**Massachusetts Estimated Medical Cost Finder**

October 13, 2022

**Team Members:** This team consists of Melanie Grudinschi, Haley Wiskoki and Shaylan Bera.

**Project Track:** The chosen project is the product track.

**1. Problem / Purpose Statement:**

Often, individuals get blindsided by their medical bills and are unaware of the general cost of medical services from their provider [1]. Not only can this cause additional stress and suffering for the patient, but it can also hinder families from adequately saving up for medical emergencies [2]. Additionally, it has been shown that “medical bills and debt“ as well as “stressors including cost of health insurance … may also trigger mental health symptoms” [2]. Therefore, the purpose of the application is to improve patient satisfaction by providing them with an estimated cost analysis based on their particular anticipated visit reason utilizing various demographic data.

**2. Objectives:**

For the proposed application, the overall objective is to provide patients with an estimated expense analysis of medical services that are specific to them. In other words, patients will have the ability to select their location, gender, visit type, insurance, etc. and the application will output an average base cost and out-of-pocket (OOP) cost after insurance, based on previous patient data. This information will also be displayed for user reference in two bar graphs: one designated for displaying costs (base & OOP) as a function of a user input, and another designated to displaying costs as a function of insurance type. The first plot allows the user to explore how the cost of various healthcare encounter types and visits change as a function of location, age, etc, while the second plots informs the user of how their current insurance stacks up against other insurance types, or no insurance at all.

For example, a female patient living in a particular county will have the ability to look up the average OOP cost of an urgent care visit near her residence by examining the data of other similar individuals (race, age, gender) living in/had visited the same county for an urgent care visit. The aim is to create a semi-personalized cost estimator using the available synthetic dataset. In theory, the application would allow users to get a rough idea of their medical expenses before they even approach a medical professional.

In terms of features, a patient will be able to filter the cost analysis data by inputting their demographics, encounter class (i.e. urgent care, ER, etc.), insurance type, and anticipated visit type (prenatal visit, general examination, etc.). Additionally, the user will be able to choose a series (encounter classes, age ranges) for which the first graph will be generated. The values in this plot would be calculated based off of the input filters, and graphed as a function of the user-defined series. Based on all user inputs, the application will output a calculated average base cost (without insurance), and average OOP cost (with insurance) as personalized by the user. The user inputs for the application will be limited to what is available in the synthetic dataset.

**3. Data Source:** The provided .csv synthetic patient datasets in D2L will be used for the application.

**4. Function Requirements:**

1.1 Select Inputs: The application shall allow a user to select/edit multiple input parameters.

1.2 Select Compare By: The application shall allow a user to select/edit the cost comparison series.

1.3 Generate Base: The application shall generate an estimated base cost (without insurance) based on the specified parameters.

1.4 Display Base: The application shall display the estimated base cost to the user.

1.5 Generate OOP: The application shall generate an estimated OOP cost (after insurance) based on the specified parameters.

1.6 Display OOP: The application shall display the estimated OOP cost (after insurance) to the user.

1.7 Generate Comparison Graph: The application shall generate a graph of estimated costs (base & OOP) as a function of the user-specified comparison series.

1.8 Display Comparison Graph: The application shall display the graph of estimated costs (base & OOP) as a function of the user-specified comparison series.

1.9 Generate Insurance Graph: The application shall generate a graph of estimated OOP cost as a function of insurance type.

1.10 Display Insurance Graph: The application shall display a graph of estimated OOP cost as a function of insurance type.

1.11 Update Outputs: The application shall automatically update all outputs (graphs and output cost estimations) any time an input is altered.

**5. Design:**

This product will be generated utilizing R Studio and the R Shiny toolbox. R Shiny allows for users to create their own interactive web applications directly in R. These apps can be embedded in an R notebook itself or simulated in a webpage. For the purpose of this project, the interactive app will be published in a website format on Github.

***5.1.*  *Input Variables:***

The input filter choices were selected for this application by finding all unique values that exist for those variables (for example, there are only 5 races in the synthetic dataset, only two races, etc.). Ages were grouped generally by ranges of 10 years, and separate ranges for those under 20 and over 70. These input filters will be used to group sub-classes of all patients in the dataset via inputs specified by the user (e.g., a dataframe of all patients ages 30-39, female, Asian, in Hampden, who received an ambulatory encounter, for a prenatal visit, with Aetna insurance). The sub-class dataframe, generated from the input filters, will be used to calculate the final estimated cost. Based on the user’s input insurance, the estimated OOP cost will also be calculated.

The X-Axis Series Selection / “Compare Results By” filter is used primarily for Graph 1, to specify the series for which cost is displayed (e.g., base & OOP costs graphed as a function of age range or race).

| **Input Filters for Y-Axis Values** | | | | | | | **X-Axis Series**  **Selection** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Location  (County) | Gender | Age | Race | Encounter Class | Visit Type | Insurance | Comparison |
| Hampden | Male | <20 | Asian | Emergency | Encounter for Symptom | Dual Eligible | Location |
| Suffolk | Female | 20-29 | Black | Ambulatory | General examination | Medicare | Gender |
| Franklin |  | 30-39 | Native | Wellness | Well child visit | Medicaid | Age |
| Bristol |  | 40-49 | White | Inpatient | Prenatal visit | Humana | Race |
| Norfolk |  | 50-59 | Other | Outpatient | Telemedicine Consultation | Aetna | Encounter Class |
| Dukes |  | **…** |  | Urgent Care | Follow-up visit | None | Visit Type |
| **…** |  | >70 |  |  | **…** | **…** |  |

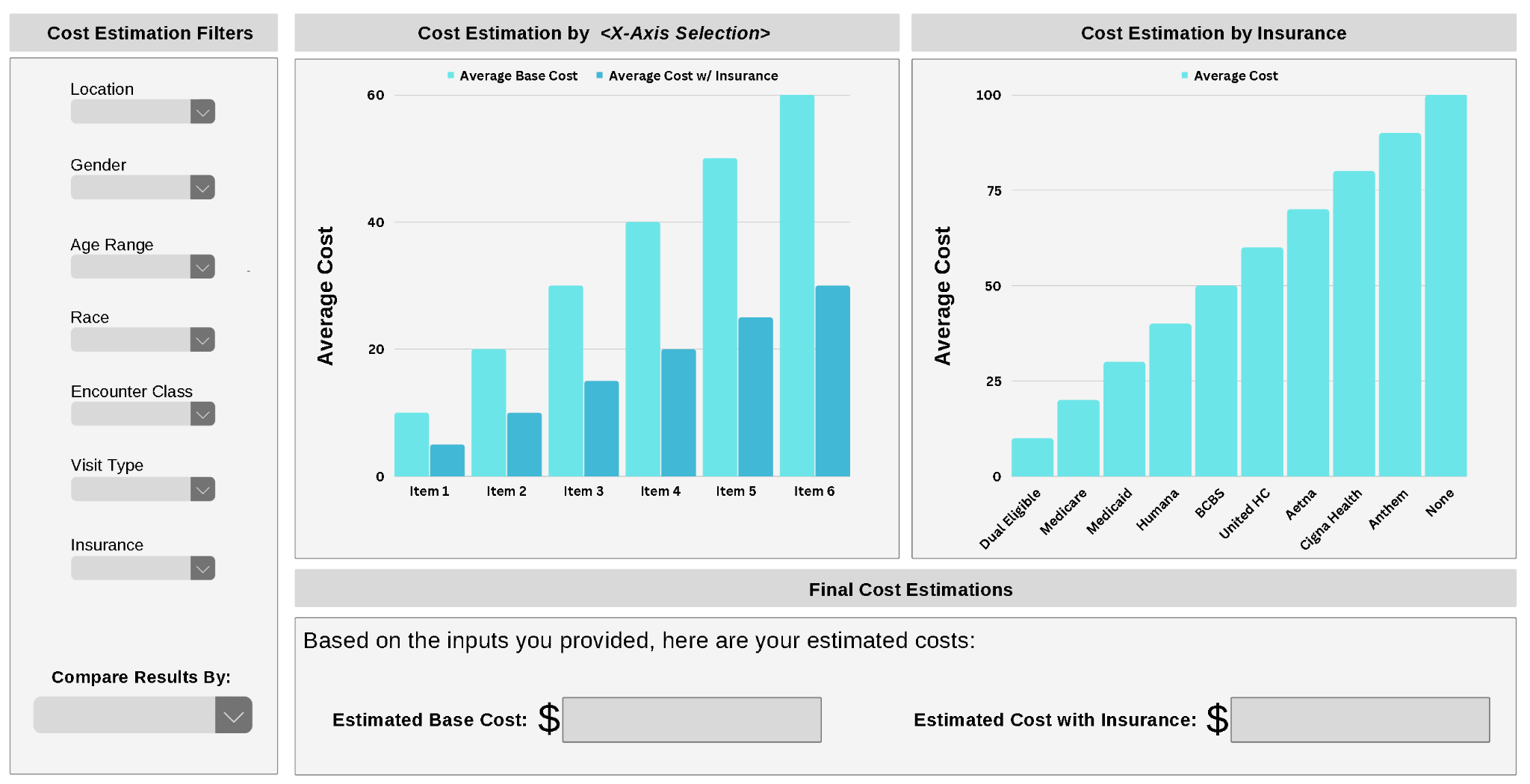
***5.2.* *Analyses:***

Using the synthetic datasets, the application will be able to match each unique patient ID to each patient’s visit, procedure, insurance, etc., similar to the methodology employed in lab 2. Utilizing libraries such as dplyr, the application will compile the datasets into one dataframe, or if needed, multiple relational databases. On the back end, a compiled table would link multiple linked patients (via age, race, gender, location) to the expenses of interest. For the age filter, another calculation will be performed to obtain the age of the patient using the birthdate and a standard chosen "date of visit". Thus, it would then pull from the table and filter the data to obtain an estimated cost (base and OOP cost) that is specific to the user inputs. The cost will be calculated by taking the average costs to patients in the dataset that fit the input criteria. The average cost calculation will be based on the raw base cost values pulled from a multi-filtered dataset. Additionally, the estimated OOP cost with insurance will be calculated by performing the average base cost minus average payer cost to determine how much the patient would pay out of pocket.

***5.3. Output:***

The output of the application will consist of four total components. The first will be a quantitative measure that displays the average estimated base cost of the anticipated visit reason based on the parameters set by the user. The application will also output an estimated patient cost after insurance coverage. The values will be rounded to the nearest whole dollar. The second piece of the output will be a graph displaying how their estimated cost compares to other variables such as patient demographic or encounter class. For example, a patient could request the cost analysis of a prenatal visit at a hospital in Nantucket County. The graph, for example, may show the estimated cost as a function of the other 14 counties. Other graphs with different input parameters may compare estimated costs based on gender, age or race. Users will have the capability to select their comparison metric. A second graph will display the estimated OOP cost as a function of different insurance types. The graphs will be continuously updated as the input variables are changed by the user. The combination of two descriptive statistics and two graphical visualizations will help the patient fully understand their estimated costs and how it compares to various other patients’ costs.

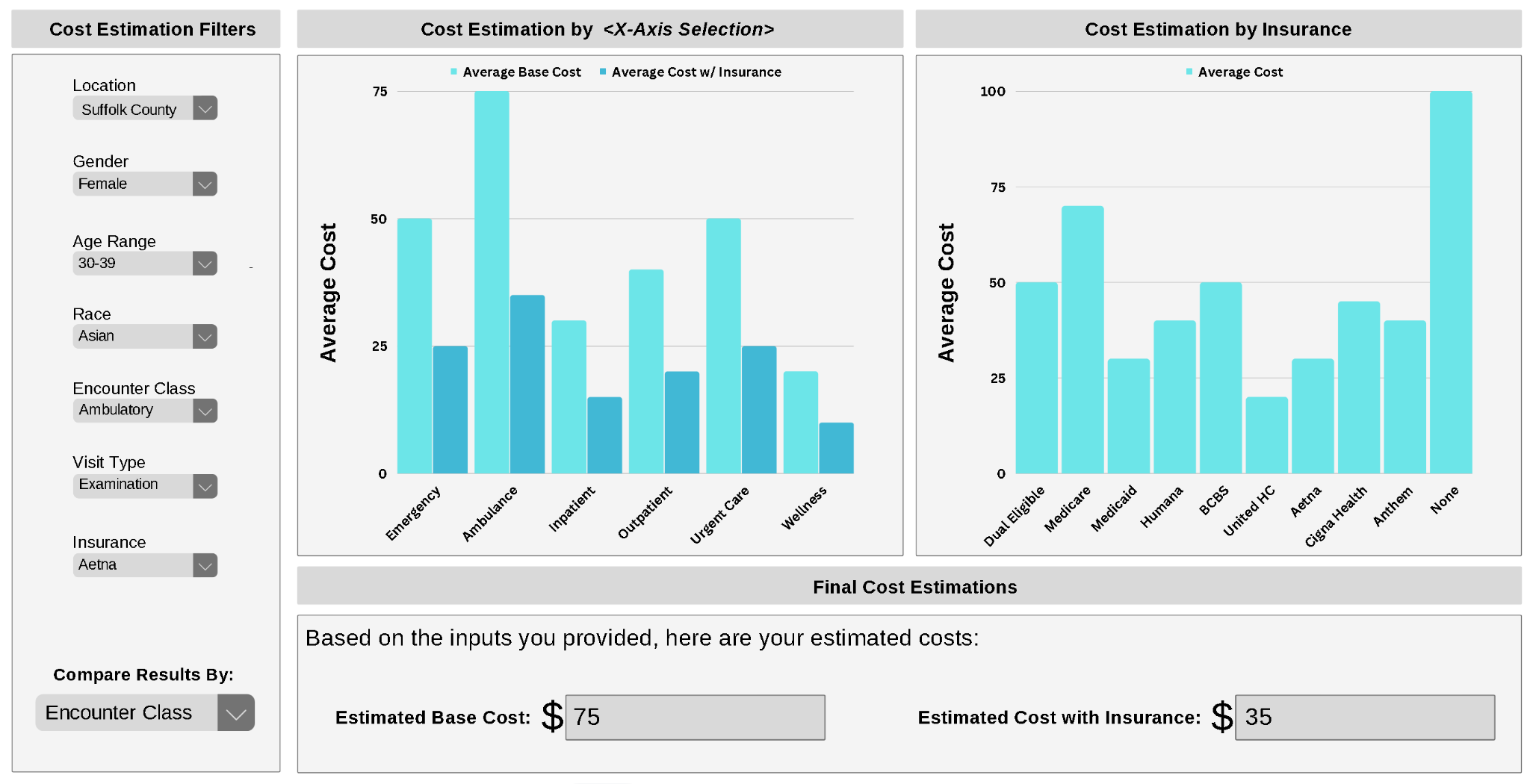
***5.4. Mockup:***

****

*Figure 1. This is the “base” model mockup, demonstrating various components of the product. See following figures for labeled diagram and sample use case.*



*Figure 2. Labeled diagram of application functionalities & widgets. Inputs are defined in more detail above in the “Inputs” section of “Design”.*



*Figure 3. Sample use case: Cost estimation filters are input by the user. The x-axis, or comparison filter, is also input by the user. As a result, the plot generates average cost values (base and w/ insurance) which are calculated using a dataframe including only those patients within Suffolk County, that are Female, aged 30-39, Asian, and Non-Hispanic, who had an Examination for Symptom visit type for a Bone Fracture, with Aetna insurance. These values are calculated for each Encounter Class (as chosen by the user) and displayed on the graph above. Given the user’s input desired Encounter Class (Ambulatory), a final cost estimation is output (both base cost and cost with insurance).*

**6. Roles / Responsibilities:**

| Name | Class Status | Project Role | Responsibilities |
| --- | --- | --- | --- |
| Melanie Grudinschi | Graduate | Web Developer Engineer | Visualization / Front-End |
| Haley Wiskoski | Graduate | Data Analyst Engineer | Analysis / Back-End |
| Shaylan Bera | Undergraduate | Data Engineer | Data Pre-Processing |

***6.1. Responsibilities Breakdown:***

1. Data preprocessing / creating dataframe - Shaylan
   * Gathering appropriate datasets
   * Data cleaning / filtering
   * Generate analysis dataframe
2. R Code
   * Input widgets - Melanie
   * Back end analysis (calculating the averages and filtering based off inputs) - Haley
   * Output cost estimation - Haley
   * Output graphs (2) - Melanie and Shaylan
   * Assist with UX design of application - Shaylan
3. Verification
   * Verification testing day - Team
4. Publication to website - Haley
5. Documentation
   * Code Documentation - Team
   * ReadMe, Github - Melanie

**7. Anticipated Milestones:**

| Milestone | Due Date | Plan | Status |
| --- | --- | --- | --- |
| Project Proposal Final Draft | 10/10 | Meet on Fri Oct 7th, 1 pm to draft the proposal. | Completed |
| Data Cleaning Process | 10/21 | Compile and filter datasets into useful dataframes. | Completed |
| Back-End Analysis Process | 11/4 | Calculate costs based on patient specific inputs. | Completed |
| Modeling / Display | 11/11 | Work on generating output graphs and webpage. | Completed |
| Verification | 11/17 | Verify dataframes, inputs, and outputs match. |  |
| First Project Draft | 11/20 | Create initial working webpage, design unfinalized. | Completed |
| Final Project Submission | 12/7 | Finalized design and webpage features. |  |

\*Continue to schedule weekly meetings as necessary.

**8.0 Citations:**

1. Pennic, Jasmine. “Medical Bills Continue to Plague Consumers as Source of Anxiety, Mystery and Frustration.” *HIT Consultant.* 10 Dec. 2021. https://hitconsultant.net/2021/12/10/medical-bills-consumers-anxiety-frustration-study/. Accessed 13 Oct. 2022.
2. Wiltshire, Jacqueline C, et al. “Problems paying medical bills and mental health symptoms post-Affordable Care Act.” *AIMS Public Health,* vol. 7,2 274-286. 6 May. 2020, doi:10.3934/publichealth.2020023

**Product Design Review**

November 7, 2022

**1. Progress Summary**

Since the proposal was submitted, our team has made large progress towards development of our product application. We started by importing the raw synthetic data and combining it into one data frame based on association of each patient with the corresponding encounter class, visit type, cost for visit and insurance type. Next, we cleaned the data frame by removing unnecessary columns and renaming data elements that were not user friendly for the website interface. Then, we created a function for calculating the age of the patient, accounting for leap years, based on a set date, their birthday, and death date if applicable. We defined age classes to be used in the age range drop-down menu. We created vectors with the appropriate drop-down choices for each category of the input widgets. Next, we created a dynamic drop-down menu that updates the Visit Reason choices based on the selected Encounter Class. Next, we became familiar with R Shiny and the UI interface by creating a title, sidebar and main panel for our website. We created the input widgets that provide the correct drop-down menus for each category with instructions above. Then, we created sample plots and text for the estimated cost metrics using the initialized application layout created. We have begun filtering the data frame by input choices to calculate the output cost and analyses for the comparison graphs. Overall, data ingestion, sorting, and cleaning methodology is roughly complete. Several small data processing functions were developed (age calculations, payer ID to names, mining unique variables from a set, etc.). The UI has been drafted with all necessary widgets, functionalities, inputs, and outputs. UI is in its intermediate stages of creation with widgets in their general layout position and possessing full functionality. Moving forward our plan is to complete the development of the graphs and cost outputs, improve the user experience, and perform verification and final documentation. The graduate team members will also complete the extra research objective of comparing to other available cost-estimation options.

**2. Effort Statement**

Our team has been meeting weekly on Zoom for about 1-1.5 hours. We met twice leading up to the proposal, and since then, we have met an additional 4 times. On average, the graduate team members spend roughly 5 hours on the project per week while the undergraduate member spends roughly 4 hours per week. During our weekly meetings, we first go over individual updates and identify major changes we have made to our project code. Then, we identify critical path problems and work together to find a solution. Finally, the team defines individual tasks/responsibilities that should be accomplished before the next weekly meeting. We have found this method to be successful and we plan to continue with this strategy for the rest of the semester.

**3. Challenges**

One of the main challenges we were facing was in regard to interoperability and version control. There were a few occasions where the same code was run on two different user computers and produced different results. We believe this may have been due to different versions of R; this was solved by using a different function that worked on all computers and updating packages. We found it inconvenient to track various versions of the project code (not typically an issue in other platforms such as Google Colab). We solved this by communicating to the group when the code would be “checked out” for development, and when importing back to the drive, updating the title to \*\_V# (incrementing the version number each time the code was updated).

Other more technical challenges we faced were in regard to the “server” portion of the RShiny application. We needed to find out how to “grab” a user’s selection from the front-end, and utilize it as a variable in the back-end for real-time computations and display, for example, how the “Visit Types” drop-down list would change dynamically as dependent on a user’s input for “Encounter Type” - discovered render() and reactive() functions that facilitate this functionality. Obstacles also came up when trying to create the general layout of the UI (how to manipulate size and position of widgets). To tackle this, we split the layout into two portions, the plots and the drop-down menus, and divided tasks among two members. Members reviewed tutorials and reported back on the progress/process during meetings.

Another challenge (still in development) is how to handle a user selection of “All” for any filter input… “All” does not exist in the dataset, so this would need to be translated to mean *no filtering* for that variable.

**Product Design Review II**

November 29, 2022

**1. Changes / Deviations**

The changes from the initial proposal and first design review are as follows. First, we are no longer including all of the csv files as originally stated. Only the payers, patients, and encounters csv files are now used. Once the team started the product implementation, we realized that these were the only files needed to meet our intended functional goals for the UI. Another change is that as opposed to all Visit Types being available in the filter (~50), we enhanced functionality, plotting aesthetic, and ease-of-use by creating a reactive input filter that will present a dynamic list of Visit Types based directly on the user’s Encounter Class input. This is because there are specific Visit Types that pertain only to certain Encounter Classes (e.g., “Urgent Care Clinic” Visit Type only used in “Urgent Care” Encounter Class). Lastly, the “All” selection wasn’t intended to be included in the initial project proposal, though, was deemed necessary for the case that a user would like to view stats for “All” age classes, for example, or another filter. The team implemented the functionality of a user selecting “All” or multiple inputs into the plots and cost estimations.

**2. Progress Summary**

Shaylan’s individual contributions include the data pre-processing steps and user interface optimization. He imported three main csv files from the synthetic dataset provided in D2L including the patients, payers, and encounters datasets. He then combined all three datasets using the *dplyr* package. Patients and encounters were combined by the common column of patient ID. The resulting patient+encounter data frame was then joined with the payers dataset via the PAYER ID column; this ultimately included all patient info, individual patient encounters, and payer information all in a single dataframe. Shaylan added a section of code to remove unnecessary columns from the resulting data frame. For example, Address, SSN, Passport, Lat, Lon, Last Name, Marital, Zip, Provider, Birthplace were all removed as they were not needed in subsequent analyses. He added additional columns such as Payer\_Name, After Insurance Cost, Age, and Age range classifications. Shaylan updated the string values within the dataframe to be capitalized and fully spelled out for aesthetic purposes in final UI drop down menus and also for ease of use in creating the plots and calculating the estimated cost values. An example is converting “M” to “Male” and “F” to “Female”. The visit description and payer names were also updated to be fully spelled out and capitalized. As for the optimization of the user interface, Shaylan edited the plot presentation by adding color themes and fixed spaces in the cost estimation statements of the UI layout. A lesson learned was the difference in functionality between the “paste” and “paste0” function in R. The “paste0” function removes the extra space between a string and variable which looked more pleasing in the cost estimation output sentences.

Melanie’s individual contributions are as follows. She performed a proof-of-concept text output of the base and after insurance cost estimation statements. This included creating a dataframe that filters by all the user’s inputs, calculating the mean of the base cost column, calculating the mean of the payer cost column and subtracting the payer cost from the base cost to obtain the estimated cost after insurance. Later, after Haley created a function to handle when a user selects “All”, Melanie integrated this code into the cost calculations. Melanie also added if/else logic with a general statement to display when the data frame is empty due to input filters which are not represented in the dataset. The estimated cost outputs are now fully functional, accurately represent that data from the data frame, and dynamically change when the user changes any input. Melanie worked with Haley to update the function which creates a series for the x-axis of the comparison plot.

Haley’s individual contributions include the following. She was able to successfully handle the “Compare By” and “All” filter inputs (more thoroughly explained in the Challenges section as this was a large obstacle we encountered). Through this, she was able to generate a data frame for specific use in calculating the y-series values for plotting. This dataframe was implemented in the generation of an additional data frame that is grouped by all unique elements of the “Compare By” selection, with columns including the mean base cost and mean cost after insurance. The values in this data frame are ultimately plotted in the final graph as two separate bar column series. She also manipulated the input widgets and plots so that upon opening the UI, all inputs are set to “NULL” by default. This eliminated the display of errors upon start, and allowed for the display of a blank plot until the user inputs all of the required parameters.

Overall, the main progress since the first design review included successfully preprocessing the raw data further, generating all filter widgets, creating a functional bar chart, outputting accurate estimation widgets, and initial UI optimization.

**3. Effort Statement**

As stated in the previous design review, our team has been meeting on Zoom every week for about 1-1.5 hours. Since the last design review, the team as a whole has met 4 times, while the two graduate students, Melanie and Haley, have met an additional 2 times for troubleshooting, pair programming, and discussion of the additional research objective. In the last few weeks, more time has been spent on the project. On average, the graduate students dedicate roughly 6 hours per week while the undergraduate member spends roughly 4.5 hours a week on the project. During our weekly meetings, we have continued to follow the same structure by discussing what each team member has accomplished since the last meeting, working through critical issues, and identifying tasks for the next meeting. In the last few weeks however, we have begun to plan out the rest of the semester in better detail since we are nearing the end of the project, such as identifying when we would begin working on the project write-up and video. Overall, the team has stayed relatively on course with the project timeline outlined in the project proposal, except for output graphs due to unforeseen challenges, which has taken considerable time to resolve. As mentioned in the previous design review, we have found this method to be successful and we plan to continue with this strategy for our last remaining meetings.

The individual contributions are stated in the Progress Summary section above, but a quick summary is as follows. Shaylan has worked on the data-preprocessing steps by integrating the desired csv files, removing unused columns, adding useful columns, and fixing up text within the combined dataframe. Shaylan has also helped with general aesthetics of the UI. Melanie has worked extensively on the UI input widgets, the base cost and after insurance cost calculations, and has also assisted Haley with updating the functions to create an x-axis series for the comparison plot. Haley has put a great deal of effort into handling the “Compare by” and “filter” inputs to generate an appropriate data frame for plotting. Haley also tackled the logic regarding the “All” input filter. Finally Haley has also worked extensively in getting the output plots to display, a recent challenge the team has been facing. Both Melanie and Haley have worked extensively on the UI in general.

**4. Challenges**

As we near completion of the project, we have faced several challenges in implementing specific features and functions of our UI. One of the main obstacles we faced was handling a user input of “All” for a specific filter, and how to handle the “Compare By” selection when generating a dataframe for calculations and plotting. Any filter with an input of “All” would require *no* filtering of the dataframe on the backend, similarly for the “Compare By” input, whatever series is selected for comparison on the graph would not need to be filtered from the dataframe. There are many combinations of possible filters that could be performed, and it is necessary to be able to handle them all, as such, we implemented a series of if/else statements to evaluate whether each filter is either set to “All” or is selected as the series for “Compare By”.

Additionally, we realized that this dataframe would work specifically for generating x and y values for graphing, though it wouldn’t work for final cost estimations. A second data frame is needed for cost estimations, as these two final output values are computed based on *all* of the user’s inputs and on all filters, without any regard for the “Compare By” selection. The same methodology was implemented in the generation of this dataframe, checking whether or not a filter is set to “All”, but without the inclusion of a “Compare By” conditional.

Another major obstacle was dynamically handling various selections of “Compare By”. In the backend, a user’s “Compare By” input is translated from a series title, to a list of elements of that selected series (e.g., if “Compare By” is Age Class, this list includes all of the unique age classes found in the filtered data frame). This list of elements changes directly with the “Compare By” selection. Additionally, R’s group\_by function only takes in an explicit column title from the data frame; we needed to implement another series of if/else statements to specify the dataframe column to be grouped based on the “Compare By” selection in order to calculate mean base cost (y-values) as a function of the unique “Compare By” elements (x-series).

Apart from these specific challenges with implementation, there were other difficulties specific to the R programming language and its operations. We struggled to understand the difference between the several reactive functions available in R Shiny such as eventReactive, observeEvent, observe, reactive, and how they operate differently. We eventually understood how these functions were different and how their variables could be handled within the UI (e.g., reactive values). Debugging is also very difficult to perform within the UI server function, as it doesn’t have the capability to simply print or view variables. This was eventually overcome, though is not easily done in an intuitive manner as it would be in other programming platforms like MATLAB or Python.

**5. Remaining Tasks**

Below are the tasks the team plans to accomplish from now until the end of the semester:

| **1. Product Design/Implementation** | **2. Product Verification** | **3. Documentation** |
| --- | --- | --- |
| Aesthetics (color, font, sizing) | Verification Testing | Publication to Website |
| Second, Insurance-Only Plot | Requirement Verification | Documentation - ReadMe, Github |
| Rotate Barplot X Labels |  | Final Design Report |
| Layout Compare By and Insurance Plot Side-by-Side |  | Recorded Presentation of Product |
| Initialize Plot Output at Startup |  |  |
| Handling “NULL”/empty data frame error message for plots |  |  |

We plan to accomplish these tasks in our next two meetings. By December 5th, we hope to have finished the RShiny application in its entirety and begin our product verifications. We will also assign parts for the final report. In the following meeting (around December 12th), we hope to finalize the report and record the video presentation.