Abstract
We apply a statistical modelling-based approach to exploring, analysing and predicting behavioural patterns of users of mobile software. The technique employed represents the behaviour of each user through a weighted mixture over data-generating distributions. In the described pilot study, we show how we have modelled the behaviour of over a hundred users of an iOS game. We illustrate how this modelling approach can be used to determine user play strategies and learning rates and show how this affects the length of time users keep returning to play the game. We describe our ongoing work, including feeding results of the modelling into the design process.

Author Keywords
Statistical modelling; admixture; mobile software

ACM Classification Keywords
H.5.2 [User Interfaces]: Evaluation/methodology

Introduction
Estimating, summarising, and predicting how an application is used is key to re-engineering and personalisation of applications. By logging every interaction a user makes, we can construct a user-trace – a chronological sequence of every in-app event representing how the user explores the application. Such
data sets can be the basis for statistical models of how users behave in navigating and using an interface.

In this work we describe an approach to exploratory analysis and visualisation of the dynamic behaviour of users of mobile software. A user-trace consists of a chronological sequence of in-app actions represented as symbols from some finite vocabulary. The information is inherently dynamic and user behaviour is highly heterogeneous within the user population. To address both these issues, each user is represented as a weighted mixture over Markov models - where each Markov model describes a different playing style. The model we learn is called a simplicial mixture of Markov chains [2]. Given such a probabilistic model, approximate inference methods are naturally applied to the problem of learning the parameters.

The potential applications of a model of user behaviour are numerous. Patterns of use encode high level user activities that will often mirror those defined by developers at design time, but this will not always be the case. Such patterns of use can be difficult to uncover, in particular when software is distributed widely, e.g. via 'app stores', and used globally. One high level aim of our own research is ‘design for appropriation’, i.e. system design for uses and contexts that developers may not be able to fully predict in advance.

We are developing an infrastructure in which logs of user interactions and contexts are streamed back to statistical analysis tools, thereby expressing scenarios and stereotypes that are common enough and specific enough to be the focus of new app design work. We intend to create app versions and/or plug-in extensions to apps, tailored to different clusters or patterns within the user population.

In future research we aim to use such stereotypes and patterns in our interactions with users, i.e. showing users how they are modelled so that their subsequent use is informed by this information, so that they understand why particular versions or extensions are recommended to them, and so that they might comment on or even correct our interpretation of their actions. The latter point reflects findings from earlier research, that pointed out that patterns of play that are not optimal in terms of gaining points or winning games may still be such fun for players that they are performed, shared and encouraged among players, e.g. the ‘pickpocket’ action in Treasure [1].

Here we describe our initial work in analysing users’ play strategies and learning rates in using an iOS game, and how this affects the length of time they continue to use the application.

**Modelling Hungry Yoshi**

As a test bed for this work, we analyse data logs of users of the Hungry Yoshi iOS application (see figure 1) - a game where users move around physically to seek out Wi-Fi access points (APs) in order to pick fruit from plantations (non-password secured APs) and feed the fruit to virtual characters called Yoshis (secured APs) [4]. Figure 2(a) describes the available in-app events for a simplified version of Hungry Yoshi, while 2(b) shows an example sequence of such symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Yoshi</th>
<th>Feed</th>
<th>Plant</th>
<th>Pick</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>View yoshi</td>
<td>Feed yoshi</td>
<td>View plant</td>
<td>Pick fruit</td>
</tr>
</tbody>
</table>

(a) Symbol table and descriptions of corresponding in-app events.

Yoshi Plant Plant Yoshi Plant Pick Pick Yoshi Feed Plant Pick

(b) An example of a typical user-trace.

Figure 2: Event descriptions and an example user-trace.
Bigram Model
As described in figure 2, the simplified version of the Hungry Yoshi application consists of four actions: Yoshi, Feed, Plant, and Pick. The simplified application can be considered a directed graph, where the \( u \)th user-trace is generated by a random walk of the graph up to a stopping time \( N(u) \).

Figure 3: Probabilistic directed graph describing the generation of paths in a Hungry Yoshi session. The actions Yoshi, Feed, Plant, and Pick are enumerated as \{1, 2, 3, 4\} in the order shown. The user begins the session in a randomly chosen state according to an initial distribution. The user with index \( u \) performs a random walk over the digraph using the transition probabilities shown, until some stopping time \( N(u) \).

Such a representation is described and visualised in figure 3, where, for the moment, the graph is assumed to be fully connected. The edge weights of the graph \( p_{i|j} \) determine the probability of the user playing the action \( i \) given that the last action played was \( j \). These are quantities we wish to learn from the data. This model constitutes a bi-gram model, where only dependencies between adjacent symbols in the user-trace are considered. More general \( n \)-gram models are obvious extensions to the current work.

Simplicial Mixture of Markov Chains
Let \( V = \{1, \ldots, L\} \) be the enumeration of the all possible in-app actions. Let \( S = \{S_u, u = 1, \ldots, M\} \) denote a set of paths, where each \( S_u \) is a sequence of \( N(u) \) symbols representing the sequence of in-app actions in \( V \), taken over the lifetime of the user \( u \). It is assumed that each \( S_u \) is randomly drawn independently from the others in the following way. It is assumed there exist \( K \) directed graphs of the form in figure 3, and a set of edge weights \( p_{i|j,k} \) for every edge \( j \to i \) and every graph \( k \). We also assume a global initial distribution over the nodes of each graph.

The different graphs are used to represent different playing styles - as different users are likely to traverse the graph in very different ways. Thus, each user is represented as a vector of mixture-weights \( \theta_u = (\theta_{1|u}, \ldots, \theta_{K|u}) \) over the corresponding graphs. A user is more likely to exhibit a particular playing style if the corresponding graph carries more weight under the user’s mixture representation. At each time step, the user \( u \) uses the mixture weights \( \theta_u \) to determine under which graph dynamics to make his or her next move. The larger \( \theta_{k|u} \), the more likely the user is to transition according to the dynamics of the \( k \)th graph.

The parameters of the above model can be learned by maximising the log-likelihood of the data under the given model [3], which expresses how much more likely the data is under one parameter than another. For \( K > 1 \), the log-likelihood is non-convex and has multiple maxima. For this reason, the procedure is randomly initiated 1000 times, and the result with the highest log-likelihood is taken as the learned estimate.
Preliminary results for the simplified Hungry Yoshi application are shown in figure 4, where we are assuming the existence of two underlying Markov chains. The corresponding digraphs for the maximum likelihood solutions for the parameters of the chains are shown in figures 4(a) and 4(b). The darkness of each arrow is proportional to its corresponding edge weight, with 1=black and 0=invisible. The digraph in figure 4(a) is similar to the digraph for the case $K = 1$ (not shown here). The digraph in figure 4(b) is very different. In this graph, Feed and Pick states have very high probabilities of returning to Plant and Yoshi, respectively. Random walks on the digraph in figure 4(b) will, with high probability, loop between the Yoshi and Plant states.

**Analysis**

**Successful vs Unsuccessful Playing Strategies**

To score highly in Hungry Yoshi, users need to find a pattern of picking fruit and feeding it to Yoshis. The behaviour seen in figure 4(a) appears to represent a successful playing style, where the graph is traversed more evenly, and more fruits are picked and Yoshis are fed. In contrast, figure 4(b) represents a playing style of someone who is not performing well, and not feeding Yoshis. This latter style is obviously not as the designer intended.

**Stopping Times**

Recall that each player is logged over their lifetime app usage, and characterised as being a weighted mixture of each playing style. Figure 5 shows two plots overlaid for the Hungry Yoshi data for $K = 2$. Consider first the green plot, which shows the weight $\theta_{1|u}$ for each user $u$ – that is, the weight of how heavily each player’s performance leans towards the playing style in figure 4(a). In figure 5, the users $u$ are ordered according to the size of $\theta_{1|u}$, so the users on the left of the graph are more heavily weighted towards the more successful strategy. It is clear that a large number of the users on the right (who carry no weight towards the successful strategy) are ‘purebreds’, with traces that conform solely to the graph in figure 4(b). In contrast, the population towards the left are an admixture of both types of generating distribution, i.e. they exhibit behavioural traits that are a mix of the two digraphs.

To give an idea of the relationship between the string length $N(u)$ (the total number of game events player $u$ has performed) and the weight $\theta_{1|u}$ of a user, a second plot is shown in blue stems with red markers, plotting $\log(N(u)/2)/s$ for each $u$. The aim of this plot is to show the correlation between the weight $\theta_{1|u}$ and the lifetime length $N(u)$ of the user. Investigations have shown $N(u)$ to be approximately log-normally distributed, hence the presence of the logarithm, while the scalings of $1/2$ and $s = \max_u \{\log N(u)\}$ are simply for visualisation purposes. Looking at this second plot, it is clear that the purebred subpopulation (to the right of the figure) who navigate the app according to the digraph in figure 4(b) are very short-term players and that the admixed subpopulation (to the left of the figure) contains the majority of long-term players.

From these results, it seems that those players who have played the game longer have spent a greater proportion of their time engaging in a successful playing strategy. It could therefore be inferred that every user undergoes a learning stage - a period of time in which they try to understand how the app works. At the end of the learning stage, those that have learned how to use the game and enjoyed it continue to play on, while those that have not understood the game or not enjoyed playing it stop using the app. The fact that every user must go through the
learning stage explains why long term users are an admixture of both playing styles, while users who quit before completing the learning stage are purebreds who have not fully understood the game, and have not got into a successful pattern of feeding Yoshis.

Returning to our aim of making tailored app versions and extensions available to users, in this case it is clear that time should be taken before choosing which versions and extensions to show to a user, allowing users to progress through the learning stage. Care should be taken to balance the wish to intervene early with the potential interruption and annoyance of those who are enjoying the process of mastering the game’s basics. Once a user appears to be playing successfully, it may then be appropriate to offer him or her the ability to announce that mastery (e.g. via posting on a social network) and to recommend new app versions and extensions – adaptations that may complicate the app in ways unsuitable for learners, but which build on successful patterns of play by offering new game features and functions. An example here, based on the digraph of 3(a), might be to introduce game features that support making plantations of particular fruit near to Yoshis that like such fruit. We emphasise that such design ideas should be based on the current patterns and dynamics of subpopulations as revealed by ongoing statistical modelling but subject to piloting with small numbers of users in the corresponding subpopulation as well as consultation with and polling of that subpopulation, thus refining the ‘mass participation’ design process outlined in [4], in which the entire population was consulted and polled about design changes indiscriminately.

We intend to continue investigating, modelling more complicated user behaviours and exploring larger data sets (we have released several instrumented applications to the public and collected log traces from hundreds of thousands of users). In particular, we are comparing the characterisations of user behaviour produced from the modelling process with more qualitative data gathered through questionnaires and interviews with game players.

Discussion and Future Work
We have presented a pilot study, applying statistical modelling to the problem of exploring and understanding the behavioural patterns of users of mobile software. The preliminary results have identified two playing styles for the Hungry Yoshi application, which are interpreted as a ‘learning’ and an ‘experienced’ style of play. Evidence suggests that short-term users are weighted strongly as learners, while long-term users are weighted more strongly as experienced players, in keeping with what one would expect.

Figure 5: Mixture weight $\theta_{1|u}$ and string length $N(u)$ for each user $u$, for $K = 2$ for simplified Hungry Yoshi model.
Open questions remain with regards to the method of inference. Firstly, the choice of \( K \): it is possible to include \( K \) as an unknown model parameter, removing the responsibility of choosing \( K \) from the researcher. However, for practical reasons it might be desirable to constrain \( K \) to be a low value such as 2 or 3. Another concern is how to deal with large action spaces. The simplified Yoshi game had only four available actions, but general software applications could have thousands of available actions, and a variety of different contexts in which they can be initiated. This can be tackled using structured representations, where actions are factored or categorised into relevant features, leading to compact representations of transition densities. This is closely tied to the method used to log the data. For a particular design or evaluation task, only high level actions may be relevant and thus logged, and low level intermediate ones might bring nothing more to the analysis. Finally, there are also other methods of approximate inference to be considered, such as sample-based and variational methods.

More generally, through the learning of statistical models, we aim to characterise the natural behaviours of an interacting population and to summarise the population based on these behaviours. When the work reaches fruition, we hope that researchers, developers, evaluators and users will use the tools developed to understand how users interact with their software and how to segment their user population. The hope is that such tools will lead to sophisticated design and personalisation schemes in mobile software engineering, where evaluators and developers will create new software versions or add-ons, based on the analysis, which are then made available to mobile devices. The users of those devices will, in turn, have app-integrated tools that enrich their understanding of these versions and add-ons, and that inform software choices and changes. Overall, we see the kind of statistical approach outlined in this paper as a key element in supporting ‘design for appropriation’ as we complete a cycle of large-scale app distribution, analysis, re-design, and end-user adaptation that might repeat multiple times throughout the lifetime of the app.

**Acknowledgements**

We thank the other members of the Populations project at the University of Glasgow and University College London for their collaboration, and acknowledge UK EPSRC funding (EP/J007617/1).

**References**


