Towards a Learning Health System: Using Reinforcement Learning to Optimize Treatment Decisions in Sepsis Patients

by

Jason Neal Kennedy

B.S. Biomedical Engineering, Washington University, 2004

M.S. Biomedical Engineering, Saint Louis University, 2008

Submitted to the Graduate Faculty of the

Graduate School of Public Health in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2021

UNIVERSITY OF PITTSBURGH

GRADUATE SCHOOL OF PUBLIC HEALTH

This thesis was presented

by

Jason Neal Kennedy

It was defended on

April 26, 2021

and approved by

Thesis Committee Chair: Jeanine Buchanich, MEd, MPH, PhD, Research Associate Professor, Department of Biostatistics, Graduate School of Public Health, University of Pittsburgh

Thesis Committee Co-Chair: Lu Tang, PhD, Assistant Professor, Department of Biostatistics Graduate School of Public Health, University of Pittsburgh

Committee Member: Jenna Colavincenzo Carlson, PhD, Assistant Professor Department of Biostatistics, University of Pittsburgh

Committee Member: Chung-Chou H. Chang, PhD, Professor, Departments of Medicine and Biostatistics, University of Pittsburgh

Committee Member: Christopher W. Seymour, MD, MSc, Associate Professor, Department of Critical Care Medicine, School of Medicine, University of Pittsburgh

Committee Member: Ada Youk, PhD, Associate Professor Department of Biostatistics, University of Pittsburgh

Towards a Learning Health System: Using Reinforcement Learning to Optimize Treatment Decisions in Sepsis Patients

Jason Neal Kennedy, MS

University of Pittsburgh, 2021

Abstract

Sepsis, a syndrome defined by dysregulated host immune response to infection and acute organ dysfunction, affects 1.7 million Americans annually and accounts for more than 1 in 5 deaths worldwide. International clinical practice guidelines recommend early sepsis identification and a one-size-fits-all treatment bundle of broad spectrum anti-microbials, intravenous (IV) fluids, and vasopressors. Emerging evidence suggests, however, that an individualized, precision treatment approach may improve early sepsis care.

We developed a precision treatment policy for IV fluids and vasopressors in early sepsis using model-free Q-learning in clinical Electronic Health Record (EHR) data. We analyzed 30,687 patients presenting with Sepsis-3 within 6 hours of hospital arrival using features in the EHR from 14 UPMC hospitals between 2013-2017. We extracted 38 model features (e.g., demographics, vital signs, laboratory variables) in 4-hour timesteps from hospital arrival until 48-hours after estimated sepsis onset. We defined patient states using K-means clustering and defined an action space that was a 5×5 matrix of IV fluid and vasopressor doses, including no drug administered and doses divided into observed dose quartiles. Awards and penalties were applied maximizing 90-day patient survival. We assessed model performance using weighted importance sampling and demonstrated that the expected value of the Q-learning model treatment policy was significantly higher than that of human clinicians. We demonstrated that model performance in patient- and

hospital- level subgroups mostly greatly exceeded clinician performance among subgroups of older patients, those with higher illness severity, and history of recent hospitalization.

In conclusion, we demonstrate that patients with early sepsis treated per a precision treatment policy of IV fluids and vasopressors developed using model-free Q-learning may have improved 90-day survival compared to those treated by standard protocol. Precision sepsis treatment strategies should be explored further, including among key clinical subgroups.

Public Health Significance: Sepsis is an important public health problem; even small care improvements may make a significant global impact. We demonstrate that a precision treatment strategy using IV fluids and vasopressors may improve sepsis patient survival. These results serve as the foundation for future study, including the development of clinical decision support tools for use at the bedside.

Table of Contents

List of Tables

List of Figures

List of Equations

1.0 Introduction

Sepsis, a dysregulated host immune response to infection resulting in acute organ dysfunction, affects 1.7 million Americans annually and accounts for more than 1 in 5 deaths worldwide (Seymour et al., 2016; Rudd et al., 2020). International clinical practice guidelines recommend early sepsis identification, treatment with broad spectrum anti-microbials, and prompt reversal of hypotension using intravenous (IV) fluids and vasopressors (Levy and Rhodes, 2018). Despite these recommendations, however, the optimal IV fluid and vasopressor dose and timing is unknown and sepsis care remains a one-size-fits-all approach (Faust and Weingart, 2017).

Reinforcement learning has been proposed as a tool for creating data-driven approaches to sepsis resuscitation and may be well-suited to the handling the complexities of optimizing treatment dose and timing amongst a highly dynamic patient population such as those presenting with sepsis (Seymour et al., 2019). Reinforcement learning combines aspects of both machine learning and dynamic system control theory, utilizing Markov decision processes to optimize behavior of an agent towards achieving a long-term goal (Burkov, 2019; Howard, 1960). Applied to the problem of clinical decision making in a hospital setting, we might imagine this framework as a learning tool for how an agent (in our case, a clinician) might optimize their series of interactions with an environment (the patient, through sequential treatment actions) to maximize an outcome (recovery, survival, and eventual hospital discharge) .

One type of reinforcement learning model that may be particularly well-suited to clinical decision support is Q-learning. In a Q-learning framework, “Q-values” are calculated that represent the expected cumulative reward of taking an action while in a given state. Rewards are fixed and specified *a priori* and the Q-value of a given action in a state is approximated by

previously observed state-action pairs. Optimal decisions are those in which the Q-value is maximized within a given state. The policy value for an entire trajectory of decision making can be inferred from the sum of a series of decisions and the relative Q-values of those decisions, and an optimized policy will select the cumulative series of decisions that maximizes Q-value over the stay (Watkins and Dayan, 1992). These Q-values can be estimated and an optimal policy inferred, independent of the policy being followed, so long as all state-action pairs are updated over time (Sutton and Barto, 2018). Applied to clinical decision making, this means that a Q-learning model is able to converge towards an optimal treatment regimen, even when a clinician does not always follow model-recommended actions or in observational data, so long as all state-action pairs are explored over time.

The most promising reinforcement learning model applied to sepsis treatment, the "AI Clinician", is a computational machine learning (ML) model created by Komorowski et al. to dynamically suggest an optimal regimen of IV fluids and vasopressors for the treatment of critically ill sepsis patients. In two retrospective cohorts, the AI Clinician demonstrated lower mortality among septic intensive care unit (ICU) patients whose actual doses most closely matched the AI Clinician's recommendation (Komorowski et al., 2018). However, this model utilized data from relatively small, highly curated data resources and has never been validated in a large, independent sepsis cohort. In addition, this model did not incorporate data from resuscitation prior to ICU admission and did not explore model performance by clinical subgroups.

In this study, we sought to apply Q-learning to a large, observational cohort of UPMC encounters using retrospective electronic health record (EHR) data and with a model structured similarly to the AI Clinician. We hypothesized that we could reproduce the model with minimal changes to features, as electronic health record systems are similar across many US-based health

systems. In addition, we sought to use a more heterogeneous population of septic patients, incorporating sepsis admissions from 14 hospitals, versus the single hospital used in the primary analysis of Komorowski et al. We hypothesized that our model would show a similar overall effect size, with some variability in performance by hospital type and size. In addition, we sought to assess model performance in both patient-level and hospital-level subgroups of interest, such as age, illness severity, and size of hospital. We hypothesized that performance of a reinforcement learning model would most greatly exceed that of clinicians in medically complex cases requiring greater degrees of intervention, since the model would have the greatest ability to optimize decisions among these patients.

2.0 Methods

The overall goal of this study was to create a computational machine learning model to dynamically suggest an optimal treatment approach to sepsis resuscitation using IV fluids and vasopressors. We accomplished this goal in 5 steps, shown in Figure 1. We began by defining a cohort of adults presenting with early sepsis and admitted to the ICU. Among that cohort, we extracted a feature set of clinical characteristics and treatment actions in 4-hour time blocks relative to sepsis onset, as well as long-term mortality outcome data. We then used time-limited carryforward and random forest imputation to handle missing data and K-means clustering as a dimension reduction technique to define similar groups of patient states. We trained a reinforcement learning model using Q-learning, with IV fluid and vasopressor administration as actions and 90-day mortality to define rewards, using 80% of the data for model training. Finally, we assessed model performance using weighted importance sampling, comparing estimated performance from the reinforcement learning model to clinician outcomes overall and among both patient- and hospital-level subgroups of interest in the 20% of data held out for model testing.

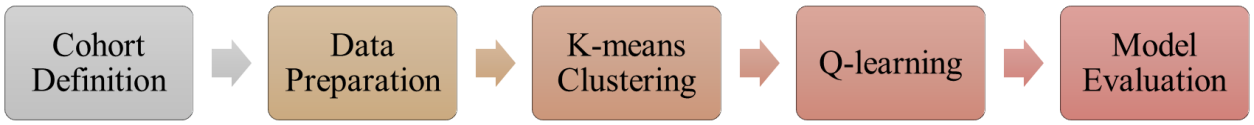


Figure 1: Flow Chart of Modeling Process

The project was approved by the University of Pittsburgh Institutional Review Board (STUDY20010238). The data for the project were obtained under a waiver from informed consent and with authorization under the Health Insurance Portability and Accountability Act.

2.1 Cohort

We used data extracted from a CERNER Electronic Health Record (Cerner, Kansas City, MO) system containing all medical encounters from 14 community and academic hospitals within the UPMC health care system. We identified all adults (age ≥ 18 years) who met sepsis-3 criteria (see Section 2.1.1) within the first 6 hours of presentation to the 14 hospitals during 2013-2017 and who were admitted to an ICU during the study window.

2.1.1 Sepsis Definition

We identified patients meeting sepsis-3 criteria within 6-hours of hospital arrival using the EHR. Specifically, sepsis-3 (Singer et al., 2016) is defined by:

i.) evidence of suspected infection, and

ii.) presence of organ dysfunction.

We defined suspected infection as the combination of prescription of antibiotics (oral or parenteral) and body fluid culture specimen sampling (blood, urine, or cerebrospinal fluid), the first of which was required within 6 hours of hospital presentation. We defined the presence of organ dysfunction as 2 or more Sequential Organ Failure Assessment (SOFA) points within the first 6 hours of hospital presentation (Vincent et al., 1996), as determined by the worst clinical values measured during the time window. The time of sepsis onset was defined as the earlier of first fluid culture or antibiotic administration.

2.1.2 Time Windowing of Model Covariates

We analyzed all data prior to and until 48 hours following the estimated onset of sepsis (Figure 2). We excluded patients in whom fluid intake/output and medication administration data was not available and those patients with any code status limitation (e.g., "do not resuscitate" or "do not intubate" orders).

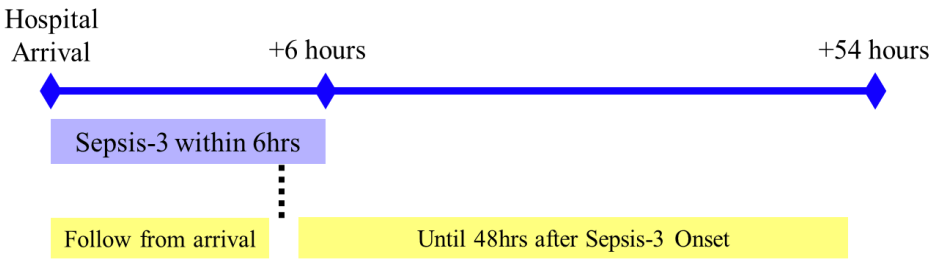


Figure 2: Time Window of Data Used in Analysis

Among this cohort, we looked at clinical information collected in the ICU in the time window between hospital presentation until 48-hour after sepsis onset (Figure 2). Prior research has shown that early resuscitation is most important for clinical outcomes, thus using the first 48 hours after sepsis onset focuses on a time window that we believe is particularly important, based on prior research (Maitland et al., 2013; Self et al., 2018). In addition, we carried forward clinical information collected prior to ICU admission, as would be available to a treating ICU clinician at the bedside. This carry forward is described below in Section 2.3.

2.1.3 Cohort Characteristics

We report characteristics of interest for the cohort, such measures of acute and chronic illness severity, as well as measures of hospital utilization. In our table of cohort characteristics, we define age, race, and gender as patient reported values at admission. SOFA score is defined as the maximum SOFA score based on documented values during the first 6-hours after hospital arrival. Elixhauser comorbidity index is based on discharge International Classification of Diseases (ICD-9-CM and ICD-10-CM) codes. Surgical admissions are defined as admissions requiring at least one surgery during the encounter. Mechanical ventilation and vasopressors are defined as mechanical ventilation lasting more than 4-hours and hospitalizations requiring any vasoactive medications during hospitalization, respectively. Hospital length of stay is defined by the number of calendar days a patient was present in the hospital. In-hospital mortality is defined as encounters with a discharge disposition of "death". Finally, 90-day all-cause mortality is defined using Social Security Death Index death records, supplemented by mandated EHR documentation of deaths that occur within the healthcare system (e.g., nursing or rehabilitation facilities, emergency departments, acute care hospitals). The 90-day all-cause mortality includes encounters in which patients died in-hospital if mortality occurred within the 90-day window.

2.2 Feature Selection and Data Processing

We selected model features based on their association with sepsis onset, resuscitation, and treatment, their use in previously described sepsis models, and their availability in the EHR at hospital presentation (Angus et al., 2001; Angus and van der Poll, 2013; Komorowski et al., 2018).

We extracted model features from the EHR in non-overlapping 4-hour time-steps relative to sepsis onset. The primary outcome was all-cause 90-day mortality, as defined above.

We selected a set of 38 patient features, including patient demographics, Elixhauser comorbidity index (Elixauser et al., 1998), vital signs, laboratory measurements, fluids and vasopressor administration, and fluid balance (Appendix Table 1). This list was chosen to match the feature set used by Komorowski et al, excluding magnesium, calcium, prothrombin time (PT), and partial thromboplastin time (PTT), which were not available in our data and which were deemed of low value by our clinical collaborators. For data elements with multiple measures within the 4-hour time step, we used worst values, defined as minimum or maximum as described in Appendix Table 1. For continuous IV fluid infusions, we derived mean hourly doses by calculating the total dose using administration start and end times then averaging over the administration period. We converted vasopressors to norepinephrine-equivalents as needed, and the maximum dose per time-step was recorded (Brown et al., 2012). We extracted intervention data (e.g., mechanical ventilation) as indicator variables in the time step in which the intervention was started.

We assessed candidate feature distributions and missingness and created a summary table of mean and standard deviation (SD) for symmetrically distributed, median and interquartile range [IQR] for skewed, and number and percent for categorical features. For time intervals in which a feature was not directly measured, we first applied a time-limited parameter-specific sample-and-hold approach in which observed values from prior time blocks were carried forward into future time blocks. This approach intuitively mimics the cognitive processes of clinicians (Hug, 2009; Komorowski et al, 2018). Vital signs were carried forward for up to 4-hours (1 time block) and laboratory values were carried forward for up to 24-hours (6 time blocks). Values measured prior to ICU admission were carried forward by the same intervals.

For remaining missing data in each cohort, we used iterative imputation by random forests with predictive mean matching to generate 25 independent imputed datasets (missRanger() in R) (Mayer, 2019; Newgard and Haukoos, 2007; Stekhoven and Buhlmann, 2011). Random forest imputation works by treating missing values within a dataset as prediction problems. Each covariate is regressed using each of the other covariates as predictors. Missing values are then predicted using the fitted random forests (Stekhoven and Buhlmann, 2011). The method has been found to produce the least prediction error in clinical and laboratory data, when compared against other commonly used missing data imputation approaches (Kokla et al, 2019; Shah et al, 2014; Waljee et al, 2013). In addition, random forest imputation is well suited to our data because it is able to handle mixed-type and non-normally distributed data. It is also computationally fast compared to other imputation methods, allowing it to scale to large datasets. For our imputations, we constructed random forests with 50 trees for each feature, per imputation. We used a sample fraction of 10% for each tree to reduce correlation between trees and predictive mean matching to ensure that predicted values were clinically plausible.

We then compared feature distributions using histograms and summary statistics for original, post time-limited carryforward, and post-imputation. For imputed values, we selected the median predicted value from the 25 imputed datasets for incorporation into analytic dataset. Score-based features of SOFA score and systemic inflammatory response syndrome (SIRS) criteria, as well as derived values of mean arterial pressure (MAP), base excess, and shock index, were re-calculated after carryforward and imputation. Skewed features were normalized using log or inverse-log transformations as shown in Appendix 1. Data were then standardized to a mean of 0 and standard deviation of 1 (subtracting the mean and dividing by the standard deviation) for K-means clustering.

2.3 Defining State Space

In a Q-learning framework, the state represents the current condition of the environment. Applied to our data, it represents the health status of a patient based on a set of their current clinical features (Beck and Pauker, 1983), including vital signs, laboratory values, and measures of both acutes and chronic illness severity. We chose to use K-means clustering as a dimension reduction technique to divide patients of similar clinical characteristics during a given 4-hour time window into groups (or states). These states are defined agnostic to outcome or clinical interventions, and a given patient may move (or not) between various states over the course of the observed study window, at each of the 4-hour time intervals.

2.3.1 K-means Clustering

K-means clustering is an unsupervised approach for partitioning data into a pre-specified number of distinct clusters (Lloyd, 1957; MacQueen, 1967). In the algorithm, cluster assignments (the "states" of the Q-learning model) are initially made randomly, and centroids are defined as the mean of features for observations within assigned clusters. Assignments are then updated iteratively to re-assign observations to nearest cluster centers by Euclidean distance, with centroids re-calculated among assigned observations on each iteration. Assignments are updated until within-cluster variation is minimized or a pre-determined number of iterations have been completed. This optimization problem is shown in Equation 1.

$$\min_{c_1 \ldots c_k} \left\{ \sum_{k=1}^{K} W(c_k) \right\} \qquad \textbf{Equation 1}$$

In this equation, K denotes a pre-determined number of clusters, c_1 through c_k denote the cluster assignment of observations, and $W(c_k)$ is the within-cluster variation, defined as the mean of the square of the Euclidean distance between each observation and its designated cluster centroid (James et al., 2017). There are numerous variants of K-means clustering, as well as Gaussian mixture models that could similarly group data to a single dimension metric. We chose to use K-means because it is efficient and scalable, and because future data could easily be mapped to the clusters by Euclidean distance. We applied the kmeans() function in R to the 38 clinical features described in Section 2.5 for this step.

2.3.2 Optimizing Number of States

The choice of number of states (*k*) available to a Q-learning model is consequential for how well a model can ultimately be optimized to make decisions based on the data. As the number of states available increases, we allow finer control over the system. For example, if we were deciding between $k = 2$ and $k = 4$, data within each of the 4 groups would likely be more homogeneous than that within the 2 groups. If this greater homogeneity extends to treatment response and outcomes, the choice of a higher *k* might allow improved understanding of response to various treatments and expected outcomes. However, as the number of states increases, so does the amount of data required to generate estimates of treatment response and outcome. Thus, we sought a choice of number of states that would allow for tight grouping of feature patterns within state while avoiding sparsely populated states that may have poor estimates of treatment response and outcome.

We determined an optimal number of states for our data by several metrics. We used Akaike and Bayesian information criteria (AIC and BIC), within-state sum of square errors (SSE), and proportion of variance explained (pseudo R^2) as indicators of how well assignments fit the

data, as well of homogeneity of characteristics within each state (Kodinariya and Makwana, 2013; Roberts, 1999). In addition, we sought a choice of k in which $n \geq 50$ within the least populated state. We explored a number of states between 50 and 2000, in intervals of 50. For comparison, the AI Clinician model by Komorowski et al. used 750 states for a similarly-defined ICU cohort of similar illness severity.

2.3.3 Assessing Fit of States to the Data

Once we determined an optimal number of states for the data, we sought to assess the fit of the states to the underlying EHR data. We examined this fit by i.) plotting the relative size and mortality of each state, ii.) creating a heat map to show feature distribution by state, iii.) using principal component analysis (PCA) to visually assess the relationship between cluster features and mortality, and iv.) comparing the cumulative density of the top 100 ICD-9-CM and ICD-10-CM codes by state to cumulative density overall. By plotting size and mortality, we aimed to visually assess the distribution of observations into states, as well as distribution of mortality by state. We wanted to ensure that there were no states with less than 50 observations, as well as ensure a varied mortality across the states. Through the heat map, we aimed to show visually whether clustering had been driven by a small subset of the features, versus variability in many features. To do this, we used the standardized and normalized values of features, as used in the K-means algorithm. This standardization and normalization ensured similar scaling of features on the heat map.

We visually assessed the relationship between cluster-specific means of features and 90-day mortality using PCA as a dimension reduction technique. PCA uses linear transformations to transform data to maximize variance within each dimension, thus creating the greatest visual

separation between points. We chose to use the first 3 principal components so that they could be put onto a 3-dimensional scatter plot and pseudo-colored each state by mortality rate within state. We then looked for a gradient of mortality rate across the plot as a qualitative assessment of relationship between model features and outcome.

Finally, we plotted the cumulative density of the top 100 ICD-9/10-CM codes by state versus the cumulative density of states overall. In doing so, we sought to demonstrate that while ICD codes were not used in determining states, the patients of similar diagnostic codes tended to group into common states at a higher than random rate. This served as a qualitative check that features used in clustering, as well as the states themselves, were indicative of patient diagnosis.

2.4 Reinforcement Learning

Reinforcement learning (RL) is a machine learning framework that deals with how to learn control strategies to interact with potentially uncertain and complex environments. Models "learn" control strategies through repeated experience and are optimized towards endpoints through positive or negative reinforcements. These models can be applied to sequential decision-making problems to maximize single or multiple long-term goals (Schaefer et al., 2005). Within the framework, there is both an "Agent" and an "Environment". The agent observes the state (S_t, S_{t+1}) of the environment and may take actions (A_t), which may change the environment. In addition, actions taken on the environment may generate a reward (R_t, R_{t+1}), also observable to the agent. Based on these observations (and/or perhaps prior knowledge) the agent forms a "Policy" for how it will behave in future situations (Sutton and Barto, 2018). This framework is depicted in Figure 3. There are many models for how the agent's policy is determined and updated over time. For this

project, we chose to use a Q-learning model, which is a non-model, Markov-based reinforcement learning approach (Watkins and Dayan, 1992).

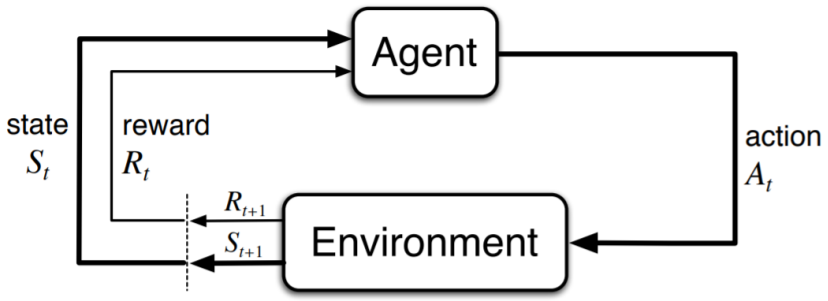


Figure 3: Reinforcement Learning Framework (from Sutton and Barto, 2018)

A RL model may ultimately optimize towards similar behaviors we might expect from a classic regression model, but the processes are quite different. Whereas in a regression model we examine the association between a given treatment and an outcome (e.g. association between vasopressor dose and patient mortality), a RL model instead looks at how a given treatment modifies the state of the environment and compares the relative values of different trajectories (i.e., sequences of treatments) in terms of designated rewards (e.g. what sequence of vasopressor and/or IV fluid administrations can be given to maximize expected reward based on prior experiences, with rewards given for 90-day survival). In terms of rewards, a regression model "rewards" each factor in a model that is associated with a favorable outcome, which would be akin to gaining a "reward" on each iteration of an agent acting on the environment in a RL model. However, RL models instead often give time-delayed rewards in which many state-action pairs are not directly given a reward, but rather move the environment in a direction that is associated with greater reward. This means that RL models can be more suited to learning optimal sequential

decisions, but also means that a greater volume of training data is required, given the more sparce nature of rewards compared to a regression-based approach.

Q-learning falls within a broad class of reinforcement learning models called temporal-difference learning, in which state-action pairs are treated as Markov decision processes, defining sequences of successive states occupied in response to actions applied (Schaefer et al., 2005). Temporal distance models begin with an action policy that can be random or informed by prior knowledge, and then are updated by observed data. There are a variety of approaches to updating the action policy, and Q-learning defines how information learned from actions are used to update the model policy as shown in Equation 2.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \qquad \textbf{Equation 2}$$

In this equation, $Q(s_t, a_t)$ represents the model policy, i.e., the action-value function describing the reward value of taking each potential action on each potential state at time t, based on the data (Watkins and Dayan, 1992). The α term represents how the model is updated upon addition of new information, based on the Reward in the next time block (R_{t+1}, discount factor, γ, current and future state (s_t and s_{t+1}), and action (action space, a and current action, a_t). These terms are explained further in Section 2.4.1.

For the model to converge, state-action pairs must be explored over the course of the data, and the policy can be updated dynamically as data are added to the model. In a prospective design, this exploration pairs can be done with epsilon greedy action selection, as shown in Equation 3.

$$Action\ (a) = \begin{cases} max\ Q_t(a|s) & P = 1 - \varepsilon \\ Random\ (a) & P = \varepsilon \end{cases} \qquad \textbf{Equation 3}$$

In this equation, ε is the exploration parameter, between 0 and 1, defining the probability (P) with which the agent explores the environment by selecting an action at random ($Random\ (a)$), versus using the current policy-optimal action ($max\ Q_t(a|s)$) (Sutton and Barto, 2018). Thus, over time we both explore the model (adding information about all state-action pairs and their associated rewards) and exploit the learned behavior (use of the optimal policy). While we cannot truly "explore" the action space in retrospective observational data, we simulate this in our data by choosing optimal treatment decisions with a probability of $1 - \varepsilon$ and explore with a probability of ε in both the model training and testing phases.

As state-action pairs are updated, the model predicted Q value converges with a probability of 1 towards q^*, the optimal action-value function (Sutton and Barto, 2018). From this action-value function, we define the AI policy as the actions with the highest state-action within each state, defined as $q^*(s)$ as shown in Equation 4.

$$q^*(s) \leftarrow argmax_a Q(s_t, a_t)\ \forall\ s \qquad \textbf{Equation 4}$$

In our model, we determined our optimal action-value function by training our model in the 80% training set and applied the value function statically to the 20% testing set to predict policy value and outcomes. Our optimal policy was created with a policy iterative approach, starting with a random policy and iteratively reevaluated using data from the training set until all training data was incorporated (yielding an "optimal" solution, given the data). The model was created in R using the ReinforcementLearning() package and the model, as well as post-model estimates are called the "AI policy" or "AI model" in this paper (Proellochs and Feuerriegel, 2020).

2.4.1 Model Parameters

The three model parameters specified in a Q-learning model are the learning rate, α, the discount factor, γ, and the exploration parameter, ε, and determine the rate at which the Q parameter is updated and state space is explored, as shown in Equation 2 and Equation 3.

The learning factor quantified an overall weight assigned to new data that is added to an existing policy. As this factor increases towards 1, new data is given a higher weight, therefore placing a higher weight on new data added to the policy, versus the existing policy. We specified a learning factor of 0.1, balancing newly added data with existing policy in the model.

The discount factor quantifies the importance of future rewards. A factor near 0 will make the agent place greater weight on near-term rewards. As the factor approaches 1, the agent places greater weight on long-term rewards. Since we did not specify intermediate rewards and because we used 90-day survival, rewarded only in the final time block of the encounter, we set our discount factor to 0.99 to encourage the agent to seek this future reward.

The exploration factor quantifies the probability that a decision is randomized, versus following the optimal policy. We used an exploration factor of 0.1 for this project. This means that there is a probability of 0.9 that treatment decisions follow the optimal decision (the decision with the highest associated policy value), and there is a probability of 0.1 that the treatment decision is randomized to 1 of the 25 possible choices. We used a soft-epsilon policy, which means that when randomized actions are chosen, decisions are randomized to any 1 of the 25 choices, rather than restricted to randomization between the 24 non-optimal choices. (Sutton and Barto, 2018; Watkins and Dayan, 1992).

2.4.1.1 State Space

Applied to our data, a "State" s represents the health status of a patient based on their current (time t) set of clinical features (Beck and Pauker, 1983), and as represented by clinical features including vital signs, laboratory values, and measures of both acutes and chronic illness severity. These features are reduced to 1 of *k* possible patient states as described in Section 2.3. states and state-action pairs are assumed memoryless, which means that states are not associated with or defined by the time block in which they occur and state-action-future state combinations are independent of the time in which they occur relative to sepsis onset (i.e., patients with similar clinical characteristics will respond similarly to treatment, irrespective of the time at which they express those characteristics) (Sutton and Barto, 2018).

2.4.1.2 Action Space

The action space defines the potential actions that can be taken within each state by the agent (clinician) on the environment (patient). We wanted to simultaneously optimize the use of both IV fluid dose and vasopressor administration. Thus, we created a 5×5 matrix of IV fluid and vasopressor doses, with 'no treatment' options for each, as well as medians of the 1st through 4th quartiles of observed doses for each treatment in the data. This matrix defined 25 potential treatment actions within each time block. We created a table to characterize the proportion of each treatment action taken under both the Clinician and AI policies.

2.4.1.3 Reward Framework

We assigned a reward (R_{t+1}) of 0 for all observations prior to the last time block (i.e., final 4-hour block). In the last time block, a reward of (+100) was applied if the patient survived through

90-days or (-100) if the patient died within 90-days to match the reward structure used by Komorowski et al. This is a sparce reward framework with no intermediate rewards. Our rationale for this reward structure was two-fold. First, by matching the reward and penalty structure used by Komorowski et al., our results would be most directly comparable with that model. Second, we were concerned that intermediate rewards such as rewarding hemodynamic stability might be confounded with patient state, as the patient characteristics one might use to define an intermediate reward are largely captured by the feature set used in creating the state space. In addition, our long-term goal is not to achieve hospital stability, but rather to create a model that optimizes towards letting patients return home from the hospital, and thus a reward policy optimized towards that primary goal is appropriate.

2.4.2 Comparison Models

Performance of the AI policy was compared to two models that we have termed the "Clinician" and "Random" policy models. The Clinician policy model represents the actual actions taken by clinicians in each time block. This model serves as the primary benchmark comparing the predicted performance of our AI policy against, and both policy value and mortality were estimated using inverse probability weighting of state-action pairs in this model. The second comparison model is the Random Policy and is used as a test of construct validity. In this policy, treatments are assigned at random, with equal likelihood. Random policy values and estimated mortality were estimated in the same way as for the AI policy, as described in Section 2.7.3. Absolute risk differences were calculated for only the AI versus Clinician policy, but we have provided policy values and estimated mortality overall and among subgroups for all three models.

2.4.3 Policy Evaluation

We evaluated performance of each of the policies in the testing data (20%). For each of the policies, we calculated policy values by weighted importance sampling (WIS), as well as predicted mortality by importance sampling. Weighted importance sampling is an inverse propensity score-based system that allows the expected value of the policy value under the AI (or Random) model to be calculated, given the distribution observed under the Clinician policy. Since the AI policy is trained using retrospective data that denotes what actually occurred under the Clinician policy, this is referred to as "Off-Policy" evaluation (i.e., what might have happened if we had gone "off" of the trajectory taken by clinicians by changing the series of actions taken). There are a number of alternative approaches for off policy evaluation use variations of inverse propensity scoring, direct model-free approximations, double robust methods, or other approaches (Voloshin et al., 2019). We chose WIS to remain consistent with the approach used in the Komorowski et al. paper.

To calculate the policy value using WIS, we start by calculating a per-step importance ratio that denotes the relative probability of each action, given the state, for the AI versus Clinician policies. The equation for calculating this ratio is given in Equation 5.

$$\text{Per Step Importance Ratio: } \rho_t = \frac{Q_{AI}(a_t|s_t)}{Q_{Clinician}(a_t|s_t)} \qquad \textbf{Equation 5}$$

This ratio is calculated for all state-action pairs. Since we treat the policy as time-invariant, these relative probabilities of actions for each state do not vary with time. Under the AI policy, the probability of taking the state-optimal action is $\{1- \varepsilon + \varepsilon/25\}$ and probability for each of the other actions is $\{\varepsilon/25\}$. Under the random policy, the probability of each action is simply 1/25, for the 25 potential actions. We multiply the per step importance ratio across the entire patient trajectory

for each encounter in the dataset to calculate a cumulative importance ratio for each encounter, which denotes the relative probability of encountering the observe trajectory under the AI (or Random) vs. Clinician policies. This is cumulative ratio is calculated as shown in Equation 6.

$$\text{Cumulative Importance Ratio: } \rho_{1:t} = \prod_{t'=1}^{t} \rho_t \qquad \textbf{Equation 6}$$

These ratios are averaged across the entire testing dataset to determine an average cumulative importance ratio. For each encounter (or observed trajectory), i, this averaged cumulative ratio is calculated as shown in Equation 7.

$$\text{Average Cumulative Importance Ratio: } w_t = \sum_{i=1}^{|D|} \frac{\rho_{1:t}^{(i)}}{|D|} \qquad \textbf{Equation 7}$$

In this equation |D| denotes the number of encounters (or trajectories in the testing dataset). We can then calculate the relative value of the trajectory under each of the policy models as shown in Equation 8.

$$V_{WIS}^{(i)} = \frac{\rho_{1:t}^{(i)}}{w_t} \left(\sum_{t=1}^{H^{(i)}} \gamma^{t-1} r_t \right) \qquad \textbf{Equation 8}$$

In this equation, H is the number of time blocks for encounter i. These trajectory-wise WIS values can then be averaged across all of the observations to give an overall estimate of the trajectory-wise WIS Policy value, as shown in Equation 9.

$$\text{Policy value}_{\text{WIS}} = \frac{1}{|D|} \sum_{i=1}^{|D|} V_{WIS}^{(i)} \qquad \textbf{Equation 9}$$

This Policy Value is calculated for the AI, Clinician, and Random policies. For the Clinician policy, the Cumulative Importance ratio = 1, since the observed trajectory is the Clinician trajectory (Komorowski et al., 2018; Sutton and Barto, 2018). The same approach can be used to estimate predicted mortality under each policy, substituting a reward value of +1 for 90-day survival and 0 for 90-day death to estimate 90-day survival probability ($P_{death} = 1 - P_{survival}$).

We retained the exploration factor during testing of the AI policy to simulate "continued learning" during model testing. In order to statistically compare the Clinician and AI policies and to generate confidence intervals around point estimates of policy values, we conducted 1,000 bootstrap samples for policy evaluation for the Clinician, AI, and Random policies. We tested whether the AI policy value was superior to the Clinician policy at a significance level of 0.05 empirically by looking at the proportion of mean differences in policy value that were greater than 0, as shown in Equation 10.

$$\text{Difference} = \text{Policy Value}_{\text{WIS,AI}} - \text{Policy Value}_{\text{WIS,Clinician}} \qquad \textbf{Equation 10}$$

We plotted the bootstrapped mean and 95% confidence intervals for each of the policy values. In addition, we plotted bootstrapped mean and 95% confidence intervals for absolute risk of 90-day mortality, as well as absolute risk difference between the AI and Clinician policies, both overall and among strata defined in Section 2.5.

2.5 Post Q-Learning Subgroup Evaluation

Finally, we evaluated performance of the AI policy versus Clinician policy among subgroups of interest. For this, we used the same Q-learning model derived in the full training set and same importance ratios generated for the overall model. However, we restricted the population from which bootstrap samples were taken to subgroups of interest and conducted 1,000 bootstrap samples within each subgroup. For each subgroup, we calculated absolute risk of 90-day mortality under the AI, Clinician, and Random policies in the same way that we calculated for the cohort overall. In addition, we calculated absolute risk difference of the AI policy versus Clinician.

We also sought to characterize variability in model performance by both patient-level and hospital-level factors. For patient-level factors, we assessed model performance by strata of age (in 4 categories; 18-39, 40-59, 60-79, and 80+ years of age), patient reported gender at admission, quartile of SOFA score during the first 6-hours after admission, hospitalization within the past 60-days, and surgical vs. medical admission. For hospital factors, we compared performance in the following strata: academic vs. non-academic hospitals, hospitals located in rural vs. suburban vs. urban environments, and by annual case volume, categorized as low, medium, and high. In addition, we sought to estimate variability in hospital performance by hospital and created estimates for each of the 14-hospitals. We created forest plots of mean and 95% confidence intervals of absolute risk difference and absolute risk for each of the strata.

3.0 Results

Results are grouped into cohort description, missing data, state space optimization, Q-learning and model assessment, and subgroup variability. The results describe the cohort and its characteristics, as well as how characteristics were modified by the time-limited parameter-specific sample-and-hold approach and random forest imputation. We present results from the unsupervised K-means clustering, both in terms of choice of *k* and assessment of fit within the chosen *k*. We then present model performance of the AI policy against the Clinician and Random policies. Finally, we present model performance groups in both patient-level and hospital-level subgroups of interest.

3.1 Cohort

The cohort used for this study contained patient-level data extracted from Cerner electronic medical records (Cerner, Kansas City, MO) from all UPMC hospital encounters from 2013-2017. The dataset contained covered 14 hospitals and 3,071,675 adult (age 18 and above) patient encounters. Of these, 123,610 (4%) met sepsis-3 criteria within 6 hours of hospital arrival. Excluding non-ICU encounters, encounters with missing medication information, and encounters that were less than 8 hours (i.e., two 4-hour time blocks), the final analytic cohort contained 30,678 patient encounters, which were split into an 80% training and 20% testing set. Details are shown in the consort diagram in Figure 4.

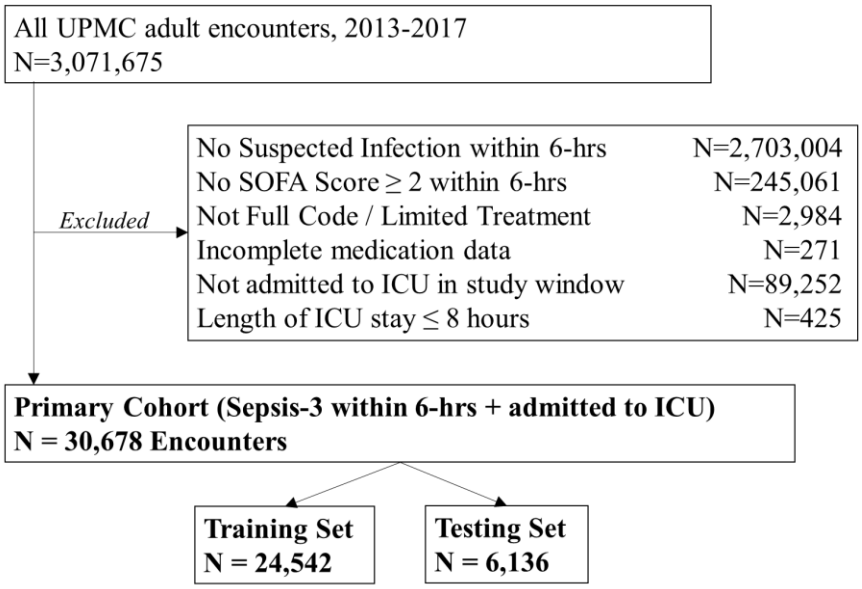


Figure 4: Consort Diagram

Patient characteristics for this cohort are presented in Table 1. The cohort had a mean age of 64 (SD 16), had a mean SOFA score of 3.9 (SD 2.8), and had a mean Elixhauser comorbidity index of 5.2 (SD 2.3). The proportion of patient encounters receiving intravenous fluids during the study window was 84% and the proportion receiving vasoactive medication was 27%. The cohort was comprised of 18% surgical vs. 82% medical encounters. The median length of hospital stay was 8 days, with 45% of encounters on mechanical ventilation during the encounter and 33% receiving vasoactive medication at some point during their encounter. The cohort had an in-patient mortality of 15% and a 90-day all-cause mortality rate of 30%.

Table 1: Cohort Characteristics

Feature	Value
No. of Encounters	30,678
Patient Characteristics	
Age (years), mean (SD)	64 (16)
Gender[a]	
Female	15,179 (50%)
Male	15,499 (50%)
Race[a]	
White	25,069 (82%)
Black	3,890 (13%)
Other[b]	1,719 (6%)
Sequential Organ Failure Score, mean (SD)[c]	3.9 (2.8)
Elixhauser Comorbidity Index, mean (SD)[d]	5.2 (2.3)
Treatment	
Fluids Recv'd in Study Window, n (%)	25,753 (84%)
Vasopressors Recv'd in Study Window, n (%)	8,358 (27%)
Outcomes and Hospital Utilization	
Surgical Admissions, n (%)	5,359 (18%)
Mechanically Ventilated, n (%)[e]	13,719 (45%)
Vasopressors, n (%)[e]	10,239 (33%)
Hospital Length of Stay (days), median [IQR]	8 [5 - 14]
In-Hospital Mortality, n (%)	4,704 (15%)
90-Day Mortality, n (%)	9,162 (30%)

[a]Patient-reported at admission.
[b]Includes Chinese, Filipino, Hawaiian, American Indian/Alaskan Native, Asian, Hawaiian/other Pacific Islander, Middle Eastern, Native American, not specified, or Pacific Islander.
[c]Corresponds to the severity of organ dysfunction, reflecting 6 organ systems each. Scores range from 0 to 4 points for cardiovascular, hepatic, hematologic, respiratory, neurological, and renal. The total score range is from 0 to 24 points.
[d]A method of categorizing comorbidities of patients based on the International Classification of Diseases, Ninth Revision diagnosis codes found in administrative data. Scores range from 0 to 31.
[e]At any time during hospitalization.

3.2 Feature Selection and Data Processing

In Table 2, we present summary statistics for the 38 features used to determine patient states, as well as missingness of these features in the raw data. These values were taken among 309,840 4-hour time blocks in the study window from the overall 30,678-encounter cohort. Missingness ranged from 0% to 51% in model features in the raw data, with highest missingness in blood-gas-related model features, including base excess, FiO_2, $PaCO_2$, and PaO_2, PF Ratio, and arterial pH (40-51%). In addition, missingness was higher for several lab values, including albumin, ALT, AST, and bilirubin, INR, and serum lactate (34-39%). In Table 2, we also show summary statistics for each model feature after the time-limited parameter-specific sample-and-hold carryforward and after random forest imputation among remaining missing values. For features with multiple measurements within a single time block or for features with skewed distributions, we show how features were pre-processed prior to K-means in Appendix Table 1.

Table 2: Model features: Originally, After Carryforward, and After Imputation

Feature[a]	Original	Post-Carryforward	Post-Imputation[b]	Missingness
Age, mean (SD)	64 (16)	64 (16)	64 (16)	0%
Albumin, mean (SD)	2.6 (0.6)	2.7 (0.7)	2.7 (0.6)	37%
ALT, median [IQR]	31 [17 - 77]	27 [16 - 56]	24 [17 - 41]	38%
AST, median [IQR]	42 [22 - 115]	34 [20 - 78]	30 [21 - 54]	38%
Base Excess, mean (SD)	-2.1 (7.5)	-1.9 (7.4)	-1.0 (6.0)	50%
Bicarbonate, mean (SD)	23 (6)	24 (6)	24 (6)	4.3%
Bilirubin, median [IQR]	0.8 [0.5 - 1.6]	0.7 [0.4 - 1.3]	0.6 [0.5 - 1.0]	38%
BUN, median [IQR]	28 [17 - 47]	26 [16 - 43]	26 [16 - 42]	4.3%
Chloride, mean (SD)	106 (8)	105 (7)	105 (7)	3.8%
Creatinine, median [IQR]	1.4 [0.9 - 2.5]	1.3 [0.8 - 2.2]	1.3 [0.8 - 2.1]	4.3%
Diastolic BP, median [IQR]	69 [60 - 80]	69 [60 - 80]	69 [60 -80]	0.5%
Elixhauser, mean (SD)	5.3 (2.3)	5.3 (2.3)	5.3 (2.3)	0.2%
FiO2, median [IQR]	50 [40 - 70]	50 [40 - 70]	40 [40 - 50]	40%
GCS, mean (SD)	12.1 (3.5)	12.1 (3.5)	12.4 (3.3)	16%
Gender (male), n. (%)	156,664 (51%)	156,664 (51%)	156,664 (51%)	0%
Glucose, median [IQR]	148 [114 - 201]	136 [108 - 180]	135 [109 - 176]	3.7%
Hemoglobin, mean (SD)	10 (2)	11 (2)	11 (2)	3.6%
Heart Rate, mean (SD)	95 (21)	95 (21)	95 (21)	0.4%
INR, median [IQR]	1.5 [1.2 - 2.1]	1.4 [1.2 - 1.8]	1.3 [1.2 - 1.6]	39%
Potassium, mean (SD)	4 (1)	4 (1)	4 (1)	3.5%
Serum Lactate, median [IQR]	2.1 [1.3 - 3.7]	1.6 [1.1 - 2.6]	1.4 [1.1 - 2.1]	34%
MAP, median [IQR]	89 [79 - 101]	89 [79 - 101]	89 [79 - 101]	0.5%
Mech Vent in Window	122,464 (40%)	122,464 (40%)	122,464 (40%)	0%
Sodium, mean (SD)	139 (7)	139 (6)	139 (5)	3.9%
SaO_2, median [IQR]	95 [93 - 98]	95 [93 - 98]	95 [93 - 98]	0.4%
$PaCO_2$, mean (SD)	44 (16)	43 (15)	42 (11)	50%
PaO_2, mean (SD)	130 (79)	128 (77)	103 (46)	51%
PF Ratio, median [IQR]	223 [143 - 332]	222 [143 - 328]	250 [163 - 375]	40%
Arterial pH, mean (SD)	7.3 (0.1)	7.4 (0.1)	7.4 (0.1)	50%
Platelets, median [IQR]	173 [114 - 241]	180 [123 - 247]	182 [130 - 241]	5.0%
Resp Rate, mean (SD)	21 (6)	21 (6)	21 (6)	0.4%
Systolic BP, median [IQR]	128 [113 -146]	128 [113 - 146]	128 [113 -146]	0.4%
Shock Index, mean (SD)	0.8 (0.2)	0.8 (0.2)	0.8 (0.2)	0.4%
SIRS in Window, mean (SD)	1.6 (1.0)	1.6 (1.0)	1.8 (1.1)	0.4%
SOFA in Window, mean (SD)	3.5 (2.9)	3.5 (2.9)	5.4 (3.3)	0%
Temperature, mean (SD)	36.8 (0.9)	36.8 (0.9)	36.8 (0.9)	0.6%
WBC Count, median [IQR]	12 [8 - 17]	12 [8 - 17]	12 [8 - 16]	5.3%
Weight, mean (SD)	85 (29)	85 (29)	85 (29)	2.6%

[a] Mean and SD presented for features with symmetric distribution; median and IQR presented for skewed data

[b] Imputed values represent median of 25 imputed datasets

Abbreviations: ALT, alanine aminotransferase; AST, aspartate aminotransferase; BP, Blood Pressure; BUN, Blood urea nitrogen; FiO2, fraction of inspired oxygen; GCS, Glasgow Coma Scale score; INR, international normalized ratio; IQR, interquartile range; MAP, mean arterial pressure; Mech Vent, mechanical ventilation; n., number; $PaCO_2$, partial pressure of arterial carbon dioxide; PaO_2, partial pressure of arterial oxygen; PF Ratio, Ratio of PaO_2 to FiO_2; Resp Rate, respiratory rate; SaO_2, oxygen saturation; SD, standard deviation; SIRS, systemic inflammatory response syndrome; SOFA, sequential organ failure assessment; WBC, White Blood Cell

3.3 Defining State Space

We determined model fit statistics (AIC, BIC, SSE, and pseudo R^2) for the dataset for K-means state assignments from k = 50 to 2,000, in intervals of 50 (Figure 5). We selected k = 750 as an optimal fit for the data, based on a combination of inflection point in the BIC and an "elbow" in the AIC and SSE.

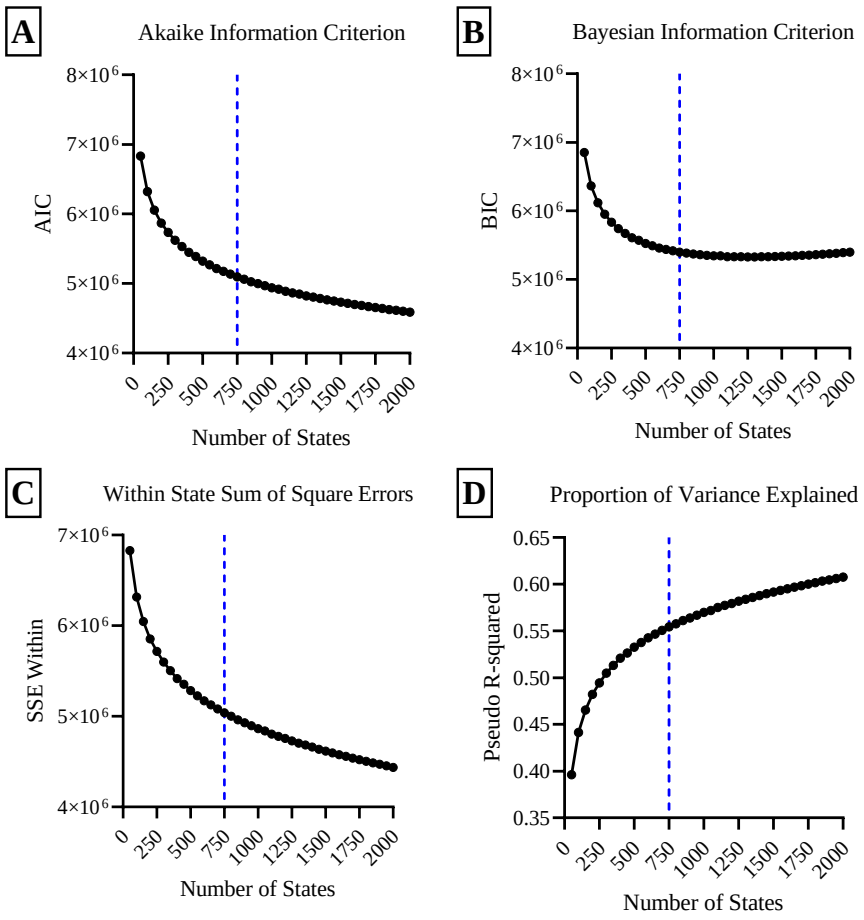


Figure 5: K-means Model Fit Statistics

For the k = 750 model, we examined the sample size of the states (i.e., the number of 4-hour blocks represented), as well as 90-day mortality within each state (Figure 6, ordered from smallest to largest size and from lowest to highest mortality, respectively). Sample size of states ranged from a minimum of 87 observations to a maximum of 937 observations. 90-day mortality ranged from 0.9% to 100% across the states. This meant that all states were above our minimum size threshold of 50 and that there was a wide gradient of mortalities. We further explored the state with 100% mortality and found that it had a mean SOFA score of 13.2, ALT of 1,399, and serum lactate of 10.2, indicating very high illness severity in this state. Additional information is provided about the state with 100% mortality in Appendix Table 2.

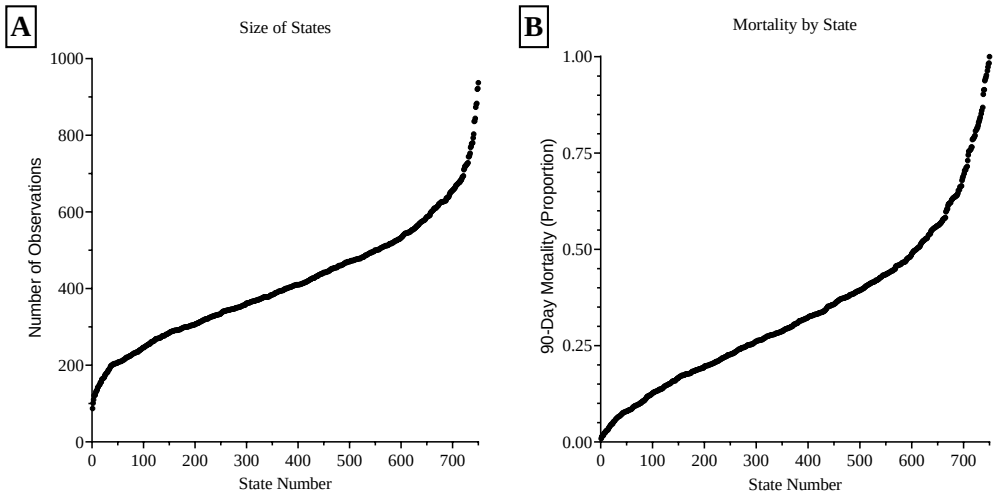


Figure 6: Number of Observations per State and Mortality of States in K=750 Model

A heat map of 75 randomly selected states from the k = 750 model shows normalized, standardized model features, with no single feature driving construction of the state space (Figure 7). It appears that there is variability in many of the features by state in the heat map, indicating that clustering was driven by multiple features.

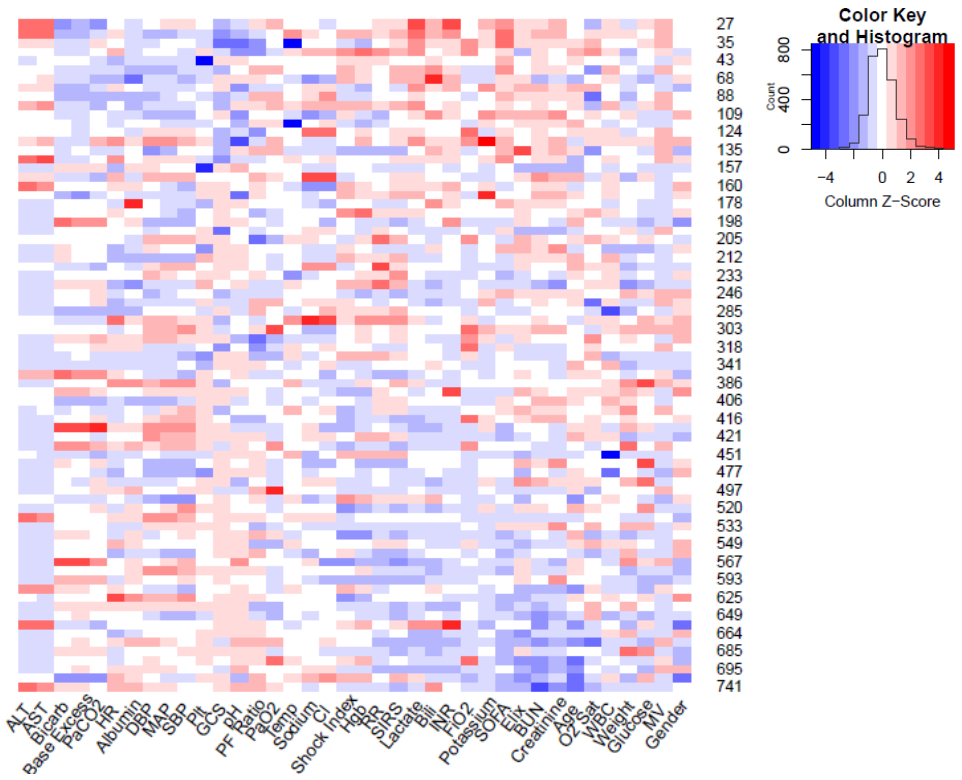


Figure 7: Heat Map of Mean Feature Values by State

In addition, we conducted a PCA of the cluster centroids using model features. We created a 3-dimensional scatter plot of the first three principal components, pseudo-colored by 90-day mortality proportion (Figure 8) to assess whether model features were generally associated with outcome. This plot shows that clusters of similar 90-day mortality tended to group together spatially, with a visually-apparent gradient in mortality across the principal component space.

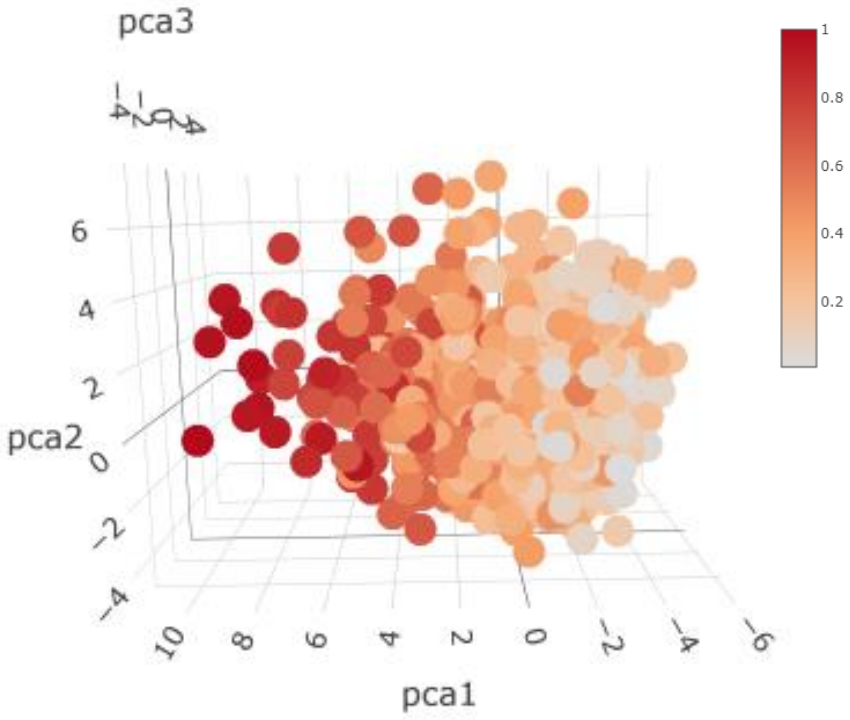


Figure 8: PCA of Model Features, Colored by 90-Day Mortality

Finally, Figure 9 shows an empirical cumulative densify function (CDF) plot of the top 100 International Classification of Diseases (ICD) diagnoses codes for the cohort, along with a CDF of the cohort by size of states, both ordered from largest to smallest. The dashed line represents the cohort by size of states and colored lines represent ICD diagnosis codes. The Orange line nearest the CDF for size represents ICD-9 code 038.9, "Unspecified Septicemia". The Yellow line with the greatest AUC is ICD-10 code N10, "Acute pyelonephritis" (kidney infection). All 100 of the ICD codes had a greater area under the curve than the overall CDF, indicating that clusters may have been associated with diagnoses, despite ICD codes not being used in the feature set. In addition, more specific conditions, such as acute pyelonephritis may have grouped more tightly than more broadly defined conditions, such as unspecified septicemia.

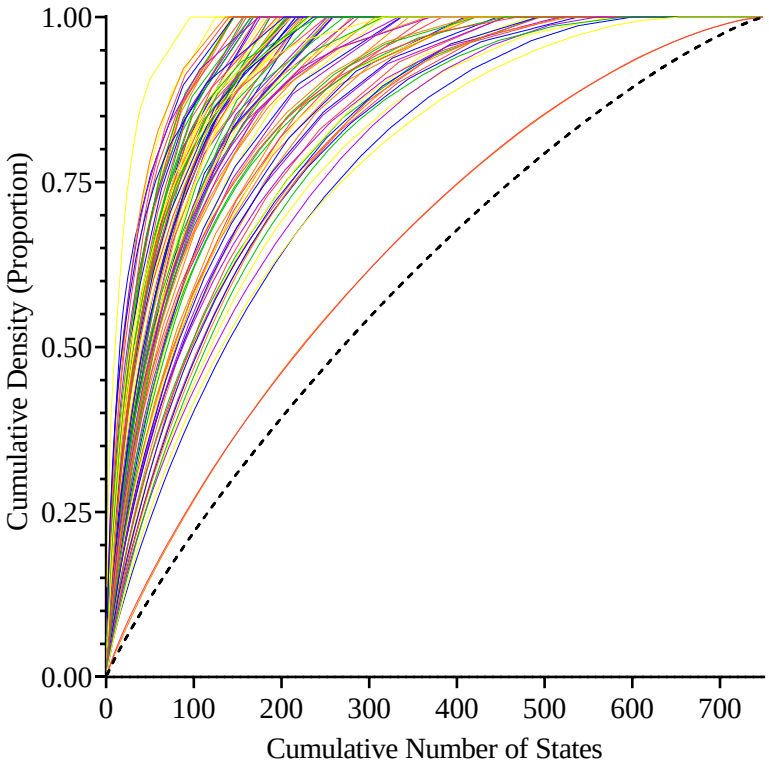


Figure 9: Cumulative Density Function of Top 100 ICD-9/10 Codes versus States Overall

3.4 Q-Learning

We derived an optimal Q-learning policy, termed our "AI policy", in the training data, representing 24,542 encounters (80% of total encounters in the cohort). We applied a learning rate, α, of 0.1 and a discount factor, γ, of 0.99 to the training data. We used an exploration parameter, ε, of 0.1 to simulate exploitation of the optimal policy in 90% of action decisions and exploration in 10% of action decisions.

We compared behavior of the AI policy model to two other models: 1.) the Clinician policy model and 2.) the Random policy model. We first compared the relative proportion of treatment actions across dosing ranges taken by the AI policy to Clinician and Random policies (shown in Table 3). Most notably, the AI policy recommended no treatment for a relatively greater portion of actions than the Clinician for both intravenous fluids and vasopressors. Under the Random policy, actions are chosen at random, with equal probability of each.

Table 3: Treatment Actions by Clinician, AI, and Random Model

Action	Intravenous Fluids (mL/4h)				Vasopressors (mcg/kg/min)			
	Range	Proportion of Actions			Range	Proportion of Actions		
		Clinician	AI	Random		Clinician	AI	Random
1	0	0.468	0.607	0.2	0	0.860	0.900	0.2
2	1-250	0.105	0.050	0.2	.001-0.09	0.036	0.022	0.2
3	251-400	0.156	0.142	0.2	0.1-0.2	0.030	0.023	0.2
4	401-700	0.109	0.064	0.2	0.21-0.5	0.040	0.028	0.2
5	>701	0.162	0.138	0.2	>0.501	0.035	0.027	0.2

We assessed performance of the AI policy compared to the Clinician and Random policies in the testing data, representing 6,136 encounters (20% of total encounters in the cohort). We drew 1,000 bootstrap samples of the testing data of size n = 6,136, taken with replacement. We

calculated trajectory-wise WIS policy values for each policy and compared these between policies. The bootstrapped mean and 95% confidence interval for policy value of the AI policy was 41.9 [41.2 - 42.7], versus 40.8 [39.9 - 41.6] in the Clinician policy and 37.6 [36.7 - 38.5] in the Random policy (Figure 10). We tested superiority of the AI policy against the Clinician model by comparing the mean policy values within each of the bootstrap samples. The AI policy value exceeded that of the Clinician policy in all 1,000 samples, for an empirical p-value of $p<0.001$, with a mean difference in policy values of 1.18 (SD 0.20). Thus, we concluded superiority of the AI policy versus the Clinician policy at a 95% confidence level.

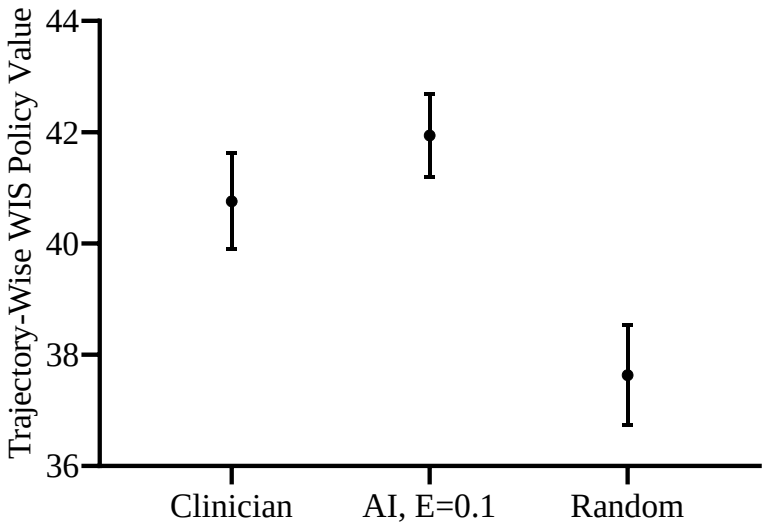


Figure 10: Trajectory-Wise WIS Policy values of Clinician, AI, and Random Models

3.5 Post Q-Learning Subgroup Evaluation

We calculated the difference in predicted 90-day mortality between the AI and Clinician policies within each bootstrap sample of the testing set. In Figure 11, we present the mean and 95% confidence interval of the absolute risk difference between the models. We calculated a mean difference in 90-day mortality of 0.59% (95% CI: 0.38% - 0.81%), with an exploration parameter, ε, retained during evaluation of the AI policy.

We obtained bootstrap samples within subgroups of interest defined *a priori* to compare policy performance within subgroups. We present patient-level subgroups in Figure 11, including age, gender, acute illness severity (SOFA score), prior hospitalization within 60-days, and surgical vs. non-surgical admission. The AI policy performance most exceeded that of the Clinician mortality (had greatest decrease in predicted 90-day mortality) in older patients, non-surgical patients with high illness severity, and those with hospitalization within the prior 60-days. For both men and women, the AI policy outperformed the Clinical policy in a similar manner. The Clinician policy performance exceeded that of the AI policy in patients 18-39 years old and in surgical admissions. Taken together, it appears that the AI policy was best in older, more medically complex cases with higher risk of prior illness and was worst in younger surgical patients of lower illness severity.

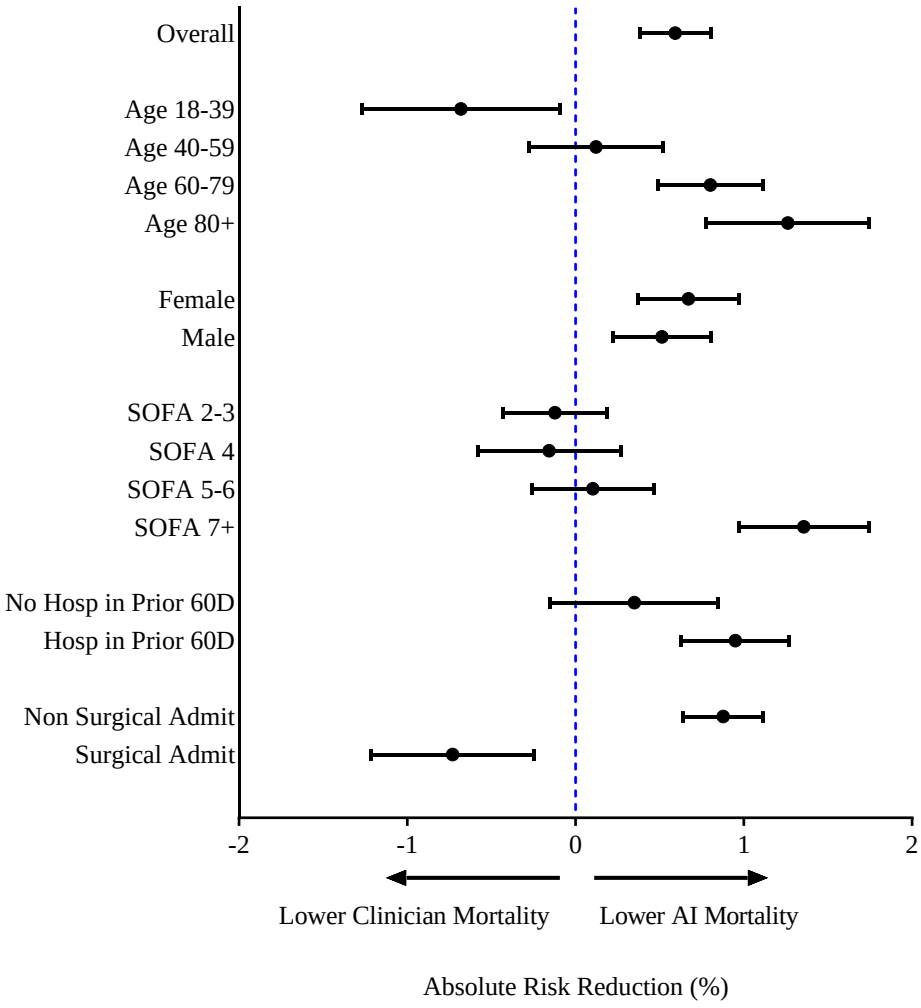


Figure 11: AI vs. Clinician Model Performance, by Patient-level Subgroups

We assessed model performance variability by hospital and by hospital-level subgroups. In Figure 12, we show a caterpillar plot of absolute risk difference by hospital, with hospitals ordered from least to greatest absolute risk reduction of the AI policy versus Clinician. We found that the mean performance of the AI policy exceeded that of the Clinician policy in all 14 hospitals, with a range of absolute risk reductions of 0.01% to 1.18%. However, the 95% confidence intervals crossed 0 for 9 of the 14 hospitals and the data generally suggest that AI policy performance versus Clinician may differ by hospital.

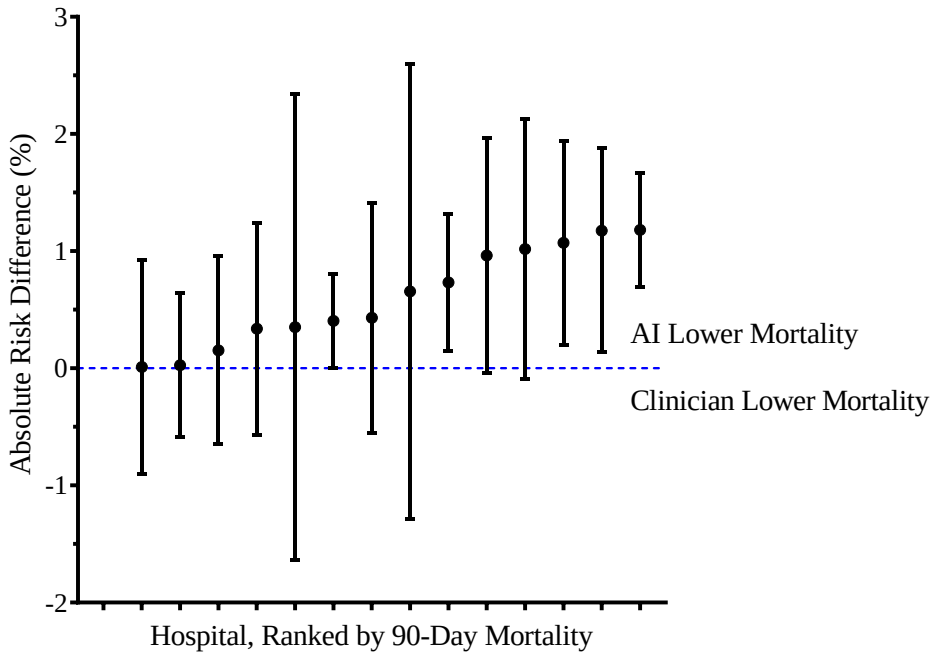


Figure 12: AI vs Clinician Performance, Ranked by Hospital

We calculated absolute risk differences between the AI and Clinician policies by hospital factors of academic vs. non-academic hospital, population density of the region surrounding the hospital, and case volume of the admitting hospitals. There absolute risk differences are presented in Figure 13. While there are differences in the mean predicted absolute risk reductions, the confidence intervals are overlapping for all subgroups, indicating no clear trends by any of the hospital subgroups. The confidence intervals for rural hospitals and hospitals with low annual volume cross 0 and have wider confidence intervals, related to the relatively smaller number of encounters present in the data for these subgroups.

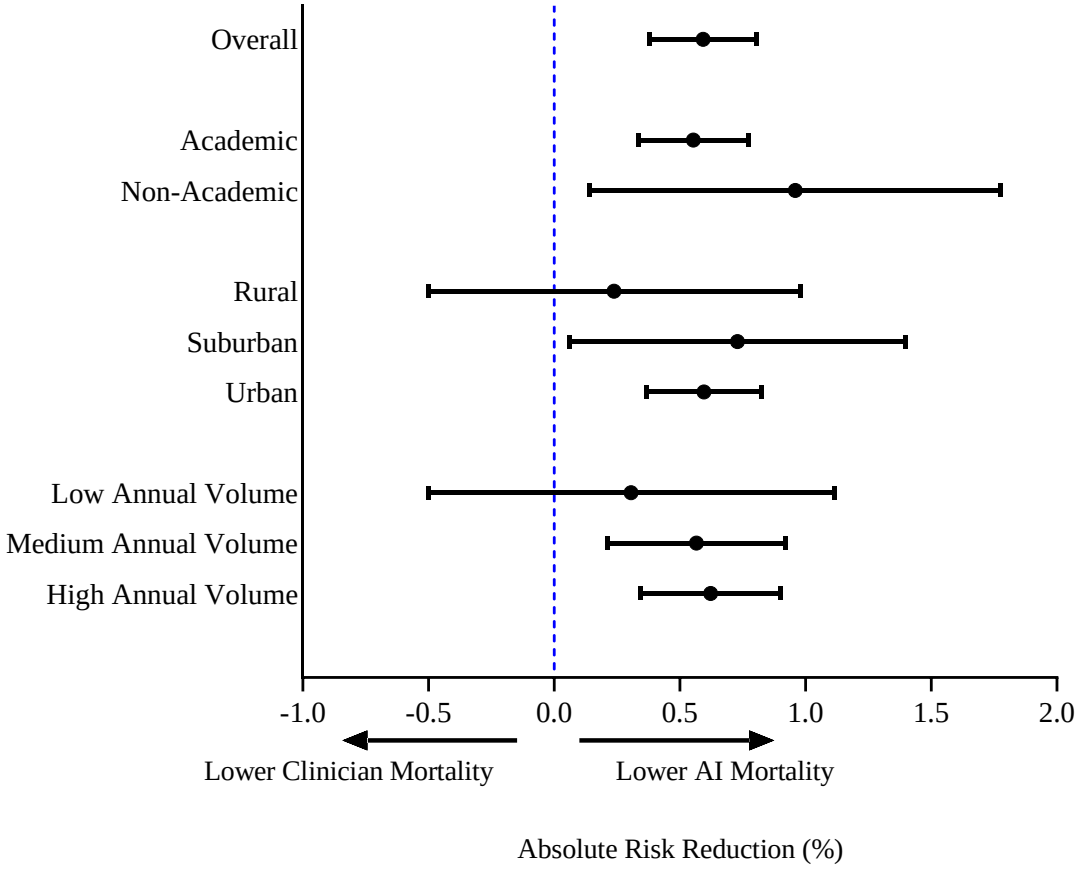


Figure 13: AI vs. Clinician Model in Hospital-level Subgroups

Predicted mortalities for all three policies for patient-level, hospital, and hospital-level subgroups are shown in Appendix Figures 1 - 3.

4.0 Discussion

Reinforcement learning methods offer potential paths towards personalized treatment decisions that can be made dynamically using the wealth of data available in the electronic health record. While approaches such as the one used in this study have been in widespread use in other fields since the 1990s, we are only just beginning to explore the use of reinforcement learning techniques such as Q-learning in clinical decision support. Prior work by Komorowski et al. demonstrated a proof of concept for using Q-learning as a decision support tool, creating a learning model for IV fluids and vasopressor administration in the MIMIC III and eICU cohorts. Our study extends this work by validating the feasibility of this approach in electronic health record information from a large, integrated health system. In addition, we extend this work by exploring model performance in clinically meaningful subgroups of interest.

We used a Q-learning model framework to determine an optimal policy for IV fluid and vasopressor administration for the first 48-hours after sepsis onset in a cohort of ICU patients. We were able to create a treatment action policy with significantly lower predicted 90-day mortality than clinician actions, even when exploring the state space during model evaluation. We investigated model performance in clinically meaningful subgroups. Our model performance most greatly exceeded that of clinicians in medically complex encounters, with higher acute illness severity and history of recent hospitalization. Intuitively, this finding makes sense because these encounters tend to require the greatest degree of clinical intervention, and thus offer the most opportunity for optimization of intervention. There was variability in model performance by hospital, but differences did not follow a clear pattern by hospital subgroups. We looked at the proportions of each treatment action taken by the AI policy model versus Clinician and found that

the AI policy generally recommended fewer vasopressors and IV fluids than clinicians. This behavior aligns with recent research suggesting that restrictive resuscitation methods may be associated with improved clinical outcomes (Andrews et al., 2017; Hjortrump et al., 2016; Reynolds et al., 2020). Both the Clinician and AI policy performance exceeded that of the Random policy, yielding additional face validity to the overall approach. While the study uses retrospective electronic health record data, the results are promising that a similar model applied prospectively could be useful as a clinical decision support tool.

There are multiple potential mechanisms that allow an AI policy to perform better than the Clinician policy. The first is leveraging the large scale of data. While clinicians are highly experienced and expertly trained, our AI policy integrated information from 14 hospitals and 5 years of data. The number of patient encounters observed by our AI policy during model training likely exceeds the number seen by even highly experienced clinicians over the course of their careers. The second is choice of endpoint. While our model sought to optimize 90-day mortality, clinicians at the bedside optimize for many outcomes. For example, end of life care is not accounted for by our model but may be important to many patients. This may be a strong reason that the AI policy performance most greatly exceeded that of clinicians among older patients. Third, there may be contraindications or barriers to fluid and/or vasopressor treatments that go unrecognized by our model. For example, if lines cannot be placed in a patient, giving IV fluids or pressors may simply not be possible in some circumstances, even if both AI and clinician might agree that they should be administered.

The study has several limitations and weaknesses that are important to acknowledge. First, we focused on patients in the ICU. In doing so, we might miss some of the key early hours of sepsis resuscitation. We decided to focus on ICU patients because an all-comer sepsis population

would have introduced greater heterogeneity into the cohort. However, creating a reinforcement learning model that encompasses entire patient stays may be both a valuable next research step and may be a useful clinical model. Second, our model used an action space that may be overly broad and under-specified. In a clinical environment, IV fluid and vasopressor administration would be considered as a part of larger treatment regimens, and understanding how these would interact with other treatments, such as steroids, is important. Within the axes of IV fluid and vasopressor administration, we put no restrictions into place for actions available to the agent for optimal or random decisions, which means that our model could potentially recommend treatment actions that differ greatly from what a clinician might consider appropriate for a given patient in a given state. Third, our model has significant capacity for further refinement. Our choice of *k* was based upon fit to the data, but we did not assess how altering number of clusters impacted model performance. Our reward structure is simplistic and our model may be strengthened through the addition of intermediate rewards. We rewarded only 90-day survival; however, adding intermediate rewards for outcomes related to IV fluid and vasopressor treatment, such as hemodynamic stability, or penalties for lengthy hospitalizations may help the model converge towards an optimal policy that more closely resembles a best outcome. Fourth, while Q-learning is an appropriate first modeling approach to the problem, we may achieve a more highly optimized solution through use of more contemporary models such as deep Q learning or other neural network-based approaches. Finally, the data from our study are retrospective and from a single health system. While it is externally validated by the findings of Komorowski et al., more work needs to be done to understand the generalizability of a model such as ours as a clinical decision support tool.

We envision that an AI policy model such as the one developed here may be some day used as a clinical support tool, in which a learning algorithm is embedded into an electronic health system and is able to provide real-time clinical decision alerts. While trained clinicians will be essential for clinical decision making, a learning health system may help move us towards more precise care in which patients are given optimal treatment regimens at optimal times, ultimately improving patient health. Our model is only a proof of concept and much more work is needed before moving it into a clinical setting. However, clinical decision support such as this one, if truly able to reduce sepsis mortality by even a single percent, could potentially save thousands of lives annually. Therefore, clinical decision support tools are much needed in the clinical environment and well-worth further research.

Appendix A – Supplementary Tables and Figures

Appendix Table 1: Directionality and Transformations for K-means

Feature	Directionality[a]	Transformation[b]
Age	Maximum	-
Albumin	Maximum	-
ALT	Maximum	Ln
AST	Maximum	Ln
Base Excess	Maximum	-
Bicarbonate	Maximum	-
Bilirubin	Maximum	Ln
BUN	Maximum	Ln
Chloride	Maximum	-
Creatinine	Maximum	Ln
Diastolic BP	Maximum	Ln
Elixhauser	Maximum	-
FiO2	Maximum	-
GCS	Minimum	-
Gender	-	-
Glucose	Maximum	Ln
Hemoglobin	Maximum	-
Heart Rate	Minimum	-
INR	Maximum	Ln
Potassium	Maximum	-
Serum Lactate	Minimum	Ln
MAP	Minimum	Ln
Mech Vent	-	-
Sodium	Maximum	-
SaO_2	Minimum	Inverse Ln
$PaCO_2$	Maximum	Ln
PaO_2	Maximum	Ln
PF Ratio	Maximum	Ln
Arterial pH	Maximum	-
Platelets	Maximum	Ln
Resp Rate	Maximum	-
Systolic BP	Maximum	Ln
Shock Index	Maximum	Ln
SIRS	Maximum	-
SOFA	Maximum	-
Temperature	Maximum	-
WBC Count	Minimum	Ln
Weight	Maximum	Ln

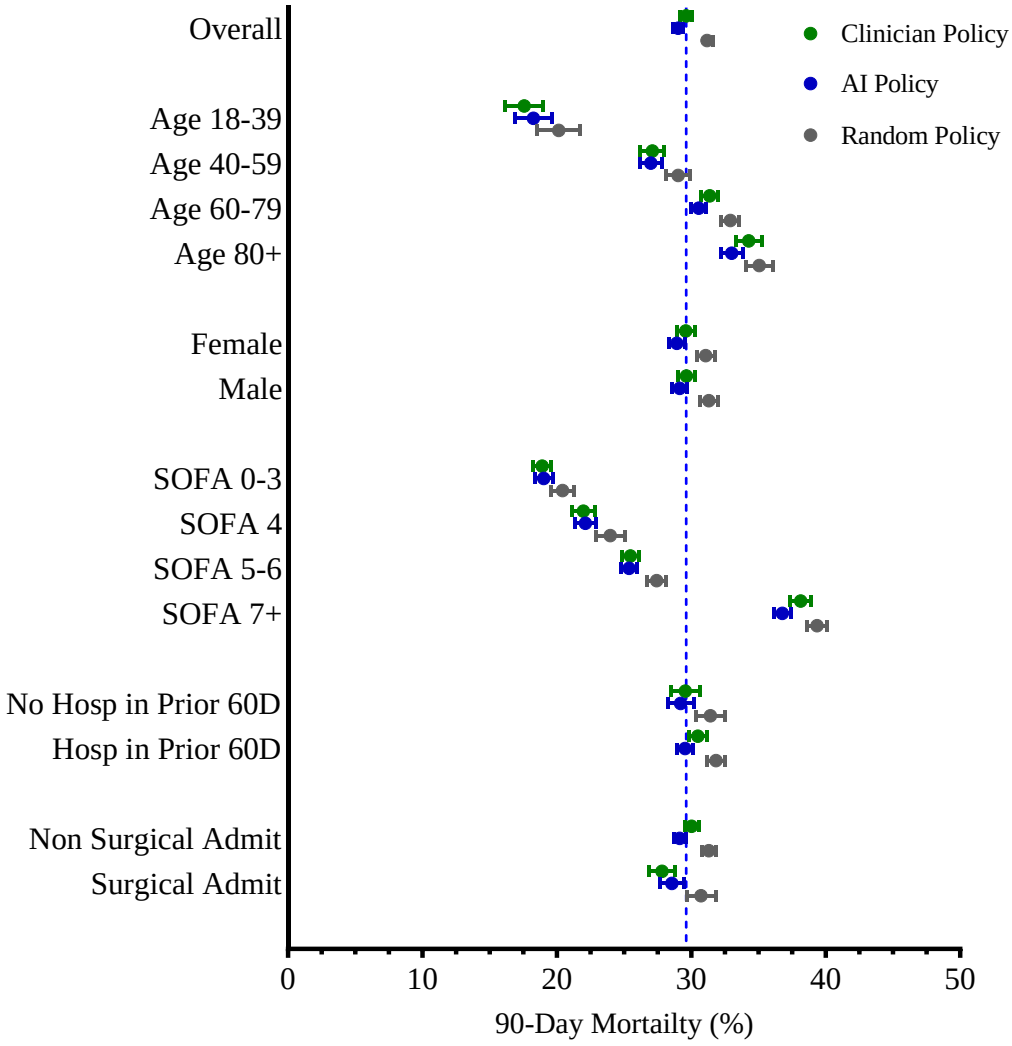
[a]Directionality denotes value selected if multiple measures performed in window
[b]Features z-transformed prior to K-means
Abbreviations: ALT, alanine aminotransferase; AST, aspartate aminotransferase; BP, Blood Pressure; BUN, Blood urea nitrogen; FiO2, fraction of inspired oxygen; GCS, Glasgow Coma Scale score; INR, international normalized ratio; MAP, mean arterial pressure; Mech Vent, mechanical ventilation; PaCO2, partial pressure of arterial carbon dioxide; PaO2, partial pressure of arterial oxygen; PF Ratio, Ratio of PaO2 to FiO2; Resp Rate, respiratory rate; SaO2, oxygen saturation; SIRS, systemic inflammatory response syndrome; SOFA, sequential organ failure assessment; WBC, White Blood Cell

Appendix Table 2: Select Model Feature Featured from State with 100% Mortality

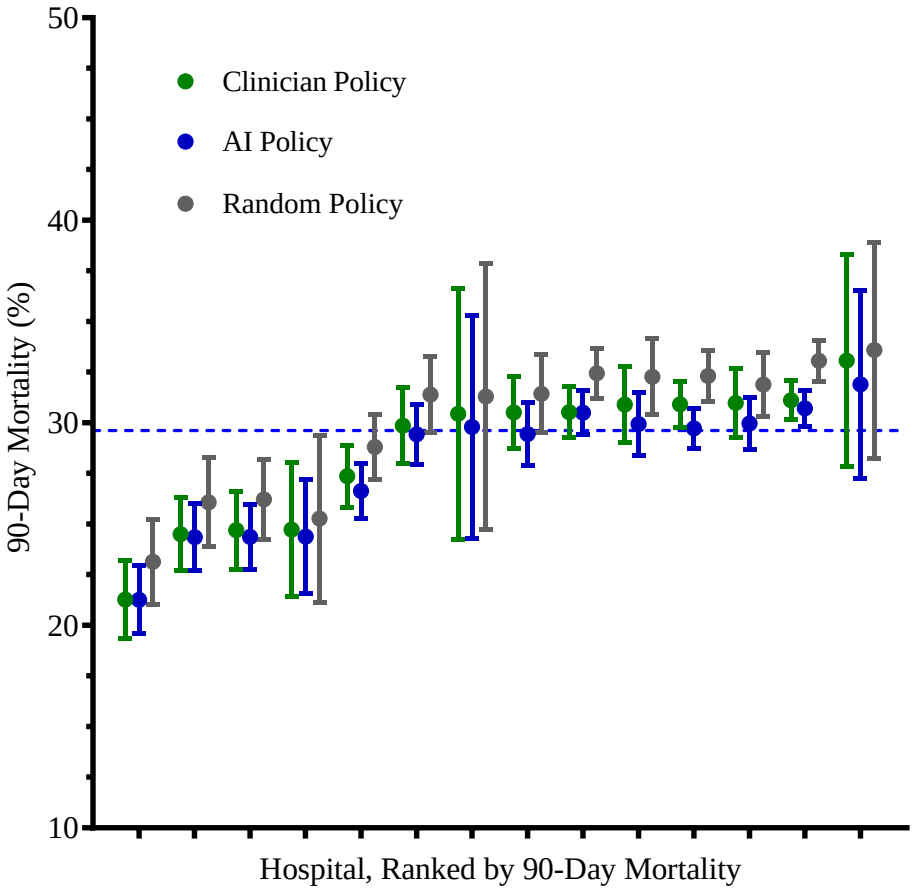
Feature	Value[a]
No. of Sample Points	136
Patient Characteristics	
SOFA Score	13.2
SIRS Criteria	2.8
ALT	1,399
AST	2,604
Glasgow Coma Scale Score	5.2
Serum Lactate	10.2
Mean Arterial Pressure	53
Shock Index	1.2
Outcome	
90-Day Mortality, n (%)	136 (100%)

[a] Mean presented for each feature

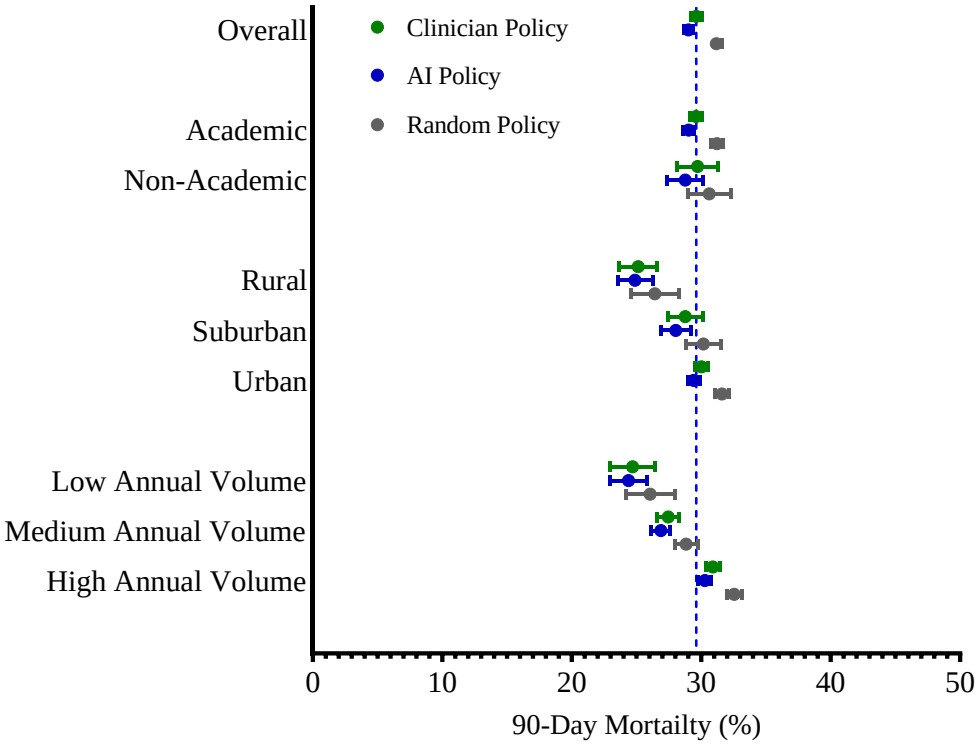
Abbreviations: ALT, alanine aminotransferase; AST, aspartate aminotransferase; SIRS, systemic inflammatory response syndrome; SOFA, sequential organ failure assessment



Appendix Figure 1: 90-Day Mortality by Policy; Patient-level Subgroups



Appendix Figure 2: 90-Day Mortality by Policy; Ranked by Hospital



Appendix Figure 3: 90-Day Mortality by Policy; Hospital-level Subgroups

Appendix B – Statistical Code

The following code was for combining and cleaning raw data files for analysis. Data cleaning was conducted in STATA 16:

```
log using "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Analysis\1 - Sepsis AI - Data
Cleaning - V2.log", replace

*Goals for this file:
* 1) Generate clean pressor data
* 2) Combine data files, label variables, change variable format as necessary
* 3) Restrict to hospitalizations of interest
* 4) Create cohort: 1 - Sepsis in 6hr, in ICU
* 5) Create carry forward data files for use in imputation (labs carried 24 hrs, vitals carried 4
hrs)

***CLEAN PRESSORS***
*Generate Clean Pressor Data File
clear
use "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Raw\Lowenstein_TR34999_V1_2020-02-
13_Step4_Vaso_LongFile_Output_SAFE_HARBORED.dta"
append using "C:\Users\Jason\Box Sync\Current Grants\Sepsis
AI\Data\Raw\Lowenstein_Spec2_TR36024_Step4_VasoLongFile_SAFE_HARBORED.dta"
merge m:1 hosp_id using "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Raw\dtt0.dta",
keep(match) nogen

*Clean Pressors - NOTE Pressors DO NOT include Dobutamine or Milrinone
keep if units == "mcg/kg/min" | units == "mcg/min" | units == "unit(s)/min"
replace pressor_name = "Norepinephrine" if pressor_name == "norepinephrine"
replace pressor_name = "Epinephrine" if pressor_name == "epinephrine"
replace pressor_name = "Dopamine" if pressor_name == "dopamine"
replace pressor_name = "Vasopressin" if pressor_name == "vasopressin"
replace pressor_name = "Phenylephrine" if pressor_name == "phenylephrine"

*Generate Norepi Equivalents - Note: here is where Dobutamine and Milrinone are cut
gen norepi_conv_factor = .
replace norepi_conv_factor = 1 if pressor_name == "Norepinephrine" | pressor_name ==
"Epinephrine"
replace norepi_conv_factor = 0.01 if pressor_name == "Dopamine"
replace norepi_conv_factor = 5 if pressor_name == "Vasopressin"
replace norepi_conv_factor = 0.45 if pressor_name == "Phenylephrine"
drop if norepi_conv_factor == .

*Generate Norepi equivalents
gen nor_equiv = dose * norepi_conv_factor
drop if nor_equiv > 1 & nor_equiv != .

*Generate a variable for time interval that observation falls in
gen interval_hours = (io_dt - dt_t0_first) / 3600000
gen interval = .

local row = 1
forvalues num = -24(4)44 {
  replace interval = `row' if interval_hours >= `num' & interval_hours < (`num'+4)
  local ++row
}

*Sum the Norepi equivalents in the time window
sort hosp_id interval
bysort hosp_id interval: gen norepi_equiv = sum(nor_equiv)
gsort +hosp_id +interval -norepi_equiv
```

```
duplicates drop hosp_id interval, force
keep hosp_id interval norepi_equiv
sort hosp_id interval

save "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Raw\Pressors - Clean - V1.dta",
replace


**CREATE DATA FILE WITH 1 ROW PER 4-HR TIME BLOCK PER ENCOUNTER
*Load Data - Long File of Everyone with a Suspected Infection
clear
use "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Raw\Lowenstein_TR34999_V1_2020-02-
13_Step4_Output_SAFE_HARBORED.dta"
append using "C:\Users\Jason\Box Sync\Current Grants\Sepsis
AI\Data\Raw\Lowenstein_Spec2_Step4_TR36024_SAFE_HARBORED.dta"

*Merge in Wide File Data for Suspected Infection Cohort
recast str44 hosp_id
merge m:1 hosp_id using "C:\Users\Jason\Box Sync\Current Grants\Sepsis
AI\Data\Raw\Lowenstein_TR34999_V1_2020-02-13_Step3_Output_SAFE_HARBORED.dta", keep(match) nogen
merge 1:1 hosp_id interval using "C:\Users\Jason\Box Sync\Current Grants\Sepsis
AI\Data\Raw\Lowenstein_SOFAResp_components_Step4_SAFE_HARBORED.dta", keep(match) nogen
merge m:1 hosp_id using "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Raw\Sepsis AI -
SOFA 6.dta", keep(match) nogen

***KEEP ONLY ENCOUNTERS OF INTEREST***
*Generate Time from Enc Start to dtt0
gen dtt0_hours = (dt_t0_first - enc_start_dt)/3600000

*Keep 2013-2017 years
keep if admit_year >= 2013 & admit_year <= 2017
*Keep if SOFA >=2
keep if sofa_6 >= 2 & sofa_6 != .
*Keep if suspected infection within 6 hours
keep if dtt0_hours >= 0 & dtt0_hours <= 6
*Drop if not full care
drop if cpr == 1
drop cs_fullcare cs_not_fullcare

*Drop data points where patient isn't in hospital yet
drop if dtt0_hours < 0 | dtt0_hours > 24
drop if interval == 1 & dtt0_hours < 22
drop if interval == 2 & dtt0_hours < 18
drop if interval == 3 & dtt0_hours < 14
drop if interval == 4 & dtt0_hours < 10
drop if interval == 5 & dtt0_hours < 6
drop if interval == 6 & dtt0_hours < 2

gen post_dtt0_hours = (enc_end_dt - dt_t0_first)/3600000
drop if interval == 18 & post_dtt0_hours < 46
drop if interval == 17 & post_dtt0_hours < 42
drop if interval == 16 & post_dtt0_hours < 38
drop if interval == 15 & post_dtt0_hours < 34
drop if interval == 14 & post_dtt0_hours < 30
drop if interval == 13 & post_dtt0_hours < 26
drop if interval == 12 & post_dtt0_hours < 22
drop if interval == 11 & post_dtt0_hours < 18
drop if interval == 10 & post_dtt0_hours < 14
drop if interval == 9 & post_dtt0_hours < 10
drop if interval == 8 & post_dtt0_hours < 6

*Drop extra variable and sort
drop post_dtt0_hours
sort hosp_id interval


**CLEAN DATA**
*Change Bad Labs/Vitals to missing (SBP, DBP, MAP, HR, RR, PF Ratio, WBC, PaO2, Lactate, Creat,
Bili == 0, DBP == 0, Negative Urine Output, Urine Output Over 4L/4hr or Fluid Input > 8L/4hr;
expert-determined cutoffs)
replace sbp = . if sbp < 40
```

```
replace dbp = . if dbp < 40
replace hr = . if hr < 40
replace rr = . if rr < 4
replace wbc = . if wbc == 0 | (wbc > 60 & wbc != .)
replace plt = . if plt > 1000 & plt != .

*Generate MAP from 2/3 DBP + 1/3 SBP
gen map = ((2*dbp) + (sbp))/3
replace map = . if dbp == . | sbp == .

*Gen PF Ratio
*Note: sofa_resp_sao2 has already been adjusted using SOFA Respiratory Logic Code
gen pf_ratio = .
replace pf_ratio = 100 * sofa_resp_sao2 / sofa_resp_fio2 if (sofa_resp_sao2 != . & sofa_resp_fio2
!= . & sofa_resp_pao2 == .)
replace pf_ratio = 100 * sofa_resp_pao2 / sofa_resp_fio2 if (sofa_resp_pao2 != . & sofa_resp_fio2
!= .)
drop sofa_resp sofa_resp_fio2 sofa_resp_pao2 sofa_resp_pao2_type sofa_resp_sao2 sofa_resp_mv

foreach var of varlist pf_ratio pao2 lactate creat bili {
  replace `var' = . if `var' == 0
}

*Fix case with GCS==2 (Note: Chart reviewed)
replace gcs = 3 if gcs == 2

*Generate Base Excess
gen base_excess = (0.02786*pco2*10^(ph-6.1))+(13.77*ph)-124.58
rename pco2 paco2

*Gen Fluid Variables
bysort hosp_id (interval) : gen fluid_sum = sum(fluid)
bysort hosp_id (interval) : gen urine_sum = sum(urine)
gen fluid_balance = fluid_sum - urine_sum

*Gen Shock Index
gen shock_index = hr/sbp
replace shock_index = . if hr == . | sbp == .

*Generate variable for if in ICU during window
rename icu icu_ever
replace icu_ever = 0 if icu_ever == .
gen icu_start_hours = (icu_start_dt - dt_t0_first) / 3600000
gen icu_end_hours = (icu_end_dt - dt_t0_first) / 3600000

gen icu = 0
replace icu = 1 if interval == 1 & icu_start_hours <= -20 & icu_end_hours > -24 & icu_ever == 1
replace icu = 1 if interval == 2 & icu_start_hours <= -16 & icu_end_hours > -20 & icu_ever == 1
replace icu = 1 if interval == 3 & icu_start_hours <= -12 & icu_end_hours > -16 & icu_ever == 1
replace icu = 1 if interval == 4 & icu_start_hours <= -8 & icu_end_hours > -12 & icu_ever == 1
replace icu = 1 if interval == 5 & icu_start_hours <= -4 & icu_end_hours > -8 & icu_ever == 1
replace icu = 1 if interval == 6 & icu_start_hours <= 0 & icu_end_hours > -4 & icu_ever == 1
replace icu = 1 if interval == 7 & icu_start_hours <= 4 & icu_end_hours > 0 & icu_ever == 1
replace icu = 1 if interval == 8 & icu_start_hours <= 8 & icu_end_hours > 4 & icu_ever == 1
replace icu = 1 if interval == 9 & icu_start_hours <= 12 & icu_end_hours > 8 & icu_ever == 1
replace icu = 1 if interval == 10 & icu_start_hours <= 16 & icu_end_hours > 12 & icu_ever == 1
replace icu = 1 if interval == 11 & icu_start_hours <= 20 & icu_end_hours > 16 & icu_ever == 1
replace icu = 1 if interval == 12 & icu_start_hours <= 24 & icu_end_hours > 20 & icu_ever == 1
replace icu = 1 if interval == 13 & icu_start_hours <= 28 & icu_end_hours > 24 & icu_ever == 1
replace icu = 1 if interval == 14 & icu_start_hours <= 32 & icu_end_hours > 28 & icu_ever == 1
replace icu = 1 if interval == 15 & icu_start_hours <= 36 & icu_end_hours > 32 & icu_ever == 1
replace icu = 1 if interval == 16 & icu_start_hours <= 40 & icu_end_hours > 36 & icu_ever == 1
replace icu = 1 if interval == 17 & icu_start_hours <= 44 & icu_end_hours > 40 & icu_ever == 1
replace icu = 1 if interval == 18 & icu_start_hours <= 48 & icu_end_hours > 44 & icu_ever == 1

drop icu_start_hours icu_end_hours

*Add in Pressors
merge 1:1 hosp_id interval using "C:\Users\Jason\Box Sync\Current Grants\Sepsis
AI\Data\Raw\Pressors - Clean - V1.dta", keep(match master) nogen
replace norepi_equiv = 0 if norepi_equiv == .
```

```
sort hosp_id interval
gen max_prev_norepi_equiv = 0
bys hosp_id: replace max_prev_norepi_equiv = max(max_prev_norepi_equiv[_n-1],norepi_equiv[_n-
1],norepi_equiv[_n-2],norepi_equiv[_n-3],norepi_equiv[_n-4],norepi_equiv[_n-5],norepi_equiv[_n-
6],norepi_equiv[_n-7],norepi_equiv[_n-8],norepi_equiv[_n-9],norepi_equiv[_n-10],norepi_equiv[_n-
11],norepi_equiv[_n-12],norepi_equiv[_n-13],norepi_equiv[_n-14],norepi_equiv[_n-
15],norepi_equiv[_n-16],norepi_equiv[_n-17]) if max_prev_norepi_equiv[_n-1] != .

*Drop Encounters with Fluid Balance Data (Negative Urine Output, Urine Output Over 4L/4hr or
Fluid Input > 8L/4hr), Gender missing
sort hosp_id interval
gen drop_tag = 0
replace drop_tag = 1 if urine < 0
replace drop_tag = 1 if urine > 4000 & urine != .
replace drop_tag = 1 if fluid > 8000 & fluid != .
by hosp_id: egen max_drop = max(drop_tag)
drop if max_drop == 1
drop drop_tag max_drop
drop if gender == .

*Generate 90-Day Mortality
gen enc_start_date = dofc(enc_start_dt)
format enc_start_date %td
gen death_days = death_date - enc_start_date if death_date - enc_start_date >= 0
gen dead_90 = 0
replace dead_90 = 1 if death_days <= 90
drop enc_start_date

*Define Actions 1-25 for the 25 squares on grid
replace fluid = 0 if fluid == .
replace norepi_equi = 0 if norepi_equi == .
sum fluid if fluid > 0, de
sum norepi_equi if norepi_equi > 0, de
gen action = .
replace action = 1 if fluid == 0 & norepi_equiv == 0
replace action = 2 if fluid == 0 & norepi_equiv > 0 & norepi_equiv <= 0.09
replace action = 3 if fluid == 0 & norepi_equiv > 0.09 & norepi_equiv <= 0.2
replace action = 4 if fluid == 0 & norepi_equiv > 0.2 & norepi_equiv <= 0.5
replace action = 5 if fluid == 0 & norepi_equiv > 0.5 & norepi_equiv != .
replace action = 6 if fluid > 0 & fluid <= 250 & norepi_equiv == 0
replace action = 7 if fluid > 0 & fluid <= 250 & norepi_equiv > 0 & norepi_equiv <= 0.09
replace action = 8 if fluid > 0 & fluid <= 250 & norepi_equiv > 0.09 & norepi_equiv <= 0.2
replace action = 9 if fluid > 0 & fluid <= 250 & norepi_equiv > 0.2 & norepi_equiv <= 0.5
replace action = 10 if fluid > 0 & fluid <= 250 & norepi_equiv > 0.5 & norepi_equiv != .
replace action = 11 if fluid > 250 & fluid <= 400 & norepi_equiv == 0
replace action = 12 if fluid > 250 & fluid <= 400 & norepi_equiv > 0 & norepi_equiv <= 0.09
replace action = 13 if fluid > 250 & fluid <= 400 & norepi_equiv > 0.09 & norepi_equiv <= 0.2
replace action = 14 if fluid > 250 & fluid <= 400 & norepi_equiv > 0.2 & norepi_equiv <= 0.5
replace action = 15 if fluid > 250 & fluid <= 400 & norepi_equiv > 0.5 & norepi_equiv != .
replace action = 16 if fluid > 400 & fluid <= 700 & norepi_equiv == 0
replace action = 17 if fluid > 400 & fluid <= 700 & norepi_equiv > 0 & norepi_equiv <= 0.09
replace action = 18 if fluid > 400 & fluid <= 700 & norepi_equiv > 0.09 & norepi_equiv <= 0.2
replace action = 19 if fluid > 400 & fluid <= 700 & norepi_equiv > 0.2 & norepi_equiv <= 0.5
replace action = 20 if fluid > 400 & fluid <= 700 & norepi_equiv > 0.5 & norepi_equiv != .
replace action = 21 if fluid > 700 & fluid != . & norepi_equiv == 0
replace action = 22 if fluid > 700 & fluid != . & norepi_equiv > 0 & norepi_equiv <= 0.09
replace action = 23 if fluid > 700 & fluid != . & norepi_equiv > 0.09 & norepi_equiv <= 0.2
replace action = 24 if fluid > 700 & fluid != . & norepi_equiv > 0.2 & norepi_equiv <= 0.5
replace action = 25 if fluid > 700 & fluid != . & norepi_equiv > 0.5 & norepi_equiv != .

*Order dataset logically, drop unnecessary vars, label data
replace cx_type = "C Diff" if cx_type == "C diff"
drop mrn_id cpr dx1-dx20 p1-p20 abx_t abx_dur cx_first
order hosp_id interval empi_id sofa_total sirs_total age alb alt ast base_excess bicarb bili bun
cl creat dbp elix fio2 gcs gender gluc hr hgb icu inr lactate map mechvent paco2 pao2 pf_ratio ph
plt k rr shock_index na o2_sat sbp temp wbc weight fluid fluid_sum urine urine_sum fluid_balance
norepi_equiv max_prev_norepi_equiv enc_start_dt enc_end_dt dt_t0_first dtt0_hours death_date
icu_start_dt icu_end_dt race admit_year hospital sofa_24 sofa_dtt0 surg mv_d vp_d hosp_los
icu_ever icu_los dead death_days dead_90 action

label variable hosp_id "Encrypted Encounter ID"
```

```
label variable interval "Time Window"
label variable empi_id "Encrypted Patient ID"
label variable sofa_total "SOFA in Window"
label variable sirs_total "SIRS in Window"
label variable age "Age (years)"
label variable alb "Albumin"
label variable alt "ALT"
label variable ast "AST"
label variable base_excess "Base Excess"
label variable bicarb "Bicarbonate"
label variable bili "Bilirubin"
label variable bun "BUN"
label variable cl "Chloride"
label variable creat "Creatinine"
label variable dbp "Diastolic BP"
label variable elix "Elixhauser"
label variable fio2 "FiO2"
label variable gcs "GCS"
label variable gender "Gender; 1=male,0=female"
label variable gluc "Glucose"
label variable hr "Heart Rate"
label variable hgb "Hemoglobin"
label variable icu "ICU in Window"
label variable inr "INR"
label variable lactate "Serum Lactate"
label variable map "MAP"
label variable mechvent "Mech Vent in Window"
replace mechvent = 0 if mechvent == .
label variable paco2 "PaCO2"
label variable pao2 "PaO2"
label variable pf_ratio "PF Ratio"
label variable ph "Arterial pH"
label variable plt "Platelets"
label variable k "Potassium"
label variable rr "Resp Rate"
label variable shock_index "Shock Index"
label variable na "Sodium"
label variable o2_sat "SaO2"
label variable sbp "Systolic BP"
label variable temp "Temperature"
label variable wbc "WBC Count"
label variable weight "Weight (kg)"
label variable fluid "Fluid Input in Window"
label variable fluid_sum "Cumulative Fluid Input"
label variable urine "Fluid Output in Window"
label variable urine_sum "Cumulative Fluid Output"
label variable fluid_balance "Cumulative Fluid Balance"
label variable norepi_equiv "Vasopressor in Window (Norepi Equiv)"
label variable max_prev_norepi_equiv "Max Vasopressor in Prior Windows (Norepi Equiv)"
label variable enc_start_dt "Encounter Start Date-Time"
label variable enc_end_dt "Encounter End Date-Time"
label variable dt_t0_first "Suspected Infection Date-Time"
label variable death_date "Death Date-Time"
label variable icu_start_dt "ICU Admit Date-Time"
label variable icu_end_dt "ICU Discharge Date-Time"
label variable race "Race; 1=white,2=black,3=other"
label variable admit_year "Year of Admission"
label variable hospital "Hospital"
label variable sofa_24 "Max SOFA in 1st 24 hours"
label variable sofa_dtt0 "Max SOFA in -24 to +48 hrs around dtt0"
label variable surg "Surgery ever during admission"
label variable mv_d "Days of Mechanical Ventilation"
label variable vp_d "Days of Vasopressors"
label variable hosp_los "Hospital Length of Stay (days)"
label variable icu_ever "ICU ever in encounter"
label variable icu_los "ICU Length of Stay (days)"
label variable dead "Inhospital Death"
label variable death_days "Days to Death"
label variable dead_90 "90-Day Mortality"
label variable dtt0_hours "Enc Start to Dtt0 (hrs)"
```

```
label variable action "Matrix of 25 Actions; Norepi increases 1,2,3,4,5; Fluid increases
1,6,11,16,21"

*Save Data
keep if icu == 1
save "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Sepsis AI - ICU Cohort - Original -
V1.dta", replace

**Generate Pre-Imputation, Pre-carryforward table of patient features (39 vars, all time
intervals)
*ICU Cohort
table1 if icu==1, vars(age contn %12.0f \ alb contn %12.1f \ alt conts %12.0f \ ast conts %12.0f
\ base_excess contn %12.1f \ bicarb contn %12.0f \ bili conts %12.1f \ bun conts %12.0f \ cl
contn %12.0f \ creat conts %12.1f \ dbp conts %12.1f \ elix contn %12.1f \ fio2 conts %12.0f \
gcs contn %12.1f \ gender cat %12.1f \ gluc conts %12.0f \ hgb contn %12.0f \ hr contn %12.0f \
icu cat %12.0%f \ inr conts %12.1f \ k contn %12.0f \ lactate conts %12.1f \ map conts %12.1f \
mechvent cat %12.0f \ na contn %12.0f \ o2_sat conts %12.0f \ paco2 contn %12.0f \ pao2 contn
%12.0f \ pf_ratio conts %12.0f \ ph contn %12.1f \ plt conts %12.0f \ rr contn %12.0f \ sbp conts
%12.1f \ shock_index contn %12.1f \ sirs_total contn %12.1f \ sofa_total contn %12.1f \ temp
contn %12.1f \ wbc conts %12.0f \ weight contn %12.0f) saving("C:\Users\Jason\Box Sync\Current
Grants\Sepsis AI\Analysis\Sepsis AI - ICU Cohort - PreCarryforward Features - V1.xlsx", replace)

**GENERATE Basic Table 1
gen mv = 0
replace mv = 1 if mv_d > 0 & mv_d != .
gen vp = 0
replace vp = 1 if vp_d > 0 & vp_d != .

preserve
duplicates drop hosp_id, force
table1, vars(age contn %12.0f \ gender cat %12.1f \ race cat %12.1f \ sofa_total contn %12.1f \
elix contn %12.1f \ fluid_sum conts %12.1f \ urine_sum conts %12.1f \ pbc cat %12.1f \ pos_pbc
cat %12.1f \ cx_source cat %12.1f \ cx_type cat %12.1f \ surg cat %12.1f \ icu_ever cat 12.1f \
mv cat %12.1f \ mv_d conts %12.0f \ vp cat %12.1f \ vp_d conts %12.0f \ hosp_los conts %12.0f \
icu_los conts %12.0f \ dead cat %12.1f \ dead_90 cat %12.1f) saving("C:\Users\Jason\Box
Sync\Current Grants\Sepsis AI\Analysis\Sepsis AI - ICU Cohort - Table 1 - V1.xlsx", replace)
restore

*Table 1 - Clinical Variables in 1st 4 hours after Sepsis onset (1 row per patient; ICU cohort
only includes patients)
*ICU Cohort
bysort hosp_id: egen icu_inwin = max(icu)
table1 if interval==7 & icu_inwin==1, vars(age contn %12.0f \ alb contn %12.1f \ alt conts %12.0f
\ ast conts %12.0f \ base_excess contn %12.1f \ bicarb contn %12.0f \ bili conts %12.1f \ bun
conts %12.0f \ cl contn %12.0f \ creat conts %12.1f \ dbp conts %12.1f \ elix contn %12.1f \ fio2
conts %12.0f \ gcs contn %12.1f \ gender cat %12.1f \ gluc conts %12.0f \ hgb contn %12.0f \ hr
contn %12.0f \ icu cat %12.0%f \ inr conts %12.1f \ k contn %12.0f \ lactate conts %12.1f \ map
conts %12.1f \ mechvent cat %12.0f \ na contn %12.0f \ o2_sat conts %12.0f \ paco2 contn %12.0f \
pao2 contn %12.0f \ pf_ratio conts %12.0f \ ph contn %12.1f \ plt conts %12.0f \ rr contn %12.0f
\ sbp conts %12.1f \ shock_index contn %12.1f \ sirs_total contn %12.1f \ sofa_total contn %12.1f
\ temp contn %12.1f \ wbc conts %12.0f \ weight contn %12.0f) saving("C:\Users\Jason\Box
Sync\Current Grants\Sepsis AI\Analysis\Sepsis AI - ICU Cohort - Table 1 Clinical Features -
V1.xlsx", replace)

*Drop extra variables
drop icu_inwin mv vp pbc pos_pbc cx_source cx_type

*Create a table of missingness for all model features, for each cohort
*ICU Cohort, Pre-Carryforward
putexcel set "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Analysis\Sepsis AI - Missingness -
V1.xlsx", sheet("ICU_Precarry") modify
quietly: putexcel A1="Missingness, Pre-Carryforward, ICU Cohort" A2="Feature" B2="Missingness
(%)"
local row=3
foreach var of varlist age alb alt ast base_excess bicarb bili bun cl creat dbp elix fio2 gcs
gender gluc hgb hr icu inr k lactate map mechvent na o2_sat paco2 pao2 pf_ratio ph plt rr sbp
shock_index sirs_total sofa_total temp wbc weight {
mdesc `var' if icu == 1
quietly: putexcel A`row'="`var'" B`row'=(r(miss)) C`row'=(r(total)) D`row'=(r(percent))
local ++row
}
```

```
***CARRY FORWARD PRIOR TO IMPUTATION***
*Carry forward 1 time block for vitals*
gsort +hosp_id -interval
foreach var of varlist base_excess dbp fio2 gcs hr map rr shock_index o2_sat paco2 pao2 pf_ratio
ph sbp temp {
  forvalues i = 18(-1)1 {
    replace `var' = `var'[_n+1] if `var' == . & hosp_id == hosp_id[_n+1] & interval == `i'
  }
}

*Carry forward 6 time blocks for labs*
foreach var of varlist alb alt ast bicarb bili bun cl creat gluc hgb inr lactate plt k na wbc {
  forvalues i = 18(-1)1 {
    replace `var' = `var'[_n+1] if `var' == . & hosp_id == hosp_id[_n+1] & interval == `i'
replace `var' = `var'[_n+2] if `var' == . & `var'[_n+1] == . & hosp_id == hosp_id[_n+2] &
interval == `i'
    replace `var' = `var'[_n+3] if `var' == . & `var'[_n+1] == . & `var'[_n+2] == . & hosp_id ==
hosp_id[_n+3] & interval == `i'
    replace `var' = `var'[_n+4] if `var' == . & `var'[_n+1] == . & `var'[_n+2] == . & `var'[_n+3]
== . & hosp_id == hosp_id[_n+4] & interval == `i'
    replace `var' = `var'[_n+5] if `var' == . & `var'[_n+1] == . & `var'[_n+2] == . & `var'[_n+3]
== . & `var'[_n+4] == . & hosp_id == hosp_id[_n+5] & interval == `i'
    replace `var' = `var'[_n+6] if `var' == . & `var'[_n+1] == . & `var'[_n+2] == . & `var'[_n+3]
== . & `var'[_n+4] == . & `var'[_n+5] == . & hosp_id == hosp_id[_n+6] & interval == `i'
  }
}

***Keep Variables for Imputation, KMeans, Action, Outcome
keep hosp_id interval empi_id sofa_total sirs_total age alb alt ast base_excess bicarb bili bun
cl creat dbp elix fio2 gcs gender gluc hgb hr icu inr k lactate map mechvent na o2_sat paco2 pao2
pf_ratio ph plt rr sbp shock_index temp wbc weight fluid norepi_equiv dtt0_hours dead_90 action
sort hosp_id interval

*ICU Cohort*
keep if icu == 1
save "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Data\Sepsis AI - ICU Cohort - Pre-
Imputation Carryforward - V1.dta", replace

*ICU Cohort, Pre-Carryforward
putexcel set "C:\Users\Jason\Box Sync\Current Grants\Sepsis AI\Analysis\Sepsis AI - Missingness -
V1.xlsx", sheet("ICU_Postcarry") modify
quietly: putexcel A1="Missingness, Pre-Carryforward, ICU Cohort" A2="Feature" B2="Missingness
(%)"
local row=3
foreach var of varlist age alb alt ast base_excess bicarb bili bun cl creat dbp elix fio2 gcs
gender gluc hgb hr icu inr k lactate map mechvent na o2_sat paco2 pao2 pf_ratio ph plt rr sbp
shock_index sirs_total sofa_total temp wbc weight {
mdesc `var' if icu == 1
quietly: putexcel A`row'="`var'" B`row'=(r(miss)) C`row'=(r(total)) D`row'=(r(percent))
local ++row
}

**Generate Pre-Imputation, Post-carryforward table of patient features (39 vars, all time
intervals)
*ICU Cohort
table1, vars(age contn %12.0f \ alb contn %12.1f \ alt conts %12.0f \ ast conts %12.0f \
base_excess contn %12.1f \ bicarb contn %12.0f \ bili conts %12.1f \ bun conts %12.0f \ cl contn
%12.0f \ creat conts %12.1f \ dbp conts %12.1f \ elix contn %12.1f \ fio2 conts %12.0f \ gcs
contn %12.1f \ gender cat %12.1f \ gluc conts %12.0f \ hgb contn %12.0f \ hr contn %12.0f \ icu
cat %12.0%f \ inr conts %12.1f \ k contn %12.0f \ lactate conts %12.1f \ map conts %12.1f \
mechvent cat %12.0f \ na contn %12.0f \ o2_sat conts %12.0f \ paco2 contn %12.0f \ pao2 contn
%12.0f \ pf_ratio conts %12.0f \ ph contn %12.1f \ plt conts %12.0f \ rr contn %12.0f \ sbp conts
%12.1f \ shock_index contn %12.1f \ sirs_total contn %12.1f \ sofa_total contn %12.1f \ temp
contn %12.1f \ wbc conts %12.0f \ weight contn %12.0f) saving("C:\Users\Jason\Box Sync\Current
Grants\Sepsis AI\Analysis\Sepsis AI - ICU Cohort - PostCarryforward Features - V1.xlsx", replace)

log close
```

Imputation was conducted in R (4.0.3) using the missRanger() package. Note that I avoided using loops due to computation time (in this section and in the Kmeans computations). Code is as follows:

```
# Sepsis AI - Reinforcement Learning with Q Learning
# Imputation using Random Forest Imputation Instead of MICE

#Load Libraries
library(haven)
library(missRanger)
library(foreign)
library(data.table)

###ICU COHORT###
#Load Data
setwd("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data")
data <- read_dta("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI - ICU
Cohort - Pre-Imputation Carryforward - V1.dta")

#ICU Cohort - Run Random Forest Imputation x 25 datasets (note: "icu" removed b/c ==1 for all)
#Note -- Not looped due to computation time
set.seed(1208102301)
data.imputed.01 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102302)
data.imputed.02 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102303)
data.imputed.03 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102304)
data.imputed.04 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102305)
data.imputed.05 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102306)
data.imputed.06 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102307)
data.imputed.07 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102308)
data.imputed.08 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102309)
data.imputed.09 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102310)
data.imputed.10 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102311)
data.imputed.11 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102312)
data.imputed.12 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102313)
data.imputed.13 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102314)
data.imputed.14 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102315)
data.imputed.15 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102316)
data.imputed.16 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102317)
data.imputed.17 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102318)
data.imputed.18 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102319)
data.imputed.19 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102320)
data.imputed.20 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102321)
data.imputed.21 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102322)
data.imputed.22 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102323)
data.imputed.23 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
set.seed(1208102324)
data.imputed.24 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)
```

```
set.seed(1208102325)
data.imputed.25 <- missRanger(data[,c(4:23,25:42)],pmm.k=5,num.trees=50,sample.fraction=0.1)

#Add Identifiers to Imputation
data.imputed.01 <- cbind(data[,1:3],data.imputed.01)
data.imputed.02 <- cbind(data[,1:3],data.imputed.02)
data.imputed.03 <- cbind(data[,1:3],data.imputed.03)
data.imputed.04 <- cbind(data[,1:3],data.imputed.04)
data.imputed.05 <- cbind(data[,1:3],data.imputed.05)
data.imputed.06 <- cbind(data[,1:3],data.imputed.06)
data.imputed.07 <- cbind(data[,1:3],data.imputed.07)
data.imputed.08 <- cbind(data[,1:3],data.imputed.08)
data.imputed.09 <- cbind(data[,1:3],data.imputed.09)
data.imputed.10 <- cbind(data[,1:3],data.imputed.10)
data.imputed.11 <- cbind(data[,1:3],data.imputed.11)
data.imputed.12 <- cbind(data[,1:3],data.imputed.12)
data.imputed.13 <- cbind(data[,1:3],data.imputed.13)
data.imputed.14 <- cbind(data[,1:3],data.imputed.14)
data.imputed.15 <- cbind(data[,1:3],data.imputed.15)
data.imputed.16 <- cbind(data[,1:3],data.imputed.16)
data.imputed.17 <- cbind(data[,1:3],data.imputed.17)
data.imputed.18 <- cbind(data[,1:3],data.imputed.18)
data.imputed.19 <- cbind(data[,1:3],data.imputed.19)
data.imputed.20 <- cbind(data[,1:3],data.imputed.20)
data.imputed.21 <- cbind(data[,1:3],data.imputed.21)
data.imputed.22 <- cbind(data[,1:3],data.imputed.22)
data.imputed.23 <- cbind(data[,1:3],data.imputed.23)
data.imputed.24 <- cbind(data[,1:3],data.imputed.24)
data.imputed.25 <- cbind(data[,1:3],data.imputed.25)

#Take the median of 21 imputations for final imputed dataset for each variable
data.combined <-
rbindlist(list(data.imputed.01,data.imputed.02,data.imputed.03,data.imputed.04,data.imputed.05,da
ta.imputed.06,data.imputed.07,data.imputed.08,data.imputed.09,data.imputed.10,data.imputed.11,dat
a.imputed.12,data.imputed.13,data.imputed.14,data.imputed.15,data.imputed.16,data.imputed.17,data
.imputed.18,data.imputed.19,data.imputed.20,data.imputed.21,data.imputed.22,data.imputed.23,data.
imputed.24,data.imputed.25))[,lapply(.SD,median), list(hosp_id,interval,empi_id)]

#Add Actions and Outcomes to Imputation
data.combined <- cbind(data.combined,data[,43:44],data[,47],data[,46])

###Clean up relational variables after imputation###
#Mean Arterial Pressure - 2/3 DBP + 1/3 SBP
data.combined$map <- ((2*data.combined$dbp) + (data.combined$sbp))/3

#Base Excess - (0.02786*pco2*10^(ph-6.1))+(13.77*ph)-124.58
data.combined$base_excess <- ((0.02786)*(data.combined$paco2)*(10^(data.combined$ph-6.1))) +
(13.77*data.combined$ph) - 124.58

#Shock Index -HR/SBP
data.combined$shock_index <- data.combined$hr/data.combined$sbp

#SOFA
data.sofa <- data.combined[,1:2]
data.sofa$resp <-  ifelse(data.combined$pf_ratio < 100 & data.combined$mechvent == 1,4,
                      ifelse(data.combined$pf_ratio < 200 & data.combined$mechvent == 1,3,
                      ifelse(data.combined$pf_ratio < 300,2,
                      ifelse(data.combined$pf_ratio < 400,1,0))))
data.sofa$neuro <- ifelse(data.combined$gcs <= 5,4,
                      ifelse(data.combined$gcs <= 9,3,
                      ifelse(data.combined$gcs <= 12,2,
                      ifelse(data.combined$gcs <= 14,1,0))))
data.sofa$cv <-    ifelse(data.combined$norepi_equiv > 0.1,4,
                      ifelse(data.combined$norepi_equiv > 0,3,
                      ifelse(data.combined$map < 70,1,0)))
data.sofa$hep <-   ifelse(data.combined$bili >= 12,4,
                      ifelse(data.combined$bili >= 6,3,
                      ifelse(data.combined$bili >= 2,2,
                      ifelse(data.combined$bili >= 1.2,1,0))))
data.sofa$coag <-  ifelse(data.combined$plt < 20,4,
                      ifelse(data.combined$plt < 50,3,
```

```
                      ifelse(data.combined$plt < 100,2,
                      ifelse(data.combined$plt < 150,1,0))))
data.sofa$renal <- ifelse(data.combined$creat >= 5,4,
                      ifelse(data.combined$creat >= 3.5,3,
                      ifelse(data.combined$creat >= 2,2,
                      ifelse(data.combined$creat >= 1.2,1,0))))
data.sofa$total <- rowSums(data.sofa[,3:8])
data.combined$sofa_total <- data.sofa$total

#SIRS
data.sirs      <- data.combined[,1:2]
data.sirs$temp <- ifelse(data.combined$temp < 36.0 | data.combined$temp > 38.0,1,0)
data.sirs$hr   <- ifelse(data.combined$hr > 90,1,0)
data.sirs$rr   <- ifelse(data.combined$rr > 20 | data.combined$paco2 < 32,1,0)
data.sirs$wbc  <- ifelse(data.combined$wbc < 4 | data.combined$wbc > 12,1,0)
data.sirs$total<- rowSums(data.sirs[,3:6])
data.combined$sirs_total <- data.sirs$total

#Save Data
write.dta(data.combined,"C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI -
ICU Cohort - Imputed - V1.dta")
```

K-means was conducted in R (4.0.3) using the kmeans() function:

```
# Sepsis AI - Reinforcement Learning with Q Learning
# Determine States Using Kmeans on Imputed Data

#Load Libraries
library(haven)
library(foreign)
library(depmixS4)

#Create Function to Calculate AIC and BIC from Clustering
kmeansAIC = function(fit){
  m = ncol(fit$centers)
  n = length(fit$cluster)
  k = nrow(fit$centers)
  D = fit$tot.withinss
  return(D + 2*m*k)
}
kmeansBIC = function (fit)
{
  m = ncol(fit$centers)
  n = length(fit$cluster)
  k = nrow(fit$centers)
  D = fit$tot.withinss
  return(D + log(n) * m * k) # using log(n) instead of 2, penalize model complexity
}

#Load Data
setwd("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data")
data.icu <- read_dta("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI -
ICU Cohort - Imputed - V1.dta")

#Log-Transform Features that are far from normal distribution
#Note: left variable names instead of columns for reference
data.icu.ln             <- data.icu
data.icu.ln$alt         <- log(data.icu$alt)
data.icu.ln$ast         <- log(data.icu$ast)
data.icu.ln$bili        <- log(data.icu$bili)
data.icu.ln$bun         <- log(data.icu$bun)
data.icu.ln$creat       <- log(data.icu$creat)
data.icu.ln$dbp         <- log(data.icu$dbp)
data.icu.ln$gluc        <- log(data.icu$gluc)
data.icu.ln$inr         <- log(data.icu$inr)
data.icu.ln$lactate     <- log(data.icu$lactate)
data.icu.ln$map         <- log(data.icu$map)
data.icu.ln$paco2       <- log(data.icu$paco2)
data.icu.ln$pao2        <- log(data.icu$pao2)
data.icu.ln$sbp         <- log(data.icu$sbp)
data.icu.ln$pf_ratio    <- log(data.icu$pf_ratio)
data.icu.ln$plt         <- log(data.icu$plt)
data.icu.ln$shock_index <- log(data.icu$shock_index)
data.icu.ln$wbc         <- log(data.icu$wbc)
data.icu.ln$weight      <- log(data.icu$weight)
data.icu.ln$o2_sat      <- log(101-data.icu$o2_sat)

#Z-Transform Data (Mean=0, SD=1)
data.icu.z <- data.icu.ln
data.icu.z[c(4:19,21:26,28:41)] <- scale(data.icu.ln[c(4:19,21:26,28:41)],center=T,scale=T)
#Scale continuous variables to

#Run K-Means Clustering for 50-2000 Clusters in Intervals of 50
set.seed(12081023)
state.icu.0050 <- kmeans(data.icu.z[, c(4:41)], centers = 50, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0100 <- kmeans(data.icu.z[, c(4:41)], centers = 100, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0150 <- kmeans(data.icu.z[, c(4:41)], centers = 150, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0200 <- kmeans(data.icu.z[, c(4:41)], centers = 200, nstart = 10, iter.max = 1000)
set.seed(12081023)
```

```
state.icu.0250 <- kmeans(data.icu.z[, c(4:41)], centers = 250, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0300 <- kmeans(data.icu.z[, c(4:41)], centers = 300, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0350 <- kmeans(data.icu.z[, c(4:41)], centers = 350, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0400 <- kmeans(data.icu.z[, c(4:41)], centers = 400, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0450 <- kmeans(data.icu.z[, c(4:41)], centers = 450, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0500 <- kmeans(data.icu.z[, c(4:41)], centers = 500, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0550 <- kmeans(data.icu.z[, c(4:41)], centers = 550, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0600 <- kmeans(data.icu.z[, c(4:41)], centers = 600, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0650 <- kmeans(data.icu.z[, c(4:41)], centers = 650, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0700 <- kmeans(data.icu.z[, c(4:41)], centers = 700, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0750 <- kmeans(data.icu.z[, c(4:41)], centers = 750, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0800 <- kmeans(data.icu.z[, c(4:41)], centers = 800, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0850 <- kmeans(data.icu.z[, c(4:41)], centers = 850, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0900 <- kmeans(data.icu.z[, c(4:41)], centers = 900, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.0950 <- kmeans(data.icu.z[, c(4:41)], centers = 950, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1000 <- kmeans(data.icu.z[, c(4:41)], centers = 1000, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1050 <- kmeans(data.icu.z[, c(4:41)], centers = 1050, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1100 <- kmeans(data.icu.z[, c(4:41)], centers = 1100, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1150 <- kmeans(data.icu.z[, c(4:41)], centers = 1150, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1200 <- kmeans(data.icu.z[, c(4:41)], centers = 1200, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1250 <- kmeans(data.icu.z[, c(4:41)], centers = 1250, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1300 <- kmeans(data.icu.z[, c(4:41)], centers = 1300, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1350 <- kmeans(data.icu.z[, c(4:41)], centers = 1350, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1400 <- kmeans(data.icu.z[, c(4:41)], centers = 1400, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1450 <- kmeans(data.icu.z[, c(4:41)], centers = 1450, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1500 <- kmeans(data.icu.z[, c(4:41)], centers = 1500, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1550 <- kmeans(data.icu.z[, c(4:41)], centers = 1550, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1600 <- kmeans(data.icu.z[, c(4:41)], centers = 1600, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1650 <- kmeans(data.icu.z[, c(4:41)], centers = 1650, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1700 <- kmeans(data.icu.z[, c(4:41)], centers = 1700, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1750 <- kmeans(data.icu.z[, c(4:41)], centers = 1750, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1800 <- kmeans(data.icu.z[, c(4:41)], centers = 1800, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1850 <- kmeans(data.icu.z[, c(4:41)], centers = 1850, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1900 <- kmeans(data.icu.z[, c(4:41)], centers = 1900, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.1950 <- kmeans(data.icu.z[, c(4:41)], centers = 1950, nstart = 10, iter.max = 1000)
set.seed(12081023)
state.icu.2000 <- kmeans(data.icu.z[, c(4:41)], centers = 2000, nstart = 10, iter.max = 1000)
```

```
#Store Cluster Assignments in a New Data Frame and Save
states.icu <- data.frame("hosp_id"    = data.icu$hosp_id,
                         "interval"    = data.icu$interval,
                         "State_0050"  = state.icu.0050$cluster,
                         "State_0100"  = state.icu.0100$cluster,
                         "State_0150"  = state.icu.0150$cluster,
                         "State_0200"  = state.icu.0200$cluster,
                         "State_0250"  = state.icu.0250$cluster,
                         "State_0300"  = state.icu.0300$cluster,
                         "State_0350"  = state.icu.0350$cluster,
                         "State_0400"  = state.icu.0400$cluster,
                         "State_0450"  = state.icu.0450$cluster,
                         "State_0500"  = state.icu.0500$cluster,
                         "State_0550"  = state.icu.0550$cluster,
                         "State_0600"  = state.icu.0600$cluster,
                         "State_0650"  = state.icu.0650$cluster,
                         "State_0700"  = state.icu.0700$cluster,
                         "State_0750"  = state.icu.0750$cluster,
                         "State_0800"  = state.icu.0800$cluster,
                         "State_0850"  = state.icu.0850$cluster,
                         "State_0900"  = state.icu.0900$cluster,
                         "State_0950"  = state.icu.0950$cluster,
                         "State_1000"  = state.icu.1000$cluster,
                         "State_1050"  = state.icu.1050$cluster,
                         "State_1100"  = state.icu.1100$cluster,
                         "State_1150"  = state.icu.1150$cluster,
                         "State_1200"  = state.icu.1200$cluster,
                         "State_1250"  = state.icu.1250$cluster,
                         "State_1300"  = state.icu.1300$cluster,
                         "State_1350"  = state.icu.1350$cluster,
                         "State_1400"  = state.icu.1400$cluster,
                         "State_1450"  = state.icu.1450$cluster,
                         "State_1500"  = state.icu.1500$cluster,
                         "State_1550"  = state.icu.1550$cluster,
                         "State_1600"  = state.icu.1600$cluster,
                         "State_1650"  = state.icu.1650$cluster,
                         "State_1700"  = state.icu.1700$cluster,
                         "State_1750"  = state.icu.1750$cluster,
                         "State_1800"  = state.icu.1800$cluster,
                         "State_1850"  = state.icu.1850$cluster,
                         "State_1900"  = state.icu.1900$cluster,
                         "State_1950"  = state.icu.1950$cluster,
                         "State_2000"  = state.icu.2000$cluster)

write.dta(states.icu,"C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI -
ICU Cohort - States - V1.dta")
save(state.icu.0750, file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI
- ICU Cohort - 750 States - V1.RData")

#Generate a Data Frame with AIC, BIC, Within Cluster SSE, and Prop of Variance Explained for k=50
to 2000 by 50s
#Generate Empty Data Frame
state.assess.icu <- data.frame("Num States" = seq(from = 50,to = 2000,by = 50),
                               "AIC"=0,
                               "BIC"=0,
                               "Within State SSE"=0,
                               "Prop of Var Explained"=0,
                               "TotSS",
                               "BetweenSS")

#AIC
state.assess.icu[1,2]  <- kmeansAIC(state.icu.0050)
state.assess.icu[2,2]  <- kmeansAIC(state.icu.0100)
state.assess.icu[3,2]  <- kmeansAIC(state.icu.0150)
state.assess.icu[4,2]  <- kmeansAIC(state.icu.0200)
state.assess.icu[5,2]  <- kmeansAIC(state.icu.0250)
state.assess.icu[6,2]  <- kmeansAIC(state.icu.0300)
state.assess.icu[7,2]  <- kmeansAIC(state.icu.0350)
state.assess.icu[8,2]  <- kmeansAIC(state.icu.0400)
state.assess.icu[9,2]  <- kmeansAIC(state.icu.0450)
```

```
state.assess.icu[10,2] <- kmeansAIC(state.icu.0500)
state.assess.icu[11,2] <- kmeansAIC(state.icu.0550)
state.assess.icu[12,2] <- kmeansAIC(state.icu.0600)
state.assess.icu[13,2] <- kmeansAIC(state.icu.0650)
state.assess.icu[14,2] <- kmeansAIC(state.icu.0700)
state.assess.icu[15,2] <- kmeansAIC(state.icu.0750)
state.assess.icu[16,2] <- kmeansAIC(state.icu.0800)
state.assess.icu[17,2] <- kmeansAIC(state.icu.0850)
state.assess.icu[18,2] <- kmeansAIC(state.icu.0900)
state.assess.icu[19,2] <- kmeansAIC(state.icu.0950)
state.assess.icu[20,2] <- kmeansAIC(state.icu.1000)
state.assess.icu[21,2] <- kmeansAIC(state.icu.1050)
state.assess.icu[22,2] <- kmeansAIC(state.icu.1100)
state.assess.icu[23,2] <- kmeansAIC(state.icu.1150)
state.assess.icu[24,2] <- kmeansAIC(state.icu.1200)
state.assess.icu[25,2] <- kmeansAIC(state.icu.1250)
state.assess.icu[26,2] <- kmeansAIC(state.icu.1300)
state.assess.icu[27,2] <- kmeansAIC(state.icu.1350)
state.assess.icu[28,2] <- kmeansAIC(state.icu.1400)
state.assess.icu[29,2] <- kmeansAIC(state.icu.1450)
state.assess.icu[30,2] <- kmeansAIC(state.icu.1500)
state.assess.icu[31,2] <- kmeansAIC(state.icu.1550)
state.assess.icu[32,2] <- kmeansAIC(state.icu.1600)
state.assess.icu[33,2] <- kmeansAIC(state.icu.1650)
state.assess.icu[34,2] <- kmeansAIC(state.icu.1700)
state.assess.icu[35,2] <- kmeansAIC(state.icu.1750)
state.assess.icu[36,2] <- kmeansAIC(state.icu.1800)
state.assess.icu[37,2] <- kmeansAIC(state.icu.1850)
state.assess.icu[38,2] <- kmeansAIC(state.icu.1900)
state.assess.icu[39,2] <- kmeansAIC(state.icu.1950)
state.assess.icu[40,2] <- kmeansAIC(state.icu.2000)

#BIC
state.assess.icu[1,3]  <- kmeansBIC(state.icu.0050)
state.assess.icu[2,3]  <- kmeansBIC(state.icu.0100)
state.assess.icu[3,3]  <- kmeansBIC(state.icu.0150)
state.assess.icu[4,3]  <- kmeansBIC(state.icu.0200)
state.assess.icu[5,3]  <- kmeansBIC(state.icu.0250)
state.assess.icu[6,3]  <- kmeansBIC(state.icu.0300)
state.assess.icu[7,3]  <- kmeansBIC(state.icu.0350)
state.assess.icu[8,3]  <- kmeansBIC(state.icu.0400)
state.assess.icu[9,3]  <- kmeansBIC(state.icu.0450)
state.assess.icu[10,3] <- kmeansBIC(state.icu.0500)
state.assess.icu[11,3] <- kmeansBIC(state.icu.0550)
state.assess.icu[12,3] <- kmeansBIC(state.icu.0600)
state.assess.icu[13,3] <- kmeansBIC(state.icu.0650)
state.assess.icu[14,3] <- kmeansBIC(state.icu.0700)
state.assess.icu[15,3] <- kmeansBIC(state.icu.0750)
state.assess.icu[16,3] <- kmeansBIC(state.icu.0800)
state.assess.icu[17,3] <- kmeansBIC(state.icu.0850)
state.assess.icu[18,3] <- kmeansBIC(state.icu.0900)
state.assess.icu[19,3] <- kmeansBIC(state.icu.0950)
state.assess.icu[20,3] <- kmeansBIC(state.icu.1000)
state.assess.icu[21,3] <- kmeansBIC(state.icu.1050)
state.assess.icu[22,3] <- kmeansBIC(state.icu.1100)
state.assess.icu[23,3] <- kmeansBIC(state.icu.1150)
state.assess.icu[24,3] <- kmeansBIC(state.icu.1200)
state.assess.icu[25,3] <- kmeansBIC(state.icu.1250)
state.assess.icu[26,3] <- kmeansBIC(state.icu.1300)
state.assess.icu[27,3] <- kmeansBIC(state.icu.1350)
state.assess.icu[28,3] <- kmeansBIC(state.icu.1400)
state.assess.icu[29,3] <- kmeansBIC(state.icu.1450)
state.assess.icu[30,3] <- kmeansBIC(state.icu.1500)
state.assess.icu[31,3] <- kmeansBIC(state.icu.1550)
state.assess.icu[32,3] <- kmeansBIC(state.icu.1600)
state.assess.icu[33,3] <- kmeansBIC(state.icu.1650)
state.assess.icu[34,3] <- kmeansBIC(state.icu.1700)
state.assess.icu[35,3] <- kmeansBIC(state.icu.1750)
state.assess.icu[36,3] <- kmeansBIC(state.icu.1800)
state.assess.icu[37,3] <- kmeansBIC(state.icu.1850)
state.assess.icu[38,3] <- kmeansBIC(state.icu.1900)
```

```
state.assess.icu[39,3] <- kmeansBIC(state.icu.1950)
state.assess.icu[40,3] <- kmeansBIC(state.icu.2000)

#Within State SSE
state.assess.icu[1,4]  <- state.icu.0050$tot.withinss
state.assess.icu[2,4]  <- state.icu.0100$tot.withinss
state.assess.icu[3,4]  <- state.icu.0150$tot.withinss
state.assess.icu[4,4]  <- state.icu.0200$tot.withinss
state.assess.icu[5,4]  <- state.icu.0250$tot.withinss
state.assess.icu[6,4]  <- state.icu.0300$tot.withinss
state.assess.icu[7,4]  <- state.icu.0350$tot.withinss
state.assess.icu[8,4]  <- state.icu.0400$tot.withinss
state.assess.icu[9,4]  <- state.icu.0450$tot.withinss
state.assess.icu[10,4] <- state.icu.0500$tot.withinss
state.assess.icu[11,4] <- state.icu.0550$tot.withinss
state.assess.icu[12,4] <- state.icu.0600$tot.withinss
state.assess.icu[13,4] <- state.icu.0650$tot.withinss
state.assess.icu[14,4] <- state.icu.0700$tot.withinss
state.assess.icu[15,4] <- state.icu.0750$tot.withinss
state.assess.icu[16,4] <- state.icu.0800$tot.withinss
state.assess.icu[17,4] <- state.icu.0850$tot.withinss
state.assess.icu[18,4] <- state.icu.0900$tot.withinss
state.assess.icu[19,4] <- state.icu.0950$tot.withinss
state.assess.icu[20,4] <- state.icu.1000$tot.withinss
state.assess.icu[21,4] <- state.icu.1050$tot.withinss
state.assess.icu[22,4] <- state.icu.1100$tot.withinss
state.assess.icu[23,4] <- state.icu.1150$tot.withinss
state.assess.icu[24,4] <- state.icu.1200$tot.withinss
state.assess.icu[25,4] <- state.icu.1250$tot.withinss
state.assess.icu[26,4] <- state.icu.1300$tot.withinss
state.assess.icu[27,4] <- state.icu.1350$tot.withinss
state.assess.icu[28,4] <- state.icu.1400$tot.withinss
state.assess.icu[29,4] <- state.icu.1450$tot.withinss
state.assess.icu[30,4] <- state.icu.1500$tot.withinss
state.assess.icu[31,4] <- state.icu.1550$tot.withinss
state.assess.icu[32,4] <- state.icu.1600$tot.withinss
state.assess.icu[33,4] <- state.icu.1650$tot.withinss
state.assess.icu[34,4] <- state.icu.1700$tot.withinss
state.assess.icu[35,4] <- state.icu.1750$tot.withinss
state.assess.icu[36,4] <- state.icu.1800$tot.withinss
state.assess.icu[37,4] <- state.icu.1850$tot.withinss
state.assess.icu[38,4] <- state.icu.1900$tot.withinss
state.assess.icu[39,4] <- state.icu.1950$tot.withinss
state.assess.icu[40,4] <- state.icu.2000$tot.withinss

#Proportion of Variance Explained
state.assess.icu[1,5]  <- state.icu.0050$betweenss / state.icu.0050$totss
state.assess.icu[2,5]  <- state.icu.0100$betweenss / state.icu.0100$totss
state.assess.icu[3,5]  <- state.icu.0150$betweenss / state.icu.0150$totss
state.assess.icu[4,5]  <- state.icu.0200$betweenss / state.icu.0200$totss
state.assess.icu[5,5]  <- state.icu.0250$betweenss / state.icu.0250$totss
state.assess.icu[6,5]  <- state.icu.0300$betweenss / state.icu.0300$totss
state.assess.icu[7,5]  <- state.icu.0350$betweenss / state.icu.0350$totss
state.assess.icu[8,5]  <- state.icu.0400$betweenss / state.icu.0400$totss
state.assess.icu[9,5]  <- state.icu.0450$betweenss / state.icu.0450$totss
state.assess.icu[10,5] <- state.icu.0500$betweenss / state.icu.0500$totss
state.assess.icu[11,5] <- state.icu.0550$betweenss / state.icu.0550$totss
state.assess.icu[12,5] <- state.icu.0600$betweenss / state.icu.0600$totss
state.assess.icu[13,5] <- state.icu.0650$betweenss / state.icu.0650$totss
state.assess.icu[14,5] <- state.icu.0700$betweenss / state.icu.0700$totss
state.assess.icu[15,5] <- state.icu.0750$betweenss / state.icu.0750$totss
state.assess.icu[16,5] <- state.icu.0800$betweenss / state.icu.0800$totss
state.assess.icu[17,5] <- state.icu.0850$betweenss / state.icu.0850$totss
state.assess.icu[18,5] <- state.icu.0900$betweenss / state.icu.0900$totss
state.assess.icu[19,5] <- state.icu.0950$betweenss / state.icu.0950$totss
state.assess.icu[20,5] <- state.icu.1000$betweenss / state.icu.1000$totss
state.assess.icu[21,5] <- state.icu.1050$betweenss / state.icu.1050$totss
state.assess.icu[22,5] <- state.icu.1100$betweenss / state.icu.1100$totss
state.assess.icu[23,5] <- state.icu.1150$betweenss / state.icu.1150$totss
state.assess.icu[24,5] <- state.icu.1200$betweenss / state.icu.1200$totss
state.assess.icu[25,5] <- state.icu.1250$betweenss / state.icu.1250$totss
```

```
state.assess.icu[26,5] <- state.icu.1300$betweenss / state.icu.1300$totss
state.assess.icu[27,5] <- state.icu.1350$betweenss / state.icu.1350$totss
state.assess.icu[28,5] <- state.icu.1400$betweenss / state.icu.1400$totss
state.assess.icu[29,5] <- state.icu.1450$betweenss / state.icu.1450$totss
state.assess.icu[30,5] <- state.icu.1500$betweenss / state.icu.1500$totss
state.assess.icu[31,5] <- state.icu.1550$betweenss / state.icu.1550$totss
state.assess.icu[32,5] <- state.icu.1600$betweenss / state.icu.1600$totss
state.assess.icu[33,5] <- state.icu.1650$betweenss / state.icu.1650$totss
state.assess.icu[34,5] <- state.icu.1700$betweenss / state.icu.1700$totss
state.assess.icu[35,5] <- state.icu.1750$betweenss / state.icu.1750$totss
state.assess.icu[36,5] <- state.icu.1800$betweenss / state.icu.1800$totss
state.assess.icu[37,5] <- state.icu.1850$betweenss / state.icu.1850$totss
state.assess.icu[38,5] <- state.icu.1900$betweenss / state.icu.1900$totss
state.assess.icu[39,5] <- state.icu.1950$betweenss / state.icu.1950$totss
state.assess.icu[40,5] <- state.icu.2000$betweenss / state.icu.2000$totss

#Total SS
state.assess.icu[1,6]  <- state.icu.0050$totss
state.assess.icu[2,6]  <- state.icu.0100$totss
state.assess.icu[3,6]  <- state.icu.0150$totss
state.assess.icu[4,6]  <- state.icu.0200$totss
state.assess.icu[5,6]  <- state.icu.0250$totss
state.assess.icu[6,6]  <- state.icu.0300$totss
state.assess.icu[7,6]  <- state.icu.0350$totss
state.assess.icu[8,6]  <- state.icu.0400$totss
state.assess.icu[9,6]  <- state.icu.0450$totss
state.assess.icu[10,6] <- state.icu.0500$totss
state.assess.icu[11,6] <- state.icu.0550$totss
state.assess.icu[12,6] <- state.icu.0600$totss
state.assess.icu[13,6] <- state.icu.0650$totss
state.assess.icu[14,6] <- state.icu.0700$totss
state.assess.icu[15,6] <- state.icu.0750$totss
state.assess.icu[16,6] <- state.icu.0800$totss
state.assess.icu[17,6] <- state.icu.0850$totss
state.assess.icu[18,6] <- state.icu.0900$totss
state.assess.icu[19,6] <- state.icu.0950$totss
state.assess.icu[20,6] <- state.icu.1000$totss
state.assess.icu[21,6] <- state.icu.1050$totss
state.assess.icu[22,6] <- state.icu.1100$totss
state.assess.icu[23,6] <- state.icu.1150$totss
state.assess.icu[24,6] <- state.icu.1200$totss
state.assess.icu[25,6] <- state.icu.1250$totss
state.assess.icu[26,6] <- state.icu.1300$totss
state.assess.icu[27,6] <- state.icu.1350$totss
state.assess.icu[28,6] <- state.icu.1400$totss
state.assess.icu[29,6] <- state.icu.1450$totss
state.assess.icu[30,6] <- state.icu.1500$totss
state.assess.icu[31,6] <- state.icu.1550$totss
state.assess.icu[32,6] <- state.icu.1600$totss
state.assess.icu[33,6] <- state.icu.1650$totss
state.assess.icu[34,6] <- state.icu.1700$totss
state.assess.icu[35,6] <- state.icu.1750$totss
state.assess.icu[36,6] <- state.icu.1800$totss
state.assess.icu[37,6] <- state.icu.1850$totss
state.assess.icu[38,6] <- state.icu.1900$totss
state.assess.icu[39,6] <- state.icu.1950$totss
state.assess.icu[40,6] <- state.icu.2000$totss

#Between SS
state.assess.icu[1,7]  <- state.icu.0050$betweenss
state.assess.icu[2,7]  <- state.icu.0100$betweenss
state.assess.icu[3,7]  <- state.icu.0150$betweenss
state.assess.icu[4,7]  <- state.icu.0200$betweenss
state.assess.icu[5,7]  <- state.icu.0250$betweenss
state.assess.icu[6,7]  <- state.icu.0300$betweenss
state.assess.icu[7,7]  <- state.icu.0350$betweenss
state.assess.icu[8,7]  <- state.icu.0400$betweenss
state.assess.icu[9,7]  <- state.icu.0450$betweenss
state.assess.icu[10,7] <- state.icu.0500$betweenss
state.assess.icu[11,7] <- state.icu.0550$betweenss
state.assess.icu[12,7] <- state.icu.0600$betweenss
```

```
state.assess.icu[13,7] <- state.icu.0650$betweenss
state.assess.icu[14,7] <- state.icu.0700$betweenss
state.assess.icu[15,7] <- state.icu.0750$betweenss
state.assess.icu[16,7] <- state.icu.0800$betweenss
state.assess.icu[17,7] <- state.icu.0850$betweenss
state.assess.icu[18,7] <- state.icu.0900$betweenss
state.assess.icu[19,7] <- state.icu.0950$betweenss
state.assess.icu[20,7] <- state.icu.1000$betweenss
state.assess.icu[21,7] <- state.icu.1050$betweenss
state.assess.icu[22,7] <- state.icu.1100$betweenss
state.assess.icu[23,7] <- state.icu.1150$betweenss
state.assess.icu[24,7] <- state.icu.1200$betweenss
state.assess.icu[25,7] <- state.icu.1250$betweenss
state.assess.icu[26,7] <- state.icu.1300$betweenss
state.assess.icu[27,7] <- state.icu.1350$betweenss
state.assess.icu[28,7] <- state.icu.1400$betweenss
state.assess.icu[29,7] <- state.icu.1450$betweenss
state.assess.icu[30,7] <- state.icu.1500$betweenss
state.assess.icu[31,7] <- state.icu.1550$betweenss
state.assess.icu[32,7] <- state.icu.1600$betweenss
state.assess.icu[33,7] <- state.icu.1650$betweenss
state.assess.icu[34,7] <- state.icu.1700$betweenss
state.assess.icu[35,7] <- state.icu.1750$betweenss
state.assess.icu[36,7] <- state.icu.1800$betweenss
state.assess.icu[37,7] <- state.icu.1850$betweenss
state.assess.icu[38,7] <- state.icu.1900$betweenss
state.assess.icu[39,7] <- state.icu.1950$betweenss
state.assess.icu[40,7] <- state.icu.2000$betweenss

#Save Output
write.dta(state.assess.icu,"C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis
AI\\Analysis\\Sepsis AI - ICU Cohort - State Assessment - V1.dta")
```

Q-Learning was conducted using the ReinforcementLearning() package in R (4.0.3). Note that some code was adapted from Ruishen Lyu, 2020. (Lyu, 2020):

```
# Sepsis AI - Reinforcement Learning with Q Learning
#Load Libraries
library(haven)
library(writexl)
library(mice)
library(ReinforcementLearning)
library(Rfast)
library(data.table)
library(gplots)
library(ggplot2)
library(ggpubr)
library(tidyr)
library(foreign)
library(psych)
library(DescTools)

#Load Data
setwd("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis")
data.icu <- read_dta("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI -
ICU Cohort - Imputed - V1.dta")

#Transform and Normalize as in K means
#Log-Transform Features that are far from normal distribution
data.icu.ln             <- data.icu
data.icu.ln$alt         <- log(data.icu$alt)
data.icu.ln$ast         <- log(data.icu$ast)
data.icu.ln$bili        <- log(data.icu$bili)
data.icu.ln$bun         <- log(data.icu$bun)
data.icu.ln$creat       <- log(data.icu$creat)
data.icu.ln$dbp         <- log(data.icu$dbp)
data.icu.ln$gluc        <- log(data.icu$gluc)
data.icu.ln$inr         <- log(data.icu$inr)
data.icu.ln$lactate     <- log(data.icu$lactate)
data.icu.ln$map         <- log(data.icu$map)
data.icu.ln$paco2       <- log(data.icu$paco2)
data.icu.ln$pao2        <- log(data.icu$pao2)
data.icu.ln$sbp         <- log(data.icu$sbp)
data.icu.ln$pf_ratio    <- log(data.icu$pf_ratio)
data.icu.ln$plt         <- log(data.icu$plt)
data.icu.ln$shock_index <- log(data.icu$shock_index)
data.icu.ln$wbc         <- log(data.icu$wbc)
data.icu.ln$weight      <- log(data.icu$weight)
data.icu.ln$o2_sat      <- log(101-data.icu$o2_sat)

#Z-Transform Data (Mean=0, SD=1)
data.icu.z <- data.icu.ln
data.icu.z[c(4:19,21:26,28:41)] <- scale(data.icu.ln[c(4:19,21:26,28:41)],center=T,scale=T)
#Scale continuous variables to

#Create data frame for assessing states
data <- data.icu.z
ncol(data)
nrow(data)

#Model Features for Determining States
colnames(data[,c(4:42)])

# Number of Each Action
prop.table(table(data$action))

#Create String Variable for Treatments
data$treatment <-ifelse(data$action==1,"None",
                 ifelse(data$action==2,"0 Fluid, 0.01-0.09 NE",
                 ifelse(data$action==3,"0 Fluid, 0.09-0.20 NE",
                 ifelse(data$action==4,"0 Fluid, 0.20-0.50 NE",
```

```
                  ifelse(data$action==5,"0 Fluid, >0.50 NE",
                  ifelse(data$action==6,"1-250mL Fluid, 0 VP",
                  ifelse(data$action==7,"1-250mL Fluid, 0.01-0.09 NE",
                  ifelse(data$action==8,"1-250mL Fluid, 0.09-0.20 NE",
                  ifelse(data$action==9,"1-250mL Fluid, 0.20-0.50 NE",
                  ifelse(data$action==10,"1-250mL Fluid, >0.50 NE",
                  ifelse(data$action==11,"250-400mL Fluid, 0 VP",
                  ifelse(data$action==12,"250-400mL Fluid, 0.01-0.09 NE",
                  ifelse(data$action==13,"250-400mL Fluid, 0.09-0.20 NE",
                  ifelse(data$action==14,"250-400mL Fluid, 0.20-0.50 NE",
                  ifelse(data$action==15,"250-400mL Fluid, >0.50 NE",
                  ifelse(data$action==16,"400-700mL Fluid, 0 VP",
                  ifelse(data$action==17,"400-700mL Fluid, 0.01-0.09 NE",
                  ifelse(data$action==18,"400-700mL Fluid, 0.09-0.20 NE",
                  ifelse(data$action==19,"400-700mL Fluid, 0.20-0.50 NE",
                  ifelse(data$action==20,"400-700mL Fluid, >0.50 NE",
                  ifelse(data$action==21,">700mL Fluid, 0 VP",
                  ifelse(data$action==22,">700mL Fluid, 0.01-0.09 NE",
                  ifelse(data$action==23,">700mL Fluid, 0.09-0.20 NE",
                  ifelse(data$action==24,">700mL Fluid, 0.20-0.50 NE",
                  ifelse(data$action==25,">700mL Fluid, >0.50 NE",""))))))))))))))))))))))))))
data$treatment<-as.factor(data$treatment)

#Assess Outcome
#90-Day Mortality
unique <-data[data$interval == 7,]
sum(unique$dead_90)/nrow(unique) # 90-day mortality of 30.3%

#KMeans Clustering
#Load Previously Optimized Clusters
load(file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI - ICU Cohort -
750 States - V1.RData")
clust <- state.icu.0750$centers

# Reward and survival ranks
clusterdata         <- data
clusterdata$cluster <- state.icu.0750$cluster
clusterdata$outcome <- ifelse(clusterdata$dead_90==1,0,1) #Outcome == 1 if survived; 0 if died

reward              <- as.data.frame(matrix(NA,nrow=750,ncol=1))
reward$V1           <- c(1:750)
colnames(reward)[1] <- "cluster"
for (i in 1:750) {
  cluster               <- clusterdata[clusterdata$cluster==i,]
  cluster               <- cluster[!duplicated(cluster$hosp_id),]
  numerator             <- sum(cluster$outcome)
  denomenator           <- nrow(cluster)
  reward$probability[i] <- 1-numerator/denomenator #90-day mortality
  reward$size[i]        <- denomenator
}

reward$rank  <- rank(1-reward$probability)
reward       <- reward[order(reward$rank),]
reward$rank1 <- 1:750
write.csv(reward,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\ICU Cohort -
Reward.csv")

#Table 1 for cluster with ranks
#Merge state assignments to z-transformed data
data2         <- data
data2$cluster <- state.icu.0750$cluster
matching      <- match(data2$cluster,reward$cluster)
data2$rank    <- reward$rank1[matching]

#Merge in rewards
matching1          <- match(clusterdata$cluster,reward$cluster)
clusterdata$rank   <- reward$rank1[matching1]
clusterdata$reward <- reward$probability[matching1]
clusterdata1       <- clusterdata
data$rank          <- clusterdata1$rank
data$cluster       <- clusterdata1$cluster
```

```
#mean of the medical vitals
featuremean <- as.data.frame(matrix(NA,nrow=750,ncol=1))

#Create Summary Statistics by Cluster for Model Features
for (i in 1:750) {
  subset                    <- data[data$cluster==i,]
  featuremean$sofa_total[i]  <- mean(subset$sofa_total,na.rm=TRUE)
  featuremean$sirs_total[i]  <- mean(subset$sirs_total,na.rm=TRUE)
  featuremean$age[i]         <- mean(subset$age,na.rm=TRUE)
  featuremean$alb[i]         <- mean(subset$alb,na.rm=TRUE)
  featuremean$alt[i]         <- mean(subset$alt,na.rm=TRUE)
  featuremean$ast[i]         <- mean(subset$ast,na.rm=TRUE)
  featuremean$base_excess[i] <- mean(subset$base_excess,na.rm=TRUE)
  featuremean$bicarb[i]      <- mean(subset$bicarb,na.rm=TRUE)
  featuremean$bili[i]        <- mean(subset$bili,na.rm=TRUE)
  featuremean$bun[i]         <- mean(subset$bun,na.rm=TRUE)
  featuremean$cl[i]          <- mean(subset$cl,na.rm=TRUE)
  featuremean$creat[i]       <- mean(subset$creat,na.rm=TRUE)
  featuremean$dbp[i]         <- mean(subset$dbp,na.rm=TRUE)
  featuremean$elix[i]        <- mean(subset$elix,na.rm=TRUE)
  featuremean$fio2[i]        <- mean(subset$fio2,na.rm=TRUE)
  featuremean$gcs[i]         <- mean(subset$gcs,na.rm=TRUE)
  featuremean$gender[i]      <- mean(subset$gender,na.rm=TRUE)
  featuremean$gluc[i]        <- mean(subset$gluc,na.rm=TRUE)
  featuremean$hr[i]          <- mean(subset$hr,na.rm=TRUE)
  featuremean$hgb[i]         <- mean(subset$hgb,na.rm=TRUE)
  featuremean$inr[i]         <- mean(subset$inr,na.rm=TRUE)
  featuremean$lactate[i]     <- mean(subset$lactate,na.rm=TRUE)
  featuremean$map[i]         <- mean(subset$map,na.rm=TRUE)
  featuremean$mechvent[i]    <- mean(subset$mechvent,na.rm=TRUE)
  featuremean$paco2[i]       <- mean(subset$paco2,na.rm=TRUE)
  featuremean$pao2[i]        <- mean(subset$pao2,na.rm=TRUE)
  featuremean$pf_ratio[i]    <- mean(subset$pf_ratio,na.rm=TRUE)
  featuremean$ph[i]          <- mean(subset$ph,na.rm=TRUE)
  featuremean$plt[i]         <- mean(subset$plt,na.rm=TRUE)
  featuremean$k[i]           <- mean(subset$k,na.rm=TRUE)
  featuremean$rr[i]          <- mean(subset$rr,na.rm=TRUE)
  featuremean$shock_index[i] <- mean(subset$shock_index,na.rm=TRUE)
  featuremean$na[i]          <- mean(subset$na,na.rm=TRUE)
  featuremean$o2_sat[i]      <- mean(subset$o2_sat,na.rm=TRUE)
  featuremean$sbp[i]         <- mean(subset$sbp,na.rm=TRUE)
  featuremean$temp[i]        <- mean(subset$temp,na.rm=TRUE)
  featuremean$wbc[i]         <- mean(subset$wbc,na.rm=TRUE)
  featuremean$weight[i]      <- mean(subset$weight,na.rm=TRUE)
}

#Create a clean data fame of variable means (z-transformed)
featuremean           <- featuremean[,2:39]
colnames(featuremean) <- colnames(data)[c(4:41)]
featuremean$state     <- rownames(featuremean)
featuremean           <- format(featuremean,digits=3)
matching1            <- match(featuremean$state,reward$cluster)
featuremean$rank      <- reward$rank1[matching1]

#Save mean by cluster data
write.csv(featuremean,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\Mean
Standardized Value - ICU Cohort - By State - V1.csv")

#Turn features to numeric and create heatmap of feature means by cluster
#Clean column names for
colnames(featuremean) <- c('SOFA', 'SIRS',        'Age',     'Albumin', 'ALT',
                           'AST',  'Base Excess', 'Bicarb',  'Bili',    'BUN',
                           'Cl',   'Creatinine',  'DBP',     'Elix',    'FiO2',
                           'GCS',  'Gender',      'Glucose', 'HR',      'Hgb',
                           'INR',  'Lactate',     'MAP',     'MV',      'PaCO2',
                           'PaO2', 'PF Ratio',    'pH',      'Plt',     'Potassium',
                           'RR',   'Shock Index', 'Sodium',  'O2 Sat',  'SBP',
                           'Temp', 'WBC',         'Weight',  'State',   'Rank')

featuremean[,c(1:39)] <- lapply(featuremean[,c(1:38)],as.character)
```

```
featuremean[,c(1:39)] <- lapply(featuremean[,c(1:38)],as.numeric)

#Heatmap of Feature Means
pdf("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\Heatmap of Mean Stanrdized
Value - ICU Cohort - By State - V1.pdf")

df <- featuremean[c(1:750),c(1:38)]
df <- df[seq(1,750,10),] #NOTE: Heatmap looked too thin for each cluster, so Take Only Every 10th
Row

matching3    <- match(rownames(df),reward$cluster)
df$rank      <- reward$rank1[matching3]
rownames(df) <- df$rank
df           <- df[order(df$rank),]
df           <- as.matrix(df[,c(1:38)])
heatmap.2(df, scale='column',Rowv=FALSE,Colv=TRUE,dendrogram="none",trace="none",srtCol=52,
adjCol = c(0.8,0),col=bluered,  tracecol="#303030")
dev.off()

# Assess Action/Treatment for each state
trt<-as.data.frame(matrix(NA))
for (i in 1:750) {
  cluster         <- clusterdata[clusterdata$cluster==i,]
  trt[c(1:25),i]  <- as.data.frame((as.matrix(table(cluster$treatment))))$V1
  trt[26,i]       <- nrow(cluster)
  trt[c(27:51),i] <- trt[c(1:25),i]/nrow(cluster)
}

rownames(trt) <-
c(rownames(as.data.frame((as.matrix(table(cluster$treatment))))),"Size",paste0(rownames(as.data.f
rame((as.matrix(table(cluster$treatment))))),' Probability'))
trt<-round(trt,digits=3)
colnames(trt) <- c(1:750)
write.csv(trt,file = "C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\Treatment
Matrix Probability - ICU Cohort - Long - V1.csv")
trt           <- as.data.frame(t(trt))
trt$state     <- rownames(trt)

matching1          <- match(clusterdata$cluster,reward$cluster)
clusterdata$rank1  <- reward$rank1[matching1]
clusterdata1$rank1 <- reward$rank1[matching1]

#Action/treatment for each rank
trt2 <- as.data.frame(matrix(NA))
for (i in 1:750) {
  cluster          <- clusterdata[clusterdata$rank1==i,]
  trt2[c(1:25),i]  <- as.data.frame((as.matrix(table(cluster$treatment))))$V1
  trt2[26,i]       <- nrow(cluster)
  trt2[c(27:51),i] <- trt2[c(1:25),i]/nrow(cluster)
}

rownames(trt2)  <-
c(rownames(as.data.frame((as.matrix(table(cluster$treatment))))),"Size",paste0(rownames(as.data.f
rame((as.matrix(table(cluster$treatment))))),'Probability'))
trt2            <- round(trt2,digits=2)
colnames(trt2)  <- reward$rank1
for (i in 1:51) {
  trt2$total[i] <- sum(trt2[i,1:750])
}

write.csv(trt2,file = "C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\Treatment
Matrix Probability - ICU Cohort - Wide - V1.csv")

trt2 <- as.data.frame(t(trt2))
trt2 <- trt2[-751,]

#Heatmap of Treatments
pdf("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\Heatmap of Treatment - ICU
Cohort - V1.pdf")
df <- as.matrix(trt2[,c(1:25)])
df <- df[seq(1,750,10),] #Take Only Every 10th Row
```

```
heatmap.2(df, Rowv=FALSE,Colv=TRUE,dendrogram="none",trace="none",srtCol=360, adjCol = c(0.5,1) )
dev.off()

#Prep Data for QLearn
#Make Variable for "Next Cluster" -- Note that we will delete last records
data3                  <- clusterdata1[c(1:3,44:51)]
data3$nextcluster      <- data3$cluster
for (i in 1:(nrow(data3)-1)) {
  data3$nextcluster[i] <- data3$cluster[i+1]
}

#Assign -100 penalty for deaths and delete dead patients' last record since no next state
uni            <- unique(data3[as.character(data3$dead_90)==1,]$hosp_id)
data3$sequence <- 1:nrow(data3)
for (i in uni) {
  a            <- data3[data3$hosp_id==i&
data3$interval==max(data3[data3$hosp_id==i,]$interval),]$sequence
  data3        <- data3[data3$sequence!=a,]
  data3$reward <- ifelse(data3$sequence==a-1,-100,data3$reward)
}

#Assign +100 points for survival and delete alive patients last record since no next state
uni2           <- unique(data3[as.character(data3$dead_90)==0,]$hosp_id)
data3$sequence <- 1:nrow(data3)
for (i in uni2) {
  a            <- data3[data3$hosp_id==i&
data3$interval==max(data3[data3$hosp_id==i,]$interval),]$sequence
  data3        <- data3[data3$sequence!=a,]
  data3$reward <- ifelse(data3$sequence==a-1,100,data3$reward)
}

#Assign +100 reward for survivors, -100 for 90-day deaths (replace non 100/-100 values with 0)
data3$reward <- ifelse(data3$reward==100,100,ifelse(data3$reward==-100,-100,0))

#Create frame with state-action mortality for all 4-hr points
data.mort              <- data3[,c(1,4:7)]
table.temp             <- data.frame(table(data.mort$hosp_id))
colnames(table.temp)   <- c("hosp_id","n.states")
data.mort              <- merge(data.mort,table.temp,by =  "hosp_id")
data.mort$weight       <- 1/data.mort$n.states
data.mort$wt.mort      <- data.mort$weight*data.mort$dead_90
data.mort$state.action <- paste(data.mort$cluster, data.mort$action, sep = "_")
state.action           <- data.mort[,c(9,5,2)]

mort.denom          <- aggregate(data.mort$weight, by=list(Category=data.mort$state.action),
FUN=sum)
colnames(mort.denom) <- c("state.action","sum.wt")
mort.num             <- aggregate(data.mort$wt.mort, by=list(Category=data.mort$state.action),
FUN=sum)
colnames(mort.num)   <- c("state.action","sum.wt.mort")
mort.prob.1          <- merge(mort.num,mort.denom,by="state.action")
mort.prob.1$p.mort   <- mort.prob.1$sum.wt.mort/mort.prob.1$sum.wt
mort.prob.1          <- merge(mort.prob.1,state.action,by="state.action")
mort.prob.1          <- unique(mort.prob.1)

#Calculate predicted mortality for each STATE to fill in for missing state-action pairs
mort.denom.cluster <- aggregate(data.mort$weight, by=list(Category=data.mort$cluster), FUN=sum)
colnames(mort.denom.cluster) <- c("cluster","sum.wt")
mort.num.cluster <- aggregate(data.mort$wt.mort, by=list(Category=data.mort$cluster), FUN=sum)
colnames(mort.num.cluster)        <- c("cluster","sum.wt.mort")
mort.prob.cluster                 <- merge(mort.num.cluster,mort.denom.cluster,by="cluster")
mort.prob.cluster$p.mort.state    <- mort.prob.cluster$sum.wt.mort/mort.prob.cluster$sum.wt
state.action.cluster              <- data.frame(cluster=rep(seq_len(750), each=25),
                                                action=rep(1:25,750))
state.action.cluster$state.action <- paste(state.action.cluster$cluster,
state.action.cluster$action, sep = "_")
mort.prob.cluster                 <- merge(mort.prob.cluster,state.action.cluster,by="cluster")

#Merge State-Action Mortality with State Mortality to get final P(mort) for State-Action Pairs
st.act           <- mort.prob.1[,c(1,4)]
act              <- mort.prob.cluster[,c(6,1,5,4)]
```

```
mort.prob        <- merge(act,st.act,by="state.action", all.x=TRUE)
mort.prob$p.mort <- ifelse(is.na(mort.prob$p.mort),mort.prob$p.mort.state,mort.prob$p.mort)
mort.merge       <- mort.prob[,c(1,5)]

#Split Data into 80% Training and 20% Testing, at Encounter level (hosp_id)
hosp.id    <- as.data.frame(unique(data3$hosp_id))
train.size <- floor(0.80 * nrow(hosp.id)) ## 80% of encounters
set.seed(12081023)
train.ind  <- sample(seq_len(nrow(hosp.id)), size = train.size)
train      <- data.frame(hosp_id=hosp.id[train.ind,])
test       <- data.frame(hosp_id=hosp.id[-train.ind,])

write.dta(train,"C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI - ICU
Cohort - Training Hosp ID - V1.dta")
write.dta(test,"C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI - ICU
Cohort - Testing Hosp ID - V1.dta")

#Merge to create training and testing frames with full data for Q Learning
data3.train <- merge(data3, train, by = "hosp_id")
data3.test  <- merge(data3, test, by = "hosp_id")

#Create a data frame with only the Q Learning Parameters that is ready for Reinforcement Learning
(Training Data)
data4             <- data3.train
data4             <- data4[,c(7,12,6,10)]
data4$cluster     <- as.character(data4$cluster)
data4$nextcluster <- as.character(data4$nextcluster)
data4$treatment   <- as.character(data4$treatment)
data4$reward      <- as.numeric(data4$reward)

# Define reinforcement learning parameters
epsilon <- 0.1
control <- list(alpha = 0.1, gamma = 0.99, epsilon = 0.1)

# Perform reinforcement learning
model.icu <- ReinforcementLearning(data4, s = "cluster", a = "treatment", r = "reward",
                                   s_new = "nextcluster", iter = 100, control =
control,verbose=TRUE)

View(model.icu$Q)
# Calculate optimal policy
pol <- computePolicy(model.icu)

#Make table of Q-Value by State (column for each tx)
table2               <- as.data.frame(cbind(model.icu$Q,model.icu$Policy))
table2$state         <- as.numeric(rownames(model.icu$Q))
rownames(table2)     <- rownames(model.icu$Q)
table2               <- table2[order(table2$state),]
colnames(table2)[26] <- 'AI'

#combine with treatment with each state - Merges Original Treatment Probabilities within Each
original State
trt2$state <- rownames(trt2)
matching2  <- match(table2$state,trt$state)
for (i in 1:25) {
  table2[,27+i] <- trt[,26+i][matching2]
}
colnames(table2)[28:52] <- colnames(trt)[27:51]
colnames(table2)[28:52] <- gsub('Probability','2',colnames(table2)[28:52])

#Rearrange tables to have same order for actions
table2 <- table2[,c("state","None","1-250mL Fluid, 0 VP","250-400mL Fluid, 0 VP","400-700mL
Fluid, 0 VP",">700mL Fluid, 0 VP","0 Fluid, 0.01-0.09 NE","1-250mL Fluid, 0.01-0.09 NE","250-
400mL Fluid, 0.01-0.09 NE","400-700mL Fluid, 0.01-0.09 NE",">700mL Fluid, 0.01-0.09 NE","0 Fluid,
0.09-0.20 NE","1-250mL Fluid, 0.09-0.20 NE","250-400mL Fluid, 0.09-0.20 NE","400-700mL Fluid,
0.09-0.20 NE",">700mL Fluid, 0.09-0.20 NE","0 Fluid, 0.20-0.50 NE","1-250mL Fluid, 0.20-0.50
NE","250-400mL Fluid, 0.20-0.50 NE","400-700mL Fluid, 0.20-0.50 NE",">700mL Fluid, 0.20-0.50
NE","0 Fluid, >0.50 NE","1-250mL Fluid, >0.50 NE","250-400mL Fluid, >0.50 NE","400-700mL Fluid,
>0.50 NE",">700mL Fluid, >0.50 NE","None 2","1-250mL Fluid, 0 VP 2","250-400mL Fluid, 0 VP
2","400-700mL Fluid, 0 VP 2",">700mL Fluid, 0 VP 2","0 Fluid, 0.01-0.09 NE 2","1-250mL Fluid,
0.01-0.09 NE 2","250-400mL Fluid, 0.01-0.09 NE 2","400-700mL Fluid, 0.01-0.09 NE 2",">700mL
```

```
Fluid, 0.01-0.09 NE 2","0 Fluid, 0.09-0.20 NE 2","1-250mL Fluid, 0.09-0.20 NE 2","250-400mL
Fluid, 0.09-0.20 NE 2","400-700mL Fluid, 0.09-0.20 NE 2",">700mL Fluid, 0.09-0.20 NE 2","0 Fluid,
0.20-0.50 NE 2","1-250mL Fluid, 0.20-0.50 NE 2","250-400mL Fluid, 0.20-0.50 NE 2","400-700mL
Fluid, 0.20-0.50 NE 2",">700mL Fluid, 0.20-0.50 NE 2","0 Fluid, >0.50 NE 2","1-250mL Fluid, >0.50
NE 2","250-400mL Fluid, >0.50 NE 2","400-700mL Fluid, >0.50 NE 2",">700mL Fluid, >0.50 NE
2","AI")]

#Add In Variable for Clinician Policy
treatmentlist  <- c("state","None","1-250mL Fluid, 0 VP","250-400mL Fluid, 0 VP","400-700mL
Fluid, 0 VP",">700mL Fluid, 0 VP","0 Fluid, 0.01-0.09 NE","1-250mL Fluid, 0.01-0.09 NE","250-
400mL Fluid, 0.01-0.09 NE","400-700mL Fluid, 0.01-0.09 NE",">700mL Fluid, 0.01-0.09 NE","0 Fluid,
0.09-0.20 NE","1-250mL Fluid, 0.09-0.20 NE","250-400mL Fluid, 0.09-0.20 NE","400-700mL Fluid,
0.09-0.20 NE",">700mL Fluid, 0.09-0.20 NE","0 Fluid, 0.20-0.50 NE","1-250mL Fluid, 0.20-0.50
NE","250-400mL Fluid, 0.20-0.50 NE","400-700mL Fluid, 0.20-0.50 NE",">700mL Fluid, 0.20-0.50
NE","0 Fluid, >0.50 NE","1-250mL Fluid, >0.50 NE","250-400mL Fluid, >0.50 NE","400-700mL Fluid,
>0.50 NE",">700mL Fluid, >0.50 NE")
treatmentlist1 <- c("None 2","1-250mL Fluid, 0 VP 2","250-400mL Fluid, 0 VP 2","400-700mL Fluid,
0 VP 2",">700mL Fluid, 0 VP 2","0 Fluid, 0.01-0.09 NE 2","1-250mL Fluid, 0.01-0.09 NE 2","250-
400mL Fluid, 0.01-0.09 NE 2","400-700mL Fluid, 0.01-0.09 NE 2",">700mL Fluid, 0.01-0.09 NE 2","0
Fluid, 0.09-0.20 NE 2","1-250mL Fluid, 0.09-0.20 NE 2","250-400mL Fluid, 0.09-0.20 NE 2","400-
700mL Fluid, 0.09-0.20 NE 2",">700mL Fluid, 0.09-0.20 NE 2","0 Fluid, 0.20-0.50 NE 2","1-250mL
Fluid, 0.20-0.50 NE 2","250-400mL Fluid, 0.20-0.50 NE 2","400-700mL Fluid, 0.20-0.50 NE
2",">700mL Fluid, 0.20-0.50 NE 2","0 Fluid, >0.50 NE 2","1-250mL Fluid, >0.50 NE 2","250-400mL
Fluid, >0.50 NE 2","400-700mL Fluid, >0.50 NE 2",">700mL Fluid, >0.50 NE 2")
table2$Clinician <- treatmentlist1[apply(table2[c(27:51)], 1, which.max)]
table2[,c(2:26)] <- lapply(table2[,c(2:26)],as.character)
table2[,c(2:26)] <- lapply(table2[,c(2:26)],as.numeric)
write.csv(table2,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\AI vs Clin
Policy - ICU Cohort - 25 Treatments.csv")

#Policy Comparison - Clinician vs. AI vs. Completely Random
#Clinician Policy
pi0           <- table2[,c(1,27:51)]
colnames(pi0) <- treatmentlist
write.csv(pi0,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU Cohort -
Clinician Policy.csv")

#AI Policy
pi1           <- pi0
colnames(pi1) <- treatmentlist
pi1$pol       <- table2$AI
for(i in 1:750) {
  for(j in 2:26) {
    pi1[i,j]  <- ifelse(colnames(pi1[j])==pi1[i,27],0.904,0.004)
  }
}
pi1           <- pi1[,1:26]
write.csv(pi1,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU Cohort -
AI Policy.csv")

#Random Policy
pi2 <- pi0
pi2[,2:26] <- (pi2[,2:26] < 0) + 1/25
pi2 <- data.frame(pi2)
colnames(pi2) <- treatmentlist
write.csv(pi1,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU Cohort -
Random Policy.csv")

#Heatmap - AI Policy Values - Among Everyone
df           <- as.data.frame(t(scale(t(table2[,c(2:26)]),center = TRUE,scale=TRUE)))
matching3    <- match(rownames(df),reward$cluster)
df$rank      <- reward$rank1[matching3]
rownames(df) <- df$rank
df           <- df[order(df$rank),]
df           <- as.matrix(df[,c(1:25)])
pdf("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU Cohort - AI Policy
Values Heatmap.pdf")
heatmap.2(df, Rowv=FALSE,Colv=TRUE,dendrogram="none",trace="none",srtCol=360, adjCol =
c(0.5,1),col=bluered,  tracecol="#303030")
dev.off()
```

```
#Heatmap - Clinician Actions - Among Everyone
df<-as.data.frame(t(scale(t(table2[,c(27:51)]),center = TRUE,scale=TRUE)))
matching3<-match(rownames(df),reward$cluster)
df$rank<-reward$rank1[matching3]
rownames(df)<-df$rank
df<-df[order(df$rank),]
df<-as.matrix(df[,c(1:25)])
pdf("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU Cohort - Clniician
Action Heatmap.pdf")
heatmap.2(df, Rowv=FALSE,Colv=TRUE,dendrogram="none",trace="none",srtCol=360, adjCol =
c(0.5,1),col=bluered,  tracecol="#303030")
dev.off()

#WIS Probability Weighing Values (Ratios) to Adjust Distribution
ratios0                 <- pi0 / pi0
ratios0[is.na(ratios0)] <- 0
ratios0[,1]             <- pi0[,1]
ratios1                 <- pi1 / pi0
ratios1[ratios1 == Inf] <- 0
ratios1[,1]             <- pi1[,1]
ratios2                 <- pi2 / pi0
ratios2[ratios2 == Inf] <- 0
ratios2[,1]             <- pi2[,1]

#Estimate WIS Policy Value and Predicted Mortality by Clinician vs. AI vs Random Policy
#Mortality by Clinician Policy - Testing Set
mort.clin.1              <- data3.test[,c(1,7,4)]
colnames(mort.clin.1)    <- c("hosp_id","state","action")
mort.clin.1$state.action <- paste(mort.clin.1$state, mort.clin.1$action, sep = "_")
mort.clin.1              <- mort.clin.1[,c(1,4,2,3)]
mort.clin.1              <- merge(mort.clin.1,mort.merge,by="state.action")

mort.clin           <- aggregate(mort.clin.1$p.mort, by=list(Category=mort.clin.1$hosp_id),
FUN=mean)
colnames(mort.clin) <- c("hosp_id","p.mort")
summary(mort.clin)  #This is the estimated Clinician Mortality

#Make a crosswalk of Treatment and Action Variables
tx.crosswalk <- data.frame(treatment=c("None","0 Fluid, 0.01-0.09 NE","0 Fluid, 0.09-0.20 NE",
"0 Fluid, 0.20-0.50 NE","0 Fluid, >0.50 NE","1-250mL Fluid, 0 VP","1-250mL Fluid, 0.01-0.09 NE",
"1-250mL Fluid, 0.09-0.20 NE","1-250mL Fluid, 0.20-0.50 NE","1-250mL Fluid, >0.50 NE","250-400mL
Fluid, 0 VP", "250-400mL Fluid, 0.01-0.09 NE","250-400mL Fluid, 0.09-0.20 NE", "250-400mL Fluid,
0.20-0.50 NE", "250-400mL Fluid, >0.50 NE","400-700mL Fluid, 0 VP","400-700mL Fluid, 0.01-0.09
NE","400-700mL Fluid, 0.09-0.20 NE","400-700mL Fluid, 0.20-0.50 NE","400-700mL Fluid, >0.50
NE",">700mL Fluid, 0 VP",">700mL Fluid, 0.01-0.09 NE",">700mL Fluid, 0.09-0.20 NE",">700mL Fluid,
0.20-0.50 NE",">700mL Fluid, >0.50 NE"),action=1:25)

#Mortality by AI Policy - Define Policy by State
ai.pol <- pi1
ai.pol$treatment <- table2$AI
ai.pol <- merge(ai.pol,tx.crosswalk,by="treatment")
ai.optimal <- ai.pol[,c(2,28)]
colnames(ai.optimal) <- c("state","ai.optimal")

#Create 1000x Bootstraps of Testing set, by Encounter, and store results
n.boot <- 1000
data.3.test.merge <- data3.test[,c(1,7,4)]
bootstrap <- data.frame(sim.num=1:n.boot,mean.mort.clin=rep(NA, n.boot),mean.mort.ai=rep(NA,
n.boot),mean.mort.random=rep(NA, n.boot),t.stat=rep(NA, n.boot)) # create a data frame to store
results in
unique.test      <- unique(data3.test$hosp_id)

for(i in 1:n.boot){
  unique.test.boot.1 <-
data.frame(hosp_id=unique.test[sample(x=1:length(unique.test),size=length(unique.test),replace=TR
UE)])
  unique.test.boot.1$rownum <- rownames(unique.test.boot.1)
  unique.test.boot.1 <- merge(unique.test.boot.1,mort.clin.1,by="hosp_id")
  unique.test.boot.2 <- unique.test.boot.1[,1:5]
  unique.test.boot.3 <- unique.test.boot.1[,1:5]
```

```
  #Clinician
  mort.boot.1                 <- aggregate(unique.test.boot.1$p.mort,
by=list(Category=unique.test.boot.1$rownum), FUN=mean)
  bootstrap$mean.mort.clin[i] <- mean(mort.boot.1$x)

  #AI
  unique.test.boot.2              <- merge(unique.test.boot.2,ai.optimal,by="state")
  unique.test.boot.2$ai.decision  <- runif(length(unique.test.boot.2$hosp_id))
  unique.test.boot.2$ai.random    <- sample(1:25,length(unique.test.boot.2$hosp_id),replace=TRUE)
  unique.test.boot.2$ai.action    <- ifelse(unique.test.boot.2$ai.decision > epsilon,
  #Choose AI Optimal Policy for P(1-epsilon) cases and Randomize for P(epsilon) cases
                                        unique.test.boot.2$ai.optimal,
                                        unique.test.boot.2$ai.random)
  unique.test.boot.2$state.action <- paste(unique.test.boot.2$state,
unique.test.boot.2$ai.action, sep = "_")
  unique.test.boot.2              <- merge(unique.test.boot.2,mort.merge,by="state.action")
  mort.boot.2                     <- aggregate(unique.test.boot.2$p.mort,
by=list(Category=unique.test.boot.2$rownum), FUN=mean)
  bootstrap$mean.mort.ai[i]       <- mean(mort.boot.2$x)

  #Random
  unique.test.boot.3$action       <- sample(1:25,length(unique.test.boot.2$hosp_id),replace=TRUE)
  unique.test.boot.3$state.action <- paste(unique.test.boot.3$state, unique.test.boot.3$action,
sep = "_")
  unique.test.boot.3              <- merge(unique.test.boot.3,mort.merge,by="state.action")
  mort.boot.3                     <- aggregate(unique.test.boot.3$p.mort,
by=list(Category=unique.test.boot.3$rownum), FUN=mean)
  bootstrap$mean.mort.random[i]   <- mean(mort.boot.3$x)
}

#Load Strata and Run Testing in Subsets of Test Set
data.strata <- read_dta("C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Data\\Sepsis AI -
Strata - V1.dta")

#Strata 1 of Age (18-39)
strata.match <- data.strata[,1:2]
strata.match <- strata.match[which(strata.match$age_ind==1),]

#Create 1000x Bootstraps of Testing set, by Encounter, and store results
data.3.test.merge <- data3.test[,c(1,7,4)]
bootstrap.age1    <- data.frame(sim.num=1:n.boot,mean.mort.clin.age1=rep(NA,
n.boot),mean.mort.ai.age1=rep(NA, n.boot),mean.mort.random.age1=rep(NA, n.boot)) # create a data
frame to store results in
unique.test       <- data.frame(unique(data3.test$hosp_id))
colnames(unique.test) <- c("hosp_id")
unique.test       <- merge(unique.test,strata.match,by="hosp_id")
unique.test       <- unique.test[,1]

for(i in 1:n.boot){
  unique.test.boot.1 <-
data.frame(hosp_id=unique.test[sample(x=1:length(unique.test),size=length(unique.test),replace=TR
UE)])
  unique.test.boot.1$rownum <- rownames(unique.test.boot.1)
  unique.test.boot.1 <- merge(unique.test.boot.1,mort.clin.1,by="hosp_id")
  unique.test.boot.2 <- unique.test.boot.1[,1:5]
  unique.test.boot.3 <- unique.test.boot.1[,1:5]

  #Clinician
  mort.boot.1                 <- aggregate(unique.test.boot.1$p.mort,
by=list(Category=unique.test.boot.1$rownum), FUN=mean)
  bootstrap.age1$mean.mort.clin.age1[i] <- mean(mort.boot.1$x)

  #AI
  unique.test.boot.2              <- merge(unique.test.boot.2,ai.optimal,by="state")
  unique.test.boot.2$ai.decision  <- runif(length(unique.test.boot.2$hosp_id))
  unique.test.boot.2$ai.random    <- sample(1:25,length(unique.test.boot.2$hosp_id),replace=TRUE)
  unique.test.boot.2$ai.action    <- ifelse(unique.test.boot.2$ai.decision > epsilon, #Choose AI
Optimal Policy for P(1-epsilon) cases and Randomize for P(epsilon) cases
                                        unique.test.boot.2$ai.optimal,
                                        unique.test.boot.2$ai.random)
```

```
  unique.test.boot.2$state.action <- paste(unique.test.boot.2$state,
unique.test.boot.2$ai.action, sep = "_")
  unique.test.boot.2              <- merge(unique.test.boot.2,mort.merge,by="state.action")
  mort.boot.2                     <- aggregate(unique.test.boot.2$p.mort,
by=list(Category=unique.test.boot.2$rownum), FUN=mean)
  bootstrap.age1$mean.mort.ai.age1[i]       <- mean(mort.boot.2$x)

  #Random
  unique.test.boot.3$action       <- sample(1:25,length(unique.test.boot.2$hosp_id),replace=TRUE)
  unique.test.boot.3$state.action <- paste(unique.test.boot.3$state, unique.test.boot.3$action,
sep = "_")
  unique.test.boot.3              <- merge(unique.test.boot.3,mort.merge,by="state.action")
  mort.boot.3                     <- aggregate(unique.test.boot.3$p.mort,
by=list(Category=unique.test.boot.3$rownum), FUN=mean)
  bootstrap.age1$mean.mort.random.age1[i]   <- mean(mort.boot.3$x)
}
(this code was duplicated for each strata)

write.csv(wis.table,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU
Cohort - WIS Summary.csv")

bootstrap$arr          <- bootstrap$mean.mort.clin          - bootstrap$mean.mort.ai
bootstrap.age1$arr     <- bootstrap.age1$mean.mort.clin     - bootstrap.age1$mean.mort.ai
bootstrap.age2$arr     <- bootstrap.age2$mean.mort.clin     - bootstrap.age2$mean.mort.ai
bootstrap.age3$arr     <- bootstrap.age3$mean.mort.clin     - bootstrap.age3$mean.mort.ai
bootstrap.age4$arr     <- bootstrap.age4$mean.mort.clin     - bootstrap.age4$mean.mort.ai
bootstrap.age1$arr     <- bootstrap.age1$mean.mort.clin     - bootstrap.age1$mean.mort.ai
bootstrap.gend0$arr    <- bootstrap.gend0$mean.mort.clin    - bootstrap.gend0$mean.mort.ai
bootstrap.gend1$arr    <- bootstrap.gend1$mean.mort.clin    - bootstrap.gend1$mean.mort.ai
bootstrap.sofa1$arr    <- bootstrap.sofa1$mean.mort.clin    - bootstrap.sofa1$mean.mort.ai
bootstrap.sofa2$arr    <- bootstrap.sofa2$mean.mort.clin    - bootstrap.sofa2$mean.mort.ai
bootstrap.sofa3$arr    <- bootstrap.sofa3$mean.mort.clin    - bootstrap.sofa3$mean.mort.ai
bootstrap.sofa4$arr    <- bootstrap.sofa4$mean.mort.clin    - bootstrap.sofa4$mean.mort.ai
bootstrap.prior0$arr   <- bootstrap.prior0$mean.mort.clin   - bootstrap.prior0$mean.mort.ai
bootstrap.prior1$arr   <- bootstrap.prior1$mean.mort.clin   - bootstrap.prior1$mean.mort.ai
bootstrap.surg0$arr    <- bootstrap.surg0$mean.mort.clin    - bootstrap.surg0$mean.mort.ai
bootstrap.surg1$arr    <- bootstrap.surg1$mean.mort.clin    - bootstrap.surg1$mean.mort.ai
bootstrap.hosp1$arr    <- bootstrap.hosp1$mean.mort.clin    - bootstrap.hosp1$mean.mort.ai
bootstrap.hosp2$arr    <- bootstrap.hosp2$mean.mort.clin    - bootstrap.hosp2$mean.mort.ai
bootstrap.hosp3$arr    <- bootstrap.hosp3$mean.mort.clin    - bootstrap.hosp3$mean.mort.ai
bootstrap.hosp4$arr    <- bootstrap.hosp4$mean.mort.clin    - bootstrap.hosp4$mean.mort.ai
bootstrap.hosp5$arr    <- bootstrap.hosp5$mean.mort.clin    - bootstrap.hosp5$mean.mort.ai
bootstrap.hosp6$arr    <- bootstrap.hosp6$mean.mort.clin    - bootstrap.hosp6$mean.mort.ai
bootstrap.hosp7$arr    <- bootstrap.hosp7$mean.mort.clin    - bootstrap.hosp7$mean.mort.ai
bootstrap.hosp8$arr    <- bootstrap.hosp8$mean.mort.clin    - bootstrap.hosp8$mean.mort.ai
bootstrap.hosp9$arr    <- bootstrap.hosp9$mean.mort.clin    - bootstrap.hosp9$mean.mort.ai
bootstrap.hosp10$arr   <- bootstrap.hosp10$mean.mort.clin   - bootstrap.hosp10$mean.mort.ai
bootstrap.hosp11$arr   <- bootstrap.hosp11$mean.mort.clin   - bootstrap.hosp11$mean.mort.ai
bootstrap.hosp12$arr   <- bootstrap.hosp12$mean.mort.clin   - bootstrap.hosp12$mean.mort.ai
bootstrap.hosp13$arr   <- bootstrap.hosp13$mean.mort.clin   - bootstrap.hosp13$mean.mort.ai
bootstrap.hosp14$arr   <- bootstrap.hosp14$mean.mort.clin   - bootstrap.hosp14$mean.mort.ai
bootstrap.teach.1$arr  <- bootstrap.teach.1$mean.mort.clin  - bootstrap.teach.1$mean.mort.ai
bootstrap.teach.2$arr  <- bootstrap.teach.2$mean.mort.clin  - bootstrap.teach.2$mean.mort.ai
bootstrap.urban.1$arr  <- bootstrap.urban.1$mean.mort.clin  - bootstrap.urban.1$mean.mort.ai
bootstrap.urban.2$arr  <- bootstrap.urban.2$mean.mort.clin  - bootstrap.urban.2$mean.mort.ai
bootstrap.urban.3$arr  <- bootstrap.urban.3$mean.mort.clin  - bootstrap.urban.3$mean.mort.ai
bootstrap.totvol.1$arr <- bootstrap.totvol.1$mean.mort.clin - bootstrap.totvol.1$mean.mort.ai
bootstrap.totvol.2$arr <- bootstrap.totvol.2$mean.mort.clin - bootstrap.totvol.2$mean.mort.ai
bootstrap.totvol.3$arr <- bootstrap.totvol.3$mean.mort.clin - bootstrap.totvol.3$mean.mort.ai

mort.summary <- describe(bootstrap)
mort.summary <- rbind(mort.summary,describe(bootstrap.age1))
mort.summary <- rbind(mort.summary,describe(bootstrap.age2))
mort.summary <- rbind(mort.summary,describe(bootstrap.age3))
mort.summary <- rbind(mort.summary,describe(bootstrap.age4))
mort.summary <- rbind(mort.summary,describe(bootstrap.gend0))
mort.summary <- rbind(mort.summary,describe(bootstrap.gend1))
mort.summary <- rbind(mort.summary,describe(bootstrap.sofa1))
mort.summary <- rbind(mort.summary,describe(bootstrap.sofa2))
mort.summary <- rbind(mort.summary,describe(bootstrap.sofa3))
mort.summary <- rbind(mort.summary,describe(bootstrap.sofa4))
```

```
mort.summary <- rbind(mort.summary,describe(bootstrap.prior0))
mort.summary <- rbind(mort.summary,describe(bootstrap.prior1))
mort.summary <- rbind(mort.summary,describe(bootstrap.surg0))
mort.summary <- rbind(mort.summary,describe(bootstrap.surg1))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp1))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp2))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp3))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp4))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp5))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp6))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp7))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp8))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp9))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp10))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp11))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp12))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp13))
mort.summary <- rbind(mort.summary,describe(bootstrap.hosp14))
mort.summary <- rbind(mort.summary,describe(bootstrap.teach.1))
mort.summary <- rbind(mort.summary,describe(bootstrap.teach.2))
mort.summary <- rbind(mort.summary,describe(bootstrap.urban.1))
mort.summary <- rbind(mort.summary,describe(bootstrap.urban.2))
mort.summary <- rbind(mort.summary,describe(bootstrap.urban.3))
mort.summary <- rbind(mort.summary,describe(bootstrap.totvol.1))
mort.summary <- rbind(mort.summary,describe(bootstrap.totvol.2))
mort.summary <- rbind(mort.summary,describe(bootstrap.totvol.3))

mort.summary$ul.95 <- mort.summary$mean + qnorm(0.975)*mort.summary$sd
mort.summary$ll.95 <- mort.summary$mean + qnorm(0.025)*mort.summary$sd

write.csv(mort.summary,file="C:\\Users\\Jason\\Box Sync\\Current Grants\\Sepsis AI\\Analysis\\ICU
Cohort - Mortality Summary.csv")
```

Plots and graphical outputs were created in R, version 4.0.3 and Prism Graphpad, version 9.1.0.

Bibliography

Andrews B, Semler MW, et al. Effect of an Early Resuscitation Protocol on In-hospital Mortality Among Adults With Sepsis and Hypotension: A Randomized Clinical Trial. JAMA. 2017;318(13):1233-1240.

Angus DC, Linde-Zwirble WT, Lidicker J, et al. Epidemiology of severe sepsis in the United States. Crit Care Med. 2001;297:1303-1310.

Angus DC, van der Poll, T. Severe sepsis and septic shock. N Engl J Med. 2013;369:2063.

Beck JR, Pauker SG. The Markov Process in Medical Prognosis. Medical Decision Making. 1983;3:419-458.

Brown SM, Lanspa MJ, Jones JP, et al. Survival after shock requiring high-dose vasopressor therapy. Chest. 2012;143(3):664-671.

Burkov, A. The hundred-page machine learning book. Quebec City, Canada: Andriy Burkov, 2019.

Elixhauser A, Steiner C, Harris DR, Coffey RM. Comorbidity measures for use with administrative data. Med Care. 1998;36:8-27.

Faust JS, Weingart SD. The past, present and future of Centers for Medicare and Medicaid Services Quality Measure SEP-1: The early management bundle for severe sepsis/septic shock. . Emerg Med Clin North Am. 2017;35:219-231.

Hjortrump PB, Haase N, et al. Restricting volumes of resuscitation fluid in adults with septic shock after initial management: the CLASSIC randomised, parallel-group, multicentre feasibility trial. Intensive Care Med. 2016;42:1695–1705.

Howard RA. Dynamic programming and Markov Processes. Cambridge: MIT Press; 1960.

Hug, C. Detecting hazardous intensive care patient episodes using real-time mortality models PhD thesis, Massachusetts Institute of Technology. 2009.

James G, Witten D, Hastie T, Tibshirani R. An introduction to statistical learning with applications in R. New York, NY: Springer; 2017.

Kodinarita TM, Makwana PR. Review on determining number of cluster in K-Means Clustering. International Journal of Advance Research in Computer Science and Management Studies. 2013;1(6):90-95.

Kokla M, Virtanen J, et al. Random forest-based imputation outperforms other methods for imputing LC-MS metabolomics data: a comparative study. BMC Bioinformatics. 2019;20:492.

Komorowski M, Celi LA, Badawi O, Gordon AC, Faisal AA. The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. Nature Med. 2018;24:1716-1720.

Levy MM, Evans LE, Rhodes A. The Surviving Sepsis Campaign Bundle: 2018 update. Intensive Care Med. 2018;44(6):925-928.

Lloyd, SP. Least squares quantization in PCM. Technical Report RR-5497, Bell Lab, September 1957.

Lyu, Ruishen. Improving Treatment Decisions for Sepsis Patients by Reinforcement Learning. Master's Thesis. University of Pittsburgh. 2020.

MacQueen, JB. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam & J. Neyman (Eds.), Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, pp. 281–297). California: University of California Press. 1967.

Maitland K, George EC, et al. Exploring mechanisms of excess mortality with early fluid resuscitation: Insights from the FEAST trial. BMC Medicine. 2013;11(68).

Mayer, M. missRanger: Fast Imputation of Missing Values. R package. 2019. V2.1.0. https://CRAN.R-project.org/package=missRanger.

Newgard CD, Haukoos, JS. Advanced statistics: missing data in clinical research—part 2: multiple imputation. Acad Emerg Med. 2007;14:669-678.

Proellochs, N and Feuerriegel, S. ReinforcementLearning: Model-Free Reinforcement Learning. R package. 2020. V1.0.5. https://CRAN.R-project.org/package=ReinforcementLearning.

Reynolds PM, Wells L, MacLaren R, Scoular SK. Establishing the Therapeutic Index of Fluid Resuscitation in the Septic Patient: A Narrative Review and Meta-Analysis. Pharmacotherapy. 2020;40(3):256-269.

Roberts SJ. Novelty detection using extreme value statistics. IEE-Proceedings — Vision, Imagine and Signal Processing. 1999;146:124-129.

Rudd KE, Johnson SC, Agesa KM, et al. Global, regional, and national sepsis incidence and mortality, 1990-2017: analysis for the Global Burden of Disease Study Lancet. 2020;395:200-211.

Schaefer AJ, Bailey, MD, Schechter SM, Roberts, MS. Modeling Medical Treatment Using Markov Decision Processes. Boston, MA: Springer; 2005.

Self WH, Semler MW, Bellomo R, et al. Liberal versus restrictive intravenous fluid therapy for early septic shock: Rationale for randomized trial. Ann Emerg Med. 2018;72:457-466.

Seymour CW, Kennedy JK, Wang S, et al. Derivation, validation and potential treatment implications of novel clinical phenotypes for sepsis. JAMA. 2019;321:2003-2007.

Seymour CW, Liu VX, Iwashyna TJ, et al. Assessment of clinical criteria for sepsis for the Third International Consensus definitions for sepsis and septic shock (Sepsis-3). JAMA. 2016;315:762-774.

Shah AD, Bartlett JW, et al. Comparison of random forest and parametric imputation models for imputing missing data using MICE: A CALIBER Study. American Journal of Epidemiology. 2014;179(6):764-774.

Singer M, Deutschman, CS, Seymour CW, et al. . The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3). JAMA. 2016;315:801-810.

Stekhoven DJ, Buhlmann P. MissForest: non-parametric missing value imputation for mixed-type data. Bioinformatics. 2011;28(1):112-118.

Sutton RS, Barto AG. Reinforcement Learning: An Introduction. . Cambgidge, MA, USA: MIT Press, 2018.

Vincent JL, Moreno R, Takala J, et al. The SOFA (Sepsis-related Organ Failure Assessment) score to describe organ dysfunction/failure: on behalf of the Working Group on Sepsis-Related Problems of the European Society of Intensive Care Medicine. Intensive Care Med. 1996;22:707-710.

Voloshin C, Le HM, Yue Y. Empirical Analysis of Off-Policy Policy Evaluation for Reinforcement Learning. Real-world Sequential Decision-Making Workshop, International Conference on Machine Learning. Long Beach, USA. 2019.

Waljee AK, Mukherjee A, et al. Comparison of imputation methods for missing laboratory data in medicine. BMJ Open. 2013;3:e002847.

Watkins, CJ, Dayan, P. Q-learning. Machine Learning. 1992;8(3-4):279–292.