Predicting Missing Wearable Device Data with Panel POMP

Abstract

Missing data causes problems in reinforcement-learning based models, such as those used for mental health interventions in the Intern Health Study (IHS). When reward data are frequently missing, for example because a wearable device is not turned on or worn, reinforcement-learning based models learn more slowly. This project aims to estimate this missing physical activity data with a latent Partially Observed Markov Process (POMP) model in a panel setting with 78 participants.

Joint learning of measurement dispersion is attempted with two different pooling regimes, partial pooling and global pooling, and compared against a no-pooling POMP and a baseline ARMA model. As a prediction task, we devised a masked data-based metric to compare model performance and found structure in the measurement dispersion, but only a modest advantage for the pooling regimes.

1 Introduction

1.1 Background Information and Project Motivation

For a new medical school graduate, the first year of residency is demanding and stressful. The yearly Intern Health Study (IHS) aims to study the effect of stress on mental health outcomes, enrolling over a thousand new interns each year. [1] The study collects large amounts of data on interns including quarterly psychological surveys, DNA samples, daily mood scores, and sleep and physical activity data from wearable devices. In this project, we focus only on physical activity measured by step count in the 2024 IHS.

In addition to collecting data, recent iterations of the IHS are incorporating reinforcement-learning based mobile health interventions. At 3:00 p.m. each day, an agent decides based on the participant's context whether to send a participant a message, with some learned probability. This randomized setup allows researchers to study whether these messages influence participant behavior. In this context, the objective is to increase physical activity, so step count serves as the reward outcome of interest. To be precise, the reward for a given day is the number of steps a user takes in the 24 hours after the 3 p.m. decision point in local time.

Reinforcement learning and contextual bandit methods are becoming increasingly important in healthcare because they enable personalized interventions. In digital health, these methods can be used to decide when and how to deliver prompts that encourage healthier behaviors, such as increasing physical activity or improving medication adherence. Contextual bandits are especially

useful because they incorporate patient characteristics when selecting an action. For example, RoME is a robust contextual bandit method for mobile health applications that determines when to send notifications and how to personalize interventions based on a user's current situation. [2] Similarly, Oralytics is a mobile health intervention system that uses an online reinforcement learning algorithm to determine optimal times to deliver interventions that promote oral self-care behaviors. [3]

This project is motivated by the growing need to apply these methods in realistic settings where data are often incomplete. In our dataset, missingness occurs when users do not wear their devices, which leads to gaps in the recorded activity data. Missing data can distort the information available to the algorithm and may ultimately affect its behavior and performance. As a result, robust strategies for handling delayed and missing data are essential so that the algorithm can continue to function reliably and produce reproducible results. [4]

1.2 Past Project Context

Most past projects centered on infectious disease or financial topics. There was one project that also looked at measurements from wearable devices [5]. That project focused on the instantaneous effect of environmental noise on heart rate variability, while ours is primarily about predicting step count in the next 24 hours following a motivating message sent to study participants.

Both projects face the same issue of missing data. The heart rate variability project pooled all participants together into a population-level daily summary, taking the median measurement of each day. While we initially designed Partially Observed Markov Process (POMP) models for individual participants, we also incorporated global and partial pooling to jointly learn measurement dispersion parameters across participants in a compute-intensive, high-parameter panel POMP system. We did not find prior course projects that focused on prediction of missing activity data or implemented a panel POMP, so we emphasize predictive model selection on masked data rather than scientific inference.

1.3 Research Question

When a participant does not wear the watch, no step count data are collected during that period. On days when the watch is worn for only a short time, or not at all, a reward cannot be constructed, so that day provides no training information for the reinforcement learning based intervention. The goal of this project is to build personalized POMP models for each doctor and to estimate the missing rewards.

2 Exploratory Data Analysis

Participants wear devices from Garmin, Fitbit, and Apple, and some individuals may use more than one device. Since these devices differ in how they record and store data, as well as behavioral differences in device ownership, the data streams cannot be combined perfectly. While joint

measurement models across devices would be valuable, due to time and compute constraints, we limit ourselves to Fitbit users.

For this project, we focused only on users who had Fitbit data. Since few users had Garmin devices, it would be difficult to train the model, and Apple HealthKit acts as a central repository for health and fitness data which combines information from both phones and watches and may aggregate step counts originating from other wearables. We found it difficult to disentangle the biases associated with Apple-based measurements.

The `FitbitHourly.csv` file contains data from 299 participants with Fitbit data. We limited our analysis to the period from July 1, 2024, to September 30, 2024, the first three months of IHS 2024. Additionally, we filtered to participants with Fitbit step data in at least half of their hourly rows, which left us with 78 participants.

The chart on the right is a scatterplot of the mean and standard deviation of hourly steps for each user after filtering to hours with non-missing data. The means are quite a bit larger than would be expected under an iid Poisson observation model, which motivates the use of a latent stochastic process together with a measurement model that allows extra dispersion.

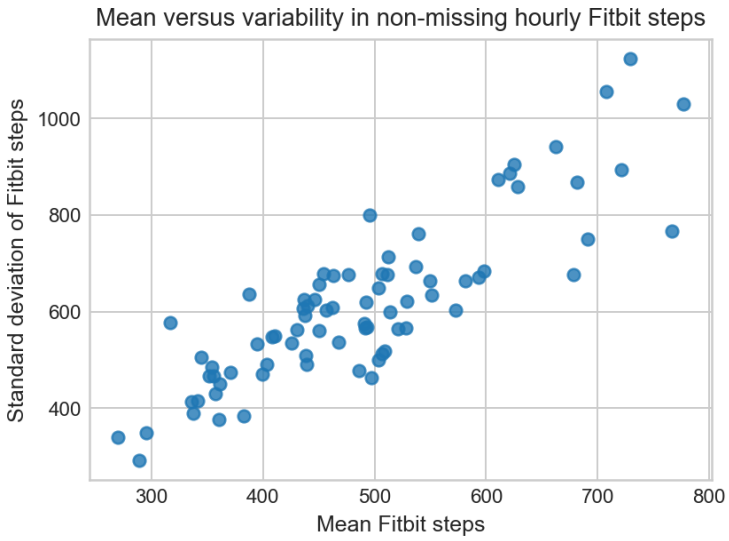


Figure 1: Mean versus variability in non-missing hourly Fitbit steps.

3 Model Design

Our aim is to estimate the 24-hour reward, the sum of the steps in the 24 hours following 3 p.m. local time. We work at the hourly level, so each hour is treated as one bucket. Fitbit step counts may be observed or missing in a given bucket, which motivates a partially observed Markov process:

$$x_{t+1} = \phi x_t + \sigma \varepsilon_t, \qquad \varepsilon_t \sim N(0,1),$$

where x_t represents hour-to-hour deviations in activity not explained by systematic daily structure, ϕ controls temporal persistence, and σ controls the magnitude of unexplained hourly variation.

Conditional on the latent state, the latent step count in hour t is

$$N_t \mid x_t \sim \text{Poisson}(\lambda_t), \qquad \log \lambda_t = \bar{\ell} + \alpha_{h(t)} + \delta_{w(t)} + x_t.$$

Here, $h(t)$ is the local hour of day and $w(t)$ is the local day of week. The hour-of-day and weekday terms are estimated empirically for each participant from the observed step series, with the construction details given in the supplementary material.

The observed Fitbit step count is modeled as a noisy measurement of the latent hourly steps:

$$Y_t \mid N_t \sim \text{NegBin}(N_t, k),$$

with variance

$$\text{Var}(Y_t \mid N_t) = N_t + \frac{N_t^2}{k}.$$

Larger values of k correspond to more Poisson-like measurement behavior. Because all participants are using Fitbit devices, k is the most natural parameter to pool across participants.

Our quantity of interest is the latent 24-hour reward

$$R_d = \sum_{t \in H(d)} N_t,$$

where $H(d)$ denotes the 24 hourly buckets following the 3 p.m. decision time on day d. Because the reward is defined using the latent counts N_t, the model can be used to reconstruct daily rewards even when hourly Fitbit observations are missing.

All 78 participants are jointly fit in a panel POMP. We compare three AR(1)-based panel models that differ only in how the measurement over-dispersion parameter is handled:

1. participant-specific k_i with no pooling,
2. a globally shared k,
3. partially pooled participant-specific k_i values.

For partial pooling, we parameterize $\eta_i = \log k_i$ and apply a Gaussian penalty inside the objective function:

$$\sum_{i=1}^{n} \ell_i(\phi_i, \sigma_i, e^{\eta_i}) - \frac{1}{2\tau_k^2} \sum_{i=1}^{n} (\eta_i - \mu_k)^2 - n \log \tau_k.$$

When τ_k is small, the model approaches the globally pooled regime; when τ_k is large, the model approaches the no-pooling regime. Inuitively, each user gets their own k, but the penalty nudges

them along the shared Gaussian. Because all participants are using Fitbit devices, k is the parameter with the clearest device-level interpretation and is therefore the most natural candidate for pooling.

Implementing this within the pypomp panel interface is a bit tricky. Each user has its own Pomp model object. The PanelPomp object takes in a list of these unit pomp objects, as well as a "theta_list", designating which parameters are shared and which are unit-specific. For the global model, the k is kept as shared. For the global, the k is kept personal for each individual, but they all share an extra two parameters, "mu_log_k_fitbit" and "log_tau_log_k_fitbit", which correspond to the parameters of the Gaussian penalty. Then, the Gaussian penalty is implemented into the objective function inside the dmeas for the individual Pomp objects.

Lastly, because the missing rewards which are being estimated aren't needed at decision time but can be backfilled later, future information can be used to estimate the missing reward. So we additionally implement the natural smoother, which performs a backwards pass after the forward pass of the filter is used for a prediction.

4 Individual Participant Demonstrations

In order to understand the dynamics of the full panel pomp, it helps to see how the model fits at an individual level. As a demonstration, we run a small global MLE search (10 starts, 100 particles, 4 iterations in stage 1) for three separate participants with low, medium, and high missingness, to get a low-precision sense of the effect of missingness on reward prediction.

Table 1

PID	LogLik	phi	sigma	k	Missingness
MDH-3531-4796	-8187.758	-0.260	0.626	1.063	49.46%
MDH-4235-8718	-10643.608	0.103	1.808	3.591	33.56%
MDH-7960-3351	-12538.727	0.618	1.486	3.364	19.29%

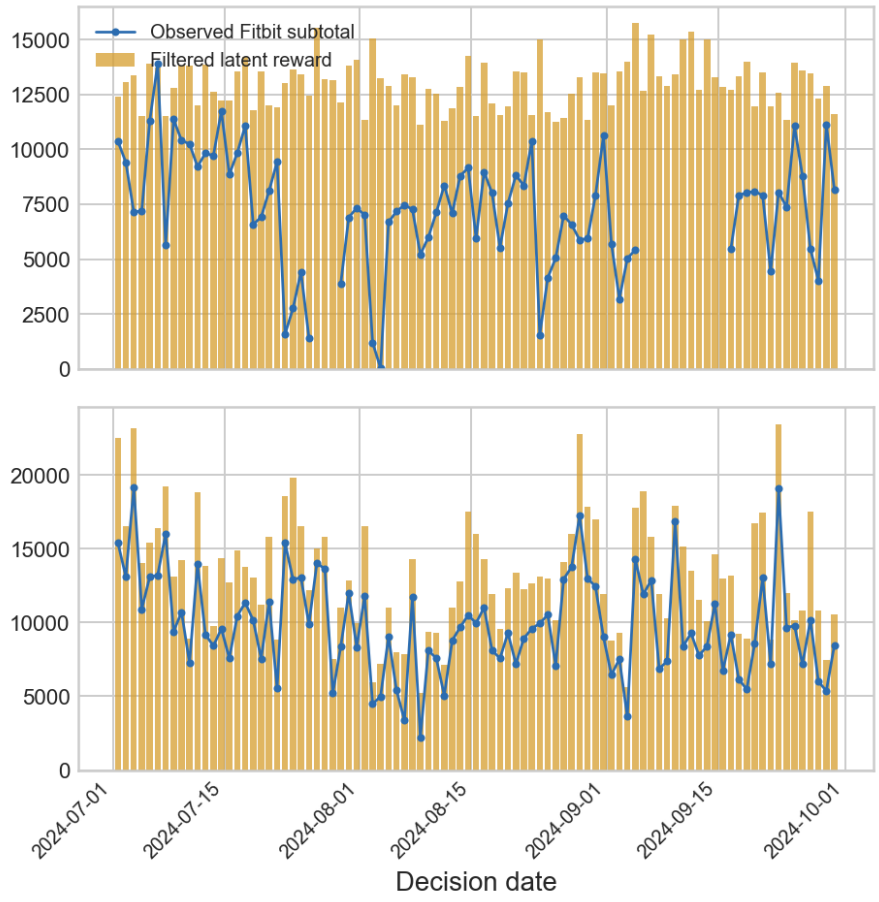


Figure 2 demonstrates the difference between the latent and observed rewards for individual participants with high and medium missingness. We see that the estimated latent reward is always greater than the observed reward. There is a larger difference for the participant with high missingness. This is expected because it is a sum of hourly buckets, for which many of the observations are missing.

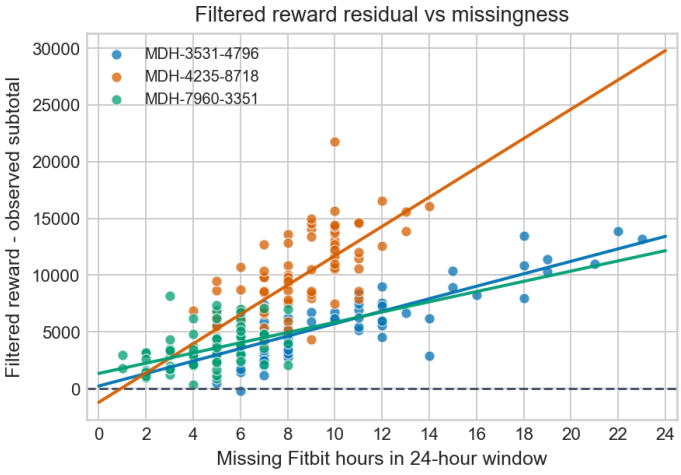


Figure 3: Filtered reward residual versus missingness for the three participant demonstrations.

Figure 3 shows a linear trend in how much the estimated latent exceeds the observed. This is unsurprising; if the observed only sums 8 non-zero buckets, the latent sums 24 non-zero buckets, and might be approximately three times larger. This discrepancy is why we use a latent model in the first place, as we expect that there are many steps the participant is taking that are not being picked up by Fitbit or other measurement devices. This phenomenon presents challenges with finding a representative benchmark that we discuss below.

5 Benchmarks for Aggregate Modeling

There were several benchmarks to consider, and narrowing them down was challenging. Since our goal was prediction rather than inference, we prioritized models that perform best on unseen data. Specifically, we aimed to improve the accuracy of predicted 24-hour rewards when users have missing hourly observations. Because we fit 78 personalized models jointly, we also need aggregate summaries that capture predictive ability across participants.

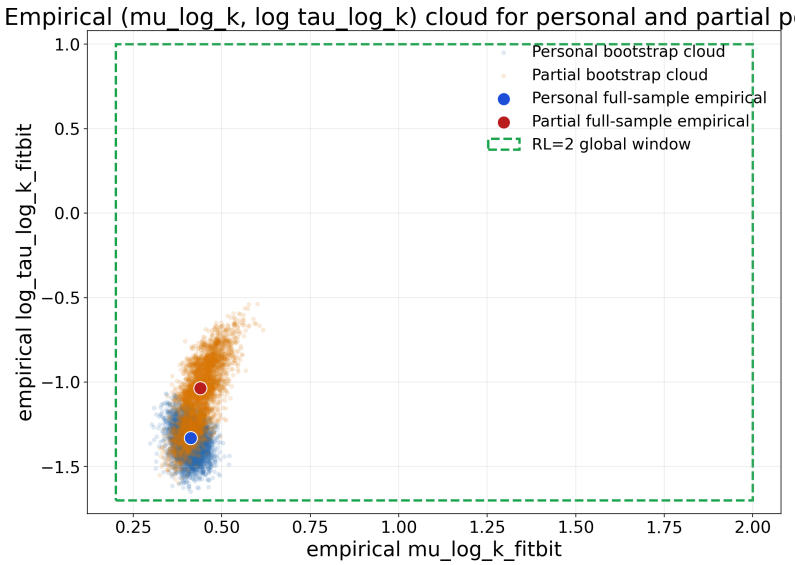
Direct error is only meaningful on hours where Fitbit originally recorded nonzero step counts. When a participant is not wearing the device, the observed series can contain zeros that do not represent true inactivity, so evaluating prediction error against those zeros would be misleading. We therefore use a masking approach: for each participant, we refit the model after masking 20% of the observed hourly Fitbit data and then evaluate reconstruction quality on the held-out masked observations. We report both hourly reconstruction metrics and a daily held-out subtotal benchmark that aligns more directly with the 24-hour reward target. Additional details on the masked subtotal metric are given in the supplementary material.

6 Inference Settings

We ran a global IF2 MLE search for each of the three models with two stages. Stage 1, 32 starts, 1024 particles, 16 iterations, with 256 evaluation particles with 16 evaluation repeats, with a smaller stage 2, 16 starts, 512 particles, 16 iterations, 256 evaluation particles repeated 8 times. Each fit took about an hour and a half on a 3080 Nvidia GPU with 10GB of VRAM, with 8704 CUDA cores.

The scale of this made computation very challenging! Considering just the version with no pooling, each of the 78 participants had 3 parameters, which comes out to 234 parameters total. Running this across 3 models was very time intensive, and getting it to function correctly in qmd was another challenge entirely!

In Table~2 we display the window for each global model that we perform the search on. k was searched on a log scale, and for the partial pooling, the variance τ and μ for the penalty term were searched also on the log scale. Figure~4 displays the global window for those distribution terms, alongside empirical bootstrap estimates from the partial-pooling and no-pooling models. It is not conclusive, but it suggests that our window does cover this region.



Parameter	Low	High
k_fitbit	0.750	12.000
log_k_fitbit	-0.288	2.485
log_tau_log_k_fitbit	-1.700	1.000
mu_log_k_fitbit	0.200	2.000
phi	-0.300	1.000
sigma	0.010	1.800

7 Results

Next, we fit the three primary models, as well as ARMA. The ARMA (p, q) is picked by best scoring on AIC independently for each individual. After fitting the three primary models, we make predictions in one pass with a filter (labeling these globalfilter, partialfilter, and personalfilter), then perform the natural smoother with a following backward pass across the data (likewise named globalsmooth, partialsmooth, personalsmooth).

Metric	globalfilter	globalsmooth	partialfilter	partialsmooth	personalfilter	personalsmooth	ARMA
Avg logLik	-8744.575	-8744.575	-8750.662	-8750.662	-8746.525	-8746.525	-2180.327
RMSE Subtotal	816.970	764.857	806.814	744.392	808.902	743.521	2196.651
Avg hr corr	0.244	0.314	0.249	0.326	0.247	0.328	0.221
Avg day corr	0.740	0.600	0.740	0.606	0.736	0.609	0.613
Avg phi	0.466	0.466	0.471	0.471	0.482	0.482	
Avg sigma	0.923	0.923	0.899	0.899	0.890	0.890	
Avg k	1.347	1.347	1.693	1.693	1.562	1.562	
SD k	0.000	0.000	1.117	1.117	0.404	0.404	

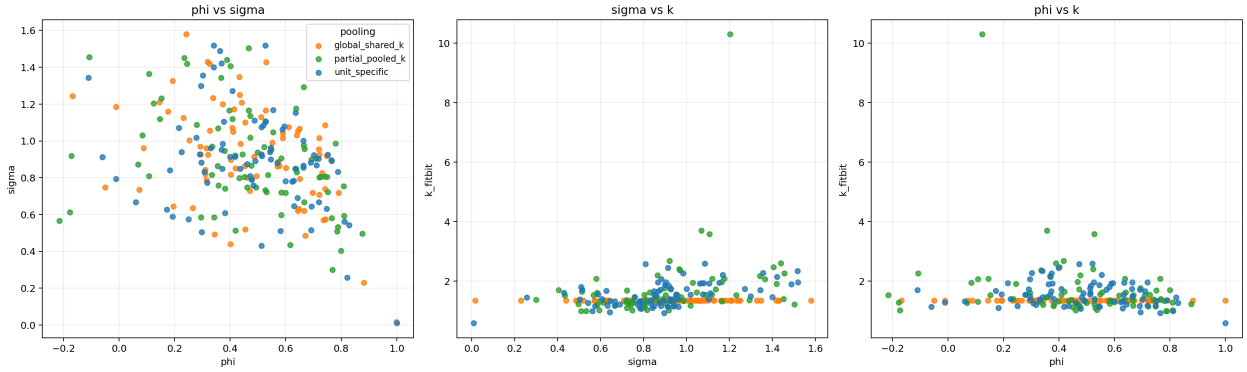
All the POMP models performed substantially better than the baseline ARMA models. The log-likelihoods appear much worse, but this is to be expected. The POMP models are not trying to estimate the data, they're trying to estimate an underlying latent that always upper bounds the data. So while ARMA has a much lower log likelihood, it just tries to fit itself to observed rewards,

and hence performs much worse on the normalized benchmark proposed for tracking prediction ability.

The "avg hr corr" and "avg day corr" are average correlations between the prediction and held-out masked data on the hourly level, and on the daily subtotal level respectively. The daily level is more relevant to the task of predicting daily level rewards, but the contrast with the hourly is of note: While these models may predict a single hour poorly, their ability to predict a larger sum of values improves relatively.

For the filtered models, they perform very similarly. The partial pooling performs slightly better on the RMSE Subtotal, but 807 vs 809 is likely within range of Monte Carlo noise. The same trend can be seen in the smoothing case; the personal smoother now performs 0.7 RMSE Subtotal units better.

The smoothers consistently gain a modest boost over the filtered versions, suggesting that the future information does help predict unseen prior information. Interestingly, their daily correlation decreases substantially! We aren't sure why this is. It also presents more of a tradeoff than expected between a filtered estimate and a smoother estimate.



Each dot is a fit for a different participant. Visibly there is a minor tradeoff between ϕ and σ. Additionally, there's a small positive trend between fit k and σ, for both partial and individual pooling. Oddly, the partial fits are slightly more spread than the individual fits.

8 Conclusion

Our project focused on predicting the 24-hour reward for participants using Fitbit step count data. We compared three different AR(1)-based panel POMP models: an individual (no pooling) model, in which each participant has their own over-dispersion parameter k; a partial pooling model, in which each participant's k value is drawn from a shared distribution with a penalty term to prevent extreme deviations; and a global pooling model, in which all participants share the same k value. We also compare to a baseline ARMA model. Our results show highly similar results across the different pooling regimes, and a substantial improvement with mechanistic models over the baseline ARMA fits. Additionally, smoothers modestly reduce error, but reduce day-level masked correlation.

To evaluate performance, we trained each model on 80% of the observed hourly data and predicted the remaining 20% of observed step counts. We calculated the masked subtotal error ($\frac{\text{RMSE}_{\text{subtotal}}}{\sqrt{m}}$),

which describes the difference between predicted and observed values, for the remaining 20%, normalized across a single day.

In future work, there are several additional methods we could explore in our model specification. Our current latent process model only incorporates baseline averages over hour of day and week. For $log\lambda_t$, we could try including additional covariates related to physical activity, such as heart rate and sleep. Additionally, we were not able to model the measurement distribution jointly across all three devices for participants who used multiple devices, and therefore could not fully leverage all available data. Instead, we focused primarily on Fitbit data due to difficulties in disentangling device-specific biases. In terms of parameter sharing, we also considered clustering participants based on shared characteristics, such as similar sleep schedules, to better capture common behavioral patterns.

In terms of model structure, there are also limitations imposed by the pypomp library. We initially intended to include a broader set of covariates and use LASSO regularization for participant-specific feature selection to improve flexibility in modeling long-term dependencies. However, this was not feasible within the current framework of the library.

There are more robust approaches we could have used to evaluate model accuracy. In our analysis, we only applied masking to 20% of non-missing hourly observations per participant. Future work could explore a range of masking proportions to assess sensitivity to missingness. Additionally, masking could be applied in different structures, such as contiguous hourly blocks, full-day removals, or random missingness patterns.

Finally, we had a major reproducibility incident in the last hours of the submission day. The complexity of the model was large, and between merging code from multiple participants, the panel pomp fits which compiled on one group member's machine with GPU was not able to compile on the other group members' machines. While attempting to fix this, so that the .qmd was robust to small RL=0 CPU runs, and large RL=2 GPU runs, with caching implemented, there was a git merge conflict. In resolving this conflict by returning to a prior branch, very expensive cached model parameters were lost. In order to submit, the model parameters for the panel pomp were recomputed, with the benchmarked results saved externally, and imported to the document.

If we had more time, our immediate priority would be to fix these compile errors, and reimplement the results to run directly from the computations within the qmd. For any future students, let this be a lesson to plan your version control and prioritize incremental change which works across devices in the very early stages of the project.

9 Contributions

- Group Member 1: Model and benchmark design, implementation of panel pomps on local GPU, and messed up version control, and reproducibility.
- Group Member 2: EDA, section 1.2, section 4 figures, analysis brainstorming, editing and formatting
- Group Member 3: EDA, abstract, introduction literature review, conclusion, analysis brainstorming, editing and formatting

Acknowledgments

This report used the template available in the STATS 531 GitHub repository. AI was used with ChatGPT 5.3 Codex extensively for coding, analysis, and formatting in Quarto, using the Codex and Quarto VS Code extensions. Microsoft Copilot and UM-GPT 5.2 were also used.

Bibliography

[1] Intern Health Study, "Intern health study." n.d. Available: https://www.internhealthstudy.org/

[2] E. K. Huch, J. Shi, M. R. Abbott, J. R. Golbus, A. Moreno, and W. H. Dempsey, "RoME: A robust mixed-effects bandit algorithm for optimizing mobile health interventions." 2025. Available: https://arxiv.org/pdf/2312.06403

[3] A. L. Trella *et al.*, "A deployed online reinforcement learning algorithm in an oral health clinical trial." 2024. Available: https://arxiv.org/pdf/2409.02069

[4] S. Ghosh *et al.*, "Reproducible workflow for online AI in digital health." 2025. Available: https://arxiv.org/pdf/2509.13499

[5] J. He, "Daily environmental noise and heart-rate variability." 2025. Available: https://ionides.github.io/531w25/final_project/project10/blinded.pdf

10 Supplementary material

10.1 Data Sources and Transformations

This project uses 2024 Intern Health Study (IHS) data provided by a group member who is currently conducting research on this topic.

The following data file was used for this project:

- `FitbitHourly.csv`
  - `PARTICIPANTIDENTIFIER`: unique ID for each doctor that participated in the study
  - `STUDY_USER_ID`: study-specific unique ID for each participant
  - `time_utc`: local time
  - `date_utc`: local date
  - `hour_of_day_utc`: local hour from 0 to 23
  - for Apple, Fitbit, and Garmin, the file records steps, heart rate, heart rate variability, sleep, and the corresponding missingness indicators
  - `moodScore`: survey-based score ranging from 0 to 10
  - `messageSent`: binary indicator for whether the agent sent the participant a message during that hour

10.2 Masked Subtotal Benchmarking Metric

Using hourly observations with nonzero Fitbit step counts, we compute error at the daily level to align with the research goal of predicting 24-hour rewards. First, define

$$\mathcal{H}_d = \{t \in d : \text{hour } t \text{ was artificially masked and } y_t \text{ is observed}\}.$$

Let $m_d = |\mathcal{H}_d|$ represent the 20% of available data that was masked but originally had nonzero step counts.

Next, define the true observed subtotal for this masked subset as

$$S_d = \sum_{t \in \mathcal{H}_d} y_t,$$

and the corresponding predicted subtotal as

$$\hat{S}_d = \sum_{t \in \mathcal{H}_d} \hat{y}_t.$$

The subtotal error is defined as

$$E_d = \hat{S}_d - S_d = \sum_{t \in \mathcal{H}_d} (\hat{y}_t - y_t),$$

It is important to normalize the subtotal error, since the raw subtotal error tends to grow with the number of masked hourly observations in the day. The normalized subtotal error is

$$Z_d = \frac{E_d}{\sqrt{m_d}}.$$

Finally, the benchmark is defined as

$$\text{RMSE}_{\text{subtotal}/\sqrt{m}} = \sqrt{\frac{1}{D} \sum_{d=1}^{D} Z_d^2}.$$

Search	Participant ID	LogLik	phi	sigma	k_fitbit

11 Supplementary Tables

Table 3

Search	Participant ID	LogLik	phi	sigma	k_fitbit
Local MLE	MDH-3531-4796	-8754.448	0.259	2.231	199.729
Local MLE	MDH-4235-8718	-28666.826	0.964	1.600	197.841
Local MLE	MDH-7960-3351	-13362.090	0.297	2.279	198.187

Table 4

Search	Participant ID	Stage	Starts	Successful fits	Keep top	Best logLik	Runtime (s)
Global	MDH-3531-4796	Stage 1	10	10	2	-8187.758	60.315
Global	MDH-3531-4796	Stage 2	2	2	1	-8190.194	11.223
Global	MDH-4235-8718	Stage 1	10	10	2	-10643.608	65.794
Global	MDH-4235-8718	Stage 2	2	2	1	-10704.069	12.641
Global	MDH-7960-3351	Stage 1	10	10	2	-12538.727	64.983
Global	MDH-7960-3351	Stage 2	2	2	1	-12551.186	12.615

Table 5

Participant ID	Parameter	Initial value	Local MLE value	Global MLE value
MDH-3531-4796	phi	0.750	0.259	-0.260
MDH-3531-4796	sigma	0.300	2.231	0.626
MDH-3531-4796	k_fitbit	200.000	199.729	1.063
MDH-4235-8718	phi	0.750	0.964	0.103
MDH-4235-8718	sigma	0.300	1.600	1.808
MDH-4235-8718	k_fitbit	200.000	197.841	3.591
MDH-7960-3351	phi	0.750	0.297	0.618
MDH-7960-3351	sigma	0.300	2.279	1.486
MDH-7960-3351	k_fitbit	200.000	198.187	3.364