Off by a Beat: Temporal Misalignment in Offline RL for Healthcare

Shengpu Tang[1], **Jiayu Yao**[2], **Jenna Wiens**[3], **Sonali Parbhoo**[4]
shengpu.tang@emory.edu, jy3491@columbia.edu, wiensj@umich.edu, s.parbhoo@imperial.ac.uk

[1]Emory University [2]Columbia University [3]University of Michigan [4]Imperial College London

Abstract

Reinforcement learning (RL) is typically applied to environments with well-defined discrete timesteps. However, real-world domains like healthcare often involve irregularly sampled time-series data that require preprocessing. After aggregating the data into fixed-length time windows, it is common practice to align each state with the action that occurred within the same window. We argue that this temporal alignment strategy is problematic, as it effectively allows a policy to rely on future information. Using a toy control task, we demonstrate that the default alignment can result in an incorrect transition function and a learned policy that systematically recommends wrong actions. More worrisome, in a case study of RL for sepsis management on the MIMIC-III dataset, we found that different alignment strategies can produce deceptively similar performance for common global metrics but result in different treatment recommendations in nearly half of the patient states. Our findings highlight an underappreciated, yet critical issue when applying RL to these domains. We advocate for a straightforward fix to prevent temporal information leakage by aligning each state with the action in the next window. Given the prevalence of the temporal misalignment issue in existing literature, we urge the community to carefully reconsider the temporal alignment step, especially when working on RL for high-stakes domains like healthcare.

1 Introduction

Research in reinforcement learning (RL) usually deals with idealized dynamical systems that evolve at discrete timesteps (Sutton & Barto, 2018). The specification of these systems is typically given in the format of an interaction protocol with respect to some environment. For example, domains in the Gymnasium package (Brockman et al., 2016; Towers et al., 2024) (including classic control problems such as LunarLander and MountainCar) implement an `env.step(action)` function that advances the environment by a single timestep. Turn-based games often happen in discrete rounds (Silver et al., 2016; 2017), e.g., in chess, the agent makes a move, and then the opponent makes a move. In these environments, there is no ambiguity in what a "timestep" means.

Yet we do not always have this level of clarity when applying RL in real-world settings, especially when the Markov decision process (MDP) needs to be defined by the practitioner. For instance, when modeling decision making tasks in healthcare, we have access to irregular time series from the electronic health record (EHR). To make such data "RL-ready", it is common to aggregate the time series at regular time windows. In the widely used sepsis management domain (Raghu et al., 2017; Komorowski et al., 2018; Jeter et al., 2019; Killian et al., 2020; Tang et al., 2020; Satija et al., 2021; Tang et al., 2022; Liang et al., 2023; Choudhary et al., 2024; Tu et al., 2025), patient data are aggregated into 4-hour windows. A state is derived in part from physiological measurements obtained during a 4-hour window, and the corresponding action is the set of treatments administered during the *same* 4-hour window. After preprocessing raw data into trajectories of state-action pairs, standard offline RL approaches are applied to optimize and evaluate treatment policies.

However, there is a subtle issue in the default temporal alignment strategy. In RL, a state-action pair typically assumes that the state temporally precedes the action, i.e., the agent observes the state and then selects an action in response. Aligning a state with the action occurring within the same window implicitly assumes that the action occurs at the end of the window (i.e., after the state has been fully observed). Yet in real-world settings, this is not guaranteed: the action may occur at the beginning or middle of the window, possibly before all components of the state are realized. Deriving the state and action from data within the same time window "collapses" the interval to an instant, introducing potential "temporal leakage" as effectively, the policy would be using future information to predict an action that has *already occurred*.

While the choice of temporal alignment might be dismissed as an innocuous design choice or a small implementation detail, we demonstrate that it has significant downstream consequences. First, on a toy continuous-time control task, we show that the default alignment leads to an incorrect estimate of the transition function, resulting in a learned policy that systematically recommends the *wrong* actions. Second, in the sepsis management domain, we find that such issues are *not easily detectable* when following standard offline RL pipelines of learning and evaluation. In particular, different alignment strategies can yield deceptively similar performance under commonly used evaluation methods, including quantitative off-policy evaluation (OPE) metrics and global qualitative trends, and yet make drastically different treatment recommendations. Our findings highlight temporal aggregation and alignment as a key preprocessing step that shapes what a policy will learn. As RL is increasingly used in high-stakes domains like healthcare, we call the community's attention to how trajectories are constructed from logged data, recognizing that correct temporal alignment and thoughtful evaluation is essential for the responsible application of RL in the real world.

2 Problem Formulation & Notation

Consider a time-series dataset D that has been aggregated at a prespecified temporal granularity, where samples are indexed by i ranging from 1 to N and timesteps are indexed by t ranging from 1 to T (without loss of generality, we assume each trajectory has the same length). Let $D = \{\{x_t^{(i)}, z_t^{(i)}\}_{t=1}^{T}\}_{i=1}^{N}$ where the superscript (i) denotes the sample index (omitted when referring to a generic sample), $x_t \in \mathbb{R}^{d_x}$ denotes the observations recorded at step t, $z_t \in \mathbb{R}^{d_z}$ denotes the interventions recorded at step t, which are d_x- and d_z-dimensional vectors. Each pair of x_t, z_t can be seen as occurring *contemporaneously* since we lose all temporal information within a time window after temporal data aggregation. We intentionally use terminologies of "observations" and "interventions" as this paper focuses on the alignment of observation-intervention pairs as state-action pairs. We consider the following alignment strategies, summarized in Table 1.

Table 1: Temporal alignment of observation-intervention pairs as state-action pairs.

Alignment	States	Actions
Original	$s_t = x_t\ ,\ t \in 1 \dots T$	$a_t = z_t\ ,\ t \in 1 \dots T-1$
Shifted	$s_t = x_t\ ,\ t \in 1 \dots T$	$a_t = z_{t+1}\ ,\ t \in 1 \dots T-1$

- Original. This approach defines the state-action pair at step t to be the observations and interventions recorded at step t. Since we need to form transition tuples of (state,action,next-state), the total number of steps is $T-1$, starting at (x_1, z_1, x_2) and ending at (x_{T-1}, z_{T-1}, x_T). Note that the intervention z_T at step T is not used since no data is observed after step T.
- Shifted. Here, the state-action pair at step t consists of the observations recorded at step t and interventions recorded at step $t+1$. The total number of steps is also $T-1$, starting at (x_1, z_2, x_2) and ending at (x_{T-1}, z_T, x_T). For clarity of comparison, we do not use the intervention z_1 at step 1; in practice, one may choose to aggregate it with x_1 and view it as part of the state s_1.

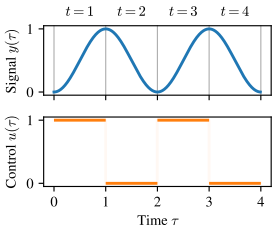
The two alignment strategies represent different interpretations of how the causal relationships of an MDP manifest in the data (Figure 6). In an MDP, state s_t causally influences action a_t according to the logging policy $\pi(a|s)$, and (s_t, a_t) causally influences s_{t+1} according to the transition model $p(s'|s, a)$. The Original alignment assumes that each action takes place at the end of the window

(i.e., observation s_t temporally precedes intervention z_t). Thus, $s_t = x_t$ is aligned with $a_t = z_t$, with $x_t \to z_t$. In contrast, the Shifted alignment assumes that an action can take place anywhere in the window, which is more realistic, and aligns $s_t = x_t$ with $a_t = z_{t+1}$ so that $x_t \to z_{t+1}$.

3 An Illustrative Example

To illustrate the difference between the two alignment strategies, consider the following continuous-time control problem, inspired by physiological regulation tasks such as hypotension management. The system involves a one-dimensional signal $y(\tau) \in \mathbb{R}$ which the agent modulates via a binary control input $u(\tau) \in \{0, 1\}$ ("off" and "on"), with τ denoting the (continuous) time. The system dynamics are such that the signal y increases when the control input u is 1, and decreases when the control input is 0. In other words, the time-derivative $y'(\tau) > 0$ when $u(\tau) = 1$, and $y'(\tau) < 0$ when $u(\tau) = 0$. As a concrete example, blood pressure rises with intravenous vasopressors, and the effect onset is nearly instantaneous (Drugs.com, 2024; 2025).

Suppose we collected data using a controller that oscillates between on and off at fixed time intervals, producing a periodic signal (Figure 1). We assume that the time series data of control inputs and signal values are aggregated at the same frequency of the controller, resulting in $T = 4$ timesteps. For simplicity, we use a binary space of observations that captures whether the signal value is decreasing or increasing ($x = 0$: $y' < 0$; $x = 1$: $y' > 0$), and a binary space of interventions equivalent to the space of control inputs. After mapping the time series data to trajectories under the two alignment strategies, we estimate the transition probabilities $p(s'|s, a)$ from the resulting dataset (Figure 2) (we do not consider the reward function in this example). Under Original, the learned transition function incorrectly implies that "taking action 0 (controller off) in state 0 (signal decreasing) leads to state 1 (signal increasing)". This contradicts the system's true dynamics, where turning the controller off should maintain or continue the decreasing trend in the signal. Consequently, optimizing a policy using this transition function will systematically recommend the *wrong* action (e.g., recommending withholding vasopressors when blood pressure is low). In contrast, Shifted learns the correct transition function and does not suffer from this issue.



Time Series Dataset

t	Observation x_t	Intervention z_t
1	$x_1 = 1$, increasing	$z_1 = 1$, on
2	$x_2 = 0$, decreasing	$z_2 = 0$, off
3	$x_3 = 1$, increasing	$z_3 = 1$, on
4	$x_4 = 0$, decreasing	$z_4 = 0$, off

Figure 1: A dataset collected on the toy control problem.

Alignment Strategy	Trajectory							Learned Transition Probabilities				
Original	s_1	a_1	s_2	a_2	s_3	a_3	s_4	(s, a)	$(0, 0)$	$(0, 1)$	$(1, 0)$	$(1, 1)$
	x_1	z_1	x_2	z_2	x_3	z_3	x_4	$s' = 0$	0.0	NA	NA	1.0
	1	1	0	0	1	1	0	$s' = 1$	1.0	NA	NA	0.0
Shifted	s_1	a_1	s_2	a_2	s_3	a_3	s_4	(s, a)	$(0, 0)$	$(0, 1)$	$(1, 0)$	$(1, 1)$
	x_1	z_2	x_2	z_3	x_3	z_4	x_4	$s' = 0$	NA	0.0	1.0	NA
	1	0	0	1	1	0	0	$s' = 1$	NA	1.0	0.0	NA

Figure 2: Trajectories and learned transition probabilities under the two alignment strategies.

4 Evidence from Empirical Experiments

To understand the effect of different temporal alignment strategies on policy optimization, we conducted experiments in two domains. First, we extended the toy control problem with a stochastic transition function and analyze how temporal alignment influences the learned transition dynamics and reward function, as well as the learned policy. Then, we investigate a full offline RL pipeline applied to the MIMIC-III sepsis management task. In both settings, we compare the final learned policies by commonly reported performance metrics, as well as intermediate modeling artifacts, such as estimated transition functions, value functions, and action distributions.

4.1 Synthetic Domain

Rationale. Building upon the toy problem discussed in Section 3, here, we simulate the scenario where we apply model-based RL to a dataset collected from this synthetic control task.

Setup. We use the same definitions as above for observations, interventions, states, and actions, and modified the environment to include rewards, discounting, and stochastic transitions. An immediate reward is given at every step, $+1$ if signal is increasing, and 0 if signal is decreasing. The discount factor is set to $\gamma = 0.9$. To introduce stochasticity to the system, we assume the controller input is ineffective with probability $p_{\text{slip}} = 0.2$ such that the previous observation persists; for example, in Figure 1 when the control input is set to "off" ($z_2 = 0$), there is a 20% chance that the signal will continue increasing ($x_2 = x_1 = 1$). Based on the system specification, the optimal policy is to always keep the controller input as "on" regardless of the state.

Learning Procedure. For each run of the experiment, we collected a dataset following a randomly generated behavior policy, which assigns randomized probabilities to the binary controller inputs. The dataset for each run contained 100 trajectories truncated at 100 steps. Given the dataset, we applied a model-based RL pipeline by first estimating the transition function $\hat{p}(s'|s,a)$ and reward function $\hat{r}(s,a)$ from data, and then using value iteration with $\hat{p}$ and $\hat{r}$ to learn the optimal value function V^* and Q^* and optimal policy π^*. We applied the same procedure to the dataset preprocessed with each alignment strategy, and compared the resulting estimated transition and reward functions, value functions, and learned policies. We present results averaged over 100 runs.

Results. As shown in Figure 3, the two alignment strategies led to markedly different results. For Original, the learned transition and reward functions are nearly independent of actions, e.g., $\hat{p}(0|0,0) = 0.61$, $\hat{p}(0|0,1) = 0.64$, and $\hat{r}(0,0) = 0.39$, $\hat{r}(0,1) = 0.36$. As a result, the learned Q^* values are also nearly independent of actions, and on average, the learned optimal policy recommends taking action 0 "off" more than 50% of the time on average (94% for state "decreasing" and 17% for state "increasing"). In contrast, for Shifted we learn the correct transition and reward functions that depend on the action, e.g., $\hat{p}(0|0,0) = 1.00$, $\hat{p}(0|0,1) = 0.20$, and $\hat{r}(0,0) = 0.00$, $\hat{r}(0,1) = 0.80$, leading to more sensible value functions and the correct optimal policy that always recommends action 1 "on". Applied to the ground-truth environment, the policy learned under Original achieves an average return of 3.34, substantially underperforming the policy learned under Shifted which achieves an average return of 9.88.

Takeaways. The Original alignment strategy learns incorrect transition and reward functions, which leads to a learned policy that systematically recommends the wrong action with high probability, whereas Shifted does not suffer from this issue.

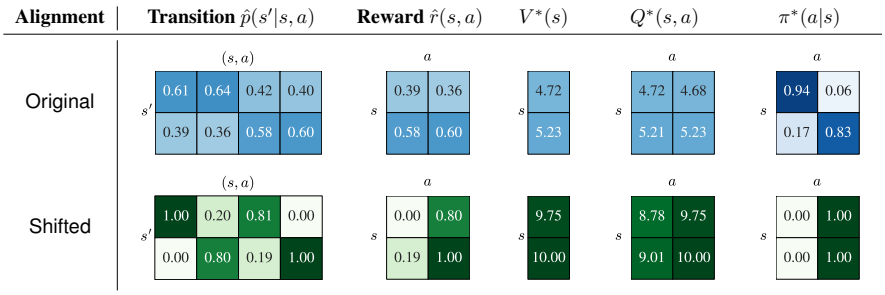


Figure 3: Comparison of transition functions, reward functions, value functions, and learned policies on the toy control problem under the two temporal alignment strategies. Rows and columns are [0,1] by default, and the columns for the transition matrix are state-action pairs [0,0; 0,1; 1,0; 1,1].

4.2 Real-World Clinical Domain

Rationale. The synthetic domain serves as an extreme example, where we assumed full knowledge of the underlying system to facilitate our understanding of what can go wrong under temporal misalignment. We now turn to a more realistic scenario that mirrors what practitioners might face. Our focus here is understanding, without access to the true environment, whether we can diagnose such failure modes using the information available in an offline RL setting.

Setup. We study the task of learning policies for sepsis treatment using the MIMIC-III EHR database (Johnson et al., 2016), following the cohort selection criteria of Komorowski et al. (2018), Killian et al. (2020), and Tang et al. (2022). This yielded 19,287 patients (9.6% mortality), which were split 70/15/15 for training, validation, and testing. For each patient, we extracted a time series from 24h pre- to 48h post-sepsis onset, aggregated at 4h intervals, with each interval containing both observations and interventions. Observations are 38-dimensional vectors comprising demographic, physiological, and lab features (carry-forward imputed as needed), discretized into 750 states via k-means clustering. Each intervention pertains to treatments recorded within the 4h window, representing total volume of intravenous (IV) fluids and amount of vasopressors administered, the dosages of which are discretized to form an action space of size 25 (Figure 7) (Tang et al., 2020). A terminal reward of $+1$ (survival) or -1 (death within 48h of ICU discharge) is assigned, with terminal states added for both outcomes; intermediate rewards are set to 0.

Learning Procedure. We constructed two dataset versions using either the Original or Shifted alignment between observations and interventions to form state-action pairs, keeping the same state space and action space, and applied a consistent model-based offline RL pipeline. Similar to the synthetic domain, the transition probabilities were learned via maximum likelihood estimation. The reward function (based on terminal outcomes) was assumed known. The optimal policy was then learned using a modified value iteration algorithm that, for each state, excluded actions that occurred fewer than five times in the training data. If the set of allowed actions is empty, the most frequent action was used. We evaluated the learned policy via off-policy evaluation (OPE) on the test set using doubly-robust estimators (DR and WDR) (Jiang & Li, 2016), leveraging a clinician policy estimated via behavior cloning. We used $\gamma = 0.99$ for learning and $\gamma = 1$ for evaluation. In addition to quantitative metrics, we qualitatively analyzed the treatment recommendations of the clinician policy π_b and the RL policy π^* by visualizing action frequencies across the test set.

Results. The two alignment strategies led to very similar results in terms of OPE performance metrics and the action frequency heatmaps (Figure 4). In both cases, the learned optimal policy achieves an estimated value ranging between 0.86 and 0.93, outperforming the behavior policy which has a value of around 0.78. The heatmaps of the observed clinician policies and the learned RL policies under different alignment strategies are nearly indistinguishable, with total variation distances (TVD) of less than 0.01 and 0.19 (in the action frequency distributions), respectively. However, since we do not have access to the true environment, both policies may appear equally reasonable to practitioners without a deeper understanding of the recommended treatments. We also monitored the algorithm performance during training: although the algorithm converged at different rates under the two alignment strategies, the distributions of the value functions look very similar throughout all iterations (Figure 8).

Upon Closer Inspection... In addition to the evaluation metrics above, we compared intermediate modeling artifacts under the two alignment strategies (as we did for the synthetic domain). While the histograms of transition probabilities look nearly identical (Figure 9), the two transition models differ substantially. Among all $18750 = 750 \times 25$ state-action pairs, 10783 (57.5%) have a next state distribution, $\hat{p}(\cdot|s, a)$, for which the TVD is > 0.10 under the two alignment strategies. Finally, we compared the two learned policies across individual states rather than globally by measuring the consistency in their optimal actions. Specifically, we identified states for which the optimal action differ between the policies learned under Original and Shifted alignment strategies, and categorized them into four "quadrants" (Figure 5-top right). Overall, the two policies recommended different optimal actions in 336 (44.8%) out of all 750 states (heatmap in Figure 5-left). In the orange quadrant (311 states in the bottom left cell), both Original and Shifted recommend action 0 ("no treatment"), which is consistent with the frequency of this action seen in past literature (Tang et al., 2020). However, in the red quadrant (171 states in the leftmost column), Shifted recommends "no treatment" whereas Original recommends some form of treatment (fluids, vasopressors or both with action index > 0). Conversely, in the pink quadrant (57 states in the bottom row), Original recommends "no treatment" whereas Shifted recommends some form of treatment. In the purple quadrant (the remaining 212 states), both policies recommend some form of treatment.

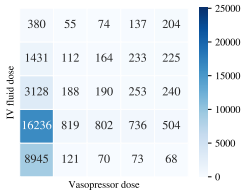
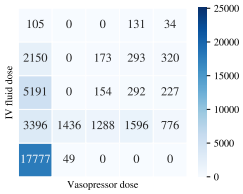
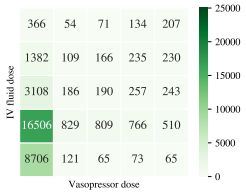
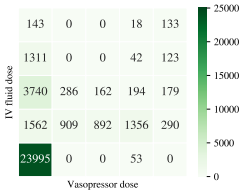
Alignment	OPE Metrics		Clinician Policy	RL Policy
Original	Policy	Value Estimate		
	π_b	0.782 ± 0.010		
	π^* (DR)	0.864 ± 0.006		
	π^* (WDR)	0.908 ± 0.003		
Shifted	Policy	Value Estimate		
	π_b	0.777 ± 0.011		
	π^* (DR)	0.870 ± 0.007		
	π^* (WDR)	0.929 ± 0.003		

Figure 4: Quantitative OPE performance on the test set, ± standard errors from 1000 bootstraps, as well as heatmap visualizations of action frequencies under clinician policy and RL policy for qualitative comparisons. Action space is defined in Figure 7.

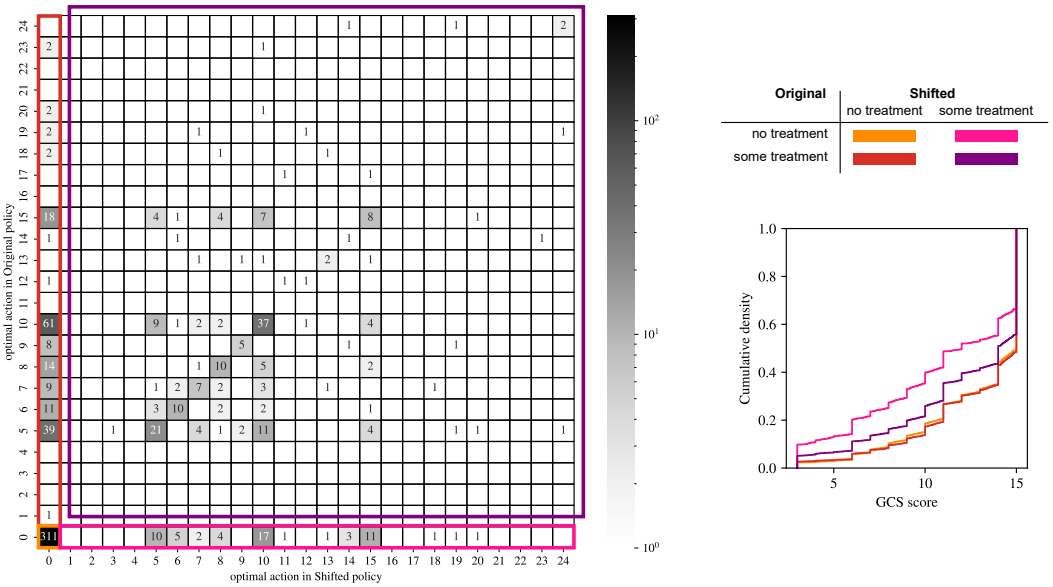


Figure 5: Top right - categorization of states based on how the two learned policies differ. Left - comparison of optimal actions of policies learned under the two temporal alignment strategies. If the two policies are consistent, the heatmap would have counts concentrated on the diagonal. Each colored box corresponds to a quadrant defined in the top right table. Bottom right - empirical cumulative distribution functions of GCS scores for patients in each quadrant.

To better understand factors associated with divergent recommendations, we characterized patients in each "quadrant" by their GCS scores (ranging 3 to 15; lower is sicker and higher is healthier) (Teasdale & Jennett, 1974) and compared the GCS score distributions using Mann-Whitney U tests with a Bonferroni correction (Figure 5-bottom right). On average, patients in the test set have a mean GCS of 12.62. Patients in the pink quadrant, where Original recommends no treatment but Shifted recommends some treatment, are significantly sicker than average (mean GCS of 10.95 with $p < 0.001$). They are even sicker than the patients in the purple quadrant, where both policies recommend some treatment (mean GCS of 12.19 with $p < 0.001$). In contrast, patients in the red quadrant, where Shifted recommends no treatment but Original recommends some treatment, are significantly healthier than average (mean GCS of 12.98 with $p < 0.001$), and they are comparable to patients in the orange quadrant (mean GCS of 12.93 with $p \geq 0.05$). Overall, the Original policy tends to recommend more aggressive treatments for healthier patients while withholding treatments

for sicker patients, which is counterintuitive. Analysis using SOFA scores (Moreno et al., 2023) led to a similar conclusion (see Figure 10 in appendix). We believe this may be an indication of the same phenomenon we saw in the synthetic domain, where Original results in a learned policy that systematically recommends the wrong actions.

Takeaways. Following a standard offline RL pipeline of policy learning and evaluation, we found that the global trends in quantitative and qualitative evaluation results failed to reveal the issue of incorrect temporal alignment. Our follow-up analysis revealed large differences in the learned policies resulting from the two alignment strategies, with the policy learned under the Original alignment likely recommending incorrect actions.

5 Discussion & Conclusion

In this work, we identify a widespread but commonly overlooked issue when applying RL to time-series datasets, where the observation and the intervention recorded within the same time window are aligned to form a state-action pair. On a synthetic control domain and a case study using real-world clinical data, we demonstrate that this default alignment strategy can lead to an incorrect transition model, and subsequently, a policy that recommends harmful actions. To the best of our knowledge, for healthcare RL domains, this issue has only been mentioned in anecdotal conversations (e.g., Tang (2024); in a GitHub issue https://github.com/microsoft/mimic_sepsis/issues/12 that had suggested an incorrect fix), and the majority of the literature still builds upon a flawed problem formulation (Table 2), including some of the authors' prior work. This may be largely due to the fact that the performance metrics will appear reasonable even under an incorrect temporal alignment (as seen in our experiments), since this issue is replicated across training and evaluation. While we only considered a simple model-based offline RL approach, it is concerning to see the big difference caused by temporal misalignment in this simple setting, when more complex RL algorithms used in practice may further obfuscate the issue.

From a causal perspective, the difference between the two alignment strategies lies their underlying assumption regarding the causal relationship between the observations x_t and the interventions z_t (Figure 6): Original assumes x_t causally influences z_t, whereas Shifted assumes x_t causally influences z_{t+1}. In practice, the observation x_t is usually constructed using all data available within the window and thus only available at the end of the window. However, the intervention z_t could occur anywhere within the window and needs to be determined in advance. Thus, the causal assumption of Original violates the temporal ordering of events, wherein x_t does not necessarily precedes z_t temporally. While we advocate for using the Shifted alignment in favor of Original, we caution that this is likely an artifact of multiple design choices, including the selection of temporal granularity (Schulam & Saria, 2018; Adams et al., 2020), the timing of observations and interventions within a time window, and the definition of state and action spaces. We urge the community to recognize temporal alignment as a crucial aspect of problem formulation, alongside other more commonly discussed elements of RL such as state, action, and reward designs (Killian et al., 2020; Tang et al., 2022), especially as RL gains attention in high-stakes domains such as healthcare.

Table 2: The temporal alignment strategy used in prior work that studied the MIMIC-III sepsis management domain, based on authors' review of publicly available code bases. Example code snippets for both Original and Shifted alignment are provided in Appendix C.

Reference	Code Repository	Original?	Shifted?
Raghu et al. (2017)	https://github.com/aniruddhraghu/sepsisrl	✓	
Komorowski et al. (2018)	https://github.com/matthieukomorowski/AI_Clinician	✓	
Jeter et al. (2019)	https://github.com/point85AI/Policy-Iteration-AI-Clinician	✓	
Killian et al. (2020)	https://github.com/MLforHealth/rl_representations	✓	
Tang et al. (2020)	https://github.com/MLD3/RL-Set-Valued-Policy	✓	
Fatemi et al. (2021)	https://github.com/microsoft/med-deadend	✓	
unpublished (2021)	https://github.com/microsoft/mimic_sepsis	✓	
Ji et al. (2021)	https://github.com/clinicalml/trajectory-inspection	✓	
Satija et al. (2021)	https://github.com/hercky/mo-spibb-codebase	✓	
Tang et al. (2022)	https://github.com/MLD3/OfflineRL_FactoredActions		✓
unpublished (2022)	https://github.com/cmudig/AI-Clinician-MIMICIV		✓
Liang et al. (2023)	https://github.com/DMU-XMU/Episodic-Memory-assisted-Approach-for-Sepsis-Treatment	✓	
Choudhary et al. (2024)	https://github.com/icu-sepsis/icu-sepsis	✓	
Tu et al. (2025)	https://github.com/OOPSDINOSAUR/RL_safety_model	✓	

Data and Code Availability

Our code is available at https://github.com/shengpu-tang/RL-Off-by-a-Beat.

Acknowledgments

The authors would like thank Michael Sjoding and Jung Min Lee for helpful discussions regarding this work. This work was supported in part by the National Library of Medicine of the National Institutes of Health (grant R01LM013325 to JW). The views and conclusions in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the National Institutes of Health.

References

Roy Adams, Suchi Saria, and Michael Rosenblum. The impact of time series length and discretization on longitudinal causal estimation methods. *arXiv preprint arXiv:2011.15099*, 2020.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Kartik Choudhary, Dhawal Gupta, and Philip S. Thomas. ICU-sepsis: A benchmark MDP built from real medical data. In *Reinforcement Learning Conference*, 2024. URL https://openreview.net/forum?id=hAb2LdotMQ.

Drugs.com. Phenylephrine hydrochloride monograph for professionals, 2024. Available: https://www.drugs.com/monograph/phenylephrine-hydrochloride.html. Last accessed on 2025-05-22.

Drugs.com. Norepinephrine bitartrate monograph for professionals, 2025. Available: https://www.drugs.com/monograph/norepinephrine-bitartrate.html. Last accessed on 2025-05-22.

Mehdi Fatemi, Taylor W. Killian, Jayakumar Subramanian, and Marzyeh Ghassemi. Medical dead-ends and learning to identify high-risk states and treatments. In *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=4CRpaV4pYp.

Russell Jeter, Christopher Josef, Supreeth Shashikumar, and Shamim Nemati. Does the "Artificial Intelligence Clinician" learn optimal treatment strategies for sepsis in intensive care? *arXiv preprint arXiv:1902.03271*, 2019. URL https://arxiv.org/abs/1902.03271.

Christina X Ji, Michael Oberst, Sanjat Kanjilal, and David Sontag. Trajectory inspection: A method for iterative clinician-driven design of reinforcement learning studies. *AMIA Summits on Translational Science Proceedings*, 2021:305, 2021.

Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pp. 652–661. PMLR, 2016. URL https://proceedings.mlr.press/v48/jiang16.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. MIMIC-III, a freely accessible critical care database. *Scientific Data*, 3(1):1–9, 2016. URL https://doi.org/10.1038/sdata.2016.35.

Taylor W Killian, Haoran Zhang, Jayakumar Subramanian, Mehdi Fatemi, and Marzyeh Ghassemi. An empirical study of representation learning for reinforcement learning in healthcare. In *Proceedings of the Machine Learning for Health NeurIPS Workshop*, pp. 139–160. PMLR, 2020. URL https://proceedings.mlr.press/v136/killian20a.

Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The Artificial Intelligence Clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11):1716–1720, 2018. URL https://doi.org/10.1038/s41591-018-0213-5.

Dayang Liang, Huiyi Deng, and Yunlong Liu. The treatment of sepsis: an episodic memory-assisted deep reinforcement learning approach. *Applied Intelligence*, 53(9):11034–11044, 2023.

Rui Moreno, Andrew Rhodes, Lise Piquilloud, Glenn Hernandez, Jukka Takala, Hayley B Gershengorn, Miguel Tavares, Craig M Coopersmith, Sheila N Myatra, Mervyn Singer, et al. The sequential organ failure assessment (SOFA) score: has the time come for an update? *Critical care*, 27(1):15, 2023.

Aniruddh Raghu, Matthieu Komorowski, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Continuous state-space models for optimal sepsis treatment: a deep reinforcement learning approach. In *Proceedings of the 2nd Machine Learning for Healthcare Conference*, volume 68, pp. 147–163. PMLR, 2017. URL https://proceedings.mlr.press/v68/raghu17a.

Harsh Satija, Philip S Thomas, Joelle Pineau, and Romain Laroche. Multi-objective SPIBB: Seldonian offline policy improvement with safety constraints in finite MDPs. *Advances in Neural Information Processing Systems*, 34:2004–2017, 2021.

Peter Schulam and Suchi Saria. Discretizing logged interaction data biases learning for decision-making. *arXiv preprint arXiv:1810.03025*, 2018.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. URL https://doi.org/10.1038/nature16961.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017. URL https://doi.org/10.1038/nature24270.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

Shengpu Tang. Reinforcement learning for healthcare decision making: The perils and promises. Invited talk at the "I Can't Believe It's Not Better!" workshop, Reinforcement Learning Conference (RLC), 2024. Amherst, MA, USA. Available at https://sites.google.com/view/rlc2024-icbinb.

Shengpu Tang, Aditya Modi, Michael Sjoding, and Jenna Wiens. Clinician-in-the-loop decision making: Reinforcement learning with near-optimal set-valued policies. In *International Conference on Machine Learning*, pp. 9387–9396. PMLR, 2020. URL https://proceedings.mlr.press/v119/tang20c.

Shengpu Tang, Maggie Makar, Michael Sjoding, Finale Doshi-Velez, and Jenna Wiens. Leveraging factored action spaces for efficient offline reinforcement learning in healthcare. In *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=Jd70afzIvJ4.

Graham Teasdale and Bryan Jennett. Assessment of coma and impaired consciousness: a practical scale. *The Lancet*, 304(7872):81–84, 1974.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulao, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Rui Tu, Zhipeng Luo, Chuanliang Pan, Zhong Wang, Jie Su, Yu Zhang, and Yifan Wang. Offline safe reinforcement learning for sepsis treatment: Tackling variable-length episodes with sparse rewards. *Human-Centric Intelligent Systems*, 5(1):63–76, 2025.

Kristine Zhang, Henry Wang, Jianzhun Du, Brian Chu, Aldo Robles Arévalo, Ryan Kindle, Leo Anthony Celi, and Finale Doshi-Velez. An interpretable RL framework for pre-deployment modeling in ICU hypotension management. *npj Digital Medicine*, 5(1):173, 2022. URL https://doi.org/10.1038/s41746-022-00708-4.

A Additional Discussion

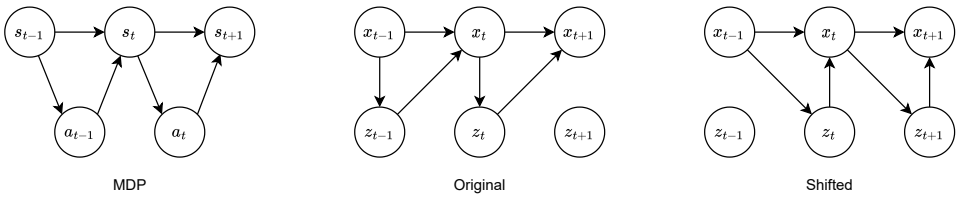


Figure 6: Causal DAGs of an MDP (left) and as implied by Original (middle) and Shifted (right).

B Additional Experimental Results

B.1 MIMIC Experiments

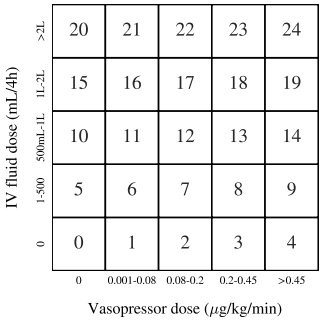


Figure 7: The action space used in the MIMIC sepsis environment. The cells are labeled with the action index ranging from 0 to 24.

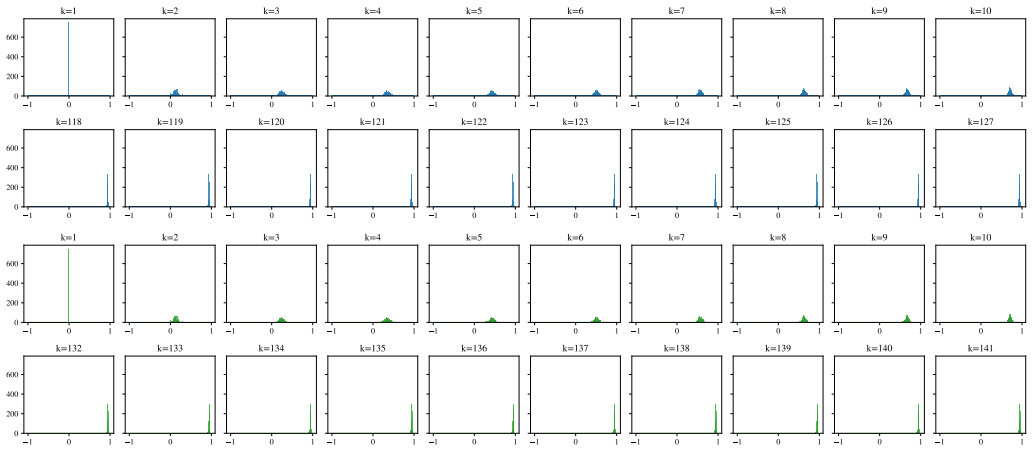


Figure 8: Distributions of value functions during policy learning, for the first 10 and last 10 iterations of the value iteration algorithm. Although Original (top two rows) and Shifted (bottom two rows) took a different number of iterations to reach convergence (127 and 141, respectively), the overall trends and the distributions of values appear similar.

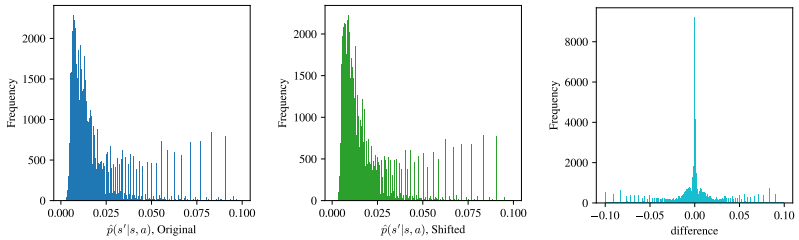


Figure 9: Comparison of the distributions of transition probabilities $\hat{p}(s'|s,a)$ for Original (left) and Shifted (middle) for all (state,action,next-state) tuples, as well as histogram showing the distribution of differences in transition probabilities for each (state,action,next-state) tuple between the two. For clarity, the frequency for 0 on the x-axis is omitted.

Similar to the GCS scores analysis in Figure 5, we compared the distributions of SOFA scores (ranging 0 to 24; higher is sicker, lower is healthier) for patients in each "quadrant" (Figure 10). On average, patients in the test set have a mean SOFA of 5.86. Patients in the pink quadrant, where Original recommends no treatment but Shifted recommends some treatment, are significantly sicker than average (mean SOFA of 7.24 with $p < 0.001$). They are even sicker than the patients in the purple quadrant, where both policies recommend some treatment (mean SOFA of 6.80 with $p < 0.001$). In contrast, patients in the red quadrant, where Shifted recommends no treatment but Original recommends some treatment, are significantly healthier than average (mean SOFA of 5.49 with $p < 0.001$), and they are comparable to patients in the orange quadrant (mean GCS of 5.29 with $p \geq 0.05$). Overall, the Original policy tends to recommend more aggressive treatments for healthier patients while withholding treatments for sicker patients, which is counterintuitive.

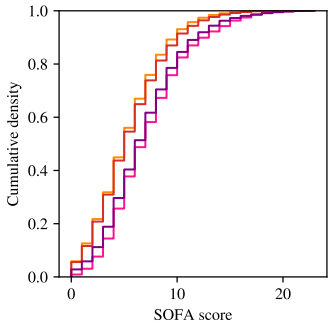


Figure 10: Empirical cumulative distribution functions of SOFA scores for patients in each quadrant.

C Example Code Snippets

To save space, some lines are omitted and some variable names have been modified.

C.1 Examples for **Original**

Raghu et al. (2017)
https://github.com/aniruddhraghu/sepsisrl/blob/master/continuous/q_network.ipynb

```
def process_train_batch(size):
    ...
    for i in a.index:
        cur_state = a.ix[i,state_features]
        iv = int(a.ix[i, 'iv_input'])        ←
        vaso = int(a.ix[i, 'vaso_input'])    ←
        action = action_map[iv,vaso]
        reward = a.ix[i,'reward']
        ...
```

Komorowski et al. (2018)
https://github.com/matthieukomorowski/AI_Clinician/blob/master/AIClinician_core_160219.m#L190C1-L202C9

```
disp('####  CREATE TRANSITION MATRIX T(S'',S,A) ####')
transitionr=zeros(ncl+2,ncl+2,nact);  %this is T(S',S,A)
sums0a0=zeros(ncl+2,nact);
     for i=1:size(qldata3,1)-1
         if (qldata3(i+1,1))~=1
         % if we are not in the last state for this patient = if there is a transition to make
         S0=qldata3(i,2); S1=qldata3(i+1,2);  acid= qldata3(i,3);    ←
         transitionr(S1,S0,acid)=transitionr(S1,S0,acid)+1;
         sums0a0(S0,acid)=sums0a0(S0,acid)+1;
         end
     end
```

Jeter et al. (2019)
https://github.com/point85AI/Policy-Iteration-AI-Clinician/blob/master/model_generation/generate_environment.m#L95

```
function [...] = generate_environment(training_set, K)

%Initialize a cell for all trajectories and clinician actions, then fill
%those cells with the appropriate values from the training set.
N                       = length(training_set);
trajectories_normalized = cell(N, 1);
actions                 = cell(N, 1);

%% Cluster the normalized patient data.
[clusters, centroids] = kmeans(trajectories_normalized, K);
...
    while hour_index < num_hours
    ...
        i      = clusters(index);
        j      = clusters(index + 1);
        action = actions(index);    ←

        transition_matrix(i, j, action) = transition_matrix(i, j, action) + 1;
        ...
    end
```

Killian et al. (2020)
https://github.com/MLforHealth/rl_representations/blob/main/scripts/split_sepsis_cohort.py#L125C1-L139C1

```
for i in trajectories:
    traj_i = train_data[train_data['traj'] == i].sort_values(by='step')
    traj_j = train_acuity[train_acuity['traj']==i].sort_values(by='step')
    data[i] = {}
    data[i]['dem'] = torch.Tensor(traj_i[dem_cols].values).to('cpu')
    data[i]['obs'] = torch.Tensor(traj_i[obs_cols].values).to('cpu')
    data[i]['actions'] = torch.Tensor(traj_i[ac_col].values).to('cpu').long()    ←
    data[i]['rewards'] = torch.Tensor(traj_i[rew_col].values).to('cpu')
    ...
```

Tang et al. (2020)

https://github.com/MLD3/RL-Set-Valued-Policy/blob/master/mimic-sepsis/mimic_sepsis_rl/1_preprocess/1_Z_reformat_data.ipynb

```
def make_trajectories(df):
    trajectories = []
    for i, g in tqdm(df.groupby('icustayid')):
        ...
        trajectory = []
        for t in range(len(g)-1):
            transition = {
                's': g.loc[t, state_features].values,
                'a': action_map[
                    int(g.loc[t, 'iv_input']),      ←
                    int(g.loc[t, 'vaso_input'])     ←
                ],
                'r': g.loc[t, 'terminal_reward'],
                's_': g.loc[t+1, state_features].values,
                'a_': action_map[
                    int(g.loc[t+1, 'iv_input']),
                    int(g.loc[t+1, 'vaso_input'])
                ],
                'done': False,
            }
            trajectory.append(transition)
        ...
```

Fatemi et al. (2021)

https://github.com/microsoft/med-deadend/blob/main/utils.py#L111

```
class DataLoader(object):
  ...
  def make_transition_data(self, release=False):
    # DataLoader: making transitions (s,a,r,s')
    ...
    for traj in self.encoded['traj'].keys():
      for t in range(self.encoded['traj'][traj]['actions'].shape[0] - 1):
        self.transition['s'][counter] = self.encoded['traj'][traj]['s'][t, :]
        self.transition['next_s'][counter] = self.encoded['traj'][traj]['s'][t+1, :]
        self.transition['actions'][counter] = self.encoded['traj'][traj]['actions'][t]    ←
        self.transition['rewards'][counter] = self.encoded['traj'][traj]['rewards'][t]
        ...
```

Satija et al. (2021)

https://github.com/hercky/mo-spibb-codebase/blob/neurips/sepsis/1_preprocess/utils.py#L28C1-L50C1

```
def make_trajectories(df):
    trajectories = []
    for i, g in tqdm(df.groupby('icustayid')):
        try:
            g = g.reset_index(drop=True)
            trajectory = []
            for t in range(len(g) - 1):
                transition = {
                    's': g.loc[t, 'state'],
                    'a': action_map[
                        int(g.loc[t, 'iv_input_NEW']),      ←
                        int(g.loc[t, 'vaso_input_NEW'])     ←
                    ],
                    'r': g.loc[t, 'reward'],
                    's_': g.loc[t + 1, 'state'],
                    'a_': action_map[
                        int(g.loc[t + 1, 'iv_input_NEW']),
                        int(g.loc[t + 1, 'vaso_input_NEW'])
                    ],
                    'done': False,
                }
                trajectory.append(transition)
        ...
```

Ji et al. (2021)

https://github.com/clinicalml/trajectory-inspection/blob/main/trajectoryInspection/mimic_utils.py#L257C1-L269C61

```
def get_traj_stats(traj, nact, ncl, death_state_idx, lives_state_idx):
    ########
    # Raw counts of transitions (Action, FromState, ToState)
    ########
    obs_tx_cts_unadjusted = np.zeros((nact, ncl+2, ncl+2))

    for index, row in traj.iterrows():
        # NOTE: Everything is 1-indexed in matlab, but 0-indexed in numpy...
        assert row['action_idx'] >= 0
        obs_tx_cts_unadjusted[int(row['action_idx']),     ←
                              int(row['from_state_idx']),
                              int(row['to_state_idx'])] += 1
    ...
```

Liang et al. (2023)

https://github.com/DMU-XMU/Episodic-Memory-assisted-Approach-for-Sepsis-Treatment/blob/main/continuous/D3QN.py#L137

```
def process_train_batch(size):
    ...
    for i in a.index:
        cur_state = a.loc[i, state_features]
        iv = int(a.loc[i, 'iv_input'])      ←
        vaso = int(a.loc[i, 'vaso_input'])  ←
        action = action_map[iv, vaso]
        reward = a.loc[i, 'reward']
    ...
```

Choudhary et al. (2024)

https://github.com/icu-sepsis/icu-sepsis/blob/main/packages/icu_sepsis_helpers/icu_sepsis_helpers/mdp_creation/create_matrices.py#L6

```
def rl_table_to_unnormalized_matrices(...):
    row = rl_table.iloc[0, :]
    for i in trange(1, len(rl_table)):
        row_next = rl_table.iloc[i, :]
        b, s, a = row['bloc'], row['state'], row['action']    ←
        s_, b_ = row_next['state'], row_next['bloc']
        expert_policy[s, a] += 1
        ...
        # one step in the episode
        if b_ == b+1:
            ...
            tx_mat[s, a, s_] += 1
        else:
            ...
```

Tu et al. (2025)

https://github.com/OOPSDINOSAUR/RL_safety_model/blob/main/utils/compute_trajectories_utils.py#L73

This repository was available at the time of writing of this paper but became unavailable when this paper was published.

```
def build_trajectories(df,state_space,action_space):
    ...
    #iterate through rows which are sorted by charttime at the creation of episode_rows
    tdiff = episode_rows.iloc[0]['charttime']
    for row in range(len(episode_rows)):
        end_index = len(episode_rows) - 1

        #get the action.py, state, reward, next state,
        #and whether or not the sequence is done in the current timestep
        state = episode_rows[state_space].iloc[row].values.tolist()
        action = episode_rows[action_space].iloc[row].values.tolist()    ←

        ...
        #add the current time step info to the lists for this episode
        states.append(deepcopy(state))
        actions.append(deepcopy(action))
        rewards.append(deepcopy(reward))
        done_flags.append(deepcopy(dflag))
        ...
```

C.2 Examples for **Shifted**

Tang et al. (2022)

https://github.com/MLD3/OfflineRL_FactoredActions/blob/main/RL_mimic_sepsis/4_BCQ/data.py#L81

```
class SASRBuffer(object):
    ...

    def load(self, filename):
        data = torch.load(filename)
        state, action, reward, not_done, pibs, next_state = [], [], [], [], [], []
        for i in range(len(data['statevecs'])):
            lng = data['lengths'][i]
            state.append(data['statevecs'][i, :lng-1, :])
            action.append(data['actions'][i, 1:lng])    ←
            reward.append(data['rewards'][i, 1:lng])
            not_done.append(data['notdones'][i, 1:lng])
            pibs.append(data['pibs'][i, :lng-1, :])
            next_state.append(data['statevecs'][i, 1:lng, :])
        self.state = torch.cat(state)
        self.action = torch.cat(action).unsqueeze(1)
        self.reward = torch.cat(reward).unsqueeze(1)
        self.not_done = torch.cat(not_done).unsqueeze(1)
        self.pibs = torch.cat(pibs)
        self.next_state = torch.cat(next_state)
```

Unpublished (2022)

https://github.com/cmudig/AI-Clinician-MIMICIV/blob/main/ai_clinician/modeling/models/common.py#L56C1-L65C5

```
def shift_actions(metadata, actions):
    """
    Shifts the actions backward so that each row provides the *next* action for
    the given state. Actions for the last observed state in each trajectory
    are set to -1.
    """
    new_actions = np.concatenate([actions[1:], np.array([-1])])
    new_actions[np.argwhere(metadata[C_BLOC].values == 1).flatten() - 1] = -1
    return new_actions
```

Zhang et al. (2022)

https://github.com/dtak/Decision-Region-for-ICU-Hypotension/blob/main/src/pipeline_mimic.py

This work addresses hypotension instead of sepsis which are two different but related conditions; we include it here for completeness since it also uses the MIMIC-III dataset.

```
def create_data(pid_list):
    '''Create dataset based on pid input'''
    orig_pid, hr, R = [], [], []
    S1, S2, Y = [], [], []
    #S1_mod, S2_mod = [], []
    for pid in pid_list:
        orig_pid = orig_pid + [pid] * (len(states[pid]) - 1)
        hr.append(states[pid].values[:-1, 0])
        if include_action_states:
            S1.append(np.hstack([states[pid].values[:-1], actions[pid].values[:-1, 1:7]]))
            S2.append(np.hstack([states[pid].values[1:], actions[pid].values[1:, 1:7]]))
        else:
            S1.append(states[pid].values[:-1])
            S2.append(states[pid].values[1:])
        #S1_mod.append(states_mod[p].values[:-1, :])
        #S2_mod.append(states_mod[p].values[1:, :])
        Y.append(actions[pid].values[1:, -3])    ←
        R.append(rewards[pid].values[:, 1])
    return np.concatenate(S1, axis=0), np.concatenate(S2, axis=0), \
            np.concatenate(Y, axis=0), np.concatenate(R, axis=0), \
            np.concatenate(hr, axis=0), np.array(orig_pid)
```