dataframe calculated in the "calculate_weight_loss" function. Similar to the modal, the outputted information is dynamic and changes based on the user's selected activity level. If a user selects a sedentary activity level, their projected weight loss is about a pound per week. Without a consistent way to track what the user eats and how many calories they burn per workout, it was hard to predict. This projection is the same for anyone who selects sedentary. To the right of the table, the weight loss over time plot was also returned from the "calculate_weight_loss" function. I liked this plot because it shows users how many pounds they can lose if they exercise regularly. The functions are pretty similar for the lightly active activity level—the modalDialog changes to their expected calories which is about 5200. The table is updated, and they're projected to lose 1.5 pounds per week, also displayed in the plot. A user with a moderately active fitness level is expected to burn 7000 per week. The updated weight loss table shows that if they do this, they are projected to lose 2 pounds per week. An individual with a very active activity level is expected to burn 8750 calories weekly. This rate shows their projected weight loss per week at 2.5 pounds. The last option is for an individual whose activity level is extra active (training twice a day). At this rate, they are expected to burn 10,500 calories per week and lose 3 pounds per week!

Seeing this, I was hesitant. I wasn't sure how well my app projected a user's weight loss. With working full time and being a full time student I admidently have not been as active as I know I could be. So, I couldn't use my personal experience to test my calculations. My brothers are both active individuals, and they told me about a fitness challenge, 75 HARD. 75 HARD is a transformative mental toughness program created by Andy Frisella. This program involves completing five daily tasks for 75 days without compromises or substitutions. It's designed to elevate discipline, fortitude, and various traits while promoting physical and mental growth through strict adherence to tasks like a structured diet, workouts, reading, water intake, and progress tracking[7]. One key component of this program is the user is expected to work twice a day for 75 days. This consistent workout paired with a strict diet gives a user excellent results. Everyone is different, so some people lose more weight than others, but according to this medium article, Some individuals shed 10, 20, or even 30 pounds during the 75 days (10 weeks)[8]. This was fun to see because, according to my application, at ten weeks with an extra active fitness level, a user is expected to lose 30 pounds.



Calories

To lose weight at the displayed rate you need to burn 5250 per week

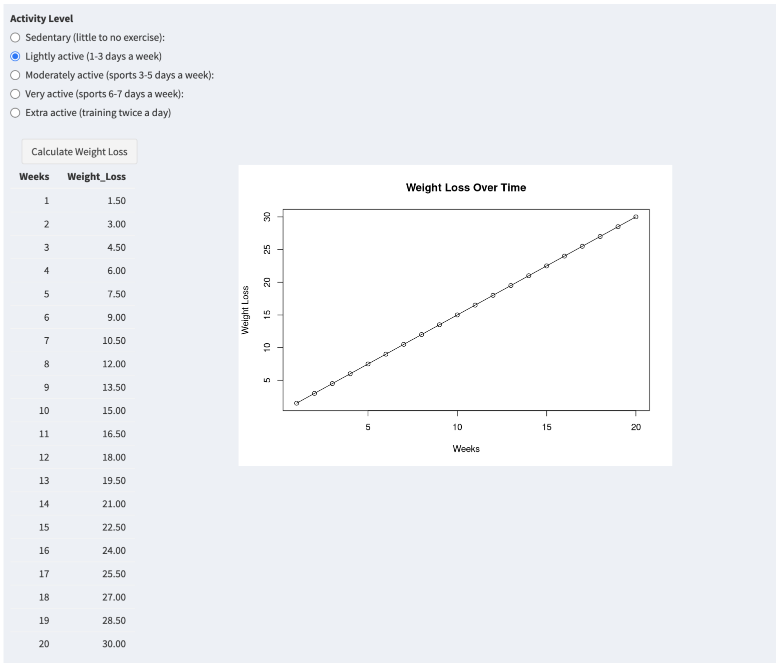
TDEE

Estimate of how many calorie

Enter BMR: 1414.46

Activity Level
- Sedentary (little to no exercise):
- [✓ Lightly active (1-3 days a week)]
- Moderately active (sports 3-5 days a week):
- Very active (sports 6-7 days a week):
- Extra active (training twice a day)

Weeks	Weight_Loss
1	1.50
2	3.00
3	4.50
4	6.00
5	7.50
6	9.00
7	10.50
8	12.00
9	13.50
10	15.00
11	16.50
12	18.00
13	19.50
14	21.00



Activity Level

- Sedentary (little to no exercise):
- [✓ Lightly active (1-3 days a week)]
- Moderately active (sports 3-5 days a week):
- Very active (sports 6-7 days a week):
- Extra active (training twice a day)

Calculate Weight Loss

Weeks	Weight_Loss
1	1.50
2	3.00
3	4.50
4	6.00
5	7.50
6	9.00
7	10.50
8	12.00
9	13.50
10	15.00
11	16.50
12	18.00
13	19.50
14	21.00
15	22.50
16	24.00
17	25.50
18	27.00
19	28.50
20	30.00

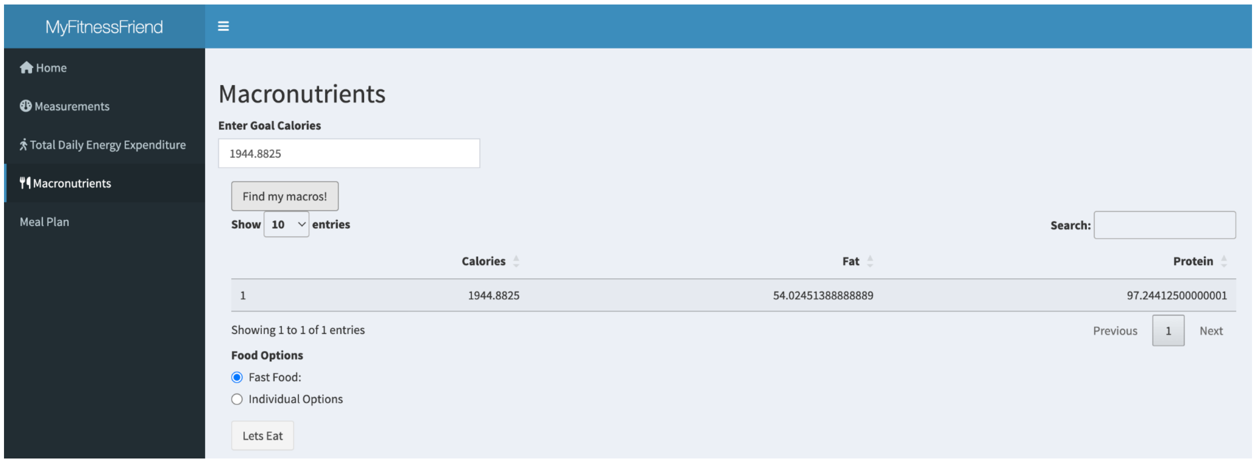
Macronutrients

Now that we know how many calories we need to burn per week, it would be helpful to include how much a person needs to eat daily. To get an accurate number of calories, I made the calculations inside of the “calculate_weight_loss” function. This is calculated by multiplying the user's basic metabolic rate by their activity level from the total daily energy expenditure. When you first enter this tab the user can either enter their own amount of goal calories or it is automatically imported from the last tab. After they enter the calories, the user can select the Find my macros!” button. To better understand what they should eat in a day, the calories are broken down into how much fat and protein the user should eat. For fat, first, I multiplied the recommended_calories by .25. Fats are typically measured in grams, so I divided it by 9 to get the appropriate amount of grams. To find the appropriate amount of protein, I assumed 20% of total calories are from protein intake. I multiplied the recommended_calories by .20 and then divided by 4 to get the proteins returned in grams.

After the button is pressed, the user is presented with a table. This table lists the amount of calories, fats, and protein recommended for that user. Under the table, there is another radio input option. Here, the user can select fast food or individual options. The user selects what they want and then clicks the “Let's Eat!” action button. Even though having fast food restaurants seems counterintuitive for a health application, I wanted to give the user an option of 5 random food items from some of the most notable fast food options. All the options are less than or equal to the protein the application recommended. This dataset was readily available from the openintro package. I used the fast-food library. The "Nutrition in fast food" dataset comprises information on the nutritional content of 515 items across various fast-food restaurants. It offers an extensive breakdown of key nutritional components, such as calories, fats (including subcategories like saturated and trans fats), cholesterol, sodium, carbohydrates (including fiber and sugar content), protein, and vitamins (A and C), along with calcium levels. Each entry in this dataset includes details on the restaurant name, item name, and whether it's categorized as a salad or not. This dataset provides a comprehensive view of the nutritional composition of fast food items, allowing for in-depth analysis and comparisons among different restaurant offerings, aiding in understanding dietary implications, and informing healthier food choices in the fast-food landscape[9]. Although this dataset has many features filled with helpful information about the food, I reduced the dataset only to display the items, calories, fats, and protein to keep everything the same.

Just in case the user doesn't want anything from a fast food place, I wanted them to have the option of seeing the nutritional value of different individual items. I found another dataset from Kaggle. This dataset is the Food Nutrition Dataset. This dataset is a comprehensive compilation providing detailed information on various food items, encompassing their nutritional composition. 96 columns offer a wide array of nutritional insights for each food item. This dataset includes valuable data on several micronutrients like vitamins and minerals such as alpha-carotene, beta-carotene, beta-cryptoxanthin, and choline, measured in micrograms (mcg) or milligrams (mg).

Additionally, it encompasses macronutrients like carbohydrates, cholesterol, and proteins, measured in grams (g), shedding light on critical dietary components. Each entry in this dataset features a unique ID for the food item, a category description, and some subcategories. The dataset appears well-documented, offering in-depth information valuable for dietary analysis, research, and exploratory data analysis (EDA) in food and nutrition studies[10]. Like the fast food dataset, I only kept the protein, calories, and fat for the user to see. I did not want to overwhelm the user with a lot of information



Macronutrients

Enter Goal Calories: 1944.8825

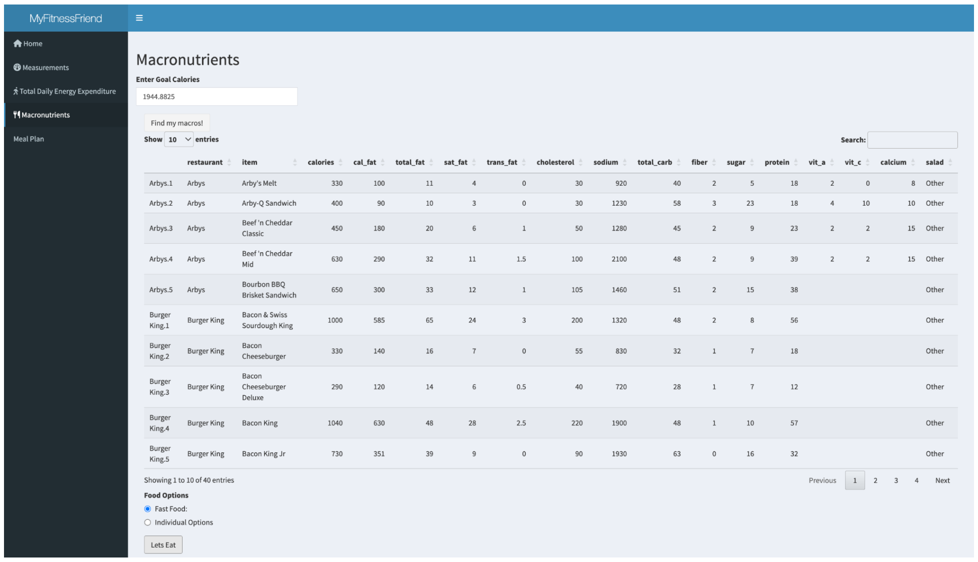
Show 10 entries

Search:

	Calories	Fat	Protein
1	1944.8825	54.02451388888889	97.24412500000001

Showing 1 to 1 of 1 entries

Food Options
- [✓ Fast Food:]
- Individual Options



Macronutrients

Enter Goal Calories

1944.8825

Find my macros!

Show 10 entries

Search:

	restaurant	item	calories	cal_fat	total_fat	sat_fat	trans_fat	cholesterol	sodium	total_carb	fiber	sugar	protein	vit_a	vit_c	calcium	salad
Arbys.1	Arbys	Arby's Melt	330	100	11	4	0	30	920	40	2	5	18	2	0	8	Other
Arbys.2	Arbys	Arby-Q Sandwich	400	90	10	3	0	30	1230	58	3	23	18	4	10	10	Other
Arbys.3	Arbys	Beef 'n Cheddar Classic	450	180	20	6	1	50	1280	45	2	9	23	2	2	15	Other
Arbys.4	Arbys	Beef 'n Cheddar Mid	630	290	32	11	1.5	100	2100	48	2	9	39	2	2	15	Other
Arbys.5	Arbys	Bourbon BBQ Brisket Sandwich	650	300	33	12	1	105	1460	51	2	15	38				Other
Burger King.1	Burger King	Bacon & Swiss Sourdough King	1000	585	65	24	3	200	1320	48	2	8	56				Other
Burger King.2	Burger King	Bacon Cheeseburger	330	140	16	7	0	55	830	32	1	7	18				Other
Burger King.3	Burger King	Bacon Cheeseburger Deluxe	290	120	14	6	0.5	40	720	28	1	7	12				Other
Burger King.4	Burger King	Bacon King	1040	630	48	28	2.5	220	1900	48	1	10	57				Other
Burger King.5	Burger King	Bacon King Jr	730	351	39	9	0	90	1930	63	0	16	32				Other

Showing 1 to 10 of 40 entries

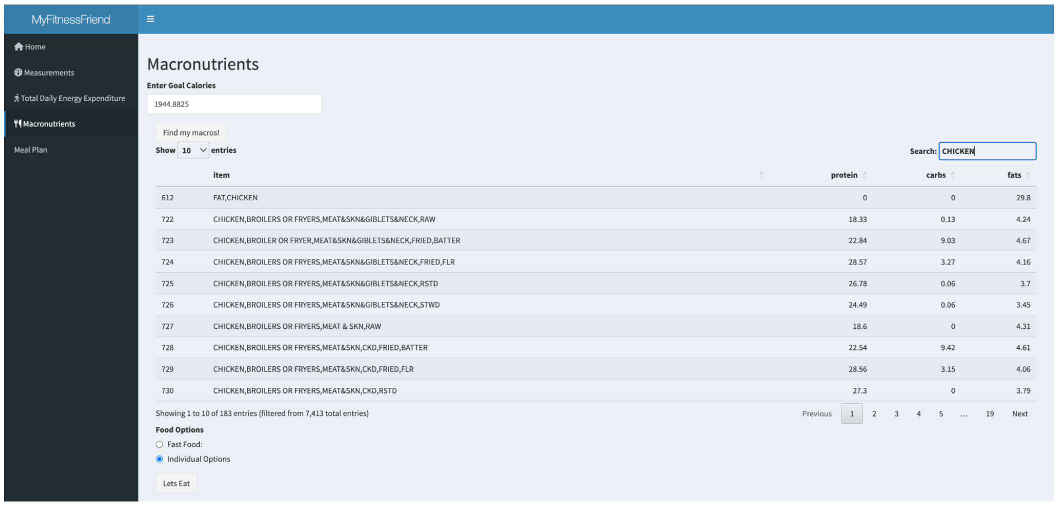
Previous 1 2 3 4 Next

Food Options

[✓ Fast Food:]

Individual Options

Lets Eat



Macronutrients

Enter Goal Calories

1944.8825

Find my macros!

Show 10 entries

Search: CHICKEN

	item	protein	carbs	fats
612	FAT,CHICKEN	0	0	29.8
722	CHICKEN,BROILERS OR FRYERS,MEAT&SKN&GIBLETS&NECK,RAW	18.33	0.13	4.24
723	CHICKEN,BROILER OR FRYER,MEAT&SKN&GIBLETS&NECK,FRIED,BATTER	22.84	9.03	4.67
724	CHICKEN,BROILERS OR FRYERS,MEAT&SKN&GIBLETS&NECK,FRIED,FLR	28.57	3.27	4.16
725	CHICKEN,BROILERS OR FRYERS,MEAT&SKN&GIBLETS&NECK,RSTD	26.78	0.06	3.7
726	CHICKEN,BROILERS OR FRYERS,MEAT&SKN&GIBLETS&NECK,STWD	24.49	0.06	3.45
727	CHICKEN,BROILERS OR FRYERS,MEAT & SKN,RAW	18.6	0	4.31
728	CHICKEN,BROILERS OR FRYERS,MEAT&SKN,CKD,FRIED,BATTER	22.54	9.42	4.61
729	CHICKEN,BROILERS OR FRYERS,MEAT&SKN,CKD,FRIED,FLR	28.56	3.15	4.06
730	CHICKEN,BROILERS OR FRYERS,MEAT&SKN,CKD,RSTD	27.3	0	3.79

Showing 1 to 10 of 183 entries (filtered from 7,413 total entries)

Previous 1 2 3 4 5 … 19 Next

Food Options

Fast Food:

[✓ Individual Options]

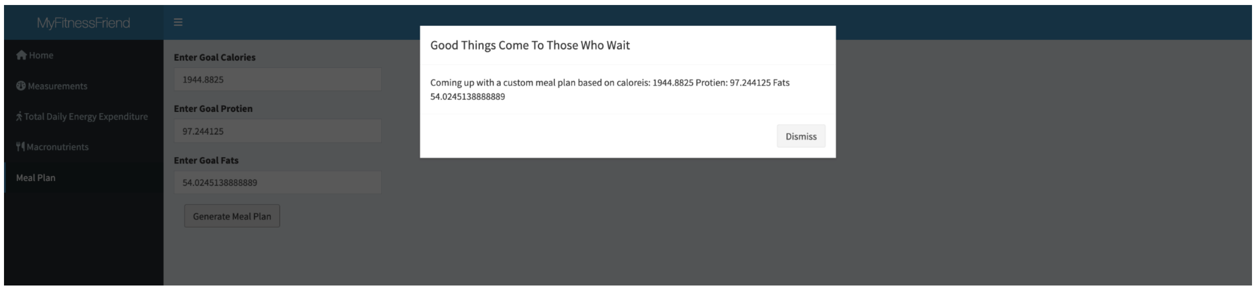
Lets Eat

Meal Plan

Sometimes, it's challenging to figure out what to cook even when you know your target macronutrients. To make it simple for my user, I added a tab, Meal Plan. Here, the user can enter their target calories, protein, and fat, or if it was calculated from the last tab, it is automatically imported. Once the user enters their macros, they can click the action button at the bottom, Generate Meal Plan. This then runs the observeEvent(input$chatGPT). The user is presented with a new modalDialog(). The title is "Good Things Come To Those Who Wait." it lets the user know that the application is trying to generate a custom meal plan. The prompt shows "Coming up with a custom meal plan based on Calories: x Protein: y Fats z." The calories, protein, and fats are all dynamic and updated based on the user's input. Inside the observed event, a new variable meal plan is made. It saves the output from the generateMealPlan function. This function comes from my package, MyFitnessFriend gpt.R file.

This R script defines a function called generateMealPlan that utilizes the ChatGPT API to create a personalized meal plan based on a user's specified daily calorie intake, protein, and fat requirements. The function uses three parameters: calories, protein, and fats, representing the user's desired nutritional values. It constructs a prompt using this information and sends a request to the OpenAI ChatGPT API. The API response provides food options along with their macronutrient details and recipe ideas. This script requires the shiny, httr, and jsonlite libraries to function, and it's designed to generate meal plans leveraging artificial intelligence capabilities provided by the ChatGPT model. The example demonstrates how to use this function by passing sample values for calories, protein, and fats. Figuring out how to use ChatGPT's API functionality wasn't as bad as I thought it would be. There is an in-depth tutorial hosted on the listening data website that makes the implementation easy to do[11].

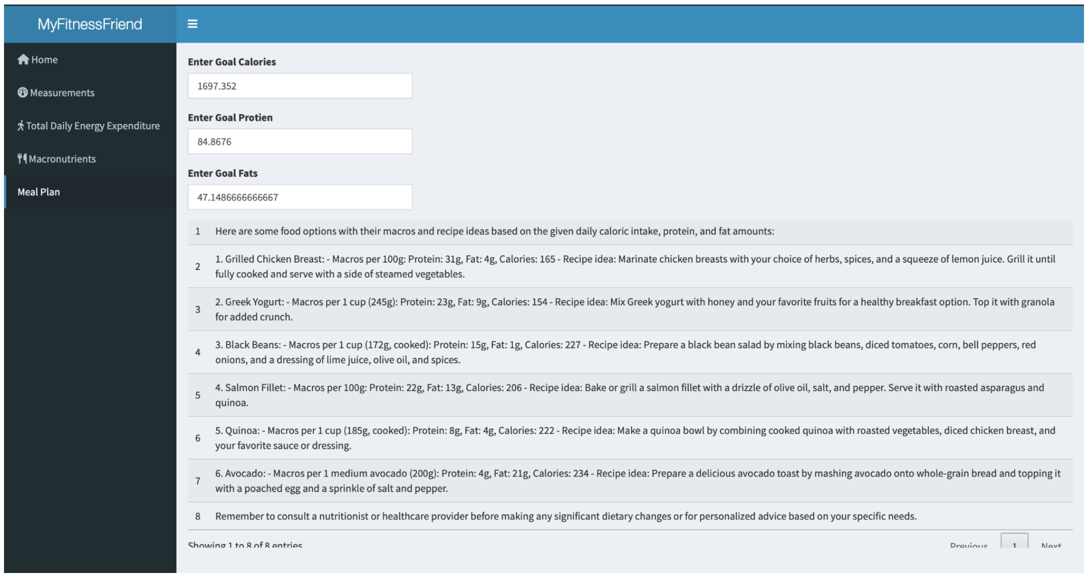
Once Chatgpt returns the response, it's stored in the mealPlan object. This response returns a lot of information in JASON format. For the purpose of this project, only the 'content' section from each 'choice' within the mealPlan. This 'content' typically contains textual descriptions or recommendations generated by the ChatGPT model. The result is a collection of text segments, one for each 'choice' in the mealPlan. Once mealPlan gets that information, the code concatenates these text segments into a single character vector named mealContent. The concatenation occurs by splitting the mealContent based on the "\n\n" pattern using the strsplit function. This results in an unstructured list of food options and their associated descriptions. To present this data in a more organized way, the code transforms this unstructured list into a structured format suitable for tabular representation. It creates a data frame named mealPlanDF, where each row corresponds to a distinct food option. The 'Food' column within this data frame contains the text segments obtained earlier, separated by the "\n\n" pattern. Finally, the renderDataTable function from the Shiny library is used to present the mealPlanDF data frame as a table in a Shiny web application. This table, named gptTable in the output, showcases the various food options and their respective descriptions or details obtained from the ChatGPT API response. This process enables users to visualize and explore the generated meal plan in a structured tabular format within a Shiny app or web interface, which makes it easier for the user to comprehend and interact with the generated meal plan data. To store my API key, I used. Renviron and .setENV() so I don't break infosec standards by uploading my key to Git Hub.



Enter Goal Calories
1944.8825
Enter Goal Protien
97.244125
Enter Goal Fats
54.0245138888889

Good Things Come To Those Who Wait

Coming up with a custom meal plan based on caloreis: 1944.8825 Protien: 97.244125 Fats 54.0245138888889



Enter Goal Calories: 1697.352

Enter Goal Protien: 84.8676

Enter Goal Fats: 47.1486666666667

1	Here are some food options with their macros and recipe ideas based on the given daily caloric intake, protein, and fat amounts:
2	1. Grilled Chicken Breast: - Macros per 100g: Protein: 31g, Fat: 4g, Calories: 165 - Recipe idea: Marinate chicken breasts with your choice of herbs, spices, and a squeeze of lemon juice. Grill it until fully cooked and serve with a side of steamed vegetables.
3	2. Greek Yogurt: - Macros per 1 cup (245g): Protein: 23g, Fat: 9g, Calories: 154 - Recipe idea: Mix Greek yogurt with honey and your favorite fruits for a healthy breakfast option. Top it with granola for added crunch.
4	3. Black Beans: - Macros per 1 cup (172g, cooked): Protein: 15g, Fat: 1g, Calories: 227 - Recipe idea: Prepare a black bean salad by mixing black beans, diced tomatoes, corn, bell peppers, red onions, and a dressing of lime juice, olive oil, and spices.
5	4. Salmon Fillet: - Macros per 100g: Protein: 22g, Fat: 13g, Calories: 206 - Recipe idea: Bake or grill a salmon fillet with a drizzle of olive oil, salt, and pepper. Serve it with roasted asparagus and quinoa.
6	5. Quinoa: - Macros per 1 cup (185g, cooked): Protein: 8g, Fat: 4g, Calories: 222 - Recipe idea: Make a quinoa bowl by combining cooked quinoa with roasted vegetables, diced chicken breast, and your favorite sauce or dressing.
7	6. Avocado: - Macros per 1 medium avocado (200g): Protein: 4g, Fat: 21g, Calories: 234 - Recipe idea: Prepare a delicious avocado toast by mashing avocado onto whole-grain bread and topping it with a poached egg and a sprinkle of salt and pepper.
8	Remember to consult a nutritionist or healthcare provider before making any significant dietary changes or for personalized advice based on your specific needs.

Showing 1 to 8 of 8 entries

Conclusion

I enjoyed doing this project. I was able to use a lot of different subjects I learned throughout my master's program. At first, I decided to make a UI where the user can interact with the openintro package and filter through their food options. I quickly realized that only doing that would make it impossible to meet the word count for the report. This helped me be more creative with the application. I didn't think about adding ChatGPT until I was about 1,000 words short of the requirement. As difficult as writing this paper was, it pushed me to create something that I am proud of. This project is stored on Github [12], so anyone can use what I made and add to it. Seeing someone in class use the live app inspired me to host my application. It's not hosted by shiny.io, where anyone can access it[13]. The only downside is that my package isn't hosted by CRAN(Comprehensive R Archive Network), so I couldn't include it in my code. Instead, I loaded the functions using Source() for the corresponding R file. Eventually, I would like to fix this issue and also have my data visualization in the measurements tab so the user can get a better comparison for their BMR.

Works Cited

[1]*ShinyUiEditor*. Shinyuieditor. (n.d.). https://rstudio.github.io/shinyuieditor/

[2]Department of Health & Human Services. (2001, August 1). *Metabolism*. Better Health Channel. https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/metabolism

[3]MediLexicon International. (n.d.). *Basal metabolic rate: What it is, calculation, and more*. Medical News Today. https://www.medicalnewstoday.com/articles/basal-metabolic-rate#:~:text=Why%20is%20BMR%20important%3F,to%20cut%20out%20each%20day.

[4]Abreu, M. (2023, May 16). *Mifflin-St. Jeor for Nutrition Professionals*. Nutrium Blog. https://nutrium.com/blog/mifflin-st-jeor-for-nutrition-professionals/#:~:text=Of%20the%20four%20predictive%20equations,to%20be%20the%20most%20reliable

[5]Forbes Magazine. (2023, November 27). *TDEE calculator: Calculate Your Total Daily Energy Expenditure*. Forbes. https://www.forbes.com/health/body/tdee-calculator/

[6]Ersever, Y. (2018, July 3). *500 person gender-height-weight-body mass index*. Kaggle. https://www.kaggle.com/datasets/yersever/500-person-gender-height-weight-bodymassindex/data

[7]Andy Frisella. (n.d.). *What is 75 hard?* https://andyfrisella.com/blogs/articles/what-is-75-hard

[8] One, P. (2023, November 9). *Can you crush weight loss goals on the 75 hard?*. Medium. https://salwriter.medium.com/can-you-crush-weight-loss-goals-on-the-75-hard-741b0545e57a#:~:text=Potential%20Weight%20Loss%3A%20While%20it's,pounds%20during%20the%2075%20days.

[9]*Nutrition in fast food*. Data Sets. (n.d.). https://www.openintro.org/data/index.php?data=fastfood

[10]Saxena, S. (2021, July 29). *Food nutrition dataset*. Kaggle. https://www.kaggle.com/datasets/shrutisaxena/food-nutrition-dataset

[11]Bhalla, D. (n.d.). *Chatgpt in R: Everything you need to know*. ListenData. https://www.listendata.com/2023/05/chatgpt-in-r.html

[12] colorado, valentina. (n.d.). *Valcolorado/MyFitnessFriend: This package integrates functions for fundamental health measurements and extends its capabilities to nutrition calculations*. GitHub. https://github.com/ValColorado/MyFitnessFriend

[13]colorado, valentina. (n.d.-a). MyFitnessFriend. https://valentina-colorado-myfitnessfriend.shinyapps.io/myFitnessFriend/