 doi:10.20944/preprints202206.0028.v1

Article

Designing Reinforcement Learning Algorithms for Digital Interventions: Pre-implementation Guidelines

Anna L. Trella [1,*], Kelly W. Zhang [1], Inbal Nahum-Shani[2], Vivek Shetty[3], Finale Doshi-Velez[1], and Susan A. Murphy [1]

[1] School of Engineering and Applied Sciences, Harvard University, 02420 Massachusetts, USA; kellywzhang@seas.harvard.edu (K.W.Z.); finale@seas.harvard.edu (F.D.-V.); samurphy@fas.harvard.edu (S.A.M.)
[2] Institute for Social Research, University of Michigan, 48109 Michigan, USA; inbal@umich.edu
[3] Schools of Dentistry & Engineering, University of California, Los Angeles, 90095 California, USA; vshetty@ucla.edu
* Correspondence: annatrella@g.harvard.edu

Abstract: Online reinforcement learning (RL) algorithms are increasingly used to personalize digital interventions in the fields of mobile health and online education. Common challenges in designing and testing an RL algorithm in these settings include ensuring the RL algorithm can learn and run stably under real-time constraints, and accounting for the complexity of the environment, e.g., a lack of accurate mechanistic models for the user dynamics. To guide how one can tackle these challenges, we extend the PCS (Predictability, Computability, Stability) framework, a data science framework that incorporates best practices from machine learning and statistics in supervised learning [1], to the design of RL algorithms for the digital interventions setting. Further, we provide guidelines on how to design simulation environments, a crucial tool for evaluating RL candidate algorithms using the PCS framework. We illustrate the use of the PCS framework for designing an RL algorithm for Oralytics, a mobile health study aiming to improve users' tooth-brushing behaviors through the personalized delivery of intervention messages. Oralytics will go into the field in late 2022.

Keywords: reinforcement learning (RL); online learning; mobile health; algorithm design; algorithm evaluation; decision support systems

1. Introduction

There is growing interest in using online Reinforcement Learning (RL) to optimize the delivery of messages or other forms of prompts in digital interventions. In mobile health, RL algorithms have been used to increase the effectiveness of the content and timing of intervention messages designed to promote physical activity [2–4] or to manage weight loss [5]. In other areas including the social sciences and education, RL algorithms are used to provide pre-trial nudges to encourage court hearing attendance [6], to personalize math explanations [7] or quiz questions during lecture videos [8]. Unlike areas such as games and some subareas of robotics, it can be extremely costly to run a digital intervention study. Further, when the study is a pre-registered clinical trial, once initiated, the trial protocol (including any online algorithms) cannot be altered, without jeopardizing trial validity. Thus design decisions are a "one-way door" [9]; namely, once we commit to a set of design decisions, they are irreversible for the duration of the trial. To prevent poor design decisions that could be detrimental to the effectiveness and the validity of study results, RL algorithms must undergo a thorough design and testing process before deployment.

The development of an RL algorithm in digital interventions requires a multitude of design decisions. These decisions include how best to accommodate the lack of mechanistic models for dynamic human responses to digital interventions and how to ensure robustness of the algorithm to potentially non-stationary/non-Markovian outcome distributions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

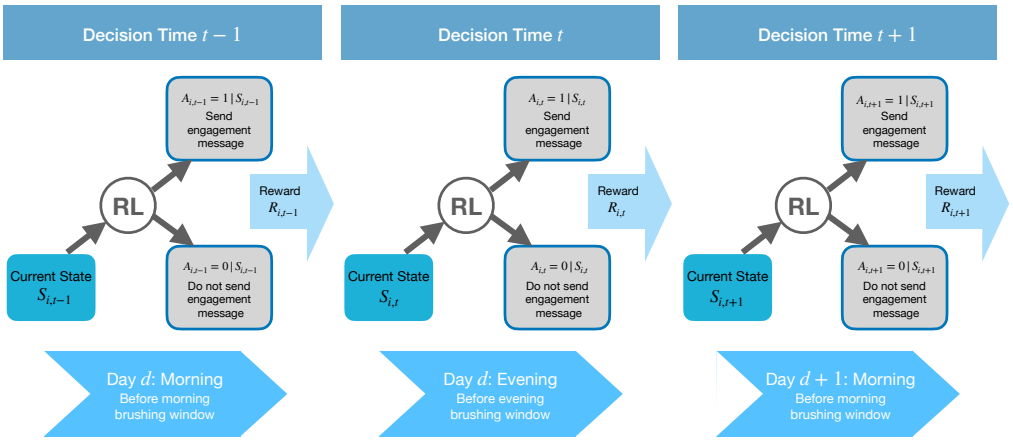

Figure 1. Decision times and actions for the RL algorithm in the Oralytics study. For each day d, there is a morning decision time and an evening decision time for user i. At every decision time t, the RL algorithm selects action $A_{i,t}$ given the current state $S_{i,t}$. After an action is taken, we observe reward $R_{i,t}$ from the user.

Further, one must not only ensure that the RL algorithm learns and quickly optimizes interventions, but also ensure that the algorithm runs stably and autonomously online within constrained amounts of time. These include decisions to ensure that the RL algorithm can obtain data in a timely manner. Time and budgetary considerations may restrict the ability to implement more complex RL algorithms. Further, it is important to ensure the data collected by the RL algorithm can be used to inform future studies and address scientific, causal inference questions. Addressing these challenges in a reproducible, replicable manner is critical if RL algorithms are to play a role in optimizing digital interventions. Therefore we need a framework for making design decisions for RL algorithms intended to optimize digital interventions.

The primary contributions of this work are two-fold:

1. **Framework for Guiding Design Decisions in RL Algorithms for Digital Interventions:** *We provide a framework for evaluating the design of an online RL algorithm to increase confidence that the inclusion of an online RL algorithm as part of the digital intervention will improve the effectiveness and maintain reproducibility and replicability of the intervention, in real-life implementation.* Specifically, we extend the PCS (Predictability, Computability, Stability) data science framework of Yu [1] to address the specific challenges in the development and evaluation of an online RL algorithm for personalizing digital interventions.
2. **Case Study:** This case study concerns the development of an RL algorithm for Oralytics, a mobile health intervention study designed to encourage oral self-care behaviors. The study is planned to go into the field in late 2022. *This case study provides a concrete implementation of the PCS framework in informing the design of an online RL algorithm.* One tool is the development of multiple simulation test-bed environments and the use of these environments along with PCS principles to evaluate the candidate RL algorithms.

2. Review of Online Reinforcement Learning Algorithms

Reinforcement learning (RL) [10,11] is the area of machine learning which is concerned with learning how to best make a sequence of decisions. In digital intervention settings, the sequence of decisions concerns which treatment (e.g., motivational messages, reminders, types of feedback, etc.) to provide given the user's current state and history. Definitions of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

decision times, state, action, reward, and update times are provided below. The decision times and actions for the RL algorithm for Oralytics are provided in Figure 1.

2.1. Decision Times

These are the times, indexed by t at which the RL algorithm may deliver a treatment (via a smart device such as a desktop computer, smartwatch, smartphone, smart speaker, wearable, etc.). The cadence of the decision times (minute level, hourly, daily, etc.) depends on the type of digital intervention. For example in Oralytics, we have two decision times per day, namely, one hour prior to the set morning and evening brushing windows specified by the user.

2.2. State

$S_{i,t} \in \mathbb{R}^d$ represents the ith user's state at time t. d is the number of features describing the user's state (e.g. current location, recent adherence to medication, current social setting, recent engagement with the intervention application, etc.). See Section 5.2 for the state definition for Oralytics.

2.3. Action

$A_{i,t} \in \mathcal{A}$ represents the decision made by the RL algorithm for the ith user at decision time t. Treatment actions in digital interventions frequently include the action of not delivering any treatment at time t. For Oralytics, the action space is $\mathcal{A} := \{0, 1\}$, where $A_{i,t} = 1$ represents sending the user an engagement message and $A_{i,t} = 0$ represents not sending an engagement message. See Section 5.1 for descriptions of the types of messages that could be sent in Oralytics.

2.4. Reward

$R_{i,t} \in \mathbb{R}$ is the reward for the ith user at decision time t, recorded after taking action $A_{i,t}$. The definition of the reward depends on the type of digital intervention. Examples include successful completion of a math problem, taking a medication, level of physical activity, etc. The reward function is the mean of $R_{i,t}$ conditional on the current state $S_{i,t}$ and action $A_{i,t}$. In Oralytics the reward is the subsequent brushing duration; see Section 5.2 for further discussion of the reward in Oralytics.

2.5. Update Times

These are the times at which the RL algorithm is updated, which typically includes updating a model of the user (e.g., a model of the user's reward function). The RL algorithm updates using user i's current history of past states, actions, and rewards up to time t, denoted $H_{i,t-1} = \{S_{i,s}, A_{i,s}, R_{i,s}\}_{s=1}^{t-1}$. Also, if the algorithm pools data across users, then the history of other users in the study, $H_{j,t-1}$ for $i \neq j$, are used to update the model for user i. For Oralytics, the update cadence is once a week.

Generally, online RL algorithms are composed of two parts: (a) fitting a model of the user and (b) an action selection strategy. The simplest type of user model is a model for the reward function, $\mathbb{E}[R_{i,t}|S_{i,t}, A_{i,t}]$. In more general cases, a model for the sum of future rewards, conditional on current state, $S_{i,t}$ and action, $A_{i,t}$ is also learned. The action selection strategy of the RL algorithm uses the user's current state $S_{i,t}$, along with the learned user model, and outputs the treatment action $A_{i,t}$ at each decision time t. The user model is periodically updated using newly collected data $H_{i,t-1}$; these updates can occur after each decision time or at longer time scales. For example in [2], the decision times are 5 times per day, but the update times are only nightly.

An online RL algorithm should quickly learn which action to deliver in which states for each user. One of the most widely used and simplest RL algorithms is a contextual bandit algorithm [12–14]. A contextual bandit algorithm, incrementally, as data accrues,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

learns the action that will lead to maximal reward in each state. The online algorithm sequentially updates an estimate of the reward function (mean of the reward conditional on state and action) and selects actions. The performance of the algorithm is often measured by the sum of rewards; the faster the algorithm learns the higher the sum.

3. PCS Framework For Designing RL Algorithms for Digital Intervention Development

The PCS Framework [1] incorporates best practices from machine learning and applied statistics to provide a set of guidelines for assessing the quality of a prediction algorithm when used to address problems in real life. The goal is to enhance the trust of the scientific community in the application of the prediction algorithm in terms of predictability of results, computability in the implementation of the algorithm, and stability in the performance results of the learning algorithm across perturbations. PCS has been adopted and extended to other domains; see Section 4 for a further discussion.

Similar to the prediction setting, there are a variety of design decisions one needs to make before deploying an RL algorithm, e.g., choosing the model class to use to approximate the reward function. While many of the original PCS principles can be used in the development and evaluation of RL algorithms, RL algorithm development also introduces new challenges for the PCS framework, particularly in the online digital intervention setting. First, the main task is not prediction, but rather in RL, the main goal is to select intervention actions so that average rewards across time are maximized for each user. We call this the goal of Personalization [15]. We generalize the PCS framework to include an evaluation of the ability of an online RL algorithm to personalize. Second, in digital intervention settings, it is important to evaluate the ability of the online RL algorithm to maximize rewards under real-world constraints. For example, there are often time constraints on computations, budgetary constraints on software engineering development, and constraints on the algorithm in terms of obtaining data in real-time. Further, the algorithm must run stably online without constant human monitoring and adjustment. The current PCS framework does not provide evaluation tools that deal with the above needs. We extend the PCS framework to the context of designing and evaluating online RL algorithms. This framework focuses on providing confidence that the online RL algorithm will lead to greater effectiveness under real-world constraints and with stability.

3.1. Personalization (P)

The PCS Framework uses (P) for predictability with the goal of ensuring that models used in data science have good predictive accuracy on both seen and unseen data. Predictive accuracy is a simple, commonly used metric for evaluating the model, but in some cases, multiple evaluation metrics or domain-specific metrics are appropriate. In our setting, the main task is Personalization. Namely, the online RL algorithm should learn to select actions to maximize *each* user's average rewards. Instead of defining a predictive accuracy metric, we want a metric to validate the extent of personalization. For example, when choosing a metric to evaluate RL algorithms for multiple users, one may be interested in not just the average over the users' sums of rewards, but other metrics that capture the variation in the sum of rewards across users. Let N be the total number of users with T total decision times. We suggest the following metrics:

- **Average of Users' Average (Across Time) Rewards:** First average users' rewards across time, and then average across users. This metric is defined as $\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T}R_{i,t}\right)$. The metric serves as a global measure of the RL algorithm's performance.
- **The 25th Percentile of Users' Average (Across Time) Rewards:** The metric shows how well an RL algorithm performs for users that don't benefit as much, namely users in the lower quartile of average rewards across time.
- **Average Reward For Multiple Time-Points:** Average users' rewards across time for multiple time-points $t_0 = 1, 2, ..., T$. Then average across users. This metric is defined

 doi:10.20944/preprints202206.0028.v1

as $\frac{\sum_{i=1}^{N}\sum_{t=1}^{t_0} R_{i,t}}{N \cdot t_0}$. The metric assesses the speed at which the RL algorithm learns across weeks in the trial.

3.2. Computability (C)

Computability focuses on the efficiency and scalability of algorithms, decisions, and processes. While the original PCS framework focused on the computability of training and evaluating models, we also consider the ability to implement the algorithm within the constraints of the study. In the online RL setting, computability encompasses all issues related to ensuring that the RL algorithm can select actions and update in a timely manner while running online. The performance of the online RL algorithm must be evaluated under the constraints of the study; key RL design constraints that could arise include:

- **Timely Access to Reward and State Information:** Since RL algorithms for digital interventions must make decisions in an online fashion, the development team must ensure that the state feature data is available at each decision time and that both the reward and state feature data is available to the algorithm at the update times. For example, due to delays in communication between sensors, the digital application, and the cloud storage, the algorithm may not have timely access to the investigators' first choice of reward, necessitating the use of an alternate.
- **Engineering Budget:** One should consider the engineering budget, supporting software needed, and time available to deliver a production-ready algorithm. In this case, a simpler algorithm may be preferred over a sophisticated one because it is easier to implement, test, and set up monitoring systems for.
- **Off-Policy Evaluation and Causal Inference Considerations:** The investigative team often not only cares about the RL algorithm's ability to learn but also the ability to use the data collected by the RL algorithm to answer scientific questions after the study is over. These scientific questions can include topics such as off-policy evaluation [16,17] and causal inference [18,19]. Thus, the algorithm may be constrained to select actions probabilistically with probabilities that are bounded away from zero and one. This enhances the ability of investigators to use the resulting data to address scientific questions with sufficient power [20].

3.3. Stability (S)

Stability measures how the results change with minor perturbations and also emphasizes the need for documentation and reproducibility of results. In online RL, stability plays two roles. First, the RL algorithm must run stably and automatically without the need for constant human monitoring and adjustment. This is particularly critical as users abandon digital interventions that have inconsistent functionality (unstable RL algorithm) [21,22]. Second, the RL algorithm should perform well across a variety of potential real-world environments. A critical tool in assessing stability to perturbations of the environment is the use of simulation test-beds. Test-beds include a variety of plausible environmental variants, each of which encodes different concerns of the investigative team. Often important state features of the user's environment are not observed and we lack an accurate mechanistic model of user behavior. The following challenges in digital intervention problems are concerns that one could design testbeds for:

- **User Heterogeneity:** There is likely some amount of user heterogeneity in response to actions, even when users are in the same context. User heterogeneity can be partially due to unobserved user traits (e.g. factors that are stable or change very slowly over time like family composition or personality type). The amount of between user heterogeneity impacts whether an RL algorithm that pools data (partially or using clusters) across users to select actions will lead to improved rewards.
- **Non-Stationarity:** Unobserved factors common to all users such as societal changes (e.g. new wave of the pandemic), and time-varying unobserved treatment burden (e.g. user's response to intervention may depend on how many days they have experienced

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

the digital intervention) may make the distribution of the reward appear to vary with time, i.e., non-stationary.

- **High Noise Environments** Digital interventions typically deliver treatments to users in highly noisy environments. This is in part because digital interventions deliver treatments to users in daily life, where many unobserved factors (e.g. social context, mood, or stress) can affect a user's responsiveness to an intervention. If unobserved, these factors produce noise. Moreover, the effect of digital prompts on a near-term reward tends to be small due to the nature of the intervention. Therefore it is important to evaluate the algorithm's ability to personalize even in highly noisy, low signal-to-noise ratio environments.

3.4. Simulation Environments for PCS Evaluation

To utilize the PCS framework, we advocate using a simulation environment for designing and evaluating RL algorithms. We aim to compare RL algorithm candidates under real-world constraints (Computability). Thus, we build multiple variants of the simulation environment, each reflecting plausible user dynamics (i.e. state transitions and reward distributions) (Stability). This is followed by simulating a digital intervention study for each simulation environment variant and RL algorithm candidate pairing. Finally, we use multiple metrics to evaluate the performance of the RL algorithm candidates (Personalization).

In the case of digital interventions, there is often no mechanistic model or physical process for user behavioral dynamics, which makes it difficult to accurately model transitions (e.g. modeling a user's future level of physical activity as a function of their past physical activity, location, local weather). Note that the goal of developing the simulators is not to conduct model-based RL [23]. Rather, here the simulators represent a variety of plausible environments to facilitate the evaluation of the performance of potential RL algorithms in terms of Personalization, Computability, and Stability across these environments. Existing data and domain expertise is most naturally used to construct the simulation environments. However, as is the case for Oralytics, the previously collected data may be scarce, i.e., we have very few data points per user. Moreover, the data may only be partially informative, e.g., the data was collected under only a subset of the actions. Next, we provide guidelines for how to build an environment simulator in such challenging settings.

Base Environment Simulator: In order to have the best chance possible of accurately evaluating how well different RL candidates will perform, we recommend first building a base environment simulator that mimics the existing data to the greatest extent possible. This involves carefully choosing the set of time-varying features and reward generating model class that will be expressive enough to model the true reward distribution well. To check how well simulated data generated by the model of the environment mimics the observed data, we recommend a variety of ways to compare distributions. This includes visual comparisons such as plotting histograms, comparing moments such as mean reward, between user variance, and within user variance, of the real data with the simulated data, and measuring how well the base model captured the variance in the data. Examples of these checks done for Oralytics are in Appendix A.4.

Variant Environment Simulators: We recommend considering many variants or perturbed simulation environments to evaluate the stability of RL algorithms across multiple plausible environments. These variants can be used to evaluate the concerns of the investigative team. For example, if the base simulator generates stationary rewards and the investigative team is concerned that the real reward distribution could not be stationary, a variant could incorporate non-stationarity into the environment dynamics.

In the case that the previously collected data does not include particular actions, as was the case for Oralytics, we recommend consulting domain experts to recommend a range of potential realistic effect sizes (differences in mean reward under the new action versus a baseline action). For example, in Oralytics we only have data under no intervention and did not have data on rewards under the intervention. Thus, using the input of the domain

 doi:10.20944/preprints202206.0028.v1

experts on the team, we imputed several plausible treatment effects (varying by certain state features and the amount of heterogeneity in treatment effects across users).

4. Related Works

4.1. Digital Intervention Case Studies

Liao [2] describe the development of an online RL algorithm for HeartSteps V2, a physical activity mobile health application. They highlight how their design decisions address specific challenges in designing RL algorithms, such as adjusting for longer-term effects of the current action and accommodating noisy data. However, they do not provide general guidelines for how to make design decisions for RL algorithms in digital intervention development.

Another related work is that of Figueroa [24] who provide an in-depth case study of the design decisions, and the associated challenges and considerations, for an RL algorithm for text messaging in a physical activity mobile application serving patients with diabetes and depression. Through this case study, they provide guidelines to others developing RL algorithms for mobile health applications. Specifically, they first categorize the challenges they faced into 3 major themes: 1) choosing a model for decision making, 2) data handling, and 3) weighing algorithm performance versus effectiveness in the real world. Then for each of these challenges, they describe how they dealt with the challenges in the design process for their physical activity intervention. In contrast, by expanding the PCS framework, this work introduces general guidelines for comprehensively evaluating RL algorithms. Moreover, we make recommendations for how to design a variety of simulation testbeds even using only sparse and partially informative existing data in service of PCS. The generality of the PCS framework allows it to be more widely applicable. For example, Figueroa [24] have an existing dataset for all actions, which makes their recommendations less applicable to those designing algorithms with existing data only under a subset of actions. The PCS framework provides more holistic guidelines for facing challenges in developing simulation environments and evaluating algorithms, beyond those faced in any particular case study.

4.2. Simulation Environments in Reinforcement Learning

In RL, simulators (generative models) may be used to derive a policy from the generative model underlying the simulator (model-based learning). Agarwal [23] use simulation as an intermediate step to learn personalized policies in a data-sparse regime with heterogeneous users, where they only observe a single trajectory per user. Wei [25] propose a framework for simulating in a data-sparse setting by using imitation learning to better interpolate traffic trajectories in an autonomous driving setting. In contrast, in PCS, the simulator is used as a crucial tool for using the framework to design, compare, and evaluate RL algorithm candidates for use in a particular problem setting.

There exist many resources aimed at improving the design and evaluation of RL algorithms through simulation; however, in constrast to this work, they do not provide guidelines for designing plausible simulation environments using existing data. RecSim [26] gives a general framework but does not advise on the quality of the environment nor how to make critical design decisions such as reward construction, defining the state space, simulating unobserved actions, etc. MARS-Gym [27] provides a full end-to-end pipeline process (data processing, model design, optimization, evaluation) and open-source code for a simulation environment for marketplace recommender systems. OpenAI Gym [28] is a collection of benchmark environments in classical control, games, and robotics where the public can run and compare the performance of RL algorithms.

There are also a handful of papers that build simulation environment testbeds using real data. Wang [29] evaluate their algorithm for promoting running activity with a simulation environment built using two datasets. Singh [30] develop a simulation environment using movie recommendations to evaluate their safe RL approach. Korzepa [31] use a simulation environment to guide the design of personalized algorithms that optimize

 doi:10.20944/preprints202206.0028.v1

hearing aid settings. Hassouni [32,33] fit a realistic simulation environment using the US timekeeping research project data. Their simulation environment creates daily schedules of activities for each user (i.e. sleep, work, workout, etc.) where each user is one of many different user profiles (i.e. workaholic, athlete, retiree) for the task of improving physical activity.

4.3. PCS Framework Extensions

The PCS framework has been extended to other learning domains such as causal inference [34], network analysis [35], and evaluating the interpretability of algorithms [36]. Despite the variety of these tasks, they can all be framed as supervised learning problems in batch data settings that can be evaluated in terms of prediction accuracy on a hold-out dataset. PCS has not been extended to provide guidelines for developing an online decision-making algorithm. This extension is needed because of the additional considerations, as discussed above, present in a real-world RL setting. Additionally, while these papers focus on evaluating how well a model accurately predicts the outcome on training and hold-out datasets, our extension includes Personalization for evaluating how well an algorithm personalizes to each user. Dwivedi and Ward [34,35] implement the original Computability principle by considering algorithm and process efficiency and scalability. Margot [36] provides a new principle Simplicity which is based on the sum of the lengths of generated rules. In our case, we extend Computability to include the constraints of the study. Finally, these papers consider the stability of results across different changes to the data (e.g. bootstrapping or cross-validation) or design decisions (e.g. choice of representation space or the embedding dimension). Our framework focuses on how stable an algorithm is in plausible real world environments which could be complex (e.g. user-heterogeneity, non-stationary, high noise).

5. Case Study: Oral Health

In this case study, we illustrate the use of PCS principles in designing an RL algorithm for Oralytics. Two main challenges are 1) we do not have timely access to a lot of features and the reward is relatively noisy and 2) we have sparse, partially informative data to inform the construction of our simulation environment testbed. In addition, there are a number of study constraints.

1. Once the study is initiated, the trial protocol and algorithm cannot be altered without jeopardizing trial validity.
2. We are using an online algorithm so we may not have timely access to certain desirable state features or rewards.
3. We have a limited engineering budget.
4. We must answer post-study scientific questions that require causal inference or off-policy evaluation.

We highlight how we handle these challenges by using the PCS framework, despite being in a highly constrained setting. The case study is organized as follows. In Section 5.1 we give background context and motivation for the Oralytics study. In Section 5.2, we explain the Oralytics sequential decision-making problem such as defining the state and reward. In Section 5.3 we describe our process for designing RL algorithm candidates that can stably learn despite having a severely constrained features space and noisy rewards. Finally, in Section 5.4, we describe how we designed the simulation environment variants to evaluate the RL algorithm candidates; furthermore, we offer recommendations for designing realistic environment variants and for constructing such environments using data for only a subset of actions.

5.1. Oralytics

Oralytics is a digital intervention for improving oral health. Each user is provided a commercially-available electric toothbrush with integrated sensors and Bluetooth connectivity as well as the Oralytics mobile application for their smartphone. There are two

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

decision times per day (prior to the user's morning and evening brushing windows) when a message may or may not be delivered to the user via their smartphone. The types of messages focus on winning a gift for oneself, winning a gift for one's favorite charity, feedback on prior brushing, and educational information. Also, once a message is delivered to the user, the app records the messages so that a user is highly unlikely to ever receive the same message twice. Oralytics will be implemented with approximately 70 users in a clinical trial where the participant duration is 10 weeks; this means each user has $T = 140$ decision times. The study duration is 2 years and the expected weekly incremental recruitment rate is around 4 users. The Oralytics mobile app will use an online RL algorithm to optimize message delivery (i.e. treatment actions) to maximize an oral health-related reward (see below). To inform the RL algorithm design, we have access to data from a prior oral health study, ROBAS 2 [37], and input from experts in oral and behavioral health. The ROBAS 2 study used earlier versions of both the electric toothbrush and the Oralytics application to track the brushing behaviors of 32 users over 28 days. Importantly, in ROBAS 2, no intervention messages were sent to the users.

5.2. The Oralytics Sequential Decision Making Problem

We now discuss how we designed the state space and rewards for our RL problem in collaboration with domain experts and the software team while considering various constraints. These decisions must be communicated and agreed upon with the software development team because they provide the RL algorithm with the necessary data at decision and update times, and execute actions selected by the RL algorithm.

1. Choice of Decision Times: We chose the decision times to be prior to each user's specified morning and evening brushing windows, as the scientific team thought this would be the best time to influence users' brushing behavior.

2. Choice of Reward: The research team's first choice of reward was a measure of brushing quality derived from the toothbrush sensor data from each brushing episode. However, the brushing quality outcome is often not reliably obtainable because it requires (1) that the toothbrush dock is plugged in and (2) that the user is standing within a few feet of the toothbrush dock when brushing their teeth. Users could fail to meet these two requirements for a variety of reasons, e.g., the user brushes their teeth in a shared bathroom where they are unable to conveniently leave the dock plugged in. Thus, we selected brushing duration in seconds as the reward (Personalization) since 120 seconds is the dentist-recommended brushing duration and brushing duration is a necessary factor in calculating the brushing quality score. Additionally, brushing duration is expected to be reliably obtainable even when the user is far from the toothbrush dock when brushing (Computability). Note that in Figure 2, a small number of user-brushing episodes have durations over the recommended 120 seconds. Hence we truncate the brushing time to avoid optimizing for over-brushing. Let $D_{i,t}$ denote the user's brushing duration. The reward is defined as $R_{i,t} := \min(D_{i,t}, 180)$.

3. Choice of State Features At Decision Time: To provide the best personalization, it is ideal that an RL algorithm has as many relevant state features as possible to make a decision, e.g., recent brushing, location, user's schedule, etc. However, our choice of the state space is constrained by the need to get features reliably before decision and update times, as well as our limited engineering budget. For example, we originally wanted a feature for the evening decision time to be the morning's brushing outcome, however, this feature may not be accessible in a timely manner. This is because in order for the algorithm to receive the morning brushing data the Oralytics smartphone app requires the user to open the app and we do not expect most users to reliably open the app after every morning brush time before the evening brushing window. A further discussion of our choice of decision time state features can be found in Appendix B.1.

4. Choice of Algorithm Update Times In our simulations, we update the algorithm weekly. In terms of speed of learning (at least in idealized settings), it is best to update the algorithm after each decision time. However, due to Computability considerations, we

 doi:10.20944/preprints202206.0028.v1

chose a slower update cadence. Specifically, for the Oralytics app, the consideration was due to the fact that we are only able to update the policy used to select actions when the user opens the app. This means that if the user does not open the app for many days, even if we try to update the app after each decision time, we would be unable to do so. Since we believe that it could be common that users could fail to open the app for a few days at a time, we chose weekly updates. In the future, we will explore other update cadences as well, e.g., once a day.

5.3. Designing the RL Algorithm Candidates

Here we discuss our use of the PCS framework to guide and evaluate the following design decisions for the RL algorithm candidates. There are some decisions that we've already made and other decisions that we encode as axes for our algorithm candidates to test in the simulation environment. See Appendix B for further details regarding the RL algorithm candidates.

1. Choice of using a Contextual Bandit Algorithm Framework We understand that actions will likely affect a user's future states and rewards, e.g., sending an intervention message the previous day may affect how receptive a user is to an intervention message today. This suggests that an RL algorithm that models a full Markov-Decision process (MDP) may be more suitable than a contextual bandit algorithm. However, the highly noisy environment and the limited data to learn from (140 decision times per user total), makes it difficult for the RL algorithm to accurately model state transitions. Due to errors in the state transition model, the estimates of the delayed effects of actions used in MDP-based RL algorithms can often be highly noisy or inaccurate. This issue is exacerbated by our severely constrained state space (i.e. we have few features and the features we get are relatively noisy). As a result, an RL algorithm that fits a full MDP model may not learn very much during the study, which could compromise Personalization and offer a poor user experience. To mitigate these issues, we use contextual bandit algorithms, which fit a simpler model of the environment. Using a lower discount factor (a form of regularization) has been shown to lead to learning a better policy than using the true discount factor, especially in data-scarce settings [38]. Thus, a contextual bandit algorithm can be interpreted as an extreme form of this regularization where the discount factor is zero. Finally, contextual bandits are the simplest algorithm for sequential decision making (Computability) and have been used to personalize digital interventions in a variety of areas [2,3,7,24,39].

2. Choice of a Bayesian Framework: We consider contextual bandit algorithms that use a Bayesian framework, specifically Posterior (Thompson) Sampling algorithms [40]. Posterior Sampling involves placing a prior on the parameters of the reward approximating function and updating the posterior distribution of the reward function parameters at each algorithm update time. This allows us to incorporate prior data and domain expertise into the initialization of the algorithm parameters. However, these naturally lead to stochastic algorithms (action selections are a not deterministic function of the data), which better facilitates causal inference analyses later on using the data collected in the study.

3. Choice of Constrained Action Selection Probabilities: We constrain the action selection probabilities to be bounded away from zero and one, in order to facilitate off-policy and causal inference analyses once the study is over (Computability). With help from the domain experts, we decided to constrain the action selection probabilities of the algorithm to be in the interval $[0.35, 0.75]$.

The following are decisions we will test using the simulation environment.

4. Choice of the Reward Approximating Function: An important decision in designing the contextual bandit algorithm is how to approximate the reward function. We consider two types of approximations, a Bayesian linear regression model (BLR) and a Bayesian zero-inflated Poisson regression model (ZIP) which are both relatively simple, well-studied, and well-understood. Formal specifications for BLR and ZIP as reward functions can be found in Appendix B.2. We consider the ZIP because of the zero-inflated nature of brushing

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

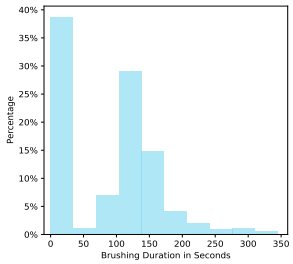

Figure 2. Histogram of brushing durations in seconds for all user brushing sessions (twice a day) in ROBAS 2.

durations on our existing dataset ROBAS 2; see Figure 2. We expected the ZIP to provide a better fit to the reward function by the contextual bandit and thus lead to increased average rewards. Moreover, the linear model for the reward function is easily interpretable by domain experts and allows them to critique and inform the model. To perform posterior sampling, both models are Bayesian with uninformative priors to both models. We discuss using informative priors further in Section 6. From the perspective of Computability and Stability, the posterior for the BLR is of closed-form, which makes it easier to write software leading to efficient and stable updates than the case for the ZIP (the posterior distribution must be approximated and the posterior approximation scheme is another axis to algorithm design that the scientific team needs to consider); see Appendix C for further discussion on how to update the RL algorithm candidates. We design both types of algorithms with weekly updates to the posterior; this decision will be re-examined in the future.

5. Choice of Cluster Size: We consider clustering users with cluster sizes $K = 1$ (no pooling), $K = 4$ (partial pooling), and $K = N$ (full pooling) to determine whether clustering in our setting will lead to higher sums of rewards (Personalization). Clustering-based algorithms pool data from multiple users to learn an algorithm per cluster (i.e. at update times, the algorithm uses $H_{i,t-1}$ for all users i in the same cluster, and at decision times, the same algorithm is used to select actions for all users in the cluster). Clustering-based algorithms have been empirically shown to perform well when users within a cluster are similar [41,42]. In addition, we believe that clustering will facilitate learning within environments that have noisy within user rewards [43,44]. There is a trade-off between no pooling and full pooling. No pooling may learn a policy more specific to the user later on in the study, but may not learn as well earlier in the study when there is not a lot of data for that user. Full pooling may learn well earlier in the study because it can take advantage of all users' data, but may not personalize as well as a no pooling algorithm, especially if users are very heterogeneous. In addition, we chose $K = 4$ because that is the expected weekly recruitment rate for the study and the update cadence is also weekly. We consider the two extremes and partial pooling as a way to explore this trade-off. A further discussion on choices of cluster size can be found in Appendix B.3.

5.4. Designing the Simulation Environment

We build a simulator that considers multiple variants for the environment, each encoding a concern by the research team, and allows us to evaluate the stability of results for each RL algorithm across the environmental variants (Stability).

Fitting Base Models: Recall that the ROBAS 2 study did not involve intervention messages. However, we are still able to use the ROBAS 2 dataset to fit the base model for the simulation environment, i.e., a model for the reward (brushing duration in seconds) under no action. Two main approaches for fitting zero-inflated data are the zero-inflated model and the hurdle model [45]. Throughout the model fitting process, we performed various checks on the quality of the model to determine whether the fitted model was sufficient (Appendix A.4). This included checks regarding whether the percentage number of zero brush times simulated by our model was comparable to that of the original ROBAS

 doi:10.20944/preprints202206.0028.v1

2 dataset. Additionally, we checked whether the model accurately captured the mean and variance of the brushing durations for non-zero brush durations across users.

The first approach we took was to choose one model class (zero-inflated Poisson) and fit a single population-level model for all users in the ROBAS 2 study. However, a single population-level model was insufficient for fitting all users due to the high level of user heterogeneity (i.e. the between and within user variance of the simulated brushing durations from the fitted model was smaller than the between user and within user variance of brushing durations in the ROBAS 2 data). Thus, next, we decided to maintain one model class, but fit one model per user for all users. However, when we fit a zero-inflated Poisson to each user, we noticed that the model provided an adequate fit for some users, but other users exhibited more variability in their brushing durations. Namely, the within user variance simulated rewards from the model fit on those users was still lower than the within user variance of the ROBAS 2 user data used to fit the model. Therefore, we considered a hurdle model [45] because it is more flexible than the zero-inflated Poisson. For Poisson distributions, the mean and variance are equal, whereas in the hurdle model the mean and variance are not conflated.

Ultimately, for each user we considered three model classes: 1) a zero-inflated Poisson, 2) a hurdle model with a square root transform, and 3) a hurdle model with a log transform, and chose one out of these three model classes for each user (Appendix A.2). Specifically, to select the model class for user i, we fit all three model classes using each user's data from ROBAS 2. Then we then chose the model class that had the lowest root mean squared error (Appendix A.3). Additionally, along with the base model that generates stationary rewards, we include an environmental variant with a non-stationary reward function; here day in study is used as a feature in the environment's reward generating model (Appendix A.1).

Imputing Treatment Effect Sizes: To construct a model of rewards for when an intervention message is sent (for which we have no data on), we impute potential treatment effects with the interdisciplinary team and modify the fitted base model with these effects. Specifically, we impute treatment effects on both the user's intent to brush, as well as treatment effects on the brushing duration when the user intends to brush. We impute both types of treatment effects because the investigative team's intervention messages were developed to both encourage users to brush more frequently and to brush for the recommended duration. Further, because the research team believes that the users may respond differently to the engagement messages depending on the context and depending on the user, we included context-aware, population-level, and user-heterogeneous effects of the engagement messages as environmental variants (Appendix A.5).

We use the following guidelines to guide the design of the effect sizes:

1. In general for mobile health digital interventions, we expect the effect (magnitude of weight) of actions to be smaller than (or on the order of) the effect for baseline features, which include time of day and the user's previous day brushing duration (all features are specified in Appendix A.1).
2. The variance in treatment effects (weights representing the effect of actions) across users should be on the order of the variance in the effect of features across users, i.e., looking at variance in parameters of fitted user-specific models.

In accordance with guideline 1 above, to set the population level effect size, we take the absolute value of the weights (excluding that for the intercept term) of the base models fitted for each ROBAS 2 user and the average across users and features (e.g., the average absolute value of weight for time of day and previous day brushing duration). For the heterogeneous (user-specific) effect sizes, for each user, we draw a value from a normal centered at the population effect sizes. In accordance with guideline 2, the variance of the normal distributions is found by again taking the absolute value of the weights of the base models fitted for each user, averaging the weights across features, and taking the empirical variance across users. In total there are eight environment variants as summarized in Table 1. See Appendix A for further details regarding the development of the simulation environments.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

S_Pop: Stationary Base Model, Population Effect Size	**NS_Pop:** Non-Stationary Base Model, Population Effect Sizes
S_Het: Stationary Base Model, Heterogeneous Effect Size	**NS_Het:** Non-Stationary Base Model, Heterogeneous Effect Sizes

Table 1. Four Environment Variants We consider 2 environment base models (stationary and non-stationary) and 2 effect sizes (population effect size, heterogeneous effect size).

RL Algorithm Candidates				
Average Rewards				
RL Algorithm	S_Het	NS_Het	S_Pop	NS_Pop
ZIP $k = 1$	100.462 (0.789)	103.072 (0.751)	107.745 (0.854)	109.892 (0.764)
ZIP $k = 4$	101.096 (0.776)	103.665 (0.781)	108.975 (0.819)	110.921 (0.783)
ZIP $k = N$	101.373 (0.799)	104.042 (0.769)	109.137 (0.822)	111.295 (0.778)
BLR $k = 1$	97.910 (0.779)	100.187 (0.750)	104.279 (0.815)	106.109 (0.752)
BLR $k = 4$	100.376 (0.764)	102.868 (0.761)	108.313 (0.837)	110.100 (0.732)
BLR $k = N$	**101.837 (0.806)**	**104.655 (0.775)**	**109.730 (0.829)**	**112.033 (0.755)**
25th Percentile Rewards				
RL Algorithm	S_Het	NS_Het	S_Pop	NS_Pop
ZIP $k = 1$	69.790 (1.293)	74.240 (0.510)	76.809 (1.193)	79.311 (0.725)
ZIP $k = 4$	70.470 (1.151)	74.199 (0.616)	77.642 (1.415)	81.491 (0.801)
ZIP $k = N$	**71.688 (1.291)**	75.218 (0.648)	78.916 (1.323)	81.194 (0.865)
BLR $k = 1$	67.975 (1.348)	71.263 (0.680)	73.413 (1.105)	75.517 (0.804)
BLR $k = 4$	69.954 (1.226)	73.871 (0.597)	78.316 (1.326)	80.975 (0.755)
BLR $k = N$	71.656 (1.276)	**75.479 (0.520)**	**79.509 (1.332)**	**82.732 (0.732)**

Table 2. Average and 25th Percentile Rewards. Average and 25th percentile rewards are defined in Section 3.1. The naming convention for environment variants is found in Table 1. "BLR" denotes a Bayesian Linear Regression reward approximating function and "ZIP" denotes a Bayesian Zero-Inflated Poisson reward approximating function. "k" refers to the cluster size. Average rewards are averaged across time, users, and 50 trials. For the 25th percentile rewards, we average rewards across time, find the lower 25th percentile across users, and then averaged that across 50 trials. The value in the parenthesis is the standard error of the mean. Best performing algorithm candidates in each environment variant are bolded.

6. Experiment and Results

We evaluate RL candidates in each of the environment variants (Stability). Following Personalization, we compare the RL candidates by their average sum of rewards, by the 25th percentile reward across all users and trials, and by their across time average reward. For each trial, we redraw 72 simulated users (approximately the expected sample size for the Oralytics study) with replacement in groups of 4. To simulate incremental recruitment of 4 users per week, the order in which we randomly select these groups of 4 is their simulated entry date. Every week in a trial, we check whether a group has completed the 10-week study duration and whether we should add another group of 4 users into the study. We cluster users by the day they enter the study (e.g. for cluster size 4 the first four users are in the first cluster and the next four users are in the second cluster, etc). We have one RL algorithm instantiation per cluster (no data shared across clusters) and the RL algorithm has a weekly update cadence with the first update starting after one week (at time $t = 14$).

We then run 50 trials for each environmental variant and algorithm candidate pairing. The results are shown in Table 2. Notice that the average rewards in Table 2 are lower than the 120 seconds of dentist-recommended brushing duration. This is because of the zero-inflated nature of our setting (i.e. the user does not brush). We see in the table that BLR and ZIP with cluster size $k = N$ perform better than other RL algorithm candidates in all environments for average reward and lower 25th percentile reward. This indicates

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

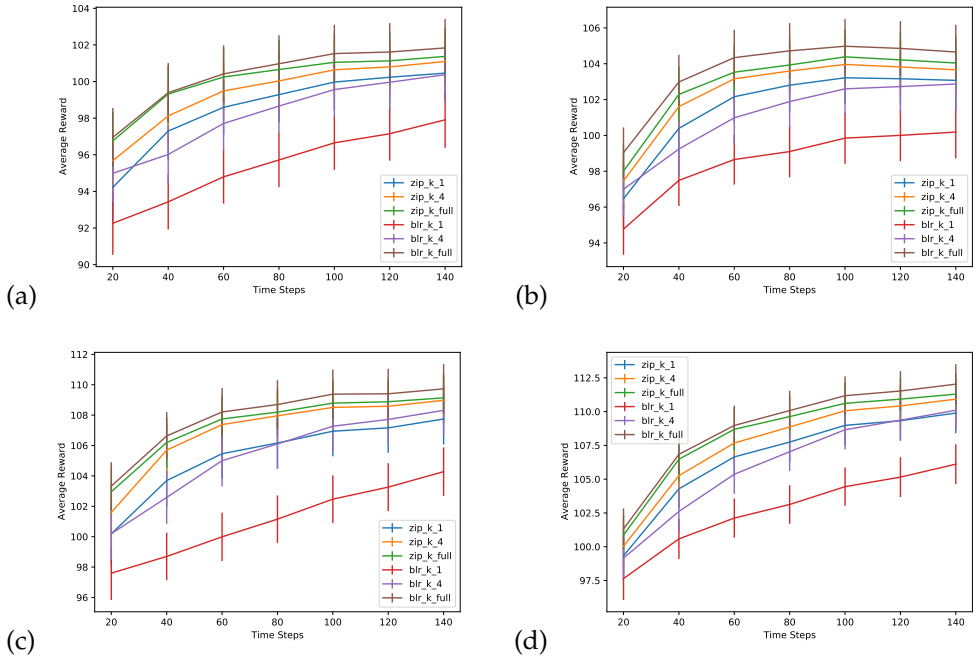

Figure 3. Average User Reward. For each time step $t_0 \in [20, 40, 60, 80, 100, 120, 140]$ figure shows the mean and 1.96 * SE of average user reward $(\frac{1}{n}\sum_{i=1}^{n}\frac{1}{t_0}\sum_{s=1}^{t_0} R_{i,s})$ on average across trials. For **(a)** Stationary Base Model and Heterogeneous Effect Size, **(b)** Non-Stationary Base Model and Heterogeneous Effect Size, **(c)** Stationary Base Model and Population Effect Size, **(d)** Non-Stationary Base Model and Population Effect Size

that even though the users are heterogeneous, the heterogeneity is not so large that pooling users' data is ever detrimental.

Next, consider Figure 3. Again, BLR and ZIP with cluster size $k = N$ perform better than other RL algorithm candidates in all environments across all time steps. We expected algorithms' average reward to increase and then decrease for non-stationary environments. This is because the RL algorithms do not incorporate non-stationarity in the reward approximating function and the modeled reward functions become increasingly biased with large time-steps $t0$. However, it seems that the effect size we considered was large enough to overcome this non-stationarity.

From Table 2 and Figure 3, BLR with cluster sizes $k = 4, N$ is comparable to ZIP. Our hypothesis was that ZIP would perform better than BLR because ZIP has more complexity than BLR. However, there is a bias-variance tradeoff between ZIP and BLR. Although ZIP has more complexity than BLR, ZIP needs to fit more parameters and therefore requires more data to learn effectively; but we are in a data-sparse setting. In addition, ZIP is constrained by the performance of the approximate posterior sampling scheme, which may be unstable and difficult to debug. On the other hand, BLR has a stable closed-form posterior update and BLR with action centering specifically does not require the knowledge of the baseline features at decision time (See Appendix C.1.1). This means that baseline features only need to be available at update time and we can incorporate more features that were not available in real-time at decision time. Also, BLR with action centering is very robust, namely, it is guaranteed to be unbiased even when the baseline reward model is incorrect [2].

We highlight the following takeaways from our experiments:

1. **BLR vs ZIP:** We prefer BLR over ZIP. BLR with pooling performs similarly or better than ZIP and is more reliable/computable.
2. **Cluster Size:** Large cluster sizes perform better especially when there are population treatment effects. They also perform well not only for the average user but also for

 doi:10.20944/preprints202206.0028.v1

users who do not benefit as much. This performance could be due to faster learning because these algorithms can leverage other users' data to make decisions. Even under heterogeneous treatment effects, the benefit of reducing variance by learning across users is not outweighed by user heterogeneity.

There are other limitations to these experiments.

Fixed Noise Variance for BLR: The experiments were run using a fixed noise variance term fit using the ROBAS 2 data set. Assuming a fixed noise variance is unrealistic; we plan to learn the noise variance along with other parameters in the real study. The fixed noise variance term could be a reason that BLR performed comparably to ZIP.

More Distinct and Complex Simulation Environments: We may not be looking widely enough across variants to find settings where these algorithms perform differently. With sufficient data per user in a highly heterogeneous user environment, then we expect cluster size $k = 1$ to do the best. In future work, we aim to add in simulation environments with greater heterogeneity and less noise to see if large cluster sizes still perform well, and we aim to create more complex simulation environment variants that are more distinct (e.g. environments where users may differ by heterogeneous demographic features like age and gender). Also, we want to impute state features of interest in the real study that were not present in the data set, such as phone engagement.

Additional RL Algorithm Candidate Considerations. We also aim to consider other axes for algorithm candidates such as algorithms with other update cadences (e.g. every night or biweekly) and algorithms with an informative prior. In initial simulations using algorithms with informative priors, we found that since the same (limited amount) of ROBAS 2 data was used to build both the simulation environment and the prior, the algorithms did not need to learn much to perform well. An open question is how to develop both simulation environments and informative priors in a realistic way using a very limited amount of data. Finally, we will also explore additional design decisions such as how to carefully design the baseline and advantage feature spaces for the RL algorithm.

These investigations will determine the final algorithm that will go into the actual study.

7. Discussion and Future Work

In this paper, we present the first extension of the PCS framework for designing RL algorithms in digital intervention settings. The case study demonstrates how to use the PCS framework to make design decisions and only highlights of our ongoing work in designing the Oralytics RL algorithm. Our illustration helps fellow researchers balance and understand the benefits and drawbacks of certain aspects of their RL algorithm for their digital intervention study.

Author Contributions: Conceptualization, A.L.T. and K.W.Z.; Methodology, A.L.T., K.W.Z., and S.A.M.; Software, A.L.T.; Validation, A.L.T., K.W.Z., I.N.-S., V.S., F.D.-V., S.A.M.; Formal Analysis, A.L.T. and K.W.Z.; Investigation, A.L.T.; Resources A.L.T., K.W.Z., I.N.-S., V.S., F.D.-V., S.A.M.; Data Curation, A.L.T. and K.W.Z; Writing—Original Draft Preparation, A.L.T., K.W.Z.; Writing—Review and Editing, A.L.T., K.W.Z., I.N.-S., V.S., F.D.-V., S.A.M.; Visualization, A.L.T.; Supervision, F.D.-V., S.A.M.; Project Administration I.N.-S., V.S., S.A.M.; Funding Acquisition I.N.-S., V.S., F.D.-V., S.A.M. All authors have read and agreed to the submitted version of the manuscript.

Funding: This research was funded by NIH grants IUG3DE028723, P50DA054039, P41EB028242, and R01MH123804. KWZ is also supported by the National Science Foundation grant number NSF CBET–2112085 and by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1745303. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable. Data is de-identified.

Data Availability Statement: The ROBAS 2 training data is available online at https://github.com/ROBAS-UCLA/ROBAS.2/blob/master/inst/extdata/robas_2_data.csv (accessed on 31 May 2022). The source code and all other supplementary resources are available online at https://github.com/StatisticalReinforcementLearningLab/pcs-for-rl.

Acknowledgments: We are grateful for the guidance and support of Dr. Wei Wei Pan, Jiayu Yao, and Doug Ezra Morrison throughout this project.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

RL	Reinforcement Learning
BLR	Bayesian Linear Regression
ZIP	Zero-Inflated Poisson
MDP	Markov-Decision Process

Appendix A discusses the development of the simulation environments.
Appendix B discusses the RL algorithm candidates.
Appendix C discusses RL algorithm posterior updates and action selection via posterior sampling.

Appendix A. Simulation Environments

Appendix A.1. Baseline Feature Space of the Environment Base Models

The ROBAS 2 dataset has a variety of features that we anticipate to be associated with brushing duration. These include the time of day (morning vs evening), weekday vs weekend, and summaries of the user's past brushing behavior. Together with domain experts in behavioral health and dentistry, we chose the following features to use to fit a model of the reward. Recall that the ROBAS 2 dataset only includes data under no brushing, so for now we are only fitting a model for the baseline reward model (i.e., the brushing duration under action $A_{i,t}$). In Appendix A.5 we discuss how to model brushing duration under action 1.

1. **Bias / Intercept Term** $\in \mathbb{R}$
2. **Time of Day (Morning/Evening)** $\in \{0, 1\}$
3. **Prior Day Total Brushing Duration (Normalized)** $\in \mathbb{R}$
4. **Weekend Indicator (Weekday/Weekend)** $\in \{0, 1\}$
5. **Proportion of Non-zero Brushing Sessions Over Past 7 Days** $\in [0, 1]$
6. **Day in Study (Normalized)** $\in [-1, 1]$

We use these features to generate two types of base reward environments (Stationary and Non-Stationary). The **Stationary model of the base environment** uses the state function $g(S_{i,t}) \in \mathbb{R}^5$ that *only includes the first five features above*. The **Non-Stationary model of the base environment** uses state $g(S_{i,t}) \in \mathbb{R}^6$ that corresponds to all of the above features.

Normalization of State Features

We normalize features to ensure that state features are all in a similar range. The Prior Day Total Brushing Duration feature is normalized using z-score normalization (subtract mean and divide by standard deviation) and the Day in Study feature (originally in the range $[1 : 28]$ since the study length of ROBAS 2 is 28) is normalized to be between $[-1, 1]$. Note that when generating rewards, Day in Study was normalized based on Oralytic's anticipated 10 week study length (range is still $[-1, 1]$).

$$\text{Normalized Total Brushing Duration in Seconds} = (\text{Brushing Duration} - 172)/118$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

$$\text{Normalized Day in Study When Fitting Model} = (\text{Day} - 14.5)/13.5$$

$$\text{Normalized Day in Study When Generating Rewards} = (\text{Day} - 35.5)/34.5$$

Appendix A.2. Environment Base Model

We consider three model classes: 1) a zero-inflated Poisson, 2) hurdle model with a square root transform, and 3) hurdle model with a log transform, and choose one out of these three model classes for each user. We define these three model classes below. Additionally, below $g(S)$ is the baseline feature vector of the current state defined in Appendix A.1, $w_{i,b}, w_{i,p}, w_{i,\mu}$ are user-specific weight vectors, $\sigma^2_{i,u}$ is the user specific variance for the normal component, and $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

1) Zero-Inflated Poisson Model for Brushing Duration

$$Z \sim \text{Bernoulli}\left(1 - \text{sigmoid}(g(S)^T w_{i,b})\right)$$

$$Y \sim \text{Poisson}\left(\exp\left(g(S)^T w_{i,p}\right)\right)$$

$$\text{Brushing Duration in Seconds} : D = ZY$$

2) Hurdle Model with Square Root Transform for Brushing Duration

$$Z \sim \text{Bernoulli}\left(1 - \text{sigmoid}\left(g(S)^T w_{i,b}\right)\right)$$

$$Y \sim \mathcal{N}\left(g(S)^T w_{i,\mu}, \sigma^2_{i,u}\right)$$

$$\text{Brushing Duration} : D = ZY^2$$

Note that the non-zero component of this model, Y^2, can also be represented as a constant times a non-central chi-squared, where the non-centrality parameter is the square of the mean of the normal distribution.

3) Hurdle Model with Log Transform for Brushing Duration

$$Z \sim \text{Bernoulli}\left(1 - \sigma\left(g(S)^T w_{i,b}\right)\right)$$

$$Y \sim \text{Lognormal}\left(g(S)^T w_{i,\mu}, \sigma^2_{i,u}\right)$$

$$\text{Brushing Duration} : D = ZY$$

Since we want to simulate brushing duration in seconds, we also round outputs of the hurdle models to the nearest whole integer. Notice that the zero-inflated Poisson model is a mixture model with a latent state. The Bernoulli draw Z is latent and represents the user's intention to brush, and the Poisson models the user's brushing duration when they intend to brush (this is because the brush time can still be zero when the user intends to brush). On the other hand, the hurdle model provides a model for brushing duration conditional on whether the user brushed or not. The Bernoulli draw Z in the hurdle model is observed.

Note that the Hurdle model is used for the simulation environment only and *not* the RL algorithm. The Hurdle model conditions on a collider (e.g. whether the person brushes their teeth), thus potentially leading to causal bias [46,47]. For example, consider an unobserved cause U, intervention A, whether the user brushed or not Z, brushing duration D, and a directed acyclic graph with $A \to Z, U \to Z, U \to D$, and $Z \to D$. Then conditioning on collider Z of treatment opens a pathway from A to D through U [48]. Suppose in reality A only impacts whether the user brushes their teeth but not the duration. Then if we condition on Z to evaluate the impact of A on D, we may erroneously learn that

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

A impacts the duration of brushing. This makes the Hurdle unsuitable as a model for an RL algorithm that aims to learn causal effects.

Appendix A.3. Fitting the Environment Base Models

We use ROBAS 2 data to fit the brushing duration model under action 0 (no message). For all model classes, we fit one model per user. All models were fit using MAP with a prior $w_{i,b}, w_{i,p}, w_{i,\mu} \sim \mathcal{N}(0, I)$ as a form of regularization because we have sparse data for each user. Weights were chosen by running random restarts and selecting the weights with the highest log posterior density.

Fitting Hurdle Models: For fitting hurdle models for user i, we fit the Bernoulli component and the non-zero brushing duration component separately. We use $D_{i,t}$ to denote the ith ROBAS 2 user's brushing duration in seconds at the t time point. Set $Z_{i,t} = 1$ if the original observation $D_{i,t} > 0$ and 0 otherwise. We fit a model for this Bernoulli component. We then fit a model for the normal component to either the square root transform $Y_{i,t} = \sqrt{D_{i,t}}$ or to inverse-log transform $Y_{i,t} = \exp(D_{i,t})$ of the ith user's non-zero brushing duration.

Fitting Zero-Inflated Poisson Models: For the zero-inflated Poisson model, we jointly fit parameters for both the Bernoulli and the Poisson components. Since the brushing durations in the ROBAS 2 data were integer values, we did not have to transform the observation to fit the zero-inflated Poisson model.

Selecting the Model Class For Each User

To select the model class for user i, we fit all three model classes using user i's data from ROBAS 2. We then chose the model class that had the lowest root mean squared error (RMSE). Namely, we choose the model class with the lowest L_i, where:

$$L_i := \sqrt{\sum_{t=1}^{T} (D_{i,t} - \hat{\mathbb{E}}[D_{i,t}|S_{i,t}])^2}$$

Recall that $D_{i,t}$ is the brush time in seconds for user i at decision time t. Definitions of $\hat{\mathbb{E}}[D_{i,t}|S_{i,t}]$ for each model class are specified below in Table A1.

Model Class	$\widehat{\mathbb{E}}[D_{i,t}\|S_{i,t}]$
Zero-Inflated Poisson	$\left[1 - \text{sigmoid}\left(g(S_{i,t})^T w_{i,b}\right)\right] \cdot \exp\left(g(S_{i,t})^T w_{i,p}\right)$
Hurdle (Square Root)	$\left[1 - \text{sigmoid}\left(g(S_{i,t})^T w_{i,b}\right)\right] \cdot \left[\sigma^2_{i,u} + (g(S_{i,t})^T w_{i,\mu})^2\right]$
Hurdle (Log)	$\left[1 - \text{sigmoid}(g(S_{i,t})^T w_{i,b})\right] \cdot \exp\left(g(S_{i,t})^T w_{i,\mu} + \frac{\sigma^2_{i,u}}{2}\right)$

Table A1. Definitions of $\widehat{\mathbb{E}}[D_{i,t}|S_{i,t}]$ for each model class. $\widehat{\mathbb{E}}[D_{i,t}|S_{i,t}]$ is the mean of user model i fitted using data $\{(S_{i,t}, D_{i,t})\}_{t=1}^{T}$.

After the procedure was run, we obtained the following number of model classes for all users in the ROBAS 2 study in Table A3.

Appendix A.4. Checking the Quality of the Simulation Environment Base Model

Checking Moments

Using the chosen user-specific models, we simulate 100 "trials". In each trial, for each of users in ROBAS 2, we use their respective model to generate a data trajectory $(S_{i,t}, R_{i,t})_{t=1}^{56}$ (note that the ROBAS 2 study had 2 brushing windows per day for 28 days for a total of 56 brushing windows). We then compute the following metrics for each of the trials and averaged across trials:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

Model Class	$\widehat{\mathbb{E}}[D_{i,t}\|S_{i,t}, D_{i,t} > 0]$
Hurdle (Square Root)	$\sigma^2_{i,u} + (g(S_{i,t})^T w_{i,\mu})^2$
Hurdle (Log)	$\exp(g(S_{i,t})^T w_{i,\mu} + \frac{\sigma^2_{i,u}}{2})$
Zero-Inflated Poisson	$\frac{\exp(g(S_{i,t})^T w_{i,p}) \exp(\exp(g(S_{i,t})^T w_{i,p}))}{\exp(\exp(g(S_{i,t})^T w_{i,p}))-1}$
	$\widehat{\text{Var}}[D_{i,t}\|S_{i,t}, D_{i,t} > 0]$
Hurdle (Square Root)	$g(S_{i,t})^T w^4_{i,\mu} + 3\sigma^4_{i,u} + 6\sigma^2_{i,u}(g(S_{i,t})^T w_{i,\mu})^2 - \widehat{\mathbb{E}}[R_{i,t}\|S_{i,t}, R_{i,t} > 0]^2$
Hurdle (Log)	$(\exp(\sigma^2_{i,u}) - 1) \cdot \exp(2g(S_{i,t})^T w_{i,\mu} + \sigma^2_{i,u})$
Zero-Inflated Poisson	$\widehat{\mathbb{E}}[D_{i,t}\|S_{i,t}, D_{i,t} > 0] \cdot (1 + \exp(g(S_{i,t})^T w_{i,p}) - \widehat{\mathbb{E}}[D_{i,t}\|S_{i,t}, D_{i,t} > 0])$

Table A2. Definitions of $\widehat{\mathbb{E}}[D_{i,t}|S_{i,t}, D_{i,t} > 0]$ and $\widehat{\text{Var}}[D_{i,t}|S_{i,t}, D_{i,t} > 0]$ for each model class. $\widehat{\mathbb{E}}[D_{i,t}|S_{i,t}, D_{i,t} > 0]$ and $\widehat{\text{Var}}[D_{i,t}|S_{i,t}, D_{i,t} > 0]$ is the mean and variance of the non-zero component of user model i fitted using data $\{(S_{i,t}, D_{i,t})\}_{t=1}^T$.

Model Class	Stationary	Non-Stationary
Hurdle with Square Root Transform	9	7
Hurdle with Log Transform	9	8
Zero-Inflated Poisson	14	17

Table A3. Number of selected model classes for the Stationary and Non-Stationary environments.

1. **Proportion of Missed Brushing Windows:**

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T}\mathbb{I}[D_{i,t} = 0]$$

2. **Average Non-Zero Brushing Duration:**

$$\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sum_{t=1}^{T}\mathbb{I}[D_{i,t} > 0]}\sum_{t=1}^{T}\mathbb{I}[D_{i,t} > 0]D_{i,t}$$

3. **Variance of Non-Zero Brushing Durations:** Let $\widehat{\text{Var}}(\{X_k\}_{k=1}^K)$ represent the empirical variance of $X_1, X_2, ..., X_K$.

$$\widehat{\text{Var}}(\{D_{i,t} : t \in [1\colon T], D_{i,t} > 0\}_{i=1}^N)$$

4. **Variance of Average User Brushing Durations:** This metric measures the degree of between user variance in average brushing.

$$\widehat{\text{Var}}\left(\left\{\frac{1}{T}\sum_{t=1}^{T}D_{i,t}\right\}_{i=1}^{N}\right)$$

5. **Average of Variances of Within User Brushing Durations:** This metric measures the average amount of within user variance.

$$\frac{1}{N}\sum_{i=1}^{N}\widehat{\text{Var}}(\{D_{i,t}\}_{t=1}^T)$$

The base models slightly overestimate the proportion of missed brushing windows in the ROBAS 2 data set. Our base models also slightly underestimate the average brushing duration. Our base models also for the most part slightly overestimate the between-user and within-user variance of rewards.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

Metrics	ROBAS 2	Stationary	Non-Stationary
Proportion of Missed Brushing Windows	0.376674	0.403114	0.397812
Average Non-Zero BDs	137.768129	131.308445	134.676955
Variance of Non-Zero BDs	2326.518304	2392.955018	2253.177853
Variance of Average User BDs	1415.920148	1699.126897	1399.615330
Average of Variances of Within User BDs	1160.723506	1405.944459	1473.239769

Table A4. Comparing Moments Between Base Models and ROBAS 2 Data set. Above we use BDs to abbreviate Brushing Durations. Values for the Stationary and Non-Stationary base models are averaged across 100 trials.

Measuring If A Base Model Captures the Variance in the Data

We measure how well the fitted base models captured (1) whether or not the user brushed, and (2) the variance of the brush time when the users did brush. To measure point (1), for each user model i, we calculate the statistic:

$$U_i := \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\mathbb{I}[D_{i,t}>0]-\widehat{\mathbb{E}}[\mathbb{I}[D_{i,t}>0]|S_{i,t}]}{\widehat{\text{Var}}[\mathbb{I}[D_{i,t}>0]|S_{i,t}]}\right)^2 \tag{A1}$$

where $\widehat{\mathbb{E}}[\mathbb{I}[D_{i,t}>0]|S_{i,t}] = 1-\text{sigmoid}(S_{i,t}^T w_{i,b})$ and $\widehat{\text{Var}}[\mathbb{I}[D_{i,t}>0]|S_{i,t}] = \widehat{\mathbb{E}}[\mathbb{I}[D_{i,t}>0]|S_{i,t}]\cdot\text{sigmoid}(S_{i,t}^T w_{i,b})$.

To measure point (2), for each user model i, we calculate the statistic:

$$U_i := \frac{1}{\sum_{t=1}^{T}\mathbb{I}[D_{i,t}>0]}\sum_{t=1}^{T}\mathbb{I}[D_{i,t}>0]\left(\frac{D_{i,t}-\widehat{\mathbb{E}}[D_{i,t}|S_{i,t},D_{i,t}>0]}{\widehat{\text{Var}}[D_{i,t}|S_{i,t},D_{i,t}>0]}\right)^2 \tag{A2}$$

Definitions of $\widehat{\mathbb{E}}[D_{i,t}|S_{i,t},D_{i,t}>0]$ and $\widehat{\text{Var}}[D_{i,t}|S_{i,t},D_{i,t}>0]$ for the non-zero component of each model class are specified in Table A2. For a user model to capture the variance in the data, U_i should be close to 1. We calculate the empirical mean $\overline{U} = \frac{1}{N}\sum_{i=1}^{N}U_i$ and standard deviation $\overline{\sigma_U} = \text{std}(U_i)$, and the approximate 95% confidence interval is $\overline{U} \pm 1.96 * \frac{\overline{\sigma_U}}{\sqrt{N}}$.

Metric	Stationary	Non-Stationary
Eqn (A1) $\overline{U}$	0.811	0.792
Eqn (A1) $\overline{\sigma_U}$	0.146	0.150
Eqn (A1) Confidence Interval	(0.760, 0.861)	(0.739, 0.844)
Eqn (A2) $\overline{U}$	3.579	3.493
Eqn (A2) $\overline{\sigma_U}$	4.861	4.876
Eqn (A2) Confidence Interval	(1.895, 5.263)	(1.803, 5.182)

Table A5. Statistic U_i for Capturing Variance in the Data. Values are rounded to the nearest 3 decimal places.

Results are in Table A5. We can see that after computing the statistic for each user, the confidence interval is close to 1. We understand that the confidence intervals do not contain 1 which implies that we are overestimating the amount of variance in $\mathbb{I}[D>0]$, and underestimating the amount of variance in $D|D>0$. In the future, we hope to improve upon this statistic by considering non-linear components in our base models.

Appendix A.5. Imputing Treatment Effect Sizes for Simulation Environments

Recall the ROBAS 2 dataset does not have any data under action 1 (send a message). Thus to model the reward under action 1 we must impute.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

Treatment Effect Feature Space

The following choice of the treatment effect (advantage) feature space was made after discussion with domain experts on which features are most likely to interact with the intervention (action). The Stationary model uses state $h(S) \in \mathbb{R}^4$ corresponding to the following features:

1. **Bias / Intercept Term** $\in \mathbb{R}$
2. **Time of Day (Morning/Evening)** $\in \{0, 1\}$
3. **Prior Day Total Brushing Duration (Normalized)** $\in \mathbb{R}$
4. **Weekend Indicator (Weekday/Weekend)** $\in \{0, 1\}$

The Non-Stationary model uses state $h(S) \in \mathbb{R}^5$ corresponding to all of the above features as well as the following:

5. **Day in Study (Normalized)** $\in \mathbb{R}$

Imputation Approach

For the Zero-Inflated Poisson model, we impute treatment effects on both the user's intent to brush (Bernoulli component) and the user's brushing duration when they intend to brush (Poisson component). Similarly, for the Hurdle models, we impute treatment effects on both whether the user's brushing duration is zero (Bernoulli component) and the user's brushing duration when the duration is nonzero.

After incorporating effect sizes, brushing duration under action A in state S is D where:

$$Z \sim \text{Bernoulli}(1 - \text{sigmoid}(g(S)^T w_{i,b} + A * h(S)^T \Delta_{i,B}))$$

$$D = \begin{cases} ZY^2, Y \sim \mathcal{N}(g(x)^T w_{i,\mu} + A * h(x)^T \Delta_{i,N}, \sigma_u^2) & \text{for Hurdle square root model} \\ Z\exp(Y), Y \sim \mathcal{N}(g(x)^T w_{i,\mu} + A * h(x)^T \Delta_{i,N}, \sigma_{i,u}^2) & \text{for Hurdle log normal model} \\ ZY, Y \sim \text{Poisson}(\exp(g(x)^T w_{i,p} + A * h(x)^T \Delta_{i,N})) & \text{for Zero-Inflated Poisson model} \end{cases}$$

$\Delta_{i,B}, \Delta_{i,N}$ are user-specific effect sizes; we will also consider population-level effect sizes (same across all users) which we denote as Δ_B, Δ_N. $g(S)$ is the baseline feature vector as described in Appendix A.1, and $h(S)$ is the feature vector that interacts with the effect size as specified above.

Heterogeneous versus Population-Level Effect Size

We consider realistic heterogeneous effect sizes (each user has a unique effect size) and a realistic population-level effect size (all users who share the same base model class also share the same effect size).

Population Level Effect Sizes: Recall that for the stationary base model we fit models for Y and Z, and get user-specific parameters $w_{i,b}, w_{i,p} \in \mathbb{R}^5$ (Zero-Inflated Poisson model) and parameters $w_{i,b}, w_{i,\mu} \in \mathbb{R}^5$ (Hurdle models). We use these parameters to form the population effect sizes as follows:

Zero-Inflated Models Effect Sizes:

- $\Delta_B = \mu_{B,\text{avg}}$ where $\mu_{B,\text{avg}} = \frac{1}{4} \sum_{d \in [2:5]} \frac{1}{N} \sum_{i=1}^{N} |w_{i,b}^{(d)}|$.
- $\Delta_N = \mu_{N,\text{avg}}$ where $\mu_{N,\text{avg}} = \frac{1}{4} \sum_{d \in [2:5]} \frac{1}{N} \sum_{i=1}^{N} |w_{i,p}^{(d)}|$.

Hurdle Models Effect Sizes:

- $\Delta_B = \mu_{B,\text{avg}}$ where $\mu_{B,\text{avg}} = \frac{1}{4} \sum_{d \in [2:5]} \frac{1}{N} \sum_{i=1}^{N} |w_{i,b}^{(d)}|$.
- $\Delta_N = \mu_{N,\text{avg}}$ where $\mu_{N,\text{avg}} = \frac{1}{4} \sum_{d \in [2:5]} \frac{1}{N} \sum_{i=1}^{N} |w_{i,\mu}^{(d)}|$.

Heterogeneous Effect Sizes: To calculate the heterogeneous effect sizes, we again group users by their chosen base model (Zero-Inflated, Hurdle Square-Root, Hurdle Log).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

We then draw effect sizes for each user from a normal distribution specific to their base model:

$$\Delta_{i,B} \sim \mathcal{N}(\mu_B, \sigma_B^2)$$
$$\Delta_{i,B} \sim \mathcal{N}(\mu_N, \sigma_N^2)$$

μ_B, μ_N are set to the population level effect sizes for that base model class as described above. To set σ_B^2, σ_N^2, we do the following:

Zero-Inflated Models

- σ_B is the empirical standard deviation over $\{\mu_{i,B}\}_{i=1}^N$ where $\mu_{i,B} = \frac{1}{4}\sum_{d\in[2:5]} |w_{i,b}^{(d)}|$. We use $w_{i,b}^{(d)}$ to denote the d^{th} dimension of the vector $w_{i,b}$; we take the minimum over all dimensions excluding $d = 1$ which represents the weight for the bias/intercept term.
- σ_N is the empirical standard deviation over $\{\mu_{i,N}\}_{i=1}^N$ where $\mu_{i,N} = \frac{1}{4}\sum_{d\in[2:5]} |w_{i,p}^{(d)}|$.

Hurdle Models:

- σ_B is the empirical standard deviation over $\{\mu_{i,B}\}_{i=1}^N$ where $\mu_{i,B} = \frac{1}{4}\sum_{d\in[2:5]} |w_{i,b}^{(d)}|$.
- σ_N is the empirical standard deviation over $\{\mu_{i,N}\}_{i=1}^N$ where $\mu_{i,N} = \frac{1}{4}\sum_{d\in[2:5]} |w_{i,\mu}^{(d)}|$.

Histograms of $\Delta_{i,B}, \Delta_{i,N}$ and values of μ_B, μ_N for each base model class are specified in Figure A1.

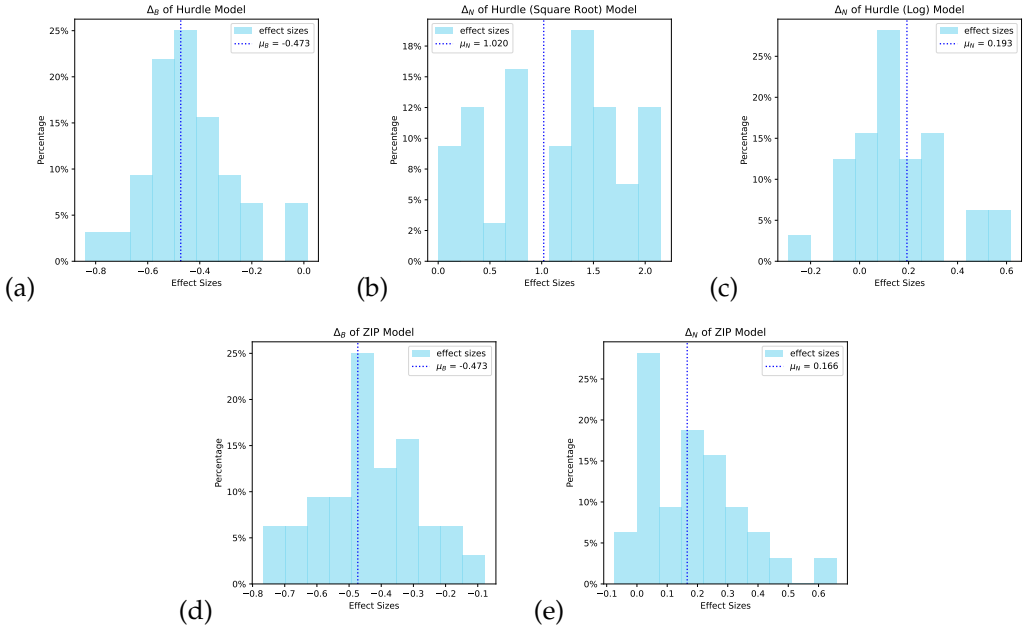

Figure A1. $\Delta_{i,B}$'s, $\Delta_{i,N}$'s, μ_B, μ_N for each base model class. For **(a)** Effect Sizes for Bernoulli Component of the Hurdle Models, **(b)** Effect Sizes for Non-Zero Component of Hurdle (Square Root) Model, **(c)** Effect Sizes for Non-Zero Component of Hurdle (Log) Model, **(d)** Effect Sizes for Bernoulli Component of the ZIP Model, **(e)** Effect Sizes for Poisson Component of the ZIP Model

Appendix B. Reinforcement Learning Algorithm Candidates

Appendix B.1. Feature Space for the RL Algorithm Candidates

Advantage Feature Space

We use $f(S) \in \mathbb{R}^3$ to denote the feature space used by our RL algorithm candidates to predict the advantage, i.e., the immediate treatment effect, is the following:

1. **Bias / Intercept Term** $\in \mathbb{R}$
2. **Time of Day (Morning/Evening)** $\in \{0, 1\}$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

3. **Prior Day Total Brushing Duration (Normalized)** $\in \mathbb{R}$

The normalization procedure for Prior Day Brushing Duration is the same as described in Appendix A.1.0.1.

Baseline Feature Space

We use $m(S) \in \mathbb{R}^4$ to denote the feature space used by the RL algorithm candidates to approximate the baseline reward function containing all the above covariates as well as the following:

4. **Weekend Indicator (Weekday/Weekend)** $\in \{0, 1\}$

Note, *the feature space used by the RL algorithm candidates is different than the feature space used to model the reward in the simulation environments, specified in Appendix A.1* and Appendix A.5; this means that the RL algorithms will have a misspecified reward model. Namely, the baseline feature space for the simulation environment has an additional **Proportion of Non-zero Brushing Sessions Over Past 7 Days** feature and the non-stationary variant has the **Day in Study (Normalized)** feature. The treatment effect feature space for the simulation environment has an additional **Weekend Indicator (Weekday/Weekend)** and the non-stationary variant has the **Day in Study (Normalized)** feature.

The rationale for not including the **Day in Study (Normalized)** feature is although we wanted to capture potential non-stationarity in brushing outcomes in order to create a realistic simulation environment, our RL algorithm candidates do not have reward functions that vary arbitrarily over time. We do not include **Proportion of Non-zero Brushing Sessions Over Past 7 Days** and **Weekend Indicator (Weekday/Weekend)** to detect the robustness of RL algorithm candidates to a misspecified reward model.

Appendix B.2. Decision 1: Reward Approximating Function

The first decision in designing the RL algorithm is the choice between using a linear model or a 0-Inflated Poisson model as the reward approximating function used by the posterior sampler (note this is separate from the reward model used to generate the environment). More information on how the posterior sampling algorithm performs action selection can be found in Appendix C.2, and how the algorithm updates at update times can be found in Appendix C.1. We describe the two candidates below.

Note that the function m for the RL algorithm's baseline reward model is only used at update times. The function f for the RL algorithm's advantage model is used at both decision and update times.

Appendix B.2.1. Bayesian Linear Regression Model

The first candidate is to use the following reward generating model with action centering used in [2] for the posterior sampler:

$$R_{i,t} = m(S_{i,t})^T \alpha_{i,0} + \pi_{i,t} f(S_{i,t})^T \alpha_{i,1} + (A_{i,t} - \pi_{i,t}) f(S_{i,t})^T \beta_i + \epsilon_{i,t} \quad \text{(A3)}$$

where $\alpha_{i,0} \in \mathbb{R}^4$ and $\alpha_{i,1}, \beta_i \in \mathbb{R}^3$. $\pi_{i,t}$ is the probability that action $A_{i,t} = 1$ is selected by the RL algorithm for user i in state $S_{i,t}$; we discuss how to compute these in Appendix C.2. The RL algorithm models $\epsilon_{i,t}$ as being drawn from $\mathcal{N}(0, \eta^2)$ (the choice of η^2 is informed by the ROBAS 2 dataset). Additionally, we put uninformative normal priors on the parameters: $\alpha_{i,0} \sim \mathcal{N}(0, \sigma_{prior} I_4)$, $\alpha_{i,1} \sim \mathcal{N}(0, \sigma_{prior} I_3)$, $\beta_i \sim \mathcal{N}(0, \sigma_{prior} I_3)$, where $\sigma_{prior} = 5$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

Appendix B.2.2. Zero-Inflated Poisson Regression Model

The second candidate is to use the zero-inflated reward generating model for the posterior sampler:

$$\begin{aligned} Z_{i,t} &\sim \text{Bernoulli}\Big(1 - \text{sigmoid}(m(S_{i,t})^T \alpha_{i,p} + A_{i,t} \cdot f(S_{i,t})^T \beta_{i,b})\Big) \\ Y_{i,t} &\sim \text{Poisson}\Big(\exp\Big(m(S_{i,t})^T \alpha_{i,p} + A_{i,t} \cdot f(S_{i,t})^T \beta_{i,p}\Big)\Big) \\ R_{i,t} &= Z_{i,t} Y_{i,t} \end{aligned} \tag{A4}$$

The above model closely resembles the zero-inflated Poisson model class used to develop the simulation environment in Appendix A.2; however, recall that the feature space used by the RL algorithm and the model to generate the environment are different. Additionally, here we model the reward directly, rather than the raw brushing duration.

Additionally, the posterior sampling algorithm will put the following uninformative normal priors on the parameters: $\alpha_{i,b}, \alpha_{i,p} \sim \mathcal{N}(0, \sigma_{prior} I_4)$ and $\beta_{i,b}, \beta_{i,p} \sim \mathcal{N}(0, \sigma_{prior} I_3)$, where $\sigma_{prior} = 5$.

Appendix B.3. Decision 2: Cluster Size

Clustering involves grouping k users together and pooling all of their data together for the RL algorithm. This means that we have one RL algorithm instantiation per cluster (no data shared across clusters). For our experiments, we draw 72 simulated users (the expected sample size for the Oralytics study) with replacement and cluster these users at random (every possible cluster is equally likely). We then keep these cluster assignments fixed across the trials.

For simplicity in running our experiments, we consider randomly formed clusters, but we are thinking of clustering by entry date in the real study. We cannot predict how many users who share the same baseline feature will join within a relatively short period of time (e.g. we cannot depend on there being 4 females within the first two weeks). Entry date is reasonable because we are guaranteed to form a cluster for those users and the domain experts believe that users who enter the study around the same time will be similar. Users who enter near the end of the study may be very different from users who enter near the beginning because of societal factors (e.g. pandemic restrictions being lifted), seasonal influences (e.g. changes in user's mood in spring versus mid-winter), and fidelity (e.g. quality of on-boarding procedures and staff experience may improve over time). One natural approach is to cluster by baseline features, however, that is not feasible for a study where the recruitment rate is slow such as in Oralytics.

Appendix C. RL Algorithm Posterior Updates and Posterior Sampling Action Selection

Appendix C.1. Posterior Updates to the RL Algorithm at Update Time

During the update step, the reward approximating function will update the posterior with newly collected data. Also, we make M draws of the parameters from the updated posterior and use them for all decision times until the next update time. Here are the procedures for how the posterior updates for the Bayesian Linear Regression model and the 0-Inflated Poisson model.

Appendix C.1.1. Bayesian Linear Regression Model

Suppose we are selecting actions for decision time t. Let $\phi(S_{i,t}, A_{i,t}) = [m(S_{i,t}), \pi_{i,t} f(S_{i,t}), (A_{i,t} - \pi_{i,t}) f(S_{i,t})]$ be the joint feature vector and $\theta_i = [\alpha_{0,i}, \alpha_{i,1}, \beta_i]$ be the joint weight vector. Notice that Equation A3 can be vectorized of the form: $R_{i,t} = \phi(S_{i,t}, A_{i,t})^T \theta_i + \epsilon_{i,t}$. Now let $\Phi_{i,1:t-1}$ be the matrix of all stacked vectors $\{\phi(S_{i,s}, A_{i,s})\}_{s=1}^{t-1}$, and $\mathbf{R}_{i,1:t-1}$ be a vector of stacked rewards $\{R_{i,s}\}_{s=1}^{t-1}$, where we have batch data of the $t-1$ decision times before the current update time.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

Recall we have Normal priors on θ_i where $\theta_i \sim \mathcal{N}(\mu_{\text{prior}}, \Sigma_{\text{prior}})$, where $\mu_{\text{prior}} = \mathbf{0} \in \mathbb{R}^{3+3+4}$ and $\Sigma_{\text{prior}} = \text{diag}(\sigma^2_{prior} I_3, \sigma^2_{prior} I_3, \sigma^2_{prior} I_4)$. The posterior distribution of the weights given current history $H_{i,t-1}$, $p(\theta_i | H_{i,t-1})$ is conjugate and is also normal.

$$\theta_i | H_{i,t-1} \sim \mathcal{N}(\mu^{posterior}_{i,t-1}, \Sigma^{posterior}_{i,t-1})$$

$$\Sigma^{posterior}_{i,t-1} = \left(\frac{1}{\eta^2} \Phi^T_{i,1:t-1} \Phi_{i,1:t-1} + \Sigma^{-1}_{prior} \right)^{-1}$$

$$\mu^{posterior}_{i,t-1} = \Sigma^{posterior}_{i,t-1} \left(\frac{1}{\eta^2} \Phi^T_{i,1:t-1} \mathbf{R}_{i,1:t-1} + \Sigma^{-1}_{prior} \mu_{prior} \right)$$

Note, we fit η^2 to the ROBAS 2 dataset and fixed it for all of our experiments. In the real study, we want to consider assigning a conjugate prior on η^2 and updating it at update times.

Appendix C.1.2. Zero-Inflated Poisson Regression Model

For the 0-Inflated Poisson regression model, the posterior distribution of the weights $\theta_i = \{\alpha_{i,b}, \beta_{i,b}, \alpha_{i,p}, \beta_{i,p}\}$ given data $H_{i,t-1}$, $p(\theta_i | H_{i,t-1})$, does not have a closed form. Therefore we use Metropolis Hastings (MH) with a normal proposal distribution as an approximate posterior sampling method.

Posterior Density

The log-likelihood of the 0-Inflated Poisson Regression Model is:

$$\log f(R_{i,t} | S_{i,t}, A_{i,t}; \theta_i) = \begin{cases} \log((1-p) + p \exp(-\lambda)) & R = 0 \\ \log p - \lambda + R \log \lambda - \log R! & R = 1, 2, 3, \ldots \end{cases}$$

where $p = 1 - \text{sigmoid}\left(m(S_{i,t})^T \alpha_{i,b} + A_{i,t} \cdot f(S_{i,t})^T \beta_{i,b}\right)$, is the probability of the user intending to brush, and $\lambda = \exp\left(m(S_{i,t})^T \alpha_{i,p} + A_{i,t} \cdot f(S_{i,t})^T \beta_{i,p}\right)$ is the expected Poisson count.

Therefore the log posterior density is:

$$\log p(\theta_i | H_{i,t-1}) \propto \sum_{n=1}^{N} \log f(R_{i,t} | S_{i,t}, A_{i,t}; \theta_i) + \log p(\theta_i)$$

Proposal Distribution

We choose a Normal distribution for our proposal distribution. At each step of MH, we propose a new sample given the old sample, $\theta^k_{\text{prop}} \sim \mathcal{N}(\theta^k_{\text{old}}, \gamma^2 I)$, where θ^k denotes the kth value of θ.

Metropolis Hastings Acceptance Ratio

The Metropolis Hastings acceptance ratio given a proposed sample θ_{prop} and an old sample θ_{old} is defined as:

$$\alpha(\theta_{\text{prop}}, \theta_{\text{old}}) := \min\left(1, \frac{p(\theta_{\text{prop}}) / q(\theta_{\text{prop}} | \theta_{\text{old}})}{p(\theta_{\text{old}}) / q(\theta_{\text{old}} | \theta_{\text{prop}})}\right)$$

Since our proposal distribution is symmetric, the log acceptance ratio becomes:

$$\log \alpha(\theta_{\text{prop}}, \theta_{\text{old}}) := \min(0, \log p(\theta_{\text{prop}}) - \log p(\theta_{\text{old}}))$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 June 2022 doi:10.20944/preprints202206.0028.v1

Appendix C.2. Action Selection at Decision Time

Appendix C.2.1. **Posterior Sampling**

Our action selection scheme at decision time selects action 1 according to the posterior probability that that arm is optimal. We define these as $\tilde{\pi}_{i,t}$ below.

Bayesian Linear Regression Model Based on the Bayesian Linear Regression model of the reward, specified by Equation (A3):

$$\tilde{\pi}_{i,t} = \Pr_{\tilde{\beta} \sim \mathcal{N}(\mu^{post}_{i,t-1}, \Sigma^{post}_{i,t-1})} \left\{ f(S_{i,t})^T \tilde{\beta} > 0 \middle| S_{i,t}, H_{i,t-1} \right\}.$$

Note that the randomness in the probability above is only over the draw of $\tilde{\beta}$ from the posterior distribution.

Zero-Inflated Poisson Model Based on the Zero-Inflated Poisson model of the reward, specified by Equation (A4):

$$\tilde{\pi}_{i,t} = \Pr_{\tilde{\alpha}_{i,b}, \tilde{\alpha}_{i,p}, \tilde{\beta}_{i,b}, \tilde{\beta}_{i,p}} \left\{ \tilde{Z}_{i,t} \tilde{Y}_{i,t} > 0 \middle| S_{i,t}, H_{i,t-1} \right\}$$

where $\tilde{Z}_{i,t} \sim \text{Bernoulli}\big(1 - \text{sigmoid}(m(S_{i,t})^T \tilde{\alpha}_{i,p} + A_{i,t} \cdot f(S_{i,t})^T \tilde{\beta}_{i,b})\big)$ and $\tilde{Y}_{i,t} \sim \text{Poisson}\big(\exp\big(m(S_{i,t})^T \tilde{\alpha}_{i,p} + A_{i,t} \cdot f(S_{i,t})^T \tilde{\beta}_{i,p}\big)\big)$. Note that the randomness in the probability above is only over the draw of $(\tilde{\alpha}_{i,b}, \tilde{\alpha}_{i,p}, \tilde{\beta}_{i,b}, \tilde{\beta}_{i,p})$ from the posterior distribution.

Appendix C.2.2. **Clipping to form Action Selection Probabilities**

Since we want to facilitate after study analyses, we clip action selection probabilities using action clipping function for some chosen values of $\pi_{\min}, \pi_{\max}$ where $0 < \pi_{\min} \leq \pi_{\max} < 1$ chosen by the scientific team:

$$\text{clip}(\pi) = \min(\pi_{\max}, \max(\pi, \pi_{\min})) \in [\pi_{\min}, \pi_{\max}] \tag{A5}$$

This means that

$$\pi_{i,t} = \text{clip}(\tilde{\pi}_{i,t})$$

References

1. Yu, B.; Kumbier, K. Veridical data science. *Proceedings of the National Academy of Sciences* **2020**, *117*, 3920–3929.
2. Liao, P.; Greenewald, K.H.; Klasnja, P.V.; Murphy, S.A. Personalized HeartSteps: A Reinforcement Learning Algorithm for Optimizing Physical Activity. *CoRR* **2019**, *abs/1909.03539*, [1909.03539].
3. Yom-Tov, E.; Feraru, G.; Kozdoba, M.; Mannor, S.; Tennenholtz, M.; Hochberg, I. Encouraging physical activity in patients with diabetes: intervention using a reinforcement learning system. *Journal of medical Internet research* **2017**, *19*, e338.
4. Hochberg, I.; Feraru, G.; Kozdoba, M.; Mannor, S.; Tennenholtz, M.; Yom-Tov, E. A reinforcement learning system to encourage physical activity in diabetes patients. *arXiv preprint arXiv:1605.04070* **2016**.
5. Forman, E.M.; Kerrigan, S.G.; Butryn, M.L.; Juarascio, A.S.; Manasse, S.M.; Ontañón, S.; Dallal, D.H.; Crochiere, R.J.; Moskow, D. Can the artificial intelligence technique of reinforcement learning use continuously-monitored digital data to optimize treatment for weight loss? *Journal of behavioral medicine* **2019**, *42*, 276–290.
6. Allen, S. Stanford Computational Policy Lab Pretrial Nudges. https://policylab.stanford.edu/projects/nudge.html, 2022.
7. Cai, W.; Grossman, J.; Lin, Z.J.; Sheng, H.; Wei, J.T.Z.; Williams, J.J.; Goel, S. Bandit algorithms to personalize educational chatbots. *Machine Learning* **2021**, pp. 1–30.
8. Qi, Y.; Wu, Q.; Wang, H.; Tang, J.; Sun, M. Bandit Learning with Implicit Feedback. In Proceedings of the Advances in Neural Information Processing Systems; Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; Garnett, R., Eds. Curran Associates, Inc., 2018, Vol. 31.
9. Bezos, J.P. 1997 LETTER TO SHAREHOLDERS. https://www.sec.gov/Archives/edgar/data/1018724/000119312516530910/d168744dex991.htm, 1997.
10. Sutton, R.S.; Barto, A.G. *Reinforcement learning: An introduction*; MIT press, 2018.

11. den Hengst, F.; Grua, E.M.; el Hassouni, A.; Hoogendoorn, M. Reinforcement learning for personalization: A systematic literature review. *Data Science* **2020**, *3*, 107–147.
12. Wang, C.C.; Kulkarni, S.R.; Poor, H.V. Bandit problems with side observations. *IEEE Transactions on Automatic Control* **2005**, *50*, 338–355.
13. Langford, J.; Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in neural information processing systems* **2007**, *20*, 96–1.
14. Tewari, A.; Murphy, S.A. From ads to interventions: Contextual bandits in mobile health. In *Mobile Health*; Springer, 2017; pp. 495–517.
15. Fan, H.; Poole, M.S. What is personalization? Perspectives on the design and implementation of personalization in information systems. *Journal of Organizational Computing and Electronic Commerce* **2006**, *16*, 179–202.
16. Thomas, P.; Brunskill, E. Data-efficient off-policy policy evaluation for reinforcement learning. In Proceedings of the International Conference on Machine Learning. PMLR, 2016, pp. 2139–2148.
17. Levine, S.; Kumar, A.; Tucker, G.; Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* **2020**.
18. Boruvka, A.; Almirall, D.; Witkiewitz, K.; Murphy, S.A. Assessing time-varying causal effect moderation in mobile health. *Journal of the American Statistical Association* **2018**, *113*, 1112–1121.
19. Hadad, V.; Hirshberg, D.A.; Zhan, R.; Wager, S.; Athey, S. Confidence intervals for policy evaluation in adaptive experiments. *Proceedings of the National Academy of Sciences* **2021**, *118*.
20. Yao, J.; Brunskill, E.; Pan, W.; Murphy, S.; Doshi-Velez, F. Power Constrained Bandits. In Proceedings of the Proceedings of the 6th Machine Learning for Healthcare Conference; Jung, K.; Yeung, S.; Sendak, M.; Sjoding, M.; Ranganath, R., Eds. PMLR, 2021, Vol. 149, *Proceedings of Machine Learning Research*, pp. 209–259.
21. Murnane, E.L.; Huffaker, D.; Kossinets, G. Mobile Health Apps: Adoption, Adherence, and Abandonment. In Proceedings of the Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers; Association for Computing Machinery: New York, NY, USA, 2015; UbiComp/ISWC'15 Adjunct, p. 261–264. https://doi.org/10.1145/2800835.2800943.
22. Dennison, L.; Morrison, L.; Conway, G.; Yardley, L. Opportunities and Challenges for Smartphone Applications in Supporting Health Behavior Change: Qualitative Study. *J Med Internet Res* **2013**, *15*, e86. https://doi.org/10.2196/jmir.2583.
23. Agarwal, A.; Alomar, A.; Alumootil, V.; Shah, D.; Shen, D.; Xu, Z.; Yang, C. PerSim: Data-Efficient Offline Reinforcement Learning with Heterogeneous Agents via Personalized Simulators. *arXiv preprint arXiv:2102.06961* **2021**.
24. Figueroa, C.A.; Aguilera, A.; Chakraborty, B.; Modiri, A.; Aggarwal, J.; Deliu, N.; Sarkar, U.; Jay Williams, J.; Lyles, C.R. Adaptive learning algorithms to optimize mobile applications for behavioral health: guidelines for design decisions. *Journal of the American Medical Informatics Association* **2021**, *28*, 1225–1234.
25. Wei, H.; Chen, C.; Liu, C.; Zheng, G.; Li, Z. Learning to simulate on sparse trajectory data. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, 2020, pp. 530–545.
26. Ie, E.; wei Hsu, C.; Mladenov, M.; Jain, V.; Narvekar, S.; Wang, J.; Wu, R.; Boutilier, C. RecSim: A Configurable Simulation Platform for Recommender Systems, 2019, [arXiv:cs.LG/1909.04847].
27. Santana, M.R.O.; Melo, L.C.; Camargo, F.H.F.; Brandão, B.; Soares, A.; Oliveira, R.M.; Caetano, S. MARS-Gym: A Gym framework to model, train, and evaluate Recommender Systems for Marketplaces, 2020, [arXiv:cs.IR/2010.07035].
28. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym, 2016, [arXiv:cs.LG/1606.01540].
29. Wang, S.; Zhang, C.; Kröse, B.; van Hoof, H. Optimizing Adaptive Notifications in Mobile Health Interventions Systems: Reinforcement Learning from a Data-driven Behavioral Simulator. *Journal of medical systems* **2021**, *45*, 1–8.
30. Singh, A.; Halpern, Y.; Thain, N.; Christakopoulou, K.; Chi, E.; Chen, J.; Beutel, A. Building healthy recommendation sequences for everyone: A safe reinforcement learning approach. In Proceedings of the FAccTRec Workshop, 2020.
31. Korzepa, M.; Petersen, M.K.; Larsen, J.E.; Mørup, M. Simulation Environment for Guiding the Design of Contextual Personalization Systems in the Context of Hearing Aids. In Proceedings of the Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization; Association for Computing Machinery: New York, NY, USA, 2020; UMAP '20 Adjunct, p. 293–298. https://doi.org/10.1145/3386392.3399291.
32. Hassouni, A.e.; Hoogendoorn, M.; van Otterlo, M.; Barbaro, E. Personalization of Health Interventions Using Cluster-Based Reinforcement Learning. In Proceedings of the PRIMA 2018: Principles and Practice of Multi-Agent Systems; Miller, T.; Oren, N.; Sakurai, Y.; Noda, I.; Savarimuthu, B.T.R.; Cao Son, T., Eds.; Springer International Publishing: Cham, 2018; pp. 467–475.
33. Hassouni, A.e.; Hoogendoorn, M.; van Otterlo, M.; Eiben, A.E.; Muhonen, V.; Barbaro, E. A clustering-based reinforcement learning approach for tailored personalization of e-Health interventions, 2018. https://doi.org/10.48550/ARXIV.1804.03592.
34. Dwivedi, R.; Tan, Y.S.; Park, B.; Wei, M.; Horgan, K.; Madigan, D.; Yu, B. Stable Discovery of Interpretable Subgroups via Calibration in Causal Studies. *International Statistical Review* **2020**, *88*, S135–S178, [https://onlinelibrary.wiley.com/doi/pdf/10.1111/insr.12427]. https://doi.org/https://doi.org/10.1111/insr.12427.
35. Ward, O.G.; Huang, Z.; Davison, A.; Zheng, T. Next waves in veridical network embedding. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **2021**, *14*, 5–17.
36. Margot, V.; Luta, G. A new method to compare the interpretability of rule-based algorithms. *AI* **2021**, *2*, 621–635.

37. Shetty, V.; Morrison, D.; Belin, T.; Hnat, T.; Kumar, S. A Scalable System for Passively Monitoring Oral Health Behaviors Using Electronic Toothbrushes in the Home Setting: Development and Feasibility Study. *JMIR Mhealth Uhealth* **2020**, *8*, e17347. https://doi.org/10.2196/17347.
38. Jiang, N.; Kulesza, A.; Singh, S.; Lewis, R. The dependence of effective planning horizon on model accuracy. In Proceedings of the Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. Citeseer, 2015, pp. 1181–1189.
39. Yom-Tov, E.; Feraru, G.; Kozdoba, M.; Mannor, S.; Tennenholtz, M.; Hochberg, I. Encouraging Physical Activity in Patients With Diabetes: Intervention Using a Reinforcement Learning System. *J Med Internet Res* **2017**, *19*, e338. https://doi.org/10.2196/jmir.7994.
40. Russo, D.; Roy, B.V.; Kazerouni, A.; Osband, I. A Tutorial on Thompson Sampling. *CoRR* **2017**, *abs/1707.02038*, [1707.02038].
41. Zhu, F.; Guo, J.; Xu, Z.; Liao, P.; Yang, L.; Huang, J. Group-driven reinforcement learning for personalized mhealth intervention. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, 2018, pp. 590–598.
42. Tomkins, S.; Liao, P.; Klasnja, P.; Murphy, S. IntelligentPooling: practical Thompson sampling for mHealth. *Machine Learning* **2021**, pp. 1–43.
43. Deshmukh, A.A.; Dogan, U.; Scott, C. Multi-task learning for contextual bandits. *Advances in neural information processing systems* **2017**, *30*.
44. Vaswani, S.; Schmidt, M.; Lakshmanan, L. Horde of bandits using gaussian markov random fields. In Proceedings of the Artificial Intelligence and Statistics. PMLR, 2017, pp. 690–699.
45. Feng, C.X. A comparison of zero-inflated and hurdle models for modeling zero-inflated count data. *Journal of Statistical Distributions and Applications* **2021**, *8*, 1–19.
46. Cole, S.R.; Platt, R.W.; Schisterman, E.F.; Chu, H.; Westreich, D.; Richardson, D.; Poole, C. Illustrating bias due to conditioning on a collider. *International journal of epidemiology* **2010**, *39*, 417–420.
47. Luque-Fernandez, M.A.; Schomaker, M.; Redondo-Sanchez, D.; Jose Sanchez Perez, M.; Vaidya, A.; Schnitzer, M.E. Educational Note: Paradoxical collider effect in the analysis of non-communicable disease epidemiological data: a reproducible illustration and web application. *International journal of epidemiology* **2019**, *48*, 640–653.
48. Elwert, F.; Winship, C. Endogenous selection bias: The problem of conditioning on a collider variable. *Annual review of sociology* **2014**, *40*, 31–53.