**Massachusetts Estimated Medical Cost Finder (Product Track)**

**Github Link:** <https://github.com/haley-wiskoski/SIE-BME-577-FinalProject>

**Shinyapps Link (~2.5 min load time):** [https://shaylanbera.shinyapps.io/BME477GroupProject/](https://shaylanbera.shinyapps.io/BME477GroupProject/?_ga=2.155482939.1052087240.1670263641-1747209555.1670263641)

December 14, 2022

**Team Members:** This team consists of Melanie Grudinschi (577), Haley Wiskoski (577) and Shaylan Bera (477).

**1. Product Need & Purpose**

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]. Moreover, 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]. As a result, the purpose of the following 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. By semi-personalizing the cost output, patients would not only get a real-time estimate of how much they may expect to pay, but would also be able to compare prices across insurance types. While the current application only utilizes synthetic data, the purpose and scope of the cost estimator website can be expanded to include real patient data, significantly improving the application’s practicality.

**2. Product Design**

This application consists of eight distinct inputs. Seven of the inputs (Location, Gender, Age Range, Race, Encounter Class, Visit Reason, and Insurance) are for the user to specify which demographics and/or encounters they would like to obtain estimated costs for. The last input, “Compare Results By”, allows the user to select a series that would be plotted as a function of cost (e.g., cost as a function of age class, or as a function of gender). These inputs are used to populate three outputs. The first output is a “Filtered Cost Estimation” graph; as mentioned, this graph displays estimated costs as a function of the user’s selected “Compare By” series. This graph displays two distinct series of values: average Base Cost and Cost After Insurance, both calculated as a function of the “Compare By” series elements. The second output is a “Cost Estimation by Insurance Type” graph, which plots the average Cost After Insurance as a function of insurance provider. Lastly, the third output provides two, final estimated costs for the user: 1) a final average Base Cost and 2) average Cost After Insurance, which are calculated directly from the user’s filter selections (first 7 inputs). The two estimated cost outputs are rounded up to the next whole dollar. These three outputs (two plots and final cost estimations) are updated and re-displayed in real-time as the user selects any new inputs or “Compare By” series. Calculations performed to obtain the values displayed in these outputs are described in further detail in the following section. For succinctness, the inputs and outputs of the application are defined below (see attached Google Sheet for list of elements for each filter type):

Inputs: *(*[*Inputs*](https://docs.google.com/spreadsheets/u/0/d/1QgbFbwDR9jcnyN2UhiKcMF9M6ooawiXV0kW_mdAdNms/edit) *Google Sheet)*

1. Demographic / encounter filters
2. Compare By filter

Outputs:

1. Filtered Cost Estimations plot
2. Cost Estimations by Insurance Type plot
3. Final Output Cost Estimations (2)
   1. Estimated Base Cost
   2. Estimated Cost After Insurance

**3. Implementation**

The pre-processing pipeline that was performed is outlined in the table below for simplicity. Pre-processing steps included Ingestion, Integration, Reduction, Transformation, and Enrichment, to prepare the data for subsequent processing and final analysis.

| **Pre-Processing Step** | **Description** |
| --- | --- |
| **Ingestion** | Read in “patients.csv”, “payers.csv”, “encounters.csv” |
| **Integration** | Combine/join the three data frames based on matched Patient ID and Payer ID |
| **Reduction** | Removal of unused columns (i.e., SSN, Address, Lat, Lon, Zip, Provider, First and Last Name, Marital Status, Birthplace, Member Months, City, State, Passport, Ethnicity) |
| **Transformation** | Renaming of columns & elements: Payer ID > Payer Title, “M” > “Male”, “F” > “Female”; “urgentcare” > “Urgent Care”, capitalization of Encounter Classes and Visit Types for aesthetic purposes |
| **Enrichment** | Added columns: Age Class, After Insurance Cost |

***3.1. Processing & Analysis (Back-end)***

Upon completion of the pre-processing procedure, the data frame was ready for subsequent processing and analysis for output display. The first output graph required the generation of a distinct data frame via the application of user inputs for filtering. It was necessary to account for any filters selected as “All” (i.e., “All” genders, races, age classes, etc.), and the user’s selection for the “Compare By” input. Given a user’s selection of the “Compare By” input, back-end processing would need to interpret that the input selection would *not* need to be applied in the filtering process. For example, if a user were to select “Age Class” as the “Compare By” series, the Age Class filter selection would not need to be applied in the filtering process (e.g., all age classes would be included in the data frame, rather than one age class); this allows for the calculation of costs as a function of all discrete age classes. Similarly, if a filter was selected as “All”, all elements of that filter would be included in the data frame. As such, a conditional if/else code block was implemented to check whether any of the first 7 filters were selected as the “Compare By” series, *or* were selected as “All”. Upon indication of either of these conditionals, the data frame would not be filtered by those elements, but rather filtered by all other demographic / encounter filters not selected as “All” or “Compare By”.

To this end, a final data frame would be generated, filtered by all of the user’s selected demographic / encounter inputs, but with the exception of those that were selected as “All”, or any filter selected for “Compare By”. The final data frame would be sub-grouped by the “Compare By” series elements (e.g., each age class), and the mean Base Cost would be calculated as a function of each of the sub-groups (average of all individual base costs of each patient encounter in the sub-group). To calculate Cost After Insurance, the mean “Payer Cost” (how much the insurance provider pays) would be calculated, and subtracted from the average Base Cost (Base Cost - Payer Cost = Cost After Insurance). These two series (Base Cost and Cost After Insurance) are then plotted as a function of the “Compare By” elements in the “Filtered Cost Estimations Graph”, where the x-elements are those that remained in the data frame after applied filtering of all other demographic / encounter filter inputs.

The second output graph, “Cost Estimations By Insurance Type”, utilized the same methodology as described above, without consideration of the “Compare By” series. The initial, raw data frame would be filtered by all of the user’s inputs, accounting for those with an “All” selection. The data frame would then be sub-grouped by insurance provider. Average Cost After Insurance is calculated similarly as Base Cost minus Payer Cost, and plotted as a function of each insurance provider that remained in the data frame after previous filtering by demographic / encounter filter inputs.

The two final cost estimations output to the user, Estimated Base Cost and Estimated Cost After Insurance, are filtered and calculated in a similar way as those for the “Cost Estimations By Insurance Type” graph. No “Compare By” series handling is necessary, though processing does account for filters selected as “All”. Average Base Cost and average Cost After Insurance are calculated globally from a data frame that had been filtered by all user inputs of demographic / encounter. These two values, as mentioned, are rounded up to the next whole dollar for simplicity. These output values, along with the two graphs, would update in real-time as the user changes their input for any filter or “Compare By” selection.

***3.2. Display & Functionality (Front-end)***

The user interface (UI) design of the application contains many key elements that contribute to the user’s overall experience. The overall website UI design is broken down into three panels (title, sidebar and main). The title panel contains the product title, “Massachusetts Estimated Medical Cost Finder” which is intuitive and straightforward for users to understand.

The sidebar panel contains instructions for selecting inputs and eight input dropdown widgets. Each dropdown menu is generated to contain all unique values of each input filter as extracted from the dataset, with the addition of an “All” option. The “Visit Type” dropdown options are not generated until a user selects a specific “Encounter Class”; a “reactive” input filter is used to update the dynamic list of Visit Types based directly on the user’s Encounter Class input. This not only enhances functionality, plotting aesthetic, and ease-of-use, but also prevents user error by inhibiting the event that a user were to select a Visit Type that pertains only to a different Encounter Class (e.g., “Urgent Care Clinic” Visit Type only used in “Urgent Care” Encounter Class). As such, it prevents the user from selecting a Visit Type which does not match the Encounter Class. The drop down menus include the ability for a user to begin typing inside the drop down instead of scrolling through the options to select their desired choice. The menu will update and only display options which match what the user has typed. Lastly, the main panel contains the two plots and two estimated costs which are updated based on the users selections in the sidebar panel. Below these plots in the main panel are where the two final estimated cost outputs are displayed.

Other important functionality of the application includes the start up display, handling a scenario with no data, and handling a possible negative Cost After Insurance value. Upon opening the application by clicking the link, the inputs are set to null by default and the plots are empty. For the initial estimated costs (final Estimated Base Cost and Estimated Cost After Insurance), a message is displayed telling the user to update the inputs based on their desired choices to obtain those values. Once the user makes a selection for all eight inputs, the display is updated by populating the two output graphs and generating the final estimated cost values. The UI, including the plots and estimated cost outputs, are updated dynamically as the user changes any of the eight input parameters on the side panel. This dynamic update includes when the user changes the “Compare By” filter. A new comparison plot is populated with the correct x-series based on the new choice. As for handling a scenario with no data, if a user selects a certain combination of inputs which is not represented in the dataset, the graphs will return empty and a message on the estimated costs will be displayed, telling the user to update their inputs because no data is found. If a scenario occurs in the dataset where the insurance company pays more than the Base Cost (Cost After Insurance > Base Cost = -Cost After Insurance), the application displays a value of “0” both in the graphs and for the final Estimated Cost After Insurance value. In a real world scenario where the insurance provider pays more than the cost of the visit, the patient would not have to pay anything out of pocket, thus a Cost After Insurance of $0. This aligns with standard insurance practice since normally insurance companies do not pay more than the entire base cost of a certain visit reason. The user can not gain money off a medical visit.

**4. Product Testing & Results**

Details of the product testing and results can be found here: [*Product Testing Results*](https://docs.google.com/spreadsheets/u/0/d/19LF3KioWBVBbj1ct6zYMfWwdXo_heBE9bSUmjlcBmws/edit)

The Systems Verification Plan contains each requirement proposed at the start of the project, indicating how and when each of the requirements were verified. The methods of verification applicable to this project are test (T) and demonstration (D). There were 11 total requirements verified. Four requirements were verified through test and the remaining seven were verified through demonstration. The specifics of each verification can be found in the [Demonstration and Test Forms](https://docs.google.com/spreadsheets/u/1/d/19LF3KioWBVBbj1ct6zYMfWwdXo_heBE9bSUmjlcBmws/edit).

The demonstration requirements were verified by opening the link to the application on shinyapps.io and ensuring that the output met the requirement to be verified. For example, we demonstrated that the application allowed a user to select and edit multiple input parameters and displayed the estimated base cost to the user. This process was repeated for all seven requirements, to demonstrate the product functionality meets the requirements. The seven requirements to be verified by demonstration passed.

One test containing eight realistic scenarios was completed to verify the four test requirements. The eight scenarios were chosen randomly. In order to verify that the product outputs, including the values and graphs, were accurate, Haley created a separate R code to filter the dataset based on hand-typed selections. This separate code included the same data preprocessing from the main product code and an added section for verification testing. The code filtered the combined payers data frame from the original code based on manually typed parameters. It outputs a value for the estimated base cost and after insurance cost. It also outputs a dataframe containing the data for the comparison graph and a dataframe containing the data for the insurance graph. The code does not use R shiny or any UI. We ran the product application and the verification testing code on separate computers. The values for all eight scenarios of the expected and application outputs were recorded on the [*Product Testing Results*](https://docs.google.com/spreadsheets/u/0/d/19LF3KioWBVBbj1ct6zYMfWwdXo_heBE9bSUmjlcBmws/edit) datasheets.

The Base Cost and After Insurance Cost values output by the product application were compared to those given by the separate R code. The values and corresponding names for the comparison graph and insurance graph were also compared between the application and the testing code. Matching values indicated the product was displaying accurate values based on the dataset. After completing all 8 runs, we found that the product was outputting accurate values for the estimated costs and graphs. The four requirements to be verified by test passed.

In summary, all of the application requirements passed verification on December 7, 2022. The product can successfully complete all of the functions outlined in the proposal and produce results that align with the expectation.

**5. Future Work**

A current limitation of the product is that there are instances where different Visit Types mean the same thing but are grouped with different Encounter Classes. For example, “Encounter for Problem” can be found in Emergency and “Encounter for problem” can be found in Ambulatory. To expand on the current product using the same synthetic dataset in the future, functionality could be added to show whether the Ambulatory or Emergency Encounter type would be cheaper for the same visit type. These are not able to be compared in the current product as they are technically different between Encounter Types due to capitalization. Using Natural Language Processing techniques may help solve this problem.

Another major improvement to the product would be to obtain more synthetic data or obtain real data for the application. Currently, it is difficult to find a real dataset which matches the information needed for the product. Collecting data manually or pulling from EHR pre-existing data could be sufficient for the expansion of this product in the future.

Moreover, once the dataset is expanded with real or additional synthetic data, it would be ideal to include more states in future versions of the cost estimator application, further expanding on the state of Massacuchets to incorporate, for example, all Northeastern states. Functionality wise, the application can also be improved to include more detailed patient inputs, including multiple visit reasons, medication information, and insurance as a function of family size. These application upgrades would help give patients a more representative prediction of their medical costs.

Finally, the additional research objective comparing other similar applications to the one proposed here was completed by both graduate students for supplemental education. This research is included in the following document: [Additional Research](https://docs.google.com/document/d/1l1FAJEOo2G2nPoYXjE4n8NHppUIN9JBLZu87zB-V7M0/edit)

**6. 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