Final Report on Modeling Retirees’ Liquidity Needs

***Team 11**: Jessica Rizzo(jmr523), Yihang Hu(yh2255), Haotian Xiang(hx294), Mingtao Jia(mj624), and Haoyang Wang(hw774), Advised by Dr. David Ruppert*

Table of Contents

Introduction

This project is sponsored by the Lifecycle team of T. Rowe Price’s Multi-Asset Division who are responsible for strategic allocation of T. Rowe’s target date funds and the development of personalized retirement investment solutions. As a result, it is crucial that they understand the behavior and spending patterns of individual investors and retirees. This information is incorporated into simulation engines used to stress test and improve their products. Of course, modeling liquidity needs is a complex task. Our project focuses on a more concrete subtask: modeling out-of-pocket medical spending based on health status. Our main objective is a parsimonious and well-fitted statistical simulation of a subset of retirees’ spending that could be interpreted as “liquidity events.” For our purposes, a liquidity event in a large savings drawdown is caused by a health shock. Given the sparsity of data, a perfectly fit model is not feasible. Instead, we will create two intuitive and empirically informed “mock-ups.”

Data Description and Preprocessing

For this project, we use data from the Health and Retirement Study. The Health and Retirement Study is a survey of individuals over the age of 50 and their spouses (regardless of spouse age) which is collected by the Institute for Social Research at the University of Michigan. It is a longitudinal survey, meaning that individuals enter the surveys in cohorts and are re-surveyed in each successive wave until the HRS loses touch with them or the individual passes away. The HRS is an extensive survey of demographic data and information on health status, income, assets, and spending. Thus, it provides an archive of health and spending data. We use the 2020 Longitudinal File which is composed of 15 waves collected annually from 1992-1995 and biannually from 1996-2020. It contains only public information (i.e., no restricted data) and is published by the RAND Institute for the Study for Aging.

From 1992-1995, different cohorts were surveyed across different years and some key variables (namely out-of-pocket medical spending) are absent in the 1996 wave. For this reason, we begin with Wave 4 which was conducted in 1998. The raw file contains more than 40,000 observations of thousands of variables. Our goal is to construct a “snapshot” of health status, so we began with a thorough review of the documentation to identify those variables most relevant to our purposes. It is during this stage that we rearrange the data so that age is the time variable. We expect age to be a significant predictor of both health status and spending. The raw data is structured so that the rows correspond to unique individuals and the columns correspond to health variables in a given wave. Our final training dataset is structured so that the rows correspond to the value of an individual at a given age and the columns correspond to health variables. Table 1 below shows our final list of variables and summarizes any adjustments that were made to the raw data. We initially considered including demographic variables as well; however, initial data exploration and analysis demonstrated that demographic variables were highly uncorrelated with spending so we quickly narrowed our focus to only health variables.

Table 1: Summary of Variables

Final Variable Name	Formula Containing HRS Variables	Adjustments	Variable Type	Description	Time Period Covered
AGE	= 1998 - RABYEAR	N/A	Continuous	Age	At time of survey
WEIGHT	= RwWTRESP	N/A	Continuous	Weight of individual in the survey, not used for modeling	N/A
HEALTH_CHANGE	= RwSHLTC	N/A	Ordered categorical	Self-reported health change	Over last 2 years
HBP	= RwHIBPS	N/A	Indicator	Diagnosed with high blood pressure?	Over last 2 years
DIABETES	= RwDIABS	N/A	Indicator	Diagnosed with diabetes?	Over last 2 years
CANCER	= RwCANCRS	N/A	Indicator	Diagnosed with cancer?	Over last 2 years
LUNGS	= RwLUNGS	N/A	Indicator	Diagnosed with pulmonary condition?	Over last 2 years
HEART_ATTACK	= RwHEARTS	N/A	Indicator	Had a heart attack?	Over last 2 years
STROKE	= RwSTROKS	N/A	Indicator	Had a stroke?	Over last 2 years
PSYCH	= RwPSYCHS	N/A	Indicator	Diagnosed with a psychological condition?	Over last 2 years

ARTHRITIS	= RwARTHRS	N/A	Indicator	Diagnosed with arthritis?	Over last 2 years
OUT_PT	= RwOUTPT	N/A	Indicator	Used outpatient care?	Over last 2 years
DRUGS	= RwDRUGS	N/A	Indicator	Used prescription medication?	Over last 2 years
HOME_CARE	= RwHOMCAR	N/A	Indicator	Used home care?	Over last 2 years
SPECIAL_FAC	= RwSPCFAC	N/A	Indicator	Used special facility care?	Over last 2 years
HOSPITAL	= RwHSPNIT	N/A	Continuous	# of nights spent in the hospital	Over last 2 years
DOCTOR	= RwDOCTIM	N/A	Continuous	# of doctor visits	Over last 2 years
NURSING_HOME	= RwNRSNIT	N/A	Continuous	# of nights spent in the nursing home	Over last 2 years
MEDICARE	= RwGOVMR	N/A	Indicator	Covered by Medicare?	In last 2 years
SPEND_SS	= RwOOPMD/ AVGSSI	Adjusted to 2020 prices and divided by average social security income	Continuous	Adjusted out-of-pocket medical spending	Spending over last 2 years divided by a yearly average
WEALTH	RwDCBAL1 + RwDCBAL2 + RwDCBAL3 + SwDCBAL1 + SwDCBAL2 + SwDCBAL3 + HwAIRA + HwASTCK + HwACHCK + HwACD + HwABOND + HwAOTHR - HwADEBT	Adjusted to 2020 prices	Continuous	Liquid wealth	At time of survey

It is important to note that the prefix of HRS variable names is significant. If the first letter is a “R” the value is for the respondent only. Almost all “R” variables have an equivalent “S” variable, which is for the spouse (if any). “H” refers to the household, for the respondent and spouse together. For our purposes, we do not use any household or spouse variables, with the exception of the wealth variables which are only available at the household level. The next character refers to the wave. For example, the prefix “R7” means that this is a respondent variable collected in the 7th wave. The prefix “Rw” is used to refer to a variable generically, ie. “RwOOPMD” refers to out-of-pocket medical spending in any wave. More information about each of the HRS variables can be found in the HRS data documentation.

In most cases, the HRS variables have simply been renamed for greater interpretability. All monetary values were adjusted to 2020 prices using the priceR package to account for inflation. The T. Rowe simulation framework also adjusts spending to account for financial status, either with social security income or liquid wealth. We tested both measures and found that social security was superior to liquid wealth for several reasons. First, the variables required to construct liquid wealth have significantly more missing values than the social security income variables. Secondly, social security income tends to be fairly stable and immune to outside influences. On the other hand, liquid wealth varies independently of spending due to many factors: economic health, non-medical expenses, etc. We found that this extra variation greatly complicated the task of modeling and simulation. For these reasons, we chose to scale by social security income. To account for small variations in the self-reported values of social security income, we construct *average social security income* (the average of all non-zero values of RwISRET reported over an individual's time in the dataset). Under this framework, every spending value for each individual is scaled by the same constant value which prevents adding additional variation into our data.

The scaling (attempts) to correct for the impact of financial status in our dataset. An expense of $1000 may present a significant financial hardship for one individual and constitute a small fraction of monthly expenses for another. Of course, average social security income is not a perfect scaling tool. Liquid wealth would likely provide a more accurate barometer for financial position; however, its use is not feasible for the reasons discussed above. Liquid wealth is still used as a criterion for inclusion in the training dataset. We set a reasonably low threshold of $1000 to filter out individuals who are heavily in debt (and have large negative liquid wealth values). Significant debt is a confounding factor which is necessary to remove before we begin modeling. Another benefit of scaling is that it increases the interpretability of our data. A value greater than 1 indicates that the respondent spent more than their average social security income in the previous two years on out-of-pocket medical expenses.

With our preliminary selection of variables complete, we then began selecting which individuals and observations to include in our final training dataset. Ideally, our training dataset should be representative of T. Rowe's customer base and the entries should be relatively complete. The raw HRS is very sparse. The RAND dataset attempts to impute as many values as possible; however, missing values still present a significant factor. The dataset distinguishes many different types of missing values by the reason the data was not collected. We treat all missing values equally and replace the HRS missing codes with *NA*. More information about the HRS missing codes can be found in the HRS data documentation. For each individual we include their observations in the training dataset once (and if) they meet the following criteria:

- Age 65 or older
- Covered by Medicare
- Greater than $1000 in liquid wealth
- Less than or equal to 2 missing values (in a given wave)
- Observed for 5 or more waves (10 or more years)

We also remove individuals with missing values in any of the criteria columns or out-of-pocket medical spending. The filtering greatly reduces the issue of missing values (as individuals with one missing value are likely to have more). From the subset of the raw data containing our variables of interest, the average individual has 160 missing values, about half of the total number of variables (this is partially due to the fact that each observation contains variables from waves where the individual was not observed). After filtering, the average individual has .26 missing values and the average observation with missing values is missing 1.11 fields. The remaining missing values were all imputed to zero. This leaves us with 58463 observations of 7717 individuals where the average individual is observed for 5.1 waves or 10.2 years. Wealth and Medicare are now dropped as they are only used as inclusion criteria and not for modeling. Finally, we use a 70/30 split to construct the training and test sets.

With the construction of our training set complete, we began to investigate the distribution of each health predictor and its association with age and spending. We will look at the log of adjusted spending to minimize the impact of extreme outliers in spending.

Figure 1 below shows the average adjusted spending by age, separated by the values of each categorical predictor. We use the 10% trimmed mean to handle extreme outliers. However, this does not help at large values of age (where there may be only a handful of observations in each category) and the plots become jagged. For HEALTH_CHANGE, positive values indicate deterioration. Special facility, home care, lungs, heart attack, drugs, psych, and stroke appear to have the greatest difference in spending. For all variables, the values of spending become jagged at high values of age.

Figure 2 below plots the average adjusted spending and the numerical predictors. HOSPITAL and NURSING_HOME represent the number of nights in each respective place in the previous two years. Note that each contains a handful of values that are larger than 730 (the number of nights in two years). Consulting the HRS documentation, it appears that this can occur when the previous interview was conducted more than two years ago. In these cases, HOSPITAL and NURSING_HOME represent the number of nights in each respective place *since the previous interview took place*. However, these observations constitute a small portion of the dataset, so we will ignore this discrepancy.

Figure 1. Categorical Predictors

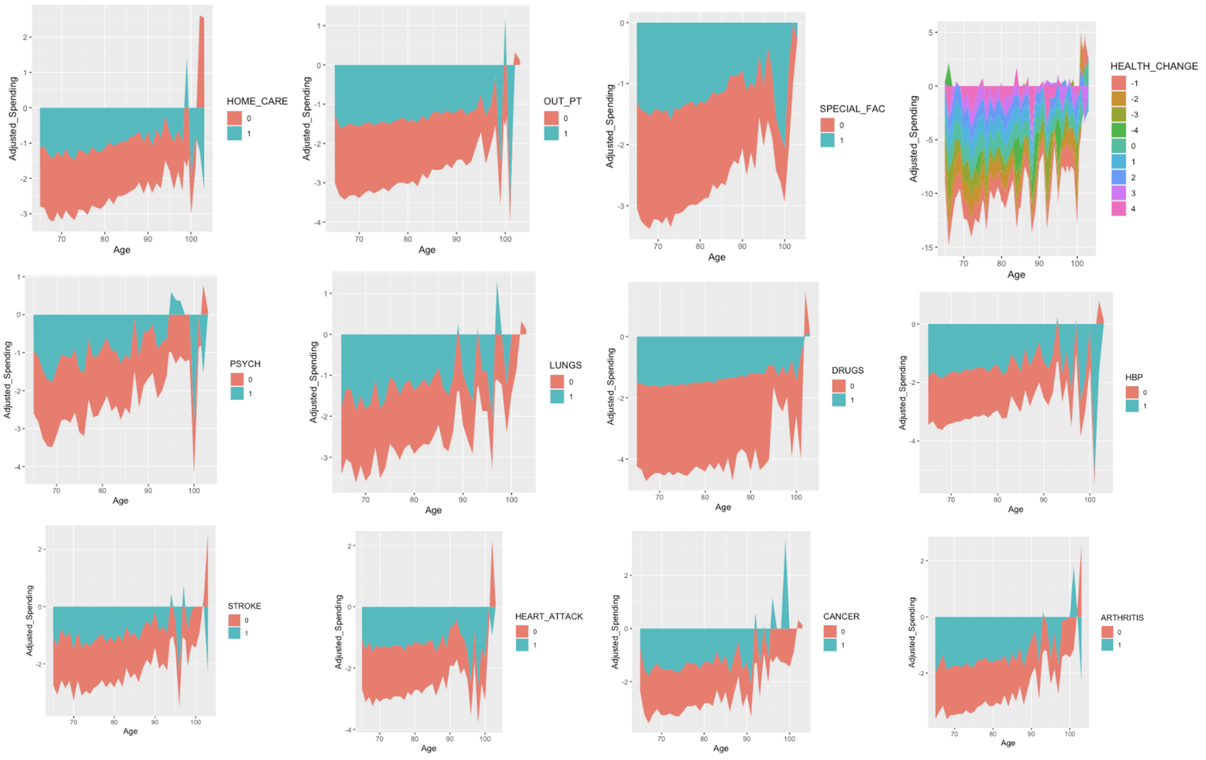
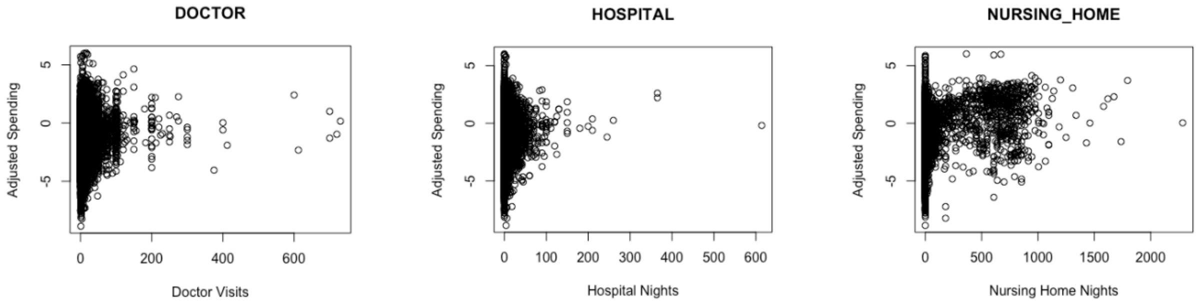
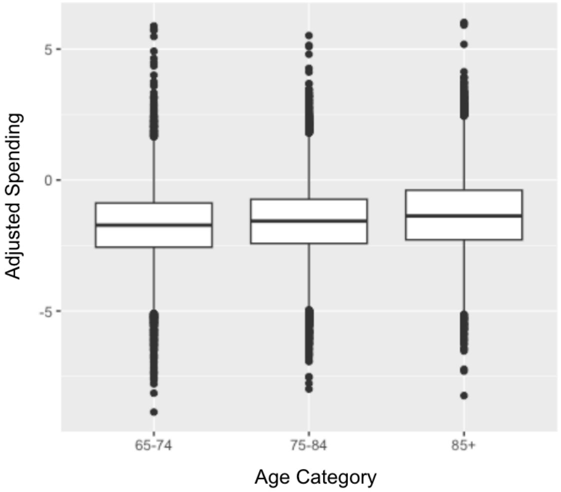


Figure 2. Continuous Predictors



Finally, in Figure 3 below, we plot the age and the log of adjusted spending. The distribution at all three age ranges is very similar, but there appears to be a gradual increase in adjusted spending with age.

Figure 3. Spending and Age



Methodologies

Figure 4 below shows ten sample spending paths from our training dataset. Our goal is to produce a simulation model that can reproduce such spending paths with a high degree of accuracy. In particular, the simulated spending paths must have the same statistical properties as the training dataset. We will compare both aggregate statistics (mean, median, and quantiles) and inter-path statistics (standard deviation, average percent increase, max, percent increase).

Figure 5 below shows one sample spending path magnified. The peaks correspond to spikes in spending caused by health shocks. For our purposes, a health shock can be many things: a heart attack, a new diagnosis, etc. Health shocks are the single biggest factor in determining spending.

Figure 4. 10 Sample Spending Paths

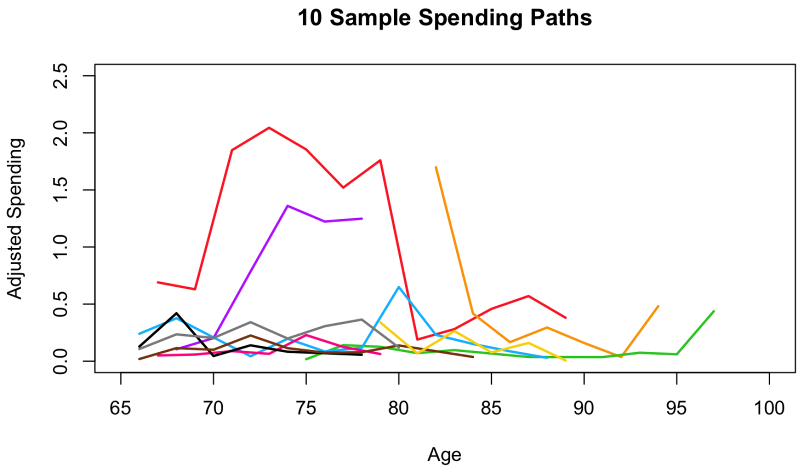
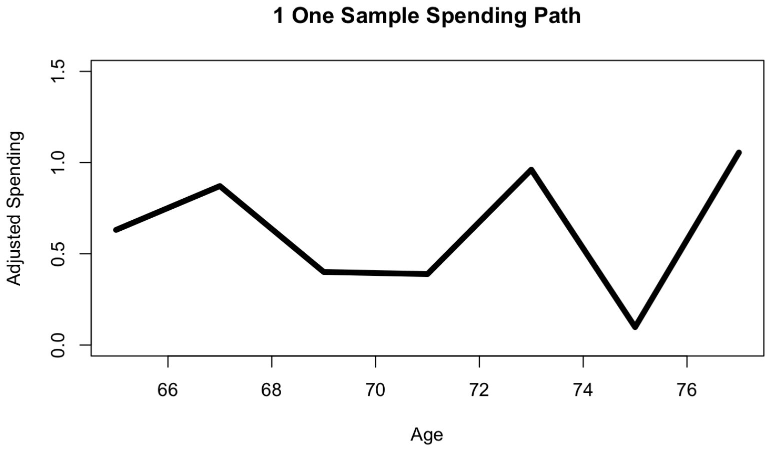


Figure 5. One Sample Spending Path



Our methodology must incorporate a way to deal with health shocks. We use two different approaches which are described in detail below. They differ in:

- the way they account for health shocks and,

- the method used for simulating the joint distribution of health variables.

More details on other attempted approaches are provided in the Appendix. We also described in detail how we arrived at the two-stage framework used in both methods.

<u>Method 1:</u>

In the first method, we use a two-stage approach. In step one, we use logistic regression to estimate the probability of a shock in a given year and then in step two, we fit separate linear regression models for adjusted spending on the predicted shock and no shock classes.

Previously, we have used the term shock loosely to describe both a sudden health event and the resulting spike in spending. To fit a logistic regression model we will need a concrete definition. The first metric we tested was t*he percent increase in spending from the previous year*. However, this threshold fails to capture successive shock years (as the increase from one shock year to another may be small or even negative) and our goal is to construct one class of high values and another of low values. Instead, we use *the percent increase from the non-shock baseline* where *the non-shock baseline* is defined as the average of all previous non-shock years. We can then define a new variable, SHOCK as:

$$SHOCK = \begin{cases} 1 & \text{if } SPEND\_SS \geq t * NON\_SHOCK\_BASELINE \\ 0 & \text{otherwise} \end{cases}$$

where *t* is the percentage increase classified as a shock. We can also define:

$$\Delta_p = P(SHOCK = 1) - \hat{P}(SHOCK = 1)$$

Figure 6 below shows the relationship between the threshold, *t*, $\Delta_{p,}$ and the test error. As the threshold increases, the absolute value of Δ_p and the test error decrease. We would like our simulation to contain an accurate approximation of the probability of a shock; however, we also want the shock value to be meaningful. Previous research conducted by the T. Rowe Price Group suggests that a 15% increase in adjusted spending (or greater) constitutes a financial burden. Thus, we chose $t = 1.15$ which has a very good Δ_p of 1% and a reasonable test error of 9%. Since our ultimate goal is simulation we prioritize minimizing Δ_p. With the optimal threshold selected, we can now fit all three models using backwards stepwise selection. Table 2 compares which variables are significant (at the .05 level) in each model. Tables 3 and 4 list the coefficients, standard errors, and p-values of the significant coefficients for each linear regression model. Note that cancer was not significant in any model so it is absent from all three tables.

Figure 6. Selecting an Optimal Threshold

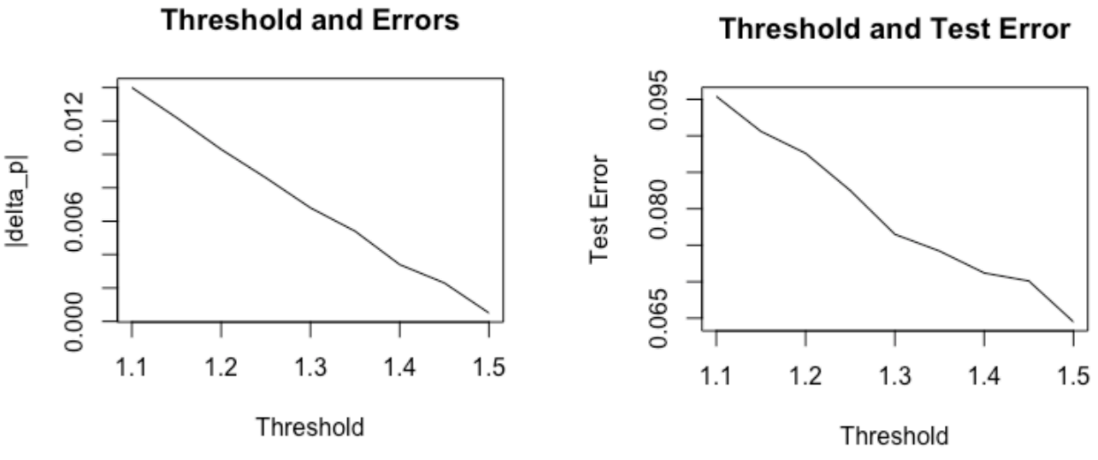


Table 2. Significant Variables (Method 1)

Variable	Step 1	Shock	No Shock

Age	✓	✓	✓
Health Change	✓		✓
High Blood Pressure	✓		
Heart Attack	✓		✓
Stroke	✓		✓
Arthritis	✓		
Drugs	✓	✓	✓
Special Facility	✓		✓
Doctor	✓	✓	✓
Hospital	✓	✓	✓
Nursing Home	✓	✓	✓
Psych		✓	✓
Outpatient			✓
Home Care		✓	✓
Lungs			✓

Table 3. Regression Coefficients, Standard Errors, and p-Values - Shock

Variable	**Coefficient**	**Standard Error**	**p-Value**
Intercept	-6.059	0.741	4.96e-16
Age	0.047	0.011	2.29e-05
Drugs	1.376	0.073	< 2e-16
Doctor	0.124	0.001	0.001
Hospital	0.004	0.001	0.004
Nursing Home	0.002	0.0002	6.58e-11
Psych	0.564	0.198	0.004
Home Care	0.255	0.120	0.034

Table 4. Regression Coefficients, Standard Errors, and p-Values - No Shock

Variable	Coefficient	Standard Error	p-Value
Intercept	-3.465	0.078	< 2e-16
Age	0.006	0.001	1.44e-10
Health Change	0.023	0.008	0.002
Heart Attack	0.112	0.034	0.001
Stroke	0.217	0.048	4.90e-06
Drugs	1.225	0.023	< 2e-16
Special Facility	0.098	0.019	1.73e-07
Doctor	0.009	0.0004	< 2e-16
Hospital	0.015	0.001	< 2e-16
Nursing Home	0.004	0.0001	< 2e-16
Psych	0.276	0.051	5.35e-08
Outpatient	0.168	0.015	< 2e-16
Home Care	0.195	0.023	3.77e-16
Lungs	0.123	0.053	0.021

Table 5 displays the performance of the Shock and No Shock models. We look at three statistics, adjusted R^2, the residual standard error (RSE), and the mean square error (MSE). R^2 is the proportion of the variance in the outcome variable that is explained by the model. It is a number between zero and one where a number closer to one indicates a better-fitting model. Adjusted R^2 is corrected for the number of predictors in the model (without the adjustment R^2 will continue to increase as the new variables are added to the model which can lead to overfitting). The formulas for the RSE and MSE are given below. Both are measurements of the prediction error, so smaller values are preferred. N is the number of observations and p is the number of predictors in the model. Y_i is the observed value of the outcome at i and Y_i “hat” is the predicted value of the outcome at i.

$$RSE = \sqrt{\sum_i (Y_i - \hat{Y}_i)^2/(N-p)}$$

$$MSE = \frac{1}{N}\sum_i (Y_i - \hat{Y}_i)^2$$

Table 5. Model Performance

Metrics/Classes	Shock	No Shock
Adj. R^2	0.387	0.128
RSE	1.425	1.265
MSE	2.116	1.646

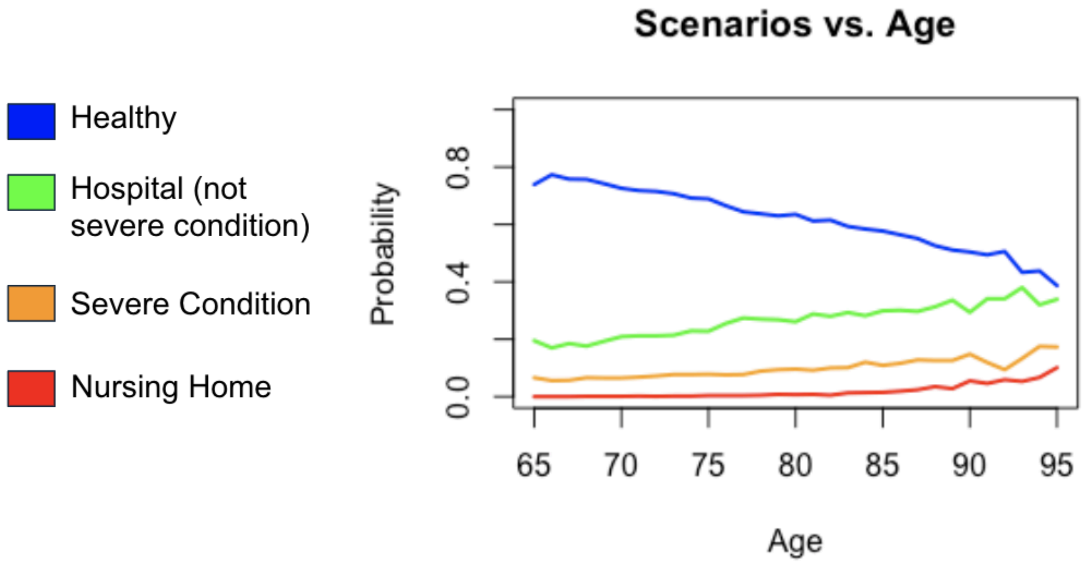
While both Adjusted R^2 values are relatively low, again, our final goal is simulation rather than prediction so we consider this reasonable performance.

The final piece needed to construct the simulation model is to estimate the joint distribution of the health variables. We simplify this task by reducing to the four most common scenarios: Healthy, Hospital (not a serious condition), Serious Condition, and Nursing Home. For our purposes, a “serious condition” is defined as “1=Yes” in any of the following columns: Heart Attack, Stroke, Psych, or Lungs variables (all are indicator variables). To construct health status, we can then simulate the values of each variable conditional on the selected scenario. Figure 7 displays how the probabilities of each scenario vary with age. As expected, the probability of being healthy declines as age increases.

We now have all of the pieces required to construct the final simulation. We simulate spending paths with starting ages and the number of waves observed at the relative frequencies they occur in the training set. Then, for a simulated person at a given age, we will simulate their spending as follows:

- Select a health status.
- Based on the selected health status, classify the year as a shock year or not.
- Use either the appropriate model to obtain an estimate of adjusted spending.

Figure 7. Approximating Joint Distribution of Health Variables

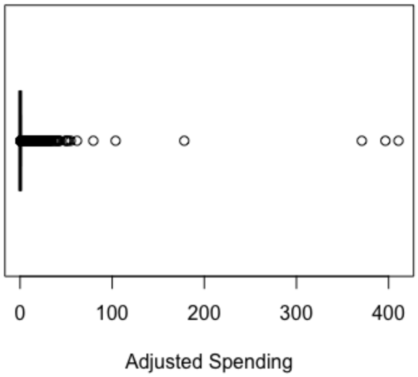


Method 2:

For Method 2, we begin by using a nonlinear dimension reduction technique t-SNE, and a density-based clustering algorithm called DBSCAN to construct clusters of similar observations from the health data.

Figure 8 below shows the adjusted spending data before dimensionality reduction. Clearly, there is a significant number of outliers. However, these points represent large shocks and are important data points which can not be removed (a simpler solution for outliers). Dimensionality reduction minimizes the impact of these outliers on the final model without removing them.

Figure 8. Adjusted Spending Data Before Clustering



Before performing the dimensionality reduction, we first normalize the data (and exclude the response variable from the feature set to remove any problematic correlation between our reduction and our response). t-SNE projects the high dimensional dataset into a smaller number of dimensions by converting similarities between data points into joint probabilities (Van Der Maaten & Hinton, 2008). This reduces the influence of noise and outliers, as well as increases the computational efficiency of the resulting clusters (Pedregosa et al., 2011). We then use DBSCAN to group the data points into neighborhoods. One benefit of DBSCAN is that it is noise and outlier-resistant (unlike other clustering methods like K-Means).

There are two major hyperparameters to tune: the perplexity in t-SNE and EPS in DBSCAN. The perplexity is similar to the number of nearest neighbors (t-SNE documentation). EPS is “the maximum distance between one sample considered as in the neighborhood of the other” (Pedregosa et al., 2011). EPS is significantly more important as t-SNE is relatively resistant to the choice of perplexity but DBSCAN is highly sensitive to the choice of EPS. After tuning, we find that perplexity = 1800 and EPS = 7 are the optimal values. The final result had three clusters, which we found could be reduced (for interpretability) down to 2 clusters without significant loss of predictive ability. Cancer and home care appear to be the most significant separating factors for the clusters. Figure 9 below shows the final clustering. The clusters are highly unbalanced (similar to the predicted classes from Method 1) which reflects the rarity of shock events. Figure 10 and Table 6 below compare the results of the clustering to the SHOCK classes from Method 1.

Figure 9. Final Clustering

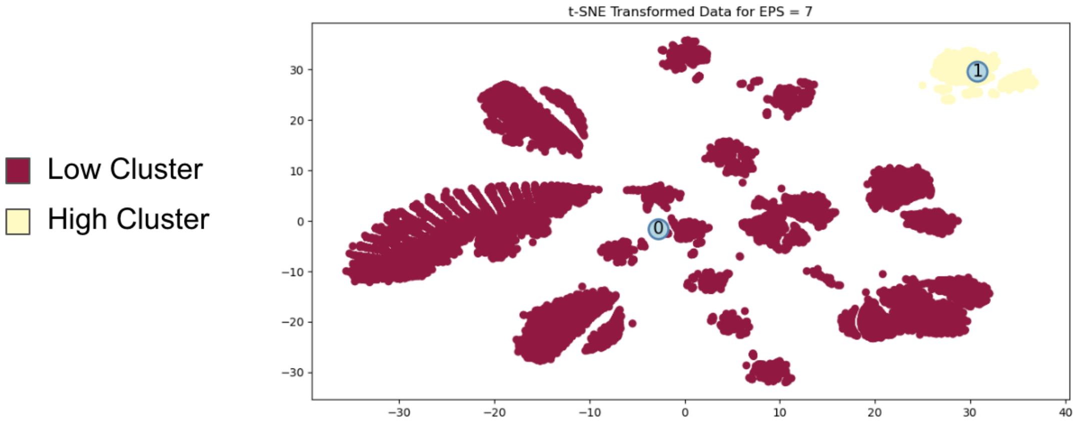
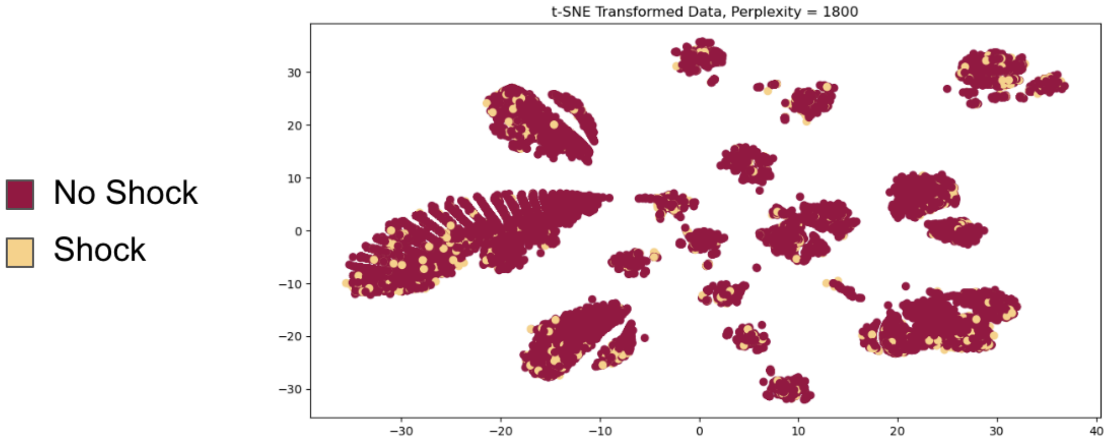


Table 6. Probability of SHOCK by Cluster

Cluster	Probability of Wealth Shock
High	9.37%
Low	6.74%
All Clusters	6.93%

Figure 10. Comparison to SHOCK Classes



The probability of a shock in the low cluster is similar to the global probability, while the probability of a shock in the high cluster is higher. Thus, our clustering result yields an intuitive result where the individuals in the high cluster have a 40% higher probability of a health shock (and correspondingly higher spending).

To verify our clustering result, we use PERMANOVA (Permutational Multivariate Analysis of Variance) and ANOSIM (Analysis of Similarities) to test if the final clusters are still statistically significantly different under our original feature space. PERMANOVA is a non-parametric multivariate statistical permutation test based on a geometric partitioning of multivariate variation under a chosen similarity measure (Anderson, 2017). The sole assumption of this test is that, under the true null hypothesis, our observations must be exchangeable, allowing them to be used in a wide variety of contexts. The null hypothesis is that the multivariate centroids and dispersion of the clusters are the same for all groups, indicating no difference between the clusters. ANOSIM is a non-parametric test that determines significant differences between two or more groups using ranked similarity (Clarke K., & Green, R. 1988). ANOSIM does not require any distributional assumptions. The null hypothesis is that there is no significant difference between intra-cluster similarity and inter-cluster similarity. In both tests, the Euclidean distance of normalized points from the original space is used as the distance metric (this was done to verify that both the clustering and the dimensionality reduction did not produce clusters that could be considered statistically similar under the original feature space). The Python package skbio.math.stats.distance was used to conduct both tests. For both ANOSIM and PERMANOVA, we reject the null hypothesis and conclude that there are statistically significant differences between the members of each cluster.

Next, a linear regression model for adjusted spending is fitted on the high and low clusters respectively. The results are displayed below in Tables 7, 8, and 9.

Table 7. Results of Linear Regression

Metrics/Clusters	High	Low
Adj. R^2	0.324	0.908
RSE	3.249	2.600
MSE	4.226	4.083

Table 8. Regression Coefficients, Standard Errors, and p-Values - Low

Variable	Coefficient	Standard Error	p-Value
Age	0.0179	0.002	0.000
Health Change	0.0350	0.013	0.007
Drugs	0.2184	0.039	0.000
Doctor	0.0027	0.001	0.000
Nursing Home	0.0089	0.000	0.000
Stroke	0.2009	0.083	0.016

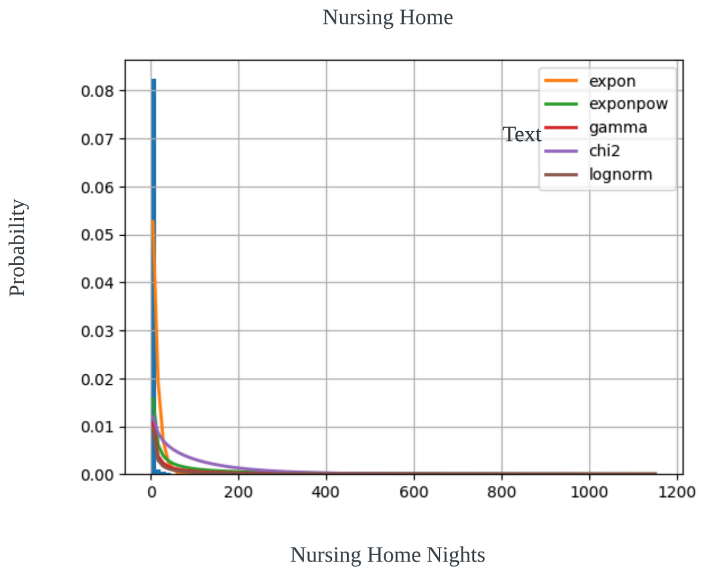
Psych	0.5215	0.089	0.000

Table 9. Regression Coefficients, Standard Errors, and p-Values - High

Variable	Coefficient	Standard Error	p-Value
Age	0.0201	0.002	0.000
Stroke	0.2098	0.092	0.023
Doctor	0.0029	0.001	0.000
Hospital	0.0050	0.002	0.004
Nursing Home	0.0091	0.000	0.000
Psych	0.5579	0.099	0.000

Finally, we approximate the distribution of each health variable. Due to the complexity of the data and the sparsity of the indicator variables, approximating the joint distribution directly is impossible. Instead, we will estimate the distribution of each variable separately (which relies on the simplifying assumption that the variables are independent). We will use the FITTER module in Python for approximation. FITTER attempts to fit 80 different distributions to the provided data and outputs a summary of the best distributions (which minimize the sum of squared errors) (FITTER documentation). Figure 11 below shows the FITTER output for the NURSING_HOME variable.

Figure 11. FITTER Output



We can now construct the final simulation. We simulate the starting ages and the number of waves observed using the same mechanism from Method 1 and construct a similar simulation model.

Results

It is important that the simulated data have the same statistical properties as the training data. We begin investigating the accuracy of the simulated data by comparing aggregate statistics across ages to the training data. For Figures 12 and 14 below, the red is the simulated data and the blue is the training data.

Figure 12. Aggregate Statistics Method 1

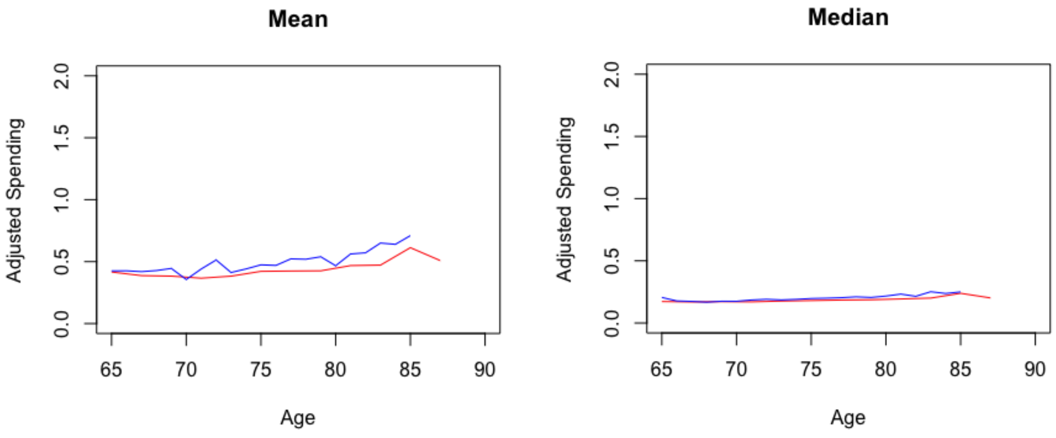
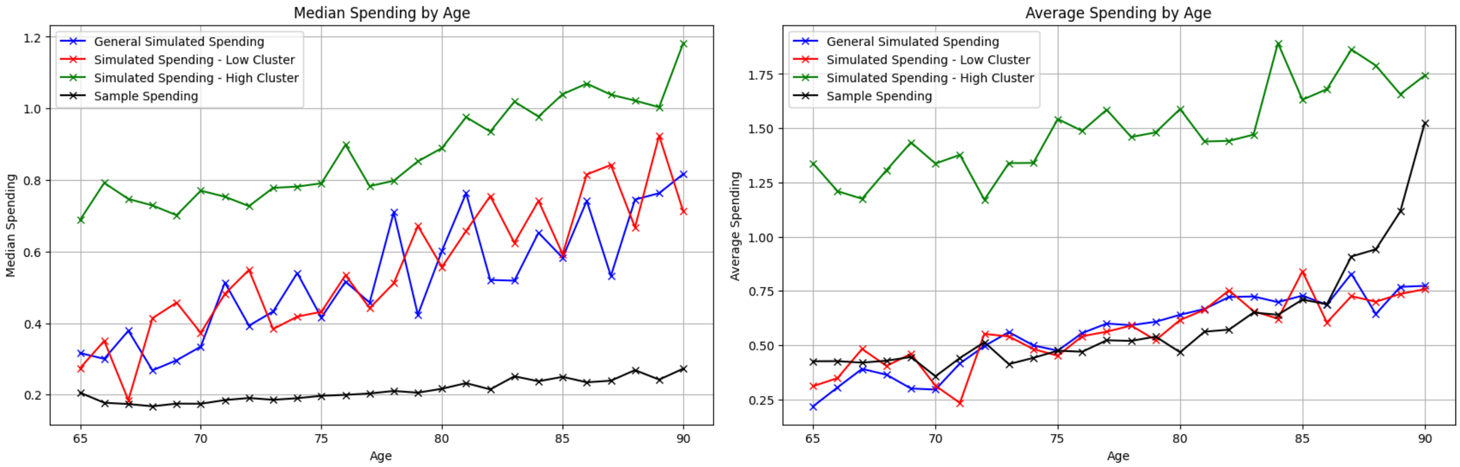
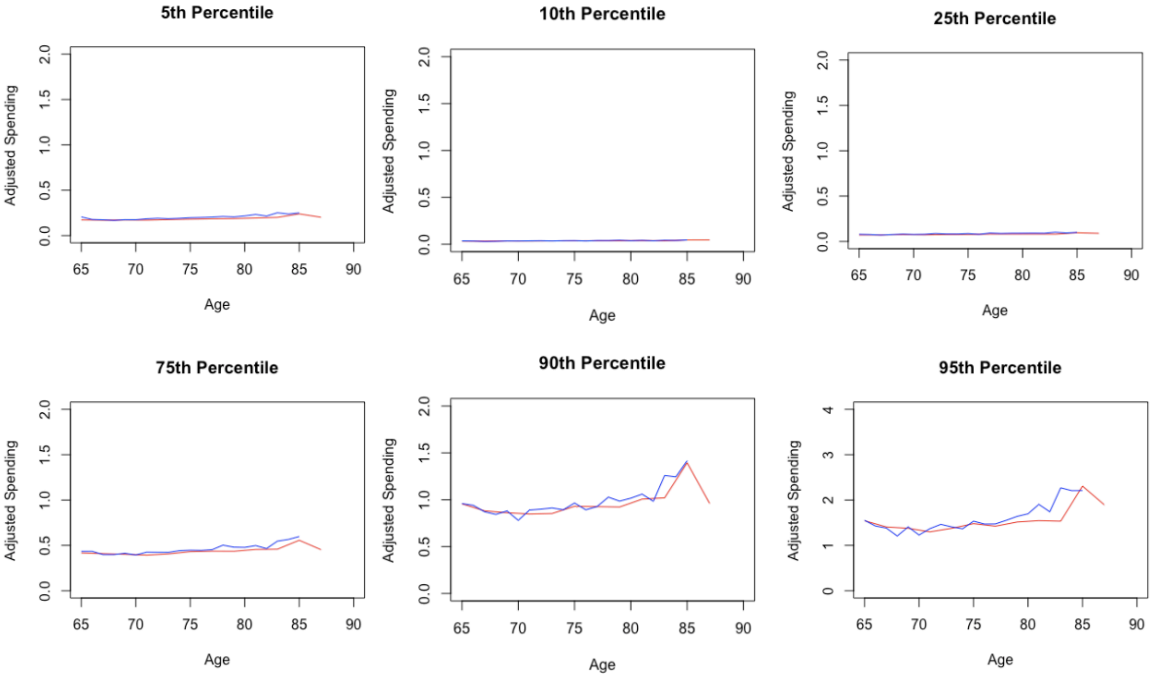


Figure 13. Aggregate Statistics for Method 2



From comparing Figures 12 and 13 it is clear that Method 1 produces a more accurate simulation than Method 2. We will limit our focus to Method 1 and conduct an extensive comparison of the simulated data to the training data, beginning with the quantiles.

Figure 14. Quantiles



The lower quantiles are very similar. The upper quantiles show the most deviation, particularly at large values of age. However, the simulation still has reasonable performance for all aggregate statistics.

It is also important that the simulated paths have similar variations to the training paths and are not too jagged. Figure 15 below shows 1000 randomly sampled paths from the simulated data set plotted over 1000 randomly sampled paths from the training set. The visual comparison is promising: however, we will compute several statistics to be sure. In Figure 16 we compare the standard deviation, average % increase, and max % increase. Note that Figure 16 below plots the log of adjusted spending to increase the readability of the plots. In general, we find that the simulated data has less variation, which is unsurprising. Due to the nature of our models and the simulation technique, the simulation cannot reproduce the most extreme wealth shocks seen in the training dataset.

Figure 15. 2000 Randomly Sampled Spending Paths

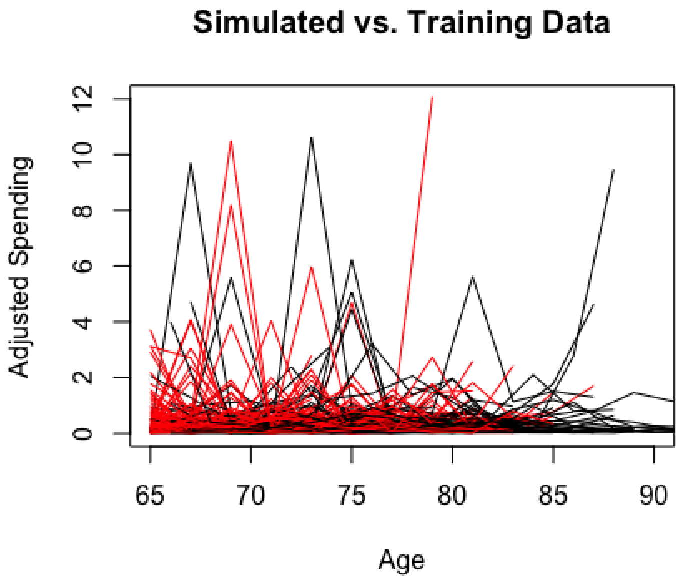
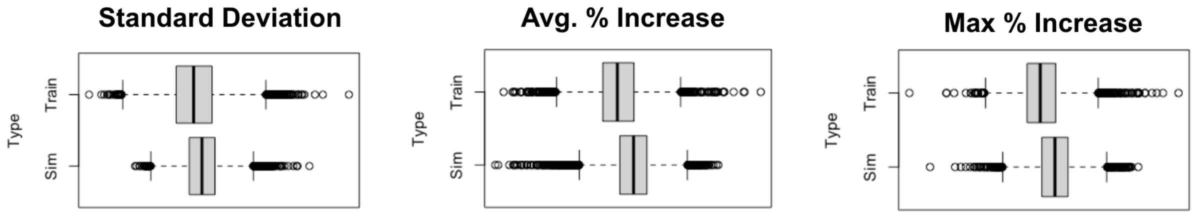


Figure 16. Inter-Path Statistics



The final measure of smoothness we compute is the average second difference. For a dataset with n observations where each observation is a spending path X_i for $1 \leq i \leq n$, then X_i is a vector of the adjusted spending values at each age, with $X_i = (S_1,..., S_j)$ where j is the number of waves individual i was observed with $j \epsilon [5, 13]$. Then we can compute the average second difference, ASD, for the dataset as:

$$ASD = \frac{\sum_i (\sum_j [S_j - 2S_{j-1} - S_{j-2}]^2/4)}{n}$$

Applying this formula to both datasets we find that the simulated data has an average second difference of 6.11 and the training data has an average second difference of 35. The average second difference mimics the second derivative and is a measure of curvature and smoothness of time series data. We find that the simulated data is smoother than the training data. This result is encouraging as it indicates that our assumption of independence of observations did not result in overly jagged paths. The results are displayed below:

Table 10. Average Second Difference Results

Dataset/Metric	ASD	ASD (w/out extreme outliers)
Simulated	6.11	5.56
Training	35.05	18.21

Conclusions

We conclude that Method 1 creates a more accurate simulation of the adjusted spending. This result is surprising given that Method 1 is a more intuitive framework that uses traditional statistical methods while Method 2 leverages machine learning techniques and automation. It appears the greatest flaw in the Method 2 framework is the simplifying assumption that the variables are independent. An interesting extension of this project would be to combine the estimated joint distribution from Method 1 with the cluster models from Method 2, which would allow for a direct comparison of the clustering and the logistic regression. Other potential improvements to Method 2 are described in detail in the Appendix.

Home care appears to be the most significant factor in determining spending as it was a significant separating factor in the clustering and significant in both the Shock and No Shock regression models. Interestingly, cancer, the other significant separating factor from Method 2, was not significant in any of the Method 1 models.

Drugs is the only indicator variables significant in all three models from Method 1. It is also the only indicator variable where the mode was 1 (meaning the majority of individuals used prescription medication). All of the continuous predictors: Age, Doctor, Hospital, and Nursing Home, were significant in all three models. Age, Hospital, and Nursing Home were also crucial for approximating health status, so we can conclude these are the three most significant variables.

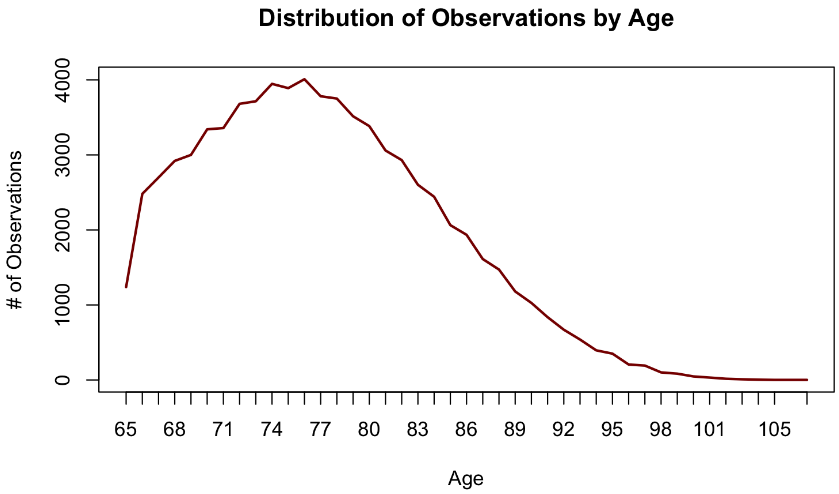
The greatest difference between the training and simulated data is that the simulated data has less variation; however, this result is a positive indication as it is significantly easier to add variability to the simulation than it is to remove unexplained variation. We considered adding more variation to the existing simulation; however, after discussion with T. Rowe, we decided that a model with less variation was preferable, even if it is different from the training set.

Ultimately, we conclude that our final simulation model fulfills the requirements outlined in the project brief and constructs a reasonable and accurate simulation of adjusted out-of-pocket medical spending for ages 65-85.

Limitations

Our model has several limitations. The first is the lack of training data at older ages. As shown in Figure 17 the number of observations at age begins to drop sharply after age 85. While our simulation may reproduce reasonable estimates of spending for ages 65-85 it must be improved, possibly by obtaining data from a different source, before being applied for individuals age 85 and older.

Figure 17. Observations by Age



The second limiting factor is that the majority of our predictors are indicator variables. It is telling that all of the most significant predictors are continuous. It is possible that the model could be greatly improved by the addition of more continuous variables. It is also possible that some significant variables were missed in our initial selection. The sheer size and complexity of the HRS make it impossible to test every combination of predictors.

The final limitation is a limitation of the HRS dataset itself. Nursing home usage is one of the main drivers of out-of-pocket medical spending at older ages; however, the HRS often loses touch with participants when they enter assisted living or a nursing home facility. The RAND Center for the Study of Aging estimates that around 54% of individuals use a nursing home at some point in their life. They compared this to previous conservative estimates of 35% (Hurd, 2017). Our dataset only estimates this probability at around 27%, half of the RAND estimate. Therefore, although our simulation may be accurate to the training dataset, that is not a guarantee that the training dataset is a faithful representation of reality.

Acknowledgments

We would like to thank Dr. Sudipto Banerjee, Dr. Louisa Schafer, and T. Rowe Price for the opportunity to work on this project.

We would also like to express our gratitude to Dr. David Ruppert for his invaluable guidance throughout this project. His expertise and insights have been crucial in shaping the direction and execution of our work.

Last, but not least, we would like to thank Dr. Xiaolong Yang and Prof. Sreyoshi Das for their support during the MPS Applied Statistics Capstone Project.

Appendix

The appendix contains more information about the other types of models that we tested and they informed our final model selection. It also contains proposed improvements that could be made to Method 2.

Notes on Regime Switching

The initial goal of our project was to utilize regime switching models to incorporate the idea of health shocks into our model. Regime switching models (also called hidden-Markov models) assume that there is an underlying Markov chain. A Markov chain is a collection of states where the movement between states is governed by a matrix of transition probabilities. In the regime-switching framework, a separate time series model can be fitted for each state. Hidden Markov models have two assumptions:

1. The future is independent of the past, conditional on the present.
2. The transition probabilities are homogeneous in time.

Applied to our specific task the second assumption would mean that the probability of having a health shock was constant across age. As evidenced by Figure 8 (and that age is highly significant in our logistic regression model from Method 1), this assumption clearly does not hold. The first assumption is also somewhat problematic, as it implies that the probability of having a health shock is independent of previous health shocks (note that Methods 1 and 2 also make this simplifying assumption). However, the most serious issue is that the final hidden-Markov model would contain a global probability of experiencing a health shock, which would not capture the dependence on age. The two-stage approaches outlined in Methods 1 and 2 capture the "framework" of regime-switching without this restrictive assumption.

<u>Notes on LSTMs</u>

A "long short term memory", LSTM model is a type of neural network with specialized memory cells. A typical neural network has "long-term memory in the form of weights" and "short-term memory in terms of activations" (where the weights represent the influence of a particular observation on the final model and activations represent the movement from node to node) (Long Short Term Memory (LSTM) — Dive into Deep Learning documentation, n.d). Memory cells contain a sort of a hybrid memory, encoding both long and short term information. Initially, this unique structure seemed well suited to our task as it could utilize both current and historical health data. It seemed likely that individuals who had already experienced one health shock would be more likely to experience another (as compared to an individual who had previously been healthy). However, we actually found the opposite. Considering all previous observations at every step was “too much information” and led to poor performance. This result supports our decision to use linear regression (and assume that observations were independent of previous observations). The LSTM also contains a large number of parameters and is therefore:

1. costly to fit and
2. requires very large sample sizes to prevent overfitting.

Notes on Utilizing a Health Variable as the Outcome:

We also considered utilizing one of the health indicator variables (such as heart attack or the HRS dataset also contains an indicator variable for hospital use) as the outcome. However, we found that none of the health variables alone “guaranteed” a shock. For example, using the hospital in a given wave did not automatically lead to higher spending in that wave. This information was crucial in informing our final model selection and creating the simplified joint distribution used in Method 1.

Potential Improvements to Method 2:

Method 2 was developed in parallel with Method 1. It is an attempt to leverage machine learning methods over traditional statistical models. The combination of t-SNE and DBSCAN was chosen for its performance on a previous project with similar parameters (survey data with sparse categorical variables). As is discussed in the Methods section, the effectiveness of DBSCAN is its reliance on density rather than the mean or variance, which can be influenced by correlation between variables and outliers. Each variable independently contributes to the local density estimation. Another benefit of clustering (as opposed to logistic regression or any other supervised learning attempt) is that it groups the data without imposing an arbitrary threshold, as in Method 1. The clustering successfully identified the high and low clusters; therefore, it seems clear that it was the assumption of independence of the predictors that contributed to the poor performance of Method 2. One potential improvement is to use the simplified joint distribution from Method 1. Another potential solution is to use Multivariate Bayesian Networks (MBNs), which are capable of modeling complex relationships among a set of variables (using probability and graph theory). They are useful in scenarios such as ours where understanding the dependency between predictors is crucial.

References

Anderson, M. J. (2017). Permutational Multivariate Analysis of Variance (PERMANOVA). In *Wiley StatsRef: Statistics Reference Online* (pp. 1–15). doi:10.1002/9781118445112.stat07841

Clarke K., & Green, R. (1988). Statistical Design and Analysis for a 'Biological Effects' Study. *Marine Ecology - Progress Series*, 46, 213–226. doi:10.3354/meps046213

"FITTER: A Python Package for Distribution Fitting." Accessed. https://fitter.readthedocs.io/en/latest/.

"Health and Retirement Study (HRS) Data Products." Institute for Social Research, University of Michigan. Accessed. https://hrs.isr.umich.edu/data-products.

Hurd, Michael D. "Average American's Risk of Needing Nursing Home Care Is Higher than Previously Estimated." *RAND Center for the Study of Aging*, RAND Corporation, 28 Aug. 2017, www.rand.org/news/press/2017/08/28/index1.html.

Long Short Term Memory (LSTM) — Dive into Deep Learning documentation. (n.d.). D2l.ai. https://d2l.ai/chapter_recurrent-modern/lstm.html

"priceR: Economics and Pricing Tools." Accessed. https://cran.r-project.org/web/packages/priceR/index.html.

Pedregosa *et al.,* Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011.

Van der Maaten, L.J.P.; Hinton, G.E. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9:2579-2605, 2008.

Code

Data Clean

******* download_and_sample.sas*******

```
/*Set libraries*/
LIBNAME proj "C:\Users\jessr\OneDrive\Desktop\MPS Project\randhrs1992_2020v1_SAS";
LIBNAME library "C:\Users\jessr\OneDrive\Desktop\MPS Project\randhrs1992_2020v1_SAS";

/*Load formats*/
proc format cntlin=proj.sasfmts;
quit;

/*Read-in data*/
DATA hrs;
        SET proj.randhrs1992_2020v1;
        KEEP HHIDPN RABYEAR RABMONTH RABDATE R1GOVMR R2GOVMR R3GOVMR
R4GOVMR R5GOVMR R6GOVMR R7GOVMR R8GOVMR R9GOVMR R10GOVMR
                R11GOVMR R12GOVMR R13GOVMR R14GOVMR R15GOVMR R1WTRESP
R2WTRESP R3WTRESP R4WTRESP R5WTRESP R6WTRESP R7WTRESP R8WTRESP
                R9WTRESP R10WTRESP R11WTRESP R12WTRESP R13WTRESP
R14WTRESP H1ATOTF H2ATOTF H3ATOTF H4ATOTF H5ATOTF H6ATOTF H7ATOTF
                H8ATOTF H9ATOTF H10ATOTF H11ATOTF H12ATOTF H13ATOTF H14ATOTF
H15ATOTF R2SHLTC R3SHLTC R4SHLTC R5SHLTC R6SHLTC R7SHLTC
                R8SHLTC R9SHLTC R10SHLTC R11SHLTC R12SHLTC R13SHLTC R14SHLTC
R15SHLTC R8CONDS R9CONDS R1HOMCAR R2HOMCAR R3HOMCAR R4HOMCAR
     R5HOMCAR R6HOMCAR R7HOMCAR R8HOMCAR R9HOMCAR R10HOMCAR
R11HOMCAR R12HOMCAR R13HOMCAR R14HOMCAR R15HOMCAR R2DRUGS
R3DRUGS
     R4DRUGS R5DRUGS R6DRUGS R7DRUGS R8DRUGS R9DRUGS R10DRUGS
R11DRUGS R12DRUGS R13DRUGS R14DRUGS R15DRUGS R2OUTPT R3OUTPT
R4OUTPT
     R5OUTPT R6OUTPT R7OUTPT R8OUTPT R9OUTPT R10OUTPT R11OUTPT
R12OUTPT R13OUTPT R14OUTPT R15OUTPT R2SPCFAC R3SPCFAC R4SPCFAC
R5SPCFAC
     R6SPCFAC R7SPCFAC R8SPCFAC R9SPCFAC R10SPCFAC R11SPCFAC R12SPCFAC
R13SPCFAC R14SPCFAC R15SPCFAC R2OOPMD R3OOPMD R4OOPMD R5OOPMD
     R6OOPMD R7OOPMD R8OOPMD R9OOPMD R10OOPMD R11OOPMD R12OOPMD
R13OOPMD R14OOPMD R15OOPMD R1ISRET R2ISRET R3ISRET R4ISRET R5ISRET
     R6ISRET R7ISRET R8ISRET R9ISRET R10ISRET R11ISRET R12ISRET R13ISRET
R14ISRET R15ISRET H1ADEBT H2ADEBT H3ADEBT H4ADEBT H5ADEBT
     H6ADEBT H7ADEBT H8ADEBT H9ADEBT H10ADEBT H11ADEBT H12ADEBT
H13ADEBT H14ADEBT H15ADEBT RASSRECV R4DCBAL1 R5DCBAL1 R6DCBAL1
R7DCBAL1
                R8DCBAL1 R9DCBAL1 R10DCBAL1 R11DCBAL1 R12DCBAL1 R13DCBAL1
R14DCBAL1 R15DCBAL1 R4DCBAL2 R5DCBAL2 R6DCBAL2 R7DCBAL2 R8DCBAL2
R9DCBAL2
     R10DCBAL2 R11DCBAL2 R12DCBAL2 R13DCBAL2 R14DCBAL2 R15DCBAL2
R4DCBAL3 R5DCBAL3 R6DCBAL3 R7DCBAL3 R8DCBAL3 R9DCBAL3 R10DCBAL3
```

```
R11DCBAL3
            R12DCBAL3 R13DCBAL3 R14DCBAL3 R15DCBAL3 R5DCBAL4 R6DCBAL4
R7DCBAL4 R8DCBAL4 R9DCBAL4 R10DCBAL4 R11DCBAL4 R12DCBAL4
            R13DCBAL4 R14DCBAL4 R15DCBAL4 S4DCBAL1 S5DCBAL1 S6DCBAL1
S7DCBAL1 S8DCBAL1 S9DCBAL1 S10DCBAL1 S11DCBAL1 S12DCBAL1 S13DCBAL1
            S14DCBAL1 S15DCBAL1 S4DCBAL2 S5DCBAL2 S6DCBAL2 S7DCBAL2
S8DCBAL2 S9DCBAL2 S10DCBAL2 S11DCBAL2 S12DCBAL2 S13DCBAL2 S14DCBAL2
            S15DCBAL2 S4DCBAL3 S5DCBAL3 S6DCBAL3 S7DCBAL3 S8DCBAL3
S9DCBAL3 S10DCBAL3 S11DCBAL3 S12DCBAL3 S13DCBAL3 S14DCBAL3 S15DCBAL3
            S5DCBAL4 S6DCBAL4 S7DCBAL4 S8DCBAL4 S9DCBAL4 S10DCBAL4
S11DCBAL4 S12DCBAL4 S13DCBAL4 S14DCBAL4 S15DCBAL4 H4AIRA H5AIRA
            H6AIRA H7AIRA H8AIRA H9AIRA H10AIRA H11AIRA H12AIRA H13AIRA
H14AIRA H15AIRA H4ASTCK H5ASTCK H6ASTCK H7ASTCK H8ASTCK H9ASTCK
H10ASTCK
            H11ASTCK H12ASTCK H13ASTCK H14ASTCK H15ASTCK H4ACHCK
H5ACHCK H6ACHCK H7ACHCK H8ACHCK H9ACHCK H10ACHCK H11ACHCK H12ACHCK
            H13ACHCK H14ACHCK H15ACHCK H4ACD H5ACD H6ACD H7ACD H8ACD
H9ACD H10ACD H11ACD H12ACD H13ACD H14ACD H15ACD H4ABOND H5ABOND
H6ABOND H7ABOND
            H8ABOND H9ABOND H10ABOND H11ABOND H12ABOND H13ABOND
H14ABOND H15ABOND H4AOTHR H5AOTHR H6AOTHR H7AOTHR H8AOTHR H9AOTHR
H10AOTHR H11AOTHR
            H12AOTHR H13AOTHR H14AOTHR H15AOTHR R1HSPNIT R2HSPNIT
R3HSPNIT R4HSPNIT R5HSPNIT R6HSPNIT R7HSPNIT R8HSPNIT R9HSPNIT R10HSPNIT
R11HSPNIT
    R12HSPNIT R13HSPNIT R14HSPNIT R15HSPNIT R1NRSNIT R2NRSNIT R3NRSNIT
R4NRSNIT R5NRSNIT R6NRSNIT R7NRSNIT R8NRSNIT R9NRSNIT R10NRSNIT
R11NRSNIT
            R12NRSNIT R13NRSNIT R14NRSNIT R15NRSNIT R1DOCTIM R2DOCTIM
R3DOCTIM R4DOCTIM R5DOCTIM R6DOCTIM R7DOCTIM R8DOCTIM R9DOCTIM
R10DOCTIM R11DOCTIM
    R12DOCTIM R13DOCTIM R14DOCTIM R15DOCTIM R4DIABS R5DIABS R6DIABS
R7DIABS R8DIABS R9DIABS R10DIABS R11DIABS R12DIABS R13DIABS R14DIABS
R15DIABS
    R4CANCRS R5CANCRS R6CANCRS R7CANCRS R8CANCRS R9CANCRS
R10CANCRS R11CANCRS R12CANCRS R13CANCRS R14CANCRS R15CANCRS R4HIBPS
R5HIBPS R6HIBPS R7HIBPS
    R8HIBPS R9HIBPS R10HIBPS R11HIBPS R12HIBPS R13HIBPS R14HIBPS R15HIBPS
R4LUNGS R5LUNGS R6LUNGS R7LUNGS R8LUNGS R9LUNGS R10LUNGS R11LUNGS
R12LUNGS
    R13LUNGS R14LUNGS R15LUNGS R4HEARTS R5HEARTS R6HEARTS R7HEARTS
R8HEARTS R9HEARTS R10HEARTS R11HEARTS R12HEARTS R13HEARTS R14HEARTS
R15HEARTS
    R4STROKS R5STROKS R6STROKS R7STROKS R8STROKS R9STROKS R10STROKS
R11STROKS R12STROKS R13STROKS R14STROKS R15STROKS R4PSYCHS R5PSYCHS
R6PSYCHS
    R7PSYCHS R8PSYCHS R9PSYCHS R10PSYCHS R11PSYCHS R12PSYCHS
R13PSYCHS R14PSYCHS R15PSYCHS R4ARTHRS R5ARTHRS R6ARTHRS R7ARTHRS
R8ARTHRS R9ARTHRS
    R10ARTHRS R11ARTHRS R12ARTHRS R13ARTHRS R14ARTHRS R15ARTHRS;
RUN;

/*Check everything is created correctly*/
PROC CONTENTS data=hrs varnum;
```

```
RUN;

/*Select a random subset*/
proc surveyselect data=hrs
   out=proj.hrs_sample
   method=srs
   sampsize=5000
   seed=123;
run;

/*Export as csv*/
proc export data=proj.hrs_sample
   outfile="C:\Users\jessr\OneDrive\Desktop\MPS
Project\randhrs1992_2020v1_SAS\hrs_sample.csv"
   dbms=csv
   replace;
run;


proc export data=hrs
   outfile="C:\Users\jessr\OneDrive\Desktop\MPS
Project\randhrs1992_2020v1_SAS\hrs_new.csv"
   dbms=csv
   replace;
run;
```

***** data_clean.R*****

```
# Read-in data
hrs <- read.csv(paste(pwd, "/Data/hrs_new.csv", sep = ''))

# For price adjustments - used later
country <- "US"
inflation_dataframe <- retrieve_inflation_data(country)
countries_dataframe <- show_countries()

# Create age variable (age in wave 4)
hrs <- mutate(hrs, AGE=1998-as.numeric(RABYEAR)) %>%
  # Delete observations where age is missing
  drop_na(AGE) %>%
  # Variables that will be included in liquid assets, replace NAs w/ zeros
  ## RwDCBAL1
  mutate(R4DCBAL1=ifelse(is.na(as.numeric(R4DCBAL1)), 0, as.numeric(R4DCBAL1))) %>%
  mutate(R5DCBAL1=ifelse(is.na(as.numeric(R5DCBAL1)), 0, as.numeric(R5DCBAL1))) %>%
  mutate(R6DCBAL1=ifelse(is.na(as.numeric(R6DCBAL1)), 0, as.numeric(R6DCBAL1))) %>%
  mutate(R7DCBAL1=ifelse(is.na(as.numeric(R7DCBAL1)), 0, as.numeric(R7DCBAL1))) %>%
  mutate(R8DCBAL1=ifelse(is.na(as.numeric(R8DCBAL1)), 0, as.numeric(R8DCBAL1))) %>%
  mutate(R9DCBAL1=ifelse(is.na(as.numeric(R9DCBAL1)), 0, as.numeric(R9DCBAL1))) %>%
  mutate(R10DCBAL1=ifelse(is.na(as.numeric(R10DCBAL1)), 0, as.numeric(R10DCBAL1))) 
%>%
  mutate(R11DCBAL1=ifelse(is.na(as.numeric(R11DCBAL1)), 0, as.numeric(R11DCBAL1))) 
%>%
  mutate(R12DCBAL1=ifelse(is.na(as.numeric(R12DCBAL1)), 0, as.numeric(R12DCBAL1))) 
%>%
  mutate(R13DCBAL1=ifelse(is.na(as.numeric(R13DCBAL1)), 0, as.numeric(R13DCBAL1))) 
%>%
  mutate(R14DCBAL1=ifelse(is.na(as.numeric(R14DCBAL1)), 0, as.numeric(R14DCBAL1))) 
%>%
  mutate(R15DCBAL1=ifelse(is.na(as.numeric(R15DCBAL1)), 0, as.numeric(R15DCBAL1))) 
%>%
  ## RwDCBAL2
  mutate(R4DCBAL2=ifelse(is.na(as.numeric(R4DCBAL2)), 0, as.numeric(R4DCBAL2))) %>%
  mutate(R5DCBAL2=ifelse(is.na(as.numeric(R5DCBAL2)), 0, as.numeric(R5DCBAL2))) %>%
  mutate(R6DCBAL2=ifelse(is.na(as.numeric(R6DCBAL2)), 0, as.numeric(R6DCBAL2))) %>%
  mutate(R7DCBAL2=ifelse(is.na(as.numeric(R7DCBAL2)), 0, as.numeric(R7DCBAL2))) %>%
  mutate(R8DCBAL2=ifelse(is.na(as.numeric(R8DCBAL2)), 0, as.numeric(R8DCBAL2))) %>%
  mutate(R9DCBAL2=ifelse(is.na(as.numeric(R9DCBAL2)), 0, as.numeric(R9DCBAL2))) %>%
  mutate(R10DCBAL2=ifelse(is.na(as.numeric(R10DCBAL2)), 0, as.numeric(R10DCBAL2))) 
%>%
  mutate(R11DCBAL2=ifelse(is.na(as.numeric(R11DCBAL2)), 0, as.numeric(R11DCBAL2))) 
%>%
  mutate(R12DCBAL2=ifelse(is.na(as.numeric(R12DCBAL2)), 0, as.numeric(R12DCBAL2))) 
%>%
  mutate(R13DCBAL2=ifelse(is.na(as.numeric(R13DCBAL2)), 0, as.numeric(R13DCBAL2))) 
%>%
  mutate(R14DCBAL2=ifelse(is.na(as.numeric(R14DCBAL2)), 0, as.numeric(R14DCBAL2))) 
%>%
  mutate(R15DCBAL2=ifelse(is.na(as.numeric(R15DCBAL2)), 0, as.numeric(R15DCBAL2))) 
%>%
  ## RwDCBAL3
  mutate(R4DCBAL3=ifelse(is.na(as.numeric(R4DCBAL3)), 0, as.numeric(R4DCBAL3))) %>%
  mutate(R5DCBAL3=ifelse(is.na(as.numeric(R5DCBAL3)), 0, as.numeric(R5DCBAL3))) %>%
  mutate(R6DCBAL3=ifelse(is.na(as.numeric(R6DCBAL3)), 0, as.numeric(R6DCBAL3))) %>%
  mutate(R7DCBAL3=ifelse(is.na(as.numeric(R7DCBAL3)), 0, as.numeric(R7DCBAL3))) %>%
  mutate(R8DCBAL3=ifelse(is.na(as.numeric(R8DCBAL3)), 0, as.numeric(R8DCBAL3))) %>%
  mutate(R9DCBAL3=ifelse(is.na(as.numeric(R9DCBAL3)), 0, as.numeric(R9DCBAL3))) %>%
  mutate(R10DCBAL3=ifelse(is.na(as.numeric(R10DCBAL3)), 0, as.numeric(R10DCBAL3))) 
%>%
  mutate(R11DCBAL3=ifelse(is.na(as.numeric(R11DCBAL3)), 0, as.numeric(R11DCBAL3))) 
%>%
  mutate(R12DCBAL3=ifelse(is.na(as.numeric(R12DCBAL3)), 0, as.numeric(R12DCBAL3))) 
%>%
  mutate(R13DCBAL3=ifelse(is.na(as.numeric(R13DCBAL3)), 0, as.numeric(R13DCBAL3))) 
%>%
  mutate(R14DCBAL3=ifelse(is.na(as.numeric(R14DCBAL3)), 0, as.numeric(R14DCBAL3))) 
%>%
  mutate(R15DCBAL3=ifelse(is.na(as.numeric(R15DCBAL3)), 0, as.numeric(R15DCBAL3))) 
%>%
  ## RwDCBAL4
  mutate(R5DCBAL4=ifelse(is.na(as.numeric(R5DCBAL4)), 0, as.numeric(R5DCBAL4))) %>%
  mutate(R6DCBAL4=ifelse(is.na(as.numeric(R6DCBAL4)), 0, as.numeric(R6DCBAL4))) %>%
  mutate(R7DCBAL4=ifelse(is.na(as.numeric(R7DCBAL4)), 0, as.numeric(R7DCBAL4))) %>%
  mutate(R8DCBAL4=ifelse(is.na(as.numeric(R8DCBAL4)), 0, as.numeric(R8DCBAL4))) %>%
  mutate(R9DCBAL4=ifelse(is.na(as.numeric(R9DCBAL4)), 0, as.numeric(R9DCBAL4))) %>%
  mutate(R10DCBAL4=ifelse(is.na(as.numeric(R10DCBAL4)), 0, as.numeric(R10DCBAL4)))
%>%
  mutate(R11DCBAL4=ifelse(is.na(as.numeric(R11DCBAL4)), 0, as.numeric(R11DCBAL4)))
%>%
  mutate(R12DCBAL4=ifelse(is.na(as.numeric(R12DCBAL4)), 0, as.numeric(R12DCBAL4)))
%>%
  mutate(R13DCBAL4=ifelse(is.na(as.numeric(R13DCBAL4)), 0, as.numeric(R13DCBAL4)))
%>%
  mutate(R14DCBAL4=ifelse(is.na(as.numeric(R14DCBAL4)), 0, as.numeric(R14DCBAL4)))
%>%
  mutate(R15DCBAL4=ifelse(is.na(as.numeric(R15DCBAL4)), 0, as.numeric(R15DCBAL4)))
%>%
  ## SwDCBAL1
  mutate(S4DCBAL1=ifelse(is.na(as.numeric(S4DCBAL1)), 0, as.numeric(S4DCBAL1))) %>%
  mutate(S5DCBAL1=ifelse(is.na(as.numeric(S5DCBAL1)), 0, as.numeric(S5DCBAL1))) %>%
  mutate(S6DCBAL1=ifelse(is.na(as.numeric(S6DCBAL1)), 0, as.numeric(S6DCBAL1))) %>%
  mutate(S7DCBAL1=ifelse(is.na(as.numeric(S7DCBAL1)), 0, as.numeric(S7DCBAL1))) %>%
  mutate(S8DCBAL1=ifelse(is.na(as.numeric(S8DCBAL1)), 0, as.numeric(S8DCBAL1))) %>%
  mutate(S9DCBAL1=ifelse(is.na(as.numeric(S9DCBAL1)), 0, as.numeric(S9DCBAL1))) %>%
  mutate(S10DCBAL1=ifelse(is.na(as.numeric(S10DCBAL1)), 0, as.numeric(S10DCBAL1)))
%>%
  mutate(S11DCBAL1=ifelse(is.na(as.numeric(S11DCBAL1)), 0, as.numeric(S11DCBAL1)))
%>%
  mutate(S12DCBAL1=ifelse(is.na(as.numeric(S12DCBAL1)), 0, as.numeric(S12DCBAL1)))
%>%
  mutate(S13DCBAL1=ifelse(is.na(as.numeric(S13DCBAL1)), 0, as.numeric(S13DCBAL1)))
%>%
  mutate(S14DCBAL1=ifelse(is.na(as.numeric(S14DCBAL1)), 0, as.numeric(S14DCBAL1)))
%>%
  mutate(S15DCBAL1=ifelse(is.na(as.numeric(S15DCBAL1)), 0, as.numeric(S15DCBAL1)))
%>%
  ## SwDCBAL2
  mutate(S4DCBAL2=ifelse(is.na(as.numeric(S4DCBAL2)), 0, as.numeric(S4DCBAL2))) %>%
  mutate(S5DCBAL2=ifelse(is.na(as.numeric(S5DCBAL2)), 0, as.numeric(S5DCBAL2))) %>%
  mutate(S6DCBAL2=ifelse(is.na(as.numeric(S6DCBAL2)), 0, as.numeric(S6DCBAL2))) %>%
  mutate(S7DCBAL2=ifelse(is.na(as.numeric(S7DCBAL2)), 0, as.numeric(S7DCBAL2))) %>%
  mutate(S8DCBAL2=ifelse(is.na(as.numeric(S8DCBAL2)), 0, as.numeric(S8DCBAL2))) %>%
  mutate(S9DCBAL2=ifelse(is.na(as.numeric(S9DCBAL2)), 0, as.numeric(S9DCBAL2))) %>%
  mutate(S10DCBAL2=ifelse(is.na(as.numeric(S10DCBAL2)), 0, as.numeric(S10DCBAL2)))
%>%
  mutate(S11DCBAL2=ifelse(is.na(as.numeric(S11DCBAL2)), 0, as.numeric(S11DCBAL2)))
%>%
  mutate(S12DCBAL2=ifelse(is.na(as.numeric(S12DCBAL2)), 0, as.numeric(S12DCBAL2)))
%>%
  mutate(S13DCBAL2=ifelse(is.na(as.numeric(S13DCBAL2)), 0, as.numeric(S13DCBAL2)))
%>%
  mutate(S14DCBAL2=ifelse(is.na(as.numeric(S14DCBAL2)), 0, as.numeric(S14DCBAL2)))
%>%
  mutate(S15DCBAL2=ifelse(is.na(as.numeric(S15DCBAL2)), 0, as.numeric(S15DCBAL2)))
%>%
  ## SwDCBAL3
  mutate(S4DCBAL3=ifelse(is.na(as.numeric(S4DCBAL3)), 0, as.numeric(S4DCBAL3))) %>%
  mutate(S5DCBAL3=ifelse(is.na(as.numeric(S5DCBAL3)), 0, as.numeric(S5DCBAL3))) %>%
  mutate(S6DCBAL3=ifelse(is.na(as.numeric(S6DCBAL3)), 0, as.numeric(S6DCBAL3))) %>%
  mutate(S7DCBAL3=ifelse(is.na(as.numeric(S7DCBAL3)), 0, as.numeric(S7DCBAL3))) %>%
  mutate(S8DCBAL3=ifelse(is.na(as.numeric(S8DCBAL3)), 0, as.numeric(S8DCBAL3))) %>%
  mutate(S9DCBAL3=ifelse(is.na(as.numeric(S9DCBAL3)), 0, as.numeric(S9DCBAL3))) %>%
  mutate(S10DCBAL3=ifelse(is.na(as.numeric(S10DCBAL3)), 0, as.numeric(S10DCBAL3)))
%>%
  mutate(S11DCBAL3=ifelse(is.na(as.numeric(S11DCBAL3)), 0, as.numeric(S11DCBAL3)))
%>%
  mutate(S12DCBAL3=ifelse(is.na(as.numeric(S12DCBAL3)), 0, as.numeric(S12DCBAL3)))
%>%
  mutate(S13DCBAL3=ifelse(is.na(as.numeric(S13DCBAL3)), 0, as.numeric(S13DCBAL3)))
%>%
  mutate(S14DCBAL3=ifelse(is.na(as.numeric(S14DCBAL3)), 0, as.numeric(S14DCBAL3)))
%>%
  mutate(S15DCBAL3=ifelse(is.na(as.numeric(S15DCBAL3)), 0, as.numeric(S15DCBAL3)))
%>%
  ## SwDCBAL4
  mutate(S5DCBAL4=ifelse(is.na(as.numeric(S5DCBAL4)), 0, as.numeric(S5DCBAL4))) %>%
  mutate(S6DCBAL4=ifelse(is.na(as.numeric(S6DCBAL4)), 0, as.numeric(S6DCBAL4))) %>%
  mutate(S7DCBAL4=ifelse(is.na(as.numeric(S7DCBAL4)), 0, as.numeric(S7DCBAL4))) %>%
  mutate(S8DCBAL4=ifelse(is.na(as.numeric(S8DCBAL4)), 0, as.numeric(S8DCBAL4))) %>%
  mutate(S9DCBAL4=ifelse(is.na(as.numeric(S9DCBAL4)), 0, as.numeric(S9DCBAL4))) %>%
  mutate(S10DCBAL4=ifelse(is.na(as.numeric(S10DCBAL4)), 0, as.numeric(S10DCBAL4)))
%>%
  mutate(S11DCBAL4=ifelse(is.na(as.numeric(S11DCBAL4)), 0, as.numeric(S11DCBAL4)))
%>%
  mutate(S12DCBAL4=ifelse(is.na(as.numeric(S12DCBAL4)), 0, as.numeric(S12DCBAL4)))
%>%
  mutate(S13DCBAL4=ifelse(is.na(as.numeric(S13DCBAL4)), 0, as.numeric(S13DCBAL4)))
%>%
  mutate(S14DCBAL4=ifelse(is.na(as.numeric(S14DCBAL4)), 0, as.numeric(S14DCBAL4)))
%>%
  mutate(S15DCBAL4=ifelse(is.na(as.numeric(S15DCBAL4)), 0, as.numeric(S15DCBAL4)))
%>%
  ## HwAIRA
  mutate(H4AIRA=ifelse(is.na(as.numeric(H4AIRA)), 0, as.numeric(H4AIRA))) %>%
  mutate(H5AIRA=ifelse(is.na(as.numeric(H5AIRA)), 0, as.numeric(H5AIRA))) %>%
  mutate(H6AIRA=ifelse(is.na(as.numeric(H6AIRA)), 0, as.numeric(H6AIRA))) %>%
  mutate(H7AIRA=ifelse(is.na(as.numeric(H7AIRA)), 0, as.numeric(H7AIRA))) %>%
  mutate(H8AIRA=ifelse(is.na(as.numeric(H8AIRA)), 0, as.numeric(H8AIRA))) %>%
  mutate(H9AIRA=ifelse(is.na(as.numeric(H9AIRA)), 0, as.numeric(H9AIRA))) %>%
  mutate(H10AIRA=ifelse(is.na(as.numeric(H10AIRA)), 0, as.numeric(H10AIRA))) %>%
  mutate(H11AIRA=ifelse(is.na(as.numeric(H11AIRA)), 0, as.numeric(H11AIRA))) %>%
  mutate(H12AIRA=ifelse(is.na(as.numeric(H12AIRA)), 0, as.numeric(H12AIRA))) %>%
  mutate(H13AIRA=ifelse(is.na(as.numeric(H13AIRA)), 0, as.numeric(H13AIRA))) %>%
  mutate(H14AIRA=ifelse(is.na(as.numeric(H14AIRA)), 0, as.numeric(H14AIRA))) %>%
  mutate(H15AIRA=ifelse(is.na(as.numeric(H15AIRA)), 0, as.numeric(H15AIRA))) %>%
  ## HwASTCK
  mutate(H4ASTCK=ifelse(is.na(as.numeric(H4ASTCK)), 0, as.numeric(H4ASTCK))) %>%
  mutate(H5ASTCK=ifelse(is.na(as.numeric(H5ASTCK)), 0, as.numeric(H5ASTCK))) %>%
  mutate(H6ASTCK=ifelse(is.na(as.numeric(H6ASTCK)), 0, as.numeric(H6ASTCK))) %>%
  mutate(H7ASTCK=ifelse(is.na(as.numeric(H7ASTCK)), 0, as.numeric(H7ASTCK))) %>%
  mutate(H8ASTCK=ifelse(is.na(as.numeric(H8ASTCK)), 0, as.numeric(H8ASTCK))) %>%
mutate(H9ASTCK=ifelse(is.na(as.numeric(H9ASTCK)), 0, as.numeric(H9ASTCK))) %>%
mutate(H10ASTCK=ifelse(is.na(as.numeric(H10ASTCK)), 0, as.numeric(H10ASTCK))) %>%
mutate(H11ASTCK=ifelse(is.na(as.numeric(H11ASTCK)), 0, as.numeric(H11ASTCK))) %>%
mutate(H12ASTCK=ifelse(is.na(as.numeric(H12ASTCK)), 0, as.numeric(H12ASTCK))) %>%
mutate(H13ASTCK=ifelse(is.na(as.numeric(H13ASTCK)), 0, as.numeric(H13ASTCK))) %>%
mutate(H14ASTCK=ifelse(is.na(as.numeric(H14ASTCK)), 0, as.numeric(H14ASTCK))) %>%
mutate(H15ASTCK=ifelse(is.na(as.numeric(H15ASTCK)), 0, as.numeric(H15ASTCK))) %>%
## HwACHCK
mutate(H4ACHCK=ifelse(is.na(as.numeric(H4ACHCK)), 0, as.numeric(H4ACHCK))) %>%
mutate(H5ACHCK=ifelse(is.na(as.numeric(H5ACHCK)), 0, as.numeric(H5ACHCK))) %>%
mutate(H6ACHCK=ifelse(is.na(as.numeric(H6ACHCK)), 0, as.numeric(H6ACHCK))) %>%
mutate(H7ACHCK=ifelse(is.na(as.numeric(H7ACHCK)), 0, as.numeric(H7ACHCK))) %>%
mutate(H8ACHCK=ifelse(is.na(as.numeric(H8ACHCK)), 0, as.numeric(H8ACHCK))) %>%
mutate(H9ACHCK=ifelse(is.na(as.numeric(H9ACHCK)), 0, as.numeric(H9ACHCK))) %>%
mutate(H10ACHCK=ifelse(is.na(as.numeric(H10ACHCK)), 0, as.numeric(H10ACHCK))) %>%
mutate(H11ACHCK=ifelse(is.na(as.numeric(H11ACHCK)), 0, as.numeric(H11ACHCK))) %>%
mutate(H12ACHCK=ifelse(is.na(as.numeric(H12ACHCK)), 0, as.numeric(H12ACHCK))) %>%
mutate(H13ACHCK=ifelse(is.na(as.numeric(H13ACHCK)), 0, as.numeric(H13ACHCK))) %>%
mutate(H14ACHCK=ifelse(is.na(as.numeric(H14ACHCK)), 0, as.numeric(H14ACHCK))) %>%
mutate(H15ACHCK=ifelse(is.na(as.numeric(H15ACHCK)), 0, as.numeric(H15ACHCK))) %>%
## HwACD
mutate(H4ACD=ifelse(is.na(as.numeric(H4ACD)), 0, as.numeric(H4ACD))) %>%
mutate(H5ACD=ifelse(is.na(as.numeric(H5ACD)), 0, as.numeric(H5ACD))) %>%
mutate(H6ACD=ifelse(is.na(as.numeric(H6ACD)), 0, as.numeric(H6ACD))) %>%
mutate(H7ACD=ifelse(is.na(as.numeric(H7ACD)), 0, as.numeric(H7ACD))) %>%
mutate(H8ACD=ifelse(is.na(as.numeric(H8ACD)), 0, as.numeric(H8ACD))) %>%
mutate(H9ACD=ifelse(is.na(as.numeric(H9ACD)), 0, as.numeric(H9ACD))) %>%
mutate(H10ACD=ifelse(is.na(as.numeric(H10ACD)), 0, as.numeric(H10ACD))) %>%
mutate(H11ACD=ifelse(is.na(as.numeric(H11ACD)), 0, as.numeric(H11ACD))) %>%
mutate(H12ACD=ifelse(is.na(as.numeric(H12ACD)), 0, as.numeric(H12ACD))) %>%
mutate(H13ACD=ifelse(is.na(as.numeric(H13ACD)), 0, as.numeric(H13ACD))) %>%
mutate(H14ACD=ifelse(is.na(as.numeric(H14ACD)), 0, as.numeric(H14ACD))) %>%
mutate(H15ACD=ifelse(is.na(as.numeric(H15ACD)), 0, as.numeric(H15ACD))) %>%
# HwABOND
mutate(H4ABOND=ifelse(is.na(as.numeric(H4ABOND)), 0, as.numeric(H4ABOND))) %>%
mutate(H5ABOND=ifelse(is.na(as.numeric(H5ABOND)), 0, as.numeric(H5ABOND))) %>%
mutate(H6ABOND=ifelse(is.na(as.numeric(H6ABOND)), 0, as.numeric(H6ABOND))) %>%
mutate(H7ABOND=ifelse(is.na(as.numeric(H7ABOND)), 0, as.numeric(H7ABOND))) %>%
mutate(H8ABOND=ifelse(is.na(as.numeric(H8ABOND)), 0, as.numeric(H8ABOND))) %>%
mutate(H9ABOND=ifelse(is.na(as.numeric(H9ABOND)), 0, as.numeric(H9ABOND))) %>%
mutate(H10ABOND=ifelse(is.na(as.numeric(H10ABOND)), 0, as.numeric(H10ABOND))) %>%
mutate(H11ABOND=ifelse(is.na(as.numeric(H11ABOND)), 0, as.numeric(H11ABOND))) %>%
mutate(H12ABOND=ifelse(is.na(as.numeric(H12ABOND)), 0, as.numeric(H12ABOND))) %>%
mutate(H13ABOND=ifelse(is.na(as.numeric(H13ABOND)), 0, as.numeric(H13ABOND))) %>%
mutate(H14ABOND=ifelse(is.na(as.numeric(H14ABOND)), 0, as.numeric(H14ABOND))) %>%
mutate(H15ABOND=ifelse(is.na(as.numeric(H15ABOND)), 0, as.numeric(H15ABOND))) %>%
# HwAOTHR
mutate(H4AOTHR=ifelse(is.na(as.numeric(H4AOTHR)), 0, as.numeric(H4AOTHR)))%>%
mutate(H5AOTHR=ifelse(is.na(as.numeric(H5AOTHR)), 0, as.numeric(H5AOTHR))) %>%
mutate(H6AOTHR=ifelse(is.na(as.numeric(H6AOTHR)), 0, as.numeric(H6AOTHR))) %>%
mutate(H7AOTHR=ifelse(is.na(as.numeric(H7AOTHR)), 0, as.numeric(H7AOTHR))) %>%
mutate(H8AOTHR=ifelse(is.na(as.numeric(H8AOTHR)), 0, as.numeric(H8AOTHR))) %>%
mutate(H9AOTHR=ifelse(is.na(as.numeric(H9AOTHR)), 0, as.numeric(H9AOTHR))) %>%
mutate(H10AOTHR=ifelse(is.na(as.numeric(H10AOTHR)), 0, as.numeric(H10AOTHR))) %>%
  mutate(H11AOTHR=ifelse(is.na(as.numeric(H11AOTHR)), 0, as.numeric(H11AOTHR))) %>%
  mutate(H12AOTHR=ifelse(is.na(as.numeric(H12AOTHR)), 0, as.numeric(H12AOTHR))) %>%
  mutate(H13AOTHR=ifelse(is.na(as.numeric(H13AOTHR)), 0, as.numeric(H13AOTHR))) %>%
  mutate(H14AOTHR=ifelse(is.na(as.numeric(H14AOTHR)), 0, as.numeric(H14AOTHR))) %>%
  mutate(H15AOTHR=ifelse(is.na(as.numeric(H15AOTHR)), 0, as.numeric(H15AOTHR))) %>%
  # HwADBET
  mutate(H4ADEBT=ifelse(is.na(as.numeric(H4ADEBT)), 0, as.numeric(H4ADEBT))) %>%
  mutate(H5ADEBT=ifelse(is.na(as.numeric(H5ADEBT)), 0, as.numeric(H5ADEBT))) %>%
  mutate(H6ADEBT=ifelse(is.na(as.numeric(H6ADEBT)), 0, as.numeric(H6ADEBT))) %>%
  mutate(H7ADEBT=ifelse(is.na(as.numeric(H7ADEBT)), 0, as.numeric(H7ADEBT))) %>%
  mutate(H8ADEBT=ifelse(is.na(as.numeric(H8ADEBT)), 0, as.numeric(H8ADEBT))) %>%
  mutate(H9ADEBT=ifelse(is.na(as.numeric(H9ADEBT)), 0, as.numeric(H9ADEBT))) %>%
  mutate(H10ADEBT=ifelse(is.na(as.numeric(H10ADEBT)), 0, as.numeric(H10ADEBT))) %>%
  mutate(H11ADEBT=ifelse(is.na(as.numeric(H11ADEBT)), 0, as.numeric(H11ADEBT))) %>%
  mutate(H12ADEBT=ifelse(is.na(as.numeric(H12ADEBT)), 0, as.numeric(H12ADEBT))) %>%
  mutate(H13ADEBT=ifelse(is.na(as.numeric(H13ADEBT)), 0, as.numeric(H13ADEBT))) %>%
  mutate(H14ADEBT=ifelse(is.na(as.numeric(H14ADEBT)), 0, as.numeric(H14ADEBT))) %>%
  mutate(H15ADEBT=ifelse(is.na(as.numeric(H15ADEBT)), 0, as.numeric(H15ADEBT))) %>%
  # Construct liquid assets variables
  mutate(H4LIQA=R4DCBAL1 + R4DCBAL2 + R4DCBAL3 + S4DCBAL1 + S4DCBAL2 +
S4DCBAL3 + H4AIRA + H4ASTCK + H4ACHCK + H4ACD + H4ABOND + H4AOTHR -
H4ADEBT) %>%
  mutate(H5LIQA=R5DCBAL1 + R5DCBAL2 + R5DCBAL3 + S5DCBAL1 + S5DCBAL2 +
S5DCBAL3 + H5AIRA + H5ASTCK + H5ACHCK + H5ACD + H5ABOND + H5AOTHR -
H5ADEBT) %>%
  mutate(H6LIQA=R6DCBAL1 + R6DCBAL2 + R6DCBAL3 + S6DCBAL1 + S6DCBAL2 +
S6DCBAL3 + H6AIRA + H6ASTCK + H6ACHCK + H6ACD + H6ABOND + H6AOTHR -
H6ADEBT) %>%
  mutate(H7LIQA=R7DCBAL1 + R7DCBAL2 + R7DCBAL3 + S7DCBAL1 + S7DCBAL2 +
S7DCBAL3 + H7AIRA + H7ASTCK + H7ACHCK + H7ACD + H7ABOND + H7AOTHR -
H7ADEBT) %>%
  mutate(H8LIQA=R8DCBAL1 + R8DCBAL2 + R8DCBAL3 + S8DCBAL1 + S8DCBAL2 +
S8DCBAL3 + H8AIRA + H8ASTCK + H8ACHCK + H8ACD + H8ABOND + H8AOTHR -
H8ADEBT) %>%
  mutate(H9LIQA=R9DCBAL1 + R9DCBAL2 + R9DCBAL3 + S9DCBAL1 + S9DCBAL2 +
S9DCBAL3 + H9AIRA + H9ASTCK + H9ACHCK + H9ACD + H9ABOND + H9AOTHR -
H9ADEBT) %>%
  mutate(H10LIQA=R10DCBAL1 + R10DCBAL2 + R10DCBAL3 + S10DCBAL1 + S10DCBAL2
+ S10DCBAL3 + H10AIRA + H10ASTCK + H10ACHCK + H10ACD + H10ABOND +
H10AOTHR - H10ADEBT) %>%
  mutate(H11LIQA=R11DCBAL1 + R11DCBAL2 + R11DCBAL3 + S11DCBAL1 + S11DCBAL2 +
S11DCBAL3 + H11AIRA + H11ASTCK + H11ACHCK + H11ACD + H11ABOND + H11AOTHR -
H11ADEBT) %>%
  mutate(H12LIQA=R12DCBAL1 + R12DCBAL2 + R12DCBAL3 + S12DCBAL1 + S12DCBAL2
+ S12DCBAL3 + H12AIRA + H12ASTCK + H12ACHCK + H12ACD + H12ABOND +
H12AOTHR - H12ADEBT) %>%
  mutate(H13LIQA=R13DCBAL1 + R13DCBAL2 + R13DCBAL3 + S13DCBAL1 + S13DCBAL2
+ S13DCBAL3 + H13AIRA + H13ASTCK + H13ACHCK + H13ACD + H13ABOND +
H13AOTHR - H13ADEBT) %>%
  mutate(H14LIQA=R14DCBAL1 + R14DCBAL2 + R14DCBAL3 + S14DCBAL1 + S14DCBAL2
+ S14DCBAL3 + H14AIRA + H14ASTCK + H14ACHCK + H14ACD + H14ABOND +
H14AOTHR - H14ADEBT) %>%
  mutate(H15LIQA=R15DCBAL1 + R15DCBAL2 + R15DCBAL3 + S15DCBAL1 + S15DCBAL2
+ S15DCBAL3 + H15AIRA + H15ASTCK + H15ACHCK + H15ACD + H15ABOND +
H15AOTHR - H15ADEBT)

# Drop un-needed variables
hrs <- select(hrs, -c("R4DCBAL1", "R4DCBAL2", "R4DCBAL3", "S4DCBAL1", "S4DCBAL2",
"S4DCBAL3", "H4AIRA", "H4ASTCK", "H4ACHCK", "H4ACD", "H4ABOND", "H4AOTHR",
"H4ADEBT")) %>%
  select(-c("R5DCBAL1", "R5DCBAL2", "R5DCBAL3", "R5DCBAL4", "S5DCBAL1",
"S5DCBAL2", "S5DCBAL3", "S5DCBAL4", "H5AIRA", "H5ASTCK", "H5ACHCK", "H5ACD",
"H5ABOND", "H5AOTHR", "H5ADEBT")) %>%
  select(-c("R6DCBAL1", "R6DCBAL2", "R6DCBAL3", "R6DCBAL4", "S6DCBAL1",
"S6DCBAL2", "S6DCBAL3", "S6DCBAL4", "H6AIRA", "H6ASTCK", "H6ACHCK", "H6ACD",
"H6ABOND", "H6AOTHR", "H6ADEBT")) %>%
  select(-c("R7DCBAL1", "R7DCBAL2", "R7DCBAL3", "R7DCBAL4", "S7DCBAL1",
"S7DCBAL2", "S7DCBAL3", "S7DCBAL4", "H7AIRA", "H7ASTCK", "H7ACHCK", "H7ACD",
"H7ABOND", "H7AOTHR", "H7ADEBT")) %>%
  select(-c("R8DCBAL1", "R8DCBAL2", "R8DCBAL3", "R8DCBAL4", "S8DCBAL1",
"S8DCBAL2", "S8DCBAL3", "S8DCBAL4", "H8AIRA", "H8ASTCK", "H8ACHCK", "H8ACD",
"H8ABOND", "H8AOTHR", "H8ADEBT")) %>%
  select(-c("R9DCBAL1", "R9DCBAL2", "R9DCBAL3", "R9DCBAL4", "S9DCBAL1",
"S9DCBAL2", "S9DCBAL3", "S9DCBAL4", "H9AIRA", "H9ASTCK", "H9ACHCK", "H9ACD",
"H9ABOND", "H9AOTHR", "H9ADEBT")) %>%
  select(-c("R10DCBAL1", "R10DCBAL2", "R10DCBAL3", "R10DCBAL4", "S10DCBAL1",
"S10DCBAL2", "S10DCBAL3", "S10DCBAL4", "H10AIRA", "H10ASTCK", "H10ACHCK",
"H10ACD", "H10ABOND", "H10AOTHR", "H10ADEBT")) %>%
  select(-c("R11DCBAL1", "R11DCBAL2", "R11DCBAL3", "R11DCBAL4", "S11DCBAL1",
"S11DCBAL2", "S11DCBAL3", "S11DCBAL4", "H11AIRA", "H11ASTCK", "H11ACHCK",
"H11ACD", "H11ABOND", "H11AOTHR", "H11ADEBT")) %>%
  select(-c("R12DCBAL1", "R12DCBAL2", "R12DCBAL3", "R12DCBAL4", "S12DCBAL1",
"S12DCBAL2", "S12DCBAL3", "S12DCBAL4", "H12AIRA", "H12ASTCK", "H12ACHCK",
"H12ACD", "H12ABOND", "H12AOTHR", "H12ADEBT")) %>%
  select(-c("R13DCBAL1", "R13DCBAL2", "R13DCBAL3", "R13DCBAL4", "S13DCBAL1",
"S13DCBAL2", "S13DCBAL3", "S13DCBAL4", "H13AIRA", "H13ASTCK", "H13ACHCK",
"H13ACD", "H13ABOND", "H13AOTHR", "H13ADEBT")) %>%
  select(-c("R14DCBAL1", "R14DCBAL2", "R14DCBAL3", "R14DCBAL4", "S14DCBAL1",
"S14DCBAL2", "S14DCBAL3", "S14DCBAL4", "H14AIRA", "H14ASTCK", "H14ACHCK",
"H14ACD", "H14ABOND", "H14AOTHR", "H14ADEBT")) %>%
  select(-c("R15DCBAL1", "R15DCBAL2", "R15DCBAL3", "R15DCBAL4", "S15DCBAL1",
"S15DCBAL2", "S15DCBAL3", "S15DCBAL4", "H15AIRA", "H15ASTCK", "H15ACHCK",
"H15ACD", "H15ABOND", "H15AOTHR", "H15ADEBT"))

# To real dollars: OOPMD
hrs <- mutate(hrs, R4OOPMD=adjust_for_inflation(R4OOPMD, 1998, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R5OOPMD=adjust_for_inflation(R5OOPMD, 2000, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R6OOPMD=adjust_for_inflation(R6OOPMD, 2002, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R7OOPMD=adjust_for_inflation(R7OOPMD, 2004, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R8OOPMD=adjust_for_inflation(R8OOPMD, 2006, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R9OOPMD=adjust_for_inflation(R9OOPMD, 2008, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R10OOPMD=adjust_for_inflation(R10OOPMD, 2010, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R11OOPMD=adjust_for_inflation(R11OOPMD, 2012, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R12OOPMD=adjust_for_inflation(R12OOPMD, 2014, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R13OOPMD=adjust_for_inflation(R13OOPMD, 2016, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R14OOPMD=adjust_for_inflation(R14OOPMD, 2018, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  # To real dollars: WEALTH
  mutate(H4LIQA=adjust_for_inflation(H4LIQA, 1998, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H5LIQA=adjust_for_inflation(H5LIQA, 2000, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H6LIQA=adjust_for_inflation(H6LIQA, 2002, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H7LIQA=adjust_for_inflation(H7LIQA, 2004, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H8LIQA=adjust_for_inflation(H8LIQA, 2006, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H9LIQA=adjust_for_inflation(H9LIQA, 2008, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H10LIQA=adjust_for_inflation(H10LIQA, 2010, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H11LIQA=adjust_for_inflation(H11LIQA, 2012, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H12LIQA=adjust_for_inflation(H12LIQA, 2014, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H13LIQA=adjust_for_inflation(H13LIQA, 2016, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(H14LIQA=adjust_for_inflation(H14LIQA, 2018, country, to_date = 2020,
inflation_dataframe = inflation_dataframe, countries_dataframe = countries_dataframe)) %>%
  mutate(R15WTRESP = rep(NA, 42405)) %>%
  select(-c("H3ATOTF", "H4ATOTF", "H5ATOTF", "H6ATOTF", "H7ATOTF", "H8ATOTF",
"H9ATOTF", "H10ATOTF", "H11ATOTF", "H12ATOTF", "H13ATOTF", "H14ATOTF",
"H15ATOTF"))

# Replace HRS missing codes with NA - categorical variables
cat <- c("R4OUTPT", "R5OUTPT", "R6OUTPT", "R7OUTPT", "R8OUTPT", "R9OUTPT",
"R10OUTPT", "R11OUTPT", "R12OUTPT", "R13OUTPT", "R14OUTPT", "R15OUTPT",
"R4DRUGS", "R5DRUGS", "R6DRUGS", "R7DRUGS", "R8DRUGS", "R9DRUGS",
"R10DRUGS", "R11DRUGS", "R12DRUGS", "R13DRUGS", "R14DRUGS", "R15DRUGS",
"R4HOMCAR", "R5HOMCAR", "R6HOMCAR", "R7HOMCAR", "R8HOMCAR", "R9HOMCAR",
"R10HOMCAR", "R11HOMCAR", "R12HOMCAR", "R13HOMCAR", "R14HOMCAR",
"R15HOMCAR", "R4GOVMR", "R5GOVMR", "R6GOVMR", "R7GOVMR", "R8GOVMR",
"R9GOVMR", "R10GOVMR", "R11GOVMR", "R12GOVMR", "R13GOVMR", "R14GOVMR",
"R15GOVMR", "R4DIABS", "R5DIABS", "R6DIABS", "R7DIABS", "R8DIABS", "R9DIABS",
"R10DIABS", "R11DIABS", "R12DIABS", "R13DIABS", "R14DIABS", "R15DIABS", "R4HIBPS",
"R5HIBPS", "R6HIBPS", "R7HIBPS", "R8HIBPS", "R9HIBPS", "R10HIBPS", "R11HIBPS",
"R12HIBPS", "R13HIBPS", "R14HIBPS", "R15HIBPS", "R4CANCRS", "R5CANCRS",
"R6CANCRS", "R7CANCRS", "R8CANCRS", "R9CANCRS", "R10CANCRS", "R11CANCRS",
"R12CANCRS", "R13CANCRS", "R14CANCRS", "R15CANCRS", "R4HEARTS", "R5HEARTS",
"R6HEARTS", "R7HEARTS", "R8HEARTS", "R9HEARTS", "R10HEARTS", "R11HEARTS",
"R12HEARTS", "R13HEARTS", "R14HEARTS", "R15HEARTS", "R4LUNGS", "R5LUNGS",
"R6LUNGS", "R7LUNGS", "R8LUNGS", "R9LUNGS", "R10LUNGS", "R11LUNGS",
"R12LUNGS", "R13LUNGS", "R14LUNGS", "R15LUNGS", "R4ARTHRS", "R5ARTHRS",
"R6ARTHRS", "R7ARTHRS", "R8ARTHRS", "R9ARTHRS", "R10ARTHRS", "R11ARTHRS",
"R12ARTHRS", "R13ARTHRS", "R14ARTHRS", "R15ARTHRS", "R4STROKS", "R5STROKS",
"R6STROKS", "R7STROKS", "R8STROKS", "R9STROKS", "R10STROKS", "R11STROKS",
"R12STROKS", "R13STROKS", "R14STROKS", "R15STROKS", "R4PSYCHS", "R5PSYCHS",
"R6PSYCHS", "R7PSYCHS", "R8PSYCHS", "R9PSYCHS", "R10PSYCHS", "R11PSYCHS",
"R12PSYCHS", "R13PSYCHS", "R14PSYCHS", "R15PSYCHS")
hrs <- mutate(hrs, across(all_of(cat), ~ifelse(.x %in% c("1.Yes", "0.No"), .x, NA)))
# Replace HRS missing codes with NA and make numeric - numeric variables
num <- c("R4SHLTC", "R5SHLTC", "R6SHLTC", "R7SHLTC", "R8SHLTC", "R9SHLTC",
"R10SHLTC", "R11SHLTC", "R12SHLTC", "R13SHLTC", "R14SHLTC", "R15SHLTC",
"R4HSPNIT", "R5HSPNIT", "R6HSPNIT", "R7HSPNIT", "R8HSPNIT", "R9HSPNIT",
"R10HSPNIT", "R11HSPNIT", "R12HSPNIT", "R13HSPNIT", "R14HSPNIT", "R15HSPNIT",
"R4NRSNIT", "R5NRSNIT", "R6NRSNIT", "R7NRSNIT", "R8NRSNIT", "R9NRSNIT",
"R10NRSNIT", "R11NRSNIT", "R12NRSNIT", "R13NRSNIT", "R14NRSNIT", "R15NRSNIT",
"R4DOCTIM", "R5DOCTIM", "R6DOCTIM", "R7DOCTIM", "R8DOCTIM", "R9DOCTIM",
"R10DOCTIM", "R11DOCTIM", "R12DOCTIM", "R13DOCTIM", "R14DOCTIM", "R15DOCTIM")
hrs <- mutate(hrs, across(all_of(num), ~ifelse(.x %in% c("-1", "-2", "-3", "-4", "-5", "0", "1", "2",
"3", "4", "5"), .x, NA))) %>%
  mutate(across(all_of(num), as.numeric))

vars <- colnames(hrs)
# Construct AVGSSI = avg(non-zero values of SSI across all waves)
x <- hrs[,grep("ISRET", vars)]
AVGSSI <- apply(x,1,function(x){mean(x[x>0], na.rm=TRUE)})
hrs <- cbind(hrs, AVGSSI)
# Drop unneeded ISRET variables and accidentally includes CONDS variables
hrs <- select(hrs, -c("R4ISRET", "R5ISRET", "R6ISRET", "R7ISRET", "R8ISRET", "R9ISRET",
"R10ISRET", "R11ISRET", ,"R12ISRET", "R13ISRET", "R14ISRET", "R15ISRET", "R8CONDS",
"R9CONDS"))

# Construct wealth/SSI income-adjusted spending variables (log-transformation)
# RwMDLIQ = RwOOPMD/HwLIQA - scaled by previous wave
hrs <- mutate(hrs, "H5MDLIQ" = R5OOPMD/H4LIQA) %>%
  mutate("R6MDLIQ" = log(R6OOPMD/H5LIQA)) %>%
  mutate("R7MDLIQ" = log(R7OOPMD/H6LIQA)) %>%
  mutate("R8MDLIQ" = log(R8OOPMD/H7LIQA)) %>%
  mutate("R9MDLIQ" = log(R9OOPMD/H8LIQA)) %>%
  mutate("R10MDLIQ" = log(R10OOPMD/H9LIQA)) %>%
  mutate("R11MDLIQ" = log(R11OOPMD/H10LIQA)) %>%
  mutate("R12MDLIQ" = log(R12OOPMD/H11LIQA)) %>%
  mutate("R13MDLIQ" = log(R13OOPMD/H12LIQA)) %>%
  mutate("R14MDLIQ" = log(R14OOPMD/H13LIQA)) %>%
  mutate("R15MDLIQ" = log(R15OOPMD/H14LIQA)) %>%
  mutate("R4MDLIQ" = rep(NA, 42405)) %>%
  # RwMDSSI = RwOOPMD/AVGSSI
  mutate("R4MDSSI" = log(R4OOPMD/AVGSSI)) %>%
  mutate("R5MDSSI" = log(R5OOPMD/AVGSSI)) %>%
  mutate("R6MDSSI" = log(R6OOPMD/AVGSSI)) %>%
  mutate("R7MDSSI" = log(R7OOPMD/AVGSSI)) %>%
  mutate("R8MDSSI" = log(R8OOPMD/AVGSSI)) %>%
  mutate("R9MDSSI" = log(R9OOPMD/AVGSSI)) %>%
  mutate("R10MDSSI" = log(R10OOPMD/AVGSSI)) %>%
  mutate("R11MDSSI" = log(R11OOPMD/AVGSSI)) %>%
  mutate("R12MDSSI" = log(R12OOPMD/AVGSSI)) %>%
  mutate("R13MDSSI" = log(R13OOPMD/AVGSSI)) %>%
  mutate("R14MDSSI" = log(R14OOPMD/AVGSSI)) %>%
  mutate("R15MDSSI" = log(R15OOPMD/AVGSSI)) %>%
  # Drop AVGSSI (we will keep LIQ variables for $1000 criteria and remove them later)
  select(-c("AVGSSI")) %>%
  # Try dropping doctor
  select(-c("R4DOCTIM", "R5DOCTIM", "R6DOCTIM", "R7DOCTIM", "R8DOCTIM",
"R9DOCTIM", "R10DOCTIM", "R11DOCTIM", "R12DOCTIM", "R13DOCTIM", "R14DOCTIM",
"R15DOCTIM"))

# Build a function given a wave to pull all of the variables associated with that wave
vars <- colnames(hrs)
a <- grep("AGE", vars)
get_wave <- function(w){
  # Select variables
  vars_w <- grep(w, vars)
  # If w = 1-5, remove 11-15
  if(w <= 5){
    vars_not_needed <- grep(w+10, vars)
    vars_w <- setdiff(vars_w, vars_not_needed)
  }
  if(w==15){
    wgt <- grep("R15WTRESP", vars)
    vars_w <- vars_w[vars_w != wgt]
    vars_w <- c(1, a, wgt, vars_w)
  }else{
    # Add in HHIDPN, Age
    vars_w <- c(1, a, vars_w)
  }
  # Get data
  subset <- hrs[,vars_w]
  colnames(subset) <- c("HHIDPN","AGE", "WEIGHT", "HEALTH_CHANGE", "HBP",
"DIABETES", "CANCER", "LUNGS", "HEART_ATTACK", "STROKE", "PSYCH", "ARTHRITIS",
"OUT_PT", "DRUGS", "HOME_CARE", "SPECIAL_FAC", "HOSPITAL", "NURSING_HOME",
"SPENDING", "MEDICARE", "WEALTH", "SPEND_LIQ", "SPEND_SS")
  return(subset)
}

# Set up output dataset in long format
final_long <- as.data.frame(matrix(rep(NA, 23), 1, 23))
colnames(final_long) <- c("HHIDPN","AGE", "WEIGHT", "HEALTH_CHANGE", "HBP",
"DIABETES", "CANCER", "LUNGS", "HEART_ATTACK", "STROKE", "PSYCH", "ARTHRITIS",
"OUT_PT", "DRUGS", "HOME_CARE", "SPECIAL_FAC", "HOSPITAL", "NURSING_HOME",
"SPENDING", "MEDICARE", "WEALTH", "SPEND_LIQ", "SPEND_SS")
n <- dim(hrs)[1]
k <- 1
# Cycle through each observation
for(i in 1:n){
  # Cycle through each wave
  add <- FALSE
  for(j in 4:15){
    wave_data <- get_wave(j)[i,]
    # Compute age
    age <- wave_data$AGE + (j - 4)*2
    na_sum <- sum(is.na(wave_data))
    # Age range?
    if(age >= 65){
      # Check if we should make add true
      if(add!=TRUE && !is.na(wave_data$MEDICARE) && !is.na(wave_data$WEALTH)){
        if(wave_data$MEDICARE == "1.Yes" && wave_data$WEALTH >= 1000){
          add <- TRUE
        }
      }
      if(add==TRUE && na_sum < 3){
        wave_data$AGE <- age
        final_long[k, ] <- wave_data
        k <- k + 1
      }
    }
  }
}
# Unique obs
u <- unique(final_long$HHIDPN)
l <- length(u)
l

# Get avg. # of wvs observed
min_wvs <- 5
wvs_obs <- rep(NA, l)
for(i in 1:l){
  # How many waves observed?
  id <- u[i]
  this_id <- subset(final_long, final_long$HHIDPN==id)
  w <- length(this_id$HHIDPN)
  wvs_obs[i] <- w
  # Remove if less than 5
  if(w < 5){
    final_long <- final_long[final_long$HHIDPN!=id,]
  }
}
# How many unique individuals left?
length(unique(final_long$HHIDPN))

plot(table(final_long$AGE), xlab="Age", ylab="# of Observations", main="Distribution of
Observations by Age", type='l', col='dark red')

# Avg waves observed
mean(wvs_obs)
# Avg NAs per individual
mean(rowSums(is.na(final_long)))
# NAs by column
colSums(is.na(final_long))

# Drop Medicare, Spending and Wealth as they are no longer needed
final_long <- select(final_long, -c("SPENDING", "MEDICARE", "WEALTH"))
```

```
# Impute Categorical Predictor Values and make them numeric
final_long <- mutate(final_long, "OUT_PT" = ifelse(is.na(OUT_PT), "0.No", OUT_PT))  %>%
  mutate("DRUGS" = ifelse(is.na(DRUGS), "0.No", DRUGS))  %>%
  mutate("HOME_CARE" = ifelse(is.na(HOME_CARE), "0.No", HOME_CARE))  %>%
  mutate("SPECIAL_FAC" = ifelse(is.na(SPECIAL_FAC), "0.No", SPECIAL_FAC)) %>%
  mutate("HBP" = ifelse(is.na(HBP), "0.No", HBP)) %>%
  mutate("DIABETES" = ifelse(is.na(DIABETES), "0.No", DIABETES)) %>%
  mutate("CANCER" = ifelse(is.na(CANCER), "0.No", CANCER)) %>%
  mutate("LUNGS" = ifelse(is.na(LUNGS), "0.No", LUNGS)) %>%
  mutate("HEART_ATTACK" = ifelse(is.na(HEART_ATTACK), "0.No", HEART_ATTACK)) %>%
  mutate("STROKE" = ifelse(is.na(STROKE), "0.No", STROKE)) %>%
  mutate("PSYCH" = ifelse(is.na(PSYCH), "0.No", PSYCH)) %>%
  mutate("ARTHRITIS" = ifelse(is.na(ARTHRITIS), "0.No", ARTHRITIS))
final_long <- mutate(final_long, "OUT_PT" = ifelse(OUT_PT=="1.Yes", 1, 0)) %>%
  mutate("DRUGS" = ifelse(DRUGS=="1.Yes", 1, 0)) %>%
  mutate("HOME_CARE" = ifelse(HOME_CARE=="1.Yes", 1, 0)) %>%
  mutate("SPECIAL_FAC" = ifelse(SPECIAL_FAC=="1.Yes", 1, 0)) %>%
  mutate("HBP" = ifelse(HBP=="1.Yes", 1, 0)) %>%
  mutate("DIABETES" = ifelse(DIABETES=="1.Yes", 1, 0)) %>%
  mutate("CANCER" = ifelse(CANCER=="1.Yes", 1, 0)) %>%
  mutate("LUNGS" = ifelse(LUNGS=="1.Yes", 1, 0)) %>%
  mutate("HEART_ATTACK" = ifelse(HEART_ATTACK=="1.Yes", 1, 0)) %>%
  mutate("STROKE" = ifelse(STROKE=="1.Yes", 1, 0)) %>%
  mutate("PSYCH" = ifelse(PSYCH=="1.Yes", 1, 0)) %>%
  mutate("ARTHRITIS" = ifelse(ARTHRITIS=="1.Yes", 1, 0))
# Impute HEALTH_CHANGE and NEW_CONDS to zero
final_long <- mutate(final_long, "HEALTH_CHANGE" = ifelse(is.na(HEALTH_CHANGE), 0,
HEALTH_CHANGE)) %>%
  mutate("HOSPITAL" = ifelse(is.na(HOSPITAL), 0, HOSPITAL)) %>%
  mutate("NURSING_HOME" = ifelse(is.na(NURSING_HOME), 0, NURSING_HOME))

# Check that it worked
colSums(is.na(final_long))

# Divide by wealth measure that will be used
liq <- select(final_long, -c("SPEND_SS"))
ss <- select(final_long, -c("SPEND_LIQ"))

# Delete observations with missing outcome variable
liq <- subset(liq, liq$SPEND_LIQ < Inf & liq$SPEND_LIQ > -Inf)
ss <- subset(ss, ss$SPEND_SS < Inf & ss$SPEND_SS > -Inf)

# Liq - check consecutive waves
u <- unique(liq$HHIDPN)
l <- length(u)
for(i in 1:l){
  consec_wvs <- 1
  consec <- TRUE
  # How many waves observed?
  id <- u[i]
  this_id <- subset(liq, liq$HHIDPN==id)
  wvs <- length(this_id$HHIDPN)
  prev_age <- this_id$AGE[1]
  w <- 2
  while(consec == TRUE && w <= wvs){
    age <- this_id$AGE[w]
    if(age - prev_age != 2){
      consec <- FALSE
    }else{
      consec_wvs <- consec_wvs + 1
      prev_age <- age
      w <- w + 1
    }
  }
  # Remove if less than 5
  if(consec_wvs < 5){
    liq <- liq[liq$HHIDPN!=id,]
  }
}
# How many unique individuals left?
I <- length(unique(liq$HHIDPN))
I

# Create training and validation sets (70/30 split)
i <- sample(1:I,floor(.7*I), replace=FALSE)
train_ids <- unique(liq$HHIDPN)[i]
liq_train <- subset(liq, liq$HHIDPN %in% train_ids)
test_ids <- unique(liq$HHIDPN)[-i]
liq_test <- subset(liq, liq$HHIDPN %in% test_ids)

# SS - check consecutive waves
u <- unique(ss$HHIDPN)
I <- length(u)
for(i in 1:I){
  consec_wvs <- 1
  consec <- TRUE
  # How many waves observed?
  id <- u[i]
  this_id <- subset(ss, ss$HHIDPN==id)
  wvs <- length(this_id$HHIDPN)
  prev_age <- this_id$AGE[1]
  w <- 2
  while(consec == TRUE && w <= wvs){
    age <- this_id$AGE[w]
    if(age - prev_age != 2){
      consec <- FALSE
    }else{
      consec_wvs <- consec_wvs + 1
      prev_age <- age
      w <- w + 1
    }
  }
  # Remove if less than 5
  if(consec_wvs < 5){
    ss <- ss[ss$HHIDPN!=id,]
  }
}
# How many unique individuals left?
```

```
l <- length(unique(ss$HHIDPN))
l

# Create training and validation sets (70/30 split)
i <- sample(1:l,floor(.7*l), replace=FALSE)
train_ids <- unique(ss$HHIDPN)[i]
ss_train <- subset(ss, ss$HHIDPN %in% train_ids)
test_ids <- unique(ss$HHIDPN)[-i]
ss_test <- subset(ss, ss$HHIDPN %in% test_ids)

# Write to csv
write.csv(liq_train, paste(pwd, "/Data Clean/liq_train.csv", sep = ''))
write.csv(liq_test, paste(pwd, "/Data Clean/liq_test.csv", sep = ''))
write.csv(ss_train, paste(pwd, "/Data Clean/ss_train.csv", sep = ''))
write.csv(ss_test, paste(pwd, "/Data Clean/ss_test.csv", sep = ''))
```

*****eda.R*****

```
## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("HEART_ATTACK", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$HEART_ATTACK==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$HEART_ATTACK <- as.character(plot$HEART_ATTACK)
ggplot(plot, aes(x=AGE, y=Value, fill=HEART_ATTACK)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("STROKE", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$STROKE==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}
```

```
## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$STROKE <- as.character(plot$STROKE)
ggplot(plot, aes(x=AGE, y=Value, fill=STROKE)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("HBP", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$HBP==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$HBP <- as.character(plot$HBP)
ggplot(plot, aes(x=AGE, y=Value, fill=HBP)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("DIABETES", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$DIABETES==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$DIABETES <- as.character(plot$DIABETES)
ggplot(plot, aes(x=AGE, y=Value, fill=DIABETES)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("CANCER", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$CANCER==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$CANCER <- as.character(plot$CANCER)
ggplot(plot, aes(x=AGE, y=Value, fill=CANCER)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("LUNGS", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$LUNGS==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$LUNGS <- as.character(plot$LUNGS)
ggplot(plot, aes(x=AGE, y=Value, fill=LUNGS)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("PSYCH", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$PSYCH==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$PSYCH <- as.character(plot$PSYCH)
ggplot(plot, aes(x=AGE, y=Value, fill=PSYCH)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("ARTHRITIS", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$ARTHRITIS==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$ARTHRITIS <- as.character(plot$ARTHRITIS)
ggplot(plot, aes(x=AGE, y=Value, fill=ARTHRITIS)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("OUT_PT", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$OUT_PT==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$OUT_PT <- as.character(plot$OUT_PT)
ggplot(plot, aes(x=AGE, y=Value, fill=OUT_PT)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("DRUGS", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$DRUGS==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$DRUGS <- as.character(plot$DRUGS)
ggplot(plot, aes(x=AGE, y=Value, fill=DRUGS)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("HOME_CARE", "AGE", "Value")

k <- 1
```

```
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$HOME_CARE==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}


## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$HOME_CARE <- as.character(plot$HOME_CARE)
ggplot(plot, aes(x=AGE, y=Value, fill=HOME_CARE)) + geom_area()

## Dataset for plotting
plot <- as.data.frame(matrix(rep(NA, 234), 39*2, 3))
colnames(plot) <- c("SPECIAL_FAC", "AGE", "Value")

k <- 1
for(i in 65:103){
  age_sub <- subset(ss_train, ss_train$AGE==i)
  for(j in 0:1){
    var_sub <- subset(age_sub, age_sub$SPECIAL_FAC==j)
    plot[k,] <- c(j, i, mean(var_sub$SPEND_SS, trim = .1, na.rm=TRUE))
    k <- k + 1
  }
}

## Plot
plot$Value[is.na(plot$Value)] <- 0
plot$SPECIAL_FAC <- as.character(plot$SPECIAL_FAC)
ggplot(plot, aes(x=AGE, y=Value, fill=SPECIAL_FAC)) + geom_area()

plot(ss_train$DOCTOR, ss_train$SPEND_SS, main = "Doctor", xlab = "Doctor Visits", ylab =
"Adjusted Spending")
plot(ss_train$HOSPITAL, ss_train$SPEND_SS, main = "Hospital", xlab = "Hospital Nights", ylab
= "Adjusted Spending")
plot(ss_train$NURSING_HOME, ss_train$SPEND_SS, main = "Nursing Home", xlab = "Nursing
Home Nights", ylab = "Adjusted Spending")

ss_train$AGE_RANGE = rep(NA, length(ss_train$HHIDPN))
ss_train$AGE_RANGE <- ifelse(ss_train$AGE < 75, '65-74', ss_train$AGE_RANGE)
ss_train$AGE_RANGE <- ifelse(ss_train$AGE > 74 & ss_train$AGE < 85, '75-84',
ss_train$AGE_RANGE)
ss_train$AGE_RANGE <- ifelse(ss_train$AGE > 84, '85+', ss_train$AGE_RANGE)
ggplot(ss_train, aes(x = AGE_RANGE, y = SPEND_SS, fill = AGE_RANGE)) +geom_boxplot()
```

Method 1

*****eval_sim*****

```
# Plot n simulated spending paths
n <- 1000
for_plot <- simulation_wrapper(n)
u <- unique(for_plot$ID)
plot(x = 1, type = "n",
     xlim = c(65, 90),
     ylim = c(0, 10),
     xlab = "Age", ylab = "Adjusted Spending", main = paste(n, "Simulated Spend Paths"))
cols <- sample(colors(), n, replace = TRUE)
for(i in 1:n){
  this_id <- subset(for_plot, for_plot$ID == u[i])
  points(this_id$AGE, this_id$SPENDING, type = "l", col = cols[i])
}

# Sample of 1000 real spend paths vs. simulated
sim <- simulation_wrapper(100)
u <- unique(ss_train$HHIDPN)
l <- length(u)
ids <- sample(1:l, size = 100, replace = FALSE)
test_ids <- u[ids]
plot(x = 1, type = "n",
     xlim = c(65, 90),
     ylim = c(0, 10),
     xlab = "Age", ylab = "Adjusted Spending", main = "Simulated vs. Training Data")
s <- unique(sim$ID)
for(i in 1:100){
  this_id_train <- subset(ss_train, ss_train$HHIDPN==test_ids[i])
  this_id_sim <- subset(sim, sim$ID == s[i])
  points(this_id_sim$AGE, this_id_sim$SPENDING, type = "l", col = "red")
  points(this_id_train$AGE, exp(this_id_train$SPEND_SS), type = "l", col = "black")
}

sim <- simulation_wrapper(10000)

# Mean
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- mean(age_sub$SPENDING)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "Mean")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
```

```
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- mean(age_sub$SPEND_SS)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# Median
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- median(age_sub$SPENDING)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "Median")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- median(age_sub$SPEND_SS)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# 5th Percentile
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- quantile(age_sub$SPENDING, .05, na.rm = TRUE)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "5th Percentile")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- quantile(age_sub$SPEND_SS, .05, na.rm = TRUE)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# 10th Percentile
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- quantile(age_sub$SPENDING, .1, na.rm = TRUE)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "10th Percentile")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- quantile(age_sub$SPEND_SS, .1, na.rm = TRUE)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# 25th Percentile
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- quantile(age_sub$SPENDING, .25, na.rm = TRUE)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "25th Percentile")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- quantile(age_sub$SPEND_SS, .25, na.rm = TRUE)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")
```

```
# 75th Percentile
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- quantile(age_sub$SPENDING, .75, na.rm = TRUE)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "75th Percentile")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- quantile(age_sub$SPEND_SS, .75, na.rm = TRUE)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# 90th Percentile
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- quantile(age_sub$SPENDING, .9, na.rm = TRUE)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "90th Percentile")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- quantile(age_sub$SPEND_SS, .9, na.rm = TRUE)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# 95th Percentile
ages <- unique(sim$AGE)
sim_metric <- rep(NA, length(ages))
```

```
k <- 1
for(age in ages){
  age_sub <- subset(sim, sim$AGE == age)
  sim_metric[k] <- quantile(age_sub$SPENDING, .95, na.rm = TRUE)
  k <- k + 1
}
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 2),
     xlab = "Age", ylab = "Adjusted Spending", main = "95th Percentile")
points(ages, exp(sim_metric), type = 'l', col = "red")

ages <- 65:85
train_metric <- rep(NA, length(ages))
k <- 1
for(age in ages){
  age_sub <- subset(ss_train, ss_train$AGE == age)
  train_metric[k] <- quantile(age_sub$SPEND_SS, .95, na.rm = TRUE)
  k <- k + 1
}
points(ages, exp(train_metric), type = 'l', col = "blue")

# Standard Deviation
ids <- unique(sim$ID)
sim_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
  id_sub <- subset(sim, sim$ID == id)
  sim_metric[k] <- sd(id_sub$SPENDING)
  k <- k + 1
}

ids <- unique(ss_train$HHIDPN)
train_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
  id_sub <- subset(ss_train, ss_train$HHIDPN == id)
  train_metric[k] <- sd(id_sub$SPEND_SS)
  k <- k + 1
}

boxplot(sim_metric)
boxplot(train_metric)

# Max Percent Increase
ids <- unique(sim$ID)
sim_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
  id_sub <- subset(sim, sim$ID == id)
  idl <- length(id_sub$ID)
  percent_increase <- rep(NA, idl)
  for(obs in 2:idl){
    if(obs ==1){
```

```
   percent_increase[1] <- 0
  }else{
   percent_increase[obs] <- (id_sub$SPENDING[obs] +
id_sub$SPENDING[obs-1])/(id_sub$SPENDING[obs-1])
  }
 }
 sim_metric[k] <- max(percent_increase)
 k <- k + 1
}

ids <- unique(ss_train$HHIDPN)
train_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
 id_sub <- subset(ss_train, ss_train$HHIDPN == id)
 idl <- length(id_sub$HHIDPN)
 percent_increase <- rep(NA, idl)
 for(obs in 2:idl){
  if(obs ==1){
   percent_increase[1] <- 0
  }else{
   percent_increase[obs] <- (id_sub$SPEND_SS[obs] +
id_sub$SPEND_SS[obs-1])/(id_sub$SPEND_SS[obs-1])
  }
 }
 train_metric[k] <- max(percent_increase)
 k <- k + 1
}

boxplot(log(sim_metric))
boxplot(log(train_metric))

# Avg Percent Increase
ids <- unique(sim$ID)
sim_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
 id_sub <- subset(sim, sim$ID == id)
 idl <- length(id_sub$ID)
 percent_increase <- rep(NA, idl)
 for(obs in 2:idl){
  if(obs ==1){
   percent_increase[1] <- 0
  }else{
   percent_increase[obs] <- (id_sub$SPENDING[obs] +
id_sub$SPENDING[obs-1])/(id_sub$SPENDING[obs-1])
  }
 }
 sim_metric[k] <- mean(percent_increase)
 k <- k + 1
}

ids <- unique(ss_train$HHIDPN)
train_metric <- rep(NA, length(ids))
```

```
k <- 1
for(id in ids){
  id_sub <- subset(ss_train, ss_train$HHIDPN == id)
  idl <- length(id_sub$HHIDPN)
  percent_increase <- rep(NA, idl)
  for(obs in 2:idl){
    if(obs ==1){
      percent_increase[1] <- 0
    }else{
      percent_increase[obs] <- (id_sub$SPEND_SS[obs] +
id_sub$SPEND_SS[obs-1])/(id_sub$SPEND_SS[obs-1])
    }
  }
  train_metric[k] <- mean(percent_increase)
  k <- k + 1
}

boxplot(log(sim_metric))
boxplot(log(train_metric))

# Average Second Differences
ids <- unique(sim$ID)
sim_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
  id_sub <- subset(sim, sim$ID == id)
  idl <- length(id_sub$ID)
  sum <- 0
  for(obs in 3:idl){
    sum <- sum + (id_sub$SPENDING[obs] - 2*id_sub$SPENDING[obs-1] +
id_sub$SPENDING[obs-2])^2/4
  }
  sim_metric[k] <-sum
  k <- k + 1
}
mean(sim_metric)

ids <- unique(ss_train$HHIDPN)
train_metric <- rep(NA, length(ids))
k <- 1
for(id in ids){
  id_sub <- subset(ss_train, ss_train$HHIDPN == id)
  idl <- length(id_sub$HHIDPN)
  sum <- 0
  for(obs in 3:idl){
    sum <- sum + (id_sub$SPEND_SS[obs] - 2*id_sub$SPEND_SS[obs-1] +
id_sub$SPEND_SS[obs-2])^2/4
  }
  train_metric[k] <-sum
  k <- k + 1
}
mean(train_metric, na.rm = TRUE)
```

*******method1*******

```
u <- unique(ss_train$HHIDPN)
l <- length(u)
s <- sample(1:l, size = 10, replace = FALSE)
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 6),
     xlab = "Age", ylab = "Adjusted Spending", main = paste(10, "Sampled Spend Paths"))
colors = c("firebrick1", "dodgerblue", "darkorange", "darkorchid1", "gold", "forestgreen",
"deeppink", "black", "gray54", "chocolate4")
for(i in 1:10){
  sample_id <- u[s[i]]
  this_id <- subset(ss_train, ss_train$HHIDPN==sample_id)
  points(this_id$AGE, exp(this_id$SPEND_SS), col = colors[i], type = 'l', lwd = 2)
}

s <- sample(1:l, size = 1, replace = FALSE)
plot(x = 1, type = "n",
     xlim = c(65, 105),
     ylim = c(0, 6),
     xlab = "Age", ylab = "Adjusted Spending", main = paste(1, "Sampled Spend Paths"))
sample_id <- u[s]
this_id <- subset(ss_train, ss_train$HHIDPN==sample_id)
points(this_id$AGE, exp(this_id$SPEND_SS), col = "black", type = 'l', lwd = 4)

################################        CREATE HEALTH PROBABILITIES
############################

u <- unique(ss_train$HHIDPN)
l <- length(u)
health_probs <- as.data.frame(matrix(rep(0, 39*5), 39, 5))
colnames(health_probs) <- c("Age", "S1", "S2", "S3", "S4")
for(i in 1:l){
  this_id <- subset(ss_train, ss_train$HHIDPN==u[i])
  tl <- length(this_id$HHIDPN)
  for(j in 1:tl){
    idx <- this_id$AGE[j] - 64
    health_probs$Age[idx] <- health_probs$Age[idx] + 1
    if(this_id$STROKE[j]==1 | this_id$HEART_ATTACK[j]==1 | this_id$PSYCH[j]==1){
      health_probs$S3[idx] <- health_probs$S3[idx] + 1
    }else if(this_id$HOSPITAL[j] > 0){
      health_probs$S2[idx] <- health_probs$S2[idx] + 1
    }else if(this_id$NURSING_HOME[j] > 0){
      health_probs$S4[idx] <- health_probs$S4[idx] + 1
    }else{
      health_probs$S1[idx] <- health_probs$S1[idx] + 1
    }
  }
}
```

```
health_probs$S1 <- round(health_probs$S1/health_probs$Age, 3)
health_probs$S2 <- round(health_probs$S2/health_probs$Age, 3)
health_probs$S3 <- round(health_probs$S3/health_probs$Age, 3)
health_probs$S4 <- round(health_probs$S4/health_probs$Age, 3)
health_probs$Age <- seq(65, 103, 1)

########################    HOW DO OTHER VARS CHANGE IN EACH SCENARIO
##############################

# S3

s3 <- subset(ss_train, ss_train$STROKE==1 | ss_train$HEART_ATTACK==1 |
ss_train$PSYCH==1)

heart_attack <- subset(s3, s3$HEART_ATTACK==1)
heart_attack <- subset(heart_attack, !is.na(heart_attack$HHIDPN))
stroke <- subset(s3, s3$STROKE==1)
psych <- subset(s3, s3$PSYCH==1)

event_probs <- c(round(length(heart_attack$HHIDPN)/length(s3$HHIDPN), 2),
              round(length(stroke$HHIDPN)/length(s3$HHIDPN), 2),
              round(length(psych$HHIDPN)/length(s3$HHIDPN), 2))

#round(table(heart_attack$SPECIAL_FAC)/length(heart_attack$HHIDPN), 2)
#round(table(heart_attack$HOME_CARE)/length(heart_attack$HHIDPN), 2)
#round(table(heart_attack$HBP)/length(heart_attack$HHIDPN), 2)
#round(table(heart_attack$HEALTH_CHANGE)/length(heart_attack$HHIDPN), 2)
#round(table(heart_attack$DOCTOR)/length(heart_attack$HHIDPN), 3)
#round(table(heart_attack$HOSPITAL)/length(heart_attack$HHIDPN), 3)

#round(table(stroke$SPECIAL_FAC)/length(stroke$HHIDPN), 2)
#round(table(stroke$HOME_CARE)/length(stroke$HHIDPN), 2)
#round(table(stroke$HBP)/length(stroke$HHIDPN), 2)
#round(table(stroke$HEALTH_CHANGE)/length(stroke$HHIDPN), 2)
#round(table(stroke$DOCTOR)/length(stroke$HHIDPN), 3)
#round(table(stroke$HOSPITAL)/length(stroke$HHIDPN), 3)

#round(table(psych$SPECIAL_FAC)/length(psych$HHIDPN), 2)
#round(table(psych$HOME_CARE)/length(psych$HHIDPN), 2)
#round(table(psych$HBP)/length(psych$HHIDPN), 2)
#round(table(psych$HEALTH_CHANGE)/length(psych$HHIDPN), 2)
#round(table(psych$DOCTOR)/length(psych$HHIDPN), 3)
#round(table(psych$HOSPITAL)/length(psych$HHIDPN), 3)

# S1
s1 <- subset(ss_train, ss_train$HEART_ATTACK==0 & ss_train$STROKE==0 &
ss_train$PSYCH==0 & ss_train$NURSING_HOME==0 & ss_train$HOSPITAL==0)

#round(table(s1$HBP)/length(s1$HHIDPN), 2)
#round(table(s1$ARTHRITIS)/length(s1$HHIDPN), 2)
#round(table(s1$DRUGS)/length(s1$DRUGS), 2)
#round(table(s1$HOME_CARE)/length(s1$HHIDPN), 2)
#round(table(s1$SPECIAL_FAC)/length(s1$HHIDPN), 2)
#round(table(s1$HEALTH_CHANGE)/length(s1$HHIDPN), 3)
```

```
#round(table(s1$DOCTOR)/length(s1$HHIDPN), 3)

# S2
s2 <- subset(ss_train, ss_train$HEART_ATTACK==0 & ss_train$STROKE==0 &
ss_train$PSYCH==0 & ss_train$NURSING_HOME==0 & ss_train$HOSPITAL > 0)

#round(table(s2$HBP)/length(s2$HHIDPN), 2)
#round(table(s2$ARTHRITIS)/length(s2$HHIDPN), 2)
#round(table(s2$DRUGS)/length(s2$DRUGS), 2)
#round(table(s2$HOME_CARE)/length(s2$HHIDPN), 2)
#round(table(s2$SPECIAL_FAC)/length(s2$HHIDPN), 2)
#round(table(s2$HEALTH_CHANGE)/length(s2$HHIDPN), 3)
#round(table(s2$DOCTOR)/length(s2$HHIDPN), 3)
#round(table(s2$HOSPITAL)/length(s2$HHIDPN), 3)

# S4
s4 <- subset(ss_train, ss_train$NURSING_HOME > 0)
#round(table(s4$HBP)/length(s4$HHIDPN), 2)
#round(table(s4$ARTHRITIS)/length(s4$HHIDPN), 2)
#round(table(s4$DRUGS)/length(s4$DRUGS), 2)
#round(table(s4$HOME_CARE)/length(s4$HHIDPN), 2)
#round(table(s4$SPECIAL_FAC)/length(s4$HHIDPN), 2)
#round(table(s4$HEALTH_CHANGE)/length(s4$HHIDPN), 3)
#round(table(s4$DOCTOR)/length(s4$HHIDPN), 3)
#round(table(s4$HOSPITAL)/length(s4$HHIDPN), 3)
#round(table(s4$NURSING_HOME)/length(s4$HHIDPN), 3)

###########################        FUNCTION TO CREATE HEALTH PROFILE
###########################

create_health_profile <- function(yrs_obs, start_age){
  HealthVars <- c("HEALTH_CHANGE", "HBP", "HEART_ATTACK", "STROKE", "DIABETES",
"CANCER", "PSYCH", "OUT_PT",
                  "HOME_CARE", "ARTHRITIS", "SPECIAL_FAC", "DRUGS", "HOSPITAL",
"DOCTOR",
                  "NURSING_HOME")
  simulators <- c("S1", "S2", "S3", "S4")
  num_vars <- length(HealthVars)
  age <- start_age
  health_profile <- as.data.frame(matrix(rep(NA, yrs_obs*num_vars), yrs_obs, num_vars))
  colnames(health_profile) <- HealthVars
  for(i in 1:yrs_obs){
    # Retrieve probabilities
    probs <- subset(health_probs, health_probs$Age==age)
    if(age > 103){
      probs <- subset(health_probs, health_probs$Age==103)
    }
    # Choose situation
    # 1 - No health events, no services used
    # 2 - Hospital visit (not from serious condition)
    # 3 - Serious condition (heart attack, stroke, pysch)
    # 4 - Nursing Home
    s <- sample(1:4, size = 1, replace = TRUE, prob = probs[2:5])
    # Simulate year based on situation
    health_profile[i,] <- eval(call(simulators[s]))
    age <- age + 2
  }
  health_profile$AGE <- seq(65, by=2, length.out=yrs_obs)
  return(health_profile)
}

# Plot
plot(x = 1, type = "n",
     xlim = c(65, 103),
     ylim = c(0, 1),
     xlab = "Age", ylab = "Adjusted Spending", main = "Scenarios vs. Age")
points(health_probs$Age, health_probs$S1, type = "l", col = "blue")
points(health_probs$Age, health_probs$S2, type = "l", col = "green")
points(health_probs$Age, health_probs$S3, type = "l", col = "orange")
points(health_probs$Age, health_probs$S4, type = "l", col = "red")

####################################         SIMULATOR FOR S3
###################################

S3 <- function(){
  # Set up return dataset
  HealthVars <- c("HEALTH_CHANGE", "HBP", "HEART_ATTACK", "STROKE", "DIABETES",
"CANCER", "PSYCH",
                  "OUT_PT", "HOME_CARE", "ARTHRITIS", "SPECIAL_FAC", "DRUGS",
"HOSPITAL",
                  "DOCTOR", "NURSING_HOME")
  num_vars <- length(HealthVars)
  health_profile <- as.data.frame(matrix(rep(NA, 1*num_vars), 1, num_vars))
  colnames(health_profile) <- HealthVars
  # Set non significant variables to zero
  health_profile$DIABETES <- 0
  health_profile$CANCER <- 0
  health_profile$OUT_PT <- 0
  # Set variables were the probability does not change based on health issue
  health_profile$DRUGS <- 1
  health_profile$NURSING_HOME <- 0
  health_profile$ARTHRITIS <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  # Choose event
  # 1 - Heart Attack
  # 2 - Stroke
  # 3 - Psych
  e <- sample(1:3, size = 1, replace = TRUE, prob = event_probs)
  if(e==1){
    health_profile$HEART_ATTACK <- 1
    health_profile$STROKE <- 0
    health_profile$PSYCH <- 0
    # Choose other variables
    health_profile$SPECIAL_FAC <- sample(c(0, 1), size = 1, replace = TRUE, prob = c(.75, .25))
    health_profile$HOME_CARE <- sample(c(0, 1), size = 1, replace = TRUE, prob = c(.78, .22))
    health_profile$HBP <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.94, .06))
    health_profile$HEALTH_CHANGE <- sample(-2:3, size = 1, replace = TRUE, prob = c(.02, .1,
.44, .33, .1, .02))
    health_profile$DOCTOR <- sample(c(0:12), size = 1, replace = TRUE, prob = c(.08, .04, .09,
.13, .1, .08, .1, .05, .05, .05, .05, .05, .03))
    health_profile$HOSPITAL <- sample(c(0:8), size = 1, replace = TRUE, prob = c(.25, .18, .15,
.12, .1, .08, .05, .05, .02))
  }else if(e==2){
    health_profile$HEART_ATTACK <- 0
    health_profile$STROKE <- 1
    health_profile$PSYCH <- 0
    # Choose other variables
    health_profile$SPECIAL_FAC <- sample(c(0, 1), size = 1, replace = TRUE, prob = c(.7, .3))
    health_profile$HOME_CARE <- sample(c(0, 1), size = 1, replace = TRUE, prob = c(.68, .32))
    health_profile$HBP <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.93, .07))
    health_profile$HEALTH_CHANGE <- sample(-2:4, size = 1, replace = TRUE, prob = c(.03,
.12, .39, .31, .11, .03, .01))
    health_profile$DOCTOR <- sample(c(0:12), size = 1, replace = TRUE, prob = c(.08, .04, .09,
.13, .1, .08, .1, .05, .05, .05, .05, .05, .03))
    health_profile$HOSPITAL <- sample(c(0:8), size = 1, replace = TRUE, prob = c(.25, .18, .15,
.12, .1, .08, .05, .05, .02))
  }else{
    health_profile$HEART_ATTACK <- 0
    health_profile$STROKE <- 0
    health_profile$PSYCH <- 1
    # Choose other variables
    health_profile$SPECIAL_FAC <- sample(c(0, 1), size = 1, replace = TRUE, prob = c(.74, .26))
    health_profile$HOME_CARE <- sample(c(0, 1), size = 1, replace = TRUE, prob = c(.77, .23))
    health_profile$HBP <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.95, .05))
    health_profile$HEALTH_CHANGE <- sample(-2:3, size = 1, replace = TRUE, prob = c(.02,
.16, .43, .27, .07, .03))
    health_profile$DOCTOR <- sample(c(0:12), size = 1, replace = TRUE, prob = c(.08, .04, .09,
.13, .1, .08, .1, .05, .05, .05, .05, .05, .03))
    health_profile$HOSPITAL <- sample(c(0:8), size = 1, replace = TRUE, prob = c(.25, .18, .15,
.12, .1, .08, .05, .05, .02))}
  return(health_profile)

}

#####################################         SIMULATOR FOR S1
####################################

S1 <- function(){
  # Set up return dataset
  HealthVars <- c("HEALTH_CHANGE", "HBP", "HEART_ATTACK", "STROKE", "DIABETES",
"CANCER", "PSYCH",
                 "OUT_PT", "HOME_CARE", "ARTHRITIS", "SPECIAL_FAC", "DRUGS",
"HOSPITAL",
                 "DOCTOR", "NURSING_HOME")
  num_vars <- length(HealthVars)
  health_profile <- as.data.frame(matrix(rep(NA, 1*num_vars), 1, num_vars))
  colnames(health_profile) <- HealthVars
  # Set non significant variables to zero
  health_profile$DIABETES <- 0
  health_profile$CANCER <- 0
  health_profile$OUT_PT <- 0
  # Set major health and events and services to zero
  health_profile$HEART_ATTACK <- 0
  health_profile$STROKE <- 0
  health_profile$PSYCH <- 0
  health_profile$NURSING_HOME <- 0
  health_profile$HOSPITAL <- 0
  # Set other variables
  health_profile$ARTHRITIS <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  health_profile$HBP <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  health_profile$DRUGS <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.1, .9))
  health_profile$HOME_CARE <- 0
  health_profile$SPECIAL_FAC <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.9, .1))
  health_profile$HEALTH_CHANGE <- sample(c(-3:4), size = 1, replace = TRUE, prob = c(.002,
.021, .176, .564, .205, .03, .003, .001))
  health_profile$DOCTOR <- sample(c(0:12), size = 1, replace = TRUE, prob = c(.08, .04, .09,
.13, .1, .08, .1, .05, .05, .05, .05, .05, .03))
  return(health_profile)
}


#####################################        SIMULATOR FOR S2
####################################

S2 <- function(){
  # Set up return dataset
  HealthVars <- c("HEALTH_CHANGE", "HBP", "HEART_ATTACK", "STROKE", "DIABETES",
"CANCER", "PSYCH",
                  "OUT_PT", "HOME_CARE", "ARTHRITIS", "SPECIAL_FAC", "DRUGS",
"HOSPITAL",
                  "DOCTOR", "NURSING_HOME")
  num_vars <- length(HealthVars)
  health_profile <- as.data.frame(matrix(rep(NA, 1*num_vars), 1, num_vars))
  colnames(health_profile) <- HealthVars
  # Set non significant variables to zero
  health_profile$DIABETES <- 0
  health_profile$CANCER <- 0
  health_profile$OUT_PT <- 0
  # Set major health and events and nursing home to zero
  health_profile$HEART_ATTACK <- 0
  health_profile$STROKE <- 0
  health_profile$PSYCH <- 0
  health_profile$NURSING_HOME <- 0
  # Set other variables
  health_profile$HBP <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  health_profile$ARTHRITIS <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  health_profile$DRUGS <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.05, .95))
  health_profile$HOME_CARE <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.8, .2))
  health_profile$SPECIAL_FAC <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.75, .25))
  health_profile$HEALTH_CHANGE <- sample(c(-3:4), size = 1, replace = TRUE, prob = c(.002,
.021, .176, .564, .205, .03, .003, .001))
  health_profile$DOCTOR <- sample(c(0:12), size = 1, replace = TRUE, prob = c(.08, .04, .09,
.13, .1, .08, .1, .05, .05, .05, .05, .05, .03))
  health_profile$HOSPITAL <- sample(c(0:8), size = 1, replace = TRUE, prob = c(.25, .18, .15,
.12, .1, .08, .05, .05, .02))
  return(health_profile)
}
```

```
#####################################         SIMULATOR FOR S4
####################################

S4 <- function(){
  # Set up return dataset
  HealthVars <- c("HEALTH_CHANGE", "HBP", "HEART_ATTACK", "STROKE", "DIABETES",
"CANCER", "PSYCH",
                  "OUT_PT", "HOME_CARE", "ARTHRITIS", "SPECIAL_FAC", "DRUGS",
"HOSPITAL",
                  "DOCTOR", "NURSING_HOME")
  num_vars <- length(HealthVars)
  health_profile <- as.data.frame(matrix(rep(NA, 1*num_vars), 1, num_vars))
  colnames(health_profile) <- HealthVars
  # Set non significant variables to zero
  health_profile$DIABETES <- 0
  health_profile$CANCER <- 0
  health_profile$OUT_PT <- 0
  # Set major health and events and nursing home to zero
  health_profile$HEART_ATTACK <- 0
  health_profile$STROKE <- 0
  health_profile$PSYCH <- 0
  health_profile$NURSING_HOME <- 1
  # Set other variables
  health_profile$HBP <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  health_profile$ARTHRITIS <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.96, .04))
  health_profile$DRUGS <- 1
  health_profile$HOME_CARE <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.5, .5))
  health_profile$SPECIAL_FAC <- sample(c(0,1), size = 1, replace = TRUE, prob = c(.55, .45))
  health_profile$HEALTH_CHANGE <- sample(c(-1:4), size = 1, replace = TRUE, prob = c(.1,
.45, .25, .1, .08, .02))
  health_profile$DOCTOR <- sample(c(0:12), size = 1, replace = TRUE, prob = c(.08, .04, .09,
.13, .1, .08, .1, .05, .05, .05, .05, .05, .03))
  health_profile$HOSPITAL <- sample(c(0:8), size = 1, replace = TRUE, prob = c(.25, .18, .15,
.12, .1, .08, .05, .05, .02))
  health_profile$NURSING_HOME <- sample(c(0:730), size = 1, replace = TRUE)

  return(health_profile)
}

ages <- c()
u <- unique(ss_train$HHIDPN)
l <- length(u)
for(i in 1:l){
  this_id <- subset(ss_train, ss_train$HHIDPN==u[i])
  age <- this_id$AGE[1]
  ages <- c(ages, age)
}
write.csv(round(table(ages)/length(ages), 4), paste(pwd, "/Method 1/age_probs.csv", sep = ''))
write.csv(round(table(table(ss_train$HHIDPN))/5601, 4), paste(pwd, "/Method 1/yrs_probs.csv",
sep = ''))


# Make factor variables
```

```
ss_train <- mutate(ss_train, "HBP" = as.factor(HBP)) %>%
  mutate("DIABETES" = as.factor(DIABETES)) %>%
  mutate("CANCER" = as.factor(CANCER)) %>%
  mutate("LUNGS" = as.factor(LUNGS)) %>%
  mutate("HEART_ATTACK" = as.factor(HEART_ATTACK)) %>%
  mutate("STROKE" = as.factor(STROKE)) %>%
  mutate("PSYCH" = as.factor(PSYCH)) %>%
  mutate("ARTHRITIS" = as.factor(ARTHRITIS)) %>%
  mutate("OUT_PT" = as.factor(OUT_PT)) %>%
  mutate("DRUGS" = as.factor(DRUGS)) %>%
  mutate("HOME_CARE" = as.factor(HOME_CARE)) %>%
  mutate("SPECIAL_FAC" = as.factor(SPECIAL_FAC))

ss_test <- mutate(ss_test, "HBP" = as.factor(HBP)) %>%
  mutate("DIABETES" = as.factor(DIABETES)) %>%
  mutate("CANCER" = as.factor(CANCER)) %>%
  mutate("LUNGS" = as.factor(LUNGS)) %>%
  mutate("HEART_ATTACK" = as.factor(HEART_ATTACK)) %>%
  mutate("STROKE" = as.factor(STROKE)) %>%
  mutate("PSYCH" = as.factor(PSYCH)) %>%
  mutate("ARTHRITIS" = as.factor(ARTHRITIS)) %>%
  mutate("OUT_PT" = as.factor(OUT_PT)) %>%
  mutate("DRUGS" = as.factor(DRUGS)) %>%
  mutate("HOME_CARE" = as.factor(HOME_CARE)) %>%
  mutate("SPECIAL_FAC" = as.factor(SPECIAL_FAC))

# Remove log transformation for logistic regression
ss_train$SPEND_SS = exp(ss_train$SPEND_SS)
ss_test$SPEND_SS = exp(ss_test$SPEND_SS)
ss_train$LAST_YEAR = exp(ss_train$LAST_YEAR)
ss_test$LAST_YEAR = exp(ss_test$LAST_YEAR)

# Confusion Matrix, model, test data, threshold if predicted probability >= thresh -> predict 1,
otherwise predict 0
performance <- function(model, data, thresh){
  fit_vals <- predict(model,data,type='response')
  true_neg <- 0
  true_pos <- 0
  false_neg <- 0
  false_pos <- 0
  for(i in 1:length(data$SHOCK2)){
    if(fit_vals[i] >= thresh){
      if(data$SHOCK2[i]==1){
        true_pos <- true_pos + 1
      }else{
        false_pos <- false_pos + 1
      }
    }else{
      if(data$SHOCK2[i]==0){
        true_neg <- true_neg + 1
      }else{
        false_neg <- false_neg + 1
      }
    }
  }
  cm <- matrix(data=c(true_pos, false_pos, false_neg, true_neg), 2, 2, byrow = TRUE,
dimnames = list("Predicted" = c("+", "-"), "Actual" = c("+", "-")))
  cat("Confusion Matrix:\n")
  print(cm)
  print((false_pos + false_neg)/length(data$HHIDPN))
}

performance2 <- function(model, data, thresh){
  fit_vals <- predict(model,data,type='response')
  true_neg <- 0
  true_pos <- 0
  false_neg <- 0
  false_pos <- 0
  for(i in 1:length(data$SHOCK2)){
    if(fit_vals[i] >= thresh){
      if(data$SHOCK2[i]==1){
        true_pos <- true_pos + 1
      }else{
        false_pos <- false_pos + 1
      }
    }else{
      if(data$SHOCK2[i]==0){
        true_neg <- true_neg + 1
      }else{
        false_neg <- false_neg + 1
      }
    }
  }
  return((true_pos + false_pos)/length(data$HHIDPN))
}

performance3 <- function(model, data, thresh){
  fit_vals <- predict(model,data,type='response')
  true_neg <- 0
  true_pos <- 0
  false_neg <- 0
  false_pos <- 0
  for(i in 1:length(data$SHOCK2)){
    if(fit_vals[i] >= thresh){
      if(data$SHOCK2[i]==1){
        true_pos <- true_pos + 1
      }else{
        false_pos <- false_pos + 1
      }
    }else{
      if(data$SHOCK2[i]==0){
        true_neg <- true_neg + 1
      }else{
        false_neg <- false_neg + 1
      }
    }
  }
  return((false_pos + false_neg)/length(data$HHIDPN))
```

```
}

t <- 1.15
# Create % increase variable - TRAIN
ss_train$percent_increase = rep(NA, length(ss_train$HHIDPN))
# Check wealth shock at each year
u <- unique(ss_train$HHIDPN)
k <- 1
for(i in u){
  this_person <- subset(ss_train, ss_train$HHIDPN==i)
  person_l <- length(this_person$HHIDPN)
  avg_person <- this_person$LAST_YEAR[1]
  for(j in 1:person_l){
    ss_train$percent_increase[k] <- ifelse(avg_person == 0, round(this_person$SPEND_SS[j], 3),
round((this_person$SPEND_SS[j] - avg_person/j)/avg_person/j, 3))
    if(this_person$SPEND_SS[j] < t){
      avg_person <- avg_person + this_person$SPEND_SS[j]
    }
    k <- k + 1
  }
}

t <- 1.15
# Create % increase variable - TEST
ss_test$percent_increase = rep(NA, length(ss_test$HHIDPN))
# Check wealth shock at each year
u <- unique(ss_test$HHIDPN)
k <- 1
for(i in u){
  this_person <- subset(ss_test, ss_test$HHIDPN==i)
  person_l <- length(this_person$HHIDPN)
  avg_person <- this_person$LAST_YEAR[1]
  for(j in 1:person_l){
    ss_test$percent_increase[k] <- ifelse(avg_person == 0, round(this_person$SPEND_SS[j], 3),
round((this_person$SPEND_SS[j] - avg_person/j)/avg_person/j, 3))
    if(this_person$SPEND_SS[j] < t){
      avg_person <- avg_person + this_person$SPEND_SS[j]
    }
    k <- k + 1
  }
}

t <- seq(1.1, 1.5, .05)
ter <- rep(NA, length(t))
k <- 1
for(t1 in t){
  ss_test$SHOCK2 = ifelse(ss_test$percent_increase >= t1, 1, 0)
  ss_test$SHOCK2 = ifelse(is.na(ss_test$SHOCK2), 0, ss_test$SHOCK2)

  # Make factor
  ss_test$SHOCK2 = as.factor(ss_test$SHOCK2)

  # Shock variable with 2 classes
  ss_train$SHOCK2 = ifelse(ss_train$percent_increase >= t1, 1, 0)
  ss_train$SHOCK2 = ifelse(is.na(ss_train$SHOCK2), 0, ss_train$SHOCK2)

  # Make factor
  ss_train$SHOCK2 = as.factor(ss_train$SHOCK2)

  mod <- glm(SHOCK2 ~ AGE + HEALTH_CHANGE + HBP + HEART_ATTACK + STROKE +
ARTHRITIS + DRUGS + SPECIAL_FAC + HOSPITAL + DOCTOR + NURSING_HOME,
family="binomial", data=ss_train)
  ter[k] <- performance2(mod, ss_train, .15)
  k <- k + 1
}
ter <- abs(ter - .06)
plot(t, ter, type = "l", xlab = "|delta_p", ylab = "Threshold", main = "Threshold and Error")

t <- seq(1.1, 1.5, .05)
ter <- rep(NA, length(t))
k <- 1
for(t1 in t){
  ss_test$SHOCK2 = ifelse(ss_test$percent_increase >= t1, 1, 0)
  ss_test$SHOCK2 = ifelse(is.na(ss_test$SHOCK2), 0, ss_test$SHOCK2)

  # Make factor
  ss_test$SHOCK2 = as.factor(ss_test$SHOCK2)

  # Shock variable with 2 classes
  ss_train$SHOCK2 = ifelse(ss_train$percent_increase >= t1, 1, 0)
  ss_train$SHOCK2 = ifelse(is.na(ss_train$SHOCK2), 0, ss_train$SHOCK2)

  # Make factor
  ss_train$SHOCK2 = as.factor(ss_train$SHOCK2)

  mod <- glm(SHOCK2 ~ AGE + HEALTH_CHANGE + HBP + HEART_ATTACK + STROKE +
ARTHRITIS + DRUGS + SPECIAL_FAC + HOSPITAL + DOCTOR + NURSING_HOME,
family="binomial", data=ss_train)
  ter[k] <- performance3(mod, ss_train, .15)
  k <- k + 1
}
ter <- abs(ter - .06)
plot(t, ter, type = "l", xlab = "|delta_p", ylab = "Threshold", main = "Threshold and Test Error")

t <- 1.15
# Shock variable with 2 classes
ss_test$SHOCK2 = ifelse(ss_test$percent_increase >= t, 1, 0)
ss_test$SHOCK2 = ifelse(is.na(ss_test$SHOCK2), 0, ss_test$SHOCK2)

# Make factor
ss_test$SHOCK2 = as.factor(ss_test$SHOCK2)

# Shock variable with 2 classes
ss_train$SHOCK2 = ifelse(ss_train$percent_increase >= t, 1, 0)
ss_train$SHOCK2 = ifelse(is.na(ss_train$SHOCK2), 0, ss_train$SHOCK2)

# Make factor
ss_train$SHOCK2 = as.factor(ss_train$SHOCK2)
```

```
# Logistic regression model - variables selected w/ backwards selection
mod <- glm(SHOCK2 ~ AGE + HEALTH_CHANGE + HBP + HEART_ATTACK + STROKE +
ARTHRITIS + DRUGS + SPECIAL_FAC + HOSPITAL + DOCTOR + NURSING_HOME,
family="binomial", data=ss_train)
summary(mod)

performance(mod, ss_train, .15)
performance(mod, ss_test, .15)

# Use predicted shock value to separate data
ss_train$PRED_SHOCK <- predict(mod,ss_train,type='response') >= .15
ss_test$PRED_SHOCK <- predict(mod,ss_test,type='response') >= .15

ss_train$SPEND_SS <- log(ss_train$SPEND_SS)
ss_test$SPEND_SS <- log(ss_test$SPEND_SS)

shock_yes <- subset(ss_train, ss_train$PRED_SHOCK==TRUE)
shock_no <- subset(ss_train, ss_train$PRED_SHOCK!=TRUE)

test_shock_yes <- subset(ss_test, ss_test$PRED_SHOCK==TRUE)
test_shock_no <- subset(ss_test, ss_test$PRED_SHOCK!=TRUE)

ss_train$SPEND_SS <- log(ss_train$SPEND_SS)
ss_test$SPEND_SS <- log(ss_test$SPEND_SS)

# Look for outliers
plot(shock_yes$AGE, shock_yes$SPEND_SS)

# Model
mod_yes <- lm(SPEND_SS ~ AGE + PSYCH + DRUGS + HOME_CARE + HOSPITAL +
DOCTOR + NURSING_HOME, data = shock_yes)
summary(mod_yes)

# Residuals
plot(test_shock_yes$AGE, predict(mod_yes, test_shock_yes) - test_shock_yes$SPEND_SS)

# Test MSE
sum((predict(mod_yes, test_shock_yes) - test_shock_yes$SPEND_SS)^2, na.rm = TRUE)/921

# Look for outliers
plot(shock_no$AGE, shock_no$SPEND_SS)

# Model
mod_no <- lm(SPEND_SS ~ AGE + HEALTH_CHANGE + DRUGS + HOME_CARE +
HEART_ATTACK + DOCTOR + NURSING_HOME, data = shock_no)
summary(mod_no)

# Residuals
plot(test_shock_no$AGE, predict(mod_no, test_shock_no) - test_shock_no$SPEND_SS)

# Test MSE
sum((predict(mod_no, test_shock_no) - test_shock_no$SPEND_SS)^2, na.rm = TRUE)/16591
```

```
write.csv(mod$coefficients, paste(pwd, "Method 1/log_model.csv", sep = ''))
write.csv(mod_yes$coefficients, paste(pwd, "Method 1/yes_model.csv", sep = ''))
write.csv(mod_no$coefficients, paste(pwd, "Method 1/no_model.csv", sep = ''))

# test_person: one health profile to simulate for
# baseline_spending: an estimation of adjusted spending in a non-shock year
# start_ages: age of test person at the start of the simulation
# n: number of simulations
simulation1 <- function(test_person){

  output_data <- as.data.frame(matrix(nrow = 1, ncol = 2))
  colnames(output_data) <- c("AGE", "SPENDING")
  l <- length(test_person$AGE)

  # Models
  mod1 <- read.csv(paste(pwd, "Method 1/log_model.csv", sep = ''))[,2]
  mod_yes <- read.csv(paste(pwd, "Method 1/yes_model.csv", sep = ''))[,2]
  mod_no <- read.csv(paste(pwd, "Method 1/no_model.csv", sep = ''))[,2]
  test_person <- suppressWarnings(select(test_person, -one_of("shock")))

  # Phase 1
  x <- select(test_person, c(AGE, HEALTH_CHANGE, HBP, HEART_ATTACK, STROKE,
ARTHRITIS, DRUGS,
                             SPECIAL_FAC, HOSPITAL, DOCTOR, NURSING_HOME))
  x <- cbind("Intercept" = rep(1, l), x)
  b <- mod1
  xb <- as.matrix(x)%*%b
  p <- 1/(1 + exp(-xb))
  shock <- p >= .16
  test_person <- cbind(test_person, shock)

  # Phase 2
  shock_yes <- subset(test_person, shock == TRUE)
  y <- length(shock_yes$AGE)
  x <- select(shock_yes, c(AGE, PSYCH, DRUGS, HOME_CARE, HOSPITAL,
HEART_ATTACK, DOCTOR, NURSING_HOME))
  x <- cbind("Intercept" = rep(1, y), x)
  b <- mod_yes
  xb <- as.matrix(x)%*%b
  error <- rnorm(y, 0, 1.4)
  spend_yes <- exp(xb + error)
  spend_yes <- as.data.frame(cbind(shock_yes$AGE, spend_yes))
  colnames(spend_yes) <- c("AGE", "SPENDING")

  shock_no <- subset(test_person, shock != TRUE)
  n <- length(shock_no$AGE)
  x <- select(shock_no, c(AGE, HEALTH_CHANGE, DRUGS, HOME_CARE, HEART_ATTACK,
DOCTOR, NURSING_HOME))
  x <- cbind("Intercept" = rep(1, n), x)
  b <- mod_no
  xb <- as.matrix(x)%*%b
  error <- rnorm(n, 0, 1.2)
  spend_no <- exp(xb + error)
  spend_no <- as.data.frame(cbind(shock_no$AGE, spend_no))
  colnames(spend_no) <- c("AGE", "SPENDING")

  # Combine
  output <- rbind(spend_yes, spend_no)
  output <- output[order(output$AGE),]
  output_data <- rbind(output_data, output)
  output_data <- output_data[-1, ]

  return(output_data)
}

# n: number of simulated individuals
simulation_wrapper <- function(n){
  # probabilities for start ages and years observed
  age_probs <- read.csv(paste(pwd, "Data Clean/age_probs.csv", sep = "))[,3]
  yrs_probs <- read.csv(paste(pwd, "Data Clean/yrs_probs.csv", sep = "))[,3]

  # call simulation model on each scenario
  output_data <- as.data.frame(matrix(nrow = 1, ncol = 3))
  colnames(output_data) <- c("ID", "AGE", "SPENDING")
  for(i in 1:n){
    # Pick starting age and years observed
    start_age <- sample(65:94, size = 1, replace = TRUE, prob = age_probs)
    yrs_obs <- sample(5:12, size = 1, replace = TRUE, prob = yrs_probs)
    health_profile <- create_health_profile(yrs_obs, start_age)
    this_sim <- simulation1(health_profile)
    this_sim$ID <- paste("T", i, sep = "")
    output_data <- rbind(output_data, this_sim)
  }
  output_data <- output_data[-1,]
  return(output_data)

}

# Check that it works
test <- simulation_wrapper(10)
```

Method 2

*****Simulation.ipynb*****

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv('ss_train_shock.csv')
High = pd.read_csv('High_train.csv')
Low = pd.read_csv('Low_train.csv')
```

```
data = data.drop(data.columns[0], axis=1)

data.describe()

High.describe()

Low.describe()

"""Whole Sample"""

data.columns

from fitter import Fitter, get_common_distributions, get_distributions
def find_best_distribution(data, variable_name):

    variable_data = data[variable_name].dropna()
    f = Fitter(variable_data, distributions=get_common_distributions())
    f.fit()
    f.summary()
    return f

variable_names = ['HEALTH_CHANGE', 'HBP', 'DIABETES', 'CANCER',
       'LUNGS', 'HEART_ATTACK', 'STROKE', 'PSYCH', 'ARTHRITIS', 'OUT_PT',
       'DRUGS', 'HOME_CARE', 'SPECIAL_FAC', 'HOSPITAL', 'DOCTOR',
       'NURSING_HOME']

distribution_params = {}
for variable in variable_names:
    print(f"Analyzing {variable}")
    dist = find_best_distribution(data, variable)
    distribution_params[variable] = dist.get_best(method='sumsquare_error')
    print(dist.get_best(method='sumsquare_error'))

shock_data = data[data['SHOCK2']==1]
no_shock_data = data[data['SHOCK2']==0]
variable_names = ['SPEND_SS']
spending = [shock_data, no_shock_data]
spending_results = {}
for i in range(2):
    spending_results[i] = find_best_distribution(spending[i], 'SPEND_SS')
    print(spending_results[i].get_best(method='sumsquare_error'))

from sklearn.linear_model import LogisticRegression
def modelling(data, predictors, response):
    # Create a lagged version of 'SHOCK2' to use as the target variable
    data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)


    data_clean = data.dropna(subset=predictors + ['response_next'])

    X = data_clean[predictors]
    y = data_clean['response_next']
    #class_weights = {0: 1, 1: 13}
    model = LogisticRegression(max_iter=1000, class_weight='balanced')
    model.fit(X, y)

    return model

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score

train = pd.read_csv('ss_train_shock.csv')
test = pd.read_csv('ss_test_shock.csv')
def modelling_linear(data, data_test, predictors, response):
    data = data[data['AGE']<=90]
    data_test = data_test[data_test['AGE']<=90]
    data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)
    data_test['response_next'] = data_test.groupby('HHIDPN')[response].shift(-1)
    data_clean = data.dropna(subset=predictors + ['response_next'])

    X_train = data_clean[predictors]
    y_train = data_clean['response_next']
    X_test = data_test[predictors]
    y_test = data_test['response_next']

    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Calculate residuals
    residuals = y_test - y_pred
    mean_residuals = np.mean(residuals)
    std_residuals = np.std(residuals)

    return model, mean_residuals, std_residuals

# def modelling_linear(data, predictors, response):
#     data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)
#     data_clean = data.dropna(subset=predictors + ['response_next'])

#     X = data_clean[predictors]
#     y = data_clean['response_next']

#     pipeline = Pipeline([
#         ('scaler', StandardScaler()),
#         ('model', LinearRegression())
#     ])

#     pipeline.fit(X, y)

#     return pipeline

predictors = ['AGE', 'HEALTH_CHANGE', 'HBP', 'STROKE', 'ARTHRITIS', 'DRUGS',
'SPECIAL_FAC', 'HOSPITAL', 'DOCTOR', 'NURSING_HOME']
response = 'SHOCK2'
shock_model = modelling(data, predictors, response)

yes_pred = ['AGE', 'PSYCH', 'DRUGS', 'HOME_CARE', 'HOSPITAL', 'DOCTOR',
'NURSING_HOME']
spend_response = 'SPEND_SS'
yes_model, yes_mean, yes_resid= modelling_linear(train, test, yes_pred, spend_response)

no_pred = ['AGE', 'HEALTH_CHANGE', 'DRUGS', 'HOME_CARE', 'DOCTOR',
'NURSING_HOME']
no_model, no_mean, no_resid = modelling_linear(train, test, no_pred, spend_response)

from scipy import stats

def simulate_individual_data(num_individuals, start_age, end_age, distribution_params):
    simulated_complete_data = pd.DataFrame()

    # Simulate data for each individual
    for person_id in range(1, num_individuals + 1):
        # Initialize a DataFrame for the current individual's data across the specified age range
        individual_data = pd.DataFrame({
            'AGE': range(start_age, end_age + 1),
            'HHIDPN': person_id  # Assign the unique person ID to each row
        })

        # Simulate the data for each variable according to the distribution parameters
        for variable, params_dict in distribution_params.items():
            dist_name, params = next(iter(params_dict.items()))  # Get the distribution and its
parameters
            dist = getattr(stats, dist_name)
            individual_data[variable] = dist.rvs(size=end_age - start_age + 1, **params)

        simulated_complete_data = pd.concat([simulated_complete_data, individual_data],
ignore_index=True)

    return simulated_complete_data

# Example usage
num_individuals = 10000

stacked_simulation = simulate_individual_data(num_individuals, 65, 90, distribution_params)
stacked_simulation

def add_shock_predictions(simulated_data, shock_model, predictors):

    predicted_shocks = shock_model.predict(simulated_data[predictors])

    simulated_data['SHOCK2'] = predicted_shocks

    return simulated_data

simulated_data_with_shock = add_shock_predictions(stacked_simulation, shock_model,
predictors)
```

```
simulated_data_with_shock

def add_spend_ss_predictions(simulated_data, yes_model, no_model, yes_pred, no_pred):
    simulated_data['SPEND_SS'] = 0.0

    shock_indices = simulated_data[simulated_data['SHOCK2'] == 1].index
    yes_predictions = yes_model.predict(simulated_data.loc[shock_indices, yes_pred])
    yes_random_residuals = np.random.normal(loc=yes_mean, scale=yes_resid,
size=len(shock_indices))
    simulated_data.loc[shock_indices, 'SPEND_SS'] = yes_predictions - yes_random_residuals

    no_shock_indices = simulated_data[simulated_data['SHOCK2'] == 0].index
    no_predictions = no_model.predict(simulated_data.loc[no_shock_indices, no_pred])
    no_random_residuals = np.random.normal(loc=no_mean, scale=no_resid,
size=len(no_shock_indices))
    simulated_data.loc[no_shock_indices, 'SPEND_SS'] = no_predictions - no_random_residuals

    return simulated_data

# Example usage
simulated_data_final = add_spend_ss_predictions(simulated_data_with_shock, yes_model,
no_model, yes_pred, no_pred)
simulated_data_final

"""High Cluster"""

data = pd.read_csv('High_train.csv')

from fitter import Fitter, get_common_distributions, get_distributions
def find_best_distribution(data, variable_name):

    variable_data = data[variable_name].dropna()
    f = Fitter(variable_data, distributions=get_common_distributions())
    f.fit()
    f.summary()
    return f

variable_names = ['HEALTH_CHANGE', 'HBP', 'DIABETES', 'CANCER',
       'LUNGS', 'HEART_ATTACK', 'STROKE', 'PSYCH', 'ARTHRITIS', 'OUT_PT',
       'DRUGS', 'HOME_CARE', 'SPECIAL_FAC', 'HOSPITAL', 'DOCTOR',
       'NURSING_HOME']

distribution_params = {}
for variable in variable_names:
    print(f"Analyzing {variable}")
    dist = find_best_distribution(data, variable)
    distribution_params[variable] = dist.get_best(method='sumsquare_error')
    print(dist.get_best(method='sumsquare_error'))

shock_data = data[data['SHOCK2']==1]
no_shock_data = data[data['SHOCK2']==0]
variable_names = ['SPEND_SS']
spending = [shock_data, no_shock_data]
```

```
spending_results = {}
for i in range(2):
    spending_results[i] = find_best_distribution(spending[i], 'SPEND_SS')
    print(spending_results[i].get_best(method='sumsquare_error'))

from sklearn.linear_model import LogisticRegression
def modelling(data, predictors, response):
    # Create a lagged version of 'SHOCK2' to use as the target variable
    data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)


    data_clean = data.dropna(subset=predictors + ['response_next'])

    X = data_clean[predictors]
    y = data_clean['response_next']
    #class_weights = {0: 1, 1: 13}
    model = LogisticRegression(max_iter=1000, class_weight='balanced')
    model.fit(X, y)

    return model

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

train = pd.read_csv('ss_train_shock.csv')
test = pd.read_csv('ss_test_shock.csv')
def modelling_linear(data, data_test, predictors, response):
    data = data[data['AGE']<=90]
    data_test = data_test[data_test['AGE']<=90]
    data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)
    data_test['response_next'] = data_test.groupby('HHIDPN')[response].shift(-1)
    data_clean = data.dropna(subset=predictors + ['response_next'])

    X_train = data_clean[predictors]
    y_train = data_clean['response_next']
    X_test = data_test[predictors]
    y_test = data_test['response_next']

    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Calculate residuals
    residuals = y_test - y_pred
    mean_residuals = np.mean(residuals)
    std_residuals = np.std(residuals)

    return model, mean_residuals, std_residuals

predictors = ['AGE', 'HEALTH_CHANGE', 'HBP', 'STROKE', 'ARTHRITIS', 'DRUGS',
'SPECIAL_FAC', 'HOSPITAL', 'DOCTOR', 'NURSING_HOME']
response = 'SHOCK2'
shock_model = modelling(data, predictors, response)

yes_pred = ['AGE', 'PSYCH', 'DRUGS', 'HOME_CARE', 'HOSPITAL', 'DOCTOR',
'NURSING_HOME']
spend_response = 'SPEND_SS'
yes_model, yes_mean, yes_resid = modelling_linear(train, test, yes_pred, spend_response)

no_pred = ['AGE', 'HEALTH_CHANGE', 'DRUGS', 'HOME_CARE', 'DOCTOR',
'NURSING_HOME']
no_model, no_mean, no_resid = modelling_linear(train, test, no_pred, spend_response)

from scipy import stats

def simulate_individual_data(num_individuals, start_age, end_age, distribution_params):
    simulated_complete_data = pd.DataFrame()

    # Simulate data for each individual
    for person_id in range(1, num_individuals + 1):
        # Initialize a DataFrame for the current individual's data across the specified age range
        individual_data = pd.DataFrame({
            'AGE': range(start_age, end_age + 1),
            'HHIDPN': person_id  # Assign the unique person ID to each row
        })

        # Simulate the data for each variable according to the distribution parameters
        for variable, params_dict in distribution_params.items():
            dist_name, params = next(iter(params_dict.items()))  # Get the distribution and its
parameters
            dist = getattr(stats, dist_name)
            individual_data[variable] = dist.rvs(size=end_age - start_age + 1, **params)

        simulated_complete_data = pd.concat([simulated_complete_data, individual_data],
ignore_index=True)

    return simulated_complete_data

# Example usage
num_individuals = 10000

stacked_simulation = simulate_individual_data(num_individuals, 65, 90, distribution_params)
stacked_simulation

def add_shock_predictions(simulated_data, shock_model, predictors):

    predicted_shocks = shock_model.predict(simulated_data[predictors])

    simulated_data['SHOCK2'] = predicted_shocks

    return simulated_data

simulated_data_with_shock = add_shock_predictions(stacked_simulation, shock_model,
predictors)
```

```
simulated_data_with_shock

def add_spend_ss_predictions(simulated_data, yes_model, no_model, yes_pred, no_pred):
   simulated_data['SPEND_SS'] = 0.0

   shock_indices = simulated_data[simulated_data['SHOCK2'] == 1].index
   yes_predictions = yes_model.predict(simulated_data.loc[shock_indices, yes_pred])
   yes_random_residuals = np.random.normal(loc=yes_mean, scale=yes_resid,
size=len(shock_indices))
   simulated_data.loc[shock_indices, 'SPEND_SS'] = yes_predictions - yes_random_residuals

   no_shock_indices = simulated_data[simulated_data['SHOCK2'] == 0].index
   no_predictions = no_model.predict(simulated_data.loc[no_shock_indices, no_pred])
   no_random_residuals = np.random.normal(loc=no_mean, scale=no_resid,
size=len(no_shock_indices))
   simulated_data.loc[no_shock_indices, 'SPEND_SS'] = no_predictions - no_random_residuals

   return simulated_data

# Example usage
simulated_data_final_high = add_spend_ss_predictions(simulated_data_with_shock,
yes_model, no_model, yes_pred, no_pred)
simulated_data_final

"""Low Cluster"""

data = pd.read_csv('Low_train.csv')

from fitter import Fitter, get_common_distributions, get_distributions
def find_best_distribution(data, variable_name):

   variable_data = data[variable_name].dropna()
   f = Fitter(variable_data, distributions=get_common_distributions())
   f.fit()
   f.summary()
   return f

variable_names = ['HEALTH_CHANGE', 'HBP', 'DIABETES', 'CANCER',
     'LUNGS', 'HEART_ATTACK', 'STROKE', 'PSYCH', 'ARTHRITIS', 'OUT_PT',
     'DRUGS', 'HOME_CARE', 'SPECIAL_FAC', 'HOSPITAL', 'DOCTOR',
     'NURSING_HOME']

distribution_params = {}
for variable in variable_names:
   print(f"Analyzing {variable}")
   dist = find_best_distribution(data, variable)
   distribution_params[variable] = dist.get_best(method='sumsquare_error')
   print(dist.get_best(method='sumsquare_error'))

shock_data = data[data['SHOCK2']==1]
no_shock_data = data[data['SHOCK2']==0]
variable_names = ['SPEND_SS']
spending = [shock_data, no_shock_data]
```

```
spending_results = {}
for i in range(2):
    spending_results[i] = find_best_distribution(spending[i], 'SPEND_SS')
    print(spending_results[i].get_best(method='sumsquare_error'))

from sklearn.linear_model import LogisticRegression
def modelling(data, predictors, response):
    # Create a lagged version of 'SHOCK2' to use as the target variable
    data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)


    data_clean = data.dropna(subset=predictors + ['response_next'])

    X = data_clean[predictors]
    y = data_clean['response_next']
    #class_weights = {0: 1, 1: 13}
    model = LogisticRegression(max_iter=1000, class_weight='balanced')
    model.fit(X, y)

    return model

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

train = pd.read_csv('ss_train_shock.csv')
test = pd.read_csv('ss_test_shock.csv')
def modelling_linear(data, data_test, predictors, response):
    data = data[data['AGE']<=90]
    data_test = data_test[data_test['AGE']<=90]
    data['response_next'] = data.groupby('HHIDPN')[response].shift(-1)
    data_test['response_next'] = data_test.groupby('HHIDPN')[response].shift(-1)
    data_clean = data.dropna(subset=predictors + ['response_next'])

    X_train = data_clean[predictors]
    y_train = data_clean['response_next']
    X_test = data_test[predictors]
    y_test = data_test['response_next']

    model = LinearRegression()
    model.fit(X_train, y_train)

    # Predict on the test set
    y_pred = model.predict(X_test)

    # Calculate residuals
    residuals = y_test - y_pred
    mean_residuals = np.mean(residuals)
    std_residuals = np.std(residuals)

    return model, mean_residuals, std_residuals

predictors = ['AGE', 'HEALTH_CHANGE', 'HBP', 'STROKE', 'ARTHRITIS', 'DRUGS',
'SPECIAL_FAC', 'HOSPITAL', 'DOCTOR', 'NURSING_HOME']
response = 'SHOCK2'
shock_model = modelling(data, predictors, response)

yes_pred = ['AGE', 'PSYCH', 'DRUGS', 'HOME_CARE', 'HOSPITAL', 'DOCTOR',
'NURSING_HOME', 'CANCER']
spend_response = 'SPEND_SS'
yes_model, yes_mean, yes_resid = modelling_linear(train, test, yes_pred, spend_response)

no_pred = ['AGE', 'HEALTH_CHANGE', 'DRUGS', 'HOME_CARE', 'DOCTOR',
'NURSING_HOME', 'CANCER']
no_model, no_mean, no_resid = modelling_linear(train, test, no_pred, spend_response)

from scipy import stats

def simulate_individual_data(num_individuals, start_age, end_age, distribution_params):
    simulated_complete_data = pd.DataFrame()

    # Simulate data for each individual
    for person_id in range(1, num_individuals + 1):
        # Initialize a DataFrame for the current individual's data across the specified age range
        individual_data = pd.DataFrame({
            'AGE': range(start_age, end_age + 1),
            'HHIDPN': person_id  # Assign the unique person ID to each row
        })

        # Simulate the data for each variable according to the distribution parameters
        for variable, params_dict in distribution_params.items():
            dist_name, params = next(iter(params_dict.items()))  # Get the distribution and its
parameters
            dist = getattr(stats, dist_name)
            individual_data[variable] = dist.rvs(size=end_age - start_age + 1, **params)

        simulated_complete_data = pd.concat([simulated_complete_data, individual_data],
ignore_index=True)

    return simulated_complete_data

# Example usage
num_individuals = 10000

stacked_simulation = simulate_individual_data(num_individuals, 65, 90, distribution_params)
stacked_simulation

def add_shock_predictions(simulated_data, shock_model, predictors):

    predicted_shocks = shock_model.predict(simulated_data[predictors])

    simulated_data['SHOCK2'] = predicted_shocks

    return simulated_data

simulated_data_with_shock = add_shock_predictions(stacked_simulation, shock_model,
predictors)
```

```
simulated_data_with_shock

def add_spend_ss_predictions(simulated_data, yes_model, no_model, yes_pred, no_pred):
    simulated_data['SPEND_SS'] = 0.0

    shock_indices = simulated_data[simulated_data['SHOCK2'] == 1].index
    yes_predictions = yes_model.predict(simulated_data.loc[shock_indices, yes_pred])
    yes_random_residuals = np.random.normal(loc=yes_mean, scale=yes_resid,
size=len(shock_indices))
    simulated_data.loc[shock_indices, 'SPEND_SS'] = yes_predictions - yes_random_residuals

    no_shock_indices = simulated_data[simulated_data['SHOCK2'] == 0].index
    no_predictions = no_model.predict(simulated_data.loc[no_shock_indices, no_pred])
    no_random_residuals = np.random.normal(loc=no_mean, scale=no_resid,
size=len(no_shock_indices))
    simulated_data.loc[no_shock_indices, 'SPEND_SS'] = no_predictions - no_random_residuals

    return simulated_data

# Example usage
simulated_data_final_low = add_spend_ss_predictions(simulated_data_with_shock, yes_model,
no_model, yes_pred, no_pred)
simulated_data_final

data = pd.read_csv('ss_train_shock.csv')

def plot_average_spending_by_age(simulated_data, simulated_data_low,
simulated_data_high):
    average_spending_by_age = simulated_data.groupby('AGE')['SPEND_SS'].mean()
    average_spending_by_age_low = simulated_data_low.groupby('AGE')['SPEND_SS'].mean()
    average_spending_by_age_high =
simulated_data_high.groupby('AGE')['SPEND_SS'].mean()
    sample_spending = data[data['AGE']<=90].groupby('AGE')['SPEND_SS'].mean()
    plt.figure(figsize=(10, 6))
    average_spending_by_age.plot(kind='line', marker='x', label='General Simulated
Spending',color='blue')
    average_spending_by_age_low.plot(kind='line', marker='x', label='Simulated Spending - Low
Cluster',color='red')
    average_spending_by_age_high.plot(kind='line', marker='x', label='Simulated Spending -
High Cluster',color='green')
    sample_spending.plot(kind='line', marker='x', label='Sample Spending',color='black')
    plt.title('Average Spending by Age')
    plt.xlabel('Age')
    plt.ylabel('Average Spending')
    plt.grid(True)
    plt.legend()
    plt.show()


# Example usage with the simulated data
plot_average_spending_by_age(simulated_data_final, simulated_data_final_low,
simulated_data_final_high)
```

```
def plot_median_spending_by_age(simulated_data, simulated_data_low, simulated_data_high):
    average_spending_by_age = simulated_data.groupby('AGE')['SPEND_SS'].median()
    average_spending_by_age_low =
simulated_data_low.groupby('AGE')['SPEND_SS'].median()
    average_spending_by_age_high =
simulated_data_high.groupby('AGE')['SPEND_SS'].median()
    sample_spending = data[data['AGE']<=90].groupby('AGE')['SPEND_SS'].median()
    plt.figure(figsize=(10, 6))
    average_spending_by_age.plot(kind='line', marker='x', label='General Simulated
Spending',color='blue')
    average_spending_by_age_low.plot(kind='line', marker='x', label='Simulated Spending - Low
Cluster',color='red')
    average_spending_by_age_high.plot(kind='line', marker='x', label='Simulated Spending -
High Cluster',color='green')
    sample_spending.plot(kind='line', marker='x', label='Sample Spending',color='black')
    plt.title('Median Spending by Age')
    plt.xlabel('Age')
    plt.ylabel('Median Spending')
    plt.grid(True)
    plt.legend()
    plt.show()


# Example usage with the simulated data
plot_median_spending_by_age(simulated_data_final, simulated_data_final_low,
simulated_data_final_high)
```

*****Unsupervised Learning.ipynb*****

```
from sklearn.manifold import TSNE
from sklearn.metrics import f1_score
from scipy.stats import mode
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.cluster import DBSCAN

import numpy as np
import scipy
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import sklearn
from sklearn.model_selection import train_test_split

import matplotlib.cm as cm
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
pd.set_option('display.max_columns', None)

ss_train_shock = pd.read_csv('ss_train_shock.csv')
```

```
ss_test_shock = pd.read_csv('ss_test_shock.csv')
ss_train_shock = ss_train_shock.dropna()
ss_test_shock = ss_test_shock.dropna()
ss_train_shock

ss_train_shock.describe()

train = ss_train_shock.drop(ss_train_shock.columns[0], axis=1)
X_train = train.drop(['percent_increase','WEIGHT','HHIDPN','percent_increase','LAST_YEAR',
'SPEND_SS','SHOCK2'], axis=1)
y_train = train['SHOCK2']
other_train = ss_train_shock[['percent_increase','WEIGHT','percent_increase','LAST_YEAR',
'SPEND_SS','SHOCK2','HHIDPN']]

test = ss_test_shock.drop(ss_test_shock.columns[0], axis=1)
X_test = test.drop(['percent_increase','WEIGHT','HHIDPN','percent_increase','LAST_YEAR',
'SPEND_SS','SHOCK2'], axis=1)
y_test = test['SHOCK2']
other_test = ss_test_shock[['percent_increase','WEIGHT','percent_increase','LAST_YEAR',
'SPEND_SS','SHOCK2','HHIDPN']]

X_train

X_test

other_train

y_train

y_test

other_test

samples = pd.concat([X_train,X_test])

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_norm = scaler.fit_transform(X_train)

X = X_train_norm
X_test_norm = scaler.transform(X_test)

samples_norm = scaler.transform(samples)

from sklearn.manifold import TSNE
from sklearn.metrics import f1_score
from scipy.stats import mode
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

random_state = 42
prp_var = [50,200,300,400,500,600,700]
for i in prp_var:
    tsne = TSNE(n_components=2, init='random', random_state=random_state,
perplexity=i,n_jobs=-1)
```

```
    X_proj = tsne.fit_transform(samples_norm)
    X_test_tsne = X_proj[-16260:]
    X_train_tsne = X_proj[:37772]
    print(X_test_tsne.shape)
    print(X_train_tsne.shape)
    X_proj = X_train_tsne

    fig = plt.figure(figsize=(12, 6))
    plt.scatter(X_train_tsne[:, 0], X_train_tsne[:, 1], \
            c=[cm.Spectral(float(i)*3 / 8) for i in y_train])
    plt.title("t-SNE transformed data " + str(i))

prp_var = [800,1000,1200,1400,1600,1800]
for i in prp_var:
    tsne = TSNE(n_components=2, init='random', random_state=random_state,
perplexity=i,n_jobs=-1)
    X_proj = tsne.fit_transform(samples_norm)
    X_test_tsne = X_proj[-16260:]
    X_train_tsne = X_proj[:37772]
    print(X_test_tsne.shape)
    print(X_train_tsne.shape)
    X_proj = X_train_tsne

    fig = plt.figure(figsize=(12, 6))
    plt.scatter(X_train_tsne[:, 0], X_train_tsne[:, 1], \
            c=[cm.Spectral(float(i)*3 / 8) for i in y_train])
    plt.title("t-SNE transformed data " + str(i))

prp_var = [2000,2400,2800]
random_state = 42
for i in prp_var:
    tsne = TSNE(n_components=2, init='random', random_state=random_state,
perplexity=i,n_jobs=-1)
    X_proj = tsne.fit_transform(samples_norm)
    X_test_tsne = X_proj[-16260:]
    X_train_tsne = X_proj[:37772]
    print(X_test_tsne.shape)
    print(X_train_tsne.shape)
    X_proj = X_train_tsne

    fig = plt.figure(figsize=(12, 6))
    plt.scatter(X_train_tsne[:, 0], X_train_tsne[:, 1], \
            c=[cm.Spectral(float(i)*3 / 8) for i in y_train])
    plt.title("t-SNE transformed data " + str(i))

from sklearn.manifold import TSNE
from sklearn.metrics import f1_score
from scipy.stats import mode
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

random_state = 42
prp_var = 1800

# # Transform  dimensional data to 2 dimensions
```

```
tsne = TSNE(n_components=2, init='random', random_state=random_state,
perplexity=prp_var,n_jobs=-1)
X_proj = tsne.fit_transform(samples_norm)
X_test_tsne = X_proj[-16260:]
X_train_tsne = X_proj[:37772]
print(X_test_tsne.shape)
print(X_train_tsne.shape)
X_proj = X_train_tsne

fig = plt.figure(figsize=(12, 6))
plt.scatter(X_train_tsne[:, 0], X_train_tsne[:, 1], \
            c=[cm.Spectral(float(i)*3 / 8) for i in y_train])
plt.title("t-SNE Transformed Data, Perplexity = 1800")

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

eps_values = [3, 4, 5, 6, 7, 8]

for eps in eps_values:
    dbs = DBSCAN(eps=eps)
    clusters = dbs.fit_predict(X_proj)

    # Calculate the unique clusters and their counts
    uq = np.unique(clusters)
    num_clusters = len(uq)

    # Permute the labels based on the mode of the original labels
    labels = np.zeros_like(clusters)
    for i in uq:  # Change range(10) to unique values in clusters to handle varying cluster
numbers
        mask = (clusters == i)
        # Only assign a label if there are any points in the cluster
        if np.any(mask):
            labels[mask] = mode(y_train[mask])[0][0]

    # Plotting
    fig = plt.figure(figsize=(12, 6))
    plt.scatter(X_proj[:, 0], X_proj[:, 1], c=[cm.Spectral(float(i) / num_clusters) for i in clusters])
    plt.title(f"t-SNE transformed data with EPS = {eps}")

    # Calculate and annotate cluster centers
    centers = {}
    for i in uq:
        if i != -1:  # Skip noise points
            center = X_proj[clusters == i].mean(axis=0)
            centers[i] = center
            plt.text(center[0], center[1], str(i), ha="center", va="center", size=15,
                     bbox=dict(boxstyle="circle,pad=0.1", fc="lightblue", ec="steelblue", lw=2))

    plt.show()  # Show the plot for each value of eps

    # Print the number of clusters
    print(f"Number of clusters for eps={eps}: {num_clusters}")
    # Compute and print the F1 score
    f1 = f1_score(y_train, labels, average='micro')
    print(f"F1 Score for eps={eps}: {f1}")

eps = 7

dbs = DBSCAN(eps = eps)

#X_proj = TSNE_DONE[perp][:8000]
clusters = dbs.fit_predict(X_proj)

uq = np.unique(clusters)
num_clusters = len(uq)



#plt.scatter(kmeans.cluster_centers_[:, 0], sc.cluster_centers_[:, 1], s=300, c='black')


# Permute the labels
labels = np.zeros_like(clusters)
for i in range(10):
    mask = (clusters == i)
    labels[mask] = mode(y_train[mask])[0]



fig = plt.figure(figsize=(12, 6))
plt.scatter(X_proj[:, 0], X_proj[:, 1], \
          c=[cm.Spectral(float(i) / num_clusters) for i in clusters])
plt.title("t-SNE Transformed Data for EPS = " + str(eps))
centers = {}
counts = {}
for i in uq:
    count = 0
    center = np.zeros(2)
    for c in range(0, len(clusters)):
        if (clusters[c] != i):
            continue
        count = count + 1
        center = center + X_proj[c]
    if (count > 0):
        counts[i] = count
        centers[i] = center / count

print(num_clusters)

for i in centers.keys():
    if i > 25:
        break
    loc = centers[i]
    plt.text(loc[0], loc[1], str(i), ha="center", va="center", size=15,
            bbox=dict(boxstyle="circle,pad=0.1",
                      fc="lightblue", ec="steelblue", lw=2))
```

```
  #plt.show()
# Compute the accuracy
f1_score(y_train, labels, average='micro')

unique_clusters = np.unique(clusters)


clu_defect = {i: 0 for i in unique_clusters}
clu_total = {i: 0 for i in unique_clusters}


for i, cluster_id in enumerate(clusters):
    if y_train[i]:
        clu_defect[cluster_id] += 1
    clu_total[cluster_id] += 1

print(clu_total)
print(clu_defect)

# Calculate and print defect percentage for each actual cluster (excluding noise)
for i in unique_clusters:
    if i != -1:  # Skip noise
        total = clu_total[i]
        defect = clu_defect[i]
        defect_percentage = defect / total if total > 0 else 0
        print(f"{i}th cluster has a total of {total} and a {defect_percentage:.2%} percentage to
experience wealth shock")


clu_defect_perc = {i: (clu_defect[i] / clu_total[i] if clu_total[i] > 0 else 0) for i in unique_clusters if i
!= -1}
for i in sorted(clu_defect_perc.keys(), key=clu_defect_perc.get):
    print(f"{i}: {clu_defect_perc[i]:.2%}")

print(sum(y_train)/37772)
print(sum(y_test)/16260)

test_clusters = []

#X_test_tsne = TSNE_DONE[perp][-2000:]

for i in X_test_tsne:
    min = np.linalg.norm(X_train_tsne[0] - i)
    value = 0
    for j in range(0,len(X_train_tsne)):
        if np.linalg.norm(X_train_tsne[j] - i) >= min:
            continue
        min = np.linalg.norm(X_train_tsne[j] - i)
        value = j
    test_clusters.append(clusters[value])
test_clusters = np.array(test_clusters)

plt.scatter(X_test_tsne[:, 0], X_test_tsne[:, 1], \
          c=[cm.Spectral(float(i) / num_clusters) for i in test_clusters])
```

```
for i in centers.keys():
    if i > 25:
        break
    loc = centers[i]
    plt.text(loc[0], loc[1], str(i), ha="center", va="center", size=15,
            bbox=dict(boxstyle="circle,pad=0.1",
                      fc="lightblue", ec="steelblue", lw=2))

X_train.reset_index(drop=True, inplace=True)
X_test.reset_index(drop=True, inplace=True)
y_test.reset_index(drop=True, inplace=True)

other_train.reset_index(drop=True, inplace=True)
other_test.reset_index(drop=True, inplace=True)

X_train_clusters = X_train.copy()
X_train_clusters["cluster"] = clusters
Shock_train_clusters = pd.concat([X_train_clusters, y_train, other_train], axis=1)
Shock_train_clusters

X_test_clusters = X_test.copy()
X_test_clusters["cluster"] = test_clusters
Shock_test_clusters = pd.concat([X_test_clusters, y_test, other_test], axis=1)
Shock_test_clusters

clu_test_defect = {i:0 for i in range(0,num_clusters)}
clu_test_total = {i:0 for i in range(0,num_clusters)}
for i in range(0,len(test_clusters)):
    if y_test[i]:
        clu_test_defect[test_clusters[i]] += 1
    clu_test_total[test_clusters[i]] += 1

print(clu_test_total)
print(clu_test_defect)

for i in range(num_clusters):
    print("{:d}th cluster is: ".format(i))
    print(X_test.iloc[i])
    print("And has a total of {:d} and a {:f} chance to experience wealth
shock".format(clu_test_total[i],clu_test_defect[i]/clu_test_total[i]))
    print("-------------------------------------------------------")



clu_test_defect_perc = {i:(clu_test_defect[i]/clu_test_total[i]) for i in range(0,num_clusters)}
for i in (sorted(clu_test_defect_perc.keys(), key=clu_test_defect_perc.get)):
    print("{:d}: {:f}".format(i, clu_test_defect_perc[i]))

print("--------------------")

for i in clu_test_defect_perc.keys():
    print("{:d}: {:f}".format(i, clu_defect_perc[i] - clu_test_defect_perc[i]))

c_train_data = []
```

```
for i in range(num_clusters):
    c_train_data.append(Shock_train_clusters[Shock_train_clusters["cluster"] ==
i].drop("cluster",axis=1))

for i in range(len(c_train_data)):
    print("CLUSTER {:d}".format(i))
    display(c_train_data[i])
    print("-" * 3600)

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

columns = c_train_data[0].columns

for col in columns:
    combined_df = pd.DataFrame()

    for i, df in enumerate(c_train_data):
        temp_df = df[[col]].copy()
        temp_df['DataFrame'] = f'Cluster{i+1}'
        combined_df = pd.concat([combined_df, temp_df], ignore_index=True)

    plt.figure(figsize=(10, 6))
    sns.boxplot(x='DataFrame', y=col, data=combined_df)
    plt.title(f'Boxplot of {col} across different DataFrames')
    plt.show()

df_names = ['Low', 'High']
columns = c_train_data[0].columns
means_df = pd.DataFrame(index=columns, columns=df_names)

for col in columns:
    for name, df in zip(df_names, c_train_data):
        means_df.loc[col, name] = df[col].mean()

print(means_df)

#High cluster has a total of 785 and a 2.17% percentage to experience wealth shock

#Medium cluster has a total of 27466 and a 1.11% percentage to experience wealth shock

#Low cluster has a total of 6993 and a 0.46% percentage to experience wealth shock

#Age are very similar across all three clsuters
#High cluster experience significantly more drastic health declines and hospitalizations than
medium cluster, the same applies to medium and low clusters
#Low cluster has none of the specified diseases
#High cluster has cancer rate of 100%
#Home care, Cancer and HBP are the most significant factors differentiating high, mdeium and
low clusters

c_train_data[0].to_csv('Low_train.csv', index=False)
```

```
c_train_data[1].to_csv('High_train.csv', index=False)

from scipy.spatial import distance_matrix
from skbio import DistanceMatrix
from skbio.stats.distance import permanova
from skbio.stats.distance import anosim

c_test_data = []

for i in range(num_clusters):
    c_test_data.append(Shock_test_clusters[Shock_test_clusters["cluster"] ==
i].drop("cluster",axis=1))

for i in range(len(c_test_data)):
    print("CLUSTER {:d}".format(i))
    display(c_test_data[i])
    print("-" * 3600)

df_names = ['Low', 'High']
columns = c_test_data[0].columns
means_df = pd.DataFrame(index=columns, columns=df_names)

for col in columns:
    for name, df in zip(df_names, c_test_data):
        means_df.loc[col, name] = df[col].mean()

print(means_df)

c_test_data[0].to_csv('Low_test.csv', index=False)
c_test_data[1].to_csv('High_test.csv', index=False)

sample_size = 5000
if len(X_train_norm) > sample_size:
    indices = np.random.choice(len(X_train_norm), size=sample_size, replace=False)
    X_train_sampled = X_train_norm[indices]
else:
    X_train_sampled = X_train_norm

dm_sampled = DistanceMatrix(distance_matrix(X_train_sampled, X_train_sampled))
clusters_sampled = clusters[indices]
result_anosim = anosim(dm_sampled, clusters_sampled)
print(result_anosim)

result_permanova = permanova(dm_sampled, clusters_sampled)
print(result_permanova)
```

main

*****main.R*****

```
########################      PACKAGES        ################################

#install.packages(tidyverse)
#install.packages(priceR)
#install.packages(gtools)
#install.packages(stringi)
#install.packages(ggplot2)
#install.packages(dpylr)
#install.packages(gridExtra)

library(tidyverse)
library(priceR)
library(tidyverse)
library(gtools)
library(stringi)
library(ggplot2)
library(dplyr)
library(gridExtra)

##################      SET WORKING DIRECTORY       ##########################

# To location of Modeling_Liquidity_Needs

pwd <- setwd("/Users/libraryuser/Downloads/Modeling-Retirees-Liquidity-Needs-main")


########################      DATA CLEAN        ################################

source(paste(pwd, "/Data Clean/data_clean.R", sep = ''))

########################        EDA          ################################

source(paste(pwd, "/Data Clean/eda.R", sep = ''))

########################      METHOD 1         ################################

source(paste(pwd, "/Method 1/method1.R", sep = ''))
source(paste(pwd, "/Method 1/eval_sim.R", sep = ''))

########################      METHOD 2        ################################

# In Python
```