Markov Chain Monte Carlo Synthetic Data Generation From Casino Slot Floor Event Data

by

Courtney Bonner

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

Master of Science

Individual Interdisciplinary Studies
Department of Statistics
University of Manitoba
Winnipeg

Abstract

The gaming industry produces vast amounts of data that can inform on many different kinds of research problems. Unfortunately, this data is rarely made available for research purposes and when it is, there are often limitations presented by the researcher's need to protect the data source's identity. In an effort to solve this problem, a Markov chain based statistical process for producing synthetic slot floor event data that is realistic and statistically similar to a provided real data set is proposed. Due to the uniqueness of this statistical process, two new methods of transition probability matrix estimation are introduced. This statistical process and the new probability matrix estimation methods are tested on an anonymous data set. The process is able to replicate event data and resultant session data distributions well, but falters in approximating a realistic time series.

Acknowledgment Page

Thank you to my advisor Dr. Saman Muthukumarana for his guidance throughout the writing process.

A big thanks to Mitacs and nQube Data Science Inc. for their financial assistance in the completion of this project.

Thank you to Dr. Jason Fiege and Dr. Anastasia Baran for offering their industry and academic expertise, computational resources, and their invaluable contributions without which this project would not have been possible.

Finally, thank you to my committee members Dr. Julien Arino and Dr. Carson Leung for their time in reviewing this work.

Dedication Page

This work is dedicated to the memory of my father for his perpetual encouragement and for believing in me always.

Contents

Abstract **i**

Acknowledgment Page **ii**

Dedication Page **iii**

Contents **v**

List of Tables **ix**

List of Figures **xiii**

1 Introduction **1**

1.1 Motivation 1

1.1.1 The Importance of Gambling Research 1

1.1.2 Scarcity of Casino Data Research 3

1.1.3 Simulation of Slot Floor Configurations 4

1.2 Slot Machines 4

1.3 Concept 8

1.4 Markov Chains 10

1.5 Literature Review 13

1.5.1 Simulation in Gambling Research 13

1.5.2 Studies of Real Casino Slot Floor Data 15

1.5.3 Probability Matrix Estimation 16

2 The Real Data Set **21**

2.1 About the Data 21

2.2 Data Cleaning 22

2.2.1 Conversion to Session Data 27

2.3 Data Preparation 31

3 Probability Matrix Estimation **33**

3.1 Uniform Occupancy Method 33

3.2 Probability Update Method 35

3.2.1 Counts Formulation 36

3.2.2 Probabilities Formulation 37

3.2.3 Classifying Parameters 39

3.2.4 Special Cases 42

3.2.5 Algorithmic Considerations 47
3.3 Challenges 47
3.3.1 Sacrificing Speed For Memory 47
3.3.2 Cycles 49
3.4 Bayesian Formulation 50
3.5 Results 52
3.5.1 Performance 52
3.5.2 Matrix Properties 57

4 Synthetic Data Generation **59**
4.1 Building Probability Matrices and Distributions 59
4.2 Data Structures 62
4.3 Data Generation 63

5 Results **65**
5.1 Fitting Distributions 66
5.2 Jensen-Shannon Divergence 69
5.2.1 Coin-in Per Event 73
5.2.2 Coin-in Per Session 78
5.2.3 Time Elapsed Between Events 81

5.2.4 Session Duration 85

5.2.5 Games Per Minute 88

5.2.6 Games Played Per Event 92

5.2.7 Games Played Per Session 98

5.2.8 Frequency of Machine Occurrences in Event Data . . . 101

5.2.9 Frequency of Machine Occurrences in Session Data . . 102

5.2.10 Frequency of Patron ID Occurrences in Event Data . . 105

5.2.11 Frequency of Patron ID Occurrences in Session Data . 108

5.3 The Time Series 108

5.3.1 Time Series Cycles 111

5.3.2 Time Series Distributions 119

5.4 Summary 129

6 Conclusions **133**

6.1 Limitations 136

6.2 Future Work 137

Bibliography **139**

List of Tables

1.1 Machine Properties Available From Data Set 6

2.1 Raw Data Fields 22

2.2 Machine Information Fields. 23

3.1 Number of Non-Zero Values in Probability Matrices 58

5.1 Results of fitting a Gaussian distribution to session duration, coin-in per session, and games per minute in the real data. . . 67

5.2 Results of fitting a Skew Normal distribution to session duration, coin-in per session, and games per minute in the real data. . . 67

5.3 Comparing the residuals between the distributions in generated data and the distributions fitted to the real data. Here, the distribution fitted to real data was compared to the distribution found in the generated data. 68

5.4 Comparing residuals between the distributions in generated and real data and their fitted distributions. Here, each quantity in each distribution received its own, unique fit. 70

5.5 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for coin-in per event. 73

5.6 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for coin-in per session. 78

5.7 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for time elapsed between events. 81

5.8 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for duration of sessions. 85

5.9 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for games per minute of session time. To obtain the games per minute value for each session, the number of games played during the session is divided by the session length in minutes. 91

5.10 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for games played per event. 92

5.11 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for games played per session. 95

5.12 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of machine occurrences in event data. 98

5.13 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of machine occurrences in session data. 102

5.14 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of player ID occurrences in event data. 105

5.15 Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of player ID occurrences in session data. 108

5.16 Comparing the length and time span of real and generated event data sets ($J = 10$). The ratios are equal to the number of events/sessions/visits per day in the generated data divided by the number in the real data. 127

5.17 Comparing the length and time span of real and generated higher occupancy event data sets (J = 100). The ratios are equal to the number of events/sessions/visits per day in the generated data divided by the number in the real data. 128

5.18 Means of the machines played per visit distributions in all data sets. 129

List of Figures

2.1 The theoretical win (machine hold multiplied by its total coin-in) of each machine divided by the number of days on which at least one session is observed. The units of the y-axis are assumed to be in cents even though this is likely inaccurate for a minority of the machines. 26

2.2 The distribution of coin-in in the session and event data. Top: The distribution of coin-in in a session. Bottom: The distribution of the change in the coin-in meter during an event. 28

3.1 The mean absolute error as a function of the number of data points on a linear scale with $\alpha = -1$. 42

3.2 The distribution of all possible mean errors when $\alpha = -1$. . . 43

3.3 The mean absolute error as a function of the number of data points on a logarithmic scale with $\alpha = -0.5$. 43

3.4 The mean absolute error as a function of the number of data points on a logarithmic scale with $\alpha = 0$. 44

3.5 The mean absolute error as a function of the number of data points on a logarithmic scale with $\alpha = 0.5$. 44

3.6 A collection of box plots comparing the convergence properties of all tested values of α. 45

3.7 An example of a small cycle that is simple to detect without the use of communicating classes. A state within the cycle is transitioned to from another state outside the cycle. The cycle is inescapable. 50

3.8 An example of a small cycle that is difficult to detect without the use of communicating classes. A state within the cycle is transitioned to from another state outside the cycle from which the cycle is inescapable. 50

3.9 The base-10 logarithm of the mean absolute error between the initial probability matrix and the resultant probability matrix in three different scenarios. In the Probability Update method, $\alpha = 0$. Top: The mean absolute error as a function of the number of players. Middle: The mean absolute error as a function of the number of iterations at high occupancy (80%). Bottom: The mean absolute error as a function on the number of iterations at low occupancy (40%). 54

3.10 The base-10 logarithm of the maximum absolute error between the initial probability matrix and the resultant probability matrix in three different scenarios. In the Probability Update method, $\alpha = 0$. Top: The mean absolute error as a function of the number of players. Middle: The mean absolute error as a function of the number of iterations at high occupancy (80%). Bottom: The mean absolute error as a function on the number of iterations at low occupancy (40%). 55

5.1 The difference the relative frequency of coin-in per session between data generated using the given probability estimation method and the real data. 74

5.2 Relative frequency of coin-in per event in real data and data generated with the given probability estimation method plotted together. 75

5.3 The difference in the relative frequency of coin-in per event between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 76

5.4 Relative frequency of coin-in per session in real data and data generated with the given probability estimation method plotted together. 79

5.5 The difference in the relative frequency of coin-in per session between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 80

5.6 Relative frequency of time between consecutive events in real data and data generated with the given probability estimation method plotted together. 82

5.7 The difference in the relative frequency of the time between consecutive events between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 83

5.8 Relative frequency of session duration in real data and data generated with the given probability estimation method plotted together. 86

5.9 The difference in the relative frequency of session duration between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 87

5.10 Relative frequency of games per minute of session time in real data and data generated with the given probability estimation method plotted together. 89

5.11 The difference in the relative frequency of games per minute of session time between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 90

5.12 Relative frequency of games played per event in real data and data generated with the given probability estimation method plotted together. 93

5.13 The difference in the relative frequency of games played per event between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 94

5.14 Relative frequency of games played per session in real data and data generated with the given probability estimation method plotted together. 96

5.15 The difference in the relative frequency of games played per session between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. 97

5.16 Relative frequency of machine occurrences in events in real data and data generated with the given probability estimation method plotted together. The machines are assigned an index in order of their prevalence in the real data. This index is what is reported here. 99

5.17 The difference in the relative frequency of machine occurrences in event data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The machines are assigned an index in order of their prevalence in the real data. This index is what is reported here. 100

5.18 Relative frequency of machine occurrences in sessions in real data and data generated with the given probability estimation method plotted together. 103

5.19 The difference in the relative frequency of machine occurrences in session data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The machines are assigned an index in order of their prevalence in the real data. This index is what is reported here. 104

5.20 Relative frequency of player occurrences in event data in real data and data generated with the given probability estimation method plotted together. 106

5.21 The difference in the relative frequency of player occurrences in event data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The players are assigned an index in order of their prevalence in the real data. This index is what is reported here. 107

5.22 Relative frequency of player occurrences in sessions in real data and data generated with the given probability estimation method plotted together. 109

5.23 The difference in the relative frequency of player occurrences in session data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The players are assigned an index in order of their prevalence in the real data. This index is what is reported here. 110

5.24 Plots of the number of events per day in each generated data set and the one real data set in chronological order. 112

5.25 Plots of the number of active machines per day in each generated data set and the one real data set in chronological order. . . . 113

5.26 Plots of the number of unique player IDs per day in each generated data set and the one real data set in chronological order. 114

5.27 A discrete Fourier transform was performed on the events per day signal. The magnitude of the frequencies found in the signal is plotted for each generated data set ($J = 10$) and the real data. 116

5.28 A discrete Fourier transform was performed on the events per day signal. The magnitude of the frequencies found in the signal is plotted for each generated data set ($J = 100$) and the real data. 117

5.29 Plots of the number of sessions per day in each generated data set and the one real data set in chronological order. 118

5.30 Histograms of the number of events per day in each generated data set and the one real data set. 120

5.31 Histograms of the number of active machines per day in each generated data set and the one real data set. 121

5.32 Histograms of the number of unique player IDs per day in each generated data set and the one real data set. 122

5.33 Histograms of the number of sessions per day in each generated data set and the one real data set. 124

5.34 Histograms of the number of players per thirty minute interval in each generated data set and the one real data set. 125

5.35 Histograms of the number of machines played per visit in lower occupancy ($J = 10$) generated data and real data. 130

Chapter 1

Introduction

1.1 Motivation

1.1.1 The Importance of Gambling Research

According to the American Gaming Association, about $41 billion dollars in tax revenue are generated annually by casinos with 1.8 million people being employed. The Manitoba Liquor and Lotteries Corporation (MLCC) reported in 2019 that casinos provided $74 million dollars in tax revenue to the province of Manitoba and VLT (Video Lottery Terminal) operations provided $199 million dollars in tax revenue that year. VLTs, or slot machines, contributed over 30% of the MLCC's net revenues which provided over $100 million dollars to venues hosting VLTs including $68 million dollars directly to First Nations communities. Gambling, via slot machines in particular, is a popular pastime that contributes greatly to local economies and generates tax revenue that contributes to funding for important government programs such as social services and infrastructure.

Gambling research from universities and gaming institutes such as the International Gaming Institute at the University of Nevada Las Vegas (UNLV) informs on its individual, societal, and economic impacts. It can also inform on the impact that casino operator decisions have on their customer base, allowing operators to maximize their revenue, in turn maximizing tax revenues and their economic contributions. In 1974, the first Gambling and Risk Taking Conference took place where researchers came together to present their latest findings in the field. The conference was founded by Dr. Bill Eadington, who is also largely considered to be the North American founder of gambling studies. It has taken place every three years since.

In addition to its economic interests, gambling research can be of academic interest as well. In Dr. Eadington's book (1), he argues that gambling provides real world applications for theories in probability, statistics, decision-making, and information. This work is an example of this where slot floor event data provided an application for a very specific type of statistical process and allowed for study of its efficacy in modelling and simulating real world slot player behaviour.

1.1.2 Scarcity of Casino Data Research

Data in the casino industry is closely guarded and rarely shared without the protection of a non-disclosure agreement (NDA). Lack of data availability is an obstacle to research and innovation by third parties in the industry. If data could be shared freely, it would allow for published studies to be replicated, and for studies to be done on data from multiple sources when currently, researchers find it difficult enough to obtain data from just one source.

Academic studies on casino data are scarce, but the data itself is abundant due to heavy regulation of the industry. Often, studies are done on simulated data (2), hypothetical data (3) or data that has been modified by researchers (4). When studies are done on real data, the source is most often obscured (2), (5), (6), (7), (8), (4), (9), (10), (11), (12), (13), (14). In one article, the data source is named, but the authors note that a limitation of the research is that the source used an unknown methodology to select the sample that was provided (15).

In (16), dice game plays stored on a publicly available blockchain were used to study gambling behaviour. The authors note that obtaining "real-life" gambling data is "very difficult, if not impossible". In (4), the authors also mention the difficulties of obtaining data from the industry.

In many studies, the authors are only able to obtain a data sample from a single source. This is often a limitation of gambling studies done on casino data (15), (5), (7), (10), (11), (12), (13), (14). In one study where data from multiple casinos was used, the data is already several years old at the time that the study is conducted (8). This is a common limitation found in gambling research in general (7), (12), (13), (8), (14).

1.1.3 Simulation of Slot Floor Configurations

Additionally, the methods outlined here may be of interest to casino operators for simulation of different combinations of players and machines. The number of players and machines are parameters to the simulation, allowing full control over the occupancy of the theoretical casino in the simulated data set as well as the set of machines available to the players in the simulation. This would allow for casinos to test out different mixes of machines and how their patrons would react or how changes to their customer base might react to their current mix of machines. The only requirement is that the types of players or machines used in the simulation were present in the data that is used to construct the simulation. The capabilities of the simulation could be expanded to utilize segmentation of machines or players to further study how more or fewer players or machines of different types would affect floor activity.

1.2 Slot Machines

A casino slot floor can be defined by a machine properties data set as well as the properties that make up an event that occurs on a machine. The machine properties data set specifies properties of each machine that are static and not affected by player activity. In the data set used in this experiment, the machine properties include machine number, manufacturer, game name, hold, cabinet type, and location on the slot floor. The slot floor refers to the encompassing area where the machines are located for patron use. See table 1.1 for descriptions of these properties.

Other properties that a machine has but were not included in the data set provided for this research are contained in "pay tables". These pay tables include information about how the probabilities of certain payouts are distributed. The volatility, or pay table standard deviation, defines how much these probabilities are allowed to vary by. The hit frequency is the probability of a payout of any amount. Pay tables often also include hold percentage as well, but this was included as part of the data provided for this research and is described in table 1.1.

An event on a slot machine is any atomic (from our perspective) operation triggered by a patron, a staff member, or by the machine itself while the machine is in operation on the slot floor. These are henceforth referred to as player-initiated events, staff-initiated events, and machine-initiated events respectively. An event can be uniquely identified by the machine it occurred on, an event code describing the type of event, the time the event occurred at, and the player identification number of the player that initiated the event if

Machine Property Name	**Description**
Machine Number	A unique, numeric identifier for this machine.
Manufacturer	The name of the manufacturer for this machine.
Game Name	The name of the game that is displayed to players.
Hold	The percentage of inserted funds that the machine keeps on average.
Cabinet Type	Refers to the type of box the machine resides in.
Bank	Machines are often grouped into banks on the slot floor. The bank number identifies the bank that this particular machine resides on.
Seat	Identifies the location of the machine within the bank.

Table 1.1: Machine Properties Available From Data Set

applicable. Most casinos allow players to be "uncarded" if they wish, in which case they will not have a player identification number.

The *meters* on a machine count some quantity throughout the life of the machine. In our data, there are three meters: a coin-in meter, a coin-out meter, and a games played meter. The coin-in meter counts the amount of currency inserted into the machine (bets) while the coin-out meter counts the amount of currency dispensed by the machine (player winnings). The units of the coin-in and coin-out meters are either in cents, multiples of the game denomination, or a mixture of the above. The game denomination is base cost of playing a single game on that machine. The games played meter counts the number of games played on the machine. A game on a slot machine usually consists of a lever pull or button press. The meters on the machine are often updated when an event occurs, but not always. Meters might be updated with each event, only for certain event types, or at regular time intervals. It is often hard to tell when meters are updated without confirmation from the data source themselves, but we can look for patterns in the data.

Defining the concept of a player session is useful when discussing slot machine play. A player session spans the length of time between when a player sits down at a machine, plays some games, and then gets up to do something else. A session consists of the machine identifier for the game they played, their player identification number if available (players may be uncarded if they choose to be), the total coin-in on the machine during the session, the total coin-out on the machine during the session, the total number of games played during the session, and the start and end times of the session. One of the ways that we have of validating our synthetic data generation method is to convert the generated event data into sessions, and compare the distributions of coin-in, coin-out, and games played to the distributions normally found in real session data. For the remainder of this paper, *session data* will refer to event data that has been converted to a set of player sessions and *event data* will refer to a set of events.

Session data can be further reduced and combined into higher-level data structures called "visits". A player's visit to the casino consists of all the sessions they played between entering and exiting the casino. A visit looks similar to a session, but has potentially many machine numbers associated with it if the player played multiple machines during their visit. The start time of the visit is the start time of their first session and the end time of their visit is the end time of their last session. Coin-in, coin-out, and games played are added up across all of their sessions.

When converting sessions into visits, it is necessary to define a time-out value that indicates when the player is believed to have left the casino and thus ended their visit. When converting data to visits, a time-out of two hours was used. If the player returned two hours after a session, it was considered be part of a new visit.

See (17) for more detail.

1.3 Concept

The major contribution of this work is probability matrix estimation methods for use when states of a statistical process are in demand by many individuals but cannot be occupied by more than one individual at the same time. A synthetic data generation algorithm that takes in real casino slot floor event data and produces simulated data based on the statistical distribution of transitions between state-like events is provided as a proof of concept.

The probability matrix is calculated based on the relative frequency of observations of player-initiated events from casino slot floor event data and the occupancy of events which are analogous to states of a Markov process.

The probability matrix is then used similarly to a Markov transition matrix to facilitate a first order Markov process to generate synthetic data consisting of players moving around the slot floor in ways similar to those found in the authentic data. The probability matrix is technically not a Markov transition matrix however because the occupancy of states at the time of the transition alters the probabilities of transitions such that the probability of transitioning to an occupied state is zero and its probability is redistributed among unoccupied states. In a true first-order Markov process, a transition's probability is solely dependent on most recent transition. Here, transition probability is dependent on the previous transition and the occupancy of states at the time of the transition.

A set of events occurring on a casino slot floor would be incomplete without associated quantities such as currency spent and won, the number of games played, and the time that passed between events. Distributions for these quantities are formed by gathering observations from the real data for each of these quantities for every possible transition. Monte Carlo simulation can then be used to draw from these distributions to form complete event data records.

As for why it was chosen to simulate event data rather than session data, simulating higher granularity data allows for higher variation in the results of the simulation. If we were to simulate session data, it would just result in a shuffling of the sessions observed, while simulating event data allows for the creation of completely different sessions while still maintaining statistical properties of the real data. If a high fidelity simulation of event data is achieved, the resulting simulated data will be even less recognizable than if the simulation were performed on session data.

1.4 Markov Chains

The stochastic process utilized for generating synthetic data is not technically a Markov process, but many of the properties still apply. Markov chains are discussed in detail in (18) and (19). A summary is provided here.

A Markov chain is a stochastic process wherein the next state of the process $X_{n+1} \in S$ is only conditionally dependent on the immediately previous state $X_n \in S$ where S is the state space of the process. This is called the Markov property as given in 1.1.

$$P\{X_{n+1} = j | X_0, \ldots, X_n\} = P\{X_{n+1} = j | X_n\} \tag{1.1}$$

A transition probability matrix $\mathbb{P}$ can be defined such that elements p_{ij} of $\mathbb{P}$ give the probability of an individual undergoing the transition from state i to state j. The rows of $\mathbb{P}$ represent the origin state and the columns represent the destination state. As such, $\mathbb{P}$ is a right-stochastic matrix, meaning that $\sum_{j=1}^{|S|} p_{ij} = 1$ where S is the set of states in the Markov chain. A Markov chain is homogeneous in time if the probabilities p_{ij} are not time-dependent.

The probability of an individual governed by the Markov process moving from state i to state j in k steps is given by the entry in the i^{th} row and j^{th}

column of the transition matrix $\mathbb{P}^k$ which is the one-step transition matrix to the k^{th} power. Notationally, the k-step probabilities are given by $P\{X_{n+k} = j|X_n = i\} = \mathbb{P}^k_{(ij)} = p^{(k)}_{ij}$. Proof of the k-step probability equation is found in (19). This leads us to the Chapman-Kolmogorov equation given in 1.2. It says that the probability of transitioning from state i to state j in $m+n$ steps is the probability of transitioning from state i to any state $k \neq i, j$ multiplied by the probability of transitioning from state k to state j summed up over all $k \in S$.

$$p^{(m+n)}_{ij} = \sum_{k \in S} p^{(m)}_{ik} p^{(n)}_{kj};\ i, j \in S \tag{1.2}$$

It is useful to define a stopping time. A stopping time τ is a time during the evolution of the chain at which a specified condition that is a function of the history of the chain up until that point is met. The Strong Markov Property says that once a stopping time is reached, the chain can be considered to be reset. The probability of seeing any given sequence of states in the future is the same as it was when the chain started. See equation 1.3 (18).

$$\begin{aligned} P(X_{\tau+1} = j_1, \ldots, X_{\tau+m} = j_m | X_0 = i_0, \ldots, X_{\tau-1} = i_{\tau-1}, X_\tau = i) \\ = P(X_1 = j_1, \ldots, X_m = j_m | X_0 = 1) \end{aligned} \tag{1.3}$$

A state of a Markov chain can be classified as either transient or recurrent. A state is recurrent if the chain revisits that state in a finite number of steps with probability one. Additionally, a positive recurrent state is a state that has a finite mean return time. If the chain has a finite number of states, then all recurrent states in the chain are positive recurrent. If the probability of a state being revisited in a finite number of steps is less than one then it is a transient state. A state is said to be accessible from another state if the probability of travelling from the initial state to the final state in a finite number of steps is greater than zero. If two states are accessible from each other, they are said to communicate. If all states in the Markov chain communicate with each other, then the chain is irreducible.

To define the period of a state, we must first consider the possible return sequences of that state. A return sequence is a possible sequence of states leading back to the initial state. The greatest common divisor of all return sequences to a given state is the period of that state. If the period of a state is equal to one, then the state is said to be aperiodic. There are two ways for a state to be aperiodic: either a transition from itself back to itself is allowed, or all possible paths have lengths that are relatively prime to each other.

The long run behaviour of a Markov chain can be examined by computing its limiting distribution $\pi \in [0,1]^{|S|}$. Elements of the limiting distribution are defined analytically by $\pi_j = \lim_{n\to\infty} P(X_n = i|X_0 = j) = P_{ij}^{(n)}$. The limiting distribution is the distribution that is eventually reached after a finite number of iterations of the Markov chain, regardless of the initial distribution. The limiting distribution of a Markov chain is said to exist if the limits defined previously exist for every state $j \in S$ and those limits form a valid probability distribution. The stationary distribution, however, is the distribution that is invariant under the transition matrix P such that $\pi = \pi P$. If the chain begins with the stationary distribution, it will never diverge from that distribution. The stationary and limiting distributions are equal to each other if the Markov chain is positive recurrent (all states in the chain are positive recurrent), the chain is aperiodic (all states are aperiodic), and the chain is irreducible.

1.5 Literature Review

1.5.1 Simulation in Gambling Research

It is believed that time-on-device, or the amount of time a player can spend on a machine before they run out of money, is the greatest factor contributing to the player's enjoyment or perceived value for their money (4), (9). Slot machine play has been simulated using modified or hypothetical pay tables to determine how machine properties such as volatility (i.e. pay table standard deviation), hold percentage (i.e. par), and hit frequency contribute to a player's time-on-device so that games can be designed in a way that maximizes their entertainment value. Most casino patrons visit the slot machines for entertainment, so it is believed that if they perceive good entertainment value they will be enticed to visit again, further contributing to a casino's revenue.

In (4), simulations were done using modified par tables from an undisclosed source to determine the effect the hold percentage had on play. The only information that the authors had access to was machine pay tables, which include the hold percentage, probability of possible outcomes, and volatility. The amount of money that virtual players started with was varied and simulations were run until different stopping criteria were reached: when players had reached a specified amount of winnings or had gone bankrupt.

In (9), simulations similar to those in (4) were run, but this time machine volatility was varied. It was found that hold percentage did not have a statistically significant effect on the number of spins a player made before going bankrupt. However, an increased machine volatility caused a decrease in the number of spins that a player made before going bankrupt.

Demand for casino table games such as Baccarat and Poker was forecasted using simulated demand data in (2). The simulated data was constructed by generating auto-correlated uniform random numbers that selected for the number of seats that are occupied for the current time of day and day of the week according to probability distributions that were functions of the level of demand that is expected for the time of day and the type of table game being simulated. This simulated demand data was used as training and test data for the machine learning methods used to do the forecasting.

1.5.2 Studies of Real Casino Slot Floor Data

In (8), the effect that sportsbook operations had on slot machine coin-in was studied using real data from three different, unnamed casinos. It was found that increased sports-betting resulted in a comparatively small increase in coin-in for all three casinos. This small increase was not statistically significant. A similar study was conducted in (11) on a single Las Vegas hotel casino which had similar results.

Customer retention and churn prediction were studied in (5) where a data mining approach was used to identify online casino patrons who are at risk of churn for the purpose of targeted churn prevention. The method was tested on data obtained from an online casino operated by a United States land-based casino. The same data mining method was applied to the problem of identifying casino patrons who were likely to play both slots and table games (7), when they are often segmented into one group or the other. The data source was an unnamed Las Vegas casino. The likelihood for casino patrons to play both table games and slots was also studied in (14) using statistical techniques on a data set obtained from a Las Vegas strip casino. They experienced success in predicting whether players would play both.

In (12), a performance potential model was fitted to real casino slot floor data obtained from an unnamed Las Vegas strip hotel casino to understand the effect that characteristics of a machine's floor placement had on coin-in. It was found that aisle and end seats, as well as seats with higher ceilings, received higher coin-in. Games with a high volatility received lower coin-in. The type of cabinet the game was encased in had little effect on coin-in.

The viability of free-play marketing offers for customer retention was studied in (13). Data from an unnamed Las Vegas strip casino was used where players were offered either \$50 or \$100 of free play to determine if the offer affected their likelihood of returning. It was found that the cost of such programs was not justified as any revenue produced was not enough to offset the cost of the program.

Slot floor data from an unnamed Las Vegas casino was used to test a player segmentation method that would divide players into segments based on their frequency of play and the magnitude of their spending in (6). The data used was high-dimensional and sparse and effectively segmented the players despite these challenges.

1.5.3 Probability Matrix Estimation

Least squares estimators for the transition probabilities of time-homogeneous Markov chains are outlined in (20) and (21). In (21), estimation of transition probabilities of time-homogeneous Markov chains when only aggregate data is available such that it is known how many individuals occupy any given state at time t, but never which states those individuals transitioned from. An ordinary least squares method is outlined as well as some weighted least squares methods under the assumption that the number of individuals present in the sample remains constant between observations. Constraints on the probabilities being positive and each row summing to one are only briefly discussed and dismissed as often unnecessary since if improper probabilities result, then it is likely that a Markov chain is not a good solution to the problem. In (20), a similar least squares estimator is discussed in detail as well as a linear programming formulation to impose the constraints. A modification to the maximum likelihood estimator (Laplace smoothing) is detailed in (22) with the goal of improving the performance of estimation of Markov transition matrices for various loss measures. The "Add-$\sqrt{N_i/k}$" estimator from (22) is used as a benchmark for the new estimation methods. It is most useful when there is a small sample of data that may not contain unlikely, but possible, observations as it disallows zero probability events by definition.

In (20), a Bayesian estimator for aggregate data is also defined. The marginal distribution of each element of the transition matrix is defined to be such that the mean is $\frac{N_{ij}}{N_i}$ where N_{ij} is the number of individuals that transitioned from state i to state j and $N_i = \sum_j N_{ij}$. They decide on a multivariate beta probability distribution for the prior. The joint posterior probability function is also given. The Bayesian estimator was determined to be superior to the least-squares estimators and the maximum-likelihood estimator.

Markov chains were used to generate hypothetical industrial pipelines (made from physical pipes) using real designs by conditioning each subsequent pipeline component on the component previous to it in the pipeline in (23). A notable process introduced in this paper is that they defined distributions for the period of each component and only allowed a component to be introduced if the time since it was introduced previously is equal to the randomly drawn period for that component. This acted to prevent groupings of components that are unrealistic in reality. They found that the majority of generated pipelines were 88% similar to the majority of real pipelines. The biggest limitation is that the real pipelines came from a single project, bringing the ability of the model to generalize into question.

In (24), a maximum likelihood estimator of a Markov transition matrix given incomplete data is defined. The incomplete data does not contain observations for each individual at every time step. The likelihood function is maximized by the iterative EM algorithm.

In (25), a Markov transition matrix for modelling wind speed data is determined through evolutionary optimization. A simple 2-norm cost function is implemented with constraints forcing the rows of the matrix to sum to one and ensuring that every element is in the range $[0, 1]$. Good convergence was observed in low dimensional situations, but improvements were made so that convergence could reliably be made in high dimensional scenarios as well.

In (26), 10 years of wind speed data from a weather station in Tangiers, Morocco is used to determine a Markov transition matrix and generate synthetic wind speed data. The synthetic data matched the real data nicely in distribution and other evaluation methods outlined.

In (27), a process for the estimation of a transition matrix for the deterioration of concrete structures is proposed. Mechanistic-empirical models are used to generate sample deterioration paths on which the maximum likelihood estimator is used to calculate transition probabilities. Where data from these sample paths are unavailable, Bayesian estimation is used. In an experiment that considered the deterioration of a concrete bridge, the residuals were found to be 54% of those when the previously derived simulation method from (28) was used. In (28), a restricted least-squares method is used to estimate the transition probabilities.

Chapter 2

The Real Data Set

2.1 About the Data

The unprocessed event data was donated by nQube Data Science Inc. from an anonymous source. Research is able to be conducted and published on this data set because it is not under a confidentiality agreement of any kind. The data set spans the time period of sixteen months between May 27 2015 to September 27 2016. There is some missing data which could be confirmed by the end and beginning of some carded sessions not being present. For the purposes of this project, the effect of missing data is negligible. There are many more consecutive events than there are events missing.

Table 2.1 explains the purpose of each present and useful event data column provided. The columns explained there are only the ones that were used. There were other columns present with unknown functions.

Event Data Column Name	**Description**
Machine Number	Unique machine identifier.
Event Code	Denotes the type of event.
Patron/Player	Unique player identifier.
CI (Coin-in) Meter	Counts the currency that is inserted into the machine.
CO (Coin-out) Meter	Counts the currency that is dispensed by the machine.
Games Meter	Counts the number of games played by the machine.

Table 2.1: Raw Data Fields

Machine information files for the casino slot floor were also provided. Relevant fields are detailed in table 2.2. Usually, the "machine number" field is unique, but this is not always the case.

2.2 Data Cleaning

For the purposes of data cleaning and evaluation, nQube Data Science Inc. allowed the use of their event to session data conversion software (session converter).

Events were removed for various reasons. Uncarded events that are not attributed to a carded session by the session converter are not useful for calculating transition matrices because they do not allow for transitions between

Machine Data Column Name	**Description**
Machine Number	A unique machine identifier.
Manufacturer	The manufacturer of the machine.
Game Name	The name of the game as displayed to players.
Hold	The percentage of coin-in that the machine keeps on average.
Cabinet Type	The type of container the game is in.
Bank Number	Numeric identifier for the bank that the machine is on.
Seat Number	The numeric position of the machine on the bank.
Time	The time at which the machine was made active.

Table 2.2: Machine Information Fields.

machines to be observed. There are two event codes that appear to be associated with a card insertion (card-in) event because they are always the first in a sequence of carded events. These two event codes often appear consecutively, causing difficulties when converting to session data. For this reason, these two event codes were consolidated into a single event code.

In the machine information data, some machines appear more than once, but with a different specified location and a different time field, indicating that the machine was moved. After a machine move occurs, players' preference for it may change and so it should be considered to be a different machine as it may not behave in the same way anymore. For this reason an asset number was added to machine information data that is the same as the machine number field, but for each move the asset number has a zero appended to it. Any events that occurred on a machine after a move took place are assigned a new machine number in the event data equal to the asset number for that machine in the appropriate date range in the machine information data.

Meters are not always strictly monotonic in this data set as one would expect. They may not report for a period of time, resulting in a chunk of zeros. They may leap ahead by an unrealistic amount, or they may be reset at times. These types of discrepancies are products of the slot management systems and the machine manufacturer's built-in procedures so it is difficult to speculate on why they occur. All of these discrepancies are fixed programmatically as follows. If a sequence of zeros are observed in one of the meter's columns, the zeros are set to the last non-zero value in that meter's column. If the meter leaps ahead by an unrealistic amount, all subsequent meter values are decreased by the jump amount. That is at least until the meter decreases to within an acceptable range again. If the meter is reset or decreases, then the difference is added to all subsequent meter values. The maximum allowed meter jump was set to $1e8$ or 20%. Meter jumps less than these thresholds were not considered to be anomalous.

An additional, and unusual, limitation of this data set is that the machine information data did not contain the denominations of the machines or any information on what the units of the coin-in and coin-out meters are. Ideally, we would have access to this information so that inferences could be made on player behaviour based on their spending. This was, however, not the goal of this work. The goal was to develop a technique to create a data set that is statistically similar to the provided input data set, but different enough that it can be shared freely for research purposes. We are not necessarily interested in retrieving accurate, aggregate, statistics from the data. For the purposes of this work, it is assumed that all machines have meters in units of cents, but as is evident from a plot of theoretical win per machine per day, this is likely not true for all machines. It is expected that machines have theoretical win per day in the range of \$100 to \$1000 (10). In figure 2.1, we find that the majority of machines have theoretical win in this range if their meter units are cents, while some have theoretical win per day of less than ten dollars under this assumption which is not realistic.

One final note can be made about the figure 2.2. The spikes in the event

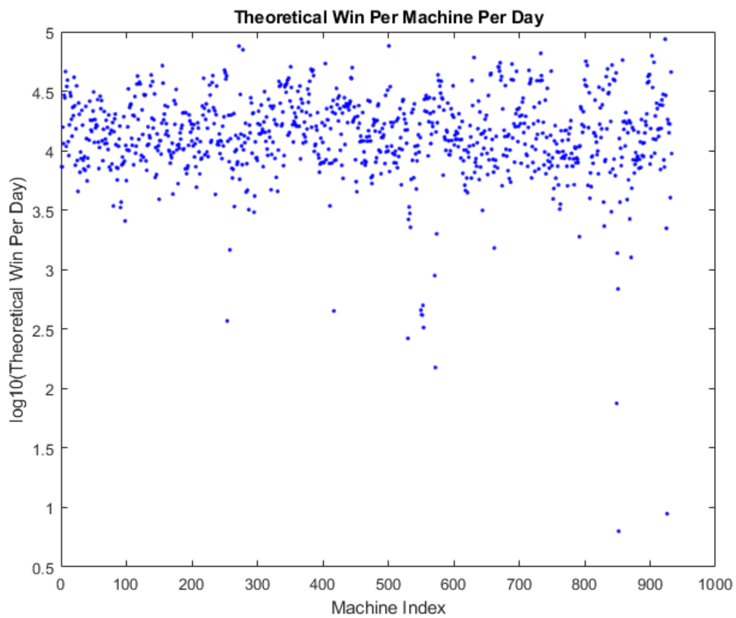

Figure 2.1: The theoretical win (machine hold multiplied by its total coin-in) of each machine divided by the number of days on which at least one session is observed. The units of the y-axis are assumed to be in cents even though this is likely inaccurate for a minority of the machines.

data set are much more pronounced in the event data distribution than in the session data distribution. This demonstrates well the value that conversion to sessions has in reduction of noise by extracting only the relevant information about player activity and discarding the rest. These spikes are likely a product of machine denominations. The largest spikes in the event data plot are at the fifty and one hundred units mark which are a couple of very common machine denominations. This also explains why values surrounding the spikes are so low. This is because they are not common multiples of common machine denominations and so would not be observed as often.

2.2.1 Conversion to Session Data

The conversion of event data to session data is an important step in evaluating the validity of the data. If proper sessions cannot be constructed from the synthetic data that is generated, then the generated event data is not realistic or useful. We must also be able to construct sessions from the real data to compare with the sessions constructed from the generated data.

A session is defined as the period of time between when a player sits down at a machine and when they stand up to go do something else. If this type of structure cannot be found in the generated event data, it would not be very useful for studying human behaviour on the casino slot floor and is thus indicative of a poor synthetic data generation algorithm.

The conversion to session data eliminates any uninteresting information including machine initiated events and aggregates player activity into an easily

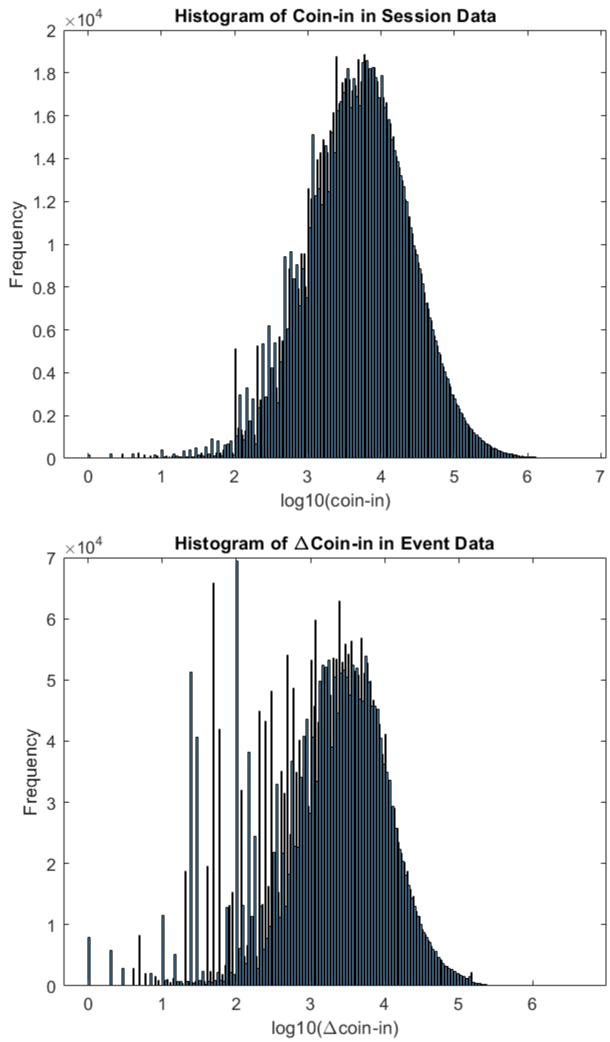

Figure 2.2: The distribution of coin-in in the session and event data. Top: The distribution of coin-in in a session. Bottom: The distribution of the change in the coin-in meter during an event.

digestible format. It also reduces noise by identifying and removing events that are not attributable to player activity such as automatically triggered events or maintenance triggered events. This reduction of noise can be observed in figure 2.2 in how the distribution of coin-in per session appears to be a much smoother distribution than the distribution of the change in coin-in caused by events.

Aside from the ambiguous meter units discussed previously, the major challenge in this process was that there was no legend for event codes provided. The meaning behind the event codes had to be inferred through industry knowledge provided by industry partner nQube Data Science Inc. as well as trial and error. As many codes for player initiated events were decoded as possible and plausible sessions were produced from the provided event data.

First, carded sessions are constructed by iterating through the event data, looking for matching pairs of card-insertion events and card removal events. Payment events that occur within sixty seconds of the card insertion event are included as well as cash-out events that occur within sixty seconds of the card removal event. Next we attempt to identify uncarded sessions. These are more difficult to identify because they do not have a clear beginning and end like the carded sessions do. A sequence of payment and potentially cash-out events in close succession defines an uncarded session. Construction of uncarded sessions will not be discussed in detail because uncarded events cannot be used to construct probability matrices as uncarded player's movement between machines cannot be tracked in their movement between slot machines. For this reason, uncarded activity was not included in validation of the simulation because it was not used to construct the simulation.

It is infeasible to create a perfect conversion so sessions must be checked for invalid properties as they are created. Sessions that have zero games played, unrealistically high rate of play, unrealistic duration, or unrealistically high coin-in are all discarded. Zero coin-in is allowed to account for free-play and zero coin-out is allowed because it is possible that players lose all of the games they play during their session. If coin-in or coin-out are less than zero, this is a major violation and would be indicative of an improperly corrected meter. The parameters governing the rejection of invalid sessions are somewhat arbitrary, but the distributions of coin-in, games played, and session duration can be used as a guide. Long tails on the distributions of these quantities can be used to identify cut-off points.

In theory, the amount of coin-in, coin-out, and games played present in the event data should equal the amount in the session data derived from the same event data. This is not the case in practice however, due to rejection of unrealistic sessions. Meter changes that are not rejected, but are unassigned, are assigned to the most recent, valid session on that machine if there is one because unassigned meter changes are assumed to be the result of delayed meter updates.

A coin-in accounting system was integrated into the converter to keep track of rejected and unassigned meter changes as a way to confirm that the event to session conversion program was working as expected as its complexity made this difficult. According to this accounting system, 0.1% of coin-in is unassigned and could not be attributed to a previous session, 3.03% of coin-in was rejected, and 96.86% of coin-in is attributed to complete and valid sessions. There was a discrepancy of -90 meter units spread out over two machines, meaning that there was a small amount of double counting occurring. A perfect slot accounting system was unrealistic due to the complexity of the conversion and the time allowed for data cleaning. The 0.1% of coin-in that could not be assigned would have occurred at the beginning of the time series where the events signifying the beginning of a session may have been missing. It was only unassignable because there were no previous sessions on that same machine to assign it to. The 3.03% of coin-in that was rejected is higher than is ideal, but this is likely due to some event codes being incorrectly identified. In a more complete data set, this amount would hopefully be lower due the reduced amount of guesswork required.

2.3 Data Preparation

Beyond the data cleaning detailed in the previous section, additional preparation is required for the data to be a suitable input for the synthetic data generation algorithm discussed in the next chapter.

Firstly, a unique event identifier is assigned to each event code and machine number combination, resulting in $N \times E$ event IDs. The event IDs make up the state space of the Markov chain. The assignment is done based on an event

ID lookup table, or in other words an $E \times N$ matrix whose elements are the event IDs. Note that E denotes the number of event codes in the data and N denotes the number of unique machine numbers. Each record in the event data is assigned its appropriate event ID based on the event code and machine number combination in that row.

Secondly, the numeric times that were added to the event data during cleaning are converted to integer-valued seconds relative to the first event in the time series. This is because the times in the data are only precise down to the second and integers are easier to work with as the possibility for rounding error is eliminated.

Next, the session data that was constructed from the event data is examined to label any events that it attributed to a carded player's session that occurred prior to the card-in event or after the card-out event with the corresponding player's identification number. This uses the information gathered by the session converter to increase the number of events used to construct the transition probability matrices, thereby increasing the fidelity of the simulation by including some event types that often do not have a player ID number associated with them. Finally, a field for the change in each meter caused by each event is added to the data as this is the quantity of interest when building the probability distributions for coin-in, coin-out, and games played given the current and next event type.

Chapter 3

Probability Matrix Estimation

3.1 Uniform Occupancy Method

The *Uniform Occupancy* method of probability matrix estimation makes use of an occupancy matrix with elements denoted by q_{ij} equal to the probability that state j is unoccupied given that state i is occupied. This probability is assumed to remain constant throughout the data set. In a real data scenario, sessions would be generated from the event data and they would be used to calculate these probabilities. However, in a test scenario where the actual time that events occur is abstracted away and we only care about the order of events, a logical occupancy array is kept, denoting which machines are occupied at each time step and this is used to calculate the conditional probabilities that make up the occupancy matrix.

Equation 3.1 gives the equation for calculating the unnormalized probability matrix elements. In equation 3.1, N_{ij} is the number of observed transitions from state i to state j, $N_i = \sum_j N_{ij}$, and $q_{ij} = P(j \; unocc \mid i \; occ)$ is an element of the pre-calculated occupancy matrix.

In equation 3.1, two quantities have an effect on the probability of a transition: N_{ij}/N_i acts to increase probability when the transition from state i to state j occurs often, and q_{ij} acts to weight the probability of transitioning from i to j more heavily when j is often occupied. The effect that q_{ij} has is present because if state j is often occupied, it will not be transitioned to very often despite its popularity, so each transition that does occur should have more weight.

$$t_{ij} = \frac{N_{ij}}{N_i q_{ij}} \tag{3.1}$$

A parallel can be drawn between this equation and the equation of conditional probability given by equation 3.2. Letting $p(A)$ be the probability of a transition from state i to state j, and $p(B)$ be the probability that the machine corresponding to state j is unoccupied, then we get the relationships given by equations in 3.3.

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \tag{3.2}$$

$$
\begin{aligned}
p(A|B) = t_{ij} &= \frac{N_{ij}}{N_i q_{ij}} \\
p(A \cap B) &= \frac{N_{ij}}{N_i} \\
p(B) &= q_{ij}
\end{aligned}
\tag{3.3}
$$

These parallels do not work however, as t_{ij} is not a true probability as the rows do not sum up to one and have to be renormalized.

3.2 Probability Update Method

The *Probability Update* method of probability matrix estimation makes no assumptions about the occupancy ratio. The result of the probability update method is a probability matrix containing the probability that a player will choose to make a certain transition given that that transition is available. Its aim is to remove occupancy from the equation entirely, allowing us to learn about the choices that players make.

With the uniform occupancy method, observing a transition between states i and j acts to reduce the probability of a transition between state i and any other state $k \neq j$ when we have no way of knowing that the player would not have chosen an occupied state over state j if they could have. The probability update method says that the probability of a transition from state i to any occupied state k should remain constant and only the probability of transitioning from state i to a currently unoccupied state should be reduced while the probability of moving from state i to the destination state j is increased.

3.2.1 Counts Formulation

Derivation of the probability update method begins by assuming that an observed transition between states i and j should increase the total count of the i^{th} row by exactly one. With the MLE (maximum likelihood estimator), that increase is given entirely to t_{ij} acting to decrease the probability of transitioning to states the player had no option of transitioning to. See equation set 3.4 for the set of equations to be solved subject to the constraints in equation 3.5. In this derivation, A is a set of unoccupied states at the time of the transition, the i subscript denotes the destination state of the transition, the j subscript denotes any state for which $j \in A$, and the k subscript denotes any state for which $k \notin A$. The origin state is assumed and all quantities are for the origin state's corresponding row in the probability matrix.

$$\begin{aligned} N_i &= N_i^{(0)} + \delta \\ N_j &= N_j^{(0)}(1+\alpha) \\ N_k &= N_k^{(0)}(1+\beta) \end{aligned} \tag{3.4}$$

$$
\begin{aligned}
N = N_i + \sum_j N_j + \sum_k N_k = N_i^{(0)} + \sum_j N_j^{(0)} + \sum_k N_k^{(0)} + 1 \\
\frac{N_k}{N} = \frac{N_k^{(0)}}{N^{(0)}} \ \forall\ k \notin A
\end{aligned}
\tag{3.5}
$$

With the formulation given in equations 3.4 and 3.5, we have three variables and two constraints, allowing for a single degree of freedom. See equation 3.6 for the solution. One obvious choice for α is zero, but the degree of freedom it allows gives the possibility for fine-tuning of the method.

$$
\begin{aligned}
N_i = N_i^{(0)} + 1 - \alpha \sum N_j - \frac{\sum N_k}{N^{(0)}} \\
N_j = N_j^{(0)}(1 + \alpha) \\
N_k = N_k^{(0)}(1 + \frac{1}{N^{(0)}})
\end{aligned}
\tag{3.6}
$$

3.2.2 Probabilities Formulation

There is an alternative formulation of this method that yields normalized probabilities rather than counts that require normalization after the fact. The equations to be solved in this case are given in equation 3.7 that are solved subject to the single constraint given in equation 3.8. It is expected that with each observation, p_i will increase, p_j will decrease, while p_k is required to remain constant.

$$\begin{aligned} p_i &= p_i^{(0)} + \epsilon \\ p_j &= p_j^{(0)}(1+q) \\ p_k &= p_k^{(0)} \end{aligned} \tag{3.7}$$

$$p_i + \sum p_j + \sum p_k = p_i^{(0)} + \sum p_j^{(0)} + \sum p_k^{(0)} = 1 \tag{3.8}$$

The solution to equations 3.7 and 3.8 is given in 3.9.

$$\begin{aligned} p_i &= p_i^{(0)} + \epsilon \\ p_j &= p_j^{(0)}(1 - \frac{\epsilon}{\sum p_j^{(0)}}) \\ p_k &= p_k^{(0)} \end{aligned} \tag{3.9}$$

This formulation also results in a single degree of freedom, allowing for fine tuning of the estimation.

Finally, it is possible to obtain a relationship between α and ϵ so we can easily convert one of the formulations into the other. This relationship is given in equation 3.10.

$$\epsilon = \sum_j p_j^{(0)} \frac{(1 - \alpha N^{(0)})}{(N^{(0)} + 1)} \tag{3.10}$$

3.2.3 Classifying Parameters

The optimal ϵ (or α) must be determined. It must be a function of the current state of the probability matrix, as a static constant ϵ would inhibit convergence. It can be thought of as the importance of a new data point, which should diminish as the number of observations increases. This property is exhibited by equation 3.10 where we find the number of observations in the denominator.

Bounds

The equations for p_i and p_j found in equation 3.9 along with the fact that $\epsilon \geq 0$ by definition leads to bounds on ϵ given in equation 3.11. At the lower bound of $\epsilon = 0$, no learning takes place. At the upper bound of $\epsilon = \sum p_j^{(0)}$, every p_j is erased, giving an equally unuseful result.

By employing equation 3.10, we can also find bounds on α. The resulting bounds are given in equation 3.12. At the lower bound of $\alpha = -1$, every N_j is erased. The upper bound of $\alpha = 1/N^{(0)}$ corresponds to $\epsilon = 0$ at which the probabilities remain constant.

$$0 \leq \epsilon \leq \sum p_j^{(0)} \tag{3.11}$$

$$-1 \leq \alpha \leq \frac{1}{N^{(0)}} \tag{3.12}$$

Optimizing Epsilon

We can learn about the optimal value of ϵ through equation 3.10. By allowing α to act as a parameter to ϵ and varying it between its bounds found in equation 3.12 we have a way of varying epsilon while still enforcing its necessary properties.

Tests were run by first initializing a random transition matrix, then using it to generate a simple data set. The scenario used contained five machines, two players and 10^8 iterations. The small number of machines allowed for a quick computation time and the small number of players meant that high occupancy would not affect the results. The high number of iterations ensured that stabilization of the error was captured if it was possible to achieve. The generated data set was then used to construct a transition matrix as we would do with the real data. Each data point was used to update the transition matrix according to the Probability Update equations 3.9. At each iteration of the calculated transition matrix, the mean absolute error between its elements and the initial transition matrix's elements was recorded. This was done for a variety of ϵ parameters with parameter α being varied between its bounds.

The results for a good α would quickly converge to small mean absolute error indicating that the least amount of data is needed to approximate the true distribution of event transitions.

In figure 3.1, we find that the mean absolute error oscillates within bands of error, never converging, for $\alpha = -1$. The four bands of error are caused by the unoccupied probabilities always being erased with every transition, eventually causing each row of the calculated probability matrix to become all zero probabilities except for one column where the most recent transition took place being equal to one. This causes a finite amount (5^5 to be exact) number of probability matrices that can be calculated at each iteration, mostly filled with zeros. The mean errors between these 5^5 possibilities and the initial transition matrix that was used to arrive at figure 3.1 were calculated, and their distribution is plotted in figure 3.2. The bands present in the histogram of figure 3.2 line up perfectly with the bands in figure 3.1.

At other negative values of α, we find that the mean absolute error oscillates within a single band shown in figure 3.3. On a logarithmic scale, the band appears to widen as more data is gained demonstrating that a model with $\alpha < 0$ may actually be unstable rather than just non-convergent. The best convergence is found for $\alpha = 0$ as seen in figure 3.4. The mean absolute error quickly converges to near zero with no oscillations. The model behaves in a stable manner, approximating the true probability matrix with great fidelity. On the logarithmic scale, it can be seen that the mean absolute error has still not yet tapered off at $1e8$ data points and continues to decrease with more data. Similar behaviour is found for all positive values of α, however the mean absolute error does not reach as low of values as it does with $\alpha = 0$. An example of this is given in figure 3.5. Note that when α is positive, the value indicated is the coefficient of α which, when positive, is proportional to one over the number of observations.

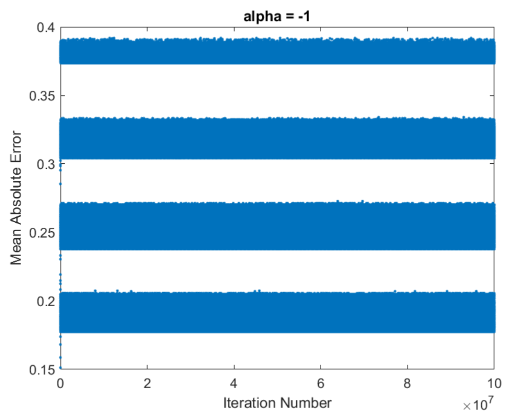

Figure 3.1: The mean absolute error as a function of the number of data points on a linear scale with $\alpha = -1$.

Figure 3.6 summarizes this information with side-by-side boxplots of the error recorded at each iteration from each experiment. For negative α, the median mean absolute error decreases as α increases, reaching a minimum at $\alpha = 0$. For non-negative α, the inter-quartile range is so narrow that it is difficult to see, demonstrating that the vast majority of the data is near zero. The median steadily increases as positive α increases.

3.2.4 Special Cases

MLE

The maximum likelihood estimator for a transition matrix from data is given by $\frac{N_{ij}}{N_i}$ ((20)) where N_{ij} is the number of moves that occur in the data set from state i to state j and $N_i = \sum_j N_{ij}$. For our type of data set, where collisions

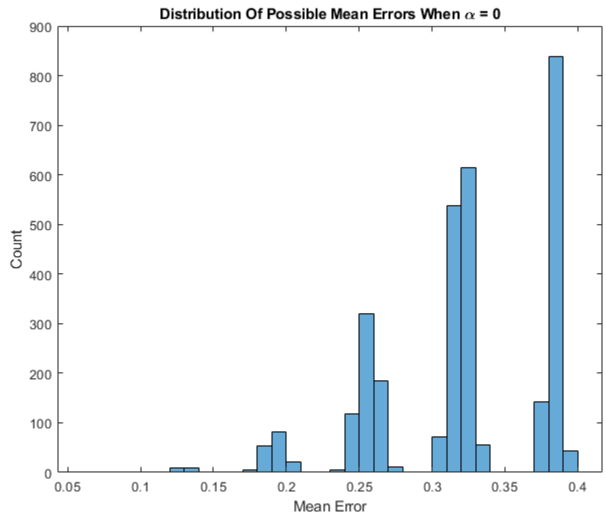

Figure 3.2: The distribution of all possible mean errors when $\alpha = -1$.

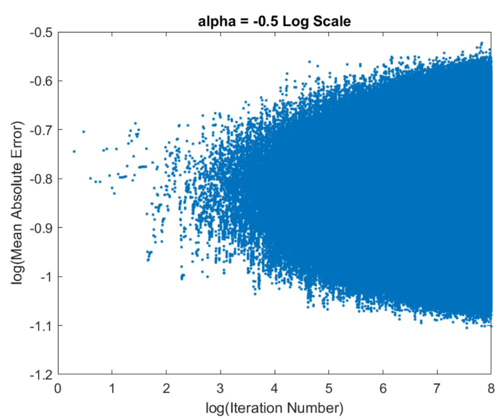

Figure 3.3: The mean absolute error as a function of the number of data points on a logarithmic scale with $\alpha = -0.5$.

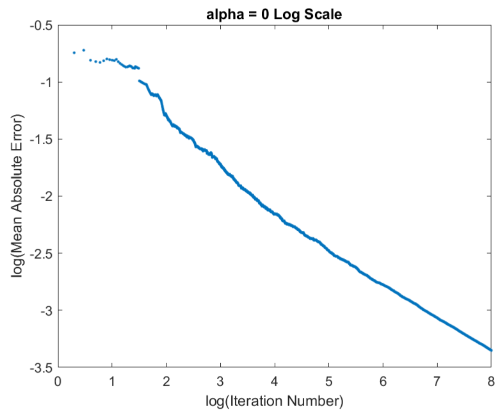

Figure 3.4: The mean absolute error as a function of the number of data points on a logarithmic scale with $\alpha = 0$.

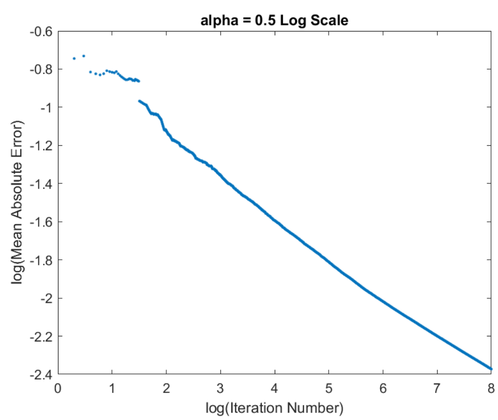

Figure 3.5: The mean absolute error as a function of the number of data points on a logarithmic scale with $\alpha = 0.5$.

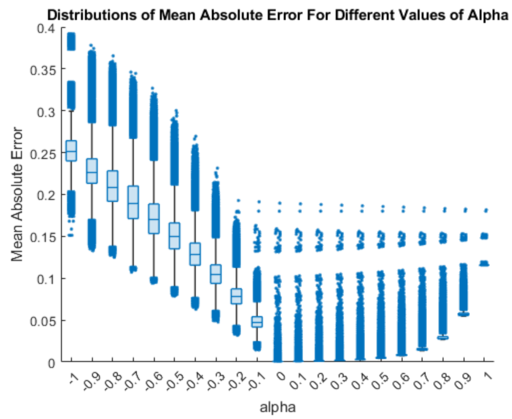

Figure 3.6: A collection of box plots comparing the convergence properties of all tested values of α.

are possible and must be corrected for, the MLE is not very useful except in the case that there is only one player in the casino. If there is only one player in the casino, then $N_k = 0 \; \forall \; k$ and $p_k = 0 \; \forall \; k$ because there will be no occupied machines.

In the counts formulation, we find that $\alpha = 0$ for this case and in the probabilities formulation, we find that $\epsilon = \frac{1-p_i^{(0)}}{N^{(0)}+1}$ where $N^{(0)} = N_i^{(0)} + \sum N_j^{(0)}$ is the number of transitions that have been recorded from the origin state thus far.

Uniform Occupancy

It is not possible to find an equation for ϵ that would produce a probability matrix equal to the one obtained from the *Uniform Occupancy* method. This is because the *Uniform Occupancy* method violates the assumption that each observation increases the count of the row by exactly one and the assumption that the probability of transitioning to an occupied state remains constant. Additionally, the probability matrix obtained from the *Uniform Occupancy* method can only be computed after data is accounted for since the occupancy is assumed to be uniform throughout.

Add-x

An add-x estimator is an estimator where some amount x is always added to the number of observations followed by a re-normalization so that are no zero probabilities. This is a common practice when predicting probabilities from a data sample whose scope is unlikely to encompass the entire probability space (29), (30). Laplace smoothing is the most common type of add-x smoothing where $x = 1$. Add-x estimators are sometimes called additive smoothing, whose applications and value are demonstrated in (30).

In (22), add-β and add-$\sqrt{n_i/k}$ estimators are introduced where β is some constant, n_i is the number of transitions from state i, and k denotes the number of states.

The add-$\sqrt{n_i/k}$ can be thought of as a special case of the *Probability Update* method. To show this, it is simplest to use the counts formulation of the *Probability Update* method because it is the count of observations being modified. For this estimator, we find that $\alpha = \frac{1-N_i-\sqrt{N_i}/k}{\sum N_j}$ where k is the number of states. The add-$\sqrt{n_i/k}$ will be used as a benchmark in the evaluation of the Probability Update and Uniform Occupancy methods.

3.2.5 Algorithmic Considerations

The *Probability Update* method of estimation produces nonzero probability for transitions that were never observed in the data because it assume that every transition is equally likely to start. It is necessary with this method for the rows of each matrix to start out as a valid probability distribution. The problem with this is that it allows transitions with very small probability for which there is no coin-in, coin-out, games played, or time interval data available to draw from so these transitions must be disallowed. See table 3.1 for the consequences of this. The method with which these transitions are disallowed is discussed in the next section.

3.3 Challenges

3.3.1 Sacrificing Speed For Memory

For large data sets with many event IDs, full probability matrices cannot be held in memory on computers with conventional amounts of RAM (32-64 GB) because over 100 GB would be required. Since each transition is initially set to be equally likely in the Probability Update method, the matrices are not sparse as they were in the Uniform Occupancy method. For this reason, modifications must be made to how probabilities are initialized. This was handled by only holding in memory elements of relevance. The probability of every element was assumed to be zero until it was observed. The initial probability was taken into account at that point. Any element that remained zero at the end was never observed and should therefore have a probability of zero. Rows were re-normalized at this point. This technique limited speed of computation as most computation had to be done on the fly so that as little data as possible was being stored in memory.

Additionally, parallelization was a necessity for the probability matrix estimation. Without it, the computation may take several days, up to a week or more, on a high-end personal computer. The computer used to run the simulations reported on here has 10 core, 3.7 GHz processor with 64 GB of RAM installed. This parallelization was done differently for the Probability Update method than it was for the Uniform Occupancy method due to the different initialization procedures of each method.

In the Uniform Occupancy method, we are able to assume that each probability is zero until a transition is observed and so the matrices are initially sparse and easily contained in memory. For this reason, the Uniform Occupancy probability matrix computation is split up by player ID. Each sequence of events made by a single player is an independent sample of transitions. No transitions are missed by splitting up the workload in this manner as they would be if the workload was split up by time for example. Each process is assigned an approximately equal number of player IDs and counts the number of each transition those players make. These counts are easily added together once each process completes to form a complete probability matrix.

For the Probability Update method, the initial probabilities are not zero and so the amount of information contained in each process needs to be minimized further. For this reason, the Probability Update matrix computation was split up by event ID. This way each process held a matrix in memory that was zero for every transition in a row corresponding to an assigned event ID. To ensure even distribution of the computational load over a number of processes, event IDs were sorted by their prevalence and assigned to processes in a turn-based manner. Process 1 would receive the least prevalent event ID, process 2 would receive the second-least prevalent and so on until the n^{th} process would receive the n^{th} least-prevalent event ID and then it would be process 1's turn again. This assignment procedure combined with the decision to split up the work load by event ID cut computation time down to about a single day.

3.3.2 Cycles

A weakness of using a first-order Markov process to generate data is that the transition matrices are not acyclic. It is possible for the process to enter a cycle that cannot be escaped resulting in a sequence of events repeating over and over again until the simulation completes. This would not result in realistic data so a solution was needed. The communicating classes (18), (19) of a transition matrix are the groupings of states which can all be reached from every other

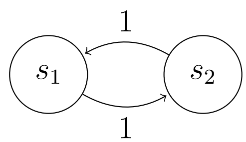

Figure 3.7: An example of a small cycle that is simple to detect without the use of communicating classes. A state within the cycle is transitioned to from another state outside the cycle. The cycle is inescapable.

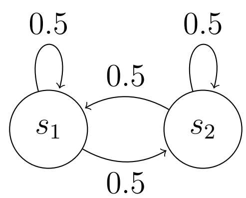

Figure 3.8: An example of a small cycle that is difficult to detect without the use of communicating classes. A state within the cycle is transitioned to from another state outside the cycle from which the cycle is inescapable.

state in the group. By determining which communicating classes are closed, i.e. inescapable, it is possible to know when an inescapable cycle has been entered. By checking at each transition whether the state transitioned to belongs in a closed cycle, the cycle can be cut short. The data generation code allows a player to remain within a cycle for as many transitions as there are states in the cycle. See figures 3.7 and 3.8 for examples of inescapable cycles.

3.4 Bayesian Formulation

Equation 3.13 gives the likelihood function for a set of transitions occurring from a given origin state. Each destination state i_m occurs k_m times in the set of transitions. This is similar to a multinomial distribution as in (20). The main problem with a Bayesian formulation is that ϵ is not necessarily constant with time, but is a function of the occupied and unoccupied state probabilities at the time of the transition. In the absence of data, all probabilities are assumed to be equal, so a uniform prior is a sensible choice.

$$P(X_t = i | p_1, ..., p_M, \epsilon) = p_i + \epsilon$$

$$P(\mathbf{X} = i_1, i_2, \ldots, i_T | p_1, ..., p_M, \epsilon) = (p_1 + \epsilon) \ldots (p_1 + k_1\epsilon)(p_2 + \epsilon) \ldots (p_2 + k_2\epsilon) \\ \ldots (p_M + \epsilon) \ldots (p_M + k_m\epsilon) \tag{3.13}$$

More naively, a true multinomial distribution (equation 3.14) could be used as the likelihood. This results in simpler to solve equations since the normalization constant is known.

$$P(\mathbf{X} = i_1, i_2, \ldots, i_T | p_1, ..., p_M, \epsilon) = (p_1 + \epsilon)^{k_1} (p_2 + \epsilon)^{k_2} \ldots (p_M + \epsilon)^{k_m} \tag{3.14}$$

With the multinomial likelihood, it is simple to use Lagrange multipliers to solve for p_i (equation 3.15). With $\epsilon = 0$, we have the MLE.

$$p_{ij} = \frac{n_{ij}(1 + \epsilon)}{n_i} - \epsilon \tag{3.15}$$

A true and accurate likelihood function proved to be too unwieldy in practice. Additionally, ϵ is not a parameter with a maximally likely value, but a function of the current probabilities and the number of observations.

3.5 Results

3.5.1 Performance

The performance of the probability matrix estimation method is measured by performing tests where an initial probability matrix is randomly generated. The size of this initial probability matrix varied with the parameters of each test performed, but for every test it was chosen for each machine to have only one event ID. This way, the probability matrix takes on a diagonal structure rather than a block-diagonal structure and the testing procedure is simplified greatly so that the focus is on the estimation method. The player is always more likely to stay on their current machine than move to another, so the probabilities in the initial matrix were heavily biased on the diagonal.

The initial probability matrix is used to generate a simulated data set and then from this data set, each estimation method is used to estimate the transition matrix that was used to generate that data. Five trials are performed for each combination of the number of players and number of iteration tested to eliminate randomness effects.

When the number of players is varied, the effect of occupancy on the estimation is measured, and when the number of iterations is varied, the effect of the size of the data set is measured. The error E for each of the five tests and for each combination of parameters is then calculated according to equation 3.16 where p_{ij} denotes an estimated probability matrix element, $p_{ij}^{(0)}$ denotes a true probability matrix element, N denotes the size of a probability matrix row or the number of machines in the test, and $t = 1, \dots, T$ where T is the trials performed for a combination of parameters. The error for each of the $T = 5$ trials is averaged to eliminate randomness effects. The results are plotted in figures 3.9.

$$E = \frac{\sum_{i,j,t} \left| p_{ij}^{(t)} - p_{ij}^{(0)} \right|}{TN^2} \tag{3.16}$$

In figures 3.9 and 3.10, there are four labels:

1. MLE: refers to the maximum likelihood estimator $\frac{n_{ij}}{n_i}$.
2. Prob Update: refers to the Probability Update method.
3. Unif Occ: refers to the Uniform Occupancy method.
4. Add-x: refers to the add-$\sqrt{n_i}/k$ estimator from (22).

From the plots in figure 3.9, the performance of the MLE method is adversely affected by increasing occupancy. This is expected because the MLE estimator is what the proposed estimators reduce to in the absence of other players. The Uniform Occupancy method has undefined error for one hundred percent occupancy as this results in a division by zero. The performance of the MLE and Add-x methods is unaffected by an increased sample size. The performance of the Add-x method is also adversely affected by increasing occupancy. At

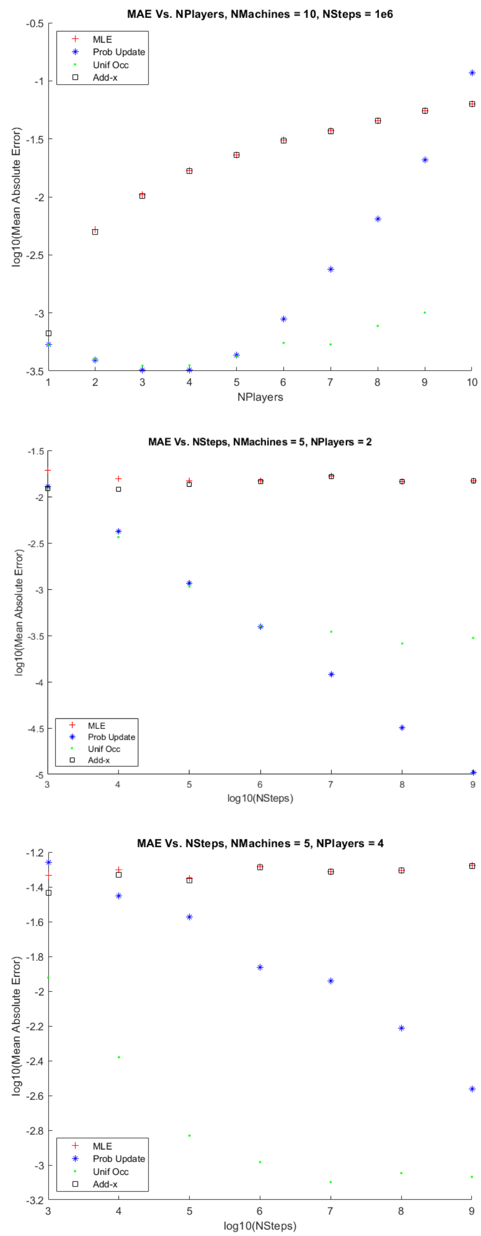

Figure 3.9: The base-10 logarithm of the mean absolute error between the initial probability matrix and the resultant probability matrix in three different scenarios. In the Probability Update method, $\alpha = 0$. Top: The mean absolute error as a function of the number of players. Middle: The mean absolute error as a function of the number of iterations at high occupancy (80%). Bottom: The mean absolute error as a function on the number of iterations at low occupancy (40%).

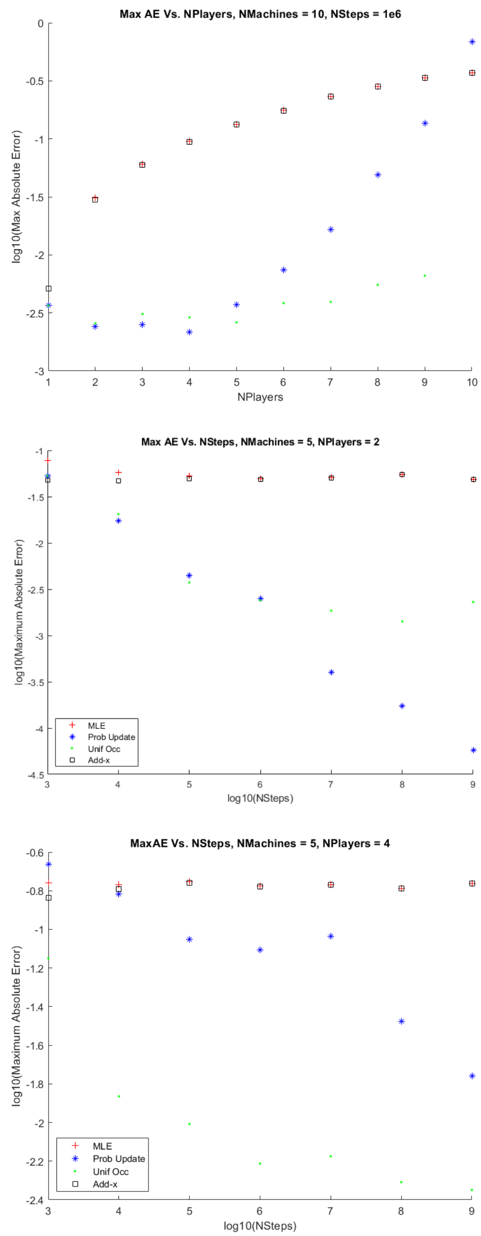

Figure 3.10: The base-10 logarithm of the maximum absolute error between the initial probability matrix and the resultant probability matrix in three different scenarios. In the Probability Update method, $\alpha = 0$. Top: The mean absolute error as a function of the number of players. Middle: The mean absolute error as a function of the number of iterations at high occupancy (80%). Bottom: The mean absolute error as a function on the number of iterations at low occupancy (40%).

occupancy ratio greater than sixty percent, the Probability Update method results are also adversely affected by increasing occupancy, but not for the same reason as the MLE and Add-x methods.

The MLE and Add-x methods are adversely affected by increasing occupancy greater than ten percent because they are not designed to estimate probability matrices where individuals compete to occupy states. The Probability Update method performs well up to about fifty percent, and satisfactorily up to about seventy percent. Its performance suffers at higher occupancy levels because there is an absence of choice for players. If many states are occupied, probabilities mostly remain constant at each iteration and the learning rate suffers. The learning rate in this scenario refers to the reduction in error per iteration or data point.

To further observe the learning rate of different methods, we look to the plots of the error as a function of the amount of data (NSteps) in figure 3.9. In both the high occupancy and low occupancy scenarios, the Uniform Occupancy method essentially ceases learning around 1e7 steps while in a low occupancy scenario, the Probability Update method continuously keeps up its learning rate with no sudden drop-off. In a high occupancy scenario, the Probability Update method also keeps up a lower rate of learning, not converging as closely to the true probability matrix as the Uniform Occupancy method does in a much fewer number of iterations. For continuously high occupancy data, the Probability Update method requires much more data to model it accurately.

Figure 3.10 shows the maximum absolute error for all scenarios studied to show that the mean absolute error is a good metric for the performance of estimation methods. The maximums have a similar functional structure to the means and are not much different from them, indicating that the mean is not being hugely affected by large outliers and all probability matrix elements are being estimated well.

When estimating a probability matrix from real data, the occupancy of the casino will be highly variable. The low occupancy time periods should make up for the high occupancy time periods, resulting in a highly accurate probability matrix for any data set that is diverse enough in its occupancy levels.

From figure 3.9, it would seem that the Uniform Occupancy method is superior to the Probability Update method for data with a constantly high occupancy ratio. If the data contains data points that took place during a low occupancy time period, then the Probability Update method will better approximate the true probability matrix.

3.5.2 Matrix Properties

Table 3.1 gives the number of non-zero values in the probability matrices. The *Probability Update* matrices will contain non-zero probabilities for transitions that were never observed in the data because each data point affects probabilities corresponding to unoccupied machines as well. Because these transitions have not been observed there will be no coin-in, coin-out, games played or time

	Unif Occ	Prob Update (Before Removal)	Prob Update (After Removal)
Card In	121 529 (0.001%)	130 355 (0.001%)	121 529 (0.001%)
Card Out	412 774 (0.004%)	8 792 056 (0.009%)	413 407 (0.004%)

Table 3.1: Number of Non-Zero Values in Probability Matrices

interval values to draw from so the probabilities for these transitions must be zeroed out. Table 3.1 shows the number of non-zero probabilities before and after these transitions are zeroed out.

The number of non-zero entries in the card-out *Probability Update* matrix after removal is greater than the number of non-zero entries in the card-in *Uniform Occupancy* matrix. It is likely that some intra-machine transitions that are only ever seen when a card is inserted were captured by the assumption that all transitions start out as equally likely.

The card-in matrix takes on a block-diagonal structure since only transitions between event IDs on the same machine are allowed. The blocks are square with dimensions equal to the number of event codes. The card-out matrix has a less identifiable structure with the majority of entries being off the diagonal.

Chapter 4

Synthetic Data Generation

4.1 Building Probability Matrices and Distributions

There is a distinction between probability matrices and the transitions matrices that must be made. The probability matrices are what is created from the data, but do not contain actual probabilities of transitions. Actual probabilities of transitions vary with the occupancy of the slot floor during simulation. The transition matrix at a given moment is equal to the probability matrix with occupied transitions zeroed out and remaining probabilities renormalized. What is described in this section are the probability matrices that give the probability of a transition given that all transitions are available to be made.

Two separate card-in and card-out matrices are required to ensure that card-in and card-out events always occur in an alternating order. This is because there are some event IDs that can occur regardless of whether a player card is inserted or not (cash-in or cash-out type events) and this fact combined with the fact that the Markov-like process only has a "memory" of one time step, makes it possible for card-in and card-out events to occur out of order if there were only one probability matrix.

To build the transition matrices, the event data is grouped by patron ID number and then ordered by the time that the events occurred. For every patron ID, the events initiated by that player are looped through one at a time and observations are recorded in the appropriate probability matrix according to the prescribed method of calculating probabilities. If the current player's card is currently inserted, the card-in matrix is updated and if the current player's card is not currently inserted, the card-out matrix is updated. For each observation, the coin-in, coin-out, games played, and time elapsed since the current player's previous transition are all recorded in the array corresponding to the transition made. Each unique transition observed has arrays for each of these quantities that hold each value of the quantity that was observed. This creates distributions that can be drawn from in a Monte Carlo manner illustrated by equation 4.1. In equation 4.1, Y_{ij} is a random variable representing the value of a given quantity (coin-in, coin-out, games played, or time elapsed) for the transition between event ID i and event ID j. The expression N_{ij} denotes the number of times a transition from event ID i

to event ID j is observed in the data.

$$P(Y_{ij} = y | X_n = j \ \& \ X_{n-1} = i) = \frac{\sum_j Y_{ij} = y}{N_{ij}} \tag{4.1}$$

When generating data, players are chosen according to a distribution derived from the number of events that are triggered by each player in the data. An array of counts is created where each element corresponds to the number of events triggered by a specified player, that when normalized, gives the probability that a player will be chosen during data generation.

When a new player is chosen during data generation, the machine at which they begin their visit is chosen from a first-machines distribution. In a manner similar to that of the players distribution, an array of counts is kept for the machines in which a machines count is increased if it is the first machine that a player sits down at during their visit in the real data. When normalized, this array gives the probability that a new player will begin their visit at any given machine during a simulation.

In the current state of the simulation, the first-machines distribution is the limiting factor of a simulation. This means that in a real data set, you can only observe so many machines as being the first that a player sits down at, so once all of these "first machines" are occupied, no more players will be able to join the simulation until one of the players that were there first moves onto another machine or leaves. In practice, the number of players in the simulation cannot be more than the number of machines that have a non-zero probability in the first-machines distribution or else player deadlock may occur. This is a potentially fixable problem, but this was not a priority of this work as simulating more players than the number of "first-machines" observed is currently computationally infeasible.

4.2 Data Structures

Due to the large volume of data used to make decisions in the simulation, significant consideration had to be made to minimize the amount of memory used while sacrificing speed of computation as little as possible.

Probability matrices are stored using MATLAB sparse matrices, which only store non-zero values that can be accessed in $\mathcal{O}(1)$ time just like full storage matrices. The only drawback is that some speed must be sacrificed when normalizing because doing so in a quick vectorized manner produces a non-sparse matrix too big to fit in memory, usually over 100 GB. As a result, for loops are employed for normalizing which are much slower.

To store the accompanying quantity arrays (coin-in, coin-out, games played, and time elapsed), a similar structure is used. Arrays are stored in a list accompanied by an array containing the initial event ID for each entry and another array containing the destination event ID for each entry. Constant lookup time is achieved by creating a sparse matrix that is used as a dictionary for the arrays. Entries in this sparse matrix give the index of that transition's arrays in the list.

4.3 Data Generation

The process is initialized by selecting the user-defined number of players from the pool of available players, selecting a machine for each of them to start at from the first-machines distribution, and then inserting a card-in event for each player on the chosen machine for them into the data.

Next, the process loops over players for a user-defined number of iterations. For each player, the row of the transition matrix corresponding to their most recent state is modified by the current occupancy of the machines to obtain the probability of each possible transition the player could make. A transition is chosen by generating a random number according to a uniform distribution. If there are no possible transitions for the player, the player leaves and a new player is chosen to replace them according the players distribution. This new player is initialized in the same way as all the starting players were. If a transition is chosen for the player, a new data point is inserted into the data with accompanying quantities of coin-in, coin-out, games played, and time elapsed chosen from the distributions for that transition. The player may also leave and be replaced if the time chosen is greater than a user-defined time-out. The simulations done as part of this work used a time-out of two hours. This allows for the simulation to account for players leaving for a large variety of reasons not limited to the machines that they want to play being occupied.

Chapter 5

Results

Two data sets were generated using a probability matrix created by each of the four probability matrix estimation methods. One generated data set was generated with ten players in the casino at all times ($J = 10$) and the other was generated with one hundred players in the casino at all times ($J = 100$). This was to compare how the simulation behaved at different levels of occupancy. The number of machines available to the players was not limited by the simulation. Every machine that was observed in the real data set was available to players in the simulation.

The numbers of players that were simulated ($J = 10, 100$) are unrealistically low. Over 14 000 players were observed in the real data set. The current state of the simulation is severely limited computationally in that even only one hundred players takes weeks to simulate only a month's worth of data.

5.1 Fitting Distributions

The session duration, coin-in per session, and games per minute distributions found in session data have a recognizable potentially Gaussian structure and so fitting Gaussian and Skew Normal distributions to them was attempted to determine if some session-level quantities can be modeled using parametric models. The residuals between the distribution fitted to the real distribution and the generated distributions could also be used to determine how faithfully the generated data models the real data for some quantities. The distributions that were fitted include the coin-in per session distribution, the session duration distribution, and the games per minute distribution. Most event data distributions have more unique shapes that are not similar enough to any known parametric distribution and so this test of similarity could not be performed on the event data. The coin-in per event distribution appears to possibly have a Gaussian structure, except for the left side where we find large spikes at multiples of common currency values. For this reason, a Gaussian or Skew Normal distribution would not capture the features of the coin-in per event distribution well and it is therefore not useful to try to fit it. These spikes are lessened significantly in the coin-in per session distribution making it a better candidate for parametric modelling.

The fact that the event data distributions have less recognizable structures demonstrates reasoning for using an MCMC process to simulate it rather than a simpler parametric statistical model. The rest of the session data distributions

Table 5.1: Results of fitting a Gaussian distribution to session duration, coin-in per session, and games per minute in the real data.

Plot Type	Amplitude	σ	μ	Residuals
Session Duration	0.016	0.510	-2.327	0.052
Coin-in Per Session	0.012	0.649	3.602	0.091
Games Per Minute	0.032	0.116	1.076	0.219

Table 5.2: Results of fitting a Skew Normal distribution to session duration, coin-in per session, and games per minute in the real data.

Plot Type	Amplitude	ξ	ω	α	Residuals
Session Duration	0.020	-2.308	0.510	-0.046	0.052
Coin-in Per Session	0.020	3.343	0.702	0.534	0.089
Games Per Minute	0.009	1.198	0.197	-3.325	0.073

do not have such a recognizable structure, making it difficult to simulate complete session data with a parametric statistical model with high fidelity.

The probability density function for the normal distribution and skew normal distribution are found in 5.1 and 5.2 respectively.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{5.1}$$

$$f(x) = \frac{1}{\omega\sqrt{2\pi}} e^{-\frac{(x-\epsilon)^2}{2\omega^2}} \{1 + erf\left(\frac{\alpha(x-\epsilon)}{\sqrt{2}\omega}\right)\} \tag{5.2}$$

Results of fitting distributions to the real data are found in tables 5.1 and 5.2. For the session duration and coin-in distributions, the residuals are not improved much by fitting a Skew Normal distribution instead of a Normal distribution, indicating that these distributions are nearly normal.

Table 5.3: Comparing the residuals between the distributions in generated data and the distributions fitted to the real data. Here, the distribution fitted to real data was compared to the distribution found in the generated data.

Plot Type	Method Name	Gaussian Residuals	Skew Normal Residuals
Session Duration	Real	0.052	0.052
Session Duration	Add-x	0.268	0.268
Session Duration	MLE	0.266	0.266
Session Duration	Prob Update	0.179	0.179
Session Duration	Unif Occ	0.269	0.269
Coin-in Per Session	Real	0.091	0.089
Coin-in Per Session	Add-x	0.285	0.285
Coin-in Per Session	MLE	0.282	0.282
Coin-in Per Session	Prob Update	0.194	0.194
Coin-in Per Session	Unif Occ	0.282	0.282
Games Per Minute	Real	0.219	0.073
Games Per Minute	Add-x	1.072	1.048
Games Per Minute	MLE	1.068	1.046
Games Per Minute	Prob Update	1.109	1.085
Games Per Minute	Unif Occ	1.071	1.049

This is because the Gaussian distribution is a special case of the Skew Normal distribution, and so if the Gaussian fits the data just as well as the Skew Normal distribution does, the data can be said to be normal. The games per minute distribution is better fitted with a Skew Normal distribution as evidenced by its residuals. The drawback to fitting the games per minute distribution is that the spike at low games per minute caused by video poker games is not captured with a Normal or Skew Normal distribution.

The residuals between the generated data distributions and the distributions fitted to the real data are shown in table 5.3. The Gaussian and Skew Normal residuals are very similar for the session duration and coin-in per session distributions since these distributions in real data were found to be fitted well with a normal distribution. For the session duration and coin-in per session distribution, the residuals on the data generated with the Probability Update method are found to be significantly less than those on the data generated with other methods. The high residuals on the games per minute distribution quantify how poorly the generated data models this distribution.

The residuals between the generated data distributions and the distributions fitted to the generated data distributions are shown in table 5.4. From these results, we find that the real session duration and coin-in per session distributions are fit by a Gaussian distribution just as well as they are by a Skew Normal distribution, however the distributions of these quantities in generated data are found to be slightly more Skew Normal as evidenced by their smaller residuals.

5.2 Jensen-Shannon Divergence

The Jensen-Shannon divergence (31), (32) $JSD(P||Q)$ can be used as a metric for the similarity between two probability distributions P and Q which are defined on the same probability space X. It is a symmetric divergence measure that can be expressed in terms of the Kullback-Leibler divergence $D(Q||M)$ as shown in equation 5.3 where $M = \frac{1}{2}(P + Q)$. When the base-2 logarithm is used, the Jensen-Shannon divergence has the range $[0, 1]$. Values near zero

Table 5.4: Comparing residuals between the distributions in generated and real data and their fitted distributions. Here, each quantity in each distribution received its own, unique fit.

Plot Type	Method Name	Gaussian Residuals	Skew Normal Residuals
Session Duration	Real	0.052	0.052
Session Duration	Add-x	0.119	0.092
Session Duration	MLE	0.121	0.094
Session Duration	Prob Update	0.134	0.118
Session Duration	Unif Occ	0.122	0.095
Coin-in Per Session	Real	0.091	0.089
Coin-in Per Session	Add-x	0.166	0.138
Coin-in Per Session	MLE	0.162	0.138
Coin-in Per Session	Prob Update	0.163	0.163
Coin-in Per Session	Unif Occ	0.464	0.139
Games Per Minute	Real	0.219	0.073
Games Per Minute	Add-x	0.206	0.202
Games Per Minute	MLE	0.207	0.267
Games Per Minute	Prob Update	0.203	0.197
Games Per Minute	Unif Occ	0.205	0.205

indicate that the distributions are similar and values near one indicate the distributions are quite different.

$$
\begin{aligned}
JSD(P||Q) &= \frac{1}{2}D(P||M) + \frac{1}{2}D(Q||M) \\
&= \frac{1}{2}\sum_{x\in X} P(x)log(\frac{P(x)}{M(x)}) + \frac{1}{2}\sum_{x\in X} Q(x)log(\frac{Q(x)}{M(x)})
\end{aligned}
\tag{5.3}
$$

We can apply the Jensen-Shannon divergence measure to distributions from the generated and real data to assess their similarity. A Jensen-Shannon divergence of zero would indicate that the distributions are exactly the same, but for the purposes of anonymization, this result is undesirable. Generated data distributions should retain the shape and domain of the real data distribution, but there should be differences present to make it more difficult to trace the source data set of the simulation. If the source of the data set used to perform the simulation is untraceable, then the generated data set may be considered a fundamentally different data set unbound by a non-disclosure agreement.

The most desirable result ultimately depends on how different the generated data set and real data set are required to be for the generated data to be sufficiently anonymous. This anonymity threshold depends on how much statistical signatures (such as the Jensen-Shannon divergence between distributions of certain quantities) of data from different casinos vary in reality.

In this section, it will be determined if generated data distributions retain their shape and domain well by reporting the Jensen-Shannon divergence for a variety of important quantities derived from the data as well as plotting both the difference between the relative frequency in the real data and generated data and the relative frequency in the real data alongside the relative frequency in the generated data. This way the similarity in the shape of distributions can be compared both visually and numerically.

The relative frequency plots of the two distributions being compared are made comparable by binning the histogram of the given quantity in the generated data according to the binning in the histogram of the same quantity in the real data. This sometimes results in some values found in the generated data not being included in the plots if they are found outside the range of the bins in the histogram of the real data, resulting in a slightly smaller sample size than expected. These eliminated data points always have a very small frequency and are not very extreme compared to the real data domain. Ignoring them does not significantly impact the results.

The plots and numeric results were computed for both the higher occupancy ($J = 100$) and lower occupancy data ($J = 10$), but there was not a very noticeable discrepancy between them. For this reason, the plots and results are displayed only for the low occupancy data in this section. Any differences that are present in how well the two data sets approximate the distributions examined in this section can be gathered from figure 5.1 where the Jensen-Shannon divergences for each pair of data sets and each distribution from those data sets are plotted side-by-side. The high occupancy data set approximates the distributions of games per minute and machine occurrences in session data

Table 5.5: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for coin-in per event.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0002	0.0001	0.0010
MLE	0.0002	0.0001	0.0010
Prob Update	0.0093	0.0005	0.0156
Unif Occ	0.0002	0.0001	0.0010

worse than in the low occupancy data. These are both session level quantities and are as a result likely more distorted by the increased amount of randomness that more players introduce. The games-per-minute distribution was already poorly approximated by the low occupancy data and it appears that whatever is causing this poor approximation is exacerbated by the increased number of randomly made decisions. The one instance where the higher occupancy data performs noticeably better is on the time-between-events distribution. This is an event-level quantity and so the increased randomness may contribute to a better approximation due to the increased number of decisions being made and thus a larger sample size. We do not observe these patterns with other session level or event level quantities however so there may be some other underlying reason for the larger discrepancies we observe for these particular quantities.

5.2.1 Coin-in Per Event

In figures 5.3 and 5.2 we see that MLE-based methods (MLE, Add-X, and Uniform Occupancy) approximate the real coin-in per event distribution very

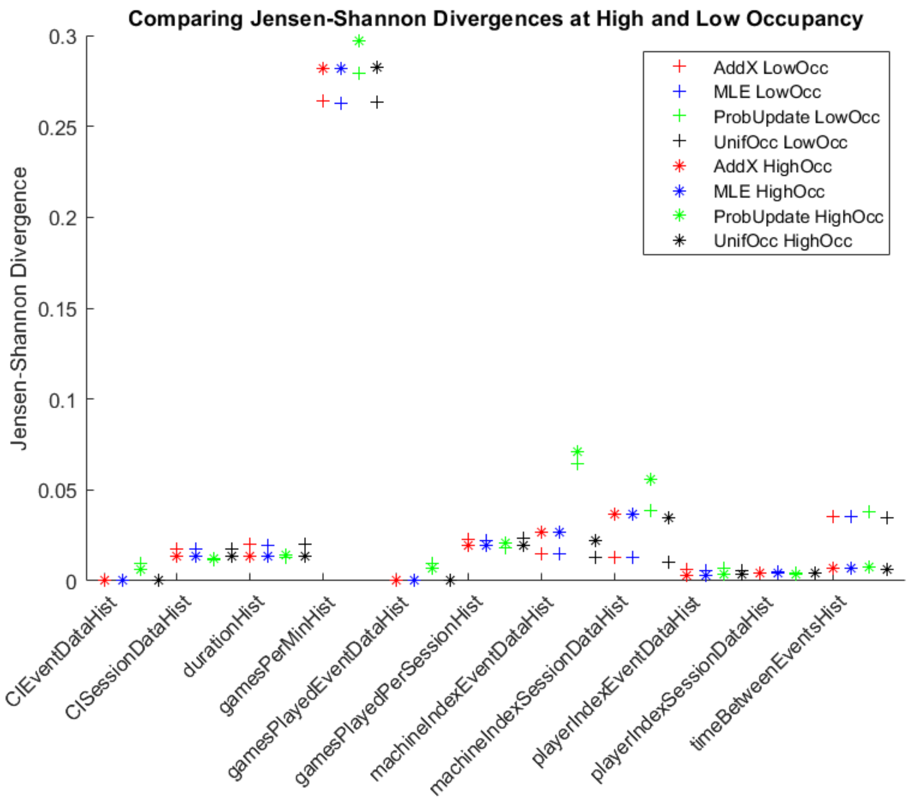

Figure 5.1: The difference the relative frequency of coin-in per session between data generated using the given probability estimation method and the real data.

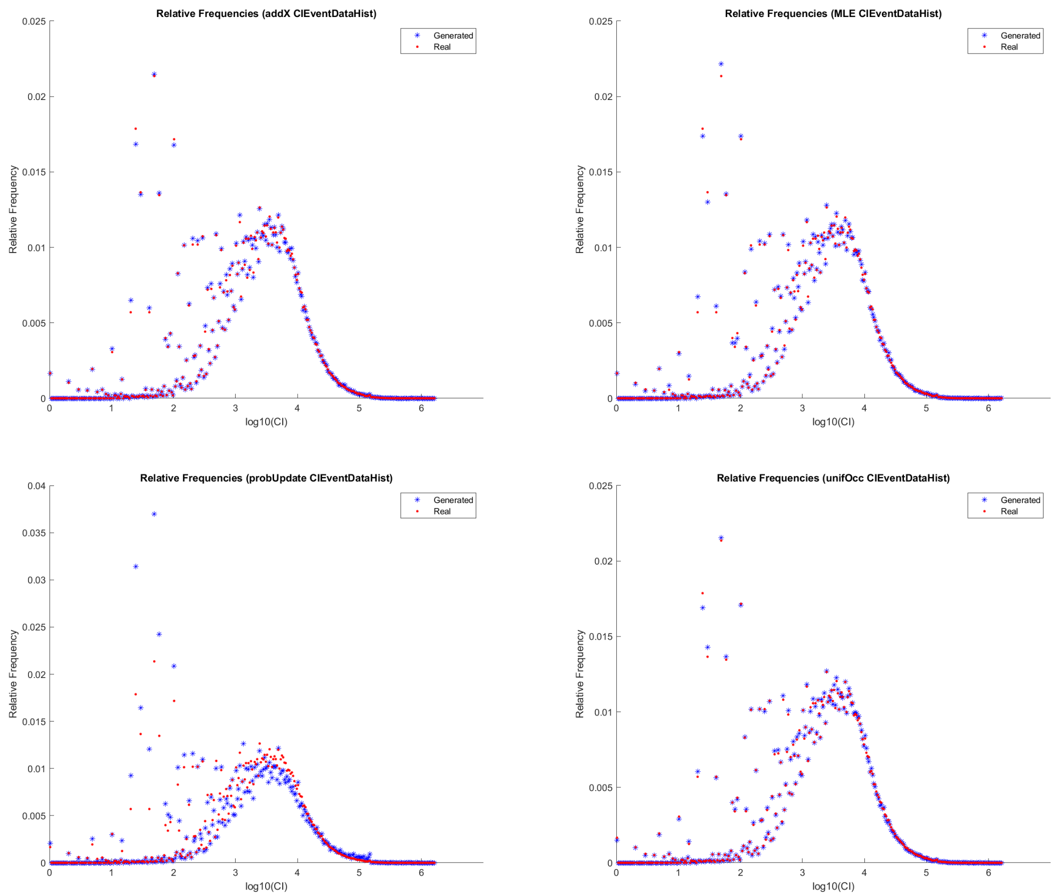

Figure 5.2: Relative frequency of coin-in per event in real data and data generated with the given probability estimation method plotted together.

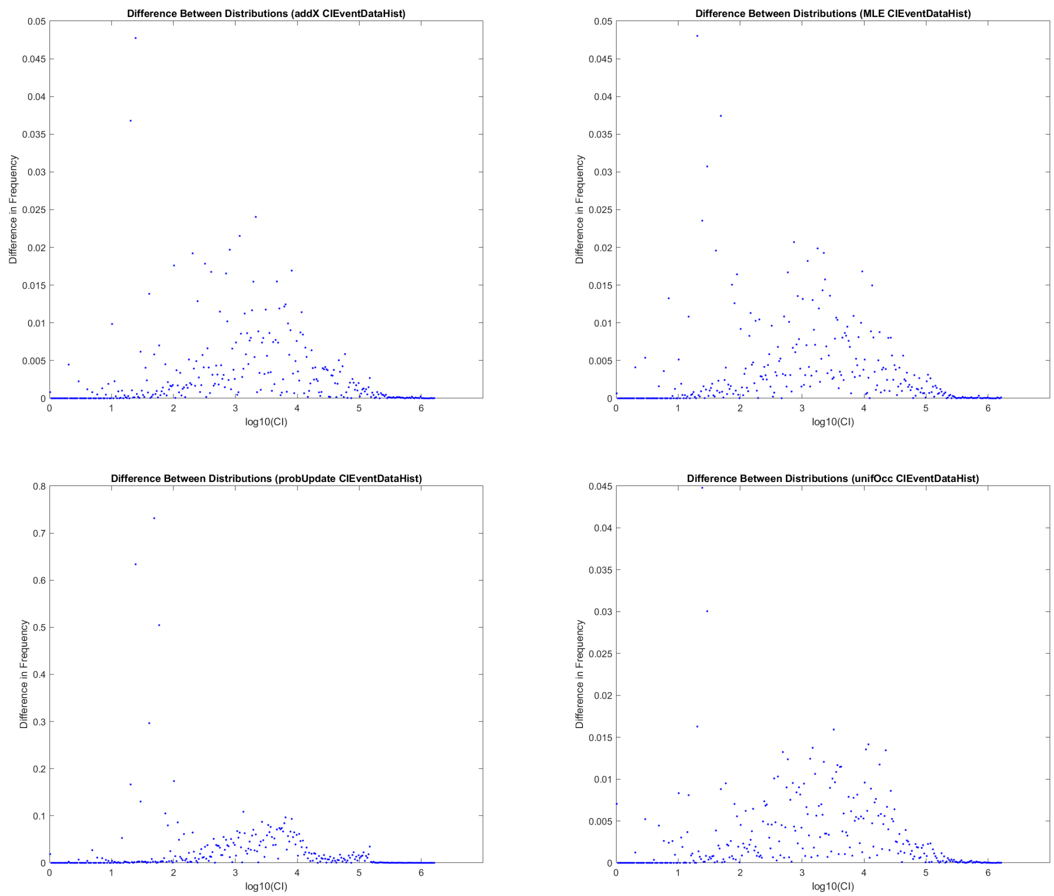

Figure 5.3: The difference in the relative frequency of coin-in per event between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

closely. For the Probability Update method, the left-hand side of the distribution is more chaotic, while the right-hand side follows more closely to the real distribution, but not to the extent of the distributions created using the MLE-based methods. These graphical observations are evidenced by the numeric results shown in table 5.5 where the Probability Update method has by far the highest JSD, mean divergence, and maximum divergence while the values given for all other methods are smaller and identical. Numerically, the Uniform Occupancy method's distribution is indistinguishable from the distribution produced by the benchmark methods.

The biggest discrepancy between the Probability Update distribution and the real one occurs at the fifty cent mark. A fifty cent insertion is about twice as common in the generated data as it is in the real data. The largest discrepancies occur at low values of coin-in which are common multiples of five and ten. These are amounts of money that any person passing by would be likely to have in their pocket and are small enough amounts that someone would insert this amount of money for a quick game as they are passing by. This implies that the exact frequency with which players insert these amounts into a machine is much more difficult to predict than, for instance, the frequency with which someone will input one hundred dollars into a machine as that is more commonly a calculated decision. This is just a hypothesis however and cannot be proven without access to more data sets. This is also likely the same reason that the left side of the coin-in per event distribution deviates from the apparent Gaussian structure of the distribution more than the right side in

Table 5.6: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for coin-in per session.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0177	0.0011	0.0050
MLE	0.0172	0.0011	0.0051
Prob Update	0.0119	0.0006	0.0045
Unif Occ	0.0176	0.0011	0.0050

the real data as well. The fact that the Probability Update method replicates these discrepancies similarly but not exactly as the other methods do may be a sign that the simulation is capturing the underlying nature of the distribution rather than just copying it.

Using the MLE-based methods, the coin-in per event distribution is nearly indistinguishable from the real one. The Probability Update method results in a coin-in per event distribution that is distinguishable while still maintaining the shape of the real data distribution.

5.2.2 Coin-in Per Session

In figures 5.5 and 5.4, the MLE based methods produce extremely bimodal frequency difference plots due to the generated data distribution leaning slightly to the left of the real data distribution. This feature is much less pronounced for the distribution produced by the Probability Update method. All distributions from generated data have a cleaner left-hand side than in the coin-in per event distribution, but it still deviates visibly from the apparent Gaussian structure

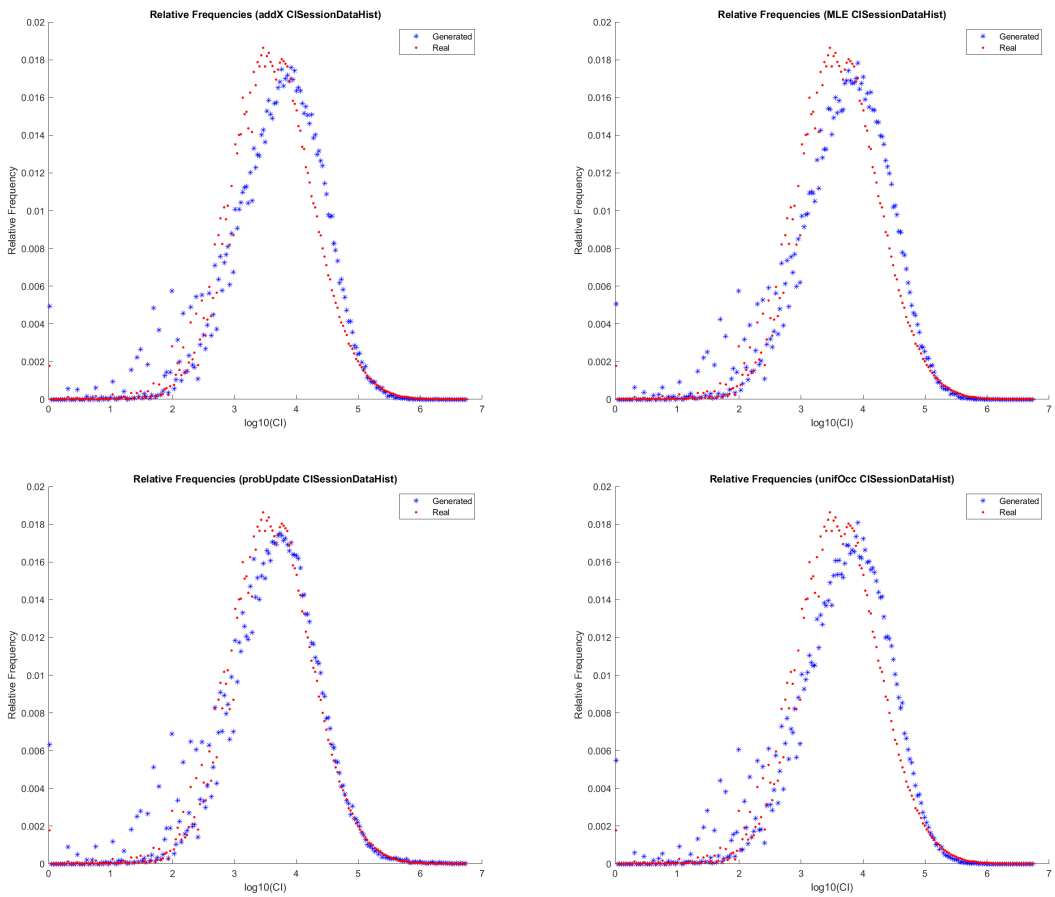

Figure 5.4: Relative frequency of coin-in per session in real data and data generated with the given probability estimation method plotted together.

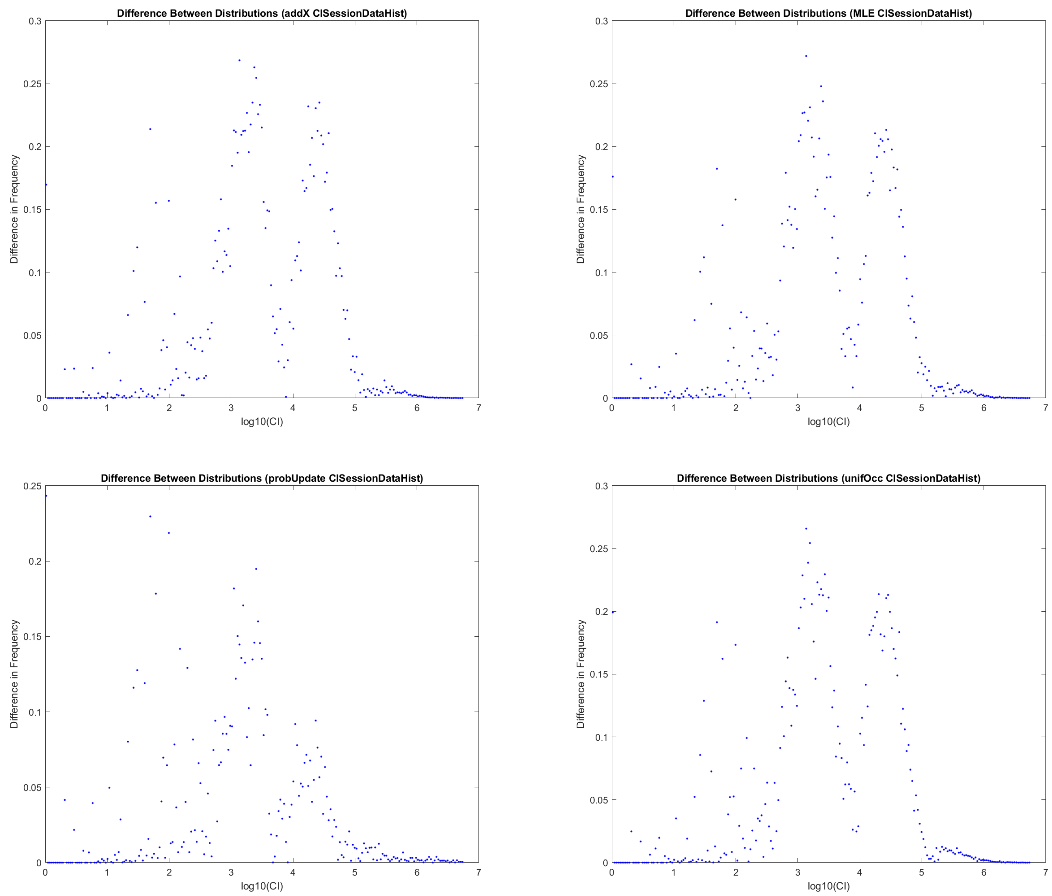

Figure 5.5: The difference in the relative frequency of coin-in per session between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

Table 5.7: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for time elapsed between events.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0352	0.0006	0.0031
MLE	0.0352	0.0006	0.0033
Prob Update	0.0377	0.0008	0.0100
Unif Occ	0.0344	0.0006	0.0032

compared to the real data distribution's right-hand side. The improvement in the magnitude of these deviations demonstrates the value that conversion to sessions has in the mitigation of noise.

The JSDs tabulated in table 5.6 show that all MLE-based methods perform well and about the same in their approximation of the distribution of coin-in per session. Again, the Uniform Occupancy method's results are no different than those of the benchmark methods. The Probability Update method performs significantly better however. The Probability Update method approximates the distribution of coin-in per session better than any other method, but approximates the distribution of coin-in per event worse than any other method. This may indicate that the Probability Update method is producing a new version of the event data rather than just shuffling it.

5.2.3 Time Elapsed Between Events

The plots shown in figure 5.6 show two sudden jumps in probability at fifteen minutes (2.95 base-10 log seconds) and one hour (3.55 base-10 log seconds).

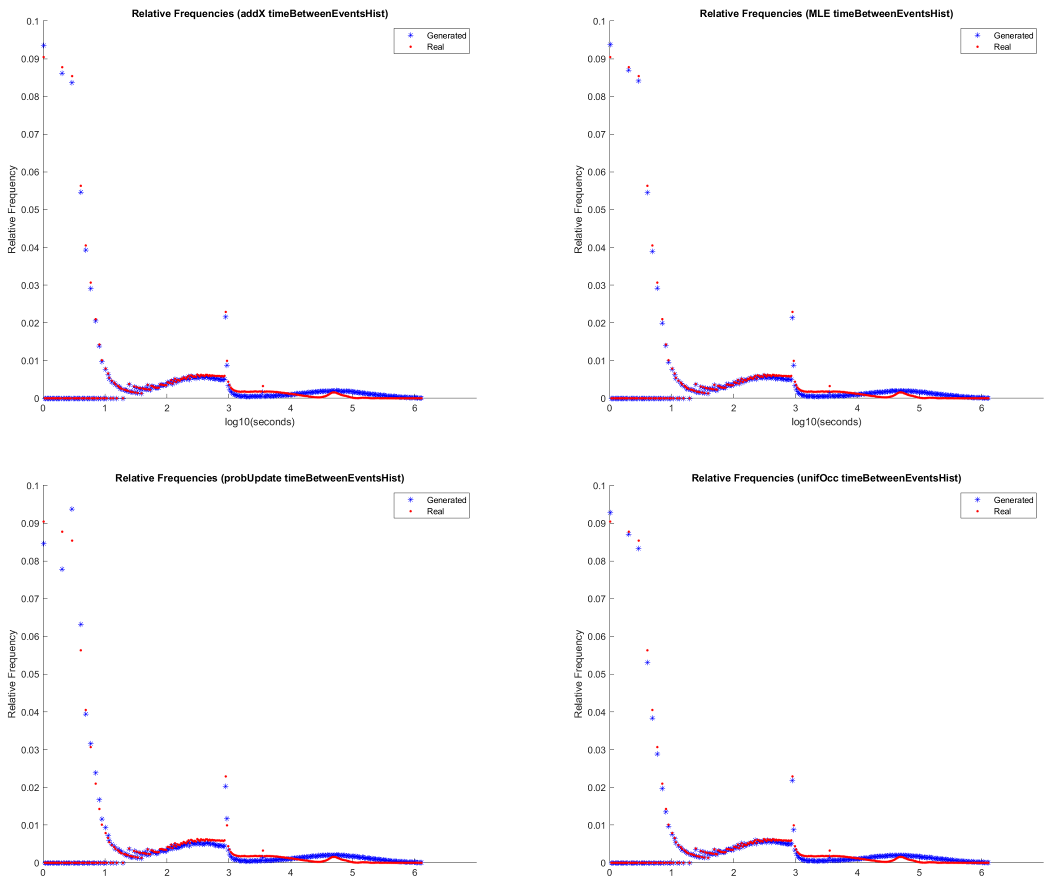

Figure 5.6: Relative frequency of time between consecutive events in real data and data generated with the given probability estimation method plotted together.

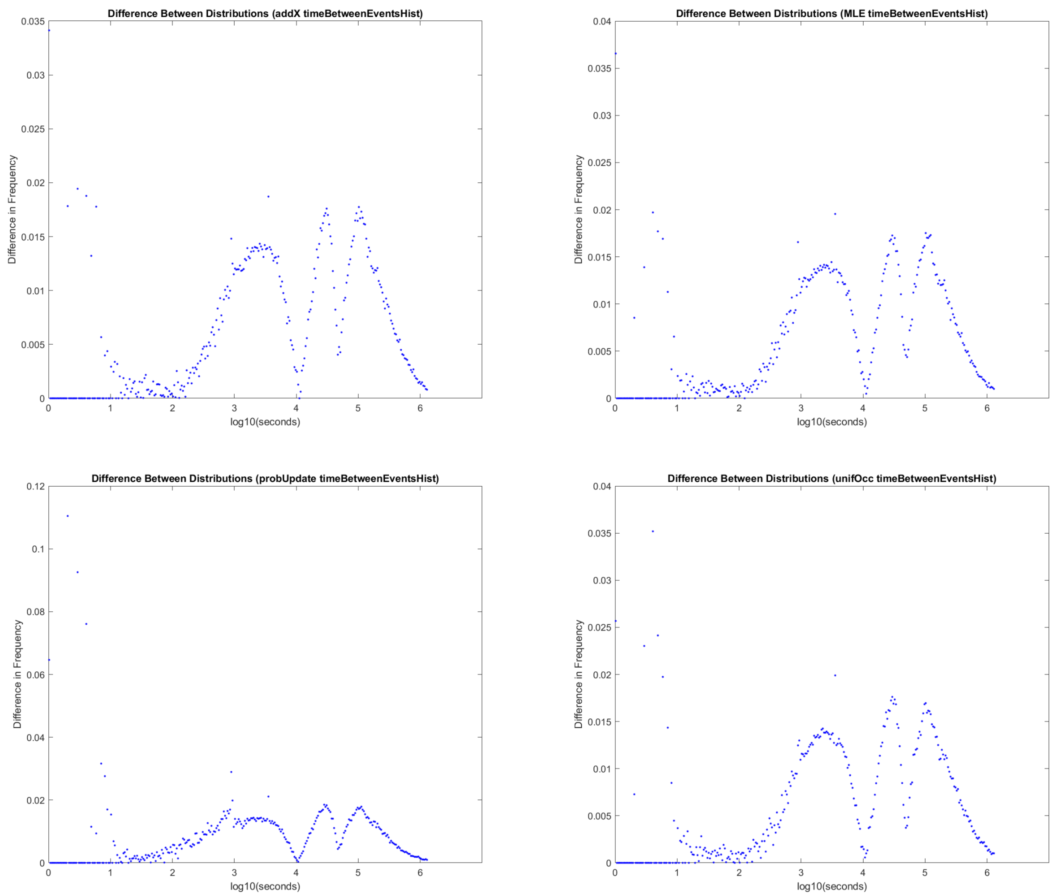

Figure 5.7: The difference in the relative frequency of the time between consecutive events between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

These jumps are likely indicative of some polling events being present in the data despite best efforts to remove them. This is likely a product of having to speculate on the meaning of the event codes in the provided data set and are unlikely to be player triggered events.

The distributions produced using the MLE-based methods follow the real distribution closely up until $1e3$ seconds. The distribution produced by the Probability Update method has a similar pattern to those of the other methods, but also performs poorly for the four lowest non-zero probabilities. These probabilities occur at one, two, three, and four seconds. The granularity of time is one second, so this is why these lower integer values are surrounded by zero probability. Just as in the coin-in plots, lower value probabilities diverge from their real counterparts more than for higher values for distributions produced using the Probability Update method. The shapes of the distributions begin to differ noticeably near $1e4$ seconds which is equal to about two hours, which is the timeout used for when a player is forced to leave the simulation. Perhaps a more suitable timeout should be chosen closer to $1e5$ seconds where the real data distribution converges to near zero probability.

Numeric results are given in table 5.7. Here, the Uniform Occupancy method approximates the real data distribution the closest. Again, the Probability Update method approximates an event-level quantity distribution the most poorly, but not by a significant amount for most of the distribution. For the Probability Update method, it is the maximum divergence that is significantly larger than it is for all other methods. If we want the general shape of the

Table 5.8: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for duration of sessions.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0202	0.0009	0.0044
MLE	0.0198	0.0009	0.0044
Prob Update	0.0129	0.0006	0.0045
Unif Occ	0.0200	0.0009	0.0045

distribution from generated data to match the distribution from real data, but not exactly, the Probability Update method does this the best as its shape is the same as any other method's distribution, but it has the largest divergences. The Probability Update method produced a distribution with large divergence from the real distribution especially for those small integer values as discussed previously. For the best match, the Uniform Occupancy method performs the best at approximating the distribution of time elapsed between events.

5.2.4 Session Duration

In figure 5.8, all methods result in sessions that tend to have a longer duration than those from the real data. This might indicate that the way that it is decided that a player will leave is inaccurate, causing them to stay longer than they would in reality. The *Probability Update* method captures the right-hand tail of the distribution well however, which can also be seen in figure 5.9 where the difference in relative frequency drops off faster than in the plots for other methods. The left hand tail of all of the distributions from generated data do not match up as well, indicating that while sessions tend to be longer in

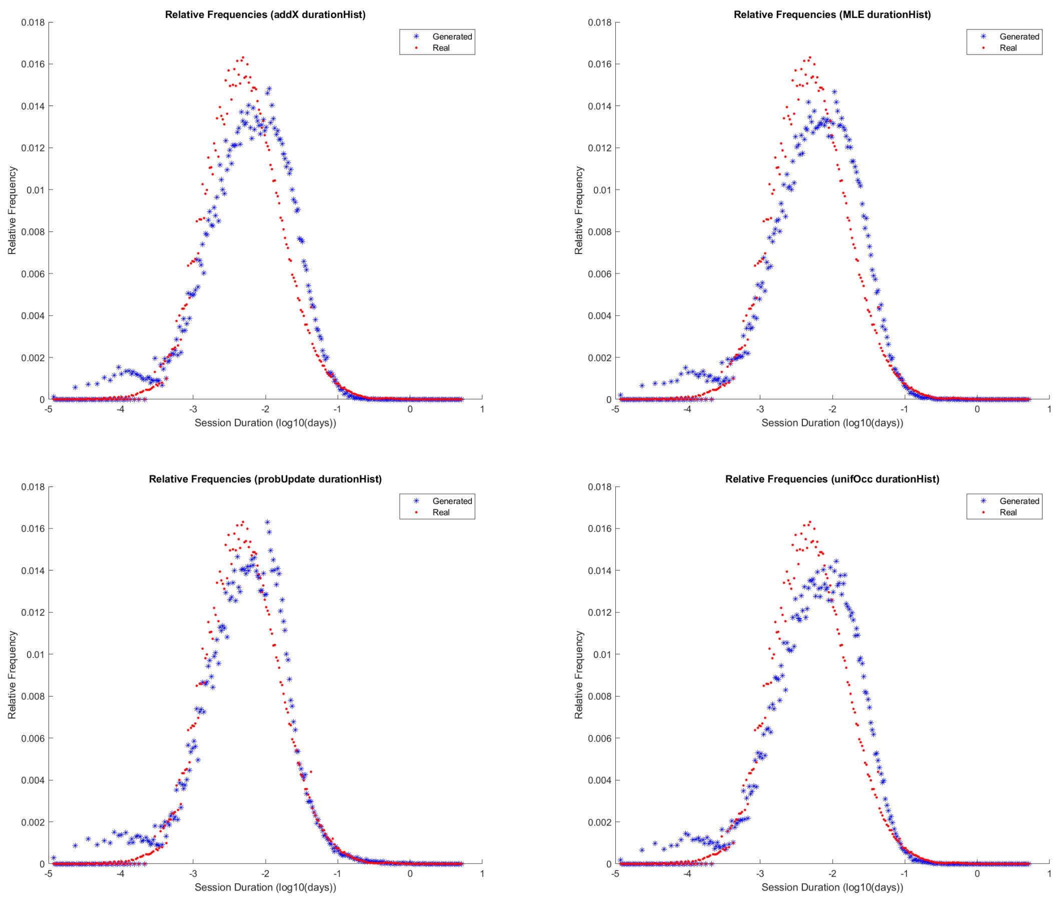

Figure 5.8: Relative frequency of session duration in real data and data generated with the given probability estimation method plotted together.

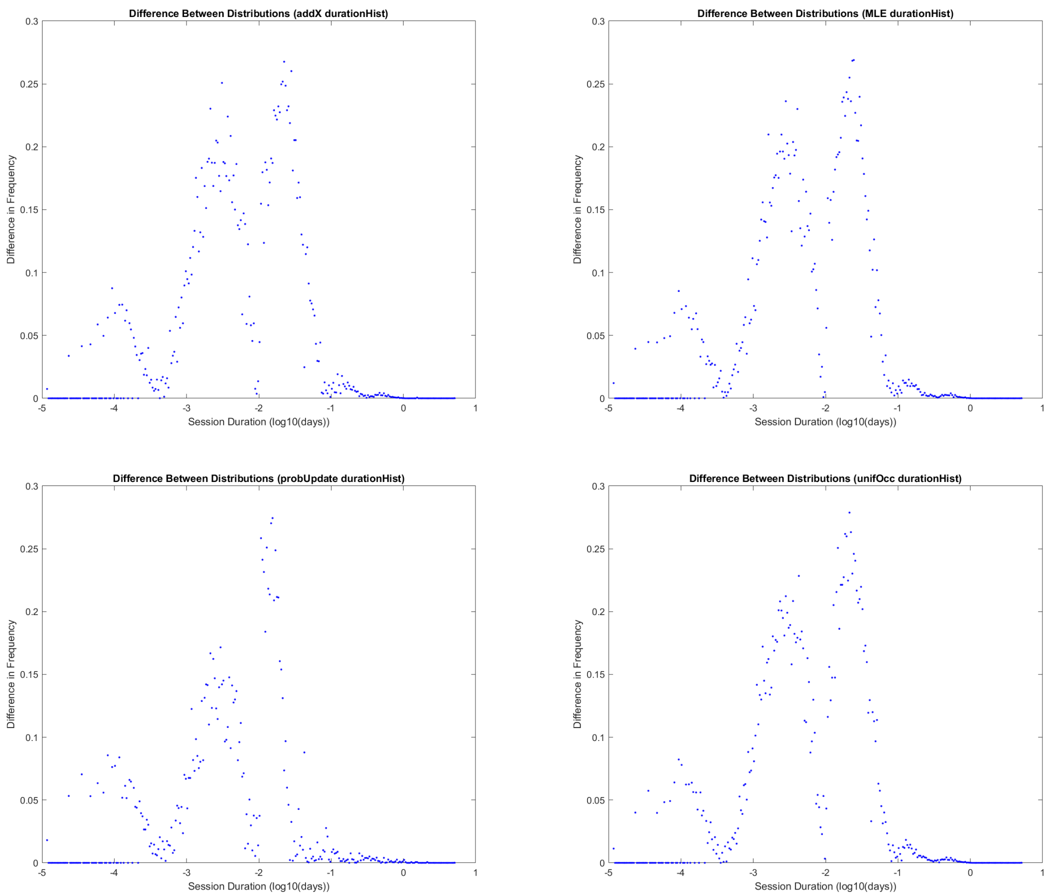

Figure 5.9: The difference in the relative frequency of session duration between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

generated data, there is also a higher frequency of very short sessions. The fatter left-hand tail of the generated session duration distributions can possibly be attributed to the granularity of time that has been adopted in the simulation. Each bin of non-zero probability in the left hand tail encompasses an integer number of seconds. The higher frequency of these sessions in the generated data indicates that players are too often ending their sessions after only a few seconds

The histogram of duration shows a very smooth distribution compared to the coin-in distributions in the previous sections. This is because duration can be incremented second by second, not in certain denominations like coin-in often is depending on the denomination of the machine it is inserted into.

Numeric results for the session duration distribution are found in table 5.8. Once again, the Probability Update method performs better than all other methods when approximating the distribution of a session-level quantity. The Uniform Occupancy does not perform significantly better or worse than the benchmark methods.

5.2.5 Games Per Minute

Figure 5.10 shows that the games per minute distribution from generated data is distorted compared to the real data distribution. The peak of the distribution from generated data aligns with the peak of the distribution from the real data, but this is the extent to which an argument of good fit could be made.

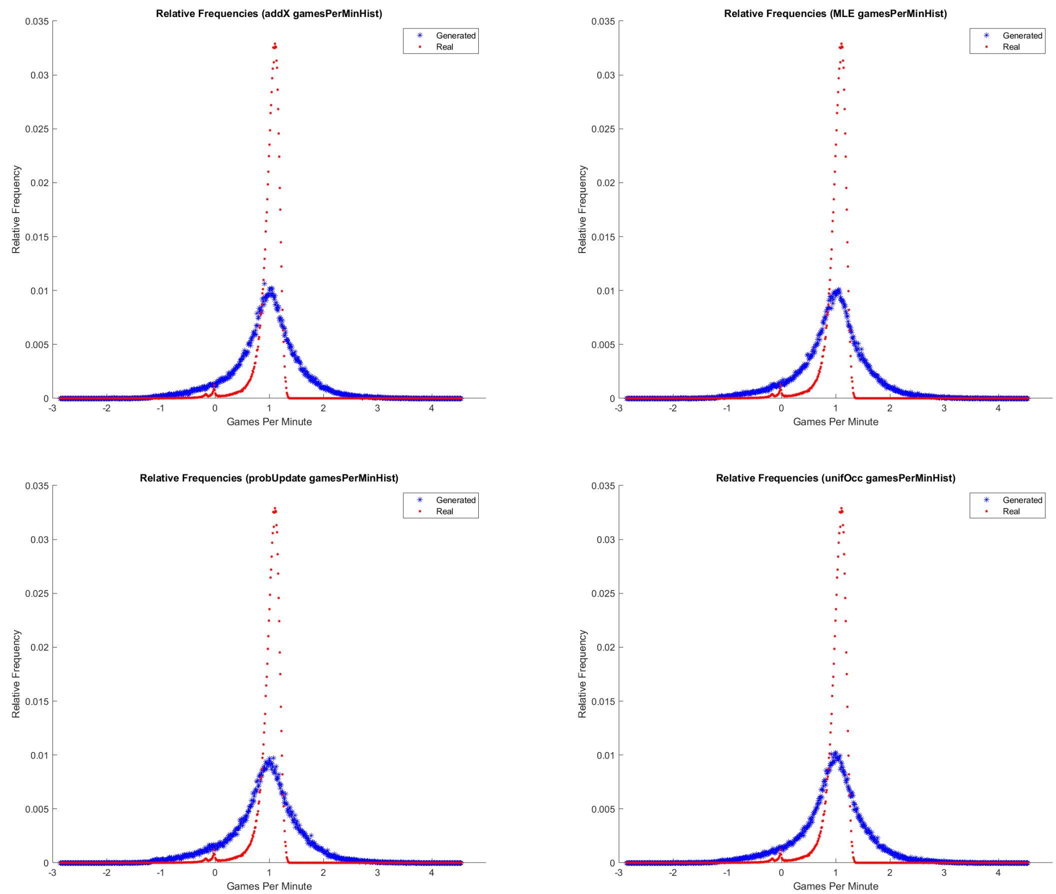

Figure 5.10: Relative frequency of games per minute of session time in real data and data generated with the given probability estimation method plotted together.

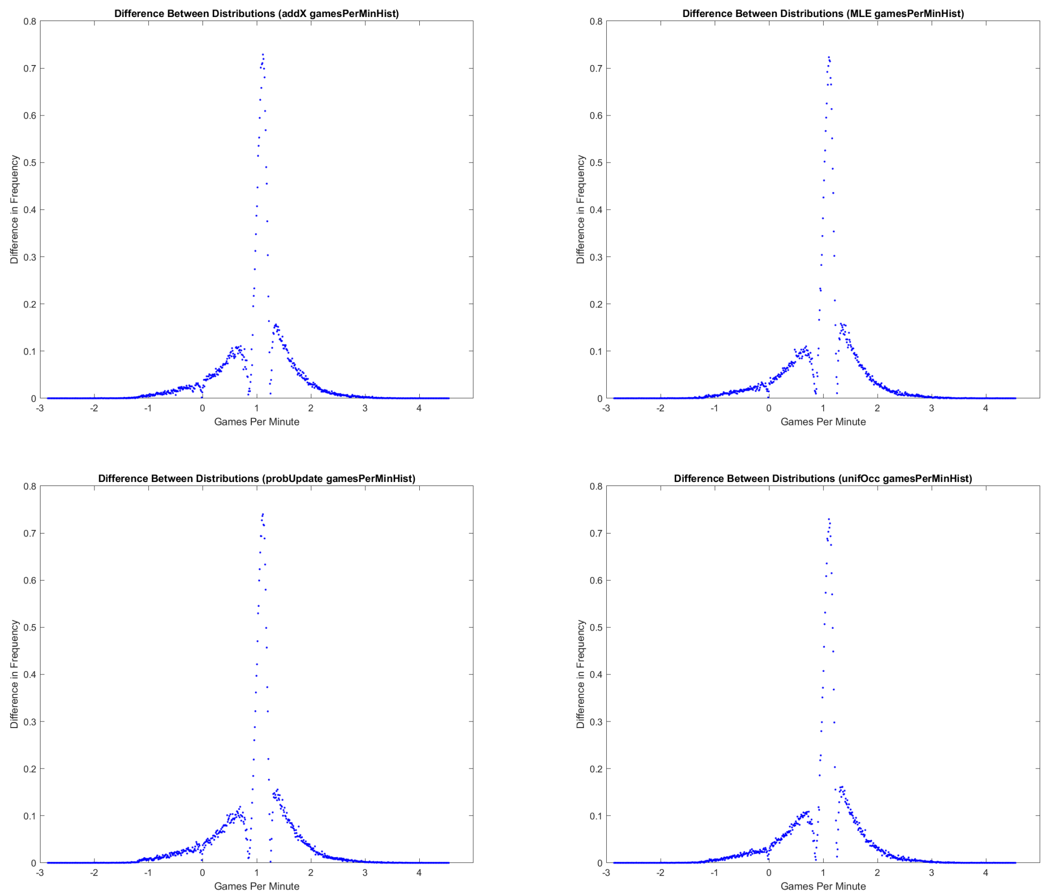

Figure 5.11: The difference in the relative frequency of games per minute of session time between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

Table 5.9: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for games per minute of session time. To obtain the games per minute value for each session, the number of games played during the session is divided by the session length in minutes.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.2642	0.0013	0.0240
MLE	0.2626	0.0013	0.0238
Prob Update	0.2790	0.0014	0.0243
Unif Occ	0.2635	0.0013	0.0240

There is a higher frequency of games per minute lower than the mean and games per minute higher than the mean in the generated data, whereas the vast majority of sessions in the real data have about ten games per minute with very little deviation. The higher frequency of many games per minute might be attributed to the higher frequency of shorter sessions shown by figure 5.8. Additionally, the small features seen near zero in the real data distribution in figure 5.10 are not captured by the generated data. These small peaks in probability for low games per minute can possibly be attributed to automatic poker and keno games which are present in the data set but take much longer to play than just the pull of a slot machine handle. However, the machine information data provided gives no indication that these types of games are present in the real data so the reason for this small peak is just speculation.

The numeric results in table 5.9 agree that the games per minute distribution is poorly approximated by the simulation regardless of the probability matrix estimation method. The exact reason for the poor approximation is unclear,

Table 5.10: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for games played per event.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0002	0.0001	0.0007
MLE	0.0001	0.0001	0.0010
Prob Update	0.0093	0.0007	0.0561
Uunif Occ	0.0001	0.0001	0.0006

but it is a compounded session-level quantity. This means that it takes into account both session duration and games played per session. The greater the level of abstraction required to create a quantity, the poorer the approximation of its distribution is likely to be. The JSDs are too high to indicate that the distributions are very similar, but not so high that the distributions are considered to be entirely different either.

5.2.6 Games Played Per Event

The plots in figure 5.13 and 5.12 show that we are approximating the distribution of games played per event as accurately as we are the distribution of coin-in per event. Coin-in, coin-out, games played, and time elapsed are all drawn concurrently in a correlated manner so this is to be expected. The reason we did not see results like this for the time elapsed distribution is that there are other factors at play when determining the time elapsed between events as discussed in the appropriate section.

In figure 5.12, there is a pattern of overlapping lines of probability in

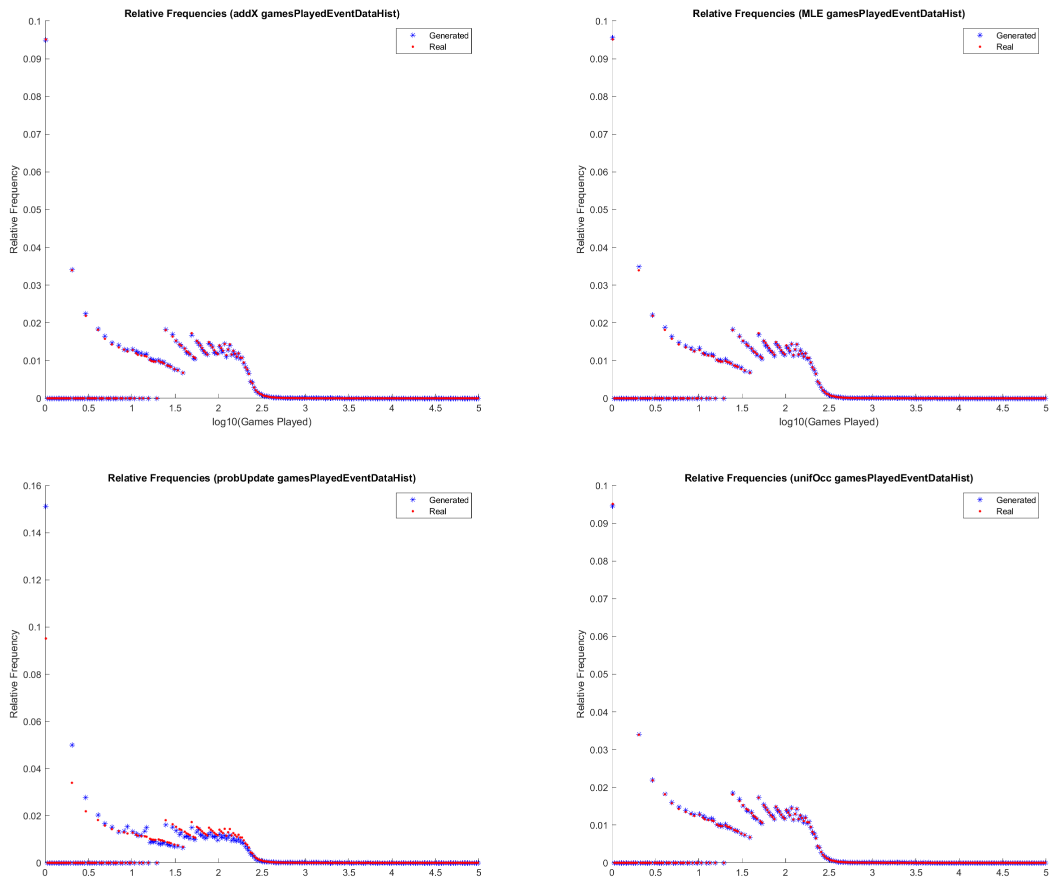

Figure 5.12: Relative frequency of games played per event in real data and data generated with the given probability estimation method plotted together.

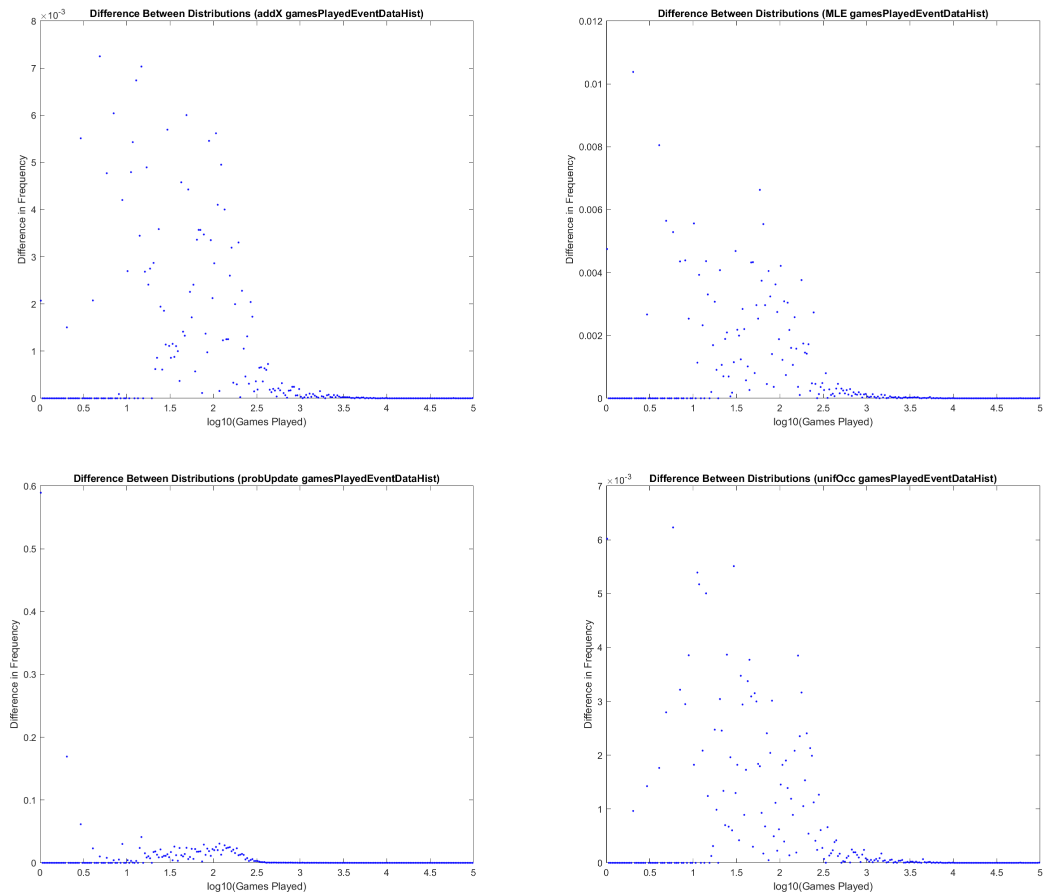

Figure 5.13: The difference in the relative frequency of games played per event between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

Table 5.11: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for games played per session.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0227	0.0012	0.0201
MLE	0.0224	0.0012	0.0199
Prob Update	0.0180	0.0008	0.0254
Unif Occ	0.0234	0.0012	0.0208

the range [1,2.5]. These may be caused by machine specific properties that may allow a player to engage in multiple games per figurative handle pull, causing certain quantities of games played per event to be more likely than neighbouring quantities. In the range [0,1], we again see quantities with non-zero probability surrounded by quantities with exactly zero probably caused by the quantification of games played. A player cannot play a fractional number of games.

Numeric results tabulated in table 5.10 show JSDs very close to or the same as those in table 5.5 for the coin-in distribution. Since games played and coin-in per event are perfectly correlated, this is to be expected.

It should be noted that while this distribution would not be of particular interest when studying a slot floor since it does not lend any insights into player behaviour, it is discussed here merely as a metric for the efficacy with which the simulation produces a viable data set.

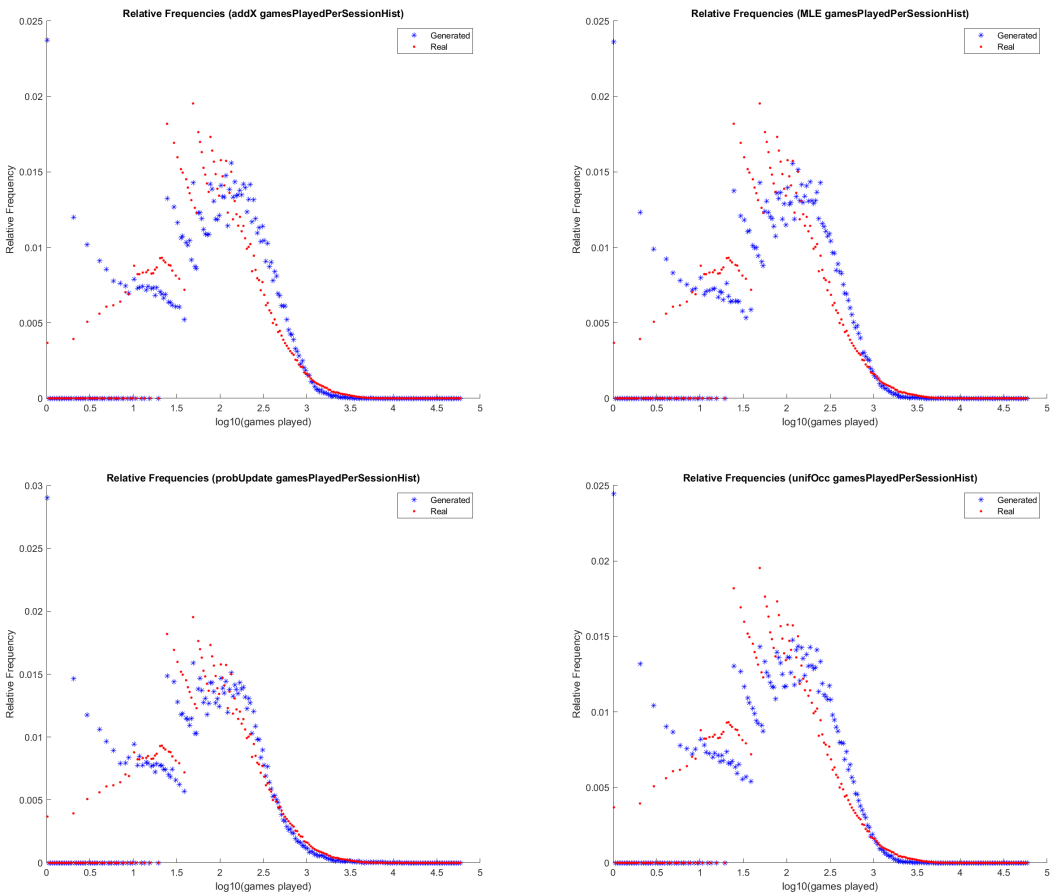

Figure 5.14: Relative frequency of games played per session in real data and data generated with the given probability estimation method plotted together.

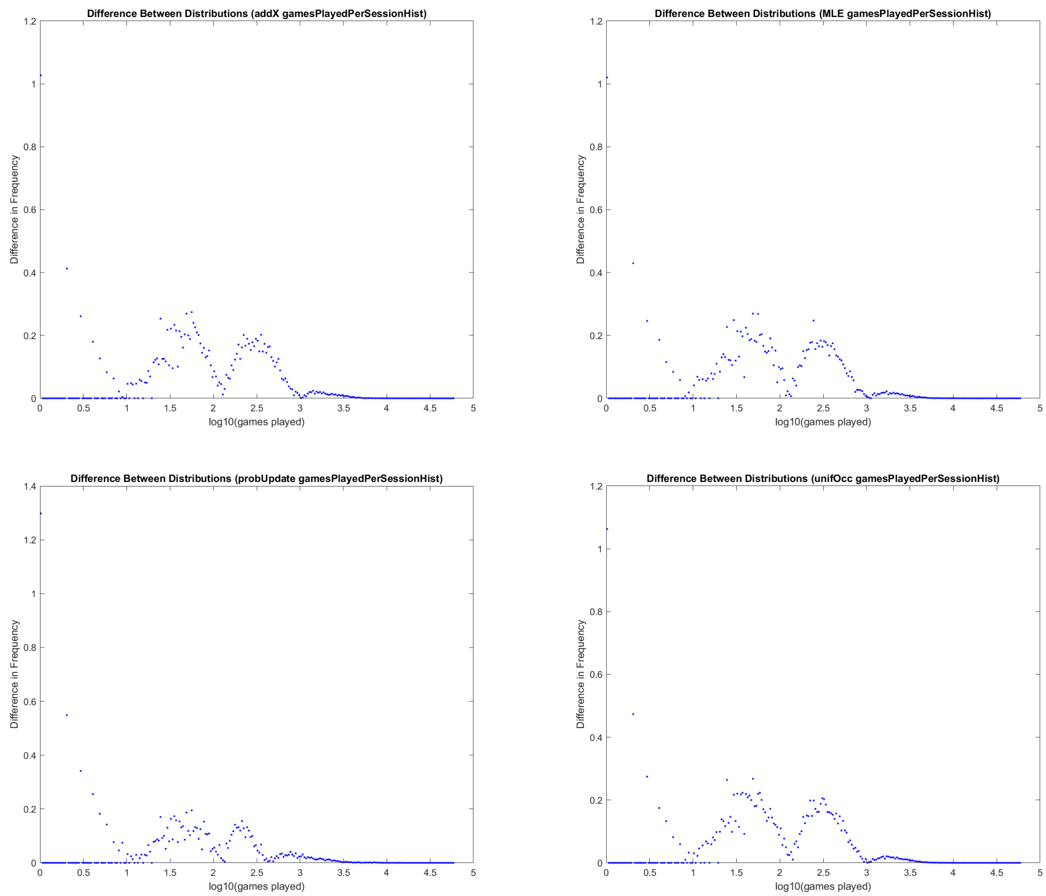

Figure 5.15: The difference in the relative frequency of games played per session between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution.

Table 5.12: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of machine occurrences in event data.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0150	0.0007	0.0036
MLE	0.0146	0.0006	0.0036
Prob Update	0.0647	0.0015	0.0101
Unif Occ	0.0127	0.0006	0.0039

5.2.7 Games Played Per Session

The poor fit demonstrated in figures 5.15 and 5.14 can mostly be attributed to the probability of values in the range [0,1] where the generated and real distributions trend in different directions. These are all sessions with less than ten games played and are much more common than they should be. This is especially true for sessions with one game played which is very low for an entire session. These sessions do also occur in the real data however, but with a much lower frequency. Outside of this range, the overall shape of the distribution is maintained, but not very closely for any method.

The numeric results in table 5.11 show that the Probability Update method approximates another session level quantity better than any other method. Its high maximum divergence shows that it has the largest disagreement in the low end of the distribution.

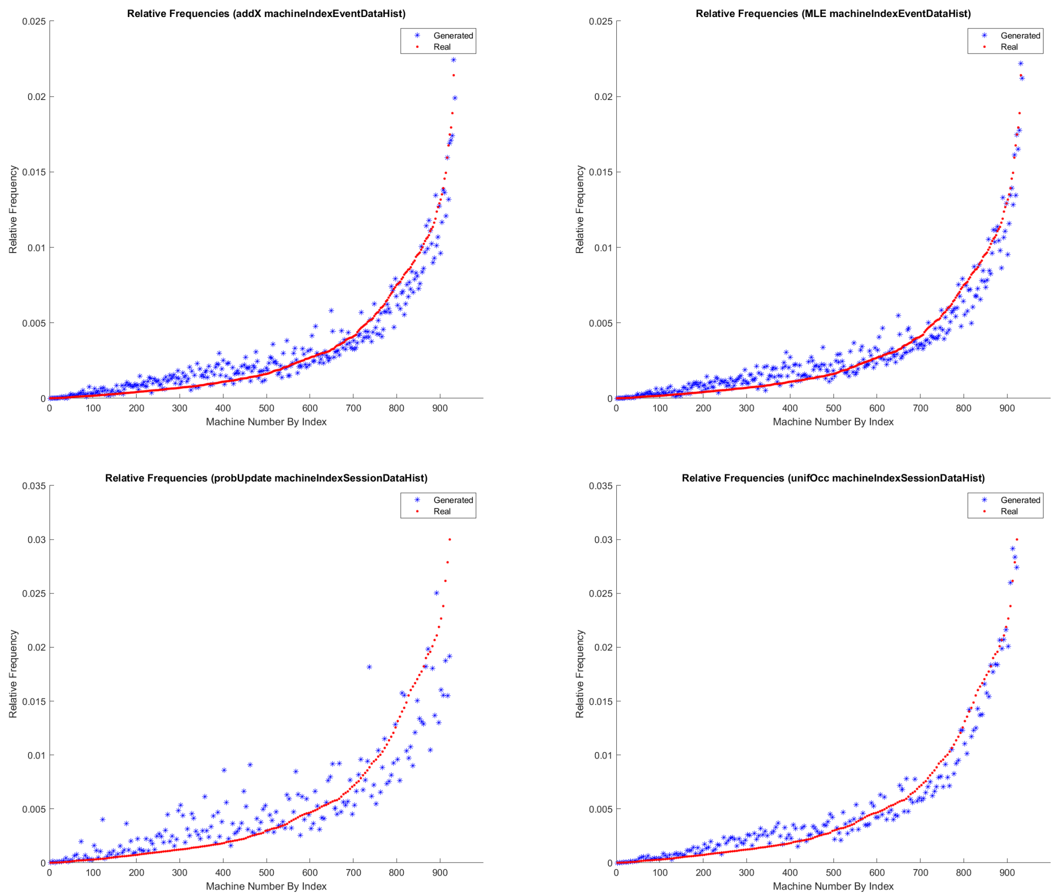

Figure 5.16: Relative frequency of machine occurrences in events in real data and data generated with the given probability estimation method plotted together. The machines are assigned an index in order of their prevalence in the real data. This index is what is reported here.

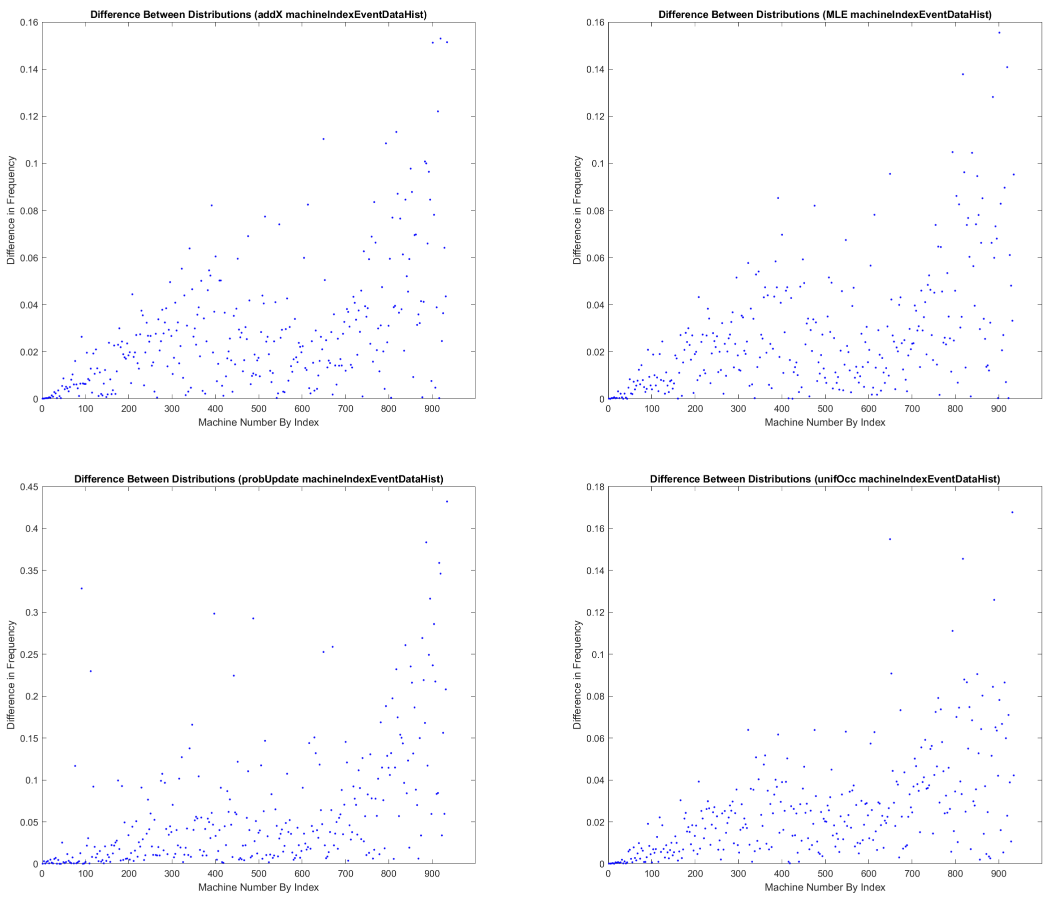

Figure 5.17: The difference in the relative frequency of machine occurrences in event data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The machines are assigned an index in order of their prevalence in the real data. This index is what is reported here.

5.2.8 Frequency of Machine Occurrences in Event Data

Machine numbers are drawn directly from the distribution of machine numbers in real data when a new player enters the simulation, but players can also transition between machines, introducing the possibility for the distribution of machine occurrences in generated data to become distorted as compared to the real data distribution. This is what we see has happened looking at figures 5.17. Distributions from data simulated using the MLE-based methods follow the real distribution much more closely than the distribution from data simulated using the Probability Update method. This would imply that using the Probability Update method, players are either transitioning between machines during their visit more often than with other methods, or there is just more variance in their choices.

We can determine which is the case by examining the distribution of the number of machines a player plays during their visit (i.e. before they are forced to leave by the simulation). The mean number of machines a player plays during their visit is the lowest for the Probability Update method at 2.93 games. All other methods have means around 3.3. The data generated with the Probability Update method also has the lowest percentage of visits that consisted of more than one machine played at 65%. All other methods produced data with 69% of visits consisting of more than one machine played. Interestingly, if we convert the real sessions to visits, we find that 65% of real visits consisted of more than one machine played, matching up precisely with

Table 5.13: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of machine occurrences in session data.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0129	0.0010	0.0048
MLE	0.0130	0.0010	0.0048
Prob Update	0.0388	0.0019	0.0124
Unif Occ	0.0104	0.0009	0.0034

the percentage in the data generated with the Probability Update method. We can then conclude that a simulation that uses a probability matrix constructed using the Probability Update method allows players to have more variance in their choice of machine to play. Players in such a simulation are less restricted than in a simulation using a probability matrix constructed using other methods. Judging by the JSD in table 5.12, the Uniform Occupancy method allows for even less variance than the benchmark methods do.

5.2.9 Frequency of Machine Occurrences in Session Data

The distributions of machine occurrences in generated session data follow the real distributions more closely than they did in the event data as can be seen in figures 5.19 and 5.18. JSDs are also reduced in comparison to those in the previous section as well as shown in table 5.13. Again, the real distribution is approximated the worst by the Probability Update method. We wouldn't expect this to change after the conversion to session data. Similarly, the Uniform Occupancy method approximates the distribution the best as it did in the event data as well. It can still be speculated that the Probability

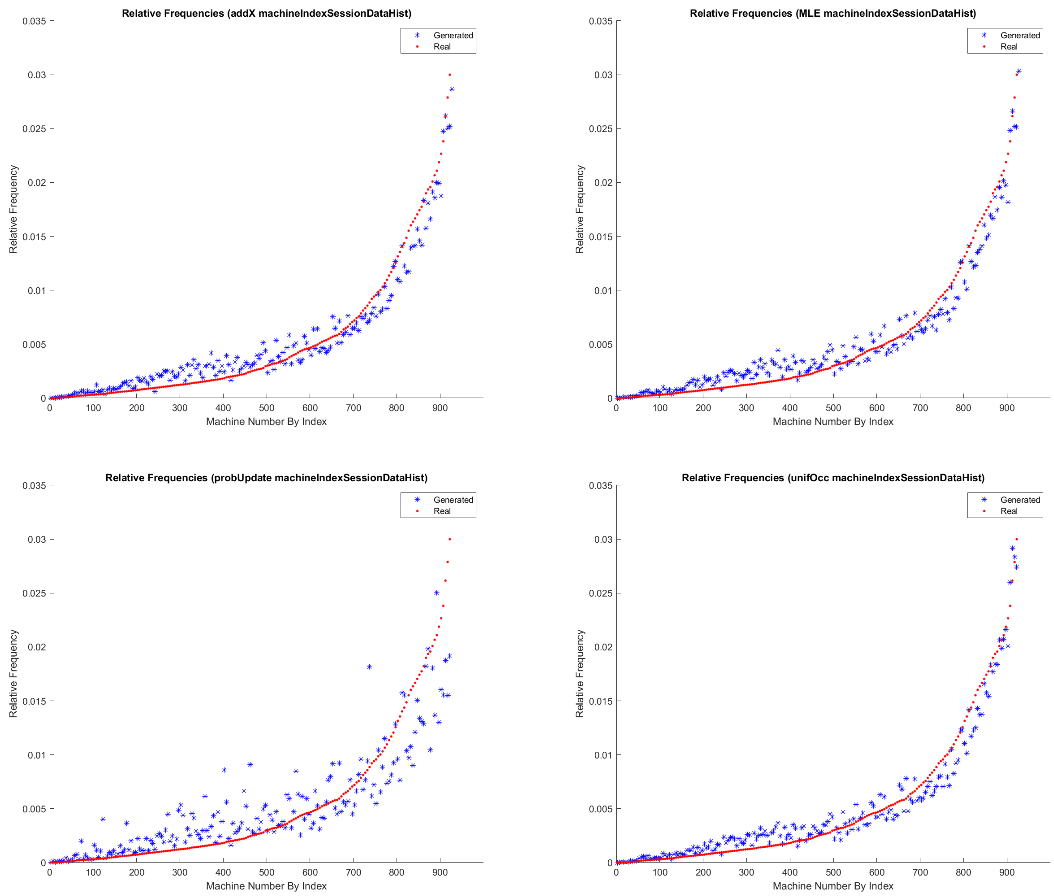

Figure 5.18: Relative frequency of machine occurrences in sessions in real data and data generated with the given probability estimation method plotted together.

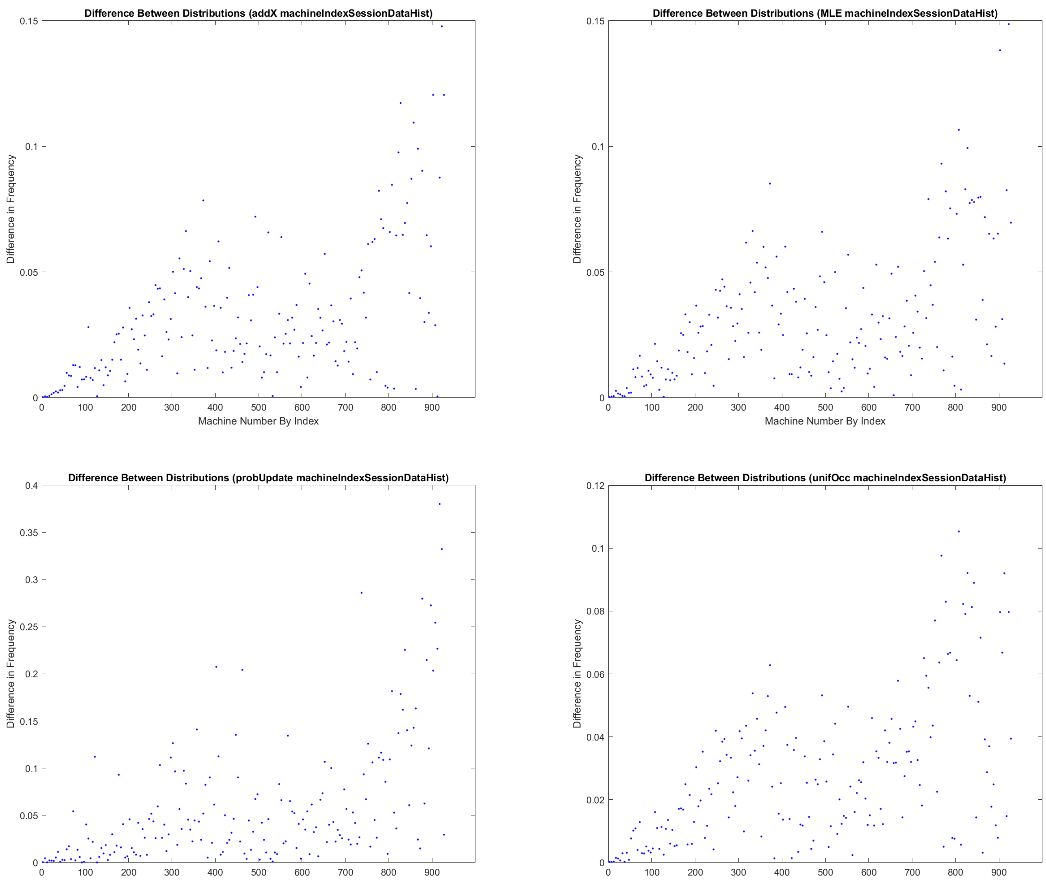

Figure 5.19: The difference in the relative frequency of machine occurrences in session data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The machines are assigned an index in order of their prevalence in the real data. This index is what is reported here.

Table 5.14: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of player ID occurrences in event data.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0061	0.0002	0.0041
MLE	0.0056	0.0002	0.0032
Prob Update	0.0070	0.0003	0.0034
Unif Occ	0.0056	0.0002	0.0045

Update method allows the most freedom of choice to simulated players without sacrificing the fidelity of the data as demonstrated by the still very low JSD.

5.2.10 Frequency of Patron ID Occurrences in Event Data

The distribution of player occurrences in event data is approximated very well by all methods. The method for choosing player IDs is not dependent on the probability matrix. As discussed in a previous section, player IDs are drawn directly from their distribution in the real data with no mitigating factors caused by the probability matrix as with the distribution of machines. The frequency with which players are drawn does depend on the probability matrix however which allows for the small amount of variation we see in the distributions from generated data (figures 5.21 and 5.20). JSDs and other divergences are reported in table 5.14.

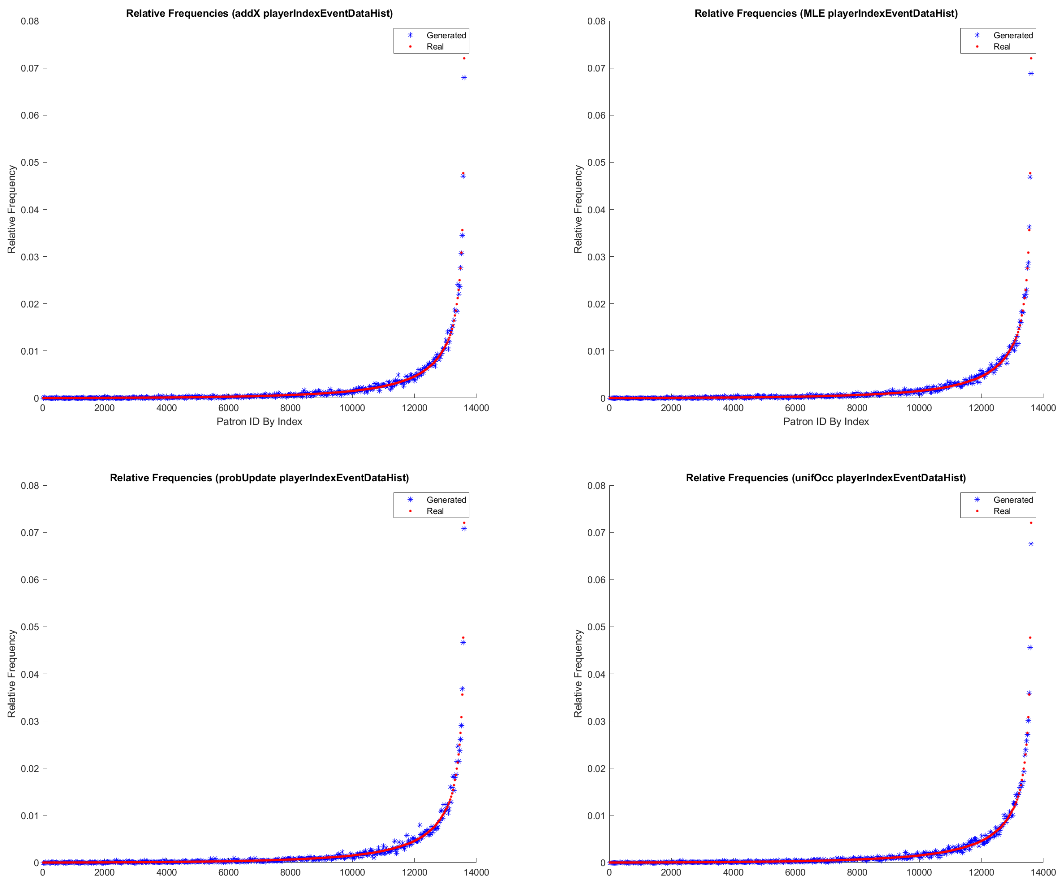

Figure 5.20: Relative frequency of player occurrences in event data in real data and data generated with the given probability estimation method plotted together.

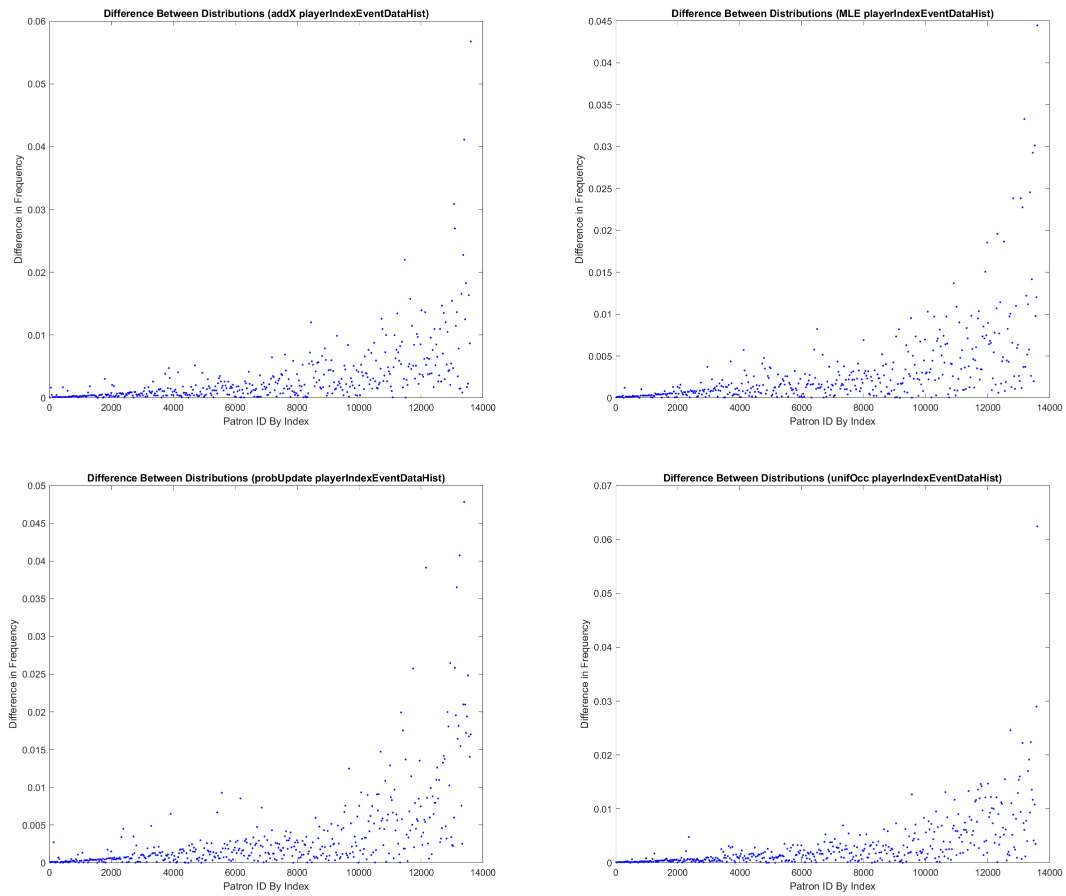

Figure 5.21: The difference in the relative frequency of player occurrences in event data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The players are assigned an index in order of their prevalence in the real data. This index is what is reported here.

Table 5.15: Jensen-Shannon divergences, mean divergence between the distributions, and maximum divergence between the distributions for frequency of player ID occurrences in session data.

Method Name	JSD	Mean Divergence	Max Divergence
Add-x	0.0045	0.0004	0.0120
MLE	0.0047	0.0004	0.0115
Prob Update	0.0044	0.0004	0.0110
Unif Occ	0.0041	0.0004	0.0151

5.2.11 Frequency of Patron ID Occurrences in Session Data

The distribution of player occurrences in generated session data approximates the real session data distribution well (figures 5.23 and 5.22). Just as with the distribution of machine index, the distribution in session data is better approximated than the distribution in event data as evidenced by the divergences shown in 5.15. These distributions may be better approximated in session data because there can be a highly variable number of events in a session. The number of events in a session is highly influenced by the probability matrix, where as any number of events starting with a card-in event and ending with a card-out event is considered only one session. This is one way in which conversion to sessions reduces noise in data.

5.3 The Time Series

The goal of the simulation is not only to model the real data statistically as discussed in the previous section, but to also model the structure of the time

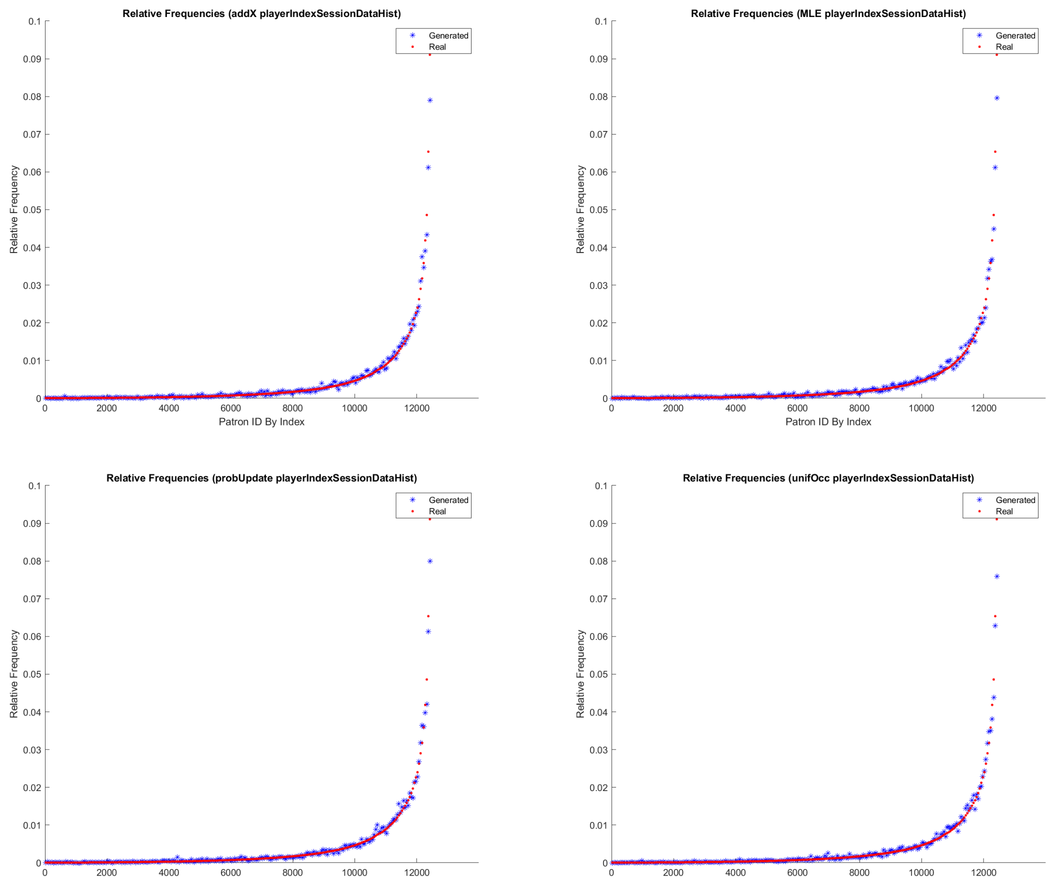

Figure 5.22: Relative frequency of player occurrences in sessions in real data and data generated with the given probability estimation method plotted together.

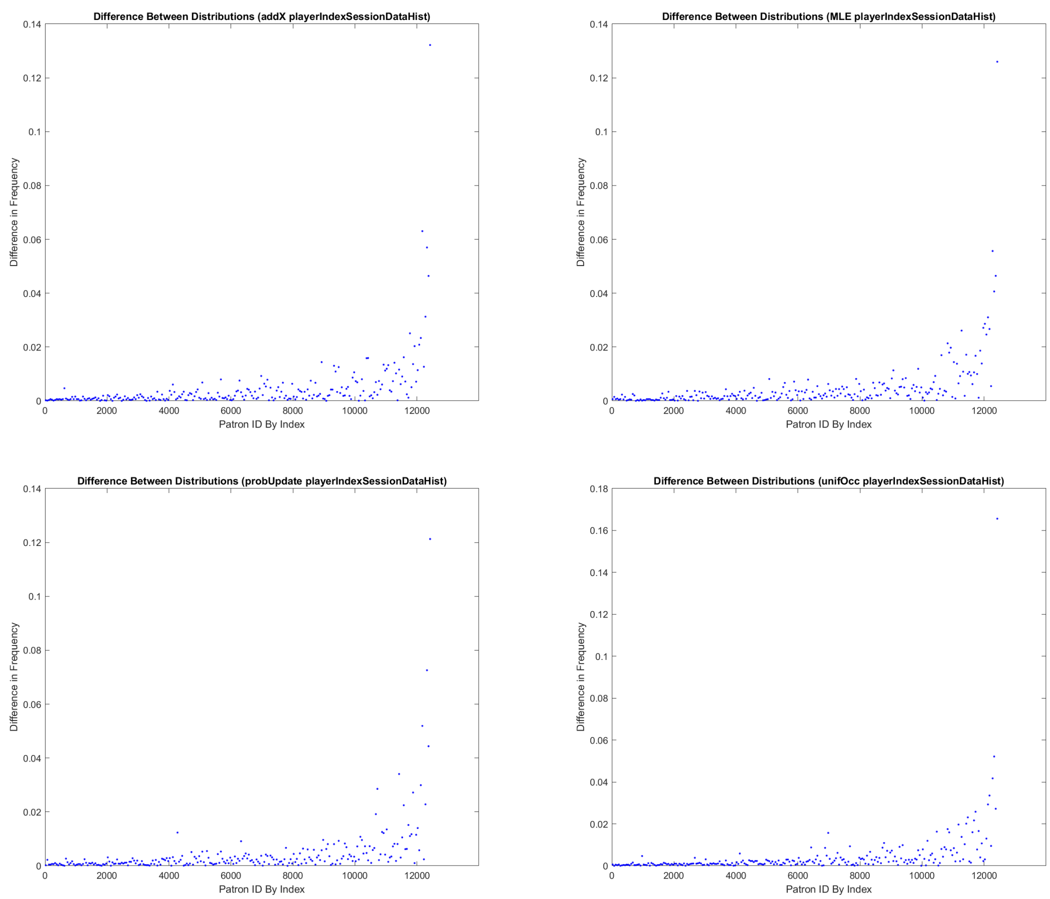

Figure 5.23: The difference in the relative frequency of player occurrences in session data between data generated using the given probability estimation method and the real data as a percentage of the maximum relative frequency found in the real data distribution. The players are assigned an index in order of their prevalence in the real data. This index is what is reported here.

series accurately as well.

It should be noted that the simulated data always displays a drop off in the number of events near the end of the simulation. These final events occur as players make their final moves at certain times into the future with some players completing their prescribed number of events and exiting the simulation sooner than others. If the simulation continued, more events would occur near these times, filling out the data set. For this reason, the final three days are removed from the simulated data.

5.3.1 Time Series Cycles

Event Data

The events, active machines, and players per day are plotted for real data and each generated data set in figures 5.24, 5.25, and 5.26. The daily numbers in the generated data contain an oscillatory structure as seen in the real data and hold steady throughout the data set for all generated data as they do in the real data for the majority of days. The oscillatory structure in the generated data does differ from the real data in the period of oscillations however.

The number of daily active machines in generated data is just as high as in the real data, showing that the simulation is utilizing the entire floor just as much as in the real data. The number of daily active players is lower in the generated data because of the low number of players allowed in the simulation ($J = 10$). In the higher occupancy generated data ($J = 100$), the number

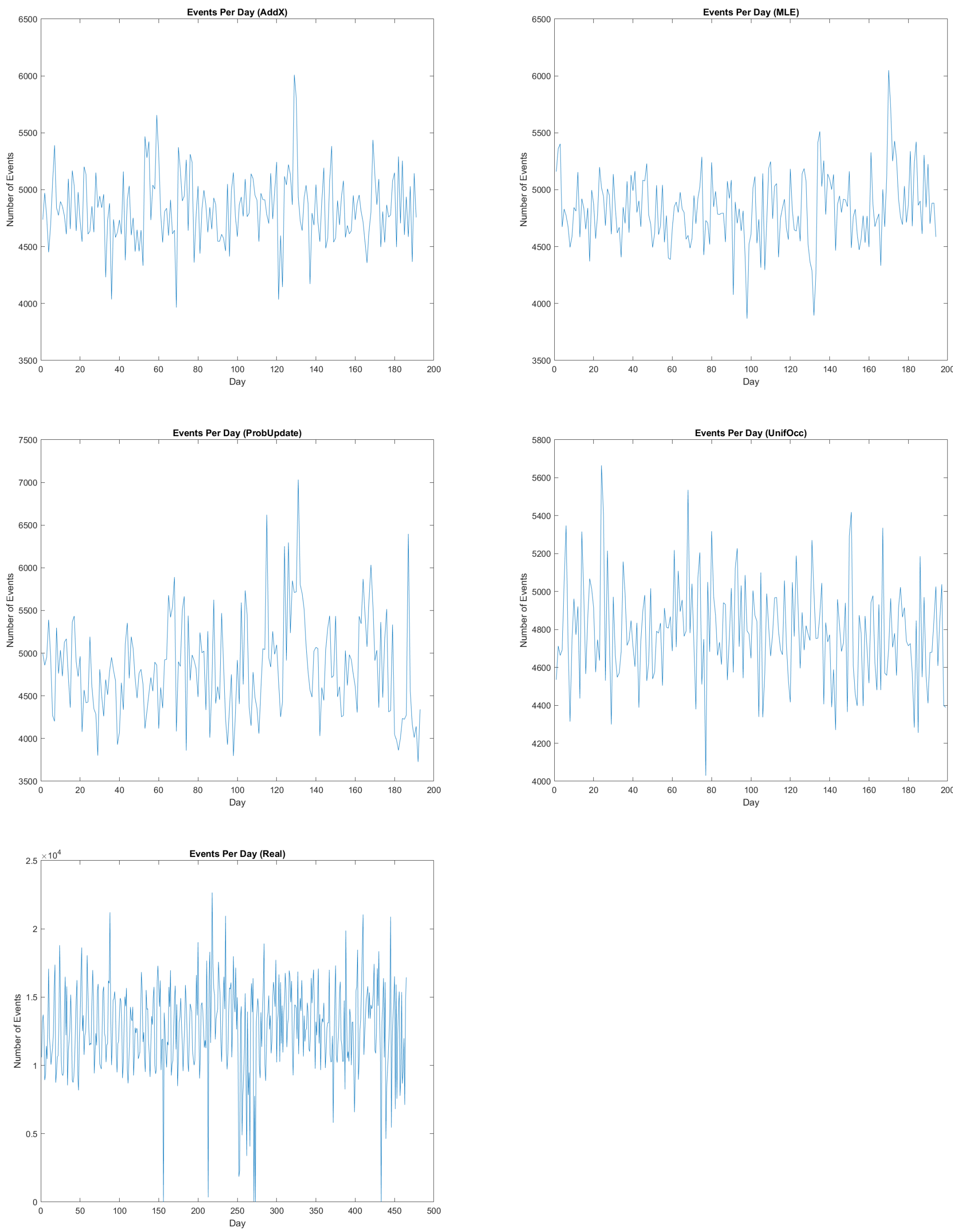

Figure 5.24: Plots of the number of events per day in each generated data set and the one real data set in chronological order.

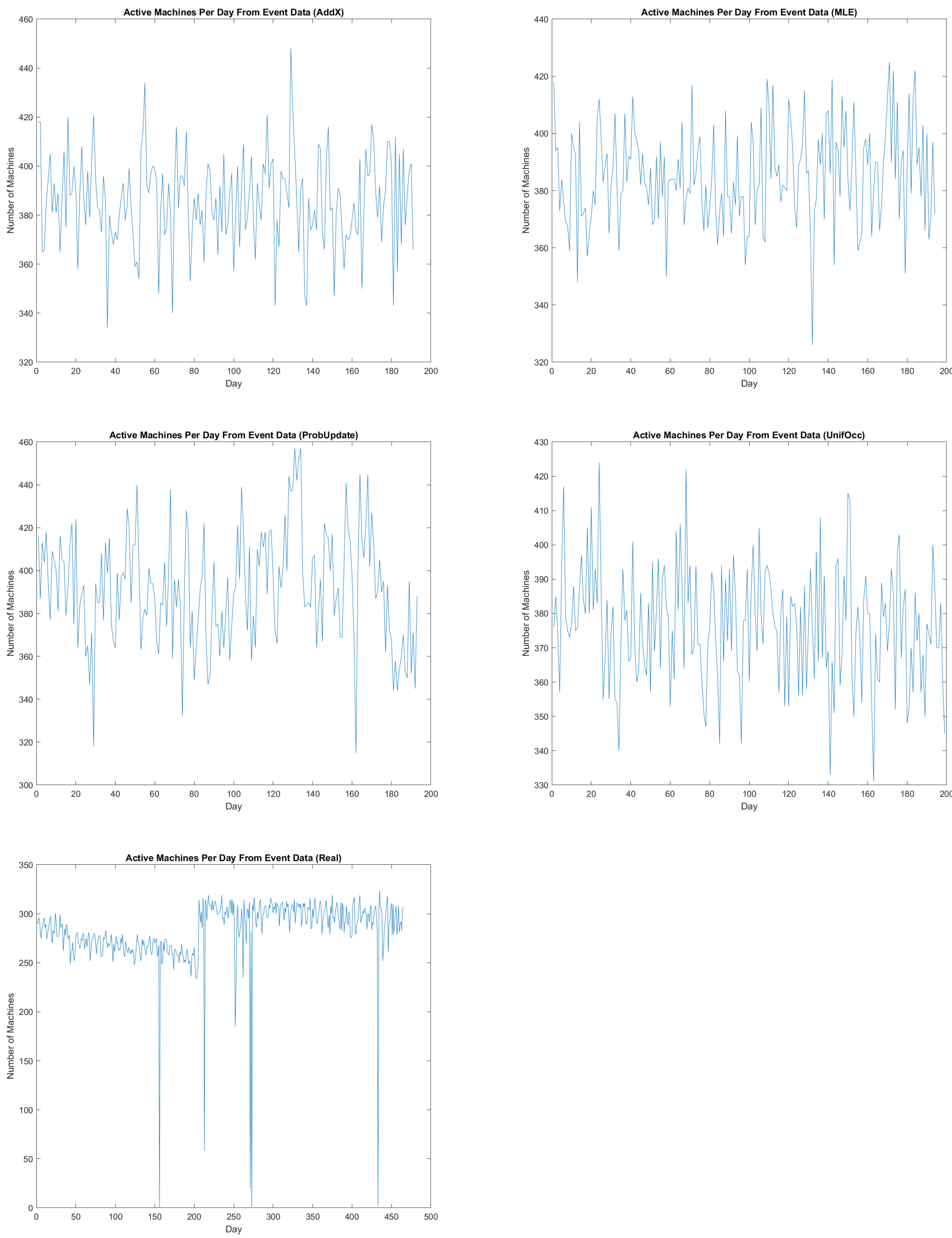

Figure 5.25: Plots of the number of active machines per day in each generated data set and the one real data set in chronological order.

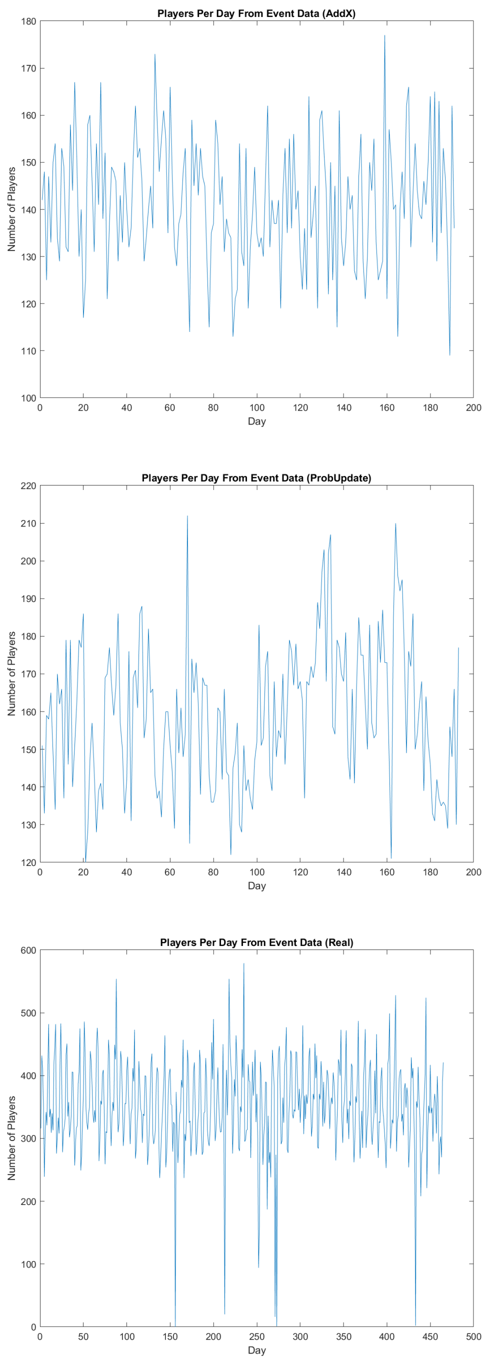

Figure 5.26: Plots of the number of unique player IDs per day in each generated data set and the one real data set in chronological order.

of daily active players is much higher than in the real data which is to be expected. A realistic simulation would require some trial and error to find the best setting for the J parameter.

A discrete Fourier transform using MATLAB's fast Fourier transform method was performed on the "events per day" signal found in each generated data set and the real data set. This was done for both occupancy levels ($J = 10$ and $J = 100$). The purpose of this experiment was to show how the frequency of oscillations differed in the generated and real data sets. The results are plotted in figures 5.27 and 5.28. The real data has its strongest frequency at $f = 0.15$ which is equivalent to a weekly period. The lower occupancy generated data ($J = 10$) does not exhibit a frequency of oscillations that is especially prevalent as compared to the others. In the higher occupancy generated data ($J = 100$), there are some frequencies that are more prevalent than others, but the structure of the frequency distribution still does not resemble the structure found in the real data. This demonstrates that a time-dependent method for controlling the number of players in the simulation is needed to more accurately model the time series found in the real data.

Session Data

Figures of the number of players and machines per day are not displayed here as they are going to be the same as in the event data. The conversion to sessions does not alter the number of unique players or machines observed in a

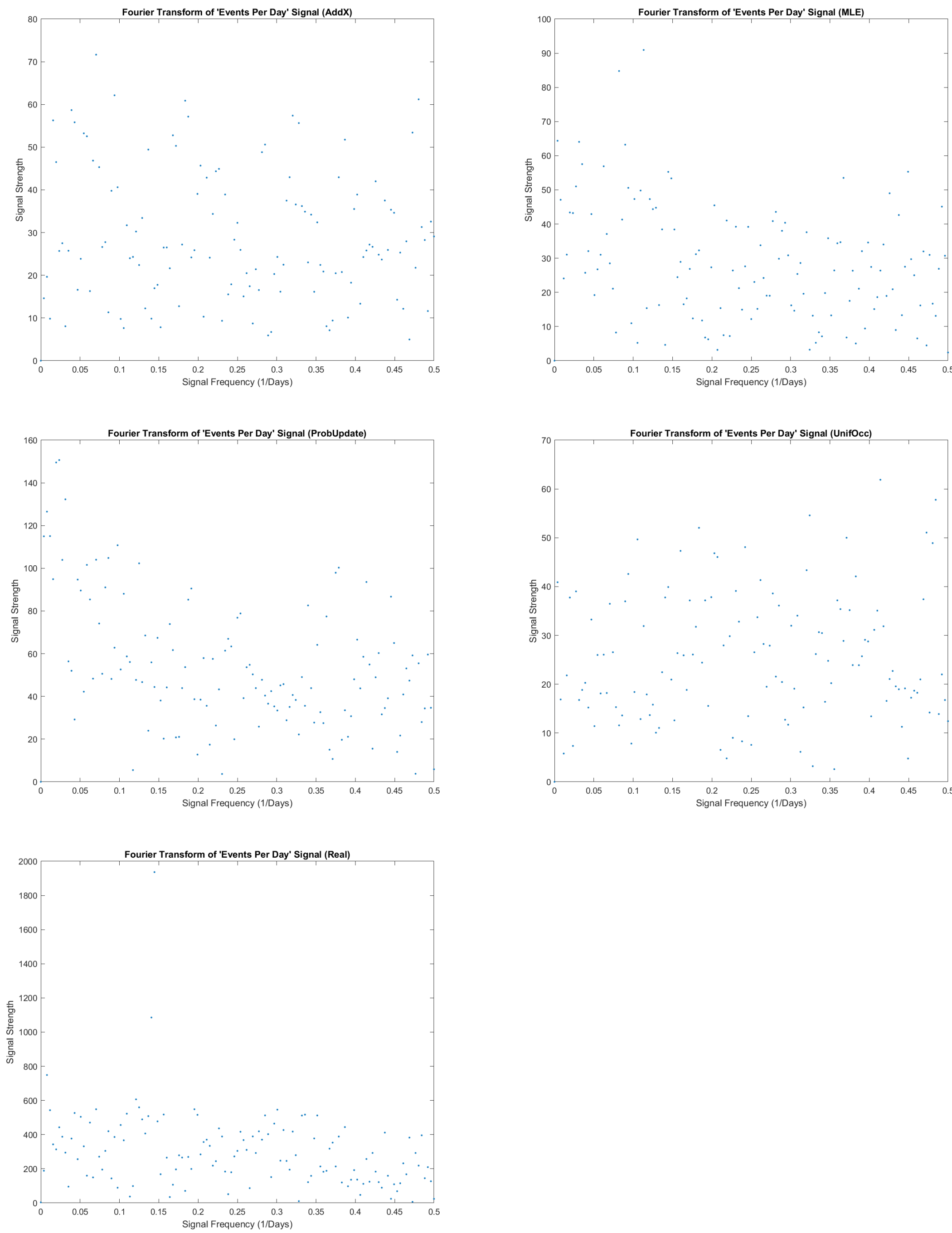

Figure 5.27: A discrete Fourier transform was performed on the events per day signal. The magnitude of the frequencies found in the signal is plotted for each generated data set ($J = 10$) and the real data.

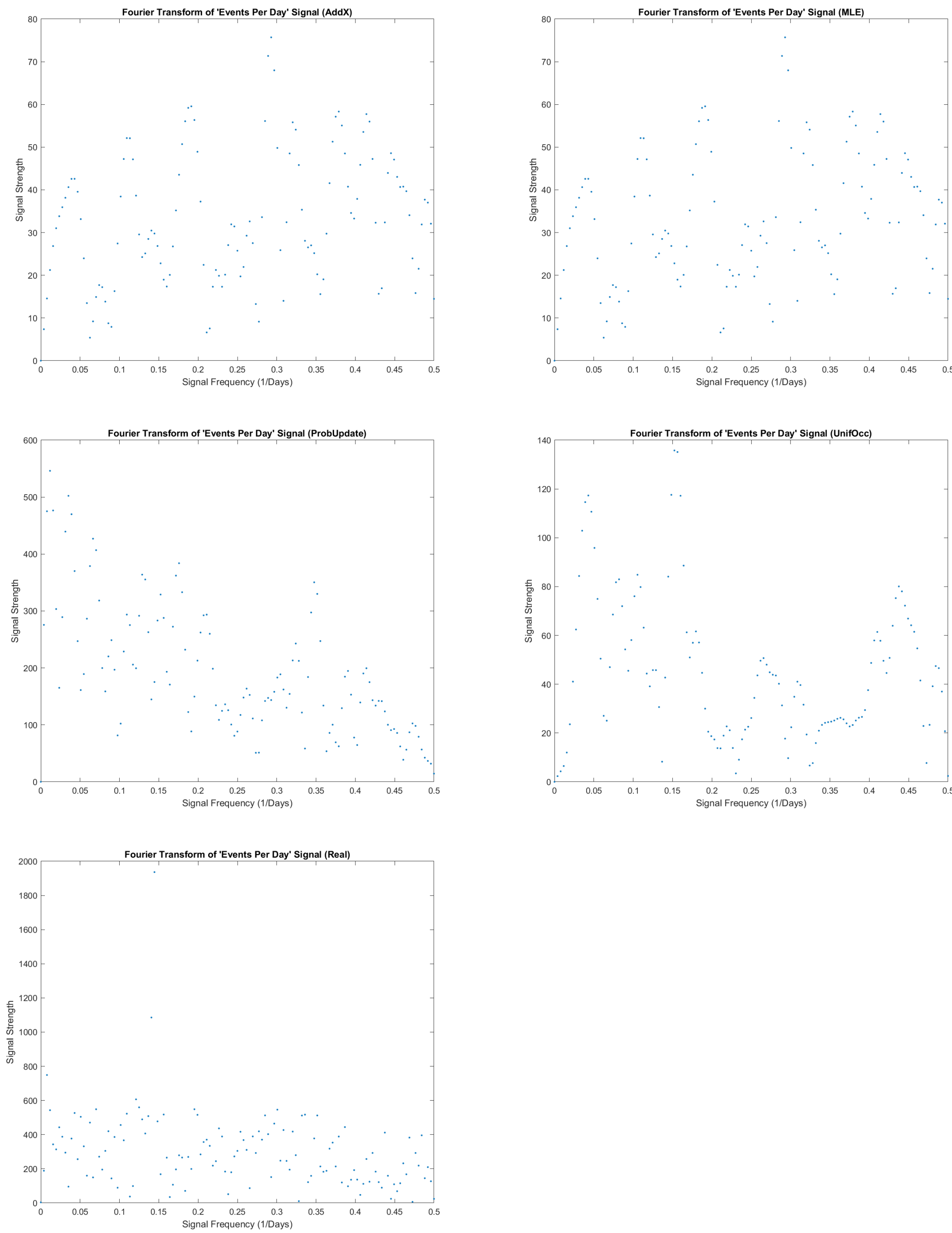

Figure 5.28: A discrete Fourier transform was performed on the events per day signal. The magnitude of the frequencies found in the signal is plotted for each generated data set ($J = 100$) and the real data.

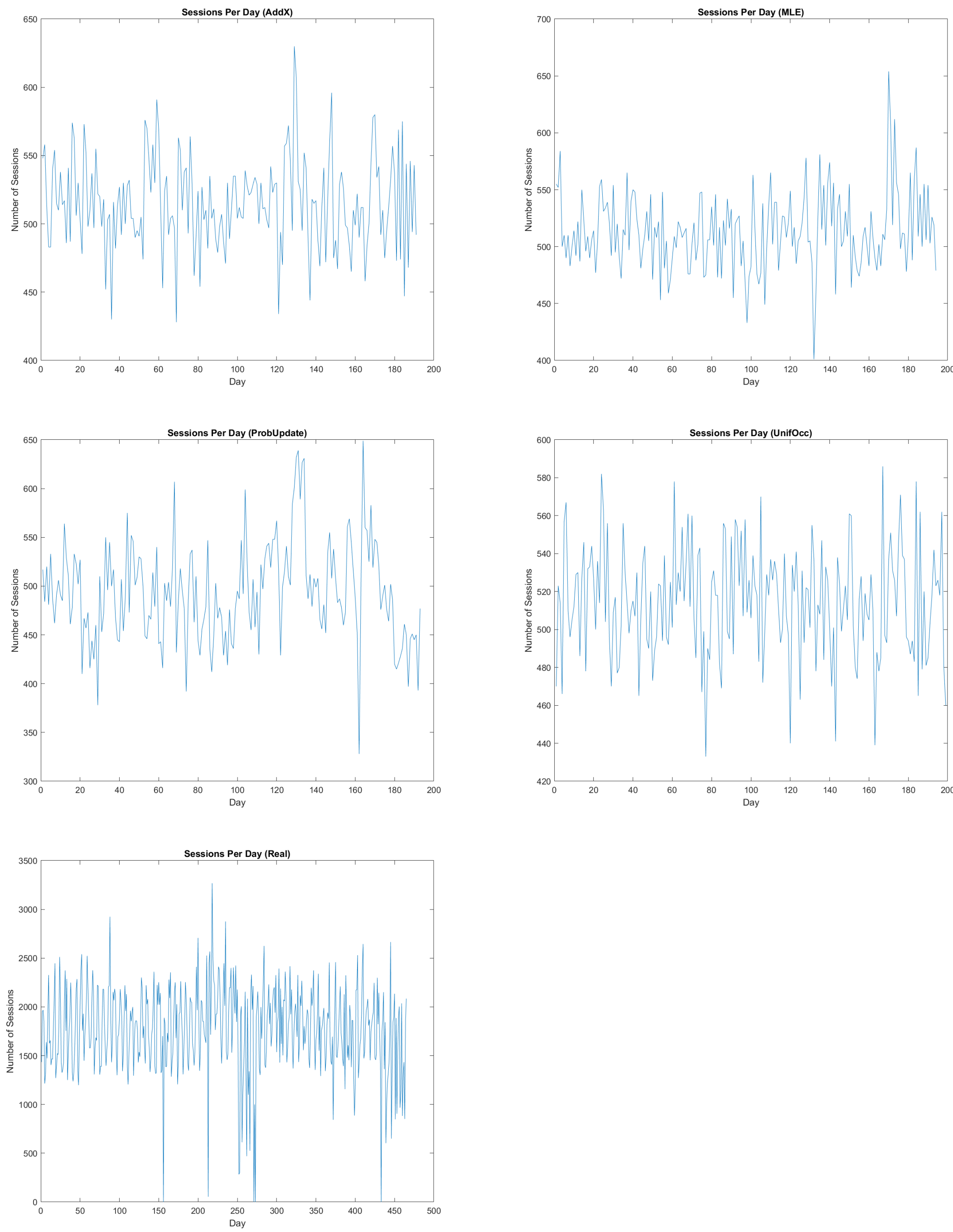

Figure 5.29: Plots of the number of sessions per day in each generated data set and the one real data set in chronological order.

given time span significantly.

The number of sessions per day (figure 5.29) again displays an oscillatory structure as in the real data, holding steady throughout the data set. This demonstrates that the event data consistently produces valid sessions.

5.3.2 Time Series Distributions

Event Data

In figure 5.30, the low occupancy ($J = 10$) generated data is shown to have a much lower density of events than the real data. Tables 5.16 and 5.17 are further evidence of this by tabulating the average number of events and sessions per day for both the low occupancy and higher occupancy generated data as compared to the real data.

In the generated data, there tends to be slightly more active machines per day than in the real data (figure 5.31). In reality, the casino may have often had a section of the floor closed off, resulting in a section of machines being unused most of the time. The simulation allows any machines that were used in the real data at any time to be played at all times.

The histograms shown in figure 5.32 are for the low occupancy generated data ($J = 10$) and so the number of players per day in the generated data tend to be much lower than in the real data. For the high occupancy data ($J = 100$), the number of players per day tends to be much higher than in the real data.

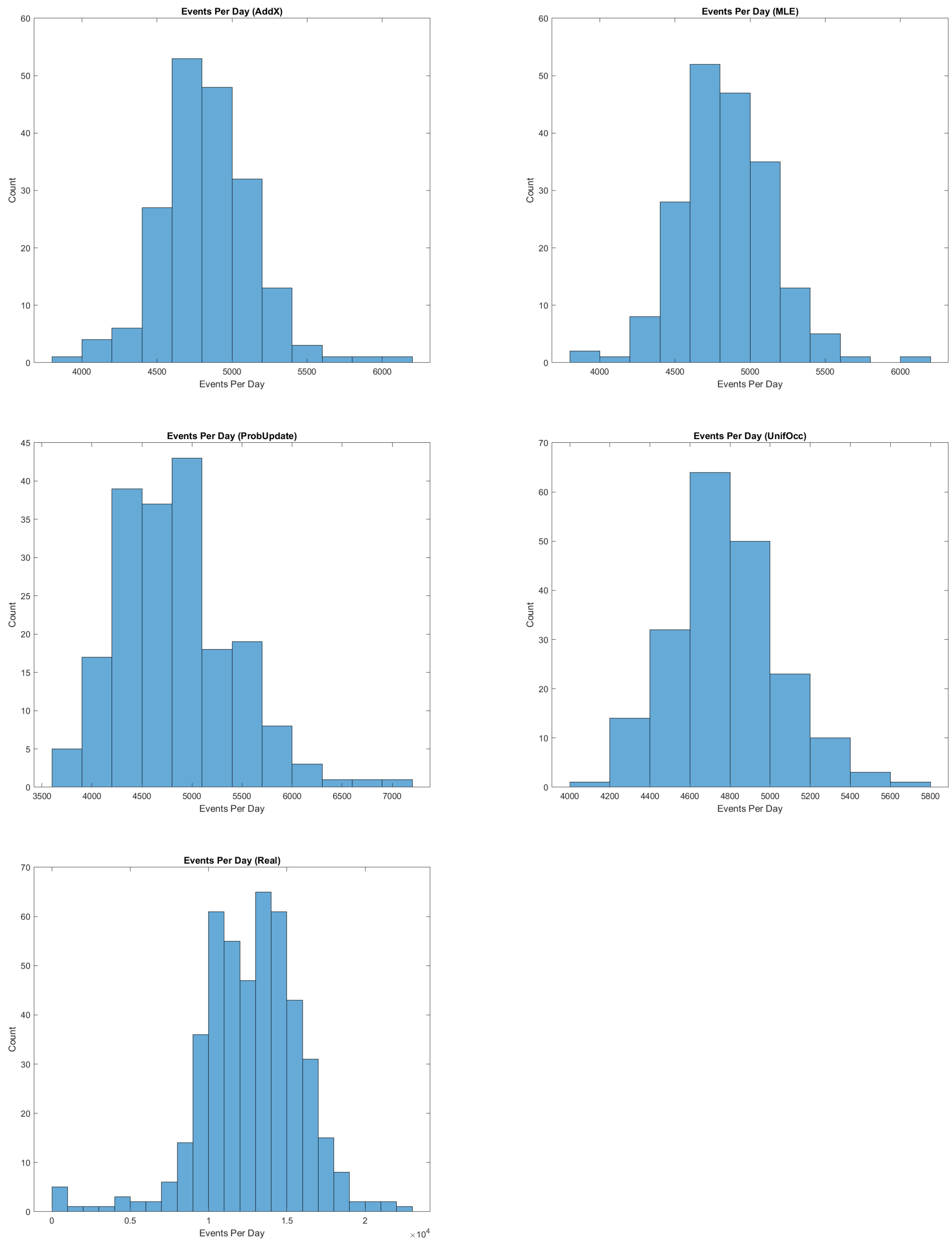

Figure 5.30: Histograms of the number of events per day in each generated data set and the one real data set.

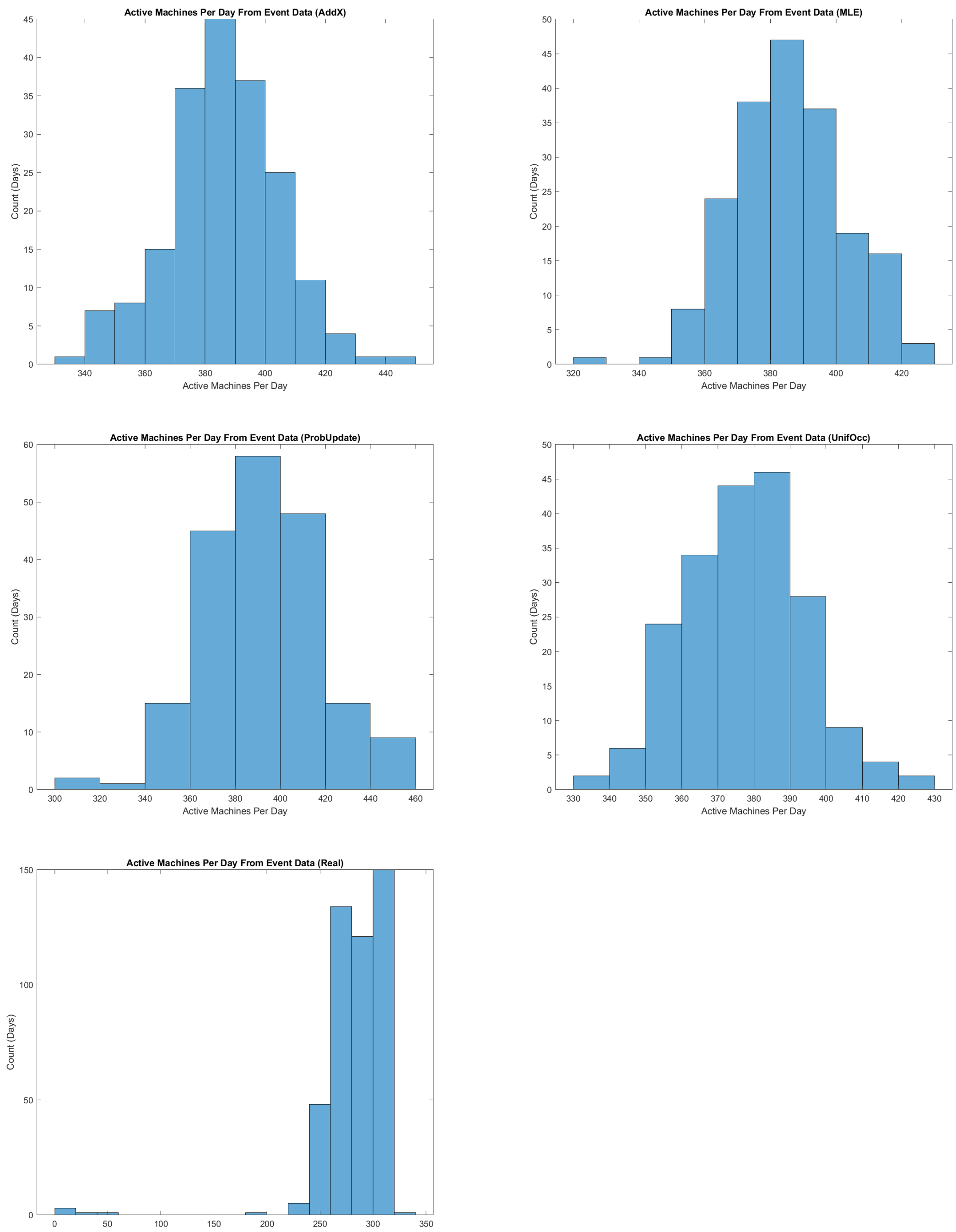

Figure 5.31: Histograms of the number of active machines per day in each generated data set and the one real data set.

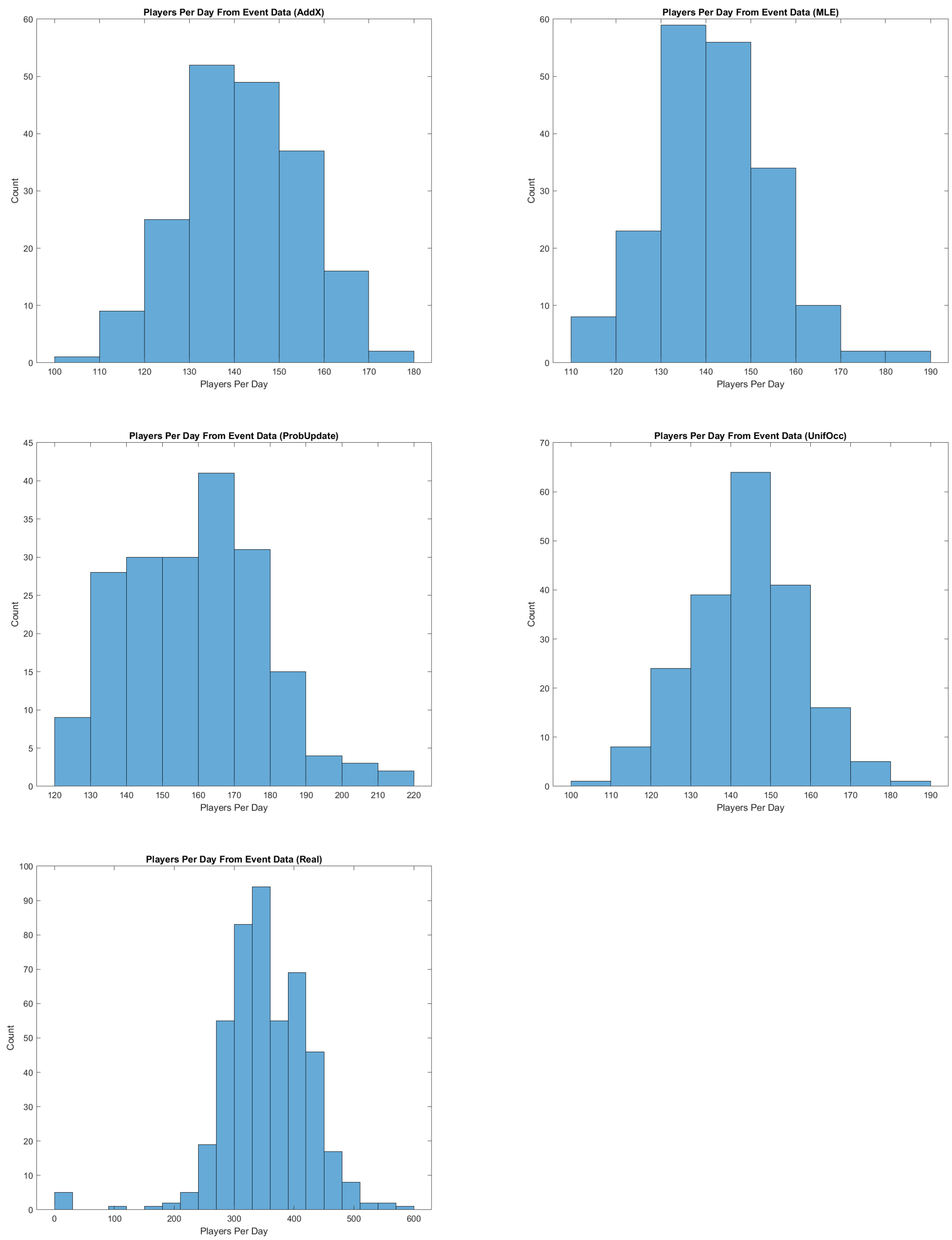

Figure 5.32: Histograms of the number of unique player IDs per day in each generated data set and the one real data set.

Session Data

The histograms of machines per day and players per day are not shown for the session data because the conversion to sessions does not alter these quantities greatly.

Similar to the events per day histograms in figure 5.30, figure 5.33 shows that the low occupancy generated data is much lower density than the real data. This is not a defect of the simulation, but simply a product of simulating a small number of players. This is discussed further in a later section.

Visits

Visits were used to determine the number of players present in the casino during each thirty minute time interval during the time span of each data set. This allowed for the study of the effect that the number of players allowed in the simulation had on the number of players actually in the casino.

In figure 5.34, it is clear that the distribution of players per interval in the generated data is very different from the distribution in the real data, making them difficult to compare. The spike we see in the distribution in the real data is likely due to the daily cycle present in the real data and represents the off hours of slot floor which are not present in the generated data. The imaginary slot floor which the generated data represents is always operating at peak hours.

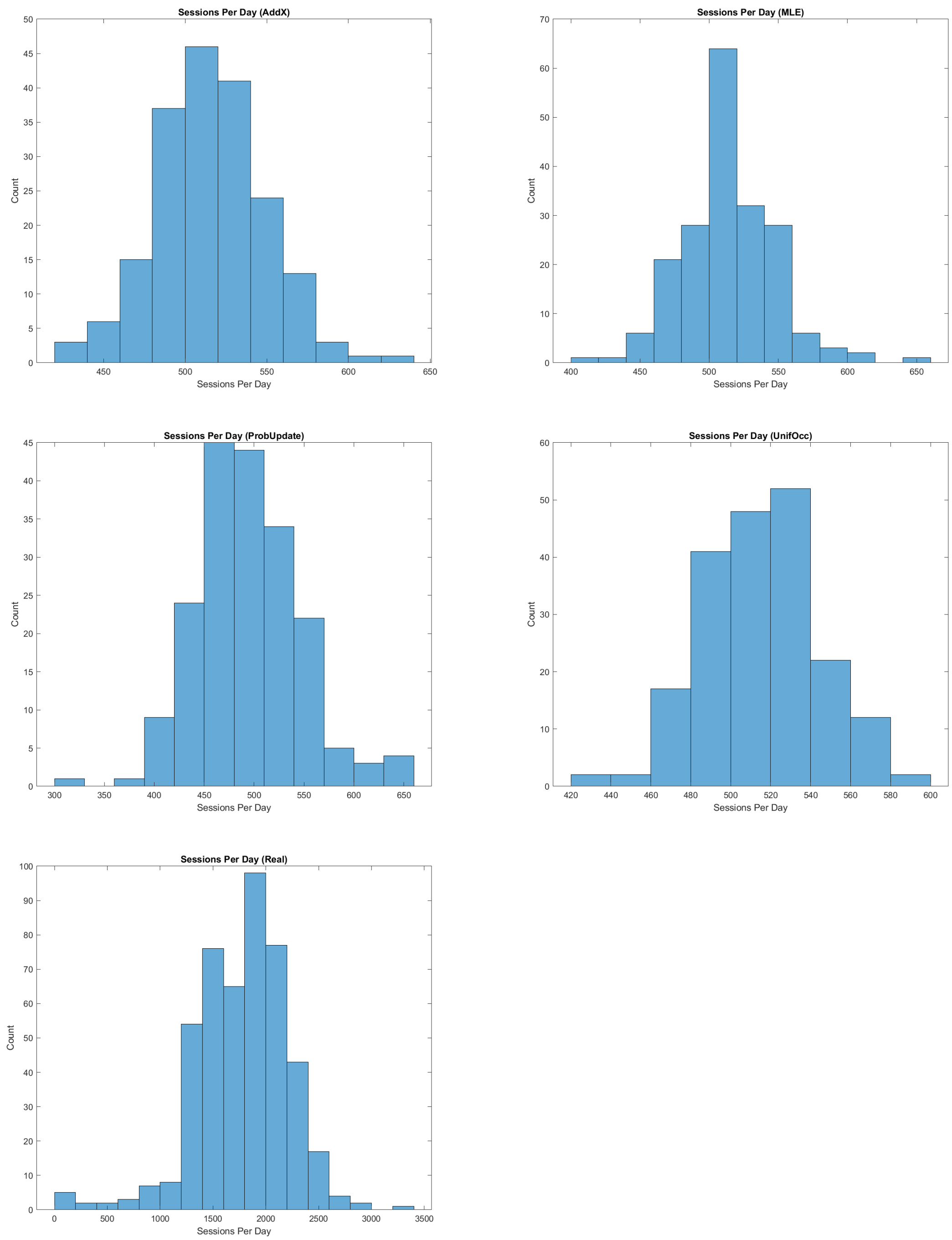

Figure 5.33: Histograms of the number of sessions per day in each generated data set and the one real data set.

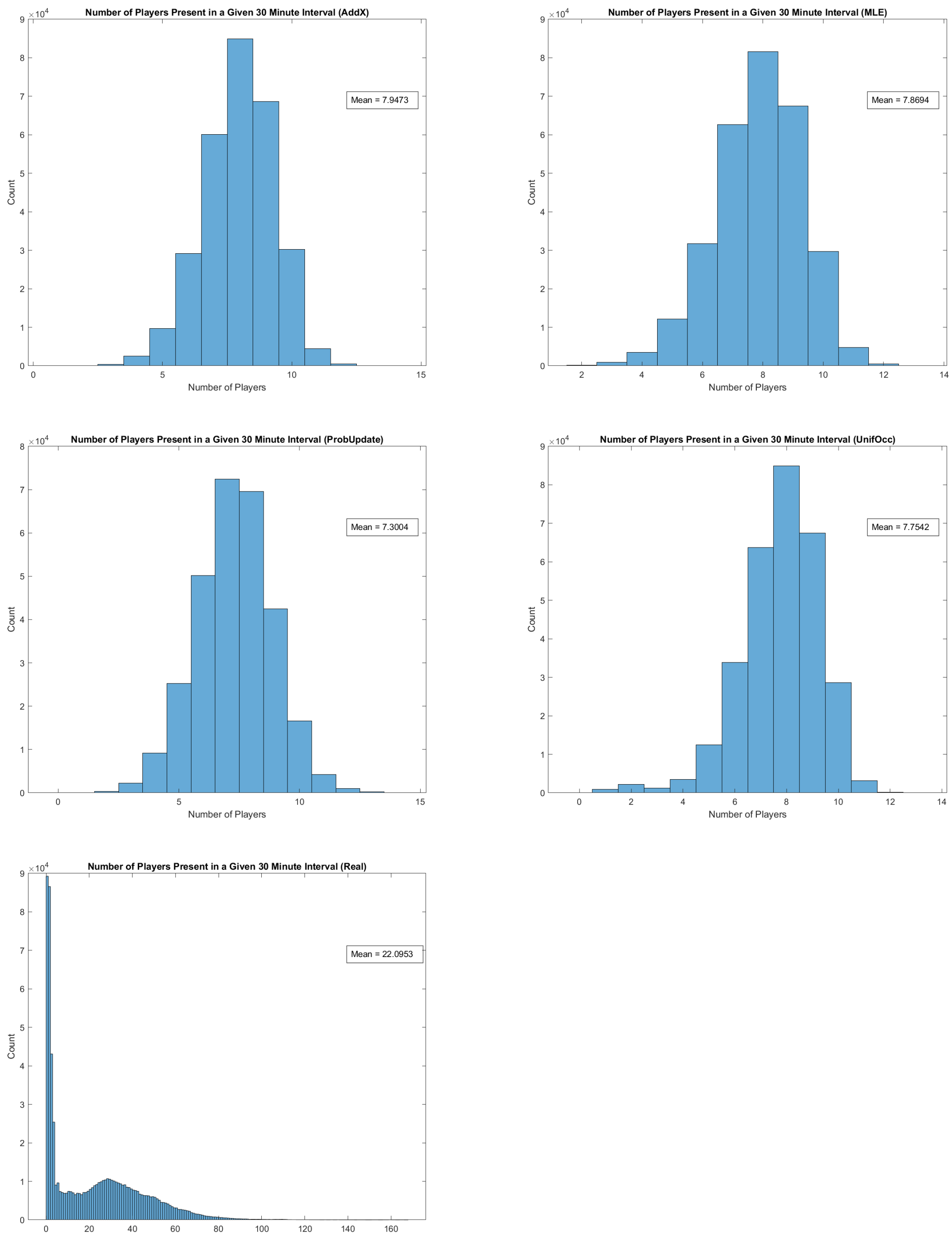

Figure 5.34: Histograms of the number of players per thirty minute interval in each generated data set and the one real data set.

Density

Even though the simulation with $J = 100$ is run for the same number of iterations, the resulting data sets are much shorter in time span. The higher the J parameter, the denser the resultant data set is. This is made evident by comparing the number of events, sessions, and visits per day for both generated data sets in tables 5.16 and 5.17. In these tables, the ratio of events/sessions/visits per day in generated data to real data are also displayed. These give an indication as to whether a change in the J parameter results in changes to the events/sessions/visits per day that are proportional, allowing for a choice of J that would result in generated data with densities that emulate the real data. The ratios for event and session data are close, but with a slightly wider gap in the higher occupancy data. The visits ratios indicate that way more visits are produced for the generated data than in the real data. This would imply that players are not playing multiple machines in the simulated data as often as they do in the real data.

To quantify this effect further, we can examine the distributions of the number of machines that players play in the visit for each data set. These distributions are plotted in figure 5.35. The distributions in generated data taper off near fifteen and have maximums in the range [30, 35] while the real data distribution tapers off around twenty, with a maximum near ninety. The means of these distributions for low occupancy and higher occupancy generated data and real data are tabulated in table 5.18. As indicated by the ratios in

Table 5.16: Comparing the length and time span of real and generated event data sets ($J = 10$). The ratios are equal to the number of events/sessions/visits per day in the generated data divided by the number in the real data.

	Real	Add-x	MLE	Prob Update	Unif Occ
Number of Events	15712274	977255	985182	981507	986418
Number of Sessions	1559364	105114	105654	101559	105654
Number of Visits	182817	29222	29623	32659	30132
Time Span In Days	490	202	205	204	210
Events Per Day	32066	4838	4806	4811	4697
Sessions Per Day	3182	520	515	498	503
Visits Per Day	373	145	145	160	143
Events Per Day Ratio	1.00	0.15	0.15	0.15	0.15
Sessions Per Day Ratio	1.00	0.16	0.16	0.16	0.16
Visits Per Day Ratio	1.00	0.39	0.39	0.43	0.39

Table 5.17: Comparing the length and time span of real and generated higher occupancy event data sets (J = 100). The ratios are equal to the number of events/sessions/visits per day in the generated data divided by the number in the real data.

	Real	Add-x	MLE	Prob Update	Unif Occ
Number of Events	15712274	2445454	2445454	2474857	2525915
Number of Sessions	1559364	218529	215152	215533	219561
Number of Visits	182817	68764	68764	75923	72261
Time Span In Days	490	51	51	55	53
Events Per Day	32066	47950	47950	44997	47659
Sessions Per Day	3182	4285	4219	3919	4143
Visits Per Day	373	1348	1348	1399	1363
Events Per Day Ratio	1.0	1.5	1.5	1.4	1.5
Sessions Per Day Ratio	1.0	1.3	1.3	1.2	1.3
Visits Per Day Ratio	1.0	3.6	3.5	3.8	3.7

Table 5.18: Means of the machines played per visit distributions in all data sets.

	Real	Add-x	MLE	Prob Update	Unif Occ
Low Occupancy ($J = 10$)	4.09	3.33	3.31	2.93	3.31
High Occupancy ($J = 100$)	4.09	2.93	2.93	2.71	2.86

table 5.16 and 5.17, the real data has the highest mean number of machines played per visit. The larger the J parameter, the larger the disparity between the means, which is likely caused by increased collisions due to the higher occupancy.

5.4 Summary

When evaluating how well each probability matrix estimation method is able to estimate probability distributions in data, we saw that the *Uniform Occupancy* method made little to no improvement over the other MLE-based methods (Add-x and the MLE). While the *Probability Update* method consistently performed the best at approximating session data distributions, it also consistently performed the worst at approximating event data distributions. This along with evidence provided by the machine occurrences distributions implies that the *Probability Update* method excels at producing a possible new version of the real event data, while the MLE-based methods are better at replicating the real event data. The *Probability Update* method appears to succeed at learning true player preferences as if they had the choice of playing any machine on the slot floor. For the purposes of creating unidentifiable and realistic synthetic

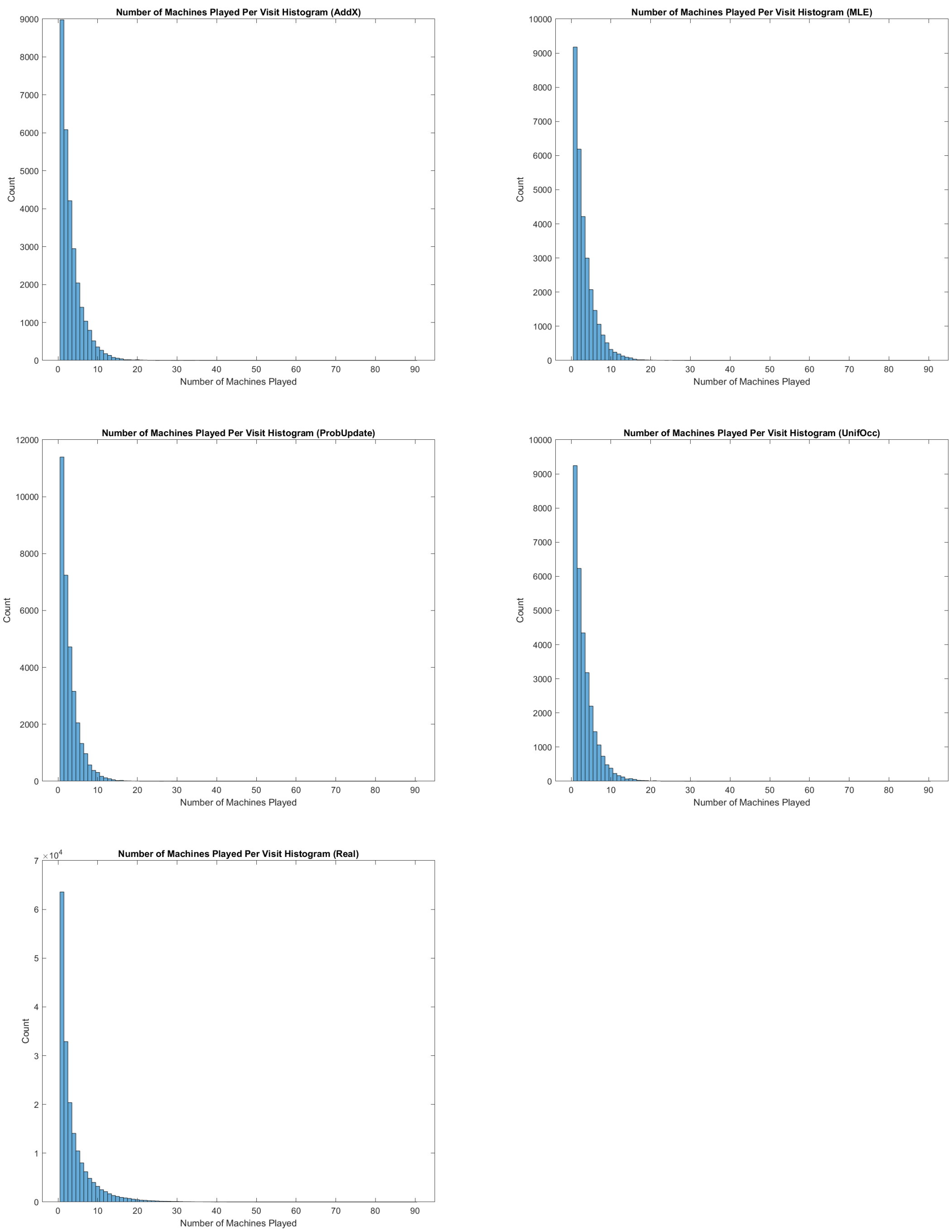

Figure 5.35: Histograms of the number of machines played per visit in lower occupancy ($J = 10$) generated data and real data.

data, it appears to be the superior method.

The simulation, regardless of probability matrix estimation method used, succeeded at producing a consistent level of sessions and events. However, the simulation did not succeed at emulating the weekly cycle normally found in a real data time series or in producing a level of visits that was proportional to the number of events produced. Simulated players do not play as many machines in a visit as they do in the real data. Further work is required to resolve these deficiencies.

Chapter 6

Conclusions

Two new probability matrix estimation methods for use as part of Markov processes where many individuals occupy the process simultaneously, but no more than one individual may occupy a state at a time were proposed. New methods were needed that took into account the occupancy of states at the time of a transition so that probabilities were as undistorted as possible due to the players having limited choices.

The first new method proposed was the Uniform Occupancy method, which assumed that the probability of a state being unoccupied given that another state was occupied is constant throughout the entire data set. The other new method proposed was the Probability Update method, which made no assumptions, but relied on players in the source data set having choices and consequently required there to be periods of time where occupancy was low enough to allow for those choices. A Bayesian method was discussed, but it was determined that an accurate likelihood function would be too unwieldy in practice.

The two new probability matrix estimation methods proposed were compared to two benchmark methods: the maximum likelihood estimator (MLE) and the Add-x estimator. Neither of these estimators take into account occupancy in any way so they were good benchmarks to measure how well the newly proposed methods accounted for it. The Probability Update and Uniform Occupancy methods showed good convergence to the true probability matrix in test scenarios as compared to the benchmark methods. The Probability Update method showed poor convergence as compared to the Uniform Occupancy method in test scenarios when occupancy remained steadily at or above sixty percent. The Uniform Occupancy method did not converge for one hundred percent occupancy as this results in a division by zero. The Uniform Occupancy method did not benefit greatly from additional data points past about 1e6 data points, but at any occupancy level the Probability Update method continued to reduce error up until the 1e9 data points that were tested.

When simulating real data, the Probability Update method approximated distributions of session level quantities better than any other method, suggesting that it modeled player behaviour the best. The benchmark methods and the Uniform Occupancy method performed very similarly in modeling distributions of event and session level quantities. These other methods modeled event level quantities better than the Probability Update method did, suggesting that the Probability Update method allowed for more variance in player choices that still resulted in sessions that best resembled real data sessions. The simulation failed to model the games per minute distribution accurately and only managed to model the location of the peak of the distribution. The games per minute distribution in generated data did not concentrate its probability enough at the mean like in the real data where the standard deviation is very small. This was the one compound session level quantity that was tested that was a function of two session level quantities so the failure to model it accurately was attributed to the level of abstraction it held.

When fitting Gaussian and Skew Normal distributions to some session level quantities including coin-in per session and session duration, it was found that the real data distributions were modelled well by a Gaussian distribution, whereas modelling of generated data distributions was improved slightly by using a Skew Normal distribution.

The simulation suffered in its modelling of the data time series. While the simulation resulted in an oscillatory time series that holds steady throughout just as in the real data, the period of oscillations was on average significantly shorter than in the real data. Additionally, the density of sessions and events was proportional to the number of players in the simulation (the J parameter), but the density of player visits was much higher than it should be in the generated data. This indicated that players were on average not visiting as many machines as they did in the real data and that the probability matrices did not allow for inter-machine transitions as often as they should.

6.1 Limitations

The uniformity of the meter units across all machines is not guaranteed due to the lack of denomination or unit information in the data. It was inferred that they most likely are the same for every machine, but the possibility that they are not potentially compromises the fidelity of the distributions of coin-in, coin-out and games played per event collected from the real data. However, the goal of this work was to simulate the real data, so it is likely that if we are able to simulate this data set well, we could also simulate a data set with uniform meter units just as well.

The methods of simulation discussed in this work only apply to carded players. It is not possible to simulate uncarded players in the same way due to the inability to track them.

The current computational ability of the simulation is lacking in that its computational complexity is proportional to the number of players. This means that simulating a larger, more realistic number of players is infeasible in the amount of time it would take to do.

It was discussed how studies that use old data often state this as a limitation. Even though the data used in this study is several years old, it could be argued that this is in fact not a limitation as if old data can be simulated well, newer data should not require entirely new methods to simulate.

6.2 Future Work

As is a common limitation of data research, this work could be greatly improved upon by insights that would be offered by multiple data sets from a variety of casinos. The statistical variation in distributions from multiple data sets would help to inform on the amount that generated data distributions should differ from their source data set in order to guarantee full anonymity.

The fidelity of the generated time series could be improved upon by introducing a dynamic player replacement scheme. The biggest reason for the poorly approximated time series is likely that the simulation aims to keep the number of players in the casino constant at all times, which is not realistic.

The simulation currently treats all players as unique individuals instead of as members of a group. If the simulation were capable of handling a player segmentation, it would become a much more versatile tool for studying how user-defined groups of players behave on the floor, or how they would behave if changes were made.

Some work was done to determine the effect that the J parameter has on the resultant generated data set, but with only two data points it is difficult to make too many inferences. In future, it would be beneficial to generate data for a larger variety of settings for the J parameter. This could be made easier by parallelizing the simulation to split up the workload among multiple cores.

Theoretically, these simulations could be used to study how changes to the slot floor, including different mixes of machines, would affect players and profits, but this was not studied as part of this work. These capabilities must be studied in more depth in order to determine the extent of their worth as opposed to other methods of customer behaviour prediction.

Bibliography

[1] W. R. Eadington, *Gambling and Society: Interdisciplinary Studies on the Subject of Gambling*. Thomas, 1976. (Cited on page 2.)

[2] B. King, "Forecasting casino gaming traffic with a data mining alternative to croston's method," *UNLV's Gaming Research and Review Journal*, vol. 20(2), Nov 2016. (Cited on pages 3 and 14.)

[3] G. D. I. Barr and I. N. Durbach, "A monte carlo analysis of hypothetical multi-line slot machine play," *International Gambling Studies*, vol. 8(3), Oct 2008. (Cited on page 3.)

[4] A. F. Lucas and A. K. Singh, "The house edge and play time: Do industry heuristics fairly describe this relationship?," *UNLV Gaming Research and Review Journal*, vol. 25(1), Sept 2021. (Cited on pages 3, 13 and 14.)

[5] E. Suh and M. Alhaery, "Customer retention: Reducing online casino player churn through the application of predictive modeling," *UNLV's*

Gaming Research and Review Journal, vol. 20(2), Nov 2016. (Cited on pages 3, 4 and 15.)

[6] R. Iaci and A. K. Singh, "Clustering high dimensional sparse casino player tracking datasets," *UNLV's Gaming Research and Review Journal*, vol. 16(1), Nov 2012. (Cited on pages 3 and 16.)

[7] E. Suh and M. Alhaery, "Predicting cross-gaming propensity using e-chaid analysis," *UNLV's Gaming Research and Review Journal*, vol. 19(1), June 2015. (Cited on pages 3, 4 and 15.)

[8] A. F. Lucas, "Exploring the relationship between race and sports book wagering activity and daily slot and table game play," *UNLV's Gaming Research and Review Journal*, vol. 18(2), Dec 2014. (Cited on pages 3, 4 and 15.)

[9] A. F. Lucas, A. K. Singh, and L. P. Gewali, "Simulating the effect of pay table standard deviation on pulls per losing player at the single-visit level," *UNLV Gaming Research and Review Journal*, vol. 11(1), Dec 2012. (Cited on pages 3, 13 and 14.)

[10] A. F. Lucas and K. D. Brandmeir, "Estimating the short-term effects of an increase in par on reel slot performance," *UNLV Gaming Research and Review Journal*, vol. 9(2), 2007. (Cited on pages 3, 4 and 25.)

[11] B. L. Abarbanel, A. F. Lucas, and A. K. Singh, "Estimating the indirect

effect of sports books on other in-house gaming volumes," *UNLV Gaming Research and Review Journal*, vol. 15(2), 2011. (Cited on pages 3, 4 and 15.)

[12] A. F. Lucas and W. T. Dunn, "Estimating the effects of micro-location variables and game characteristics on slot machine volume: A performance-potential model," *Journal of Hospitality and Tourism Research*, vol. 29(2), May 2005. (Cited on pages 3, 4 and 15.)

[13] A. F. Lucas, W. T. Dunn, and A. K. Singh, "Estimating the short-term effect of free-play offers in a las vegas hotel casino," *Journal of Travel and Tourism Marketing*, vol. 18(2), Oct 2008. (Cited on pages 3, 4 and 16.)

[14] E. Suh and M. Alhaery, "Maximizing player value through the application of cross-gaming predictive models," *International Journal of Contemporary Hospitality Management*, vol. 26(8), 2015. (Cited on pages 3, 4 and 15.)

[15] S. Dragicevic, G. Tsogas, and A. Kudic, "Analysis of casino online gambling data in relation to behavioural risk markers for high-risk gambling and player protection," *International Gambling Studies*, vol. 11(3), Dec 2011. (Cited on pages 3 and 4.)

[16] J. Meng and F. Fu, "Understanding gambling behaviour and risk attitudes using cryptocurrency-based casino blockchain data," *Royal Society Open Science*, vol. 7(10), Oct 2020. (Cited on page 3.)

[17] W. T. Dunn, *Slot Performance Analysis*. Dunn Gaming Solutions, 2014. (Cited on page 8.)

[18] R. Serfozo, *Basics of Applied Stochastic Processes*. Springer, 2009. (Cited on pages 10, 11 and 49.)

[19] N. Privault, *Understanding Markov Chains*. Springer, 2018. (Cited on pages 10, 11 and 49.)

[20] T. C. Lee, G. G. Judge, and A. Zellner, "Maximum likelihood and bayesian estimation of transition probabilities," *Journal of the American Statistical Association*, vol. 63(324), pp. 1162–1179, Dec 1968. (Cited on pages 16, 17, 42 and 50.)

[21] J. D. Kalbfleisch and J. F. Lawless, "Least-squares estimation of transition probabilities from aggregate data," *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, vol. 12(3), pp. 169–182, Sept 1984. (Cited on page 16.)

[22] Y. Hao, A. Orlitsky, and V. Pichapati, "On learning markov chains," *32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, Dec 2018. (Cited on pages 17, 46 and 53.)

[23] M. Al-Alawi, A. Bouferguene, and Y. Mohamed, "Random generation of industrial pipelines' data using markov chain model," *Advanced Engineering Informatics*, vol. 38, pp. 725–745, Oct 2018. (Cited on page 18.)

[24] C. Sherlaw-Johnson, S. Gallivan, and J. Burridge, "Estimating a markov transition matrix from observational data," *The Journal of the Operational Research Society*, vol. 46(3), pp. 405–410, Mar 1995. (Cited on page 18.)

[25] Z. Song, X. Geng, A. Kusiak, and C. Xu, "Mining markov chain transition matrix from wind speed time series data," *Expert Systems with Applications*, vol. 38, pp. 10229–10239, Aug 2011. (Cited on page 18.)

[26] H. Nfaoui, H. Essiarab, and A. Sayigh, "A stochastic markov chain model for simulating wind speed time series at tangiers, morocco," *Renewable Energy*, vol. 29(8), pp. 1407–1418, Jul 2004. (Cited on page 18.)

[27] D. Mizutani, N. Lethanh, B. T. Adey, and K. Kaito, "Improving the estimation of markov transition probabilities using mechanistic-empirical models," *Frontiers in Built Environment*, vol. 3(58), Oct 2017. (Cited on page 19.)

[28] N. Lethanh, J. Hackl, and B. T. Adey, "Determination of markov transition probabilities to be used in bridge management from mechanistic-empirical models," *Journal of Bridge Engineering*, vol. 22(10), Jul 2017. (Cited on page 19.)

[29] H. S. C. D. Manning, P. Raghaven, *Introduction to Information Retrieval*. Cambridge University Press, 2008. (Cited on page 46.)

[30] D. Valcarce, J. Parapar, and A. Barreiro, "Additive smoothing for

relevance-based language modelling of recommender systems," *CERI '16: Proceedings of the 4th Spanish Conference on Information Retrieval*, June 2016. (Cited on page 46.)

[31] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information Theory*, vol. 37(1), Jan 1991. (Cited on page 69.)

[32] F. Nielsen, "On a generalization of the jensen–shannon divergence and the jensen–shannon centroid," *Entropy (Basel, Switzerland)*, vol. 22(2), Feb 2020. (Cited on page 69.)