User Interface Optimization for Quantified Self

Yugdeep Bangar

Master's Thesis

TUDelft

User Interface Optimization for Quantified Self

by

Yugdeep Bangar

to obtain the degree of
Master of Science in Computer Science
at the Delft University of Technology.

Student number:	4733967		
Thesis committee:	Prof. Dr. Alan Hanjalic,	TU Delft	Chair & Supervisor
	Ir. Thomas Langerak,	ETH Zurich	External Supervisor
	Prof. Pablo Cesar,	TU Delft	Member
	Dr. Willem-Paul Brinkman,	TU Delft	Member

TU Delft

Abstract

Self-tracking has expanded exponentially in an era defined by the ubiquitous presence of wearable technologies and smart devices. From health and fitness to finances and productivity, these devices empower users to delve into their quantified self (QS) through an almost infinite amount of visualizations. However, a user has limited time to engage with the data, and the device with which they interact has limited screen space — this calls for the need to suggest the user proactively with valuable and informative plots. An algorithm can suggest and select plots of user activities and adapts to a user's changing requirements while offering maximum usefulness in the information. We leverage combinatorial optimization to handle the multi-objective task of extracting the most informative k plots from a much larger pool of N plots. The novel optimization formulation encapsulates plot features into four unique objective functions designed to capture diverse aspects of data usefulness. We evaluate the efficacy of our selection method *in silico* for various diverse usage scenarios. The simulation results show that the proposed methodology is efficacious in realizing three objectives ('relevance', 'freshness', and 'likability') while identifying the need for refinement in the fourth objective ('noteworthiness'). Our work demonstrates the design, development, and evaluation of a selection algorithm that delivers a relevant yet fresh selection of visualizations for a quantified-self user interested in keeping track of information related to several activities.

Contents

List of Tables

List of Figures

Index of Notation

Data Sources

M	Plot Features
P	Plotted User Data
G	User Goals
I	Interaction Log
S	Selection Log

Input Set, Parameters, and Decision Variables

U	Set $\{u_1, u_2, \dots, u_n\}$ of all plots (i.e. UI element bank)
u_i	i-th plot in the UI element bank
n	Number of plots in UI element bank
k	Number of plots that needs to be selected from U
X	Set $\{u_1, u_2, \dots, u_n\}$ of all binary decision variables
x_i	i-th binary decision variable signifying if the plot u_i is included in final selection of k plots or not $x_i \in \{0, 1\} \quad \forall x_i \in X$
G'	Set of activities marked as Goals by the user

Raw Feature Values

$M_{act}(u_i)$	Set of activities depicted in plot u_i
$M_{plt}(u_i)$	Plot-type for plot u_i
$I^{+}_{M_{plt}(u_i)}$	Number of 'likes' for plots of same plot-type as plot u_i
$I^{-}_{M_{plt}(u_i)}$	Number of 'dislikes' for plots of same plot-type as plot u_i
$M_{ant}(u_i)$	Analysis-type for plot u_i
$I^{+}_{M_{ant}(u_i)}$	Number of 'likes' for plots of same analysis-type as plot u_i
$I^{-}_{M_{ant}(u_i)}$	Number of 'dislikes' for plots of same analysis-type as plot u_i
$M_{frt}(u_i)$	Frequency-type for plot u_i
$I^{-}_{M_{frt}(u_i)}$	Number of 'likes' for plots of same frequency-type as plot u_i
$I^{-}_{M_{frt}(u_i)}$	Number of 'dislikes' for plots of same frequency-type as plot u_i

$P_{data}(u_i)$	Data values for plotted user data in plot u_i
G_j^*	Target value for j-th goal activity
$P_{t_{start}}(u_i)$	Start time (i.e. x-min) of user data plotted in plot u_i
$P_{t_{end}}(u_i)$	End time (i.e. x-max) of user data plotted in plot u_i
$S_{t_{last}}(u_i)$	Last time plot u_i was selected for display to the user

Quantified Feature Values

Q	Set $\{q_{act}, q_{plt}, \ldots, q_{toc}\}$ of all quantified feature values
$q_{act}(u_i)$	Score of plot u_i on feature quantifying relevance of activities feature
$q_{plt}(u_i)$	Score of plot u_i on feature quantifying likability of plot-type
$q_{ant}(u_i)$	Score of plot u_i on feature quantifying likability of analysis-type
$q_{frt}(u_i)$	Score of plot u_i on feature quantifying likability of frequency-type
$q_{tre}(u_i)$	Score of plot u_i on feature quantifying trendiness in data
$q_{goc}(u_i)$	Score of plot u_i on feature quantifying goal completion time
$q_{daf}(u_i)$	Score of plot u_i on feature quantifying freshness in plotted data

Weights and Objective Functions

F	Set $\{f_{rel}, f_{fre}, f_{lik}, f_{toc}\}$ of all objective functions
$w_{j,k}$	Weight of quantified feature value q_k in the objective function f_j
$f_{rel}(u_i)$	Score of plot u_i on the ’relevance’ objective
$f_{fre}(u_i)$	Score of plot u_i on the ‘freshness’ objective
$f_{lik}(u_i)$	Score of plot u_i on the ‘likability’ objective
$f_{daf}(u_i)$	Score of plot u_i on the ‘noteworthiness’ objective

Hyperparameters

ϵ_{fre}	Hyperparameter for calibrating weight of ‘freshness’ objective
ϵ_{lik}	Hyperparameter for calibrating weight of ‘likability’ objective
ϵ_{not}	Hyperparameter for calibrating weight of ‘noteworthiness’ objective
λ_{fre}	Remapped hyperparameter for calibrating weight of ‘freshness’ objective
λ_{lik}	Remapped hyperparameter for calibrating weight of ‘likability’ objective
λ_{not}	Remapped hyperparameter for calibrating weight of ‘noteworthiness’ objective

Evaluation Metrics

$e_{rel}(u_i)$	Score of plot u_i on evaluation metric for ‘relevance’

$s_{rel}(s_j)$	Mean score of plots selected in j-th selection on evaluation metric for ‘relevance’
$d_{rel}(t_x, t_y)$	Mean score of all plots selected in a given duration on evaluation metric for ‘relevance’
E_{rel}	Evaluation metric for measuring efficacy of the system in ensuring ‘relevance’
$e_{fre}(u_i)$	Score of plot u_i on evaluation metric for ‘freshness’
$s_{fre}(s_j)$	Mean score of plots selected in j-th selection on evaluation metric for ‘freshness’
$d_{fre}(t_x, t_y)$	Mean score of all plots selected in a given duration on evaluation metric for ‘freshness’
E_{fre}	Evaluation metric for measuring efficacy of the system in ensuring ‘freshness’
$e_{lik}(u_i)$	Score of plot u_i on evaluation metric for ‘likability’
$s_{lik}(s_j)$	Mean score of plots selected in j-th selection on evaluation metric for ‘likability’
$d_{lik}(t_x, t_y)$	Mean score of all plots selected in a given duration on evaluation metric for ‘likability’
E_{lik}	Evaluation metric for measuring efficacy of the system in ensuring ‘likability’
$e_{not}(u_i)$	Score of plot u_i on evaluation metric for ‘noteworthiness’
$s_{not}(s_j)$	Mean score of plots selected in j-th selection on evaluation metric for ‘noteworthiness’
$d_{not}(t_x, t_y)$	Mean score of all plots selected in a given duration on evaluation metric for ‘noteworthiness’
E_{not}	Evaluation metric for measuring efficacy of the system in ensuring ‘noteworthiness’

1 Introduction

Wellness has emerged as one of the most prominent objectives for many people in the modern world. It involves several sub-components such as physical well-being, financial health, acquiring new skills and so on. For instance, a person employed in a sedentary job might be interested in running to stay active while also wanting to reduce their screen time on the phone. However, it is difficult to consistently strive towards our goals unless we can somehow keep track of our activities in the relevant area. It is therefore said that if we can measure it, we can manage it. Thus, firstly we need data to monitor the activities.

In the general population, wellness-related data collection and analysis are mostly done by the users themselves since it is expensive to hire professionals. Quantified-self or self-trackers refers to the users who do self-tracking and analysis of gathered data for self-knowledge and wellness gains. Quantified-self practitioners and activities being tracked by them are increasing at an exponential pace. The Economist did a cover story on the quantified self movement in March 2022 (in Figure 1.1) and reports that self-tracking tools are going to transform the health sector [1].

The Economist
The Fed: more rate pain awaits
Alibaba and the 40 officials
Why dodgy money loves London
How to save America's Supreme Court
The quantified self
How data can help you live longer

Figure 1.1: The cover page of Technology Quarterly (May 7th 2022) of The Economist with several article on the topic of quantified self [2].

The widespread adoption of wearable technologies such as fitness and sleep trackers in recent years has brought tracking tools to a much greater population. At the same time, portable and wearable computers such as smartphones and smartwatches are offering ever-increasing computational power and user-interaction (UI) capabilities to analyse, visualize and interact with their data. Therefore, a great variety of plots, tables, and notifications can be generated at any given time using data being collected through self-tracking devices.

The amount of personal data and ways to present it are so much today that there is now an explosion in choice of the visualizations that can be used to summarize user data. As argued in the *Paradox of Choice* [3] albeit in a different context, abundance in choice leads to powerlessness and frustration. It stems from choosing 'one' among many other options, which means giving up the rest of the opportunities. Moreover, the time available to a user is limited and it is not possible to sift through all the possible plots and reports that can be generated from their data. Therefore, users would want to have the most useful information presented to them. Achieving this is challenging, as usefulness is a multi-faceted concept.

1.1. Research questions

In this thesis, we address the problem of identifying the most useful information from the abundant activity data streams that can be provided to a user at every interaction with the user-interface of a self-tracking tool. For this purpose, we develop and evaluate an algorithm that can select the most useful data plots (further also referred as **UI Elements**) given the user context, a predefined set of data types and the types of plots presenting that data (also referred to as **UI Element Bank**). In other words, we aim at selecting effectively from a discrete, large, but limited set of data visualizations related to different activities. Towards that end, our work attempts to address the following research questions:

1. What makes an UI Element useful to a user?
2. Given the relevance criteria, what are the effective ways to select an UI Element given the use context?
3. How does the proposed way of selecting UI Elements perform from the perspective of user satisfaction?

1.2. Contribution

In this thesis, we consider the situation in which a user interacts with a self-tracking tool and set out to develop a selection algorithm that can select a limited number of most useful UI Elements out of a large UI Element Bank at every interaction.

To address the first research question, we investigate the possibilities for defining a utility function to quantify the expected usefulness of a plot. This utility function takes as input a number of features representing the plots which we define and model for this purpose. These features take into account, for example, a user's goals, measured activity trends, but also personal preferences on how the information is presented. For the latter, we rely on user interaction, where the user can give positive or negative feedback on the UI Element offered by the system. The features are selected such to reflect the four **aspects of plot usefulness** we focus on in this thesis: *relevance, likability, freshness* and *noteworthiness*.

We approach the second research question of selecting optimal UI Elements by investigating the ways how to solve the utility maximization problem resulting in a ranked list of UI Elements. In particular, we exploit the potential of Integer Programming techniques, for which we argue to be potentially effective for this purpose. The result of our investigation is an optimization engine (**UI Optimizer**) that takes into account user goals, interaction history and user preferences, and the information in the UI Element Bank.

Finally, the third research question is approached by implementing and evaluating our optimization engine. In order to do that, we rely on a self-tracking tool that can analyse a user's activity data from a fitness tracker, generate a UI Element Bank and enable the user to interact with the selected plots. We devise an evaluation framework implemented in Python and develop hypotheses on how the selection algorithm should behave in different scenarios depending on the usage frequency, patterns in user data and feedback provided by the user. These hypotheses are then tested by simulating a number of predefined use scenarios. The plots chosen by the selection algorithm in these simulation are analysed for whether the hypothesized behaviour is observed or not. The results indicate that the performance of the UI Optimizer matches the expectations expressed by the hypotheses in majority of the scenarios, which shows effectiveness of our optimization approach.

1.3. Thesis Outline

The remainder of the thesis is organized as follows. In Chapter 2, we first discuss the overall design of a self-tracking tool and thereafter the components that relate to our proposed optimization engine are further elaborated. In Chapter 3, we define the feature engineering of plots, the overall utility function, the optimization approach used in solving the utility maximization problem, and provide the necessary implementation details. In Chapter 4, we start with describing the validation approach used to evaluate the optimization engine. Thereafter, the results from the validation experiments are provided and discussed. In Chapter 5, we conclude by reflecting on our work and its limitations and give a view on the future work.

2
Background

In this chapter, we first contextualize our work by reviewing the existing literature related to our research. Thereafter, we introduce a system architecture of a self-tracking tool providing the framework for our research. While our study focuses on a self-tracking for limited (health and wellbeing related) input datastreams, the proposed approach for selecting useful UI Elements can be used for any number of data inputs and activity data coming from multiple domains. Similarly, the current approach is designed and developed for a single output device but it can easily be adapted to support optimization across devices.

2.1. Related Work

Much work has been done in the area of UI optimization, including the usage of Integer Programming (IP) techniques, as well the quantified self domain. In this section we summarize the related work that has been done so far.

2.1.1. Quantified-self

As for the quantified-self, the initial thrust of research was focused on understanding the motivation of people who use self-tracking technologies [4] [5]. The primary driving force identified by these studies was self-improvement and make productivity gains. Fawcett [6] reviewed the opportunities and challenges posed by the QS movement. The paper underscores that there is currently a lack of scientifically tested mechanisms for generating insights from the QSelfers data-streams. Swan [7] in 2013 forsees much of the trajectory the quantified-self phenomenon has taken. For example, she emphasized the need for data collection, integration, and analysis using newer models specifically suited for the self-tracking data. Moreover, she posits that the long-term objective of QS systems would be to offer systemic monitoring approach to an individual. However, the paper and subsequent research in the quantified-self domain has not considered the problem of user-interface optimization for QS applications that aggregate multiple self-tracking data streams.

2.1.2. System design for quantified-selfers

Human-Computer Interaction domain has produced a number of design frameworks over the years that are relevant to Quantified-Selfers. Unremarkable computing [8, 9], relevance of context [10], and activity-centric computing [11] offers guiding principles for developing a UI management system. Hassan, Dias, and Hamari [12] employ survey data to argue that quantified-self as a motivational design for pursuing user engagement offers informational feedback as well as affective feedback. In their analysis, gamification, a lot more researched and widely-known class of motivational design, only offers affective feedback. The paper recommends combining several types of feedback in system design to increasing the likelihood of a user perceiving benefits from the use of the system and continue their use of it.

2.1.3. Designing utility function for optimizing user interface

Finally, there is a rich body of literature on user-interface optimization and management. Oulasvirta [13] summarizes model-driven combinatorial approach. Related approaches that have been validated through UI development and subsequent user studies include that of multi-armed bandits [14] and integer programming [15]. Gajos and Weld [16] take a machine learning approach for eliciting user preferences to learn a factored cost function. Coming from a expert systems perspective, Tarkkanen et al. [17] describes an incremental approach to develop the user interface using multi-objective optimization. To sum up, we find several approaches that can be employed for user-interface optimization. However, none of these UI optimization techniques have been implemented in the context of quantified-self applications so far.

2.1.4. Research gaps

Despite considerable work on quantified-self and IP techniques, the prospects of IP techniques in UI optimization for quantified self haven't been examined so far. In this thesis we bring the method of IP technique to the domain of quantified-self.

2.2. Overall system architecture

Figure 2.1 illustrates the overall system diagram for a self-tracking system we developed. The system collects data from self-tracking devices and other services in an aggregated form. This data is analysed to generate data visualizations available as device-own plots (UI Elements). The plots are collected in the UI Element Bank and form the input for the core optimization engine (UI Optimizer). On the output side of the UI Optimizer, the selected plots are delivered by the system to the users via output devices, such as smartphone app, to provide them insights into their daily life. The optimization engine has the task to intelligently pick the most useful plots. It is designed to use plot characteristics from the UI Element Bank, pre-specified interest areas by the user in form of User Goals, and the information from previous user interactions (Interaction History) to make the optimal selection. More specifically, the optimization engine takes five inputs that we explain in more detail in the subsequent section: a) From the UI Element Bank: Plot Features (further denoted by M) and Plotted User Data (further denoted by P); b) User Goals (further denoted by G); c) from the Interaction History: Selection Log (further denoted by S) and Interaction Log (further denoted by I). The optimization engine itself is the topic of the next chapter.

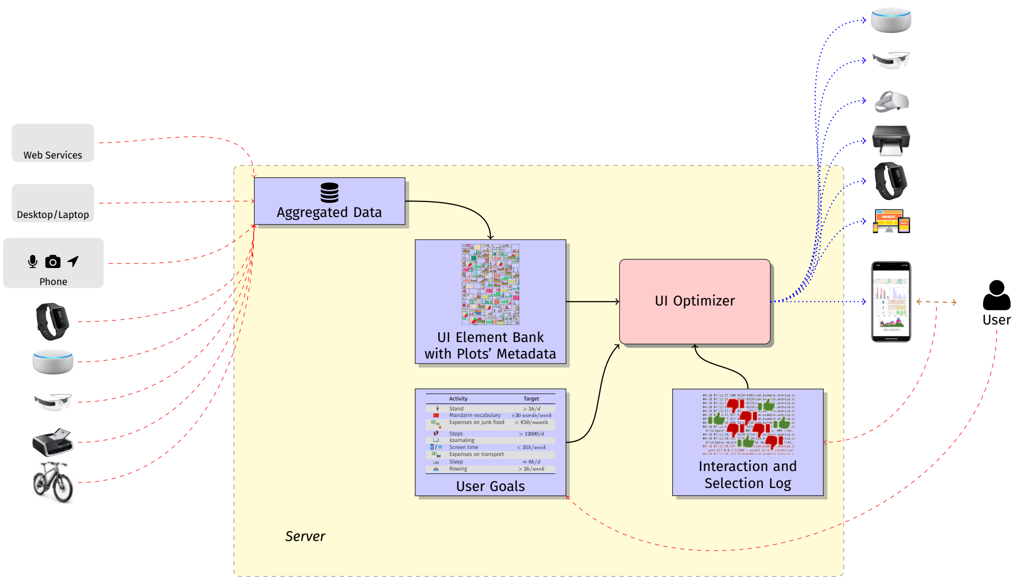


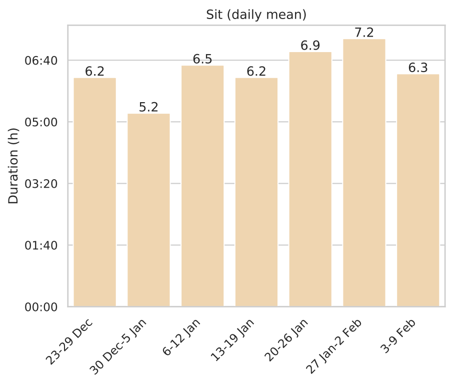
Figure 2.1: Overall schematic diagram for an automated quantified-self information system

2.3. Data sources for the optimization engine

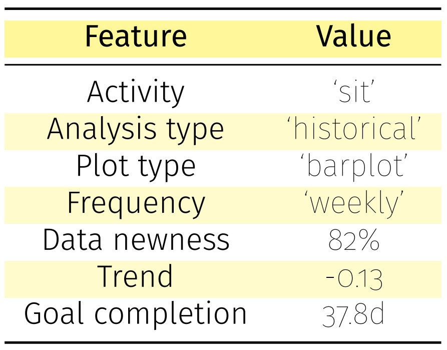
We now discuss and illustrate each of the five components of the self-tracking system that acts as input to the optimization engine as discussed in the previous section. An overview of these components will help us in understanding the data available to the optimization system to make the plot selection.

2.3.1. Plots Features (M)

Every plot that is generated using an available datastream has certain characteristics associated with it, which are represented by features. For example, Figure 2.2a shows a bar-plot for the number of hours a user has been sitting for the past several weeks. The table in Figure 2.2b summarizes the features of this bar plot. In the system architecture deployed by us, the visualization generator that parses the aggregated user data to UI elements also exports machine-readable features for each element for subsequent usage by the optimization engine.



(a) An example plot in UI element bank



Feature	Value
Activity	'sit'
Analysis type	'historical'
Plot type	'barplot'
Frequency	'weekly'
Data newness	82%
Trend	-0.13
Goal completion	37.8d

(b) Features associated with the adjacent plot

Figure 2.2: An example of plot and associated plot features

2.3.2. Plotted User Data (P)

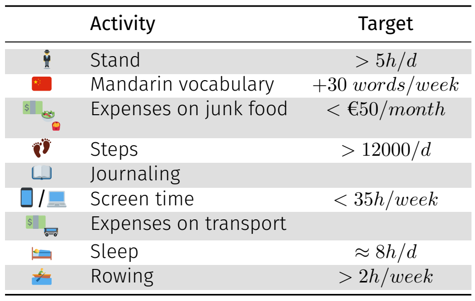
Plots having time series data can be analysed by the optimization engine for noticing any trends. As illustrated by Table 2.3, current user data in plots such as barplot, areaplot etc. are therefore logged in a separate table of plotted user data (P) that holds the numerical values of the depicted data for usage by optimization engine.

Table 2.1: Example of plotted user data (P) at a given time

Plot ID	Plotted Data Values
1	{3.4, 3.1, …4.4}
2	ϕ
.	.
.	.
542	{12.4, 10.5, …12.2}

2.3.3. User Goals (G)

Users logging their daily activities typically have some goals in mind. They also typically use some numerical targets that can help them quantify their goals in the area of life that they want to monitor or improve upon. For example, a person having a desk job who wants to avoid prolonged hours sitting (goal) can do so by keeping a tab of the number of hours they spend sitting using a smartwatch and by setting a target, for example, to stand at least 5 hours a day. As shown in Figure 2.4, the goals could be for various domains of life as long as a data stream is available to monitor it. In our system, we use such explicitly provided targets to improve the UI Element selection by prioritizing plots that capture one or more of these targets.



	Activity	Target
🧍	Stand	$> 5h/d$
🇨🇳	Mandarin vocabulary	$+30\ words/week$
💵🍔	Expenses on junk food	$<$ €$50/month$
👣	Steps	$> 12000/d$
📖	Journaling	
📱/💻	Screen time	$< 35h/week$
💵🚌	Expenses on transport	
🛏	Sleep	$\approx 8h/d$
🚣	Rowing	$> 2h/week$

Figure 2.3: An example of user-goals and the corresponding targets

2.3.4. Interaction Log (I)

While the 'User Goals' data source serves to steer the selection of the relevant plots, users could still have preferences for one plot over another. Therefore, another mechanism is required to learn the user preferences in a dynamic, but non-intrusive way through continuous interactions to help the system to better serve the user over time .

In our prototype system, the feedback is taken using the "like" or "dislike" button that can optionally be clicked by the users to suggest whether a plot was useful to them or not. The feedback could potentially also be taken in another way, such as swiping left and right, or by tracking the time spent on looking at each plot. To keep the UI and selection algorithm simple, the likes/dislikes are attributed to all features of the plot in question. Our selection algorithm is designed to make use of the collected logs for adapting to the user taste as will be detailed in the next chapter.

Table 2.2: A sample log of interaction data (I)

Session ID	Plot ID	Rating
1	14	+
1	52	+
.	.	.
.	.	.
2	334	+
2	13	-
3	245	+
.	.	.
.	.	.
53	242	-

2.3.5. Selection Log (S)

Additionally to the Interaction Log, a log of all plots that were presented to the user is also kept even if no explicit feedback was received in terms of ”like”/”dislike”. This log helps in keeping track of what information has already been delivered to the user in the recent past. More specifically, the system keeps a log of plot indices that are selected whenever a user interacts with the user interface system. Table 2.3 illustrates the data fields that are logged.

Table 2.3: A sample log of selection log (S) data

Session ID	Start Time	Session Duration	Selected Plots IDs
1	`2020-01-15 12:18:30`	23	{14, 52, …, 367}
2	`2020-01-15 16:32:56`	53	{334, 123, …, 13}
3	`2020-01-16 07:18:30`	35	{52, 245, …, 62}
.	.	.	.
.	.	.	.
53	`2020-01-45 15:23:63`	85	{67, 3, …, 242}

In the above table, the field *Session ID* keeps a unique value for each interaction while *Start Time* logs the time when user refreshes the plot selection (i.e., starts a new session). *Session Duration* logs the session duration (in seconds) while *Selected Plots* holds the ranked indices of k plots and the order in which they were displayed to the user during that session.

3 Method

Designing a Graphical User Interface (GUI) for Quantified-Self applications can entail various design tasks, such as functionality selection, icon selection, menu design, grid layout or widget selection, to name a few. In this thesis, we focus on a simple user interface where users are presented with a fixed number of plots of their logged data per interaction with the self-tracking system, with each button having ”like” and ”dislike” button that can be optionally interacted with. There are no icons, menus or complex grid layouts. As depicted in Figure 3.1 (right), we consider the plots presented column-wise, ranked according to their match with the selection criterion. The size of all the selected plots is the same and there is only one output device. In other words, we use a simple user interface layout where the only variables are the plot-selection criteria and where the main task is to optimize the plot selection given the criteria.

In the subsequent sections, we first narrow down conceptually on the optimization approach that would be most suitable for the problem at hand. Thereafter, we propose the selection criteria that reflect our goal to select the most useful plots for the user in the given context. Finally, we translate these criteria into the optimization method we implement.

3.1. Plot selection as combinatorial optimization problem

Optimization is the process of searching the solution space for candidates that yield the most desirable value with regard to some criterion. Combinatorial optimization is a sub-field of optimization that consists of finding an optimal object from a finite set of objects, where the set of feasible solutions is discrete or can be reduced to a discrete set [18] [19]. Over years, combinatorial optimization has emerged as a flexible and powerful tool for computational generation and adaptation of GUIs [20].

Combinatorial optimization is especially suited to decide on the GUI design and interactions. It is distinguished by its algorithmic capacity, controllability, and generalizability. In comparison to formal methods, such as logic, combinatorial optimization offers an effective but a flexible way of expressing design knowledge and objectives in a computable manner. Compared to data-driven approaches based on machine learning, such as artificial neural networks, combinatorial optimization allows one to control the design outcomes via specific design objectives [21].

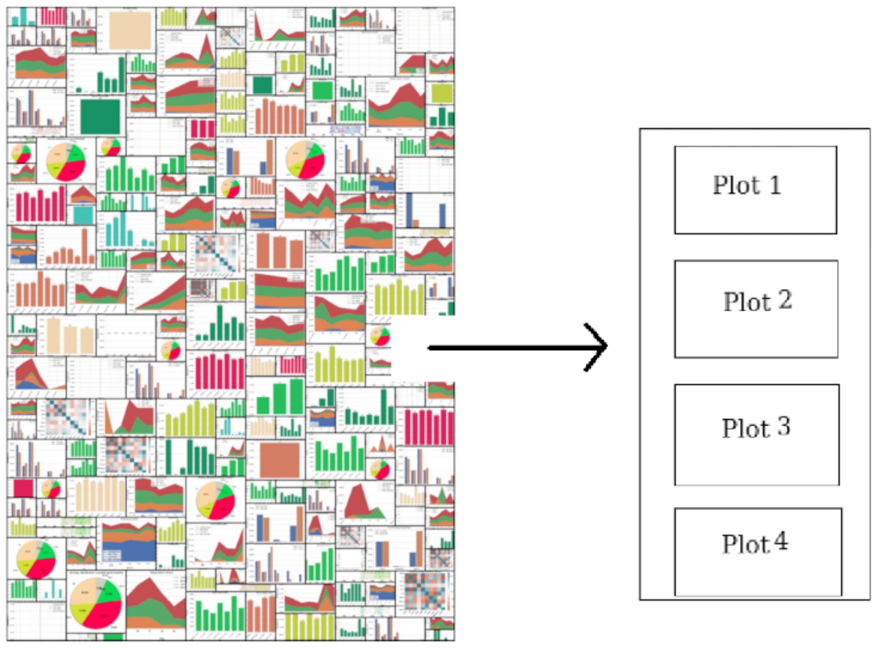


Figure 3.1: An illustration of the interaction context with the selected plots: plots extracted from the UI Element Bank are presented to the users column-wise, ranked according to their match with the selection criterion.

3.1.1. Selection problems in GUI design

Selection problems in GUI design involve choosing a set of given elements $U = \{u_1, ..., u_n\}$, with each element u_i being equipped with a nonempty set of attributes $A_i \subseteq A$, $A = \{a_1, ..., a_m\}$, to meet given requirements while optimizing one or multiple objective functions. Oulasvirta et al. [21] divides the selection problem for UI design in four sub-types. Among them, the *packing* and *covering* problems are of most interest to us.

In packing problems we have positive capacities c_j for each attribute a_j, and only those subsets of U are feasible that do not exceed the capacity for each attribute. On the other hand, covering problems have have a certain requirement r_j for each attribute a_j. The task is to select a subset of U that satisfies all requirements; that is, the number of selected elements with attribute a_j is at least r_j for each attribute. [21]

In packing problems, the goal is to find a selection of the elements that do not exceed the capacities and maximizes the total profit while in covering problems, the goal is to make a selection that minimizes the total costs.

3.1.2. Plot selection as *packing*-type widget selection problem

The plot selection problem is closest to widget selection problem discussed in the literature [21]. In the widget selection problem, there are multiple widgets, or widget types, but there is only a limited-size canvas to place them on. The plots generated in a Quantified Self's system can be considered as widgets of same type but each plot/widget having different attributes. The objective of widget selection problems (or plot selection problem in our case) can be formulated as both packing and covering problems, as discussed in previous section.

We argue that, the plot selection problem that we aim to take up in this project are best suited to be formulated as packing problems. The goal of widget selection problem formulated as a packing problem is to find a selection of elements that, without exceeding the capacities, maximizes some positive value, such as usefulness to end users.

3.2. Linear Programming for solving optimization problems

The optimization problem addressed in this thesis can be formulated as a *linear program*, in which the set of feasible solutions is formed by finitely many constraints and where the constraints are linear. Linear programs can be expressed in canonical form as

$$\text{Find a vector} \quad \mathbf{x} \tag{3.1}$$

$$\text{that maximizes} \quad \mathbf{c}^\mathsf{T}\mathbf{x} \tag{3.2}$$

$$\text{subject to} \quad A\mathbf{x} \leq \mathbf{b} \tag{3.3}$$

$$\text{and} \quad \mathbf{x} \geq \mathbf{0}. \tag{3.4}$$

Here the components of $\mathbf{x}$ are the variables to be determined, $\mathbf{c}$ and $\mathbf{b}$ are given vectors, and A is a given coefficient matrix. The function whose value is to be maximized or minimized ($\mathbf{x} \mapsto \mathbf{c}^\mathsf{T}\mathbf{x}$ in this case) is called the *objective function*. The inequalities $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ are the constraints which specify a convex polytope over which the objective function is to be optimized.

Linear programming problems can be solved by various methods, one of which is the *simplex method* [22]. In *integer linear programming* (ILP), some or all of the variables of the linear program are restricted to be integers [23], as is the case in our research context.

3.3. Multi Objective Linear Programming

The usefulness of a plot is a multi-faceted criterion (see next section for details), which translates into a multi-objective optimization scenario. In such a scenario, the resulting objectives f_i for $i = 1, \ldots, k$ may conflict with each other. A solution that is optimal for one objective may perform poorly for others. One common technique in solving multi-objective linear programming problems is to combine the objectives into a single objective expression [24]. This can be done in many ways, for example, by using weighted sums, the lexicographic method or goal-programming methods. The resulting problem can then be solved via well-established algorithms [25]. Alternatively, one function f_j can be considered as the main objective function, which is to be optimized given the bounds of values of the other objective functions as constraints, i.e., $f_i(x) \leq \varepsilon_i$. This method is usually referred to as the ε-constrained method [26]. Mavrotas [27] argues that later has several advantages over the weighing method and weighting method cannot produce unsupported efficient solutions in multi-objective integer and mixed integer programming problems. For this project, we therefore use ε-constrained method to solve the multi-objective integer linear programming formulation of our research problem.

3.4. Modeling the plot selection problem

The main goal of plot selection is to identify plots that are most useful in a given interaction. As discussed in Section 2.1, usefulness (i.e. utility function) can be modeled in various ways [13]. In this thesis we map usefulness on four criteria, for which we believe to reflect best the needs of the users in a quantify-self application context. Furthermore, following the principle of the ε-constrained optimization method, within these four criteria we select one of them as the main one (the primary optimization objective) and the other three as constraints (the secondary optimization objectives). We elaborate on these criteria in the following subsections.

3.4.1. Primary optimization objective

In our project, the primary concern is that to find plots that are most relevant to user goals. If a user has explicitly marked ”number of steps taken on a day” as an activity they want to particularly focus on and perhaps also have goal about it, the selection system should prioritize plots that have information on this activity. We denote this criterion by **relevance** (f_{rel}) and consider it the necessary condition for a successful interaction with a self-tracking system.

3.4.2. Secondary optimization objectives

The GUI of a quantified self is often accessed by its user multiple times a day. If we only have relevance of plots to user’s goals as the sole criterion, it is likely that the same types of plots gets selected over and over again. This may introduce dullness in the information presented to the user and result in the users gradually losing interest, decreasing their interactions and thus losing track of their activity data. In order to prevent this, we propose to have the optimizer prioritize plots that bring **freshness** (f_{fre}) in plot selection. For example, if a barplot on ”hours of sleep a day” during last week has already been displayed during an interaction, there should be some penalty in choosing it again in a subsequent interaction.

Our user interface allows a user to provide their feedback (using ”like” and ”dislike” button provided for each plot) to how useful they found a plot. Leveraging this functionality, if we find in interaction log that a user is consistently rating line-charts positively and pie-charts negatively, we should be prioritizing line-charts in the selection. Therefore, we would like to prioritize selection of plots that are expected to have high **likability** (f_{lik}), as inferred from the user preferences.

As discussed in previous chapter, the quantified-self system proposed for this project has a visualization generator that also provides metadata on the generated plots. For instance, the data depicted in a bar-plot for weekly average of kilometers ran in the last month is available as a list of y-values such as $[12.2, 10.2, 6.3, 5.4, 5.6]$. This information can be used to evaluate whether the depicted data has a certain pattern to it, such as increasing/decreasing trend. It is likely that plots having such **noteworthiness** (f_{not}) in terms of displayed data would be found useful by the users. We thus consider it as another criterion while selecting plots for display.

3.4.3. Formulation of the optimization problem

For every plot, a decision needs to be made whether it is chosen or not in the final selection. We therefore have a binary integer variable x_i for each plot u_i that takes value of 0 or 1 where 0 signifies that plot is not selected while 1 signifies that plot is selected, i.e. $x_i \in \{0, 1\} \quad \forall x_i \in X$.

We have only one constraint where we limit the total number of plots that are to be selected. In other words, we want to select k plots from the full set of n ($n = |U|$) plots:

$$\sum_{i=1}^{n} x_i = k$$

As discussed before, we use the ε-constrained method to solve the multi-objective optimization problem by choosing one criterion $\sum_{i=1}^{n} x_i \times f_{rel}(x_i)$ as the main optimization objective and bind the values of the other objective functions f_k in the constraints as follows:

$$\sum_{i=1}^{n} x_i \times f_{fre}(u_i) \geq \varepsilon_{fre},$$

$$\sum_{i=1}^{n} x_i \times f_{lik}(u_i) \geq \varepsilon_{lik}, \text{and}$$

$$\sum_{i=1}^{n} x_i \times f_{not}(u_i) \geq \varepsilon_{not}.$$

The values for ε-parameters (i.e. ε_{fre}, ε_{lik}, and ε_{not}) are hyper-parameters of our optimization model. They are discussed in detail in Section 3.7. The modified optimization problem, after applying the ε-constrained method can now be formulated as follows:

$$\text{maximize} \sum_{i=1}^{n} x_i \times f_{rel}(u_i) \tag{3.5}$$

$$\text{such that} \sum_{i=1}^{n} x_i \times f_{fre}(u_i) \geq \varepsilon_{fre}, \tag{3.6}$$

$$\sum_{i=1}^{n} x_i \times f_{lik}(u_i) \geq \varepsilon_{lik}, \tag{3.7}$$

$$\sum_{i=1}^{n} x_i \times f_{not}(u_i) \geq \varepsilon_{not}, \tag{3.8}$$

$$\sum_{i=1}^{n} x_i = k, \tag{3.9}$$

$$x_i \in \{0,1\} \quad \forall x_i \in X \tag{3.10}$$

Objective functions $f_k \in F$, with $F = \{f_{rel}, f_{fre}, f_{lik}, f_{not}\}$, are high-level in nature. In the next sections we identify and formulate a set Q of seven lower-level plot features (q_j) which become the building blocks for these objective functions. Each objective function (i.e. f_{rel}, f_{fre}, f_{lik}, and f_{not}) is thereafter taken as a linear combination of a subset of plot features that have a bearing on that objective. In mathematical terms, the relationship between an objective functions (f_k) and plot features (q_j) in general can be formulated as follows

$$f_k(u_i) = \frac{\sum_{j \in Q} w_{i,j} \times q_j(u_i)}{\sum_{j \in Q} w_{i,j}} \qquad \forall u_i \in U, \quad \forall f_k \in F \quad \text{and} \quad \forall q_j \in Q \tag{3.11}$$

In the above equation, $w_{i,j}$ is the weight of plot feature q_j in the value of the objective function f_k, for which it holds $w_{i,j} \in [0,1]$ and $\sum_{j \in Q} w_{i,j} = 1$.

The definition of the plot features in section 3.5 will be followed by the introducion of the weights $w_{i,j}$ in Section 3.6. Finally, we will revisit the modified optimization problem discussed in previous section and values of hyper-parameters ε_k in Section 3.7.

3.5. Plot features

Every plot in the UI element bank has several properties, also referred to as features, such as type of plot, frequency of data depicted, x-values, y-values, etc. In this project we identify and mathematically formulate seven plot features which can act as approximations for the high-level plot selection criteria (i.e. relevance, likability, freshness, and noteworthiness). In order to keep the optimization problem computable, we quantify the categorical properties so that we only have to deal with numerical values.

We introduce the plot features using the following toy example. Let us suppose a person uses an activity tracker to capture their activity data. Using this data, the visualization engine creates a plot shown in Figure 3.2, which depicts the average number of hours per day a person spends time sitting over consecutive weeks. Table 3.1 shows raw values of the features for this plot collected from the

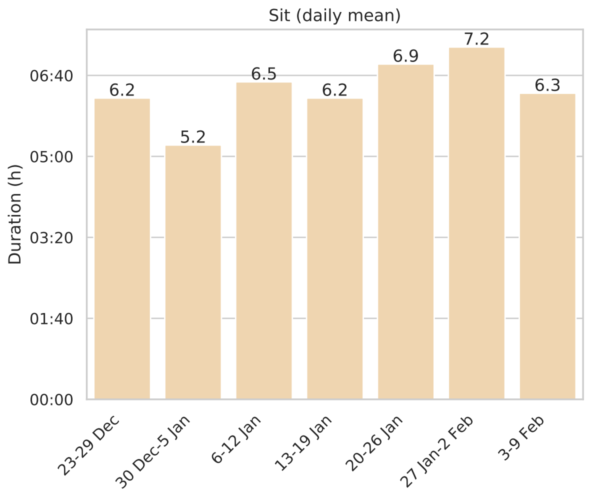


Figure 3.2: An example plot of user data depicting weekly average hours spent sitting as gathered from a fitness tracker

different data sources introduces in the previous chapter (M, G, I, S, and P) . In the remainder of this section, we briefly describe each of the seven features used in our optimization system and how raw values are quantified (using q_j) so that the resulting numeric values can be used to compute objective functions f_k. For the example plot provided in Figure 3.2, the numeric values for the features are listed in *'Quantified feature value'* column of Table 3.1.

3.5.1. Activity (q_{act})

This feature captures the user's activities depicted in the plot. This field can have one or more values. For example, in Figure 3.3 first plot has ten activities while the second plot only has five.

A plot is more relevant if many of the activities depicted in the plot are part of user-defined goals (i.e. activities of interest) as discussed in previous chapter. If we denote by $A_G = \{a | a \in G'\}$ the set of activities present in user goals, and by $A_i = \{a | a \in M_{act}(u_i)\}$ the set of activities present in plot i then we can quantify the activity features using the following expression:

$$q_{act}(u_i) = \frac{|A_i \cap A_G|}{|A_i|}$$

As an example, let's take a plot u_i showing data about three activities such that $A_i = \{'sit', 'walk',$

Table 3.1: Raw and quantified feature values for the example plot in Figure 3.2

Feature	M, G, I, S, P	Raw feature values	$q_j(u_i)$	Quantified feature value

Activity	$M_{act}(u_i)$
	G_{all}
		{‘sit’}
		{‘sit’, ‘walk’, ‘meditate’}
			$q_{act}(u_i)$	1.00

Plot type	$M_{plt}(u_i)$ $I^+_{M_{plt}(u_i)}$ $I^-_{M_{plt}(u_i)}$	‘barplot’ 5 10	$q_{plt}(u_i)$	0.33
Analysis type	$M_{ant}(u_i)$ $I^+_{M_{ant}(u_i)}$ $I^-_{M_{ant}(u_i)}$	‘historical’ 6 6	$q_{ant}(u_i)$	0.50
Frequency type	$M_{frt}(u_i)$ $I^+_{M_{frt}(u_i)}$ $I^-_{M_{frt}(u_i)}$	‘weekly’ 7 3	$q_{frt}(u_i)$	0.70
Trend	d_i	{6.2 ,5.2 …, 6.3}	$q_{tre}(u_i)$	0.13
Goal completion	$P_{data}(u_i)$ $M_{frt}(u_i)$ G_j^{Target}	{6.2 ,5.2 …, 6.3} ‘weekly’ 8.0	$q_{goc}(u_i)$	0.83
Data freshness	$P_{t_{end}}$ $P_{t_{start}}(u_i)$ $S_{t_{last}}(u_i)$	2020-02-09 13:46:34 2019-12-23 00:00:00 2020-01-15 17:24:23	$q_{daf}(u_i)$	0.72

$'meditate'\}$, while the user is having goals for four activities such that $A_G = \{'walk', 'sleep', 'meditate', 'phone-usage'\}$. In this example, two (i.e. $'walk'$ and $'meditate'$) of three activities in the current plot are part of user goals. Using the above relation, $q_{act}(u_i)$ for this plot will be 0.67. $q_{act}(u_i)$ takes value between 0 and 1, inclusive of the limits. For any given plot, the value for this feature may change if user goals are updated.

3.5.2. Plot type (q_{plt})

Plot type represents the type of visualization used to represent data. The main thrust of this feature is that if a user has expressed strong preference for a particular plot type through the ratings in previous interactions, we should prioritize the plots having that plot type for selection in subsequent interactions. Therefore, we make use of the following variables to calculate how many times the current plot type ($M_{plt}(u_i)$) has been positively or negatively rated by a user in previous interactions:

- $I^+_{M_{plt}(u_i)}$ is the number of times a user has positively rated plots having the same *plot type* as in plot u_i (i.e. $M_{plt}(u_i)$).

- $I^-_{M_{plt}(u_i)}$ is the number of times a user has negatively rated plots having the same *plot type* as in plot u_i (i.e. $M_{plt}(u_i)$).

These values can be gathered from the *interaction log* (I) data source discussed in the previous chapter. Making use of above definitions, we define the feature *plot type* as follows:

$$q_{plt}(u_i) = \frac{I^+_{M_{plt}(u_i)}}{max(1,\ I^+_{M_{plt}(u_i)} + I^-_{M_{plt}(u_i)}}$$

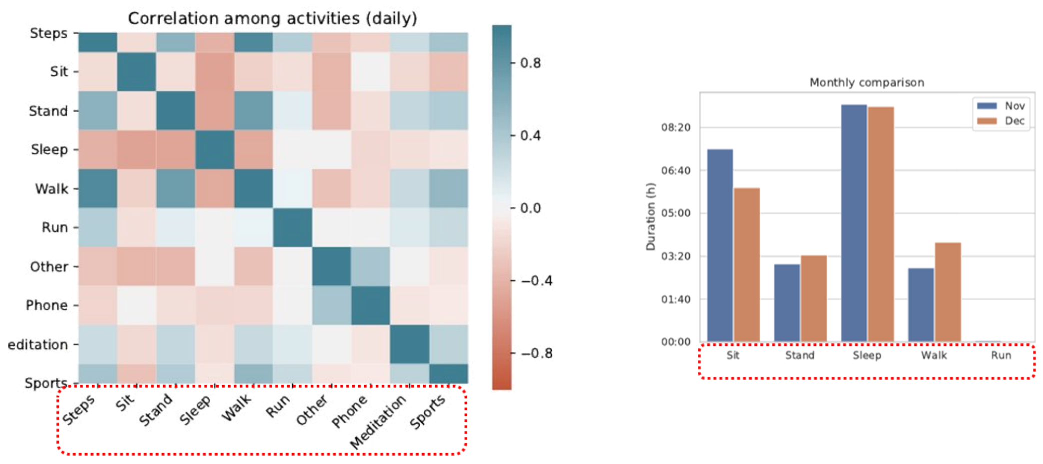
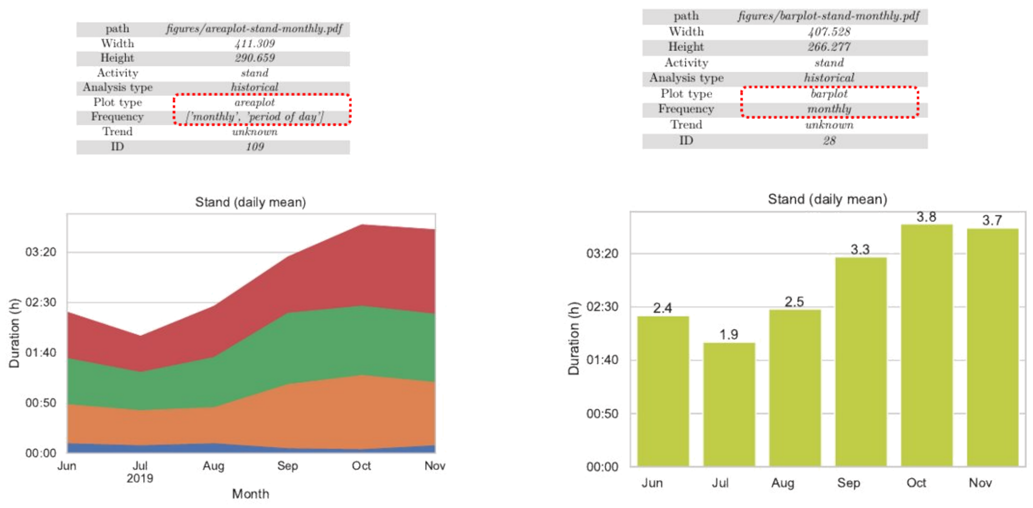


Figure 3.3: Example plots showing information about different subsets of activities to illustrate *Activity* feature (q_{act})



path	*figures/areaplot-stand-monthly.pdf*
Width	*411.309*
Height	*290.659*
Activity	*stand*
Analysis type	*historical*
Plot type	*areaplot*
Frequency	*['monthly', 'period of day']*
Trend	*unknown*
ID	*109*

path	*figures/barplot-stand-monthly.pdf*
Width	*407.528*
Height	*266.277*
Activity	*stand*
Analysis type	*historical*
Plot type	*barplot*
Frequency	*monthly*
Trend	*unknown*
ID	*28*

Figure 3.4: Two different types of visualizations to illustrate the *Plot type* feature (q_{plt}): on the left is a stacked area plot and on the right is a barplot.

We illustrate this feature on the two example plots in Figure 3.4. If a plot u_i being of the *'barplot'* type (i.e. $M_{plt}(u_i) = \text{'}barplot\text{'}$). has been displayed 28 times in previous interactions, out of which six times it got positively rated (i.e. $I^{+}_{M_{plt}(u_i)} = 6$) and four times negatively (i.e. $I^{-}_{M_{plt}(u_i)} = 4$), the above equation tells us that the $q_{plt}(u_i)$ for the plot u_i will be 0.6. q_{ant} takes value between 0 and 1, inclusive of the limits. For any given plot, the value for this feature will change at the next interaction if a user gives positive/negative rating to a displayed plot.

3.5.3. Analysis type (q_{ant})

The same type of plot can show different type of information. For instance, a barplot can show historical data for one particular activity. In an another type of visualization, barplot can be used to depict categorical distribution of time across different activity categories. Figure 3.5 illustrates a couple of analysis types.

Similar to the previous feature, we make use of the interaction log of user ratings for quantifying this feature. If a user has expressed strong preference for a particular analysis type through the ratings in previous interactions, we should prioritize the plots having that analysis type for selection in subsequent

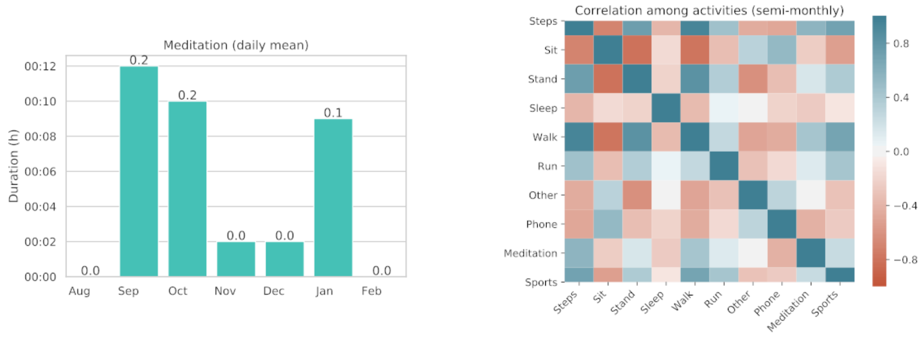


Figure 3.5: Example plots to illustrate the *Analysis type* feature (q_{ant}): plot on the left shows 'historical' analysis of meditation data while the one on the right depicts 'correlation' between different activities.

interactions. Therefore, we make use of the following variables to calculate how many times a plots of the current analysis type ($M_{ant}(u_i)$) have been positively or negatively rated by a user in previous interactions:

- $I^{+}_{M_{ant}(u_i)}$ is the number of times a user has positively rated plots having the same *analysis type* as in plot u_i (i.e. $M_{ant}(u_i)$).
- $I^{-}_{M_{ant}(u_i)}$ is the number of times a user has negatively rated plots having the same *analysis type* as in plot u_i (i.e. $M_{ant}(u_i)$).

Making use of above definitions, we define the feature *analysis type* as follows:

$$q_{ant}(u_i) = \frac{I^{+}_{M_{ant}(u_i)}}{max(1,\ I^{+}_{M_{ant}(u_i)}\ +\ I^{-}_{M_{ant}(u_i)})}$$

If a plot u_i of the *'historical'* analysis type (i.e. $M_{ant}(u_i)\ =\ 'historical'$) has been displayed 15 times in previous interactions, out of which seven times it got positively rated (i.e. $I^{+}_{M_{ant}(u_i)}\ =\ 3$) and seven times negatively (i.e. $I^{-}_{M_{ant}(u_i)}\ =\ 7$), the above relation tells us that the $q_{ant}(u_i)$ for the plot u_i will be 0.3. q_{ant} takes value between 0 and 1, inclusive of the limits. For any given plot, the value for this feature will change at the next interaction if a user gives a positive/negative rating to a displayed plot.

3.5.4. Frequency type (q_{frt})

The same type of plot and analysis type can be deployed for different frequency intervals. For instance, an areaplot having historical data for average sleep hours can have frequency (i.e. aggregation) levels as daily, weekly, fortnightly, monthly, and so on. Figure 3.6 provides example of two such such frequency types.

Similar to the previous two features, we make use of interaction log of user ratings for quantifying this feature. If a user has expressed strong preference for a particular frequency type through the ratings in previous interactions, we should prioritize the plots having that frequency type for selection in subsequent interactions. Therefore, we make use of following variables to calculate how many times a plot of the current frequency type ($M_{frt}(u_i)$) has been positively or negatively rated by a user in previous interactions:

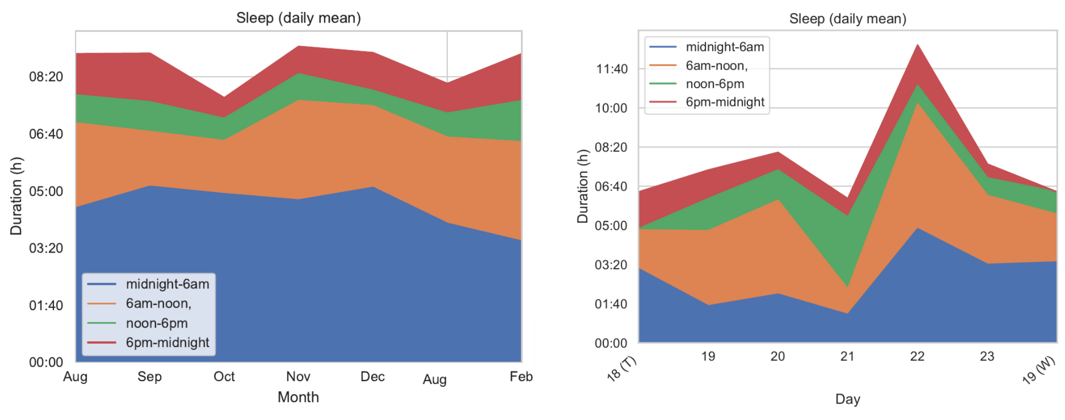


Figure 3.6: Example plots to illustrate *frequency type* feature (q_{frt}): plot on the left show sleep data in an areaplot with *'daily'* frequency while the one on right depicts sleep data in an areaplot with *'monthly'* frequency.

- $I^{+}_{M_{frt}(u_i)}$ is the number of times a user has positively rated plots having the same *frequency type* as in plot u_i (i.e. $M_{frt}(u_i)$).
- $I^{-}_{M_{frt}(u_i)}$ is the number of times a user has negatively rated plots having the same *frequency type* as in plot u_i (i.e. $M_{frt}(u_i)$).

Making use of the above definitions, we define the feature *frequency type* as follows:

$$q_{frt}(u_i) = \frac{I^{+}_{M_{frt}(u_i)}}{max(1,\ I^{+}_{M_{frt}(u_i)} + I^{-}_{M_{frt}(u_i)})}$$

If a plot u_i of the *'monthly'* frequency type (i.e. $M_{frt}(u_i) = 'monthly'$) has been displayed 20 times in previous interactions, out of which five times it got positively rated (i.e. $I^{+}_{M_{frt}(u_i)} = 5$) and five times negatively (i.e. $I^{-}_{M_{frt}(u_i)} = 5$), the above relation tells us that the $q_{frt}(u_i)$ for the plot u_i will be 0.5. q_{frt} takes value between 0 and 1, inclusive of the limits. For any given plot, the value for this feature will change at the next interaction if a user gives a positive/negative rating to a displayed plot.

3.5.5. Trend (q_{tre})

Some visualizations are conducive to easily allow the quantification of information depicted by them. For instance, we can calculate the slope of a historical barplot to know the overall trend in the data. This is an example of a trend feature, which we illustrate in figure 3.7 by three examples.

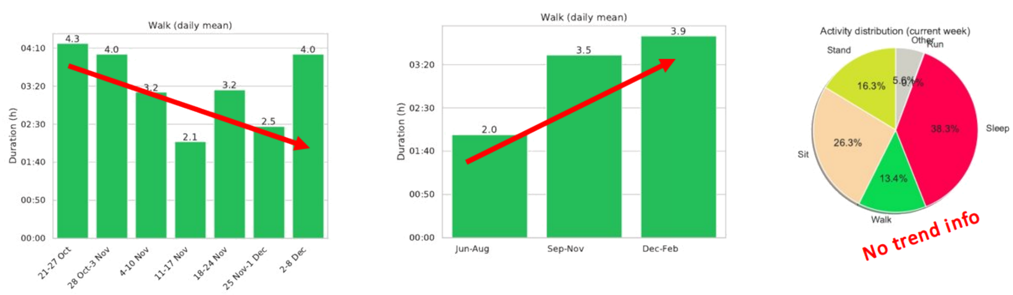


Figure 3.7: Example plots to illustrate the *trend* feature (q_{tre}): the first figure has an overall negative slope (i.e trend), second has a positive trend, and no trend value can be determined for the third type of plot.

We make use of the following information in order to model the trend feature:

- d_i is the array of time-series values displayed in a plot i. $P_{data}(u_i)$ can be gathered from the *Plotted user data* (P) data source discussed in the previous chapter.

- $trend(\mathbf{y}) = \beta$, where β is the slope obtained from fitting a list of values $\mathbf{y}$ to a simple regression model (i.e. $\mathbf{y} = \alpha + \beta\mathbf{x}$) using the least-squares approach.

We define the feature *trend* as follows:

$$q_{tre}(u_i) = \begin{cases} \frac{|max(min(trend(P_{data}(u_i)), 2), -2)|}{2}, & \text{if } |P_{data}(u_i)| > 1 \\ \phi, & \text{otherwise} \end{cases}$$

In the context of this thesis, we take modulus of slope values with the understanding that both positive and negative slopes would be interesting for the user. A plot having zero slope will have zero value for the quantified feature and the maximum value (i.e. 1) is attained by a plot when either $trend(\mathbf{y}) \leq -2$ or $trend(\mathbf{y}) \geq 2$. This capping ensures that slope values for unusually steep plots remain comparable to the values for plots having gentle slopes. If we now take the centre plot in Figure 3.7 as an example, the list of values displayed in the plot is such that $P_{data}(u_i) = \mathbf{y} = \{2.0, 3.5, 3.9\}$ and $\mathbf{x} = \{-2, -1, 0\}$. Using least-squares approach, we can calculate that $trend(\mathbf{y}) = \beta = 0.95$, such that $\mathbf{y} = \alpha + \beta\mathbf{x}$. Finally, using the above relation tells us that $q_{tre}(u_i)$ for the plot u_i will be 0.475.

When computable, q_{tre} takes value between 0 and 1, inclusive of the limits. For any given plot, the value for this feature may change at the next interaction if user data that is displayed in the plot gets updated.

3.5.6. Goal completion (q_{goc})

With the *'goal completion'* feature, we go one step further from the previously discussed *'Trend'* features. Here we quantify how far a user is from their specified goal value. Using the slope (β) calculated in the previous section, we solve the linear equation to find the expected time required to complete the goal. The feature can only be calculated where the target value is available for the activity and the slope value can be calculated from the data depicted in the plot. Moreover, the same activity can have different goal completion values based on the aggregation level and the values displayed in a plot.

In order to quantify the goal completion feature, we rely on the following information:

- d_i is the array of time-series values displayed in plot i. d_i can be gathered from *plotted user data* (P) data source discussed in previous chapter.

- α and β are, respectively the y-intercept and the slope obtained from fitting $\mathbf{x}$ and $\mathbf{y}$ to a simple regression model (i.e. $\mathbf{y} = \alpha + \beta\mathbf{x}$) using the least-squares approach.

- G_i^* is the target value for goal activity G_i as defined by the user in the *user goals* (G) data source.

Primarily, we want to find the duration in which a user will be able to achieve their goal (i.e. G_i^*) going by the current trend. We make use of the relation $y = \alpha + \beta x$, to find the x^* value by substituting y with G_i^* and using α and β values calculated previously. As a result, we get the $x^* = \frac{G_i^* - \alpha}{\beta}$.

To make the *goal completion* feature comparable across all frequency types, we make use of the plot frequency type (i.e. $M_{frt}(u_i)$) to calculate the number of days ($days(x)$) in each interval of the plots, which is in turn used to normalize the feature value:

$$days(x) = \begin{cases} 1, & \text{if } x = daily \\ 7, & \text{if } x = weekly \\ 30, & \text{if } x = monthly \\ 90, & \text{if } x = quarterly \\ 365, & \text{if } x = annually \end{cases}$$

The above function is used to calculate the final feature value using following the formuation:

$$q_{goc}(u_i) = \begin{cases} \frac{max(min(days(M_{frt}(u_i))*\frac{G_i^* - \alpha}{\beta}),365),0)}{365}, & \text{if } |A_i| = 1 \\ & \text{and } A_i \in G \\ & \text{and } G_i^* \neq \phi \\ & \text{and } |d_i| > 1 \\ & \text{and } |\beta| > 0 \\ \phi, & \text{otherwise} \end{cases}$$

The *if* condition in the above formulation essentially means the following:

1. a plot should have only one activity,
2. the activity should be part of user goals,
3. a user should have defined a target value for that activity,
4. there should be at least two values being displayed on the plot so that α and β can be calculated, and
5. the slope should not be zero.

We illustrate this feature on the example of the first two plots in Figure 3.7, that we will refer in the following as the left and the right plot. The list of values displayed in the right plot is such that $P_{data}(u_i) = \mathbf{y}\{2.0, 3.5, 3.9\}$ and $\mathbf{x} = \{-2, -1, 0\}$. Using the least-squares approach, we can calculate that $\alpha = 4.08$ and $\beta = 0.95$. Since, it is a plot with *'quaterly'* frequency type, $days(M_{frt}(u_i)) = 90$. Now, let us suppose *'walk'* is an activity part of user goals (G) and the user has a target of walking of 5 hours every day (i.e. $G_i^* = 5$). Using the previously listed formulation for feature *goal completion*, we can calculate that $q_{goc} = 0.24$.

When computable, $q_{goc}(u_i)$ takes value between 0 and 1, inclusive of the limits. The value of zero for this feature means a user is already at the target level or beyond for the activity depicted in the goal. The value of one implies that a user is 365 or more days away from achieving their goal. For any given plot, the value for this feature may change at the next interaction if user data that is displayed in the plot gets updated.

3.5.7. Data freshness (q_{daf})

A relevant plot may keep getting selected over and over again. Moreover, it may be the case that since the last time a plot was displayed to the user, not much has been changed in the plotted data and thus there might be repetition in the selection. This would particularly be true for plots having a long frequency type, such as plots tracking an activity with monthly or quarterly frequency, where there would be little change in plotted data in a day or two. To limit the repetition of the same plots over and over again, we introduce a feature that quantitatively estimates the new data that is available in the plot compared to its previously displayed version. This feature can help in prioritizing plots that have significantly more new data. To do so, we rely on the following information:

- $t_{current}$ is the current time.
- t_{start} is the start time (i.e. earliest x value) of plotted data. This value can be taken from plotted user data (P) data source.
- $t_{last\ selection}(u_i)$ is the time when plot u_i was last selected. This value can be gathered from selection log (S).

Using the above values, we formulate the feature *data freshness* as follows:

$$q_{daf}(u_i) = \frac{t_{current} - t_{last\ selection}}{t_{current} - t_{start}}$$

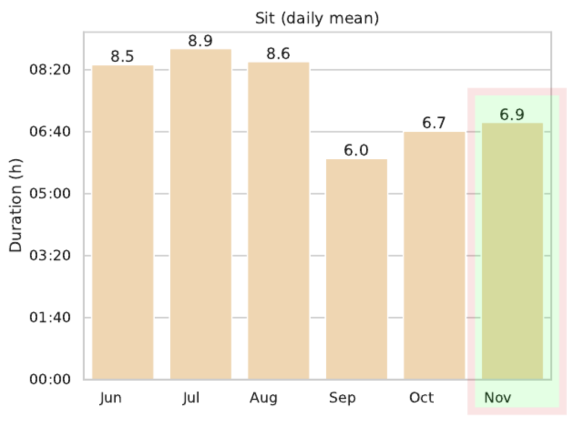


Figure 3.8: Example plot to illustrate the *data freshness* feature (q_{tre}): last displayed plot to the user having the timestamp of `2019-11-16 12:34:12`

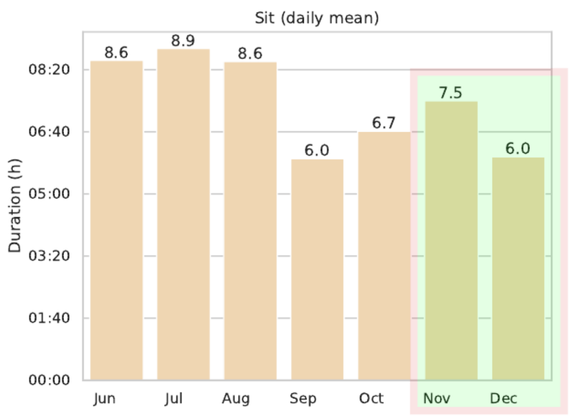


Figure 3.9: Example plot to illustrate *data freshness* feature (q_{tre}): currently generated plot having the timestamp of `2019-12-09 10:34:12`

To illustrate this feature, we suppose that the plot for monthly mean hours spent sitting by a user (as shown in Figure 3.8) was last displayed (i.e. $t_{last\ selection}$) at `2019-11-16 12:34:12`. Now let us assume that the current time (i.e. $t_{current}$) is `2019-12-09 10:34:12`, and the plot with the updated data looks like in Figure 3.9. In Figure 3.9, we can see that the smallest x value (i.e. t_{start}) is `2019-06-01 00:00:00`. With these three timestamps, and using the previously listed formulation for feature *goal completion*, we can calculate that $q_{daf} = 0.18$.

When computable, $q_{daf}(u_i)$ takes value between 0 and 1, inclusive of the limits. For any given plot, the value for this feature may change at the next interaction if user data that is displayed in the plot gets updated and also when a plot gets selected to be displayed.

3.6. Defining the weights

As discussed in Section 3.4, the four objective functions are high-level in nature and are to be understood as linear combination of a subset of low-level quantifiable plot features that are relevant to an objective function. As a parameter of this linear combination, we introduced $w_{i,j}$ as the weight of a plot feature q_j in the value of the objective function f_k. In this section we discuss the way of selecting the values for these weights, which is mainly defined by an analysis of the relevance of the individual features for a given objective function.

For the **relevance** objective, the only pertinent plot feature is *activity*. A plot is deemed more relevant if most of the activities depicted in it are part of user goals. Therefore, we model the relevance of the plot directly by the activity feature, set its weight to 1 and the weights of all other features to 0. In others words, we define $f_{rel}(u_i)$ as following:

$$f_{rel}(u_i) \; = \; q_{act}(u_i) \tag{3.12}$$

A similar reasoning can be applied for the **freshness** objective, where the relevant plot feature is *data freshness*. A plot is deemed more fresh if most of the data depicted in it has not been previously presented to the user. Therefore, we define $f_{fre}(u_i)$ as following:

$$f_{fre}(u_i) \; = \; q_{daf}(u_i) \tag{3.13}$$

For the **likability** objective, the relevant plot features are all those that capture user ratings. According to our feature definitions, plot features *'plot type'*, *'analysis type'*, and *'frequency type'* make use of the interaction log to quantify a user's preference for the plot's plot type, analysis type and frequency type, respectively. We take all these three features to contribute equally to the likability objective with higher value on any of the three feature implying greater preference for the plot. Therefore, we define $f_{lik}(u_i)$ as following:

$$f_{lik}(u_i) \; = \; \frac{1}{3}(q_{plt}(u_i) \; + \; q_{ant}(u_i) \; + \; q_{frt}(u_i)) \tag{3.14}$$

For the **noteworthiness** objective, the relevant plot feature are all those that make use of patterns in plotted data. According to our feature definitions, plot features *'trend'* and *'goal completion'* make use of plotted user data to quantify certain patterns in plot. We take these two features to contribute equally to the noteworthiness objective such that higher value on any of the two feature implies that the plot would be interesting for the user. Therefore, we define $f_{not}(u_i)$ as following:

$$f_{not}(u_i) \;=\; \frac{1}{2}(q_{tre}(u_i) \;+\; q_{goc}(u_i)) \tag{3.15}$$

Table 3.2: Value of weights

Feature		**Relevance** f_{rel}	**Freshness** f_{fre}	**Likeability** f_{lik}	**Noteworthiness** f_{not}
Activity	$q_{act}(u_i)$	$w_{rel,act} = 1$	$w_{fre,act} = 0$	$w_{lik,act} = 0$	$w_{not,act} = 0$
Plot type	$q_{plt}(u_i)$	$w_{rel,plt} = 0$	$w_{fre,plt} = 0$	$w_{lik,plt} = \frac{1}{3}$	$w_{not,plt} = 0$
Analysis type	$q_{ant}(u_i)$	$w_{rel,ant} = 0$	$w_{fre,ant} = 0$	$w_{lik,ant} = \frac{1}{3}$	$w_{not,ant} = 0$
Frequency type	$q_{frt}(u_i)$	$w_{rel,frt} = 0$	$w_{fre,frt} = 0$	$w_{lik,frt} = \frac{1}{3}$	$w_{not,frt} = 0$
Trend	$q_{tre}(u_i)$	$w_{rel,tre} = 0$	$w_{fre,tre} = 0$	$w_{lik,tre} = 0$	$w_{not,tre} = \frac{1}{2}$
Goal completion	$q_{goc}(u_i)$	$w_{rel,goc} = 0$	$w_{fre,goc} = 0$	$w_{lik,goc} = 0$	$w_{not,goc} = \frac{1}{2}$
Data freshness	$q_{daf}(u_i)$	$w_{rel,daf} = 0$	$w_{fre,daf} = 1$	$w_{lik,daf} = 0$	$w_{not,daf} = 0$
		$\sum_{j\in Q} w_{rel,j} = 1$	$\sum_{j\in Q} w_{fre,j} = 1$	$\sum_{j\in Q} w_{lik,j} = 1$	$\sum_{j\in Q} w_{not,j} = 1$

Table 3.2 summarizes the values for all weights. Firstly, it may be noted with the values proposed by us for different weights that a low-level plot feature contributes to exactly one high-level objective. However, this is not strictly necessary in such formulations. If a plot feature is understood to have a bearing on multiple objective functions, an alternative formulation could assign positive weights for that plot feature with multiple objective functions. Secondly, we have assumed equal weights for all the contributing plot features in this project. However, advanced formulations could make use of variable and/or dynamic weights. For example, the weights could be treated as *parameters* in a reinforcement learning (RL) based approach that are dynamically learned. Since RL-based approach would require significantly more interaction log data (i.e. I) than we had for this project, the values of weights $w_{i,j}$ are kept static for this project. However, if significantly large interaction log is available, perhaps through prolonged usage of the optimization system by a user and/or in a multi-user setting, RL-based approach for updating values of weights $w_{i,j}$ could be explored. We leave this and other potential alternative approaches for the future work.

3.7. ε-values as hyper-parameters

As discussed in Section 3.4, we introduced ε-values (i.e., ε_{fre}, ε_{lik}, and ε_{not}) to modify our multi-objective linear programming problem into a standard linear programming formulation with one objective function. In effect, ε-values limit the selection (i.e. x_i values) such that a minimum threshold must be satisfied for each of three secondary objectives (i.e. $\sum_{i=1}^{n} x_i \times f_{fre}(u_i)$ for freshness, $\sum_{i=1}^{n} x_i \times f_{lik}(u_i)$ for likability, and $\sum_{i=1}^{n} x_i \times f_{not}(u_i)$ for noteworthiness) for any valid solution to our integer programming problem formulation. As evident from the problem formulation, the higher an ε-value, the stricter are the requirements for the corresponding objective. Since the ε-values directly govern the strictness imposed by a secondary objective, we consider them hyperparameters of our optimization model.

The values of ε_i (i.e. ε_{fre}, ε_{lik}, and ε_{not}) are deployed to put a minimum on overall score of an objective function (i.e. $\sum_{i=1}^{n} x_i \times f_i(u_i)$. It is therefore pertinent to set the ε_i value in relation to the magnitude of the summation score. As discussed in previous sections, the overall objective score depends on the objective score of candidate plots, which in turn depends on the quantified plot features. Moreover, quantified plot feature values themselves depend on certain underlying raw values. Therefore, it is difficult to set meaningful ε-values in an absolute manner. We therefore, make use of percentile scores from ranked lists of $f_i(u_i)$ values to set ε_i.

Let us define a sequence F_i of objective scores f_i for all plots as follows:

$$F_i =< f_i(u_1), f_i(u_2), \dots, f_i(u_n) > \tag{3.16}$$

Now, let us reindex the sequence F_i in the monotone increasing order as R_i such that:

$$R_i =< r_i(1), r_i(2), \dots, r_i(n) > \tag{3.17}$$

Now the maximum value that ε_i should take is $\sum_{j=1}^{k} r_i(j)$. In other words, the overall objective score while selecting k plots can take a maximum value of the sum of top-k objective scores. We define the ε_i value on a scale from 0 to the previously listed maximum value. To do that, we define a function $l_i(c)$ such that:

$$l_i(c) = c \times \sum_{j=1}^{k} r_i(j) \tag{3.18}$$

Using the above relation, having $\varepsilon_i = l_i(1)$ means that we want to maximize the overall score for objective f_i. On the other hand, having $\varepsilon_i = l_i(0)$ translates to nullifying any requirements for the objective f_i. We use λ_i as a value between 0 and 1, inclusive of the limits, to set the hyperparameters ε_i as following:

$$\varepsilon_{fre} = l_{fre}(\lambda_{fre}) \tag{3.19}$$
$$\varepsilon_{lik} = l_{lik}(\lambda_{lik}) \tag{3.20}$$
$$\varepsilon_{not} = l_{not}(\lambda_{not}) \tag{3.21}$$
$$\text{s. t. } \lambda_i \in [0,1] \; \forall \, i \in \{fre, \, lik, \, not\} \tag{3.22}$$

We find $\lambda_i = 0.25 \, \forall \, i \in \{fre, \, lik, \, not\}$ as a reasonable choice to set the hyperparameters. In the following chapter we discuss results obtained using aforementioned values and variations on them.

3.8. Implementation design choices

Since ILP is NP-hard, heuristic methods are often used to solve it and various implementations of these algorithms are readily available. SInce the focus of this thesis project is on modeling the problem, we do not dive deeper into an analysis of implementation posisbilities for solving ILP problems. Instead, for this project, the Python-MIP package is used for implementing ILP programs which in turn uses Cbc (Coin-or branch and cut) as the solver.

Solving the ILP problem provide us with values x_i as 0 or 1, where $x_i = 0$ means that the plot u_i is not selected, while $x_i = 1$ says that plot u_i is selected for the current display. As would be evident from Equation 3.9, out of the total n plots, k plots will be selected(i.e. plots with $x_i = 1$). It is possible to rank these k plots using several metrics such as score on primary objective, combined score across all objectives, etc. It is known that in a vertical layout such as ours, higher ranked UI elements get more visibility and user's attention. Therefore, displaying the k plots in a ranked order can result in similar type of plots (i.e. those would rank high according to the implemented criterion) getting disproportionate visibility. Therefore, in our implementation, we display the selected k selected plots in a random order.

3.9. Overall schema of optimization engine

We introduced the raw data sources for the optimization engine in Section 2.3. The raw values in these data sources are used to formulate quantified plot features (q_k) as discussed in Section 3.5. Thereafter, the plot features are variously combined to get values for higher level objective functions (f_j). Finally, ε_l values are deployed to put constraints on overall scores of secondary objective functions (f_{fre}, f_{lik}, and f_{not}) while maximizing for overall score for primary objective (i.e. f_{rel}) to identify the plots (x_i) that delivers the most utility to the user. The selected plots are then randomly sorted for display to the user. The process from taking raw values from input data sources to generating the user-interface with selected plots can be summarized as follows:

$$(M,\ G,\ P,\ I,\ S)\ \rightarrow q_k \xrightarrow{w_{(j,k)}} f_j \xrightarrow{\varepsilon_l} x_i \rightarrow \text{User Interface generated with selected plots}$$

Figure 3.10 depicts the overall schema of optimization system.

Table 3.4 summarizes the plot features defined in previous section and the nature of data (i.e. user goals, interaction log, and user data) utilized to compute it.

Table 3.3: Summary of plot features and nature of data sources utilized in computing them

Feature		**User Goals** G	**Selection Log** S	**Interaction Log** I	**Plotted User Data** P
Activity	$q_{act}(u_i)$	Yes			
Plot type	$q_{plt}(u_i)$			Yes	
Analysis type	$q_{ant}(u_i)$			Yes	
Frequency type	$q_{frt}(u_i)$			Yes	
Trend	$q_{tre}(u_i)$				Yes
Goal completion	$q_{goc}(u_i)$	Yes			Yes
Data freshness	$q_{daf}(u_i)$		Yes		Yes

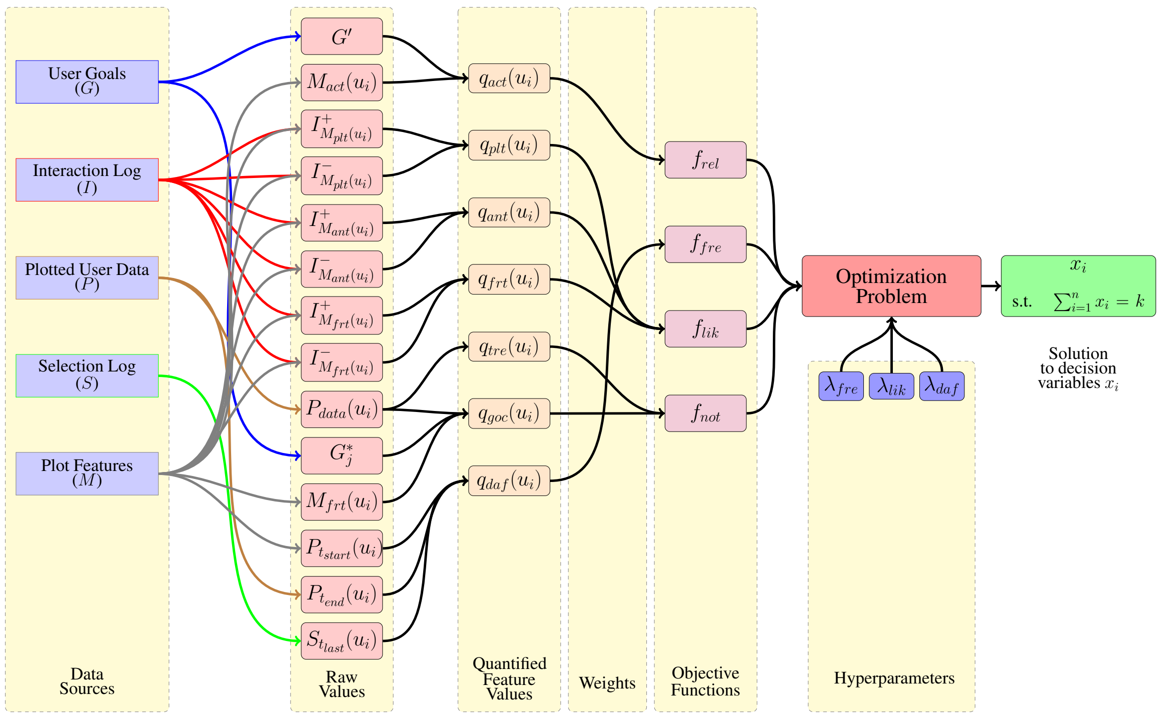


Figure 3.10: Overall schema of the optimization engine

Table 3.4: Summary of plot features and their contributions to objective functions

Feature		Relevance f_{rel}	Freshness f_{fre}	Likeability f_{lik}	Noteworthiness f_{not}
Activity	$q_{act}(u_i)$	Yes			
Plot type	$q_{plt}(u_i)$			Yes	
Analysis type	$q_{ant}(u_i)$			Yes	
Frequency type	$q_{frt}(u_i)$			Yes	
Trend	$q_{tre}(u_i)$				Yes
Goal completion	$q_{goc}(u_i)$				Yes
Data freshness	$q_{daf}(u_i)$		Yes		

3.10. Simulation model and system design for evaluating the optimization engine

We use a full-fledged implementation of quantified-self system spanning various tools implemented in various programming languages sketched in the following chapter to generate the UI element bank and other data sources required by the optimization engine. The optimization engine is implemented in Python as per formulations discussed in the previous chapter. Additionally, we use a simple JavaScript based web-app to function as user interface for the quantified-self system and communicate with the system. Finally, we also developed a test-suite in Python to simulate user-interactions with the User Interface and analyse performance of the optimization. Figure 3.11 depicts the schema of simulation model that we use for evaluating the optimization engine.

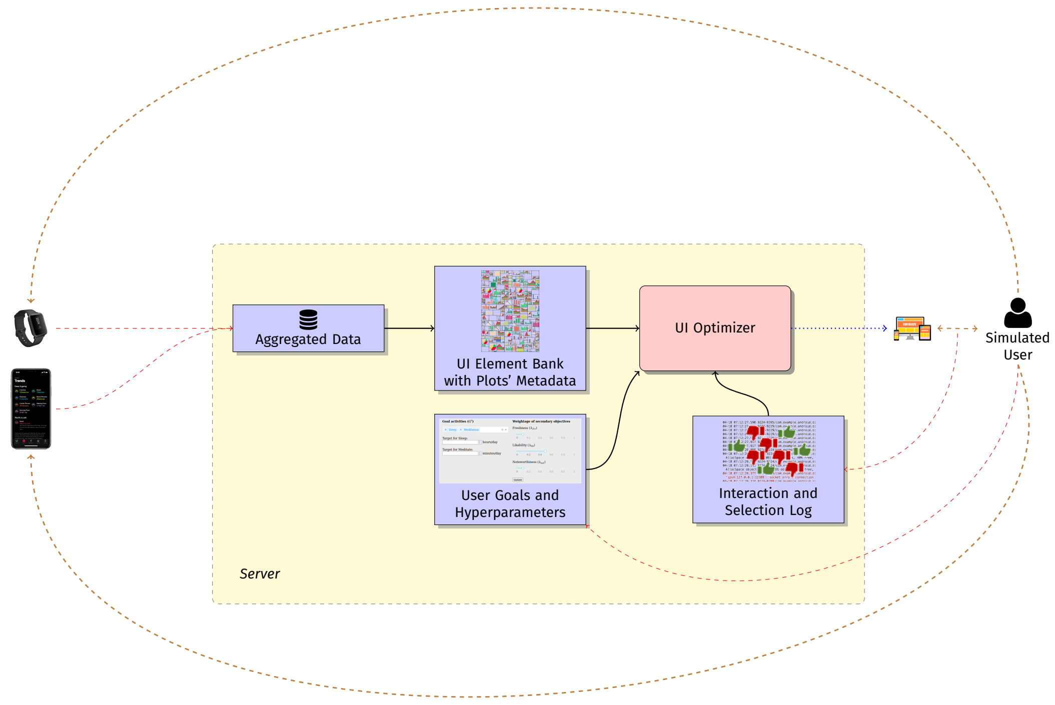

Aggregated Data
UI Element Bank with Plots' Metadata
UI Optimizer
Simulated User
User Goals and Hyperparameters
Interaction and Selection Log
Server

Figure 3.11: Simulation model to evaluate the optimization engine

The activity data used by the quantified-self system was collected over several months using an activity tracker and self-logging apps on phone by the author. The raw values are altered for anonymization purposes. Nevertheless, the masked values retain the real-world character to serve for the evaluation purposes.

The UI element bank is populated at any given time instance in the available time range of the data stream. The selection tool implements our optimization algorithm in python, and provides the IDs of the selected plots which are then displayed on a web-app. The user has the ability to interact with the web-app in terms of clicking the "like" and "dislike" button. The statistics on features of plots that are selected are logged for evaluation if the system behaves in line with our hypothesis or not. Another tool was developed to automate user interactions over the specified period of time so that many runs of the optimization engine can be made.

In the following chapter we evaluate the optimization system discussed in this chapter using the above-mentioned simulation model to analyse the performance of plot selection methodology for several usage scenarios of a quantified-self user.

4 Evaluation

As already discussed in Chapter 2, the function of the optimization engine in the larger quantified-self system is to take into consideration plot features, user goals, selection log, previous user interactions, and the patterns in plotted user data to select a small number of plots that should be presented to a user to maximize the multi-criteria (i.e., relevance, freshness, likability, and noteworthiness) utility. In Chapter 3, we discussed the design of the optimization engine and the underlying mathematical formulations. Since we aim to optimize for relevance, freshness, likability, and noteworthiness in plot selection, we use the same criteria for evaluating the performance of our optimization engine.

In this chapter, we first provide details of user data and the simulation model used to evaluate the optimization engine. Secondly, we discuss the criteria for evaluating the optimization methodology described in the previous chapter. Thirdly, the results from the simulation model to evaluate the efficacy of the optimization engine on identified criteria are provided. Finally, we reflect on the obtained results and the effectiveness of our methodology in optimizing UI for a quantified-self application.

4.1. Evaluation Setting

The activity data used by the quantified-self system was collected over several months using the author's activity tracker and self-logging apps. The details of the simulation conditions are provided in table 4.1.

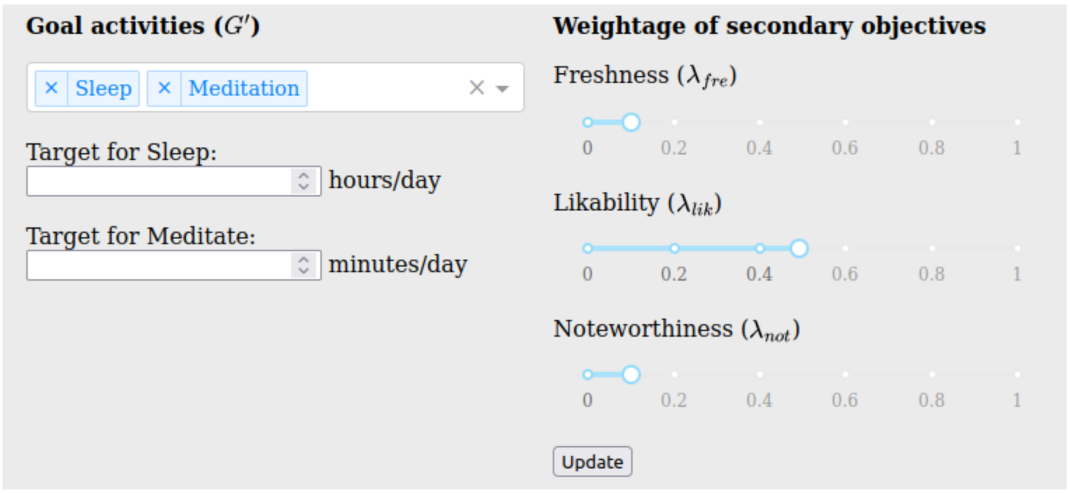
The user data consists of ten different activities for nine months. The optimization engine is deployed at the beginning of the fourth month so that sufficient historical data is available for visualization. As a result, the simulation of plot selection using the proposed optimization engine is carried out for six months (i.e., from the beginning of the fourth month to the end of the ninth month).

The simulation model outlined in the previous chapter is used in conjugation with different parameter values to create several scenarios of system usage by the user to mimic different real-world conditions. For instance, if we want to simulate a user who starts using the optimization engine out-of-the-box without specifying any goal activities or inclination for objective functions (i.e., "relevance", "freshness", "likability", and "noteworthiness"), the goal activities are left as blank in the system, and the hyperparameters take default values. Alternatively, if a user has a greater preference for the "likability" objective and is interested in 'Sleep' and 'Meditate' activities, the parameters can be specified accordingly. Fig-

Table 4.1: Data-streams and parameters employed by simulation model for evaluating optimization engine

Measure	**Value**
Data stream start date	June 1, 2019 00:00:00
System deployment date (T_{start})	September 1, 2019 00:00:00
Data stream end date (T_{end})	February 29, 2020 23:59:59
Simulation duration (T)	Six months
All activities	10: {'Steps', 'Sit', 'Stand', 'Sleep', 'Walk', 'Run', 'Other', 'Phone', 'Meditation', 'Sports'}
Goal activities *(when marked)* (G')	3: {'Sleep', 'Walk', 'Meditation'}
Number of total plots (N)	170
Number of plots to be selected (k)	7
Frequency for updating UI element bank	once every three hours
Frequency for renewing plot selection	at every update of UI element bank
Interaction frequency	$unif\{0,3\}$ interactions per day
Rating frequency	$unif\{0,3\}$ plots per interaction
Default values for hyperparameters	$\lambda_{fre} = \lambda_{lik} = \lambda_{not} = 0.25$

ure 4.1 illustrates settings that may be specified to suit the aforementioned scenario by a user of the proposed quantified-self system.


Goal activities (G')

× Sleep × Meditation

Target for Sleep: hours/day

Target for Meditate: minutes/day

Weightage of secondary objectives

Freshness (λ_{fre})

Likability (λ_{lik})

Noteworthiness (λ_{not})

Figure 4.1: User interface in our simulation model for updating the **settings** (i.e. parameter values for 'Goals activities' (G') and λ values) by the quantified-self system user in our simulation model. The setting in the above figure models one **scenario** (i.e. a hypothetical real-world usage situation) where a user has a greater preference for the "likability" objective and is interested in tracking 'Sleep' and 'Meditation' activities.

To sum up, **scenarios** are hypothetical real-world usage situations that are implemented for evaluation by using suitable **settings** (i.e. value for goal activities and hyperparameters λ_{fre}, λ_{lik}, and λ_{not}). In this chapter, we will create several scenarios for each of the evaluation metrics and the corresponding settings used to model that scenario will also be provided.

4.2. Evaluation criterion, evaluation metrics, and hypotheses

We evaluate our optimization engine in terms of efficacy of the proposed system to meet the objectives. We thus have an evaluation criterion corresponding to each of the four objectives discussed and outlined in Chapter 2 and Chapter 3. For each of the four **evaluation criteria**, we propose qualitative scenarios (and the corresponding settings of parameter values) using which we can evaluate the efficacy of the optimization engine by comparing the output (i.e. selection log) of optimization engine. Thereafter, we formulate an **evaluation metric** to make quantitative comparison of the various scenarios/settings suitable for the criterion. Subsequently, we make **hypotheses** for the evaluation metric value for different scenarios if the system were to satisfy the evaluation criterion (i.e. prove effective in achieving the corresponding objective).

4.2.1. Relevance

We evaluate the realization of the relevance objective by measuring the sensitivity of the optimization engine in adapting to changing user goals. If we consider theactivities the quantified-self user wants to track especially, the selection algorithm should prioritize those goal activities. To evaluate it, we compare the share of "relevant" plots (i.e., those related to goal activities, like "Sleep," "Walk," and "Meditate") with and without specifying the same to the system. Equation 4.1 defines the evaluation metric for relevance (i.e., P_{rel}) as the share of relevant goal activities G' in all the activities A_i depicted in the given plot u_i.:

$$P_{rel}(u_i) = \frac{|A_i \cap G'|}{|A_i|} \tag{4.1}$$

Equation 4.2 defines for *i*-th selection/interaction s_i, the "relevance" metric S_{rel} as the share of "relevant" plots (out of total k plots) selected during that interaction:

$$S_{rel}(s_i) = \frac{\sum_{j=1}^{k} P_{rel}(u_j)}{k} \times 100 \tag{4.2}$$

Equation 4.3 defines the evaluation metric for relevance (i.e., D_{rel}) for a given duration as the average of "relevant" plot share (i.e. S_{rel}) of all interactions occurring between t_{intial} and t_{final}.

$$D_{rel}(t_{initial}, t_{final}) = \frac{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} S_{rel}(s_i)}{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} 1} \tag{4.3}$$

Equation 4.4 defines the main evaluation metric for measuring efficacy of the system in ensuring "relevance" (i.e., E_{rel}) as average of "relevant" plot share (i.e. S_{rel}) in all interactions occurring during the entire simulation duration (i.e. between T_{start} and T_{end}).

$$E_{rel} = D_{rel}(T_{start}, T_{end}) \tag{4.4}$$

We make the following hypotheses for the expected value of the evaluation metric E_{rel} in various scenarios to validate the usefulness of our proposed selection method in ensuring the "relevance" in plot selection:

- **Hypothesis 1a (H_{rel}):** For default λ values (i.e., λ_{fre}, λ_{lik}, and λ_{not}), we should be seeing a higher proportion of "relevant" plots (i.e., E_{rel}) in a scenario where goals-activities are marked in the optimization engine compared to a scenario where no goal-activities are specified.

- **Hypothesis 1b (H'_{rel}):** If we decrease the importance of secondary objectives (i.e., freshness, likability, and noteworthiness), the share of "relevant" plots should increase. In other words, E_{rel} should be negatively co-related with λ values.

4.2.2. Freshness

With the freshness objective, we aimed to show users plots containing data they have not seen in previous interactions. Therefore, we evaluate the realization of the freshness objective by measuring what percentage of k plots can be deemed ”fresh.” To do so, we define Equation 4.5 that considers a plot to be ”fresh” if the raw value of data-freshness (i.e., $M_{daf}(u_i)$ for that plot is greater than 50%:

$$P_{fre}(u_i) = \left\{ \begin{array}{ll} 0, & \text{for } M_{daf}(u_i) < 0.5 \\ 1, & \text{for } M_{daf}(u_i) \geq 0.5 \end{array} \right\} \tag{4.5}$$

Equation 4.6, Equation 4.7, and Equation 4.8 defines the share of “fresh” plots in the interaction s_i, all interactions occurring between t_{intial} and t_{final}, and all interactions occurring during the whole of simulation duration (i.e. between T_{start} and T_{end}), respectively:

$$S_{fre}(s_i) = \frac{\sum_{j=1}^{k} P_{fre}(u_j)}{k} \times 100 \tag{4.6}$$

$$D_{fre}(t_{initial}, t_{final}) = \frac{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} S_{fre}(s_i)}{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} 1} \tag{4.7}$$

$$E_{fre} = D_{fre}(T_{start}, T_{end}) \tag{4.8}$$

We make the following hypotheses for the expected value of evaluation metric E_{fre} in various scenarios to validate the usefulness of our proposed selection method in ensuring the “freshness” in plot selection:

- **Hypothesis 2a (H_{fre}):** For default λ values, we should be seeing a significant share of ”fresh” plots (i.e., E_{fre}) in all scenarios, irrespective of whether goals have been marked or not.

- **Hypothesis 2b (H'_{fre}):** The share of ”fresh” plots (i.e. E_{fre}) should be positively co-related with the value of λ_{fre}. As a result, a scenario having a higher λ_{fre} value should result in a greater share of “fresh” plots than a scenario that uses a lower λ_{fre} value, with the remaining λ values staying the same.

4.2.3. Likability

With the likability objective, we aimed at prioritizing plot types that the user rated positively in previous interactions. We thus evaluate the realization of the likability objective by measuring what percentage of k plots can be said to have likable properties. We do this by means of Equation 4.9, which defines that a plot is considered to be ”likable” if the percentile score is greater than 75 for any of the following three likability-related quantified plot vectors: i) likability of *plot type* (i.e., $q_{plt}(u_i)$); ii) likability of *frequency type* (i.e., $q_{frt}(u_i)$); iii) likability of *analysis type* (i.e., $q_{ant}(u_i)$). :

$$P_{fre}(u_i) = \left\{ \begin{array}{ll} 1, & \text{if } q_{plt}(u_i) \geq 75 \quad \text{or } q_{frt}(u_i) \geq 75 \quad \text{or } q_{ant}(u_i) \geq 75 \\ 0, & \text{otherwise} \end{array} \right\} \tag{4.9}$$

Equation 4.10, Equation 4.11, and Equation 4.12 defines the share of “likable” plots in the interaction s_i, all interactions occurring between t_{intial} and t_{final}, and all interactions occurring during the whole of simulation duration (i.e. between T_{start} and T_{end}), respectively.

$$S_{lik}(s_i) = \frac{\sum_{j=1}^{k} P_{lik}(u_j)}{k} \times 100 \tag{4.10}$$

$$D_{lik}(t_{initial}, t_{final}) = \frac{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} S_{lik}(s_i)}{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} 1} \tag{4.11}$$

$$E_{lik} = D_{lik}(T_{start}, T_{end}) \tag{4.12}$$

We make the following hypotheses for the expected value of evaluation metric E_{lik} in various scenarios to validate the usefulness of our proposed selection method in ensuring the "likability" in plot selection:

- **Hypothesis 3a (H_{lik}):** For default λ values and after having a sufficiently large interaction log of user ratings for the plots, we should be seeing a significant share of "likable" plots (i.e., E_{lik}) in all scenarios, irrespective of whether goals have been marked or not.
- **Hypothesis 3b (H_{lik}):** The share of "likable" plots (i.e. E_{lik}) should be positively co-related with the value of λ_{lik}. As a result, a scenario having a higher λ_{lik} value should result in a greater share of "likable" plots than a scenario which uses a lower λ_{lik} value, with the remaining λ values staying the same.

4.2.4. Noteworthiness

With the noteworthiness objective, we aimed to prioritize plot types with significant trends in the plotted user data. We evaluate the realization of this objective by simulating a strong trend in the time series data of a non-goal activity and then measuring the selection rate of plots for that activity.

For the evaluation of "noteworthiness", we simulate the logged data such that data values for activity "Phone" are relatively constant for the first 30% (i.e., from (T_{start} to $T_{start} + \frac{3T}{10}$)) of system usage duration. The data values for this activity thereafter show a strong upward trend for the next 20% of the duration followed by strong downward trend for the next 20% duration. The data values are again stable for the final 30% (i.e., from $T_{end} - \frac{3T}{10}$ to T_{end}) of the duration. Figure 4.2 depicts the simulated values for "Phone" usage activity in the user's datastreams employed for evaluation of efficacy in achieving "noteworthiness" objective.

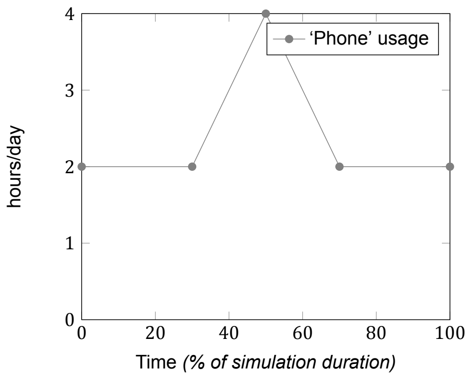


Figure 4.2: Simulated values for 'Phone' activity in user's datastream

With the above simulation, the metric of interest in evaluating "noteworthiness" is the mean share of plots related to activity 'Phone' in a given interval (i.e., $E_{not}(t_{initial}, t_{final})$). We define the intervals (i.e. $(t_{initial}, t_{final})$) as 5% intervals of the simulation duration. Equation 4.13 defines the evaluation metric for "noteworthiness" (i.e., $P_{not}(u_i)$) as the share of 'Phone' activity in all the activities A_i depicted in the plot u_i.

$$P_{not}(u_i) = \frac{|A_i \cap \{\text{'Phone'}\}|}{|A_i|} \tag{4.13}$$

Equation 4.14 and Equation 4.15 define the share of "fresh" plots in the interaction s_i and all interactions occurring between t_{intial} and t_{final}, respectively.

$$S_{not}(s_i) = \frac{\sum_{j=1}^{k} P_{not}(u_j)}{k} \times 100 \tag{4.14}$$

$$D_{not}(t_{initial}, t_{final}) = \frac{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} S_{not}(s_i)}{\sum_{\forall s_i \in \{t_{initial}, t_{final}\}} 1} \tag{4.15}$$

Equation 4.16 defines the main evaluation metric for measuring efficacy of the system in ensuring "noteworthiness" (i.e., E_{not}) as a difference between the share of "noteworthy" plots selected between the central part (i.e. during the time $T_{start} + \frac{3T}{10}$ and $T_{end} - \frac{3T}{10}$) when there is strong trend in underlying data and the remaining duration when the data values are relatively constant.

$$E_{not} = D_{not}(T_{start} + \frac{3T}{10}, T_{end} - \frac{3T}{10}) - \frac{D_{not}(T_{start}, T_{start} + \frac{3T}{10}) + D_{not}(T_{end} - \frac{3T}{10}, T_{end})}{2} \tag{4.16}$$

We make the following hypotheses for the expected value of evaluation metric E_{not} in various scenarios to validate the usefulness of our proposed selection method in ensuring the "notworthiness" in plot selection:

- **Hypothesis 4a (H_{not}):** For default λ values, we should be seeing a higher proportion of plots related to "Phone usage" in the central part of system usage duration (i.e., where the data values have strong upward or downward trends).
- **Hypothesis 4b (H'_{not}):** The rise in the selection rate of plots related to "Phone" activities during the central part vis-à-vis initial and final parts (i.e. E_{not}) should be positively co-related with the value of λ_{not}.

4.3. Results

In this section, we present and discuss the evaluation metrics' results and validity of our hypotheses. We do this using different scenarios that we define per objective in the corresponding tables.

4.3.1. Relevance

Table 4.2 provides the mean values for three scenarios and the employed parameter settings for the whole simulation duration. Figure 4.3 depicts the results from evaluating the optimization engine for the realization of *relevance* objective for every 5% interval for each of these three scenarios.

As provided in Table 4.2, the mean share of relevant plots (i.e., those depicting "Sleep", "Walk", and "Meditate" activities) is 39.45% for the simulation duration in Scenario 1a where the goal activities are not marked. In Scenario 1b, we mark the goal activities and see the mean share of relevant plots increase to 61.82% for the simulation duration. Therefore, we do see the increase in the share of

	Scenario	Goal Activities	λ_{fre}	λ_{lik}	λ_{not}	E_{rel}
1a	No goals are set	ϕ	0.25	0.25	0.25	**39.45%**
1b	Goals are defined and system prioritizes all objectives	G'	0.25	0.25	0.25	**61.82%**
1c	Goals are defined and system mainly prioritizes "relevance"	G'	0.10	0.10	0.10	**82.14%**

Table 4.2: Results for evaluation metric for relevance averaged across all interactions for the whole of simulation duration (i.e., E_{rel}) for different scenarios.

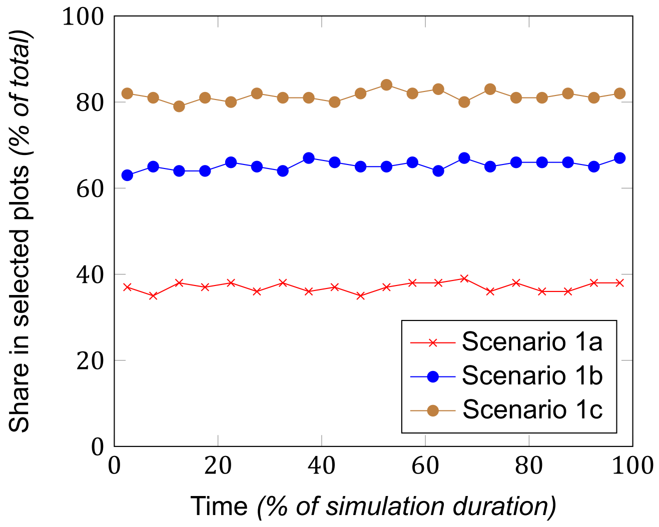


Figure 4.3: Aggregated mean share of "relevant" plots (i.e. plots depicting G' activities) in the selected plots for every 5% interval for —
Scenario 1a: No goals are set with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$;
Scenario 1b: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$
Scenario 1c: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.1$, $\lambda_{lik} = 0.1$, and $\lambda_{not} = 0.1$

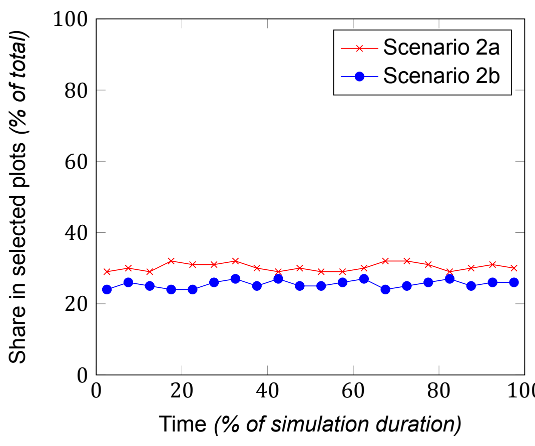
relevant plots from Scenario 1a to 1b and the obtained results corroborate our Hypothesis 1a. Similarly, we see the share of relevant plots increase as we decrease the $lambda$ values in Scenario 1c compared to their magnitude in Scenario 1b. This proves our Hypothesis 1b that decreasing the importance of secondary objectives should increase the share of relevant plots. Moreover, the same pattern in the share of relevant plots in the three scenarios can be observed for each of the 5% intervals of the simulation duration in Figure 4.3. Hence, the results obtained from the simulation of the optimization engine corroborate our hypotheses 1a and 1b.

4.3.2. Freshness

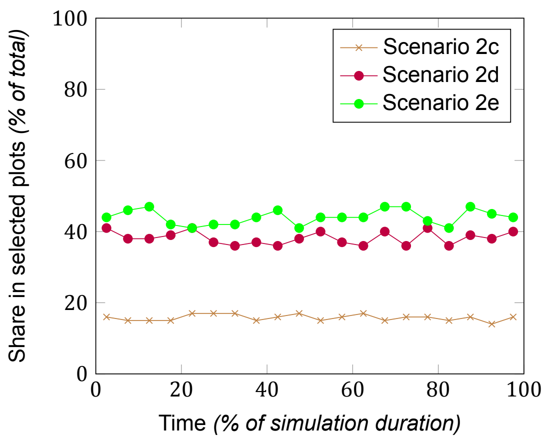
Table 4.3 provides the mean values for five scenarios for the whole simulation duration. Figure 4.4 depicts the results from evaluating the optimization engine for the realization of the *freshness* objective for every 5% interval for each of the five scenarios.

	Scenario	Goal Activities	λ_{fre}	λ_{lik}	λ_{not}	E_{fre}
2a	No goals are set	ϕ	0.25	0.25	0.25	**30.62%**
2b	Goals are defined and system prioritizes all objectives	G'	0.25	0.25	0.25	**25.79%**
2c	Goals are defined and system mainly prioritizes “relevance”	G'	0.10	0.10	0.10	**15.43%**
2d	Goals are defined and system mainly prioritizes “relevance” and “freshness”	G'	0.30	0.10	0.10	**38.56%**
2e	Goals are defined and system mainly prioritizes “freshness”	G'	0.50	0.10	0.10	**43.68%**

Table 4.3: Mean share of “fresh” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E_{fre}).



(a) *Scenario 2a:* No goals are set with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$;
Scenario 2b: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$



(b) *Scenario 2c:* G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.10$;
Scenario 2d: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.30$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.10$; and
Scenario 2e: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.50$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.10$

Figure 4.4: Aggregated mean share of “fresh” plots in the selected plots for every 5% interval

As provided in Table 4.3, the mean share of “fresh” plots is 30.62% for the whole simulation duration in Scenario 2a, where the goal activities are not marked. In Scenario 2b, we mark the goal activities and see the mean share of “fresh” plots comes down to 25.79%. The same pattern in the share of “fresh” plots in the two scenarios can be observed for each of the 5% intervals of the simulation duration in Figure 4.4a. Therefore, we see a significant number of “fresh” plots being selected when no goals are specified and when goals activities are marked. This corroborates our Hypothesis 2a.

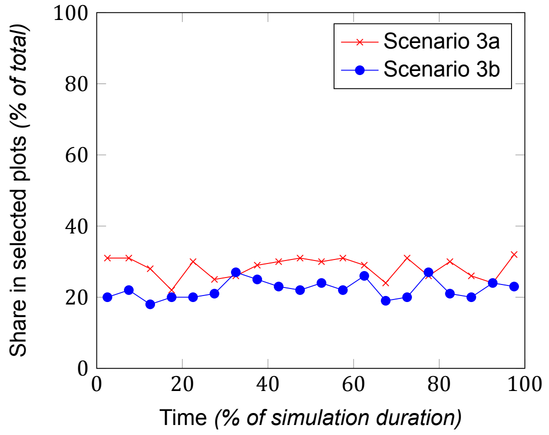
The share of “fresh” plots increases as we go on, increasing the λ_{fre} value from 0.1 in Scenario 2c to 0.3 in Scenario 2d and further to 0.5 in Scenario 2e. The same pattern in the share of “fresh” plots in the three scenarios can be observed for each of the 5% intervals of the simulation duration in Figure 4.4b. Preceding results validate our Hypothesis 2b (H_{fre}).

4.3.3. Likability

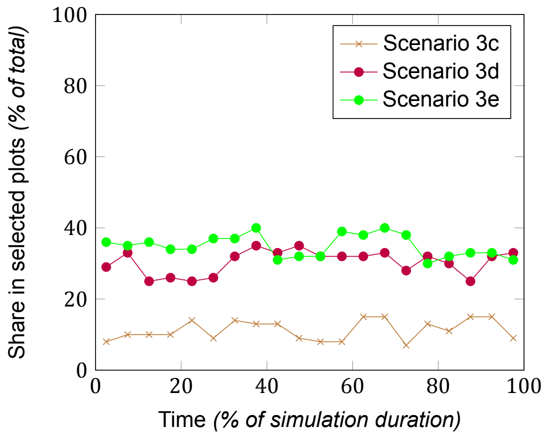
To evaluate the realization of *likability* objective, Table 4.4 provides the mean values of the evaluation metric E_{lik} for five scenarios for the total simulation duration. Figure 4.5 depicts the results for the metric $E_{lik}(t_{initial}, t_{final})$ from evaluating the optimization engine for every 5% interval for each of the five scenarios.

	Scenario	Goals	λ_{fre}	λ_{lik}	λ_{not}	E_{lik}
3a	No goals are set	ϕ	0.25	0.25	0.25	**26.93%**
3b	Goals are defined and system prioritizes all objectives	G'	0.25	0.25	0.25	**22.14%**
3c	Goals are defined and system mainly prioritizes “relevance”	G'	0.10	0.10	0.10	**10.35%**
3d	Goals are defined and system mainly prioritizes “relevance” and “likability”	G'	0.10	0.30	0.10	**29.72%**
3e	Goals are defined and system mainly prioritizes “likability”	G'	0.10	0.50	0.10	**34.94%**

Table 4.4: Mean share of “likable” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E_{lik}) for several scenarios



(a) *Scenario 3a:* No goals are set with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$;
Scenario 3b: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$



(b) *Scenario 3c:* G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.10$;
Scenario 3d: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.30$, and $\lambda_{not} = 0.10$; and
Scenario 3e: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.50$, and $\lambda_{not} = 0.10$

Figure 4.5: Aggregated mean share of “fresh” plots in the selected plots for every 5% interval

The mean share of “likable” plots for is 26.93% for the entire simulation duration in Scenario 3a and 22.14% in Scenario 3b. As we see a significant number of “likable” plots selected both when no goals are specified and when goals activities are marked, our Hypothesis 3a is validated.

The share of "likable" plots increases as we go on, increasing the λ_{fre} value from 0.1 in Scenario 3c to 0.3 in Scenario 3d and further to 0.5 in Scenario 3e. Moreover, the same pattern in the share of "likable" plots in the three scenarios can be observed for each of the 5% intervals of the simulation duration in Figure 4.5b. All these results prove our Hypothesis 3b.

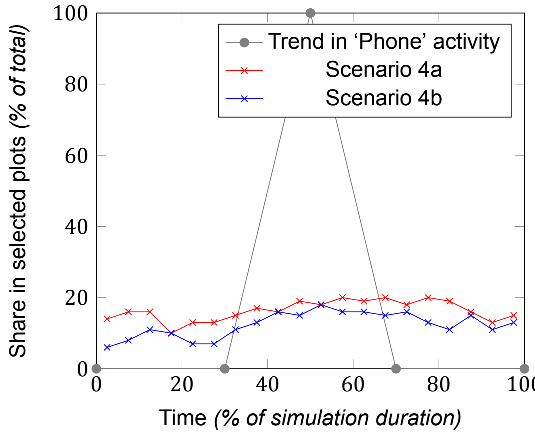
4.3.4. Noteworthiness

To evaluate the realization of the *noteworthiness* objective, Table 4.5 provides the mean values of the evaluation metrics $E_{not}(0, \frac{3T}{10})$, $E_{not}(\frac{3T}{10}, \frac{7T}{10})$, $E_{not}(\frac{7T}{10}, T))$, and E_{not} for five scenarios for the simulation duration.

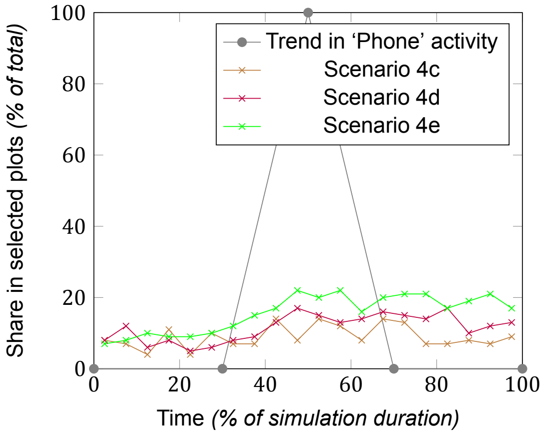
Figure 4.6 depicts the results from evaluating the optimization engine for every 5% interval for each of the five scenarios. Additionally, the figure plots the trend in "Phone" data stream that underlies the evaluation metrics for the "noteworthiness" component.

	Scenario	Goals	λ_{fre}	λ_{lik}	λ_{not}	$E_{not}(0, \frac{3T}{10})$	$E_{not}(\frac{3T}{10}, \frac{7T}{10})$	$E_{not}(\frac{7T}{10}, T)$	E_{not}
4a	No goals are set	ϕ	0.25	0.25	0.25	13.24%	18.30%	16.58%	**3.39%**
4b	Goals are defined and system prioritizes all objectives	G'	0.25	0.25	0.25	8.21%	14.45%	13.72%	**3.49%**
4c	Goals are defined and system mainly prioritizes "relevance"	G'	0.10	0.10	0.10	7.56%	10.42%	10.90%	**1.19%**
4d	Goals are defined and system mainly prioritizes "relevance" and "noteworthiness"	G'	0.10	0.10	0.30	8.65%	14.66%	13.18%	**3.75%**
4e	Goals are defined and system mainly prioritizes "noteworthiness"	G'	0.10	0.10	0.50	7.94%	19.37%	17.74%	**6.53%**

Table 4.5: Mean share of "Phone usage" plots in the selected plots across all interactions in the three intervals of the simulation duration (i.e. $E_{not}(T_{start}, T_{start} + \frac{3T}{10})$, $E_{not}(T_{start} + \frac{3T}{10}, T_{end} - \frac{3T}{10})$, and $E_{not}(T_{end} - \frac{3T}{10}, T_{end})$) for several scenarios. The last column tabulates values for the main evaluation metric E_{not}.



(a) *Scenario 4a:* No goals are set with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$;
Scenario 4b: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.25$, $\lambda_{lik} = 0.25$, and $\lambda_{not} = 0.25$



(b) *Scenario 4c:* G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.10$;
Scenario 4d: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.30$; and
Scenario 4e: G' are marked as goal activities with hyperparameter values as $\lambda_{fre} = 0.10$, $\lambda_{lik} = 0.10$, and $\lambda_{not} = 0.50$

Figure 4.6: Aggregated mean share of "fresh" plots in the selected plots for every 5% interval

As provided in Table 4.5, the increase in the mean share of "noteworthy" plots (i.e. those depicting 'Phone' activity) in the central part is 3.93% for the entire simulation duration in Scenario 4a, where the goal activities are not marked. In Scenario 4b, we mark the goal activities incremental gain in “noteworthy” plots is 3.49%. The gain in “noteworthy” plots during the central part of simulation duration when there is significant trend in ‘Phone’ usage is not as significant as expected. Moreover, the trend for share of “noteworthy” plots in the scenarios 4a and 4b can be observed for each of the 5% intervals of the simulation duration in Figure 4.6a and it too shows only a marginal increase in share of “noteworthy” plots during the central part of simulation duration. Therefore, these results provide insufficient evidence to validate our Hypothesis 4a (H_{not}).

The share of “noteworthy” plots increases as we proceed with increasing the λ_{not} value from 0.1 in Scenario 4c to 0.3 in Scenario 4d and further to 0.5 in Scenario 4e. However, as with the previous hypothesis, the sensitivity of the optimization engine to λ_{not} is rather weak and the results are thus inconclusive for our Hypothesis 4b (H'_{not}).

4.4. Efficacy of proposed approach

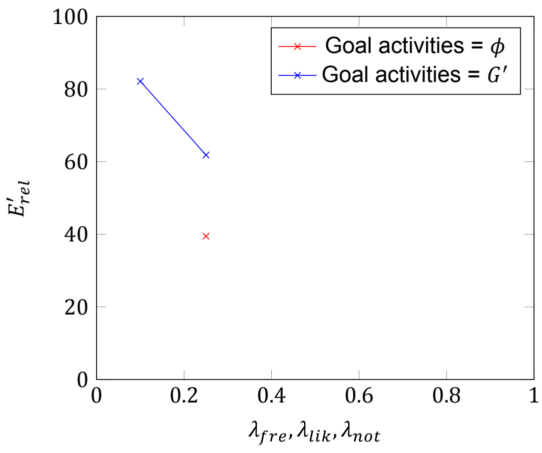
Table 4.6 summarizes the results obtained for all scenarios discussed earlier in this chapter. Figure 4.7 illustrates the results obtained for E'_{rel}, E'_{fre}, E'_{lik}, and E'_{not} for various of hyperparameters λ_{fre}, λ_{lik}, and λ_{not}.

	Scenario	Goals	λ_{fre}	λ_{lik}	λ_{not}	E'_{rel}	E'_{fre}	E'_{lik}	E'_{not}
1a/2a/3a/4a	No goals are set	ϕ	0.25	0.25	0.25	39.45%	30.62%	26.93%	3.39%
1b/2b/3b/4b	Goals are defined and system prioritizes all objectives	G'	0.25	0.25	0.25	61.82%	25.79%	22.14%	3.49%
1c/2c/3c/4c	Goals are defined and system mainly prioritizes “relevance”	G'	0.10	0.10	0.10	82.14%	15.43%	10.35%	1.19%
2d	Goals are defined and system mainly prioritizes “relevance” and “freshness”	G'	0.30	0.10	0.10		38.56%		
2e	Goals are defined and system mainly prioritizes “freshness”	G'	0.50	0.10	0.10		43.68%		
3d	Goals are defined and system mainly prioritizes “relevance” and “likability”	G'	0.10	0.30	0.10			29.72%	
3e	Goals are defined and system mainly prioritizes “likability”	G'	0.10	0.50	0.10			34.94%	
4d	Goals are defined and system mainly prioritizes “relevance” and “noteworthiness”	G'	0.10	0.10	0.30				3.75%
4e	Goals are defined and system mainly prioritizes “noteworthiness”	G'	0.10	0.10	0.50				6.53%

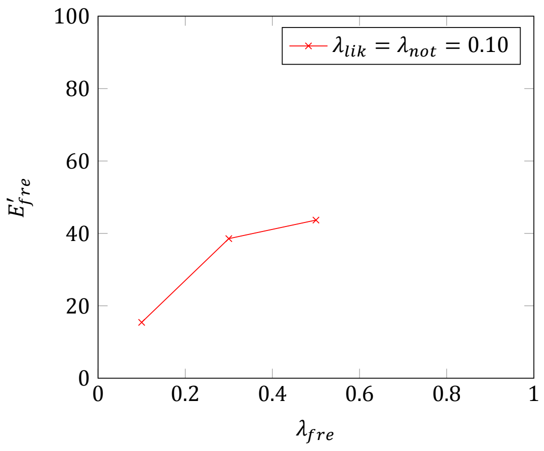
Table 4.6: Mean share of “relevant”, “fresh”, and “likable” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E'_{rel}, E'_{fre}, and E'_{lik} respectively) for several scenarios

The inverse relationship between the λ values and selection of “relevant” plots is made clear by Figure 4.7a, which displays the mean share of “relevant” plots in the selected plots across all interactions for the entire simulation duration for different values of hyperparameters. The positive correlation between the λ_{fre} value and selection of “fresh” plots are made evident by Figure 4.7b. The positive correlation between the λ_{lik} value and selection of “likable” plots is evident from Figure 4.7c. A positive correlation, albeit weak, is evident between the λ_{not} value and selection of “noteworthy” plots from Figure 4.7d.

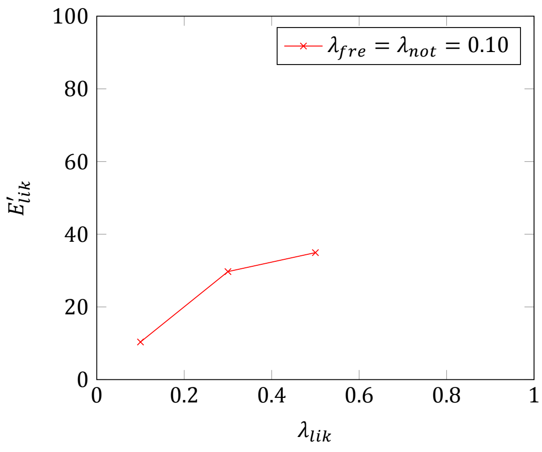
Based on the results obtained in Table 4.6 and Figure 4.7 and as discussed in previous sections, Table 4.7 summarizes our findings on the efficacy of the proposed system for achieving the objectives of “relevance”, “freshness”, “likability”, and ”noteworthiness”. As evident from Table 4.7, our optimization engine shows its usefulness in ensuring selection and calibrating the relative share of “relevant”, “fresh”, and “likable” plots. However, our evaluation results provide inconclusive evidence regarding the efficacy of the proposed optimization engine for prioritizing the “noteworthy” plots for selection.



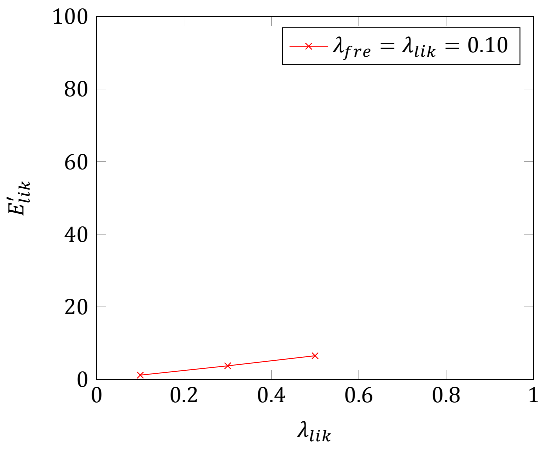
(a) Mean share of “relevant” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E'_{rel}) for different values of hyperparameters while having a) no goals specified; and b) G' as goal activities. In the above analysis, all the λ values are equal (i.e. $\lambda_{fre} = \lambda_{lik} = \lambda_{not}$)



(b) Mean share of “fresh” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E'_{fre}) for different values of hyperparameters.



(c) Mean share of “likable” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E'_{lik}) for different values of hyperparameters.



(d) Mean share of “noteworthy” plots in the selected plots across all interactions for the whole of simulation duration (i.e. E'_{not}) for different values of hyperparameters.

Figure 4.7: Results obtained for E'_{rel}, E'_{fre}, E'_{lik}, and E'_{not} for various of hyperparameters λ_{fre}, λ_{lik}, and λ_{not}.

Hypothesis	Evaluation objective	Controlled parameters(s)	Varied parameter(s)	Scenarios compared	Evaluation Metric	Hypothesis validation from results
H_{rel}	Is the system selecting "relevant" plots on marking the goal-activities?	$\lambda_{fre}, \lambda_{lik}, \lambda_{not}$	G'	1a, 1b	E_{rel}	Yes
H'_{rel}	Is the system sensitive to λ-values to allow calibration of the share of "relevant" plots?	G'	$\lambda_{fre}, \lambda_{lik}, \lambda_{not}$	1b, 1c	E_{rel}	Yes
H_{fre}	Is the system selecting "fresh" plots both when no goals have been specified and when they have been marked?	$\lambda_{fre}, \lambda_{lik}, \lambda_{not}$	G'	2a, 2b	E_{fre}	Yes
H'_{fre}	Is the system sensitive to λ_{fre} value to allow calibration of the share of "fresh" plots?	$G', \lambda_{lik}, \lambda_{not}$	λ_{fre}	2c, 2d, 2e	E_{fre}	Yes
H_{lik}	Is the system selecting "likable" plots both when no goals have been specified and when they have been marked?	$\lambda_{fre}, \lambda_{lik}, \lambda_{not}$	G'	3a, 3b	E_{lik}	Yes
H'_{lik}	Is the system sensitive to λ_{lik} value to allow calibration of the share of "likable" plots?	$G', \lambda_{fre}, \lambda_{not}$	λ_{lik}	3c, 3d, 3e	E_{lik}	Yes
H_{not}	Is the system selecting "noteworhtly" plots both when no goals have been specified and when they have been marked?	$\lambda_{fre}, \lambda_{lik}, \lambda_{not}$	G'	4a, 4b	E_{not}	Maybe
H'_{not}	Is the system sensitive to λ_{not} value to calibration of the share of "noteworthy" plots?	$G', \lambda_{fre}, \lambda_{lik}$	λ_{not}	4c, 4d, 4e	E_{not}	Maybe

Table 4.7: Summary of results from evaluation of proposed user-interface optimization engine

5 Conclusion

An algorithm for selecting plots of user activities that adapts to the changing requirements of a user while managing to offer maximum usefulness in the information provided at every interaction, is key to securing pleasant and continued use of a self-tracking tool. It is especially true when we have high volume user data streams along with computational power and UI capabilities to analyse and present in umpteen ways.

As seen in last section of the previous chapter, we have successfully demonstrated design and development of a selection algorithm that delivers relevant yet fresh selection of visualizations for a quantified-self user interested in keeping track of information related to several activities. Moreover, our algorithm learns from user preferences provided by them over time to prioritise UI elements that are particularly found useful by the user. We have thus been able to achieve three out of four objectives we set forward in our research question for our UI optimization engine.

5.1. Reflection

Available data, computational power and UI capabilities are always on an increase. Wearable devices have contributed to the volume of available data. Increasingly capable hardware on smartwatches and smartphones have increased the available computational power on the edges and their always connected nature enables them to tap into much higher computational power available through APIs of several cloud services. Finally, the devices are increasing in their UI capabilities with the wearable and more recently that of AR/VR hardware. Therefore, we anticipate increasing importance of our UI optimization engine for self-tracking tool that thrives exactly in the aforementioned conditions.

Selection algorithm functions as the most critical interface where computer output gets tuned to user requirements. For an application such as self-tracking tool, making this interaction intelligent and user-centric is of utmost importance. People make non-obligatory personal efforts in making use of self-tracking tools. Therefore, it becomes critical to ensure consistent usage of these tools by the users. Our optimization approach was guided by this understanding of keeping the user's preferences and limitations at the center. We find that an effective UI optimization engine is not only central to the full-stack development of a self-tracking tool but also plays a key role in the bigger picture by assisting users in achieving their wellness goals.

5.2. Limitations

While we use real-life data for evaluation of our system, a multiple user study could not be carried out due to objective reasons. A lab study along the lines of what was carried out in [15] would have been most suitable since it also deals with UI optimization using combinatorial techniques. The same holds for a user study spread over several weeks that would also have been useful in making a qualitative evaluation of the optimization engine to complement the quantitative analysis of the selection results. Additionally, the scenarios aggregate results by changing values for only the subsets of variables. For examples, multiple runs can be made for a set of values of hyperparameters. Finally, while the UI element features already allows for quantification of the distance from goal values, the performance of this feature in the selection output has not been evaluated. This, coupled with recent implementations of the interior point method [28], can be explored for faster computation.

5.3. Future Work

In addition to the need to expand the evaluation protocol by addressing the limitations of the current work, as explained in the previous section, here we elaborate on some further suggestions on how to expand the scope of this research. To start with, we used a web-based (optimized for desktop) UI delivery system. Introducing another UI output device, such a smartphone, would be an interesting addition to the research question. For this we would need to augment the optimization engine to allow selection of UI elements based on capabilities of the new UI output device. Furthermore, the UI elements currently comprise only of plots. However, they could also include push notifications using a mobile app, summary emails, custom elements on watch-face of a smartwatch etc. Therefore, expanding the optimization scope across different classes of UI elements would be a worthwhile research direction. Lastly, we can consider alternative optimization techniques in addition to the Integer Programming method used in our study. Particularly interesting wwould be to explore application of reinforcement learning methods to achieve better performance. To sum up, future work on the study can look into introducing new classes of UI elements, new classes of UI output devices, and adding more optimization techniques. Most importantly, the current study as well aforementioned future exploration areas would greatly benefit from an elaborate, long-term multi-user study.

Bibliography

[1] "Wearable Devices Are Connecting Health Care to Daily Life". In: *The Economist* (May 2, 2022). ISSN: 0013-0613. URL: `https://www.economist.com/technology-quarterly/2022/05/02/wearable-devices-are-connecting-health-care-to-daily-life`.

[2] *The Quantified Self | May 7th 2022*. The Economist. May 7, 2022. URL: `https://www.economist.com/weeklyedition/2022-05-07`.

[3] Barry Schwartz. *The Paradox of Choice: Why More Is Less*. HarperCollins, Dec. 22, 2003. 296 pp. ISBN: 978-0-06-000568-9. Google Books: `zutxr7rGc_QC`.

[4] Eun Kyoung Choe et al. "Understanding Quantified-selfers' Practices in Collecting and Exploring Personal Data". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada). CHI '14. New York, NY, USA: ACM, 2014, pp. 1143–1152. ISBN: 978-1-4503-2473-1. DOI: `10.1145/2556288.2557372`.

[5] Matthias Bode. "The Digital Doppelgänger Within". In: *Assembling Consumption* (2016), pp. 119–135.

[6] Tom Fawcett. "Mining the Quantified Self: Personal Knowledge Discovery as a Challenge for Data Science". In: *Big Data* 3.4 (Dec. 2015), pp. 249–266. ISSN: 2167-647X. DOI: `10.1089/big.2015.0049`.

[7] Melanie Swan. "The Quantified Self: Fundamental Disruption in Big Data Science and Biological Discovery". In: *Big Data* 1.2 (June 2013), pp. 85–99. ISSN: 2167-6461. DOI: `10.1089/big.2012.0002`. pmid: `27442063`.

[8] Peter Tolmie et al. "Unremarkable Computing". In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Apr. 20, 2002, pp. 399–406. ISBN: 978-1-58113-453-7. DOI: `10.1145/503376.503448`.

[9] Qian Yang, Aaron Steinfeld, and John Zimmerman. "Unremarkable AI: Fitting Intelligent Decision Support into Critical, Clinical Decision-Making Processes". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk). CHI '19. New York, NY, USA: ACM, 2019, 238:1–238:11. ISBN: 978-1-4503-5970-2. DOI: `10.1145/3290605.3300468`.

[10] Paul Dourish. "What We Talk About When We Talk About Context". In: *Personal Ubiquitous Comput.* 8.1 (Feb. 2004), pp. 19–30. ISSN: 1617-4909. DOI: `10.1007/s00779-003-0253-8`.

[11] Jakob E. Bardram et al. "Activity-Centric Computing Systems". In: *Communications of the ACM* 62.8 (July 24, 2019), pp. 72–81. ISSN: 0001-0782. DOI: `10.1145/3325901`.

[12] Lobna Hassan, Antonio Dias, and Juho Hamari. "How Motivational Feedback Increases User's Benefits and Continued Use: A Study on Gamification, Quantified-Self and Social Networking". In: *International Journal of Information Management* 46 (June 1, 2019), pp. 151–162. ISSN: 0268-4012. DOI: `10.1016/j.ijinfomgt.2018.12.004`.

[13] A. Oulasvirta. “User Interface Design with Combinatorial Optimization”. In: *Computer* 50.1 (Jan. 2017), pp. 40–47. ISSN: 0018-9162. DOI: `10.1109/MC.2017.6`.

[14] J. Derek Lomas et al. “Interface Design Optimization As a Multi-Armed Bandit Problem”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA). CHI ’16. New York, NY, USA: ACM, 2016, pp. 4142–4153. ISBN: 978-1-4503-3362-7. DOI: `10.1145/2858036.2858425`.

[15] Seonwook Park et al. “AdaM: Adapting Multi-User Interfaces for Collaborative Environments in Real-Time”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI ’18*. The 2018 CHI Conference. Montreal QC, Canada: ACM Press, 2018, pp. 1–14. ISBN: 978-1-4503-5620-6. DOI: `10.1145/3173574.3173758`.

[16] Krzysztof Gajos and Daniel S. Weld. “Preference Elicitation for Interface Optimization”. In: *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*. UIST ’05. New York, NY, USA: Association for Computing Machinery, Oct. 23, 2005, pp. 173–182. ISBN: 978-1-59593-271-6. DOI: `10.1145/1095034.1095063`.

[17] Suvi Tarkkanen et al. “Incremental User-Interface Development for Interactive Multiobjective Optimization”. In: *Expert Systems with Applications* 40.8 (June 15, 2013), pp. 3220–3232. ISSN: 0957-4174. DOI: `10.1016/j.eswa.2012.12.035`.

[18] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer Science & Business Media, Feb. 12, 2003. 2024 pp. ISBN: 978-3-540-44389-6. Google Books: `mqGeSQ6dJycC`.

[19] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Berlin Heidelberg, Mar. 12, 2014. 530 pp. ISBN: 978-3-662-21709-2. Google Books: `uYQ0kAEACAAJ`.

[20] Gang Yu. *Industrial Applications of Combinatorial Optimization*. Springer Science & Business Media, Mar. 14, 2013. 366 pp. ISBN: 978-1-4757-2876-7. Google Books: `lqfxBwAAQBAJ`.

[21] Antti Oulasvirta et al. “Combinatorial Optimization of Graphical User Interface Designs”. In: *Proceedings of the IEEE* 108.3 (Mar. 2020), pp. 434–464. ISSN: 1558-2256. DOI: `10.1109/JPROC.2020.2969687`.

[22] F. A. Ficken. *The Simplex Method of Linear Programming*. Courier Dover Publications, June 17, 2015. 68 pp. ISBN: 978-0-486-79685-7. Google Books: `E152CQAAQBAJ`.

[23] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, June 11, 1998. 488 pp. ISBN: 978-0-471-98232-6. Google Books: `zEzW5mhppB8C`.

[24] R.T. Marler and J.S. Arora. “Survey of Multi-Objective Optimization Methods for Engineering”. In: *Structural and Multidisciplinary Optimization* 26.6 (Apr. 1, 2004), pp. 369–395. ISSN: 1615-1488. DOI: `10.1007/s00158-003-0368-6`.

[25] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 2001. 1274 pp. ISBN: 978-0-07-232169-2. Google Books: `O0A4vgAACAAJ`.

[26] Miettinen Kaisa. *Nonlinear Multiobjective Optimization*. Vol. 12. International Series in Operations Research & Management Science. Boston, USA: Kluwer Academic Publishers, 1999.

[27] George Mavrotas. “Effective Implementation of the ε-Constraint Method in Multi-Objective Mathematical Programming Problems”. In: *Applied Mathematics and Computation* 213.2 (July 15, 2009), pp. 455–465. ISSN: 0096-3003. DOI: `10.1016/j.amc.2009.03.037`.

[28] Yongshuai Liu, Jiaxin Ding, and Xin Liu. “IPO: Interior-Point Policy Optimization under Constraints”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (04 Apr. 3, 2020), pp. 4940–4947. ISSN: 2374-3468. DOI: 10.1609/aaai.v34i04.5932.