ESTIMATING ERROR AND BIAS OF OFFLINE RECOMMENDER SYSTEM EVALUATION RESULTS

by

Mucun Tian

A thesis

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Computer Science

Boise State University

August 2019

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Mucun Tian

Thesis Title: Estimating Error and Bias of Offline Recommender System Evaluation Results

Date of Final Oral Examination: 31 May 2019

The following individuals read and discussed the thesis submitted by student Mucun Tian, and they evaluated his presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Michael D. Ekstrand, Ph.D.	Chair, Supervisory Committee
Maria Soledad Pera, Ph.D.	Member, Supervisory Committee
Hoda Mehrpouyan, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Michael D. Ekstrand, Ph.D., Chair of the Supervisory Committee. The thesis was approved by the Graduate College.

DEDICATION

Dedicated to my advisor, mentor, Dr. Michael D. Ekstrand.

ACKNOWLEDGEMENTS

Thanks for all people who helped me. Special thanks to Dr. Michael D. Ekstrand for his guidance to my thesis work and professional development. I am also grateful to my committee, Dr. Maria Soledad Pera and Dr. Hoda Mehrpouyan, for the feedback and encouragement. This thesis also benefits from the excellent teaching of probability theory and computational statistics by Dr. John Chiasson and Dr. Leming Qu. Thank you to my fellow students in PIReT for a great research environment.

I would like to extend my gratitude to my wife, parents, and older sister for their love and support.

This work was facilitated by the R2 cluster provided by Boise State University's Research Computing Department [1]. This material is based upon work supported by the National Science Foundation under Grant No. IIS 17-51278.

ABSTRACT

Recommender systems are software applications deployed on the Internet to help people find useful items (e.g. movies, books, music, products) by providing recommendation lists. Before deploying recommender systems online, researchers and practitioners generally conduct offline evaluations to compare the accuracy of top-N recommendation lists among candidate algorithms using users' history consumption data. These offline evaluations typically use metrics and methodologies borrowed from machine learning and information retrieval and have several well-known biases that affect the validity of their results, including popularity bias and other biases arising from the missing-not-at-random nature of the data used. The existence of these biases is well-established, but their extent and impact are not as well-studied. In this work, we employ controlled simulations with varying assumptions about the distribution and structure of users' preferences and the rating process to estimate the distributions of the errors in recommender experiment outcomes as a result of these biases. We calibrate our simulated datasets to mimic key statistics of existing public datasets in different domains and use the simulated data to assess the error in estimating true accuracy with observable rating data. We find inconsistency of the evaluation metric scores and the order in which they rank recommendation algorithms in the synthetic true preference and the observation dataset. Simulation results show that offline evaluations are sometimes fooled by intrinsic effects in the data generation process into mistakenly ranking algorithms. The extent of these effects is sensitive to assumptions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

IBP	Indian Buffet Process
IR	Information Retrieval
LDA	Latent Dirichlet Allocation
MAP	Mean Average Precision
MAR	Missing at Random
MNAR	Missing not at Random
MRR	Mean Reciprocal Rank
nDCG	normalized Discounted Cumulative Gain
TREC	Text Retrieval Conference

CHAPTER ONE: INTRODUCTION

Recommender systems have long been used to reduce information overload people are facing on the Internet by presenting them with recommendation lists of interesting items (e.g. movies, books, music, and products). These systems learn overall or individual user tastes from data sets collected through user-item interactions (e.g. ratings or purchases). For example, if a user gives a rating of “like” to action movies, then a system may recommend more action movies to the user.

To evaluate the effectiveness of a proposed new recommender system, the standard method is to conduct an online evaluation using an A/B test. An A/B test randomly splits users into two test groups, exposes each group of users to a different recommender, and monitors users’ response to each recommender; it then compares metrics that are relevant to the business’s goals such as sales volume. Online evaluation directly measures business goals and users’ satisfaction about recommended items, but it is costly and time-consuming: a new recommender may hurt the experience of users who are used to the old system, and it also takes months to develop and test. Further, for academic researchers, much research can only be done offline since accessing a large number of users for online experiments is not always feasible. Even in commercial settings that perform A/B tests, an offline evaluation is often done before an online evaluation to first gain confidence with a new algorithm.

However, the data sets collected for offline training and evaluating recommender systems are very sparse (i.e. relevance information about most items is missing), and the

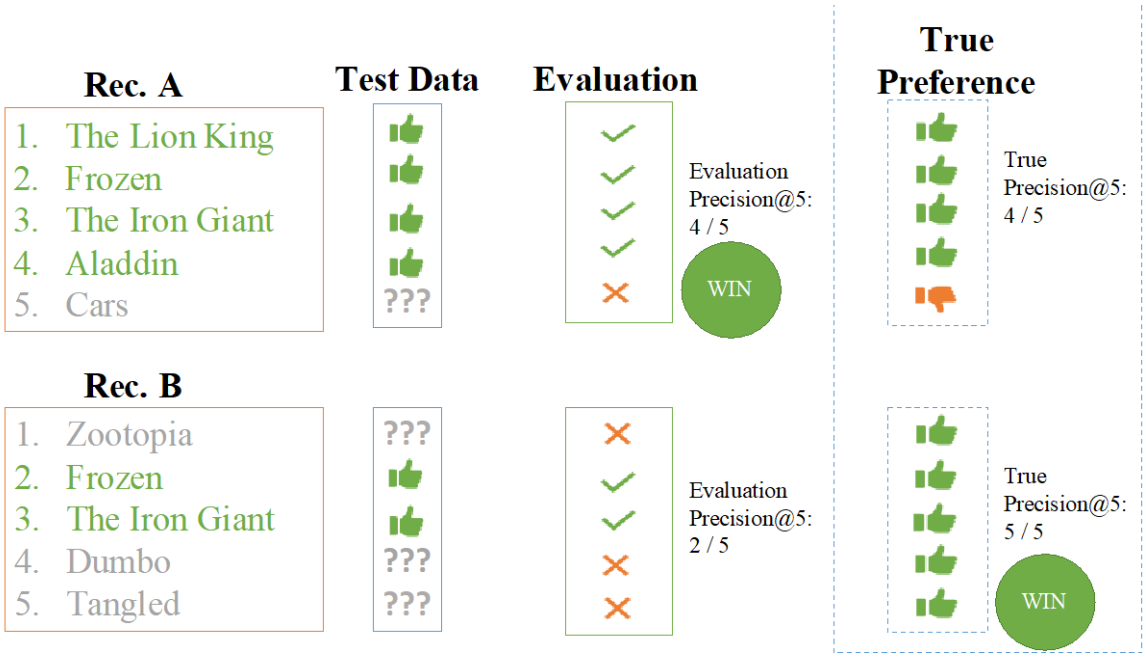

Figure 1: An example of evaluation metric computation.

information present is usually not a random sample of user preferences due to the complexities of user interactions with the system and its products. For example, users may be more likely to know about popular items and may also be more likely to consume items they like, so data on items they would like is missing not uniformly at random. This causes a number of well-known problems for recommender systems in general but has particular impact on offline evaluations of their effectiveness.

Figure 1 shows an example of how these problems can affect the computation of a common metric, the *precision* of a recommendation list. Precision measures the fraction of recommended items that are relevant to the user. The first column contains two recommendation lists for a user. In the test data, we can observe relevance information of movies (green color) the user has rated “like” but not unrated ones (grey color). The standard assumption is that unrated items are not relevant to the user. Based on this assumption and the observed relevance information, the evaluation result says that Rec. A

is better than Rec. B. But what if the user would actually like *Zootopia*, *Dumbo*, and *Tangled* if the user saw them? In this case, Rec. B would be a good recommendation list since it helps the user not only find movies they would like, but to find *novel* movies that they would like and have not seen. Assume the user's complete true preferences, which we cannot obtain in the traditional offline evaluation practice, are shown as in the last column. This "true precision" prefers Rec. B over Rec. A, which contradicts the evaluation result on observed data. Another evaluation issue shown in Figure 1 is popularity bias. In Rec. A, all movies are much more popular than ones in Rec. B (as measured by rating count on MovieLens). Since movies in Rec. A are popular movies they are more likely to be rated and to appear in the test data. A popular recommender like Rec. A that just recommends popular items can achieve a higher observed precision than a personalized recommender like Rec. B, even though Rec. B recommends more novel items that the user would like.

Unfortunately, while this missing data causes significant challenges for evaluating recommender effectiveness, its exact impact on experimental outcomes remains unknown. Specifically, we do not know how often missing data skews recommendation list evaluations, or how much effect this has on the outcomes of evaluation experiments. For commonly-used data sets, we do not know the underlying ground truth, and we also do not know the particular observation process (the process by which users discover and rate movies, turning preferences into observable data). We cannot look behind to compare the true metric values with the observed values to know how often these issues happen. This gives rise to ***metric error***, which we define as the difference between experimental results with observable data and the results that would be obtained if

complete user-item relevance information were available. To study how much the metric error impacts on evaluation outcomes, we are interested in how it is distributed. Current research, however, lacks a good understanding of the *distribution* of the error in experimental outcomes.

In this work, we present a simulation study that estimates the extent of missing data errors by comparing experimental outcomes on observable data with outcomes on complete data in simulated conditions. We use publicly-available data sets in multiple domains to calibrate our simulation models to produce realistic data, and then run offline evaluation experiments on these synthetic data sets. Finally, we compare commonly used evaluation metrics computed from observable data sets with ones computed from simulated true preference data sets.

1.1 Thesis Statement

Commonly-used evaluation metrics, including Precision, Recall, Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (nDCG), exhibit errors in estimating their true values. Simulations show that the degree of error varies between algorithms and assumptions on the data generation. Evaluation metrics on observable data generally underestimate the true metrics except Recall, and sometimes are fooled into mis-ranking algorithms.

1.2 Research Outline

In this thesis, we present a simulation study that estimates the extent of missing data errors by comparing experimental outcomes on observable data with outcomes on complete data in controlled conditions. We use publicly-available data sets in multiple

domains to calibrate our simulation models to produce realistic data. Our study is driven by the following research questions:

1. How do different simulation models and parameter tuning processes compare in their ability to replicate real data?
2. What error does missing data cause in evaluation metrics?
3. How does metric error change as we change the data generation process (i.e. assumptions about the data)?
4. What incorrect decisions does missing data cause?

CHAPTER TWO: RELATED WORK

Recommender systems are software applications deployed on the Internet to help people find useful items by providing recommendation lists. These recommendation lists are generated by different techniques that can be taxonomized in various ways. One categorization particularly relevant to this work is the *degree of personalization*: non-personalized recommenders, semi-personalized recommenders, and personalized recommenders [2]. The *popular* recommender is a typical non-personalized recommender that recommends the N most popular items to all the users. Semi-personalized recommenders usually recommend the same items to users in the same group in terms of their demographic information (e.g. local news recommendations based on the physical location of the user's IP address). Personalized recommenders recommend different personalized items to individual users based on their unique characteristics and consumption history (e.g. movie recommendations based on the user's profile of watched movies). One of the most common and successful personalized recommendation techniques is the *collaborative filtering* family of algorithms, including user based and item based nearest-neighbor, matrix factorization, and machine learning algorithms that mine patterns from user-item interactions for personalized recommendations [3].

2.1 Traditional Evaluation Methodologies

Recommender systems are driven by business goals (e.g. return on investment and user satisfaction). To select a recommender system that meets business goals, R&D engineers must evaluate a range of candidate algorithms on metrics that indicate or reflect

the business goals. The two main categories of approaches for evaluating recommender systems are online evaluation and offline evaluation.

2.1.1 Online Evaluation

The gold standard method for evaluating a recommender system's effectiveness or usefulness is online evaluation. Online evaluations deploy candidate algorithms online simultaneously and observe users' interactions with the systems or solicit user feedback on the recommendations and experience. Online tests can assess not only the recommendation itself, but also the user experience in which they are presented.

A/B Test

An A/B test is an online experiment that compares an existing recommendation algorithm with an alternative one in terms of business metrics. It randomly splits users into two test groups, exposing each group of users to a different test algorithm, then monitors users' interactions with the system to compute a metric relevant to the business such as conversion rate, click through rate, or sales volume. Statistical tests for randomly-controlled trials can determine if the metrics measured for these two groups have statistically significant differences, guiding the decision of whether to deploy the new algorithm [4].

User Study

A user study measures users' subjective perceptions and opinions about the system under experimentation through surveys and questionnaires. This provides insight to user behaviors and satisfaction that are available through observation [5].

The advantage of online evaluations of either type is that they directly measure recommender system performance in a way that is explicitly correlated with business or

user goals. However, they are costly and time-consuming: users who have bad experiences with the test algorithm may no longer use the system in the future, and online deployment of a new algorithm usually takes months to develop and test to make it ready for evaluation.

Online evaluations are also difficult to repeat or replay just using the data collected in previous experiments, making publicly-available datasets not suitable for redoing online experiments. For academic researchers, accessing a large number of users for online experiments is not always feasible. Much research can only be done offline, and even when online experimentation is available, it is better to first do an offline evaluation to gain confidence with a new algorithm before deploying it for online evaluation.

2.1.2 Offline Evaluation

Traditional offline evaluations use metrics and methodologies borrowed from machine learning and information retrieval to estimate the performance of recommendations. These methods follow a train-test evaluation procedure [5]:

1. partition users' consumption data into the training set and the test set.
2. train recommendation algorithms on the training set.

For each user:

3. generate a list of recommendations from a set of candidate items (typically the items that the user has not rated in the training set).
4. test prediction accuracy or ranking effectiveness using the withheld test data as the ground truth.
5. average the metric scores across all test users.

Offline evaluation metrics typically measure *prediction accuracy* or *top-N accuracy*. Prediction accuracy is measured with the error in predicting ratings, typically using either Root Mean Square Error (RMSE) or Mean Absolute Error (MAE). Metrics that measure top-*N* accuracy include Precision, Recall, Normalized Discounted Cumulative Gain (nDCG) [6], and Mean Reciprocal Rank (MRR) [7]. Early work focused on choices of evaluation metrics [8], specifically prediction accuracy vs. top-*N* accuracy, and comparison of predictive performance among various recommendation algorithms [9] using the values of these metrics. Accuracy differences measured by these metrics, however, are subtle in reflecting user goals for some tasks, and are sensitive to different data sets in use [8]. Herlocker, et al. [8] discussed the factors of data selection to perform evaluation, investigated the correlations between metrics, and explored new metrics that evaluate perspectives other than accuracy. Bellogín, et al. [10] studied the impact of the way that splits train test sets on the evaluation results.

2.2 Evaluation Problems

To measure prediction or top-*N* accuracy of recommender systems, the offline evaluation procedure requires relevance information of all recommended items to compute evaluation metrics. This information is available in supervised machine learning and controlled IR settings (e.g. TREC competitions). Recommendation scenarios, however, rarely have this ground truth information due in large part to the personalized notion of relevance for individual users. The standard practice for recommender evaluation is to assume that items missing relevance information are not relevant to the user; while this assumption is reasonable for individual items [11], its validity degrades as more items are considered (e.g. a recommendation list), particularly when relevance

data are not missing at random [12, 13]. There are several manifestations of this problem, including unknown relevant items and popularity bias.

2.2.1 Unknown Relevant Items

Unknown relevant items arise when the user would like an item but has not rated it in either the training or the test data. This item should be a good item to recommend, but since the evaluation protocol treats unrated items as irrelevant to the user, the recommender is penalized for recommending it. This is problematic in many applications where the goal of recommender systems is to recommend novel items to users that they would like. Because of unknown relevant items, offline evaluations reject excellent recommendations of novel items the user would enjoy because they were not known to the user in the system from which data was collected and are therefore missing from the test data.

2.2.2 Popularity Bias

Popularity bias is the effect that evaluations favor recommendations of popular items significantly beyond the usefulness of popularity in producing good recommendations. It arises because popular items are more likely to be exposed to users, then to be rated in both the training set and the test set, and the evaluation result is an average score of an evaluation metric across all users in the test set. A popular recommender often achieves a higher evaluation score than a personalized one just because popular items are more commonly the ‘right’ answer.

2.2.3 Impact

These problems cause significant challenges for evaluating recommender effectiveness, but their exact impact on experimental outcomes remains unknown.

Specifically, the field currently lacks a good understanding of the *distribution* of the error these problems induce in experimental outcomes, where **error** is the difference between the experimental results with available data and the results that would be obtained if complete user-item relevance information were available.

There are several existing lines of work on the validity of offline evaluations, including *changing the protocol* [14], *unbiased estimators* [15, 16], *counterfactual evaluation* [17, 18], and *simulations* [19].

2.3 Evaluation Protocols

The experimental protocol for an offline evaluation protocol defines strategies to partition data into training and test sets and selecting candidate items for recommendation for each test user. Different partitioning strategies serve different objectives and can also affect the results of an evaluation [10]. One approach to evaluation difficulties is to *change the protocol*, particularly the way that test items and candidate items are selected or analyzed, to neutralize popularity biases and reduce the likelihood of recommending misclassified decoys.

2.3.1 Popularity Treatment

Bellogín [14] proposed an alternative data splitting strategy to address popularity bias that aim to compensate for the rate at which different items appear in the test set. This strategy samples the test data so that each item appears as a test item an equal number of times; grouping data by item and sampling N users who have rated that item will accomplish this. The author also proposed a way to analyze evaluation results in order to mitigate popularity effect. This method aggregates evaluation metrics by popularity quantile; this enables analysis of the algorithm's effectiveness at different

popularity levels, and ensures that the least popular quantile of items influences the final score as much as the most popular quantile.

These methods affect absolute metric values, but not necessarily the relative performance of algorithms [10].

2.3.2 Unknown Relevant Items Treatment

Similarly, changing how the experimental protocol selects candidate items — the items the recommender considers when producing a top-N list for each user — may be useful in addressing misclassified decoys.

Standard evaluations use all items that aren't in the user's set of training ratings as candidates. If instead we use a candidate set consisting of the user's test items plus a random sample of unrated items (sampled anew for each user), we can decrease the likelihood of misclassified decoys. This is because unknown relevant items are relatively rare, so they are probably not going to be picked as a part of the sample [20].

However, this method's usefulness relies on unrealistically strong assumptions of the rareness of unknown relevant items, and it likely exacerbates popularity bias [11]. Its efficiency is therefore suspect and in need of further study.

2.4 Unbiased Estimators

Another proposed solution is to select evaluation metrics that admit statistically unbiased estimators using the observed data.

Under assumptions that (1) ratings for relevant items are missing at random and (2) the non-relevant ratings have a higher probability of being missing than the relevant ones, computing top-N hit rate (recall) using observed data is an unbiased estimator for the true value [16]. Under the same assumptions, computing non-normalized discounted

cumulative gain with observed implicit feedback data is an unbiased estimator for the true value based on complete data [15].

There are two significant limitations to this approach. First, it limits the choice of metrics; in assessing recall, Steck [16] observes that computing precision with observed data is not an unbiased estimator. Lim, et al. [15] show that the more common normalized discounted cumulative gain is biased, so producing an unbiased estimate requires sacrificing normalization.

In general, therefore, this approach requires a tradeoff between statistical validity and appropriateness of the metric to the task. If the recommendation task is best captured by a metric without an unbiased estimator, then effectiveness for that task cannot be reliably assessed. Second, the necessary assumptions are unlikely to hold in realistic scenarios. Ratings or consumption events are not sampled at random from the relevant items; the user's choice of items to rate arises from a complex discovery process based on user knowledge, social networks, and existing discovery tools.

2.5 Counterfactual Evaluation

One particularly powerful means of addressing the weaknesses of offline evaluation is to reframe the recommendation and evaluation problem as a counterfactual learning problem [17, 21, 18]. This approach aims to reconstruct from offline data an estimate of how the user would have responded had they received a different recommendation.

Counterfactual evaluation has the enormous benefit of actually measuring the problem that we most often care about, particularly from business and user response perspectives: the ability to recommend items the user will accept. Its downside is that it

represents a substantial break from historical practice and often is not applicable to commonly-used data sets. The largest available data set for counterfactual evaluation, from Criteo [21], is valuable but also opaque: the lack of descriptors for item features means that less insight can be obtained about algorithm behavior and performance.

While we should — and do — welcome such breaks when they move the field forward substantially, we would also like to understand how much knowledge under the old paradigm can be carried forward, and develop techniques when possible that can be used with more common data sets.

2.6 Simulation

Neither reframing the problem to avoid the pitfalls of classical evaluation nor choosing provably unbiased estimators answers a key question for interpreting previous results: just how wrong are they? Further, the widespread availability of data, metrics, instructions, and tools for classical evaluations makes them relatively easy to perform; if there is a way to improve their accuracy that can be deployed in existing scenarios, such techniques would significantly improve the reliability of recommender systems research and testing.

The most promising technique we see for this work is simulation. Since, by its very nature, we cannot know the underlying ground truth for observed data, and we do not know the particular process by which the observed data was generated, we can't (except in a few limited circumstances) look behind the data to compare observed metric values to what they would be if we had complete relevance data. Simulation, however, lets us open the curtain: by generating complete and observed data under a range of

scenarios, we can look at how the observed results vary based on different possible observation processes.

Chaney et al. [22] models the feedback loop between user consumption behavior and recommender systems to analyze the impact of algorithmic confounding on user utility. Their simulations found the feedback loop increases homogenization of user behavior without gaining utility, and it also affects the distribution of item consumption. Their work focuses on the impact of the feedback loop on the resulting user utility matrix rather than evaluation metrics. Cañamares & Castells [19] built a probabilistic model to analyze the conditions that determine the usefulness of popularity in recommender systems and better understand popularity bias under various conditions. They defined optimal ranking strategies that maximize the true or observed precision@1 for non-personalized recommendation. By changing conditional independence among three variables —item relevance, item discovery, and item rating — the authors analyzed how the popular recommender and the average rating recommender perform compared to optimal and random recommenders under both observable and true precision. They found that the most-popular recommender is close to the optimal recommender in observed precision and the average-rating recommender is close to optimal in true precision if rating presence is conditionally independent of relevance or no independence assumptions are made.

CHAPTER THREE: METHODS

As discussed in chapter one, the goal of this research is to estimate the distribution of *metric error* in offline evaluations, where metric error is the difference between metrics computed with observable data and those computed with true preference data. To estimate these error distributions, we need to know users' underlying true preferences for recommended items. Commonly-used public data sets, however, lack this true information, and the particular observation process that produced these data sets is also unknown. Therefore, we cannot "look behind" a data set to compare observed metric values against the "true" metric values.

To overcome these problems, we simulate the entire recommender system experiment, from preference construction through data collection to offline evaluation. With access to the ground truth data, because it is generated by the simulator, we can measure what the precision or reciprocal rank of a recommendation list would be if the data set were not missing data and compare that value to the metric obtained from the observable data an experiment would ordinarily employ. This allows us to produce a first approximation of the error in experimental results where we cannot access unbiased truth. The simulated true preferences also enable us to test experimental protocols on oracle recommenders that omnisciently return the most relevant items, irrespective of observation process. By comparing evaluation results of the oracle recommender with other recommenders, we can answer how often the evaluation mistakenly ranks a

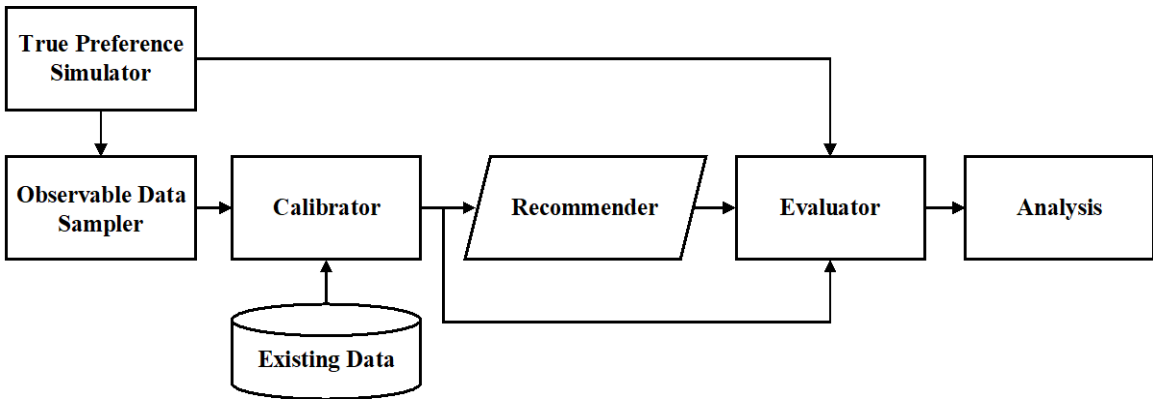

Figure 2: Simulation architecture.

baseline recommender over a perfect recommender under different generation assumptions.

Figure 2 shows our simulation architecture. We employ a two-step data generation process: we first simulate complete (binary) user-item relevance data, then sample observations (ratings or purchases) from this complete truth. This two-step data generation has two main advantages: 1. it provides flexibility in data modeling, allowing us to produce observable data with similar distributions from different true preference models; and 2. it enables us to investigate the impact of true preference assumptions and observation processes separately on the resulting metric values. For example, we can encode popularity effects into the true preference model and the *missing-at-random* assumption into the observation process through this data generation process.

To ensure that our simulated data is reasonably realistic, we tune the parameters of our simulations to mimic key statistics of existing public data sets. We then split the simulated observations into training and test data, generate recommendations for the simulated users, and measure the quality of these recommendations using both the observed test data and the underlying true preferences as ground truth.

3.1 True Preference Generators

We employ three different models for simulating items being relevant to users: a uniform model, a preferential attachment process [23, 24], and a correlated preference model [25]. These models each encode different assumptions about user-item interactions. Each sampling process produces a set U of users and sets $\tilde{I}_u \subseteq I$ of items liked by each user. Our current models simulate binary preference ("like" or "don't like"); continuous preference is a natural next step that we leave for future work.

3.1.1 Uniform Model

The uniform model assumes each user likes each item with an equal probability for a given number of items the user likes. The uniform model removes the popularity effect in users' true preferences. While it is not a realistic model, it enables us to analyze the evaluation process without popularity biases.

The uniform model is implemented as follows:

1. Draw $|\widetilde{I_u}|$ (the number of items the user likes) from a Poisson distribution with mean λ.
2. Sample $|\widetilde{I_u}|$ items without replacement uniformly from a total of $|I|$ items.

3.1.2 Popularity Based Model

User consumption data in recommender systems exhibits strong popularity effects [14], often following a long-tailed distribution with power law behavior [26]. The latent structure that generates this kind of observed data can be modeled as a preferential attachment process [23]; such processes are commonly modeled using the "Indian buffet process" (IBP). In this thesis, we employ the three-parameter generalized IBP proposed by Teh & Görür [24]. This model is capable of producing data exhibiting power law

behavior, unlike a traditional IBP [27]. The IBP model with parameters $\alpha > 0$, $\sigma \in [0, 1)$, and $c > -\sigma$ is defined as follows:

1. The first user likes $\text{Poisson}(\alpha)$ items.
2. User $(n + 1)$ likes previously-known item i with probability $\frac{m_i - \sigma}{n+c}$ (where m_i is the number of users who like item i) and likes $\text{Poisson}(\alpha \frac{\Gamma(1+c)\Gamma(n+c+\sigma)}{\Gamma(n+1+c)\Gamma(c+\sigma)})$ new items.

c controls how likely the user is to rate new vs. old items. σ governs the power-law behavior of the generated preference matrix; $\sigma = 0$ yields a traditional IBP [28], with larger values yielding stronger power-law distributions of item popularity. α controls the density of the generated preference matrix. When $\sigma > 0$, the process generates on average $\alpha * |U|^{\sigma}$ items [24]; when $\sigma = 0$ and $c = 1$, it generates approximately $\alpha * (\log|U| + \gamma)$ items on average, where γ is Euler's constant [23].

The IBP model assumes that users like items independently; if a user likes item i, it says nothing about their preference for item j. This property allows us to scale up the simulation size through parallelism at the expense of realism.

3.1.3 Correlated Preference Model

The independence assumption is deeply questionable, however, because item preferences often are correlated, and exploiting those correlations is fundamental to many recommendation techniques. Latent feature models provide a mechanism for representing correlations between items, as a user who likes an item that loads strongly on a feature is more likely to like other items that also load on the feature. One such model, suitable to simulating binary data, is the latent Dirichlet allocation model (LDA) [25]. The LDA generation process for K latent features is as follows:

1. Draw K feature-item vectors $\vec{\phi}_k \in [0,1]^{|I|}$ from $\text{Dirichlet}(\beta)$.
2. For each user:
    a. Draw a latent feature vector $\vec{\theta}_u \in [0,1]^K$ from $\text{Dirichlet}(\alpha)$.
    b. Draw n_u (the number of items) from $\text{Poisson}(\lambda)$.
    c. Draw items $i_1, \dots, i_{n_u}$ liked by user u by drawing feature $k_x \sim \text{Multinomial}\left(\vec{\theta}_u\right)$ and $i_x \sim \text{Multinomial}(\vec{\phi}_{k_x})$.
3. De-duplicate user-item pairs to produce implicit user preference samples.

To reduce the number of parameters for fitting efficiency, we use symmetric LDA, where α is a constant vector with all values equal to $a > 0$, and likewise β is constant $b > 0$. These parameters a and b control the breadth of user preferences; when $a < 1$, the values of $\vec{\theta}_u$ concentrate on a few of K dimensions, making the user's preferences concentrate on a few of items if $b < 1$. The parameter λ controls the average number of items each user likes. The parameter K controls the size of the latent feature space, affecting the diversity of user-item preference patterns in the whole true preference data.

3.2 Observation Samplers

We turn simulated preference into synthetic "rating" data sets by sampling observations of user consumption from the true preferences. We use two different models for this sampling; these encode different assumptions about the process by which users discover and consume items they like. The result is a set $I_u \subseteq \widetilde{I_u}$ for each user.

3.2.1 Profile Size Controller

Both observation models start by drawing n_u, the number of items a user will rate. Since each observable user consumes at least one item (or some larger number, such

as 20 for MovieLens data sets [29]) and user activity levels follow a heavy-tailed distribution, we draw n_u from a truncated Pareto distribution rounded to an integer in the range $[1, |\widetilde{I_u}|]$[1]. The generation of random variables from the truncated Pareto distribution is implemented by inverse transform sampling with three parameters: $m_{low} > 0$, the scale parameter controlling the lower bound of the distribution; $\alpha > 0$, the shape parameter; $m_{high} > m_{low}$, the upper bound of the distribution. Inverse transform sampling draws a random sample p from $\mathrm{Unif}(0, F_{Pareto}(m_{high}))$ and returns $F_{Pareto}^{-1}(p)$, where $F_{Pareto}(x)$ is the cumulative distribution function of Pareto distribution.

3.2.2 Uniform Sampling

The uniform sampler samples n_u items uniformly at random from $\widetilde{I_u}$ to form I_u.

This strategy encodes the *missing-at-random* (MAR) assumption, allowing us to compare our simulation results with analyses of unbiased estimators [16].

3.2.3 Popularity Sampling

The popularity-weighted sampler embodies the idea that users are more likely to consume items that they are exposed to, and they are more likely to be exposed to popular items than unpopular ones. This is one way in which observed data may violate the missing-at-random assumption underlying other work.

This strategy also samples n_u items from $\widetilde{I_u}$, but each item's selection probability is proportional to $|\widetilde{U_i}|$, where $\widetilde{U_i}$ is the set of users who like item i in the true preference

[1] We also tested the truncated beta-binomial distribution, its performance is similar to the truncated Pareto; rejection-sampling when $n_u > |\widetilde{I_u}|$ produced slightly better simulations than clamping at substantial computational expense.

data. This accounts for the popularity effect in observation in the way that items liked by more users are more likely to be consumed in the observable data than items liked by less users, because they are more widely known.

3.3 Data Sets

To realistically reason about the impact of varying assumptions about the data generation, we need a reference point with which we can compare simulated observable data sets from our different models to assess their realism. We use three data sets from different domains as reference points for calibrating the simulation process, summarized in Table 1.

ML1M [29] contains 1M ratings of 3,706 movies from 6,040 users, where each user has at least 20 ratings.

AZM5 [30] contains 65K reviews with 5-star ratings of 3.6K digital music albums from 5.5K users, where each user and each item has at least 5 reviews (the “5-core”).

STMV1 [31] contains 5M purchases of 11K video games by 71K Australian users of the Steam game distribution service.

Table 1: Summary of data sets

Datasets	Users	Items	Pairs	Density
ML1M	6,040	3,706	1,000,209	4.47%
AZM5	5,541	3,568	64,706	0.33%
STMV1	70,912	10,978	5,094,082	0.65%

These data sets cover different data sparsity, user activity, and item popularity distributions, allowing us to examine the robustness of our data generation process and assess the influence of modeling parameters on evaluation results.

3.4 Calibration

We calibrate our simulations to produce synthetic observation data sets that mimic key statistics of our reference data sets. We compare synthetic data from our simulator to our reference data sets using the K-L divergence [32] between synthetic and observed distributions of four properties: item popularity, user activity, item-item correlation, and user-user correlation. While these key statistics are not complete, they provide a good starting point since they capture data dynamics in both item and user dimensions, and data patterns reflected by these statistics are usually mined by collaborative filtering algorithms. A realistic simulation data set should be comparable to the reference data set on at least these statistics. Our goal of this calibration is to converge simulated data sets produced by various models to the same reference data set (as measured by similarity in statistical distributions) so we can investigate the impact of assumptions about the data generation on the distribution of evaluation results. We do not need to be able to compare individual users or items between synthetic and published data, like what matrix factorization does, to assess the distribution of evaluation results; we can think of the reference data set as one sample drawn from our data generation distribution with specific user and item index orders.

3.4.1 Item Popularity & User Activity

We construct item popularity and user activity distributions by counting the frequency of each popularity or activity level (profile size) in the observed data set.

If a model is producing realistic data, then the distributions of item popularity and user profile size should be comparable to those distributions in a real data set.

3.4.2 Pairwise Correlation

We compute item (and user) correlation distributions by sampling 1M unique item (user) pairs and computing the cosine similarity between their rating vectors, where a vector element is 1 if the user consumed the item and 0 otherwise. We exclude items (and users) with fewer than 5 ratings to keep the prevalence of 0-correlations (due to data sparsity) from overwhelming the comparison — if we did not do this, the metric only assessed the simulation's ability to produce a suitable number of 0s and did not meaningfully measure the distribution of nonzero correlations. We compare the distribution of these similarity values between the synthetic and reference data. To construct the distribution of these similarity values, we use Numpy's histogram function with default settings [33] and normalize the histogram to a density distribution with support of (-1, 1).

3.4.3 Parameter Optimization

We use Gaussian process minimization, as implemented by Scikit-Optimize [34], to find simulation parameters that minimize the K-L divergence from synthetic data to observed data on each distribution. Gaussian process minimization is much more efficient than commonly-used grid search or random search for large numbers of tunable parameters, particularly for non-differentiable models such as our simulation procedures.

We optimized parameters for all six models (each combination of preference and observation models) using each of the distributions on each of the data sets, yielding $3 \times 2 \times 4 \times 3 = 72$ models. By searching parameters optimized for a single distribution, we found that parameters optimized for one statistic distribution may not necessarily result in optimized results for other distributions. Unfortunately, K-L divergence is not amenable

to multi-objective optimization because values are not comparable across target distributions; a K-L divergence of 0.5 on item popularity does not mean the same thing as a divergence of 0.5 on user similarity. To overcome this limitation, we computed the *relative loss* on each statistic. Relative loss is the ratio of the K-L divergence on a statistic to the best K-L divergence obtained by optimizing our family of models for that statistic on that data set. These relative loss values are on a scale that can be compared; for example, LDA-Pop's fit on Item Popularity for ML1M is 6.9% worse than the best known model, while its fit on User Activity is 82.3% worse. We then compute the *average loss* for a model by taking the mean of the relative loss across all four statistics, providing a single score for which models can be optimized. This score weights all distributions equally; exploring the impact of different relative weightings is future work.

3.5 Evaluation Experiments

Finally, we use synthetic data from tuned simulation models to simulate recommender evaluation experiments that measure top-N accuracy.

3.5.1 Evaluation Protocols

We held out 20% of each user's observed items as testing data, generated 50-item recommendation lists, and computed commonly-used recommendation accuracy metrics using LensKit [35].

We computed each metric two ways: once with the held-out observable test data as ground truth, and again with the simulated true preference data as truth. We repeated each experiment, including data generation, 100 times.

3.5.2 Recommenders

We test evaluation metrics on three recommenders:

- The Oracle recommender knows all relevant items and always recommends top-*N* relevant candidate items for each user. This relevance information is produced by preference models that models users' complete true preferences. This basic Oracle recommender allows us to estimate evaluation results on a perfect recommender. Our current Oracle does not do anything to specifically promote novelty, but will recommend relevant items that are not in the test data as a result of its design.
- The Popular recommender recommends the most popular items using popularity statistics from the observable training data. This allows us to study how the popularity effect impact on the evaluation metrics, particularly when the sampling process is popularity-weighted sampling.
- The Random recommender recommends random unrated items for each user. This gives us the lower bound of estimating evaluation error: do evaluation metrics prefer the Random over the Oracle recommender?

Our goal of this study is to evaluate the evaluation process of recommender systems rather than recommender systems themselves. Oracle and Random recommender provide us an upper bound and a lower bound of the recommendation quality in complete data sets. Given these two reference points, we can compare how much an assumption about the data generation impact on evaluation results. For example, we can answer does the popularity assumption make evaluation favor Popular recommender over Oracle recommender?

We report results for commonly-used top-N evaluation metrics: Precision, Recall, MRR, and nDCG.

CHAPTER FOUR: RESULTS

This section presents the results of running our simulations. We start by reporting the optimization results of each model with respect to each reference data set on the average relative loss of the best known K-L divergence scores, and then we use simulated observable data generated from each model with optimized parameters to run recommender experiments and compare the metric scores on both simulated true preference and observable data sets.

4.1 Calibration Results

To answer RQ1, we tune parameters of our preference models and observation models with Gaussian process minimization as implemented by scikit-optimize, running for 150 iterations. In each iteration, scikit-optimize generates a set of parameters for our models in its search space; we then generate synthetic data with this set of parameters and compute the K-L divergence of the distribution (e.g. item popularity or item-item similarity values) from simulated data to the reference data, and scikit- optimize uses the results of this measurement to update the search direction. Table 2 summarizes the parameter spaces of our models. For each preference model, we use the number of users and items in the reference data sets as the corresponding model parameters. In the Unif model, λ controls the expected number of items a user likes; we select a search space of $(5, 2000)$ to cover all data sets. For IBP, α affects the number of items $|I|$ and the density of simulated data sets. As mentioned in Section 3, $|I|$ can be approximated by $\alpha \times |U|^{\sigma}$

Table 2: Summary of model parameters.

Model	Param.	Description	Search Space		
			ML1M	AZM5	STMV1
Unif	$\|U\|$	Number of users	6040	5541	70912
	$\|I\|$	Number of items	3706	3568	10978
	λ	Expected number of likes for a user	(5, 2000)	(5, 2000)	(5, 2000)
IBP	$\|U\|$	Number of users	6040	5541	70912
	α	Mass parameter controlling $\|I\|$.	(20, 1000)	(10, 1000)	(10, 500)
	c	Concentration parameter	(0.01, 100)	(0.01, 100)	(0.01, 100)
	σ	Stability exponent controlling power-law behavior	(0, 0.99)	(0, 0.99)	(0, 0.99)
LDA	$\|U\|$	Number of users	6040	5541	70912
	$\|I\|$	Number of items	3706	3568	10978
	K	Number of latent features	(5, 200)	(5, 200)	(5, 200)
	λ	Mean of $\text{Poisson}(\lambda)$	(5, 2000)	(5, 2000)	(100, 2000)
	a	The element value of α	(0.01, 1)	(0.01, 1)	(0.01, 1)
	b	The element value of β	(0.01, 1)	(0.01, 1)	(0.01, 1)
Truncated Pareto	m_{low}	Lower bound of the truncated Pareto distribution	(16, 24)	(4, 6)	(1, 1.2)
	α	The shape parameter	(0.1, 20)	(0.1, 20)	(0.1, 20)
	m_{high}	Upper bound of the truncated Pareto distribution	(1851, 2777)	(462, 694)	(2000, 9238)

for a large number of items and users, so we restrict the search space of α to a small space for simulating large data sets to avoid generating very dense data sets that takes large memory and simulation time.

Figure 3 – Figure 8 show key statistic distributions of our six models optimized for average relative loss on ML1M data set in a form suitable for graphical inspection [36], where the distribution of I-I Sim (and U-U Sim) is plotted using 1M samples of 5-core item-item (user-user) pairs. Unif-based models seem to only fit on the user activity distribution due to our profile size controller, while IBP-based models and LDA-based models can fit on all statistics.

Figure 9 – Figure 14 show calibration results for AZM5 data set. Both IBP-based models and LDA-based models have good fitting qualities on all statistics. This may because the reference data set is composed of users and items with at least 5 ratings, making users and items more correlated with each other than other reference data sets, then more suitable to our models. IBP-based models also have better fit on item popularity than LDA-based models. This may be due to the stability exponent parameter σ that enables IBP models more adaptable to various power-law behavior.

Figure 15 – Figure 20 show calibration results for STMV1 data set. Most of our models are not able to fit on this reference data set as well as on other reference data sets. One possible cause is that STMV1 data set is much more sparse than other two data sets and has larger user and item dimensions. Due to our limit of the search space, our models are more likely to produce extremely sparse data sets in which most users have a profile size lower than 5. Then scikit-optimize has a high chance of not finding good initial parameters during the first 25 iterations of initial random search stage. In the future work,

we hope to explore more relations between parameter search spaces and the reference data properties (e.g. density), then we can derive a proper search spaces from the reference data properties.

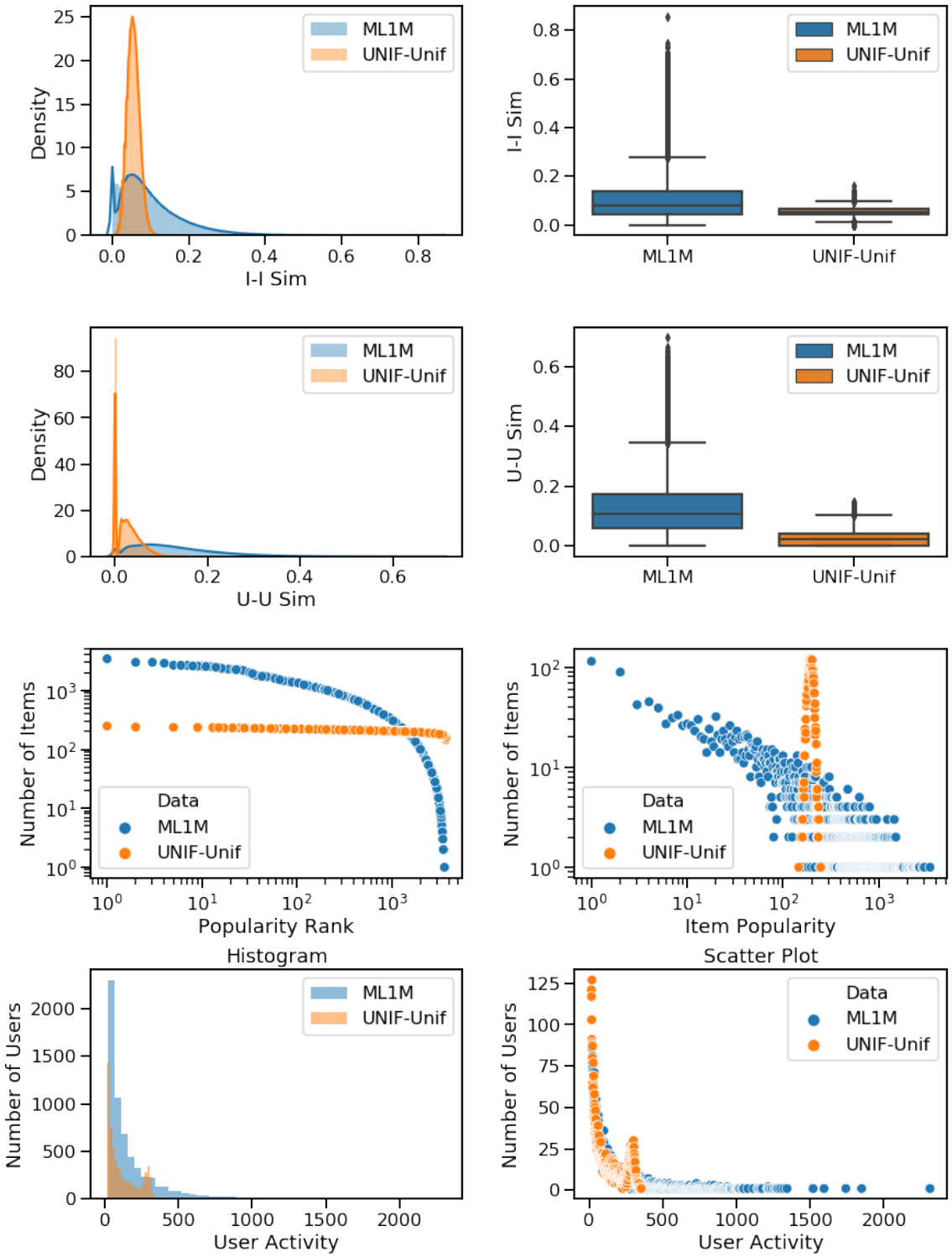

Figure 3: Key statistics of Unif-Unif model optimized to ML1M data.

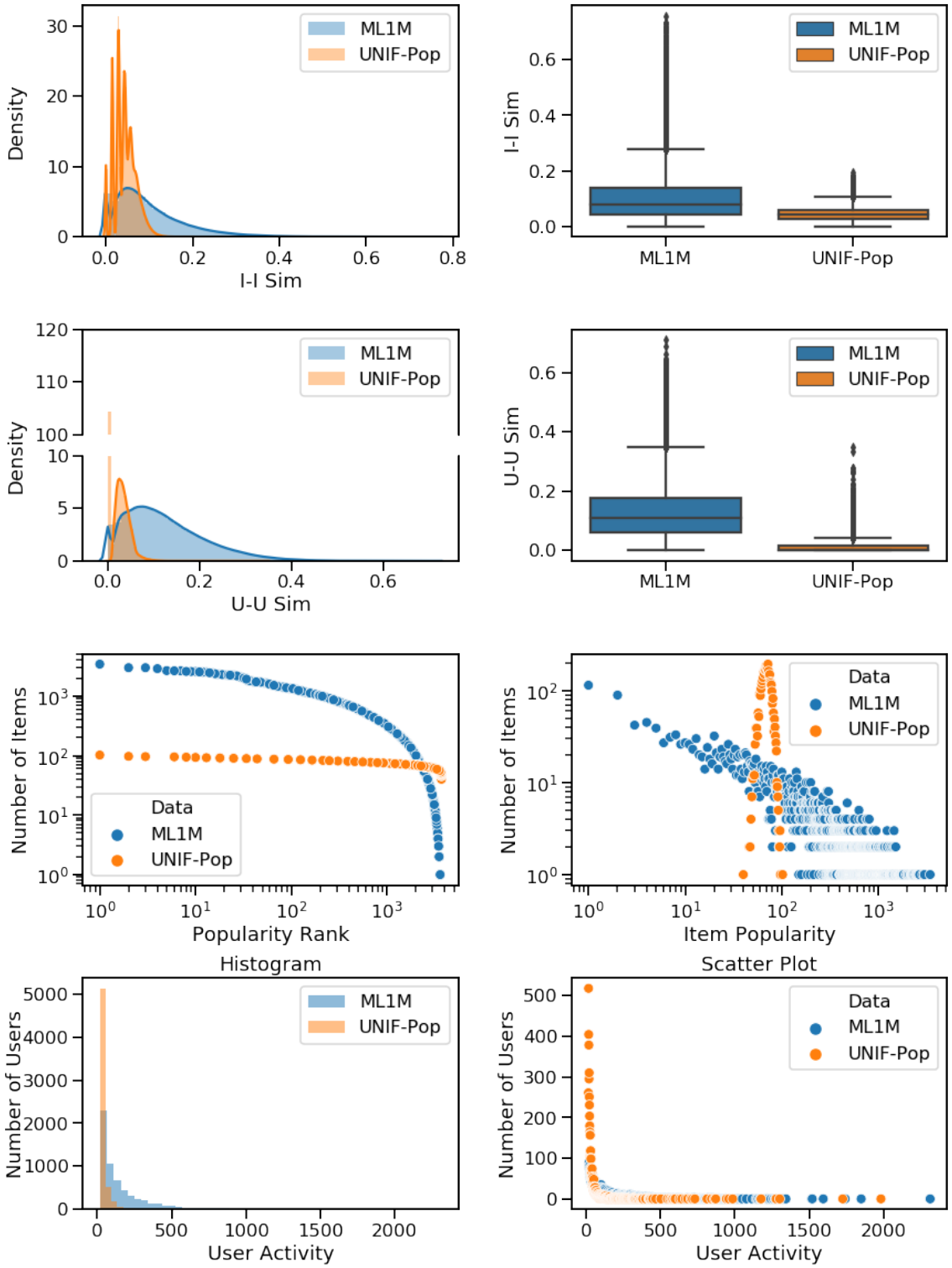

Figure 4: Key statistics of Unif-Pop model optimized to ML1M data.

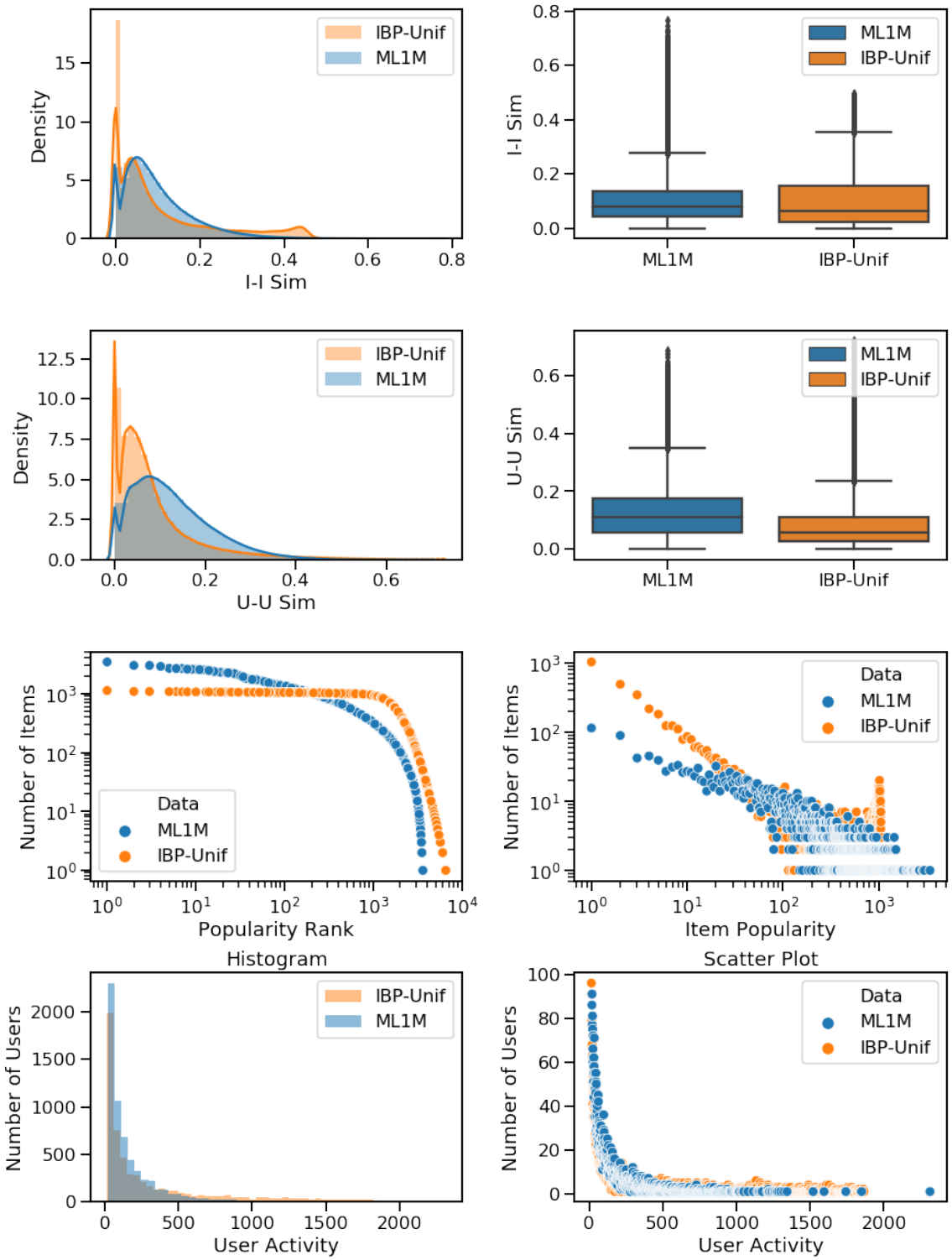

Figure 5: Key statistics of IBP-Unif model optimized to ML1M data.

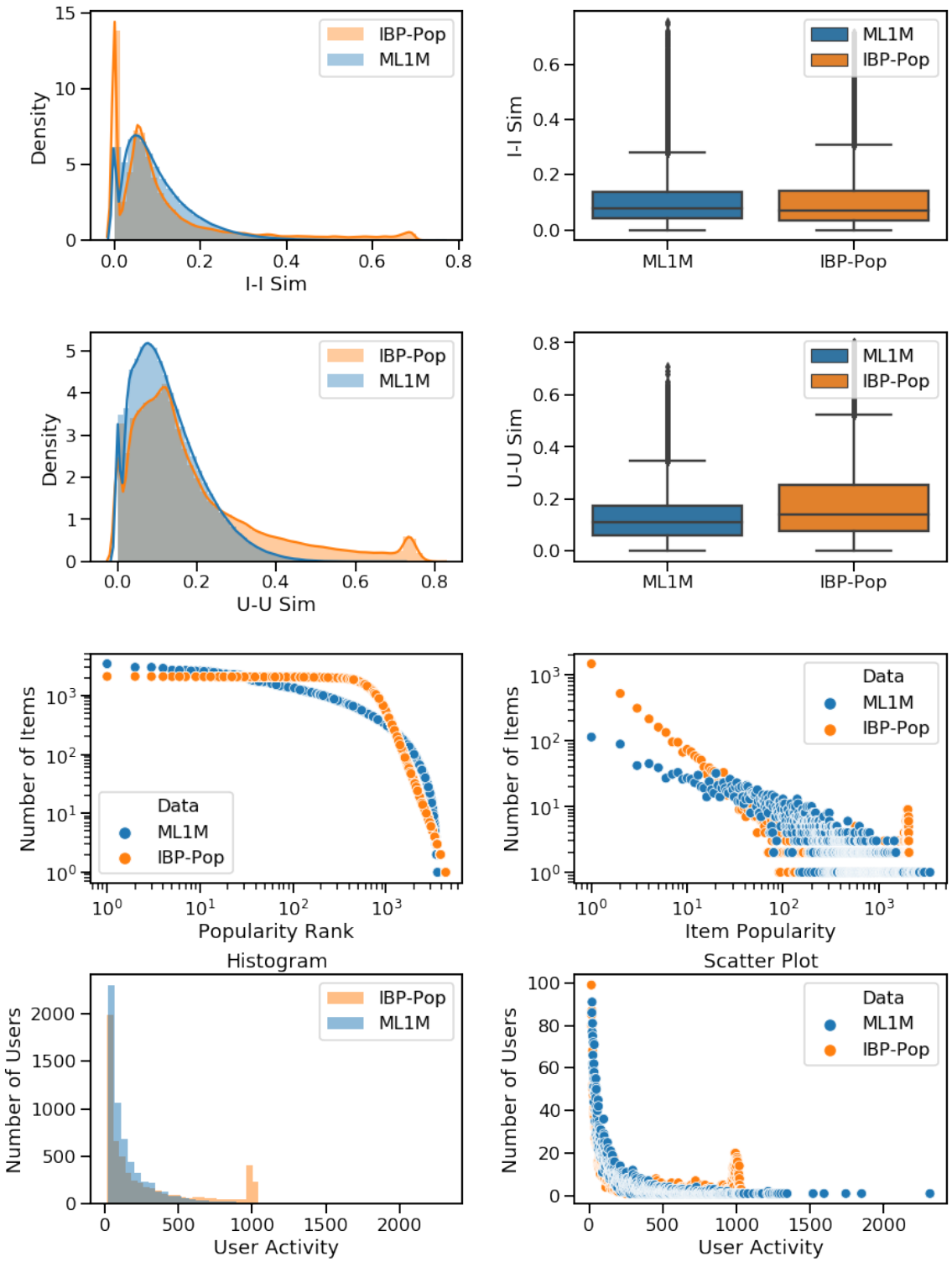

Figure 6: Key statistics of IBP-Pop model optimized to ML1M data.

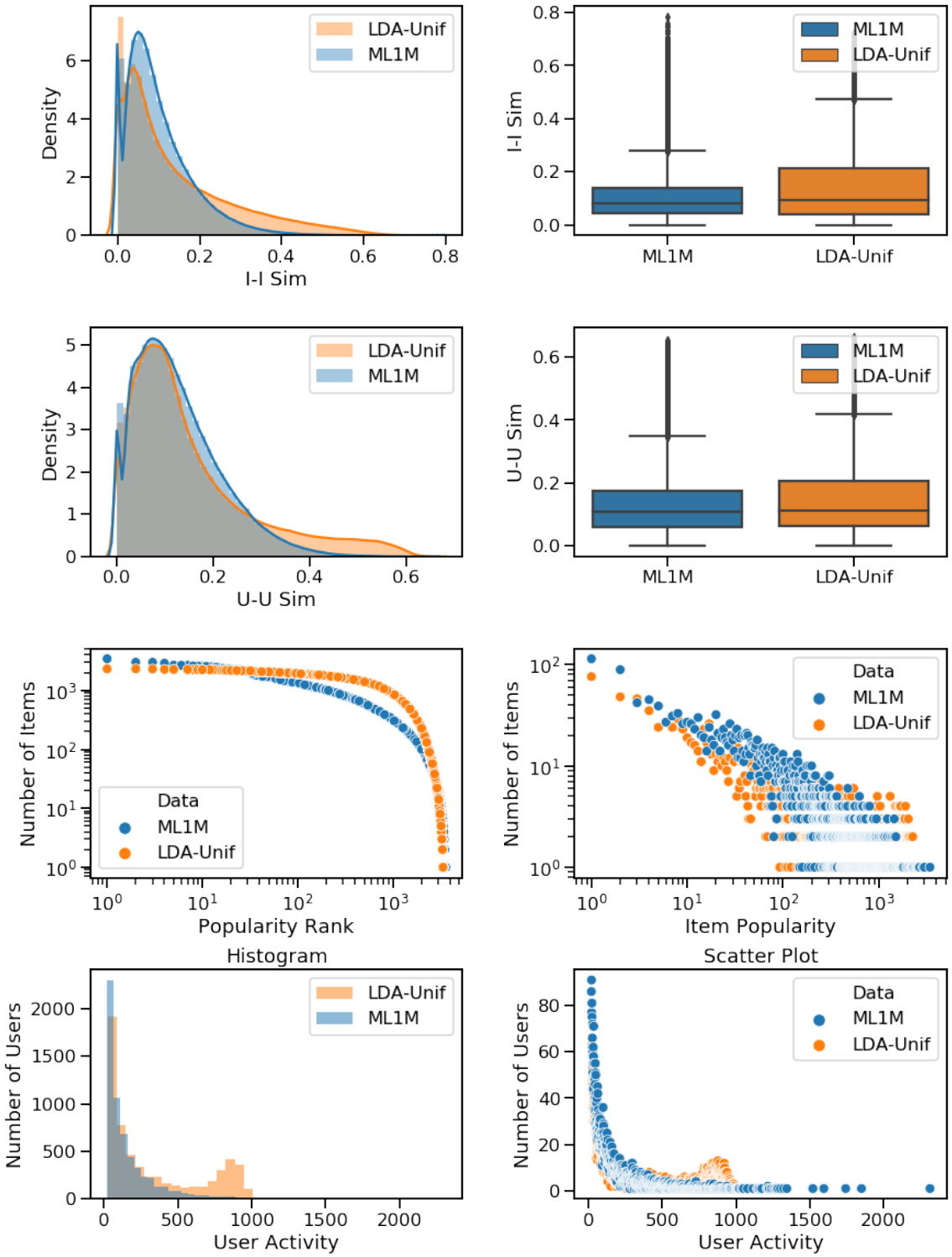

Figure 7: Key statistics of LDA-Unif model optimized to ML1M data.

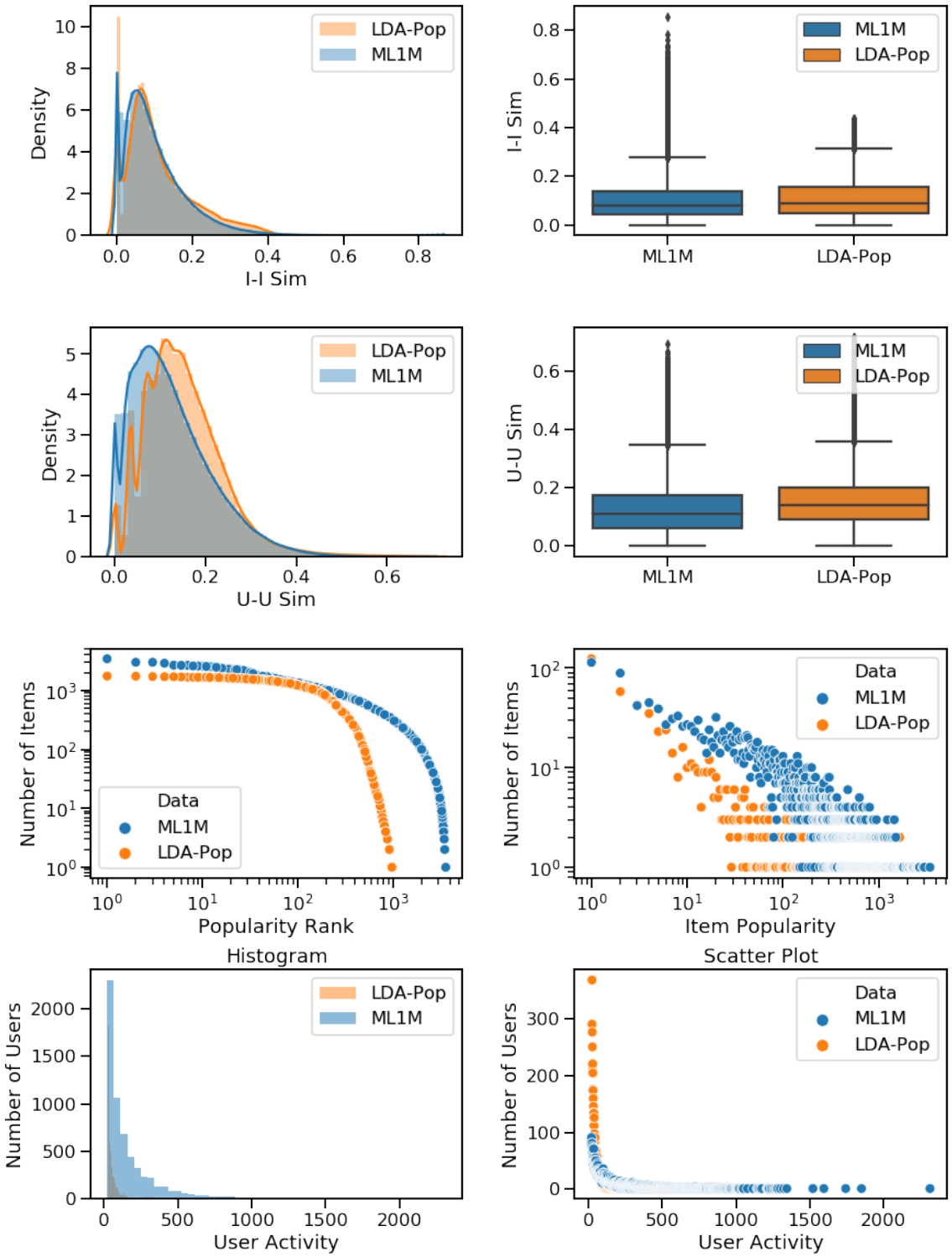

Figure 8: Key statistics of LDA-Pop model optimized to ML1M data.

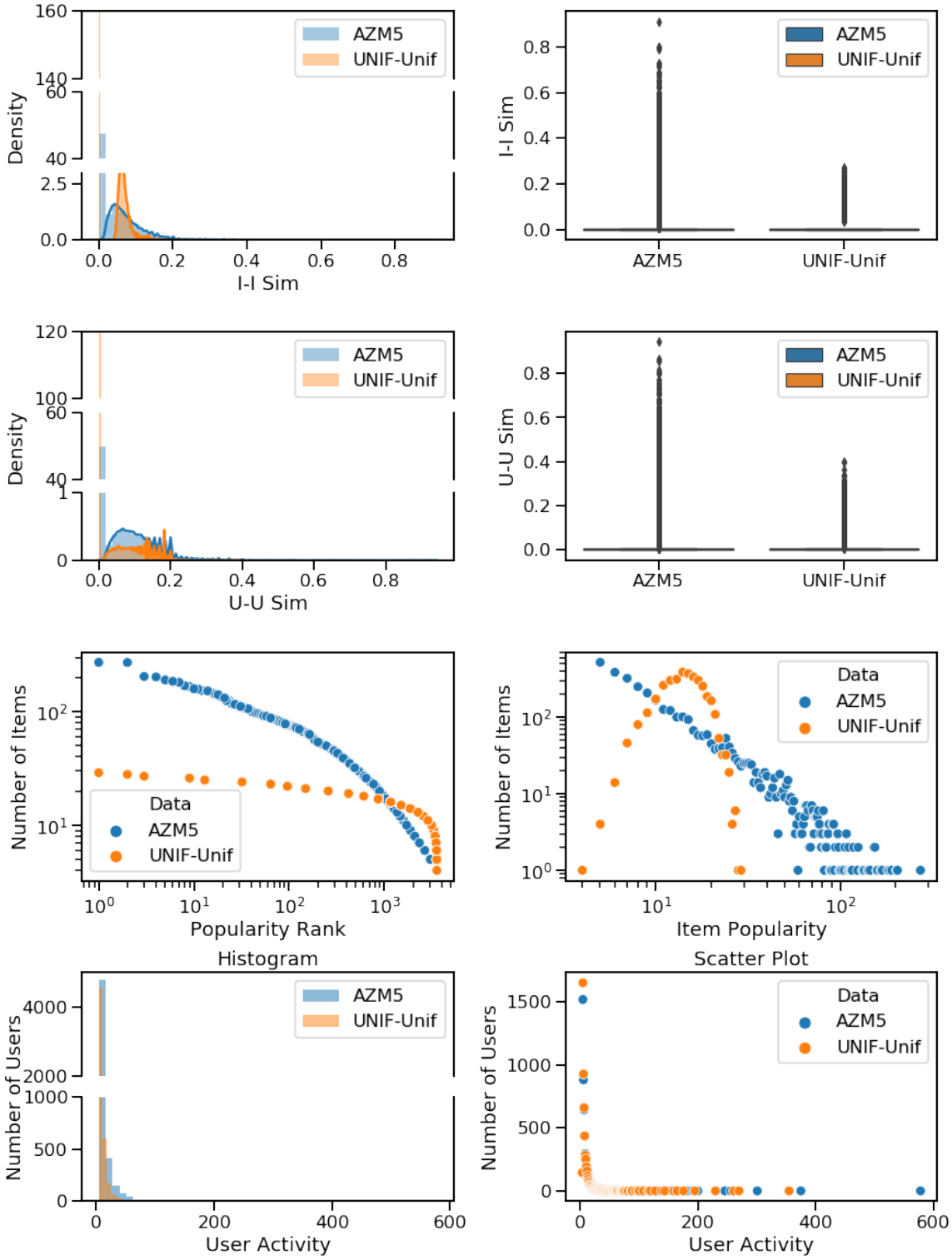

Figure 9: Key statistics of Unif-Unif model optimized to AZM5 data.

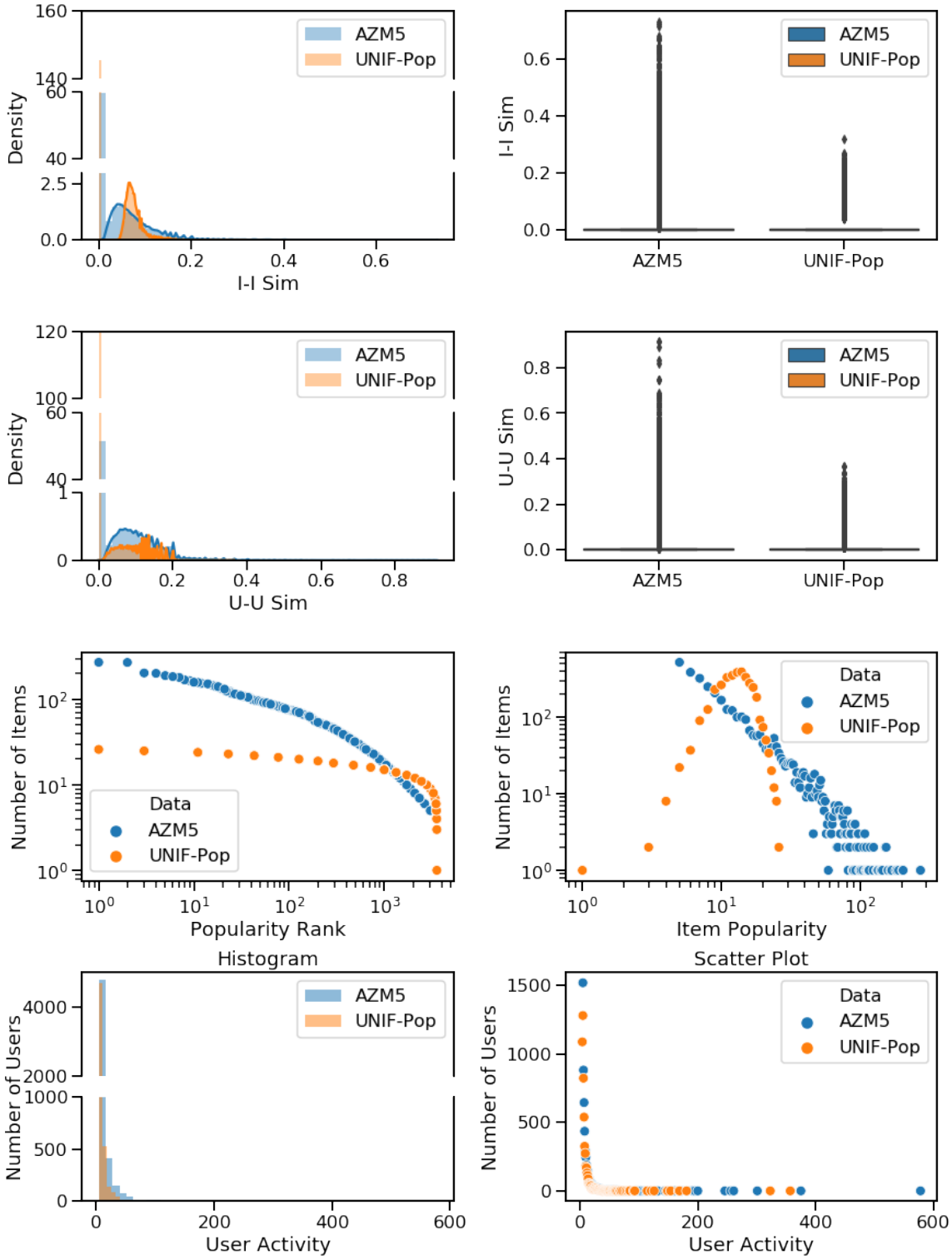

Figure 10: Key statistics of Unif-Pop model optimized to AZM5 data.

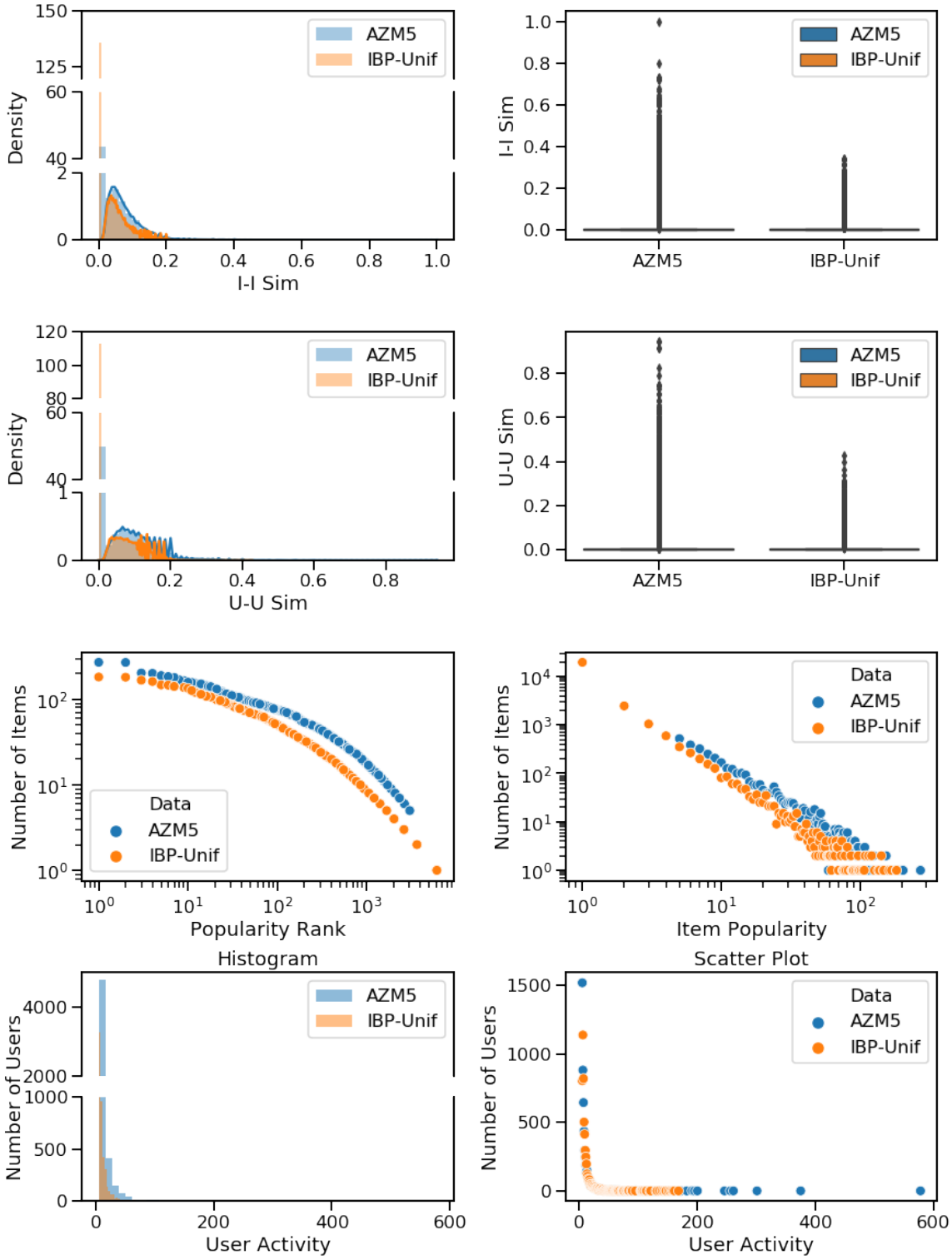

Figure 11: Key statistics of IBP-Unif model optimized to AZM5 data.

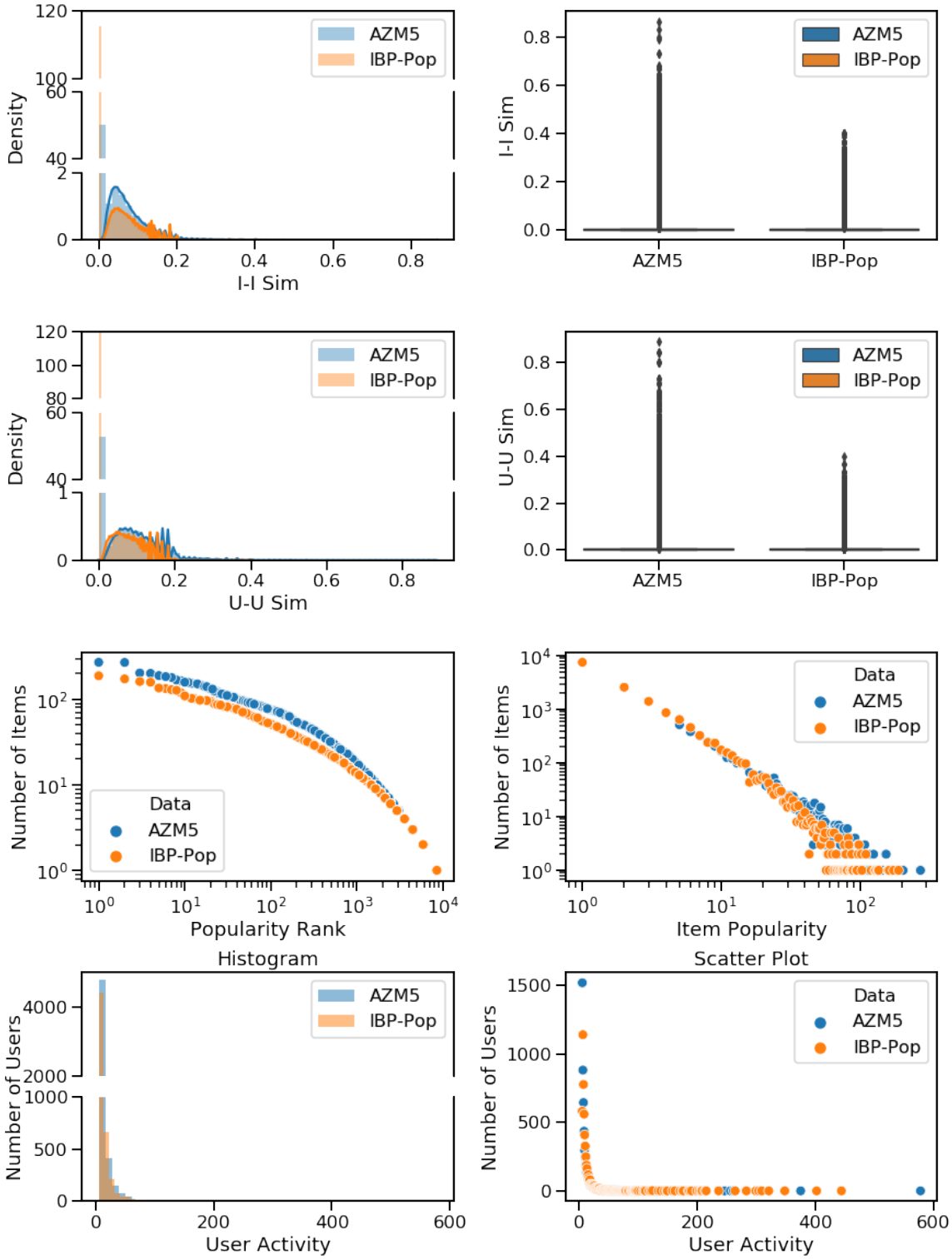

Figure 12: Key statistics of IBP-Pop model optimized to AZM5 data.

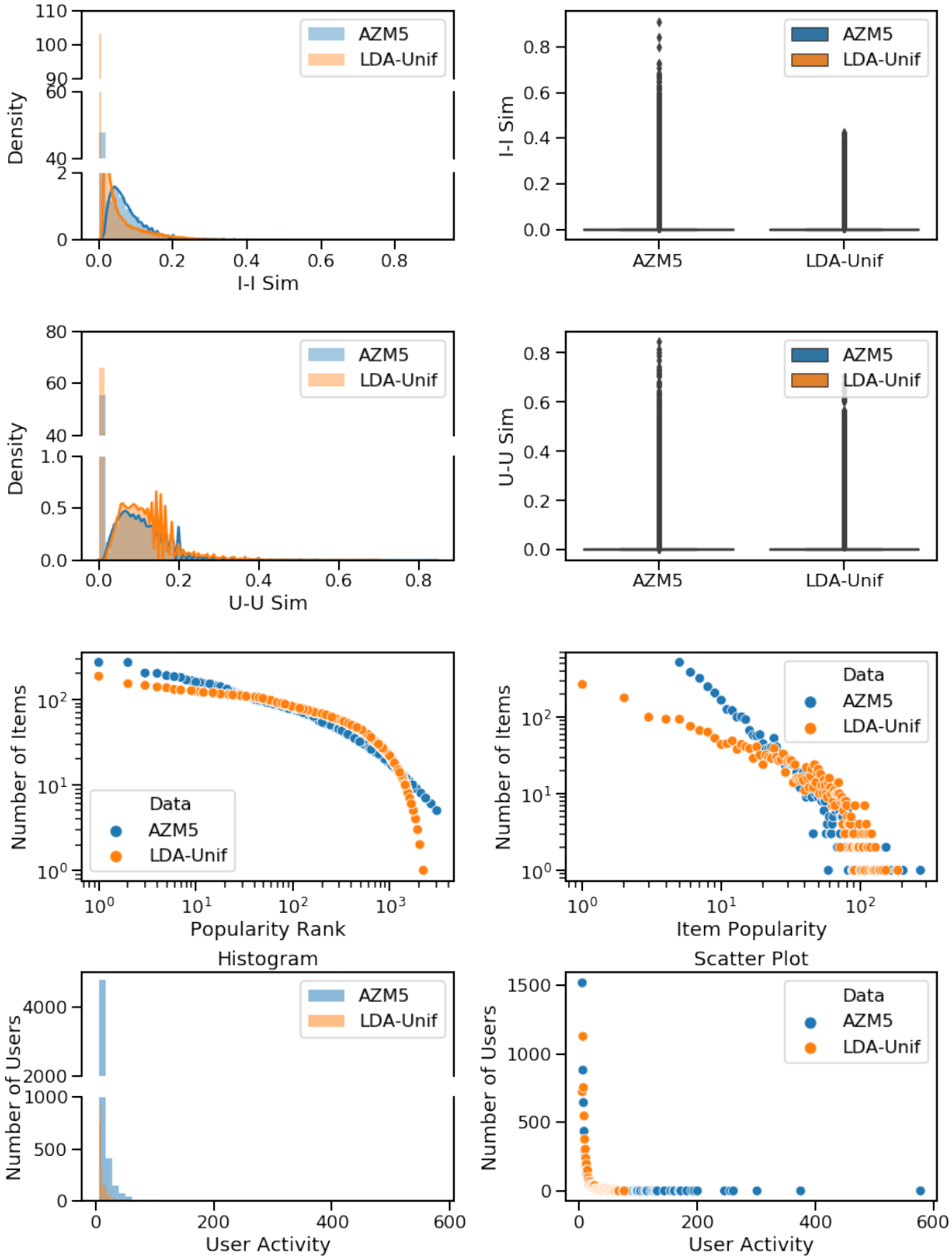

Figure 13: Key statistics of LDA-Unif model optimized to AZM5 data.

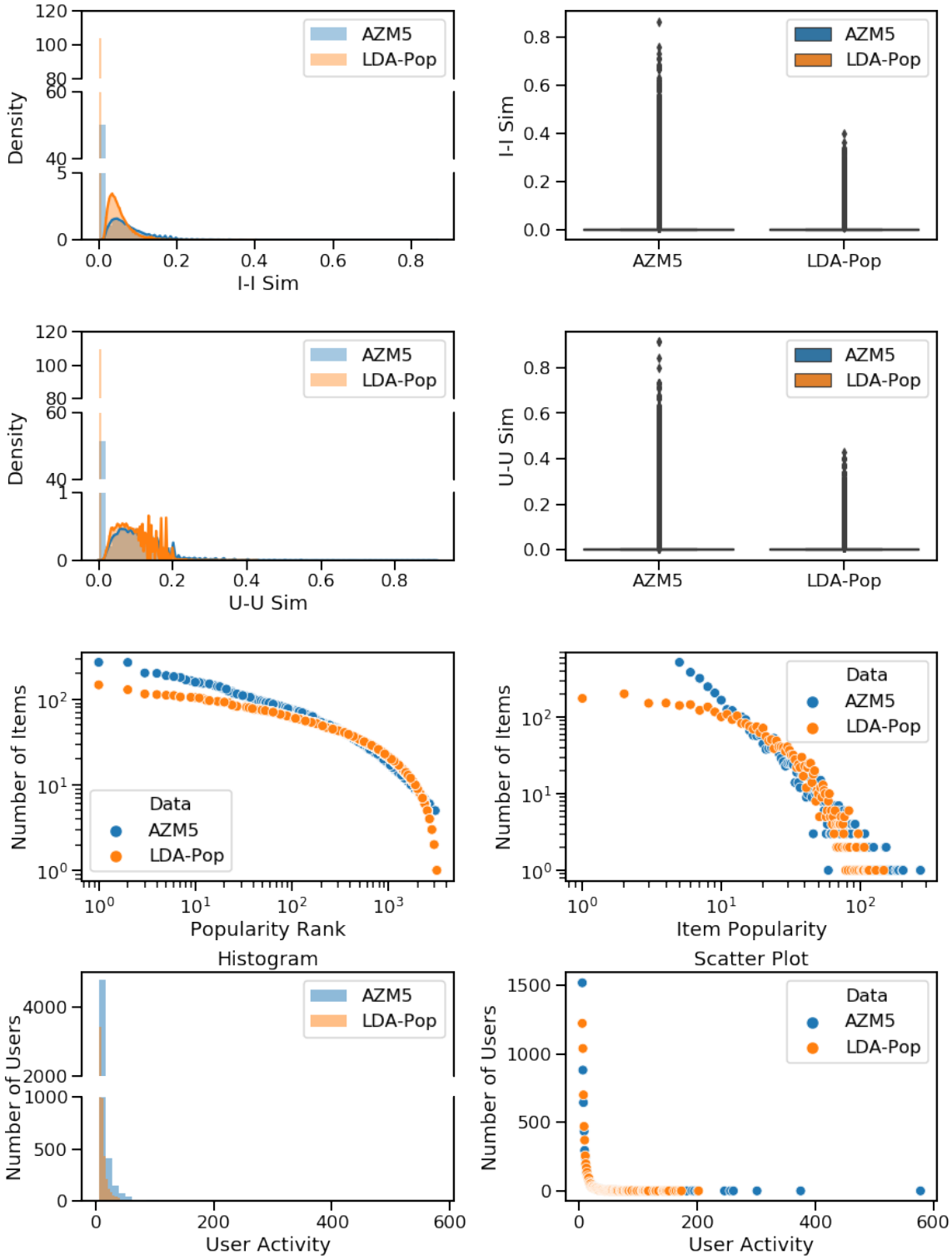

Figure 14: Key statistics of LDA-Pop model optimized to AZM5 data.

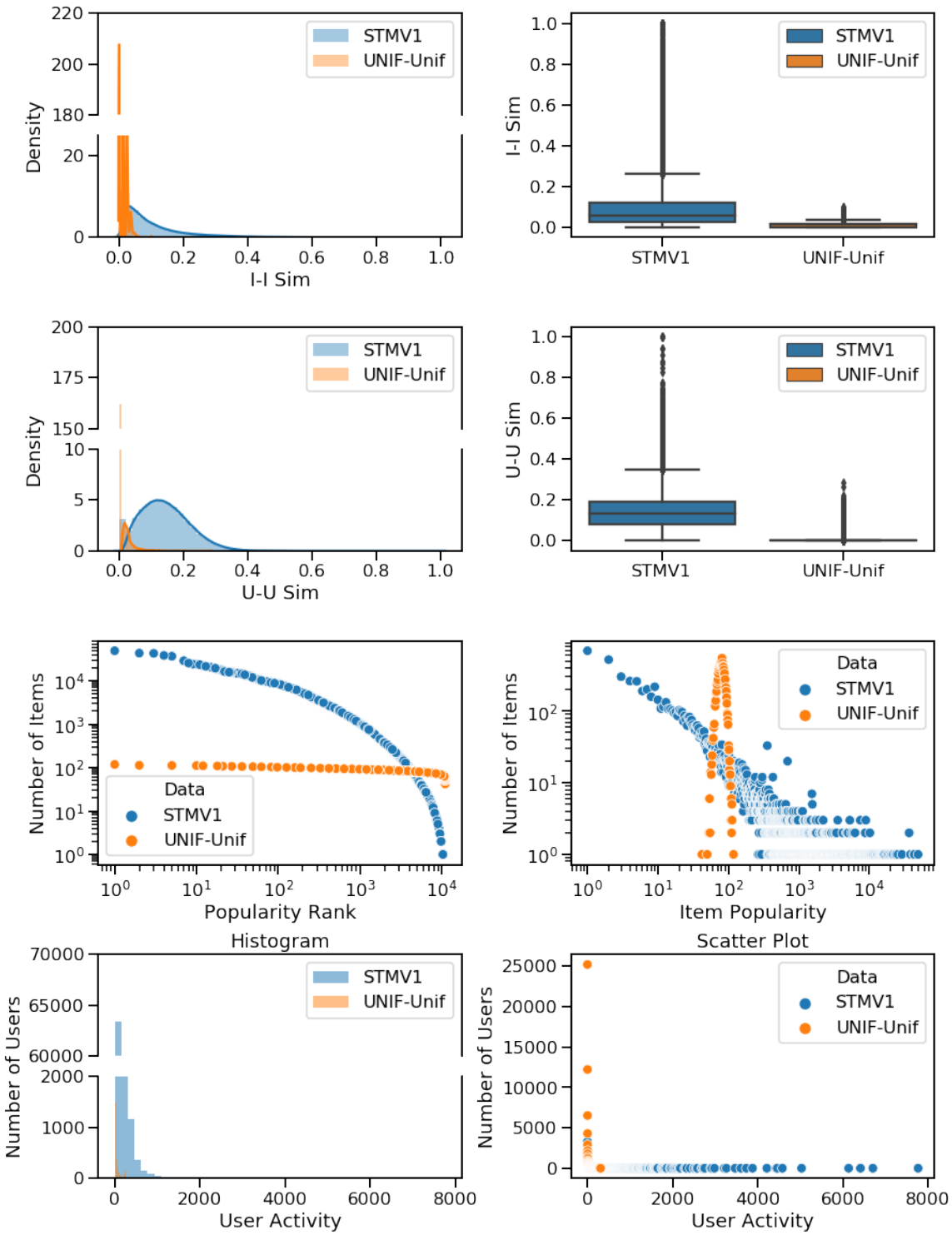

Figure 15: Key statistics of Unif-Unif model optimized to STMV1 data.

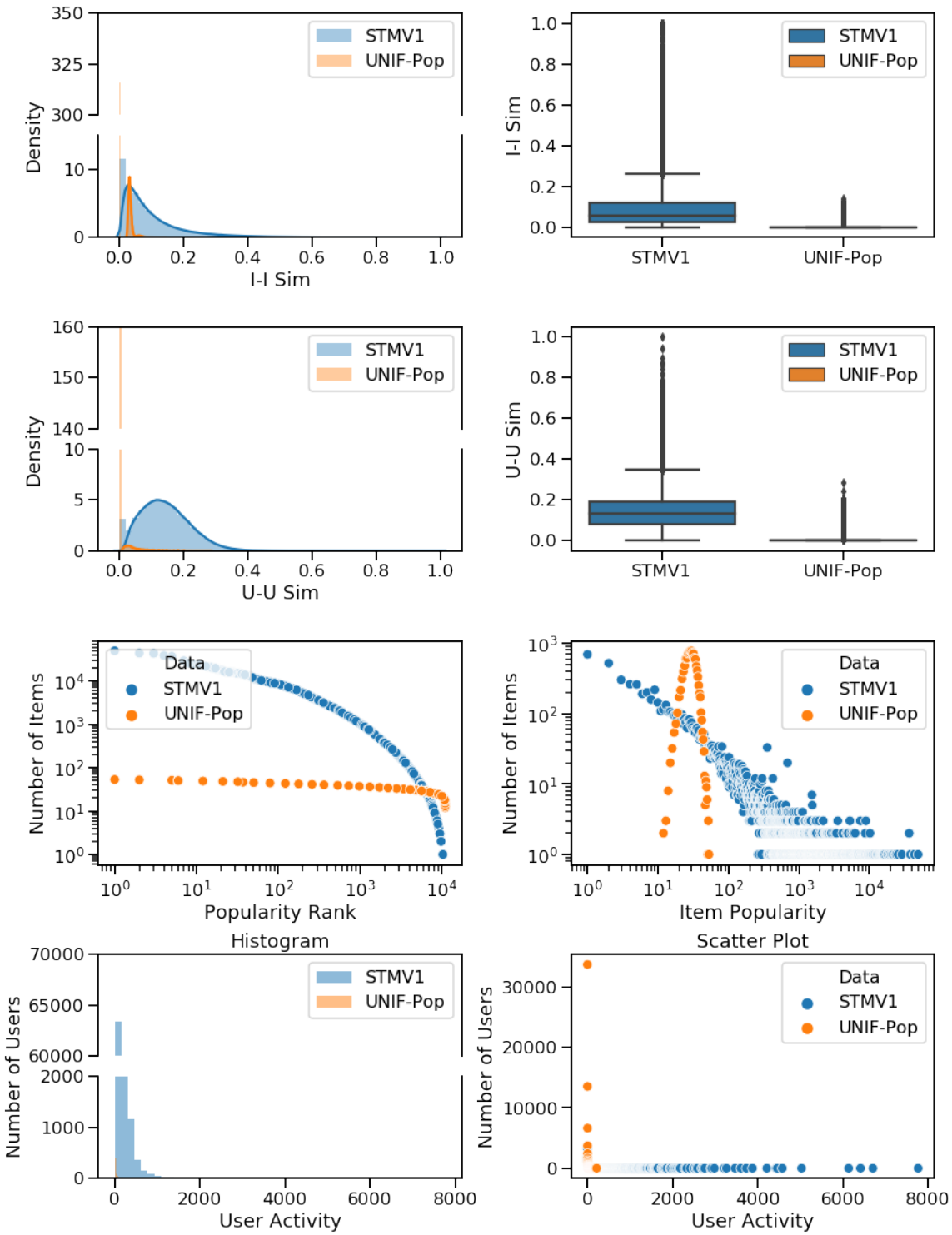

Figure 16: Key statistics of Unif-Pop model optimized to STMV1 data.

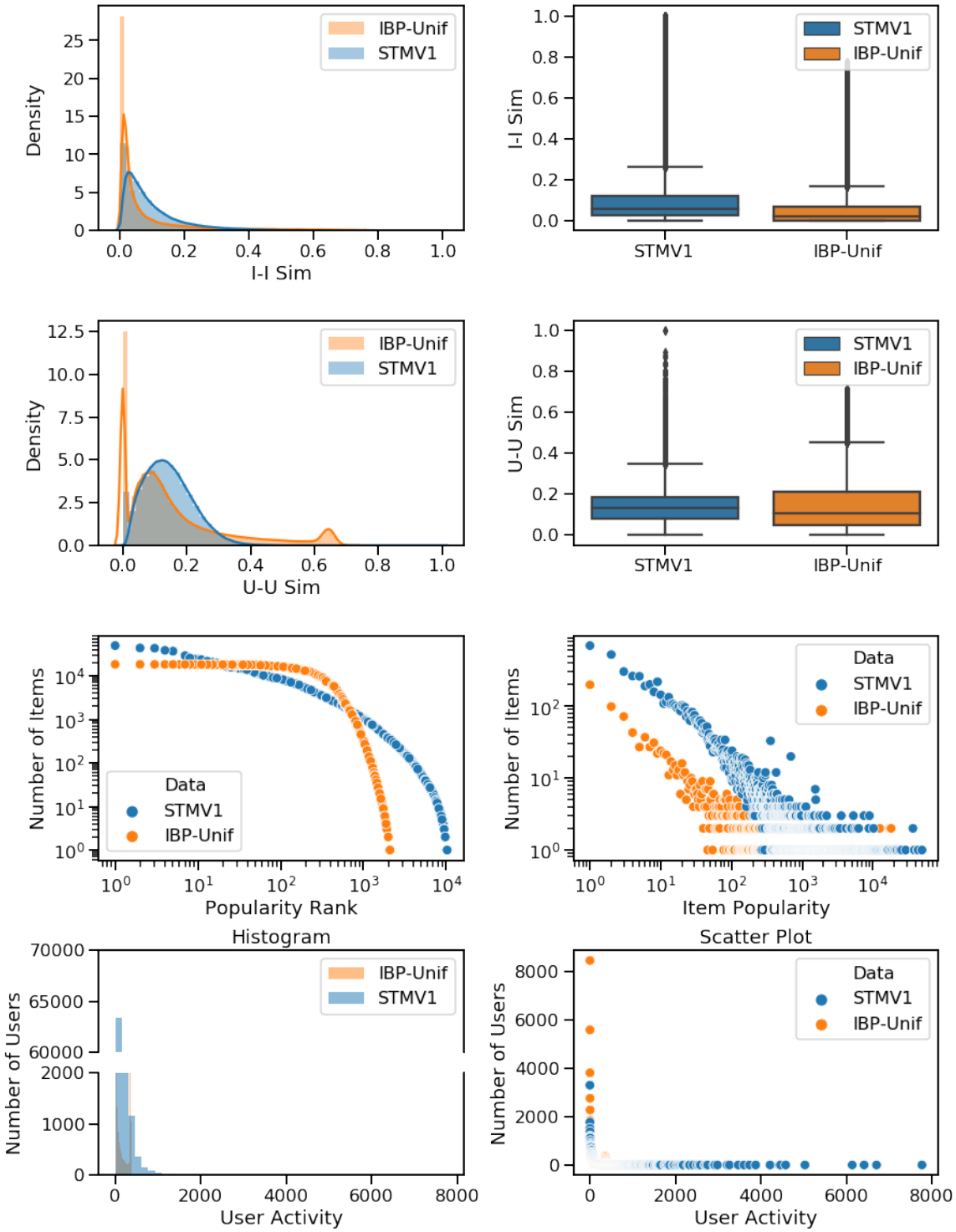

Figure 17: Key statistics of IBP-Unif model optimized to STMV1 data.

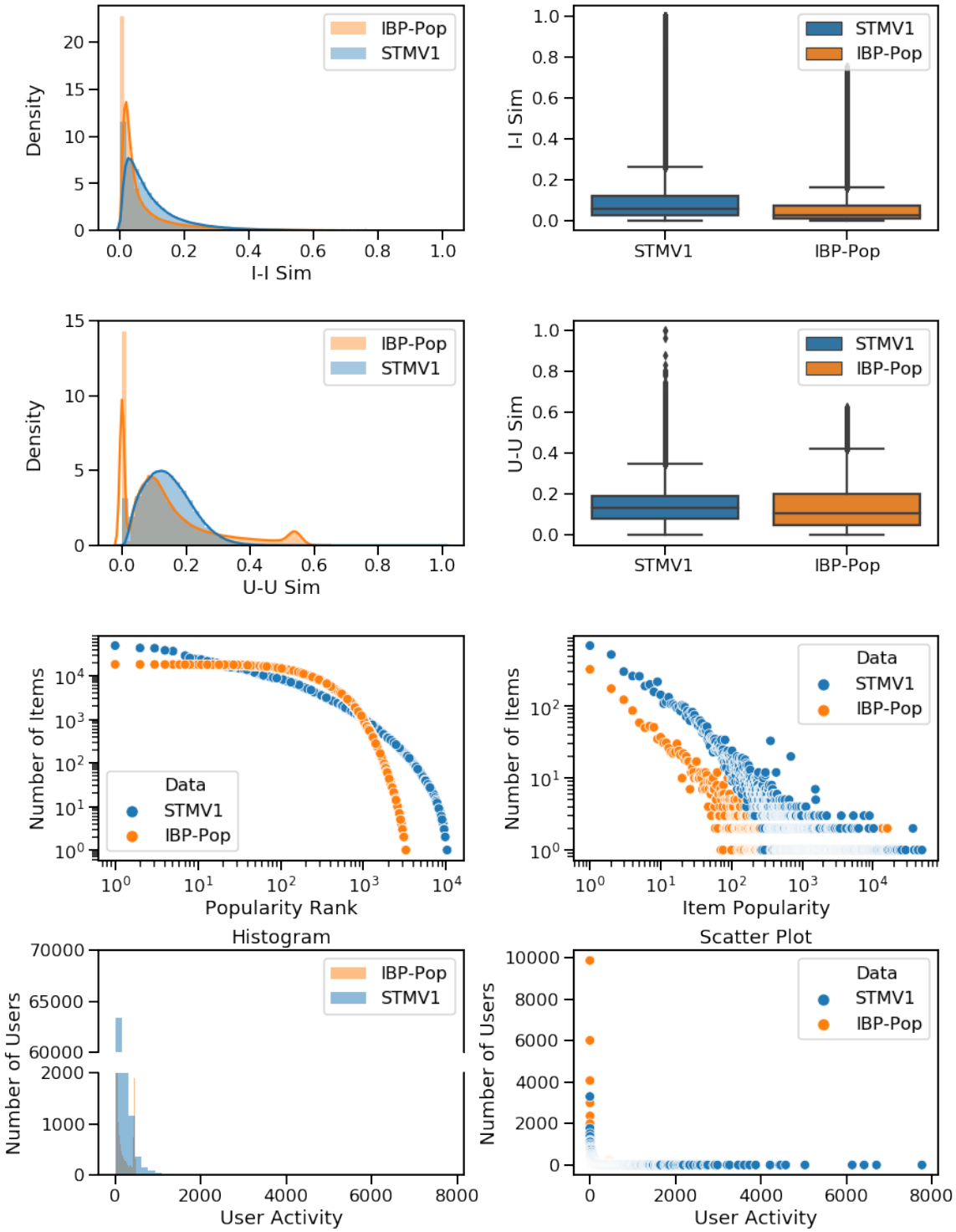

Figure 18: Key statistics of IBP-Pop model optimized to STMV1 data.

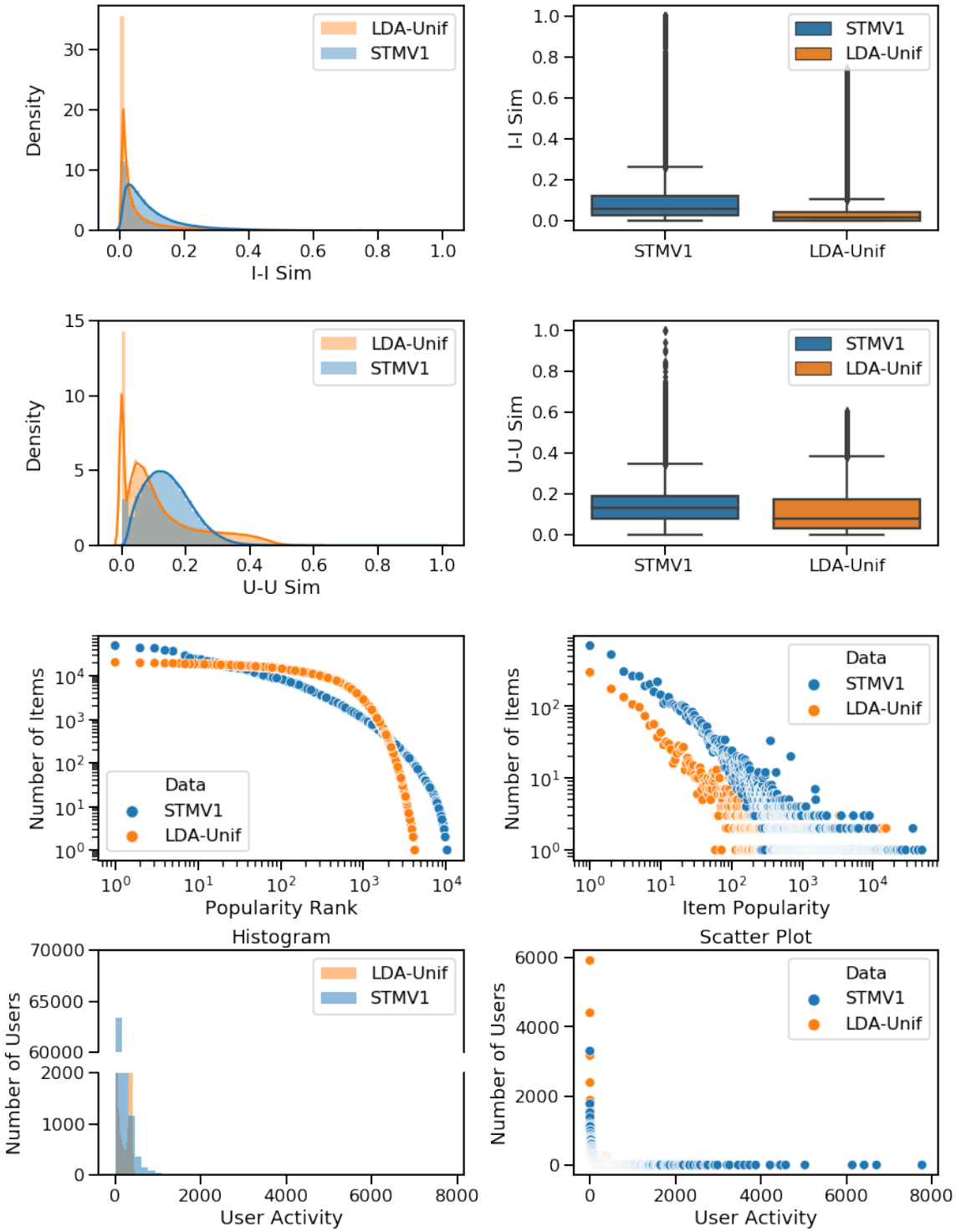

Figure 19: Key statistics of LDA-Unif model optimized to STMV1 data.

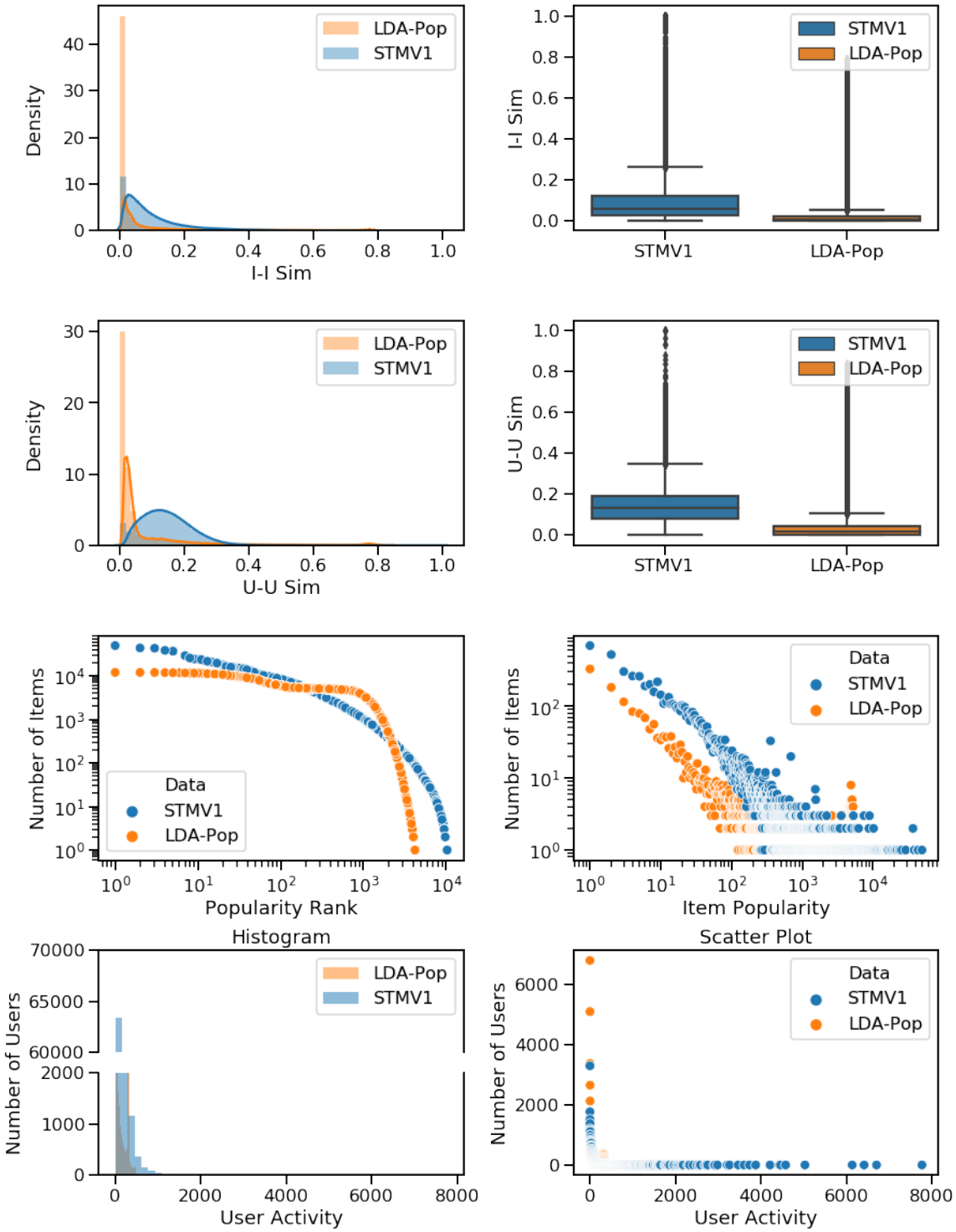

Figure 20: Key statistics of LDA-Pop model optimized to STMV1 data.

Findings: model parameters optimized for average relative loss can produce simulated data sets that capture all four statistics of reference data sets reasonably well, while parameters optimized for one statistic do not necessarily result in good fit on other statistics, demonstrated in Appendix A. Both LDA and IBP models can fit well on ML1M and AZM5 data sets, and IBP-Pop model fit the best on STMV1 data set in search spaces we explored.

4.2 Simulation Results

With the optimized models, we simulate recommender evaluations, running each simulation 100 times. We report the *error* of each metric, defined as $M^{\text{obs}} - M^{\text{truth}}$ where M^{obs} is the metric value using observable test data as ground truth and M^{truth} is the metric value using true preference data.

4.2.1 Recall

Figure 21 shows Recall values on observable test data and true preference data, faceting rows and columns by preference models and combined observation samplers and reference data sets, and Figure 22 shows its error, faceting rows and columns by preference models and observation samplers. With the uniform observation sampler, recall has no bias (error is symmetrically distributed about 0), as expected since this is the assumption under which it is an unbiased estimator [16]. Recall on Unif-Pop model also exhibits no bias because the sampling probability of our popularity-weighted sampler is computed by the popularity statistic of the preference data set; when preference data is uniformly distributed, this probability is also uniformly across all items, then the popularity-weighted sampler reduced to a uniform sampler.

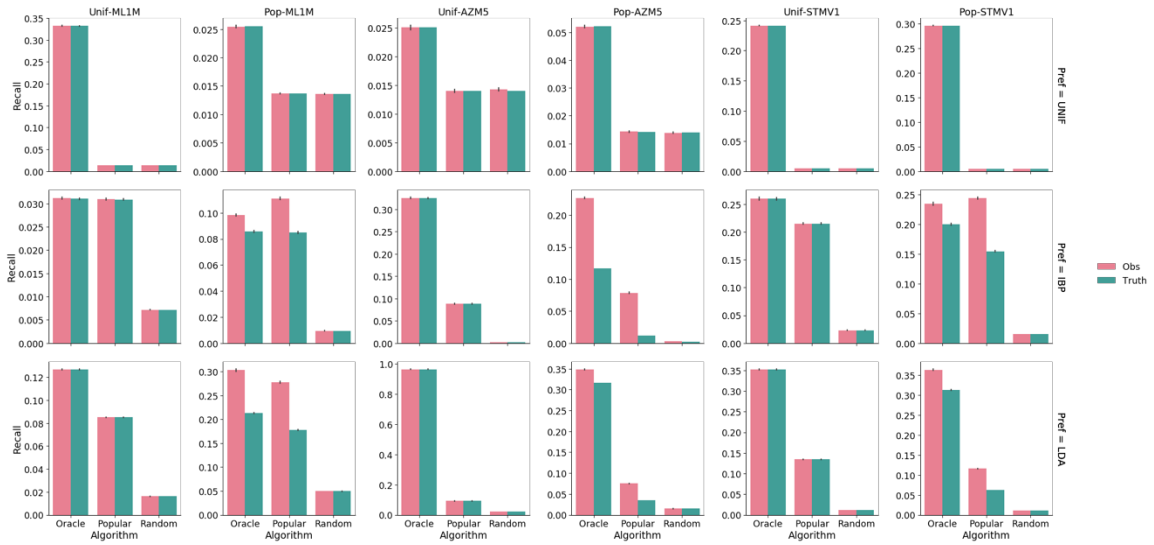

Figure 21: Recall with observable data and true preference data.

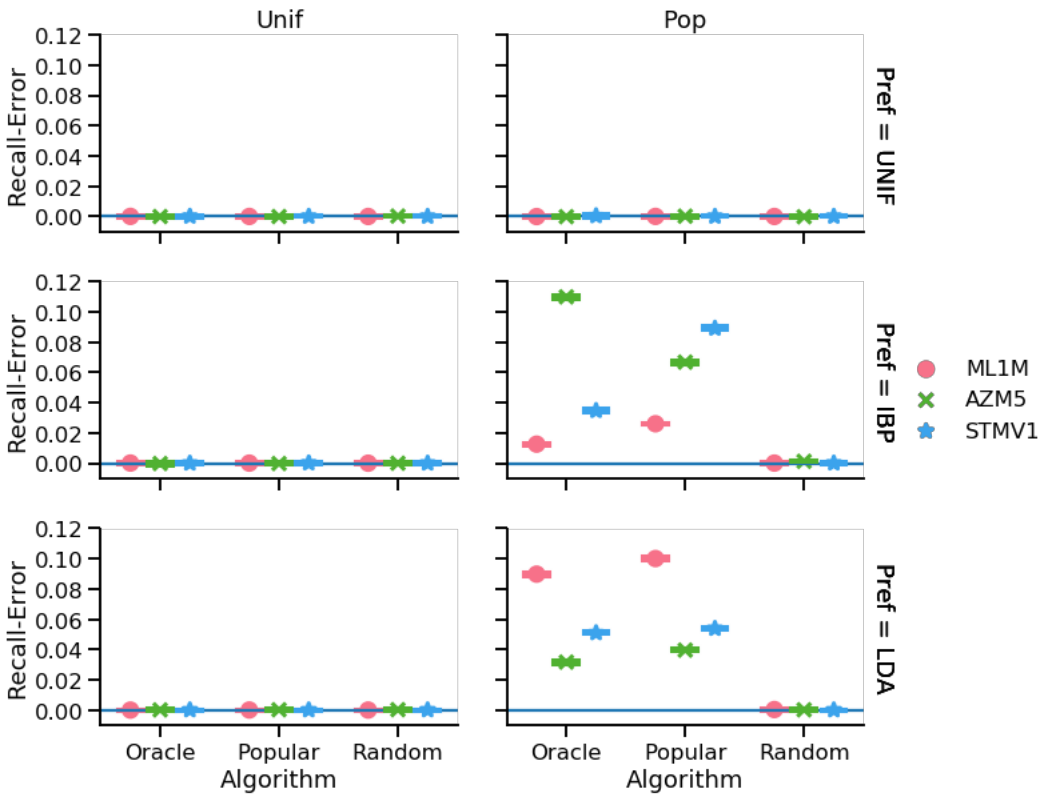

Figure 22: Error in Recall (Recall^{obs} - Recall^{truth})**.**

When observations are popularity-biased, however, observed data tends to overestimate true recall. On simulated data sets optimized with respect to ML1M and STMV1, the popular recommender outperforms the oracle recommender on the

observable data sets while comparable to the oracle recommender on the true preference data sets. This may indicate that the popularity effect can exacerbate Recall biases to more severely overestimate the Popular recommender's effectiveness.

4.2.2 Precision

Figure 23 and Figure 24 show Precision and the error in estimating it. Observed data generally underestimates precision substantially. Since our simulation models do not have users consuming irrelevant items, the set of relevant items in the true data is a superset of the observed items, increasing the opportunity for recommendations to be "correct" on the true preference data. For all models, observed data underestimates precision more for the Oracle recommender than for Popular (Oracle has smaller negative errors than Popular); this calls into question an experiment's ability to assess the performance of algorithms relative to each other, particularly when one performs extremely well.

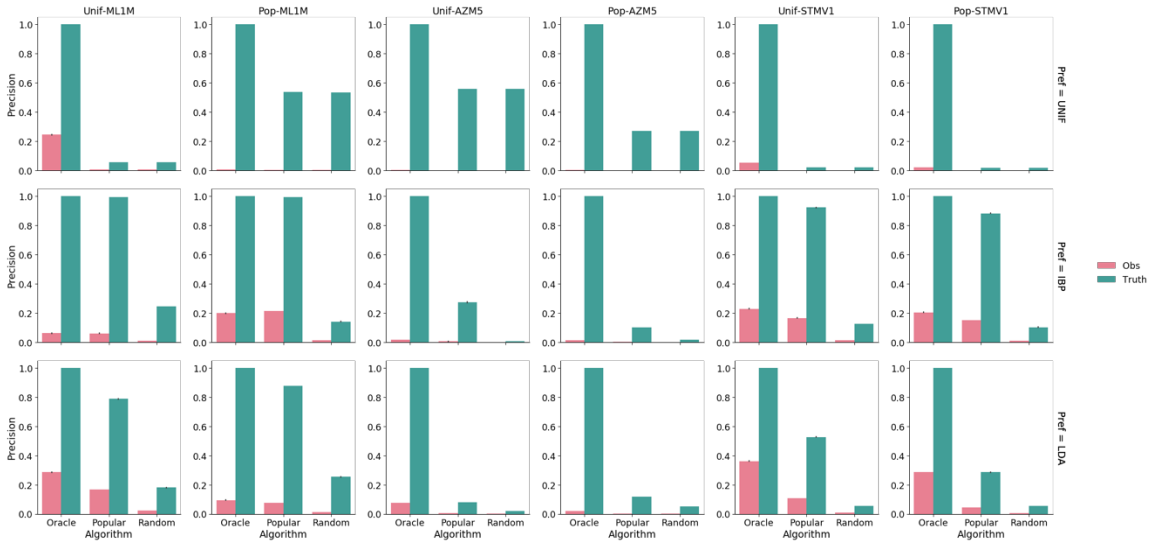

Figure 23: Precision with observable data and ground true data.

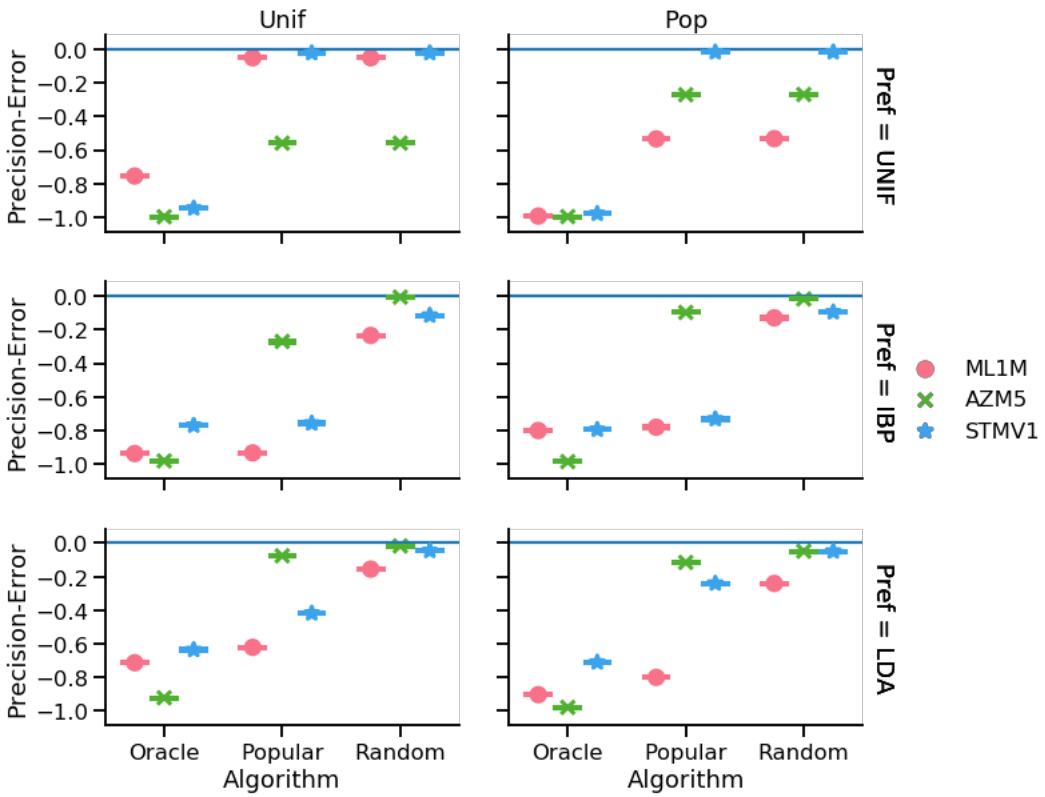

Figure 24: Error in Precision (Precision^{obs} - Precision^{truth})**.**

4.2.3 MRR

Figure 25 and Figure 26 show mean reciprocal rank and its estimation error. MRR exhibits approximately the same relevant patterns as precision; observable data underestimates true MRR (error is distributed below 0). On simulated data sets generated by IBP-Pop model optimizing with respect to ML1M and STMV1, the Popular recommender outperforms the Oracle recommender on the observable data set while comparable to the Oracle recommender on the true preference data set.

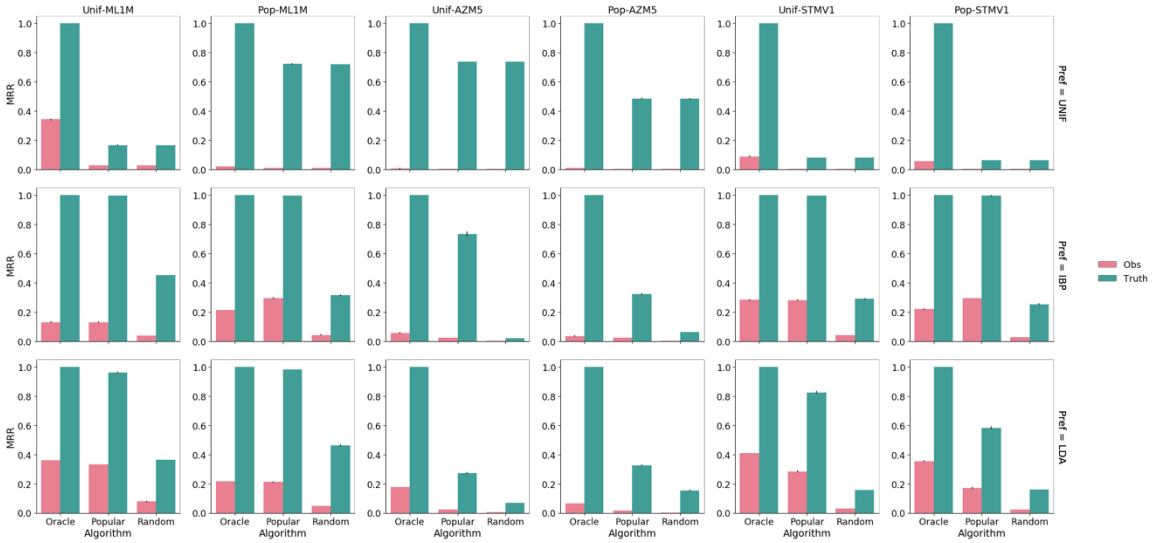

Figure 25: MRR with observable data and ground true data.

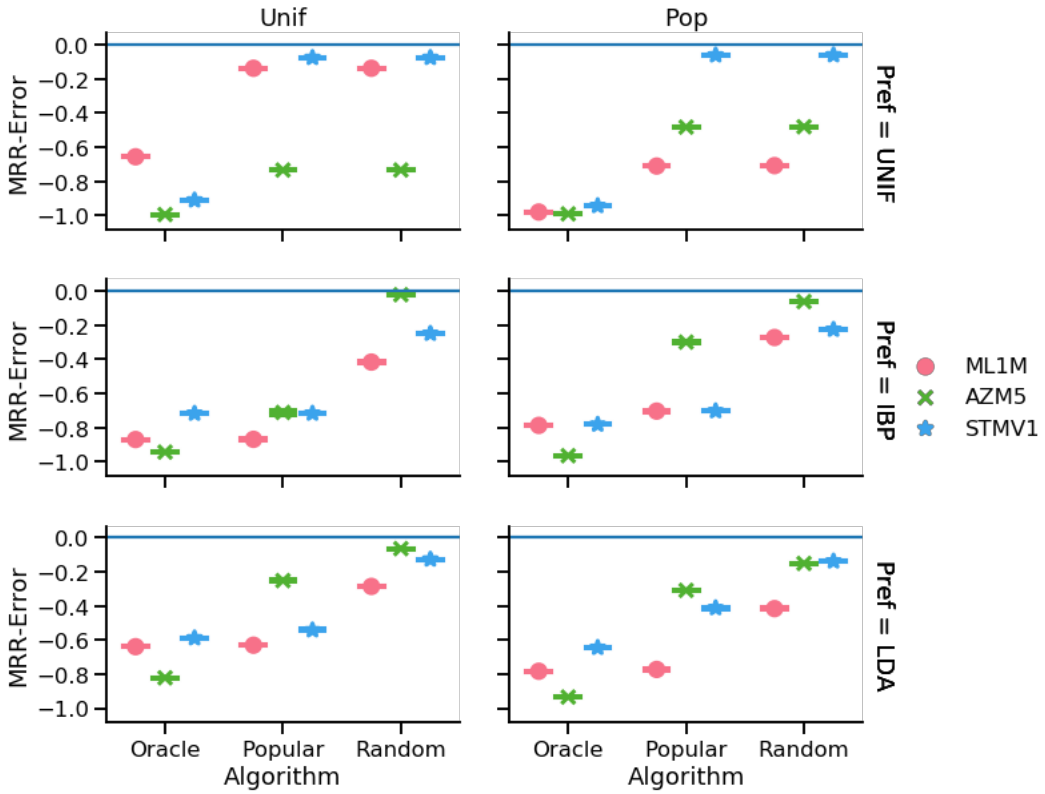

Figure 26: Error in MRR (MRR^{obs} - MRR^{truth})**.**

4.2.4 nDCG

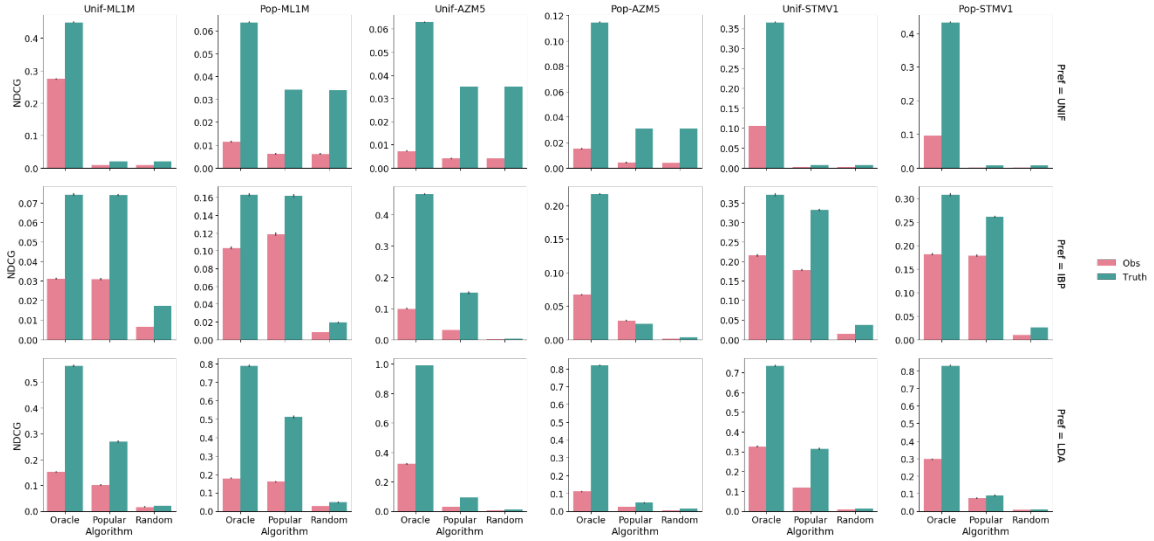

Figure 27: nDCG with observable data and ground true data.

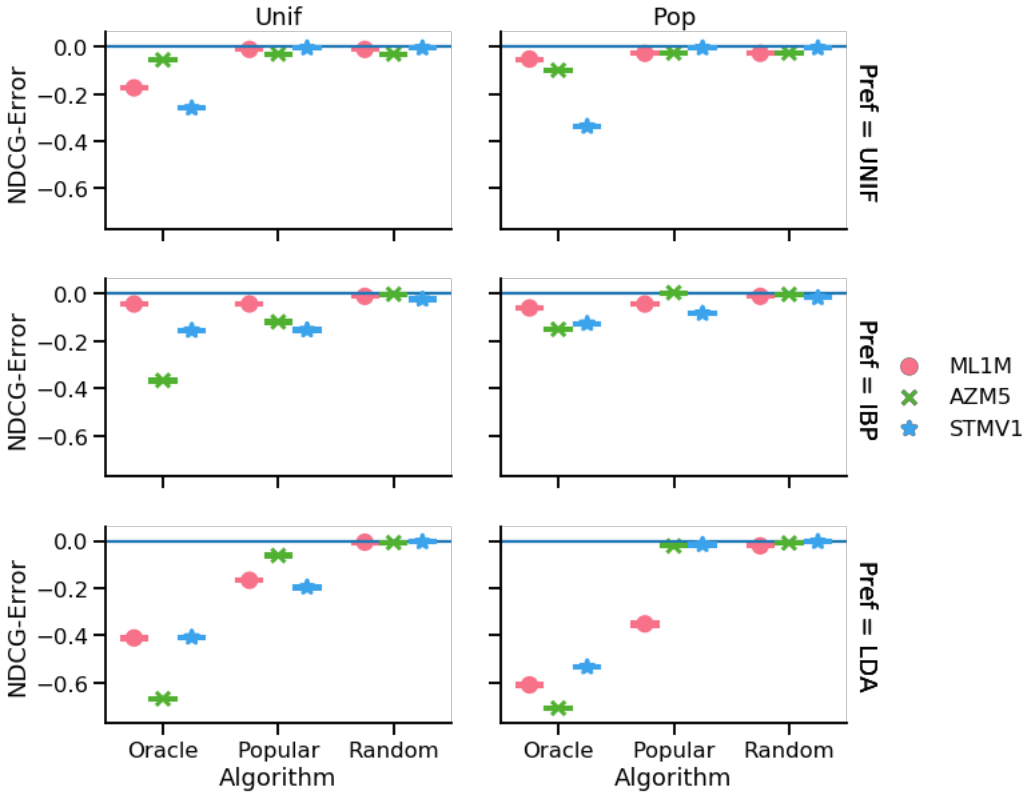

Figure 28: Error in nDCG ($nDCG^{obs}$ - $nDCG^{truth}$).

Figure 27 and Figure 28 show nDCG and its error. It is also biased, but its particular biases vary more between experimental conditions and recommenders. For example, errors for the Oracle recommender are greater than -0.4 for most reference data sets if preference models are Unif and IBP, while errors are less than -0.4 if preference model is LDA. This inconsistency means it would likely be more difficult to correct for missing data errors in evaluations using nDCG.

4.2.5 Summary

Findings: simulations validate Steck's paper [16] that Recall is unbiased when relevant items are missing uniformly at random. When observations are popularity-biased, observed data sets tend to overestimate true Recall, particularly for the Popular recommender. Observed data generally underestimates true Precision due in part to our model assumption that users only consume relevant items, and it underestimates more for the Oracle recommender than for Popular. MRR manifests similar results to Precision that observed data underestimates true MRR and generally underestimates the performance of Oracle more than that of the Popular recommender. nDCG is also biased, and its errors vary between assumptions and recommenders. If an evaluation metric underestimates all algorithms equally regardless of assumptions, then this observed metric can still reliably assess the relative performance of different algorithms; this inconsistency in nDCG errors suggests that nDCG is not reliable; the sensitivity to assumptions makes it difficult to correct for missing data errors.

4.3 Algorithm Ranking

We finally look at the prevalence of *relative performance inversions*: how often would an experiment rightly conclude that the Oracle recommender is more effective

than Popular? Table 3 shows these results. While the experimental outcomes were usually correct, there were a number of cases in which they were reliably wrong. In some cases, such as IBP with Popular observations, this is may be because popularity bias severely fooled the evaluator on the observable data. LDA did not produce this effect (except MRR on ML1M), indicating that extent of this error is sensitive to assumptions. Across all data sets, there are some cases in which MRR prefers the Popular recommender over the Oracle recommender when the underlying assumption is IBP-Pop. One possible explanation is that MRR is more sensitive to the popularity effect than other metrics we experimented.

Findings: evaluations sometimes are fooled by the popularity effect to mis-ranking the Oracle and Popular recommender. Its extent is sensitive to assumptions about the data generation.

Table 3: Percentage of runs where Oracle beats Popular.

Data	Pref	Obs	P@50	Recall	MRR	nDCG
ML1M	Unif	Unif	100	100	100	100
		Pop	100	100	100	100
	IBP	Unif	87	58	47	65
		Pop	0	0	0	0
	LDA	Unif	100	100	100	100
		Pop	100	100	89	100
AZM5	Unif	Unif	100	100	99	100
		Pop	100	100	100	100
	IBP	Unif	100	100	100	100
		Pop	100	100	99	100
	LDA	Unif	100	100	100	100
		Pop	100	100	100	100
STMV1	Unif	Unif	100	100	100	100
		Pop	100	100	100	100
	IBP	Unif	100	100	86	100
		Pop	100	5	0	85
	LDA	Unif	100	100	100	100
		Pop	100	100	100	100

CHAPTER FIVE: CONCLUSIONS AND FUTURE WORK

In this thesis, we conducted simulations to estimate error and bias in the results of offline evaluations of recommendation algorithms. To make realistic reference data, we calibrated our simulation models by comparing their results to existing data sets on several key statistics. We presented simulation approaches that generate synthetic data sets comparable to reference data sets used them to empirically estimate evaluation results on commonly used evaluation metrics and recommender systems under evaluation protocols. Our simulation techniques are generalizable to a wide range of offline metrics and evaluation protocols.

5.1 Summary of Findings

We found that optimization on average relative loss of K-L divergence is capable of capturing many dynamics of three reference data sets, while optimization on a single statistic does not necessarily result in good fitting on all statistics as shown in Appendix A.

With the exception of recall in the case where it is already known to be an unbiased estimator, we find substantial error — usually underestimation — in evaluation metrics. This may be due in part to our assumption that users only consume items they like, which makes relevant items in the true preference data be superset of ones in the observable data. Most concerningly, we find that the degree of error varies between algorithms in the same data and experimental condition, undermining estimates of relative differences in algorithm performance using offline evaluation protocols. For

example, even though evaluation results on the observable data prefers the Oracle recommender over Popular recommender in most cases, it underestimate true differences in performance between these two algorithms:

$$\mathrm{M}_{\mathrm{Oracle}}^{\mathrm{obs}} - \mathrm{M}_{\mathrm{Oracle}}^{\mathrm{truth}} \leq \mathrm{M}_{\mathrm{Popular}}^{\mathrm{obs}} - \mathrm{M}_{\mathrm{Popular}}^{\mathrm{truth}} \Rightarrow \mathrm{M}_{\mathrm{Oracle}}^{\mathrm{obs}} - \mathrm{M}_{\mathrm{Popular}}^{\mathrm{obs}} \leq \mathrm{M}_{\mathrm{Oracle}}^{\mathrm{truth}} - \mathrm{M}_{\mathrm{Popular}}^{\mathrm{truth}}.$$

Evaluations are sometimes into mis-ranking algorithms. Two out of three reference data sets, Recall, MRR, nDCG consistently reject the Oracle beating Popular on the observable data under the IBP-Pop model. The LDA model, however, doesn't show this result, indicating that the extent of this effect is sensitive to assumptions.

5.2 Implications for Recommender Research

The finding that metric errors vary across algorithms on the same data and experimental conditions – and that they are more likely to underestimate the performance of a perfect recommender – casts doubt on the reliability of offline evaluations in assessing the relative differences between algorithms' performance. This forms a basis for the long-existing question: why is the popular recommender hard to beat by a personalized recommender?

Evaluations are sometimes into mis-ranking recommender systems. This gives implications for algorithm comparison using offline evaluations. Researchers who evaluate newly proposed algorithms over baseline algorithms using publicly-available data sets and offline evaluations should take into account these effects to see whether the new algorithms are picking up overall effects in data generation or truly improving individual user utility. Unfortunately, we do not yet know precisely how to do this for a specific experiment.

5.3 Limitations and Future Work

The simulations we present here are relatively simple, and in particular do not account for rating values or relative preference in any way. In future work, we can incorporate continuous preference and observation models in the data generation procedure. The observable preference can be implemented by a continuous distribution with support of $(0, 1)$, then the observable ratings can be computed by multiplying this preference matrix with a rating scale value.

In this thesis, we also do not reflect users consuming the occasional item they do not like; in future work we hope to extend these techniques to capture a wider array of user and algorithm behavior. One particularly common factor in user rating behavior is selection bias. We can model this selection bias as a random variable that depends on the interplay of the popularity effect and user continuous preferences. This can capture the case that we sometimes consume popular items even though we do like them.

Our simulations only consider static preference and observation models at a single point in time and do not account for the feedback loop of user and recommender systems. One simple extension would be integrating the data generation process with recommender systems and calibrating observable data sets with multiple subsets of reference data sets held by timestamps. For example, we could first split users and items in the reference data by timestamps, generate observable data that simulates the earliest users and items, conduct offline evaluations on different recommender systems, introduce new items to recommend and users to interact with new items using user and item dimensions in the second time period of the reference data, simulate observable data for these newly added users and items, repeat this process until the final timestamp.

Future work will also explore more recommendation algorithms, including probabilistic Oracle recommenders and collaborative filtering recommender systems. A probabilistic Oracle recommender can be implemented by randomly recommending $M\%$ relevant items on top-N recommendation list. This always gives us $M\%$ precision, allowing us to research its impact on other metrics in controlled conditions.

Simulation is a promising tool for better understanding the recommender evaluation process. Our simulations demonstrate the unreliability of offline evaluations in assessing the relative differences and ranking between algorithms. In future work, we hope to extend these simulations to explore more causes of evaluation errors.

REFERENCES

[1] Boise State's Research Computing Department, *R2: Dell HPC Intel E5v4 (High Performance Computing Cluster),* Boise, ID, USA: Boise State University, 2017.

[2] K. Falk, Practical Recommender Systems, Shelter Island, NY, USA: Manning Publications Co., 2019.

[3] M. D. Ekstrand, J. T. Riedl and J. A. Konstan, "Collaborative Filtering Recommender Systems," *Foundations and Trends in Human-Computer Interaction,* vol. 4, pp. 81-173, 2 2011.

[4] R. Kohavi, R. Longbotham, D. Sommerfield and R. M. Henne, "Controlled Experiments on the Web: Survey and Practical Guide," *Data Mining and Knowledge Discovery,* vol. 18, pp. 140-181, 01 2 2009.

[5] G. Shani and A. Gunawardana, "Evaluating Recommendation Systems," in *Recommender Systems Handbook,* F. Ricci, L. Rokach, B. Shapira and P. B. Kantor, Eds., Boston, MA: Springer US, 2011, pp. 257-297.

[6] K. Järvelin and J. Kekäläinen, "Cumulated Gain-based Evaluation of IR Techniques," *ACM Transactions on Information Systems,* vol. 20, no. 4, pp. 422-446, Oct. 2002.

[7] P. B. Kantor and E. Voorhees, "Report on the TREC-5 Confusion Track," in *Proceedings of the Fifth Text REtrieval Conference,* Gaithersburg, Maryland, USA, 1997.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen and J. T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Transactions on Information Systems*, vol. 22, pp. 5-53, 1 2004.

[9] J. S. Breese, D. Heckerman and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 1998.

[10] A. Bellogín, P. Castells and I. Cantador, "Precision-oriented Evaluation of Recommender Systems: An Algorithmic Comparison," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, New York, NY, USA, 2011.

[11] M. D. Ekstrand and V. Mahant, "Sturgeon and the Cool Kids: Problems with Random Decoys for Top-N Recommender Evaluation," in *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference*, Palo Alto, CA, USA, 2017.

[12] B. M. Marlin, R. S. Zemel, S. Roweis and M. Slaney, "Collaborative Filtering and the Missing at Random Assumption," in *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, Arlington, Virginia, United States, 2007.

[13] B. M. Marlin and R. S. Zemel, "Collaborative Prediction and Ranking with Non-random Missing Data," in *Proceedings of the Third ACM Conference on Recommender Systems*, New York, NY, USA, 2009.

[14] A. Bellogín, "Recommender System Performance Evaluation and Prediction: An Information Retrieval Perspective," Universidad Autónoma de Madrid, Madrid, 2012.

[15] D. Lim, J. McAuley and G. Lanckriet, "Top-N Recommendation with Missing Implicit Feedback," in *Proceedings of the 9th ACM Conference on Recommender Systems*, New York, NY, USA, 2015.

[16] H. Steck, "Training and Testing of Recommender Systems on Data Missing Not at Random," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2010.

[17] L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard and E. Snelson, "Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising," *Journal of Machine Learning Research*, vol. 14, pp. 3207-3260, 2013.

[18] A. Swaminathan and T. Joachims, "Batch Learning from Logged Bandit Feedback through Counterfactual Risk Minimization," *Journal of Machine Learning Research*, vol. 16, pp. 1731-1755, 2015.

[19] R. Cañamares and P. Castells, "Should I Follow the Crowd?: A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, New York, NY, USA, 2018.

[20] P. Cremonesi, Y. Koren and R. Turrin, "Performance of Recommender Algorithms on Top-n Recommendation Tasks," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, New York, NY, USA, 2010.

[21] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham and S. Dollé, "Offline A/B Testing for Recommender Systems," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2018.

[22] A. J. B. Chaney, B. M. Stewart and B. E. Engelhardt, "How Algorithmic Confounding in Recommendation Systems Increases Homogeneity and

Decreases Utility," in *Proceedings of the 12th ACM Conference on Recommender Systems*, New York, NY, USA, 2018.

[23] T. L. Griffiths and Z. Ghahramani, "The Indian Buffet Process: An Introduction and Review," *Journal of Machine Learning Research*, vol. 12, pp. 1185-1224, 7 2011.

[24] Y. W. Teh and D. Görür, "Indian Buffet Processes with Power-law Behavior," in *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*, USA, 2009.

[25] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 3 2003.

[26] H. Steck, "Item Popularity and Recommendation Accuracy," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, New York, NY, USA, 2011.

[27] T. L. Griffiths and Z. Ghahramani, "Infinite Latent Feature Models and the Indian Buffet Process," in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, Cambridge, 2005.

[28] Z. Ghahramani, T. L. Griffiths and P. Sollich, "Bayesian Nonparametric Latent Feature Models," *Bayesian Statistics*, vol. 8, pp. 1-25, 2007.

[29] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 19:1-19:19, 12 2015.

[30] R. He and J. McAuley, "Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering," in *Proceedings of the 25th International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland, 2016.

[31] A. Pathak, K. Gupta and J. McAuley, "Generating and Personalizing Bundle Recommendations on Steam," in *Proceedings of the 40th International*

ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 2017.

[32] S. Kullback, Information Theory and Statistics, Mineola, New York, USA: Dover Publications, 1997, p. 432.

[33] T. E. Oliphant, Guide to NumPy: 2nd Edition, Scotts Valley, CA, USA: CreateSpace Independent Publishing Platform, 2015, p. 364.

[34] T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. Vinícius, cmmalone, C. Schröder, nel215, N. Campos, T. Young, S. Cereda, T. Fan, rene-rex, K. (. Shi, J. Schwabedal, carlosdanielcsantos, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. Estève, L. Besson, M. Cherti, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller and A. Fabisch, *scikit-optimize/scikit-optimize: v0.5.2*, 2018.

[35] M. D. Ekstrand, "The LKPY Package for Recommender Systems Experiments: Next-Generation Tools and Lessons Learned from the LensKit Project," *CoRR*, vol. abs/1809.03125, 2018.

[36] A. Gelman, H. S. Stern, J. B. Carlin, D. B. Dunson, A. Vehtari and D. B. Rubin, "Model Checking," in *Bayesian Data Analysis*, New York, NY, USA, Chapman and Hall/CRC, 2013, pp. 141-163.

APPENDIX A

A.1 Calibration Results

This section presents calibration results optimized for each single statistic.

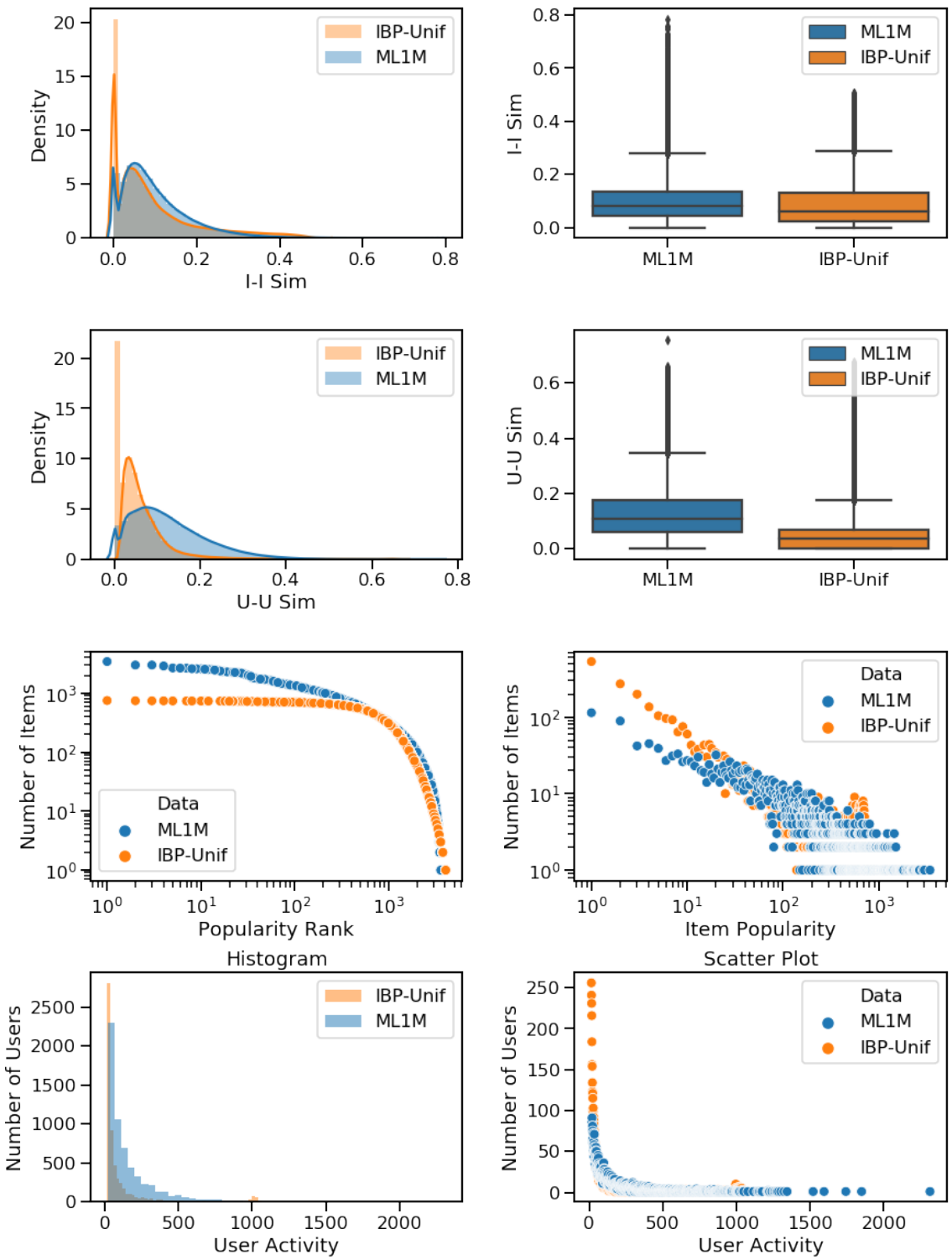

Figure 29: IBP-UNIF optimized for I-I Sim on ML1M.

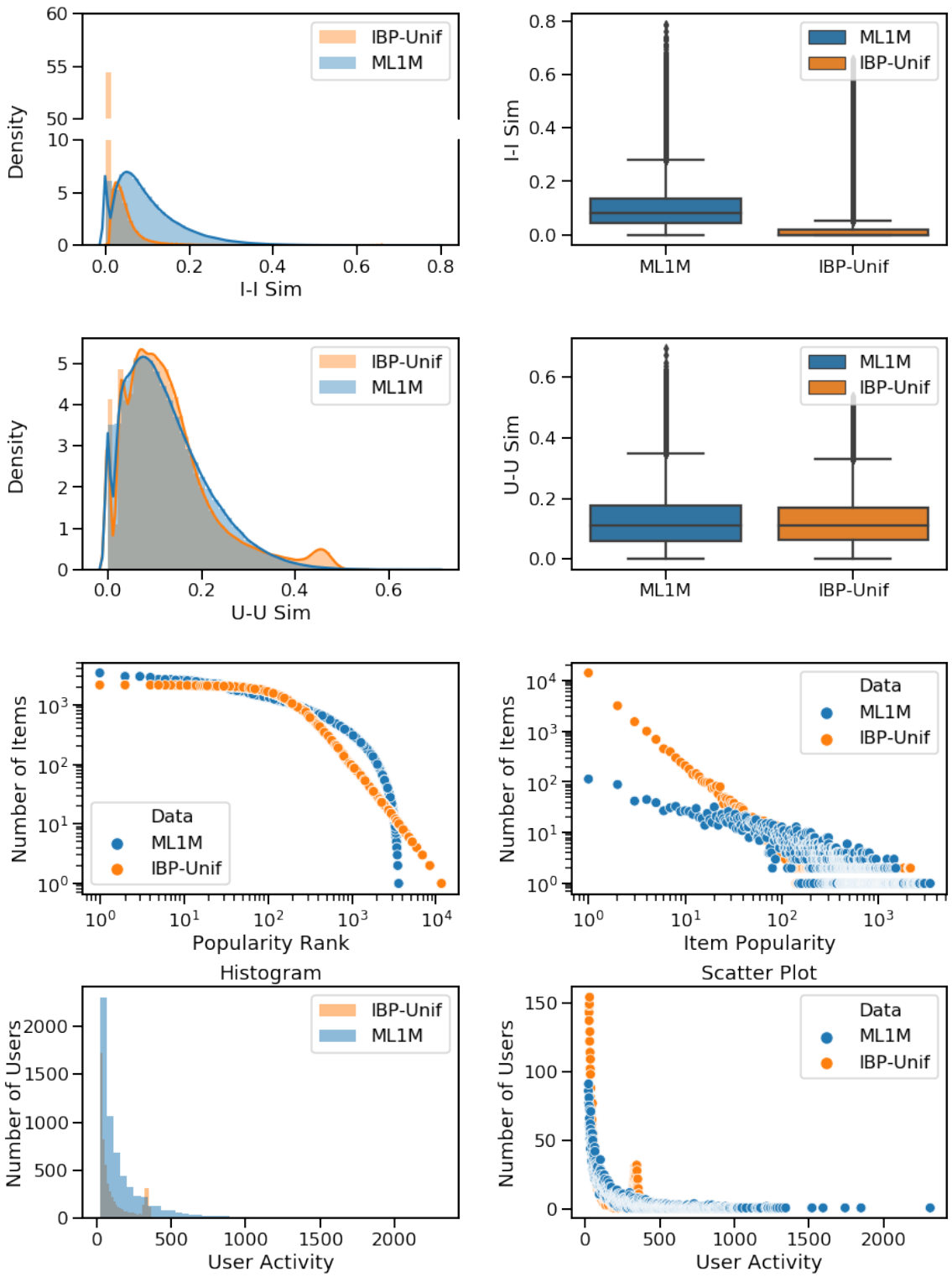

Figure 30: IBP-UNIF optimized for U-U Sim on ML1M.

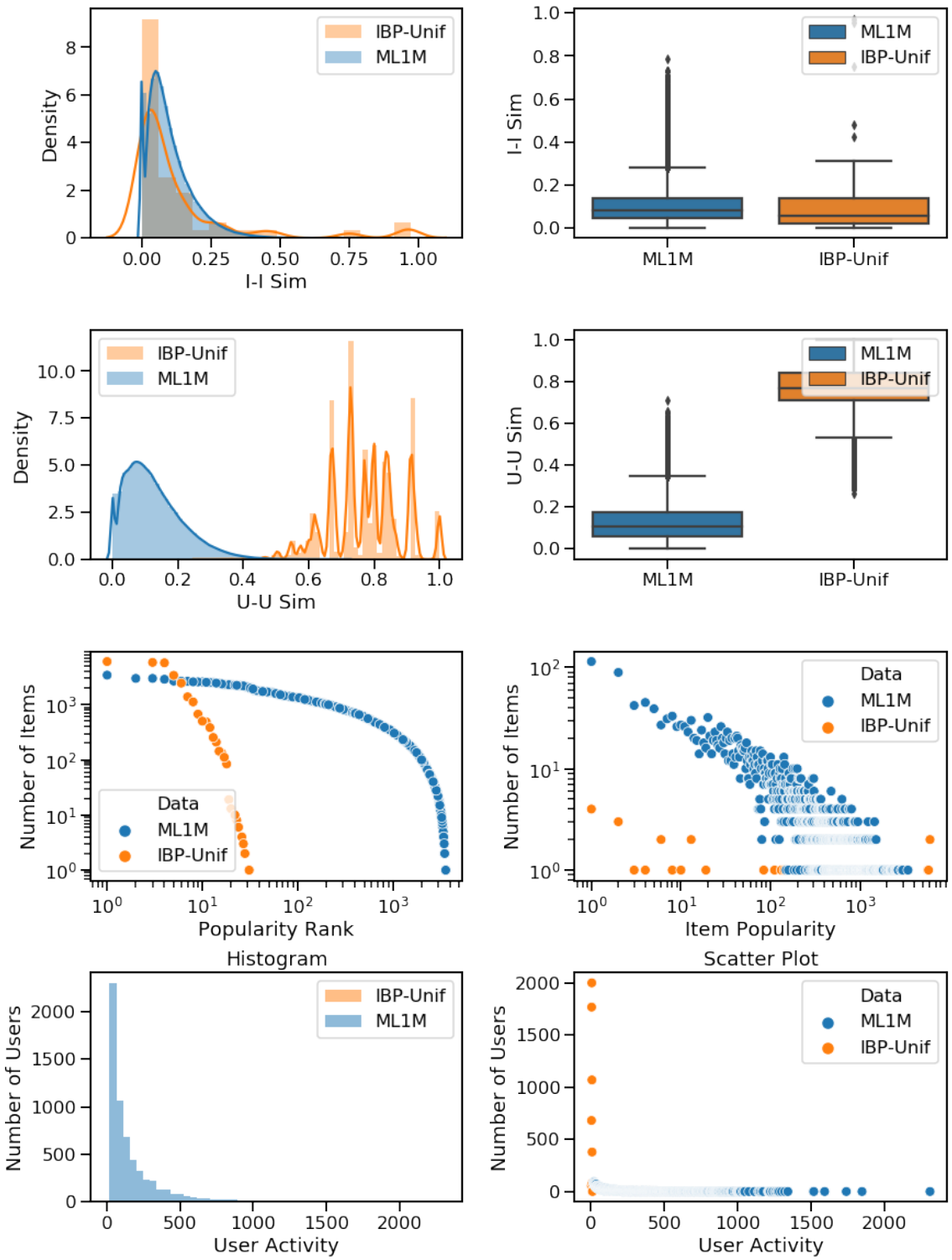

Figure 31: IBP-UNIF optimized for Item Pop on ML1M.

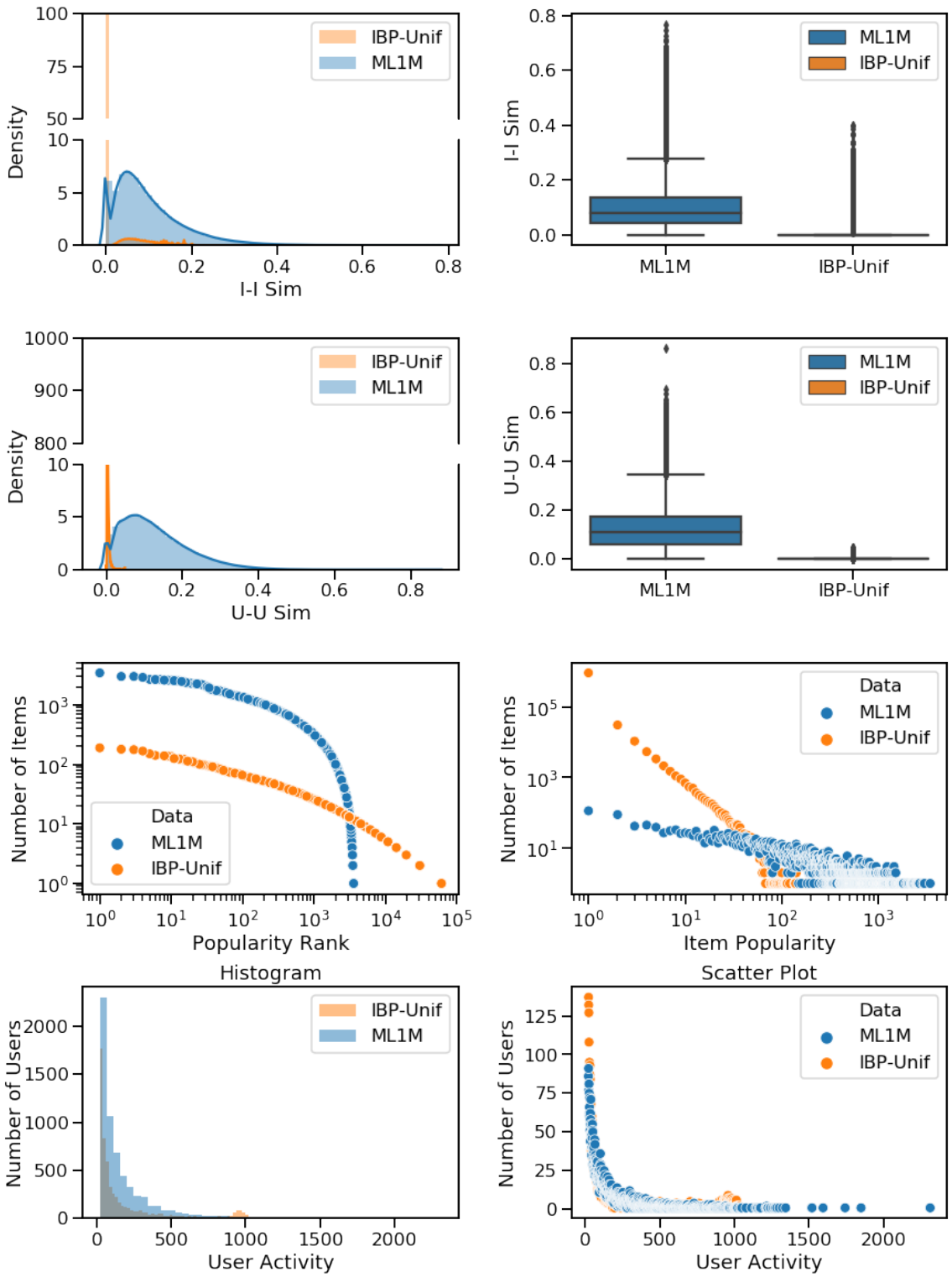

Figure 32: IBP-UNIF optimized for User Act on ML1M.

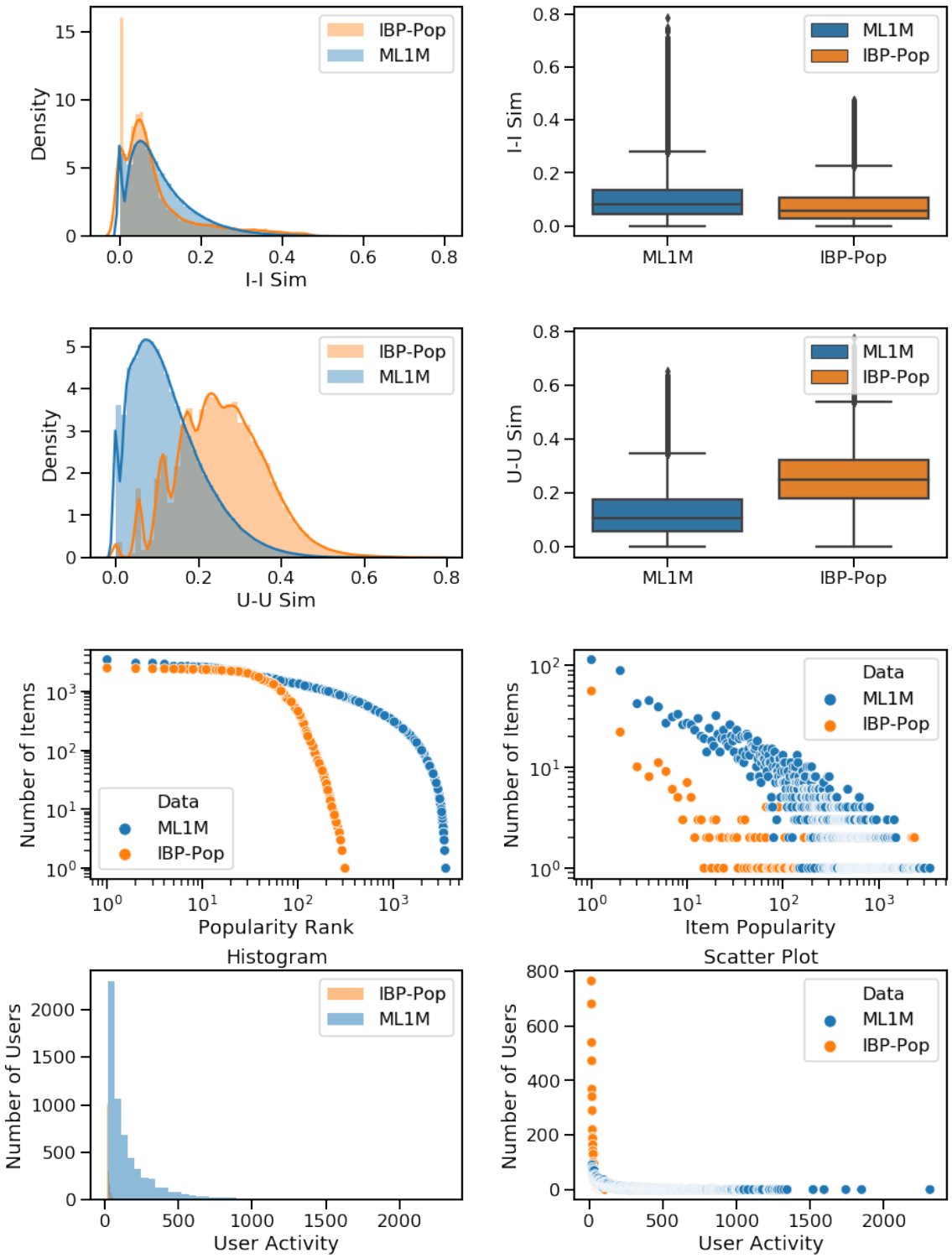

Figure 33: IBP-Pop optimized for I-I Sim on ML1M.

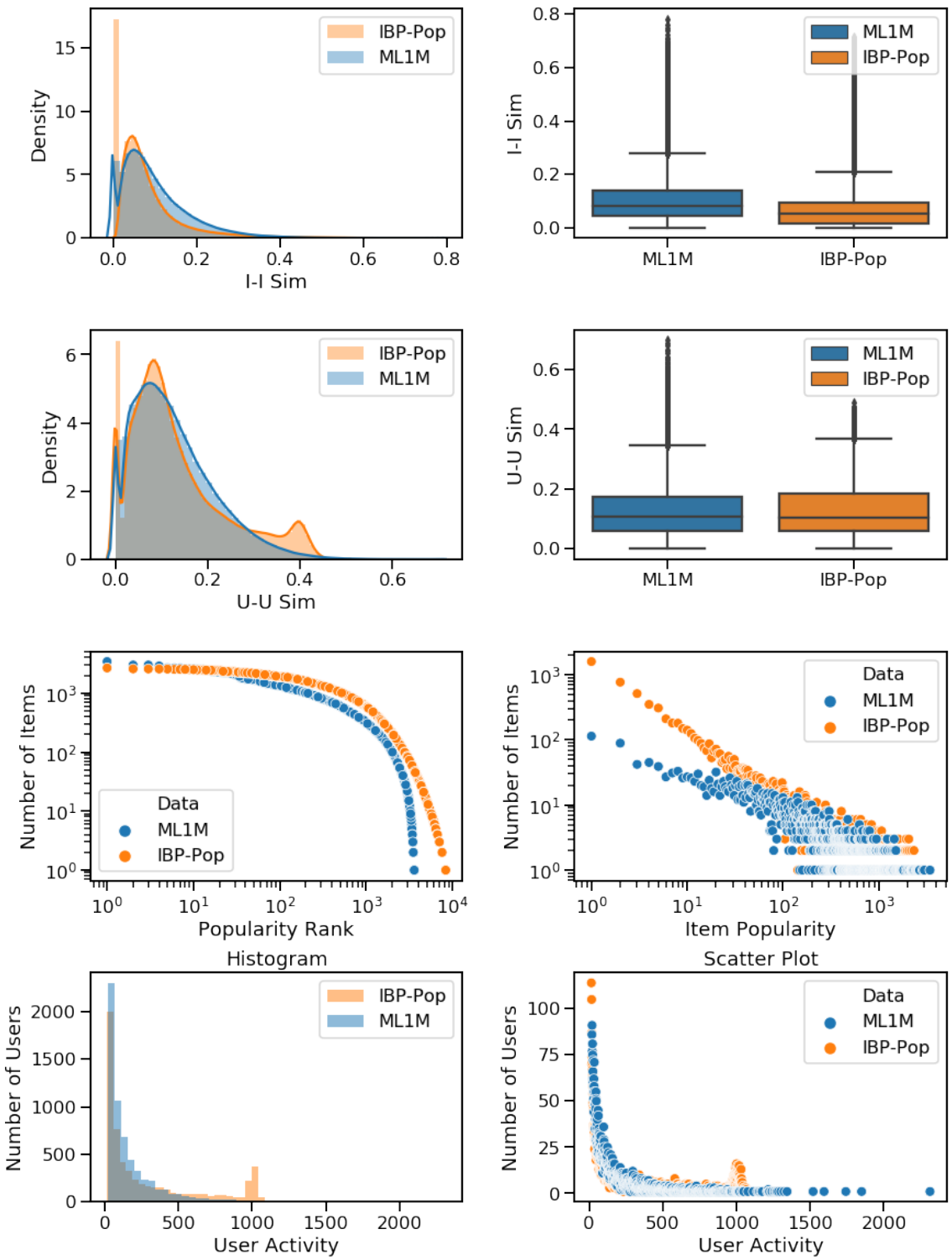

Figure 34: IBP-Pop optimized for U-U Sim on ML1M.

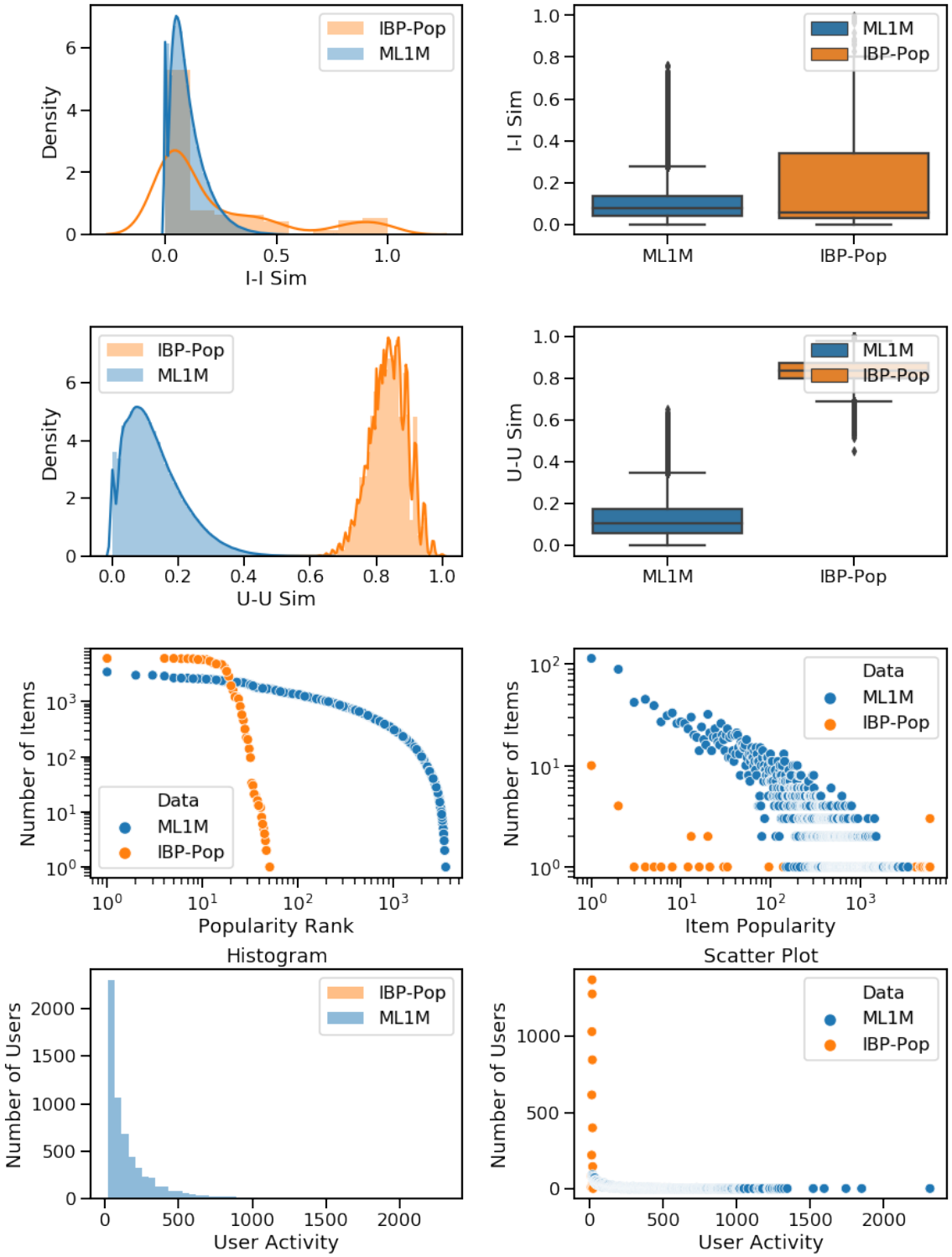

Figure 35: IBP-Pop optimized for Item Pop on ML1M.

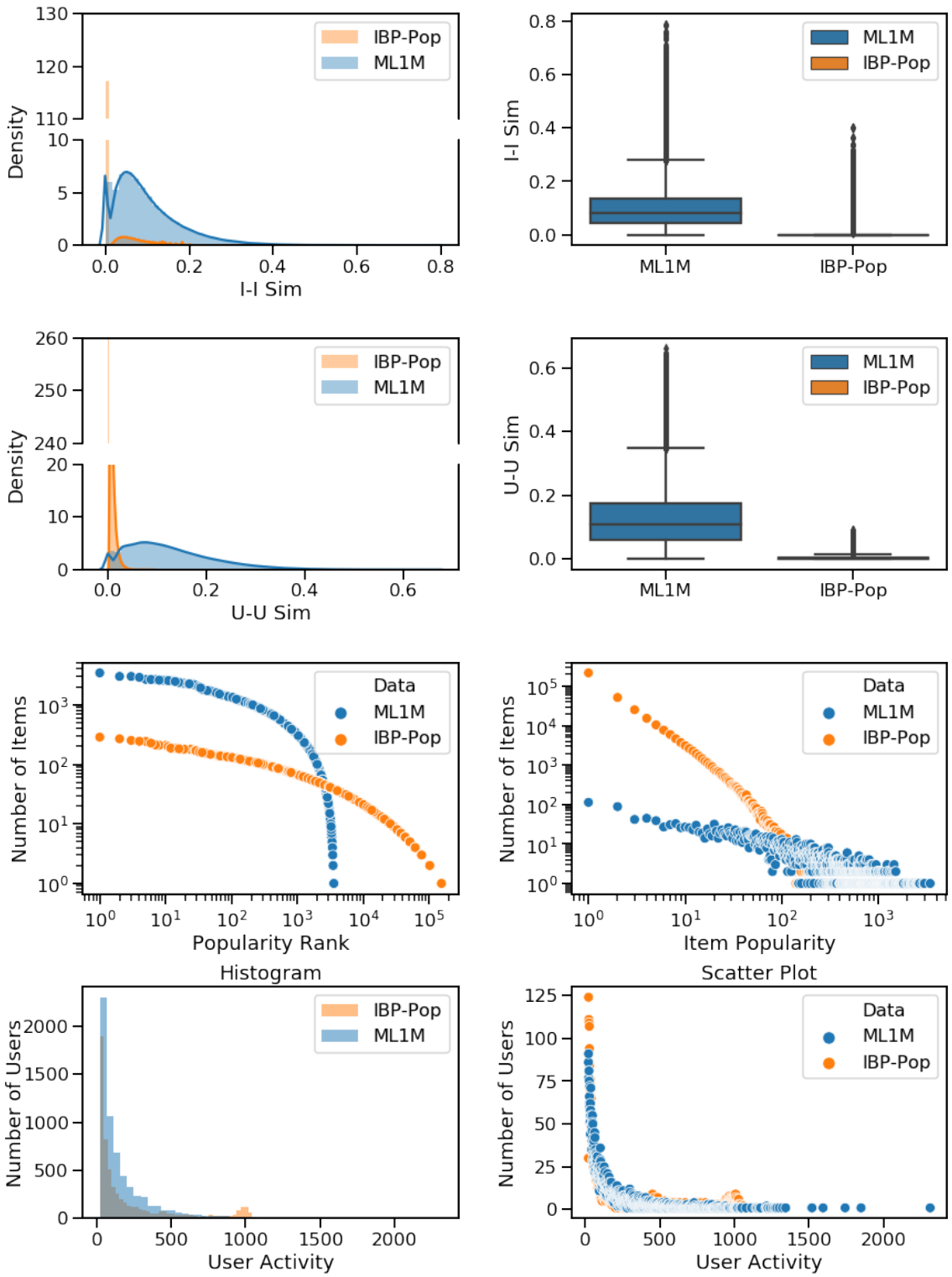

Figure 36: IBP-Pop optimized for User Act on ML1M.

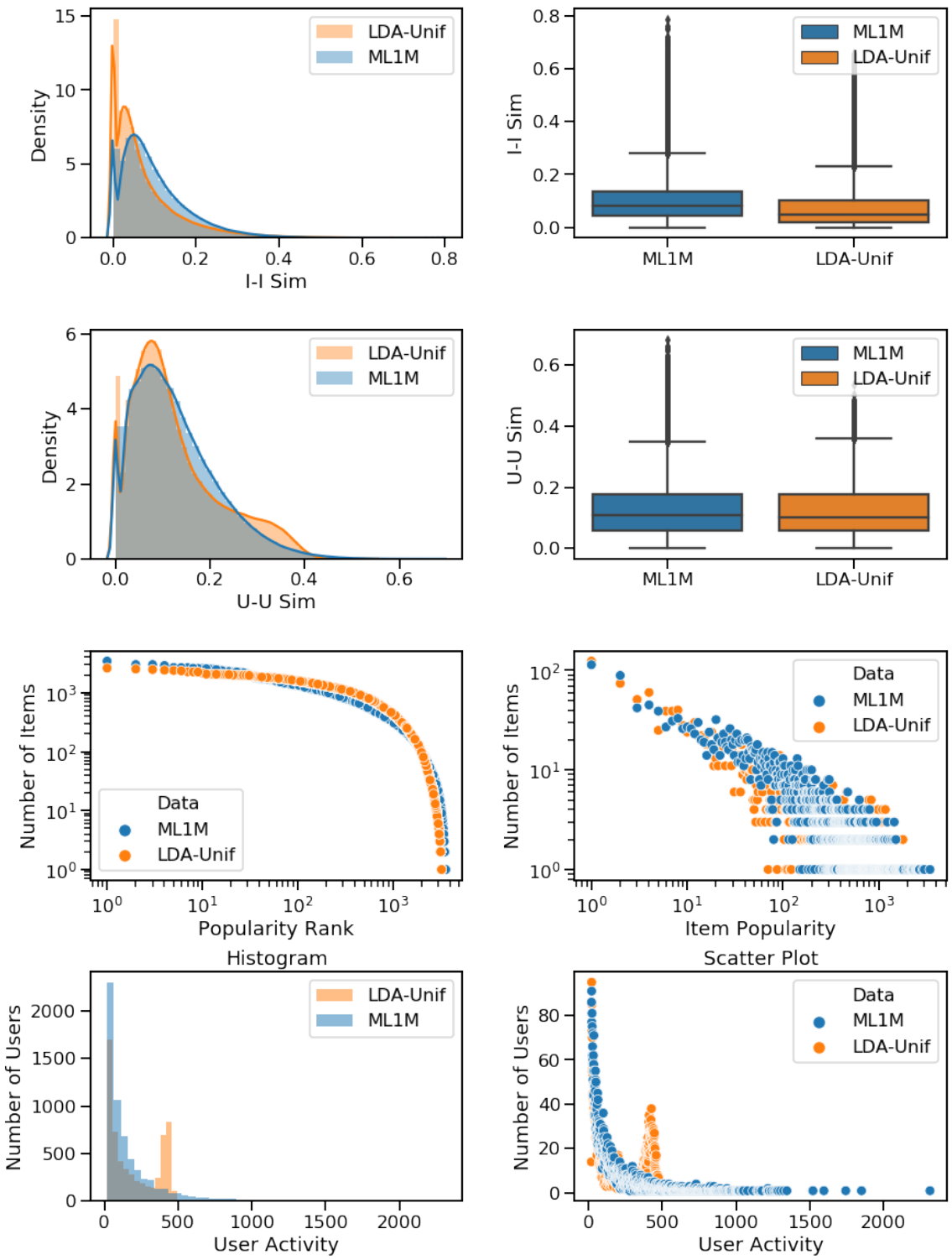

Figure 37: LDA-Unif optimized for U-U Sim on ML1M.

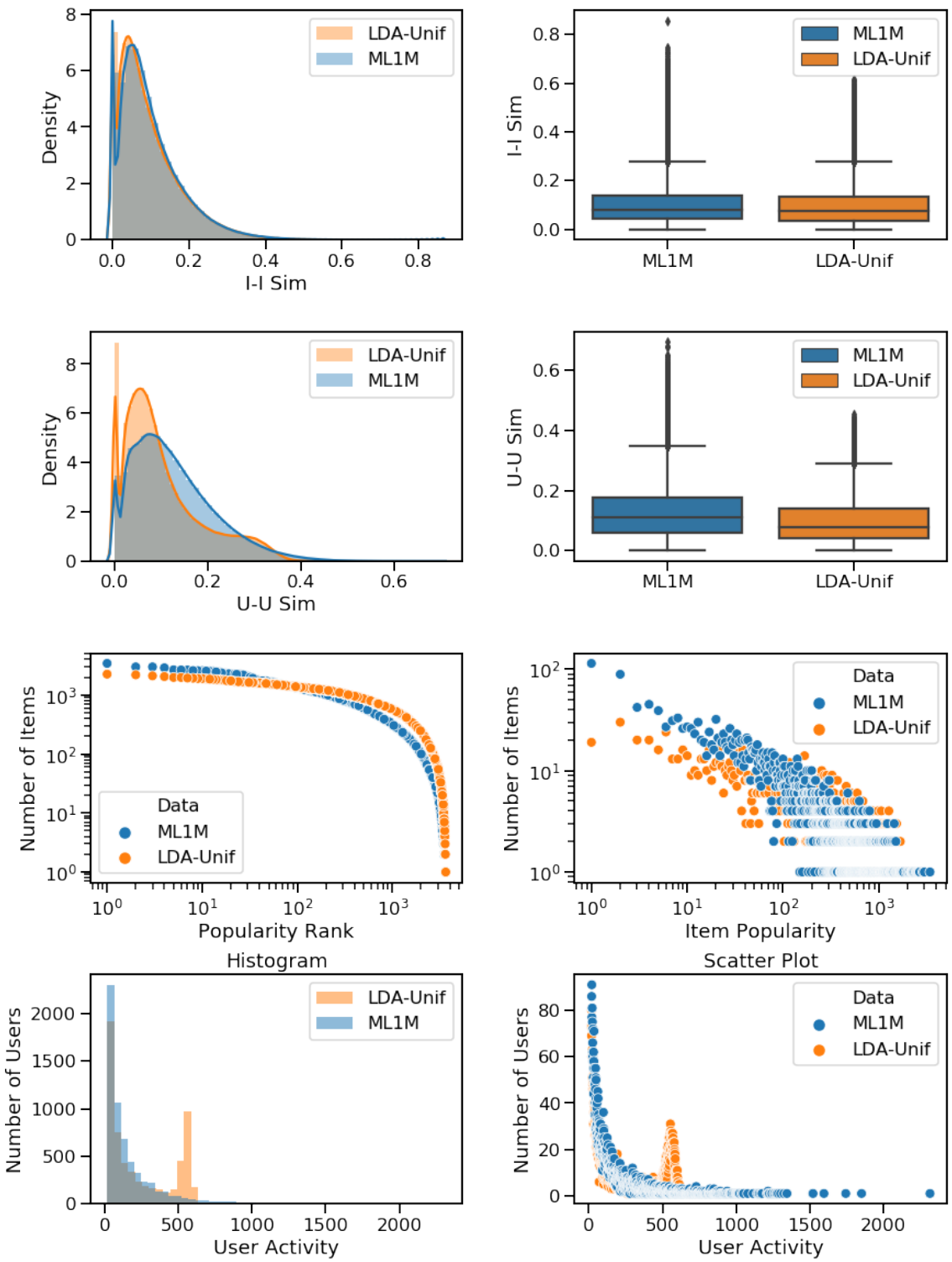

Figure 38: LDA-Unif optimized for I-I Sim on ML1M.

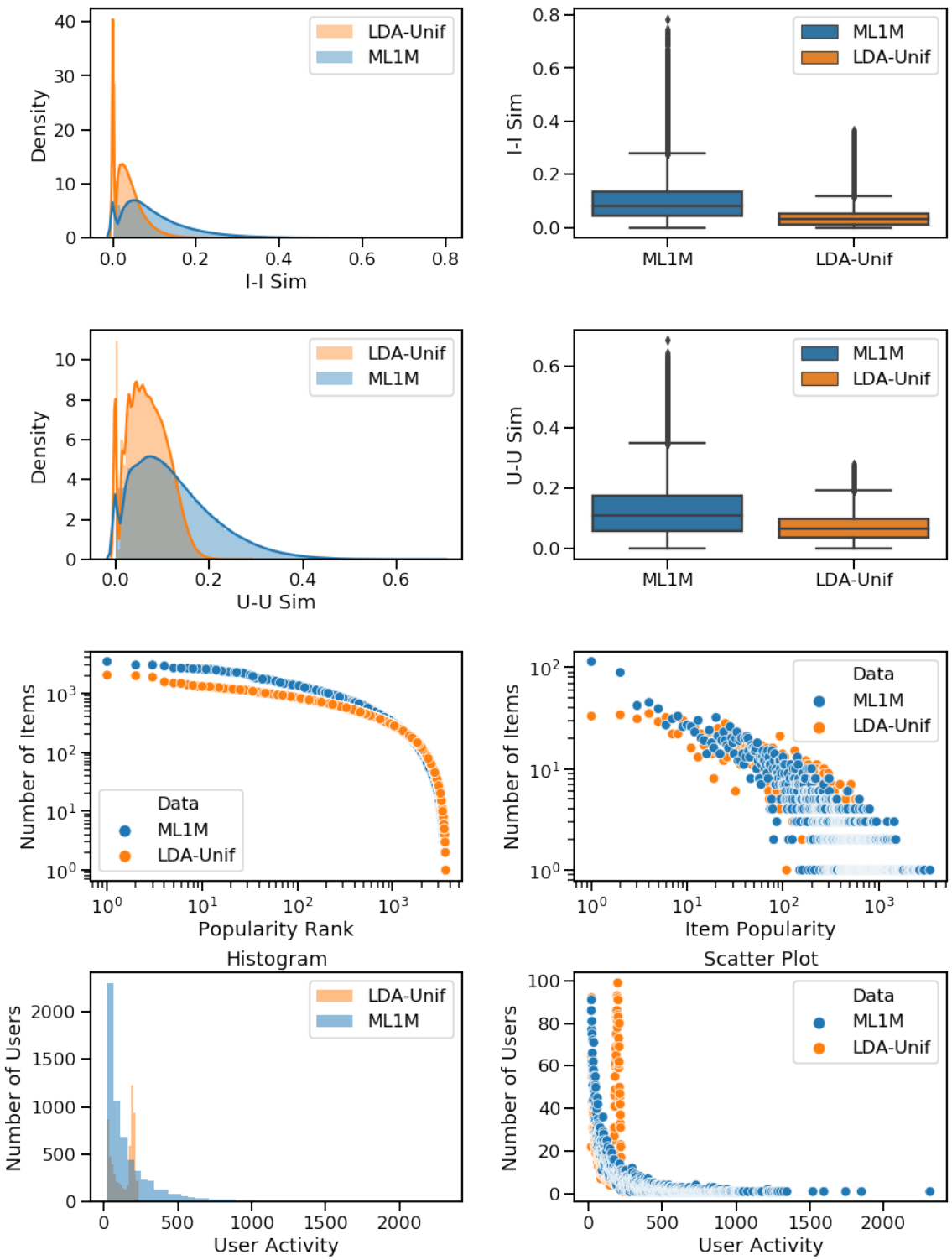

Figure 39: LDA-Unif optimized for Item Pop on ML1M.

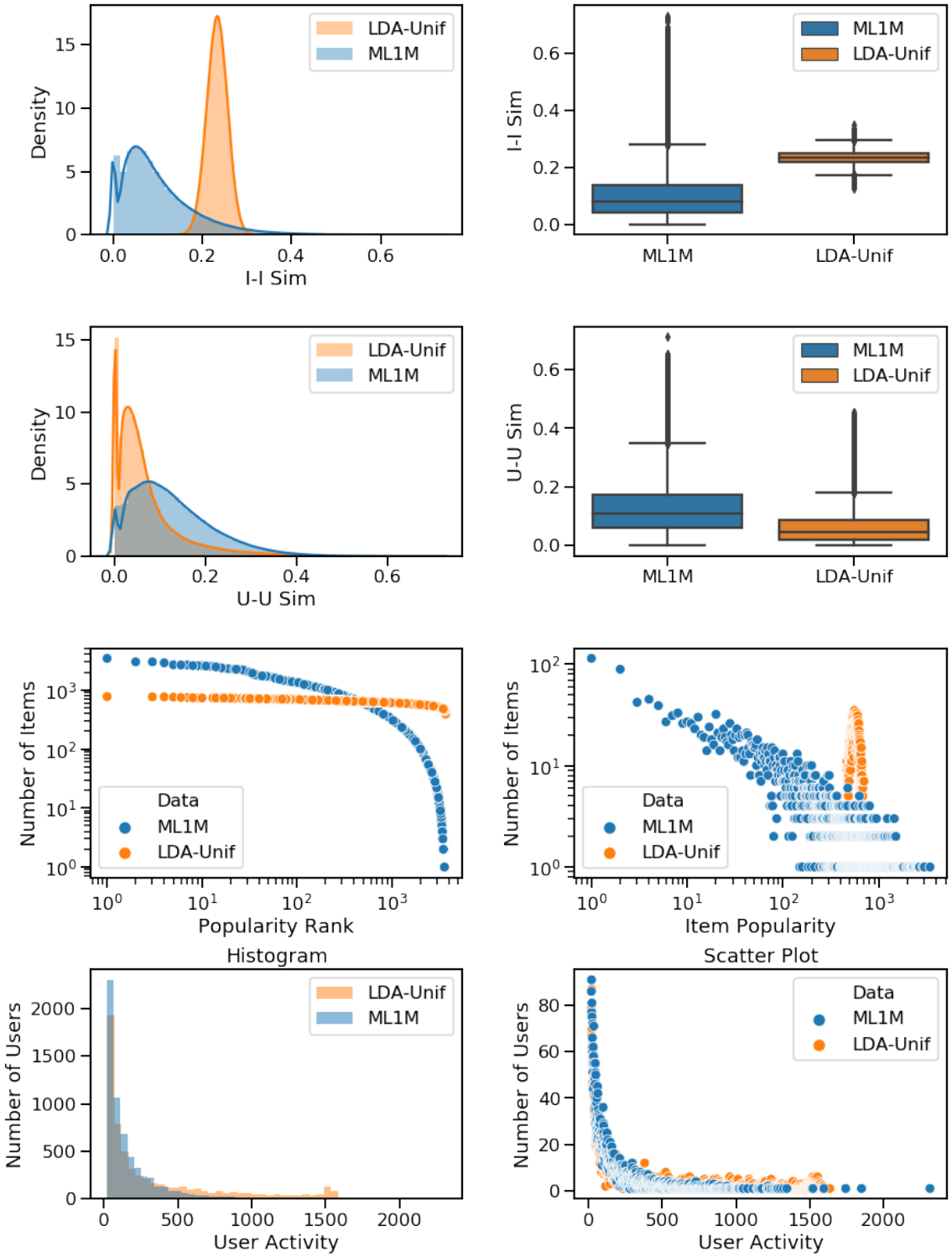

Figure 40: LDA-Unif optimized for User Act on ML1M.

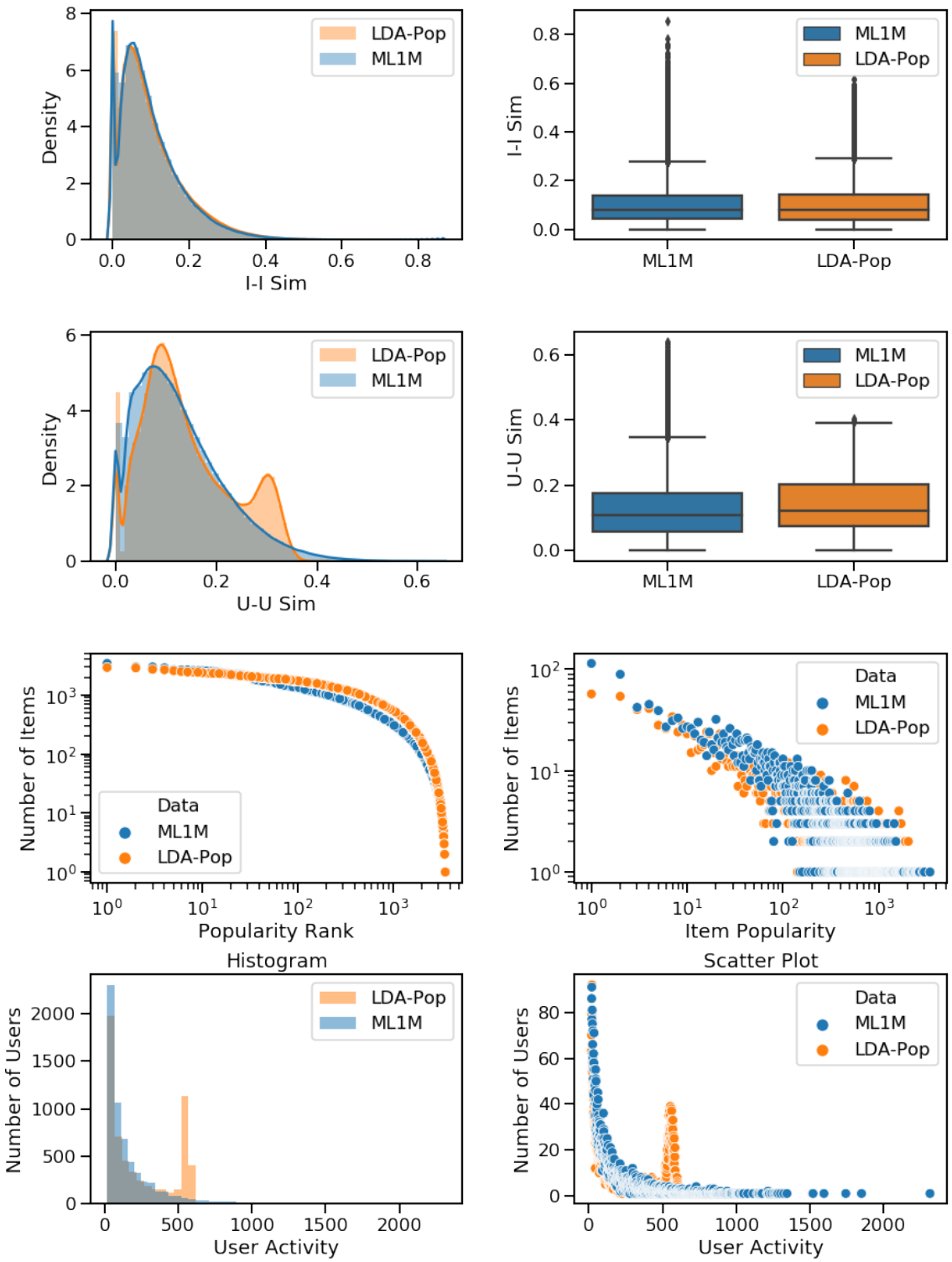

Figure 41: LDA-Pop optimized for I-I Sim on ML1M.

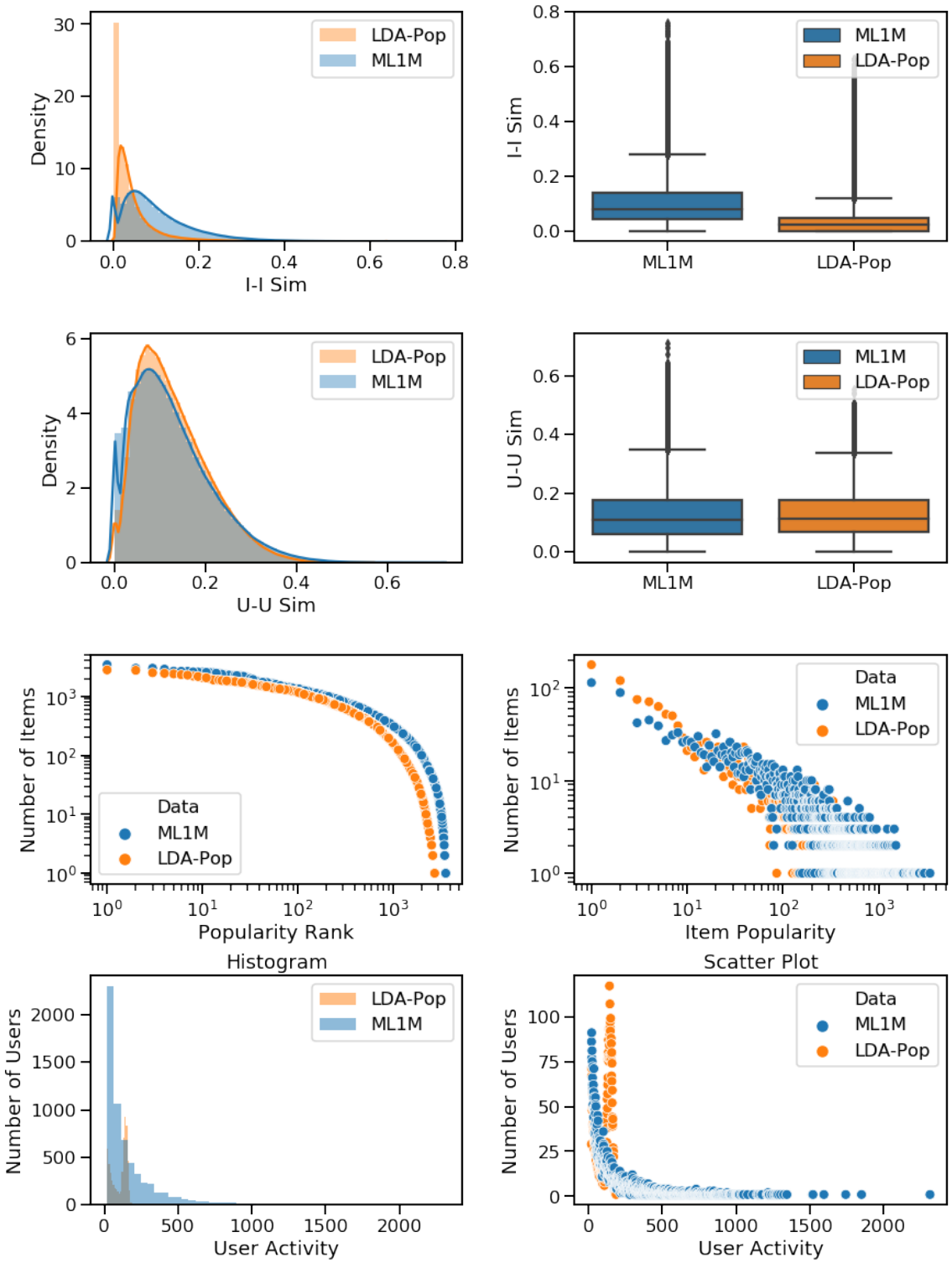

Figure 42: LDA-Pop optimized for U-U Sim on ML1M.

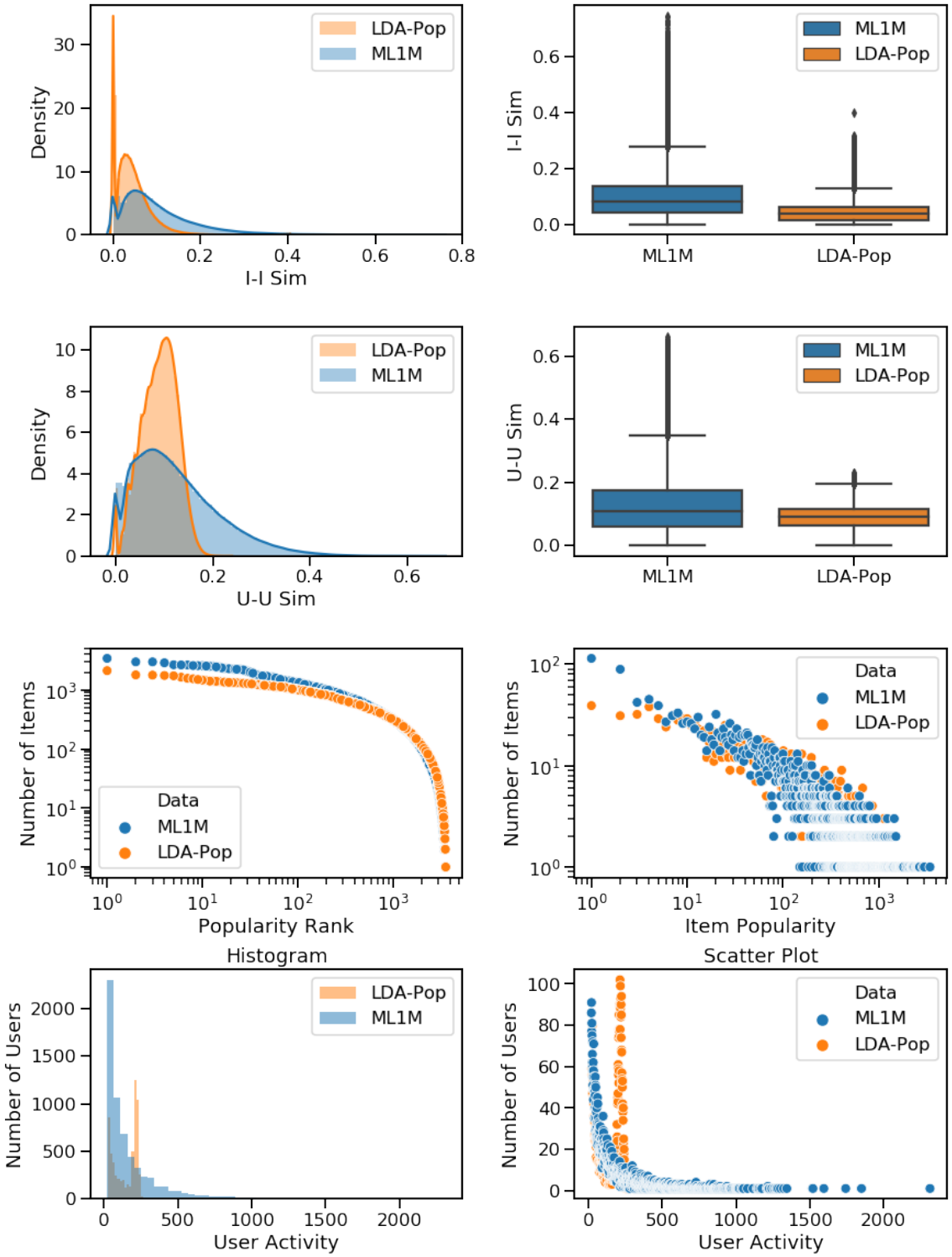

Figure 43: LDA-Pop optimized for Item Pop on ML1M.

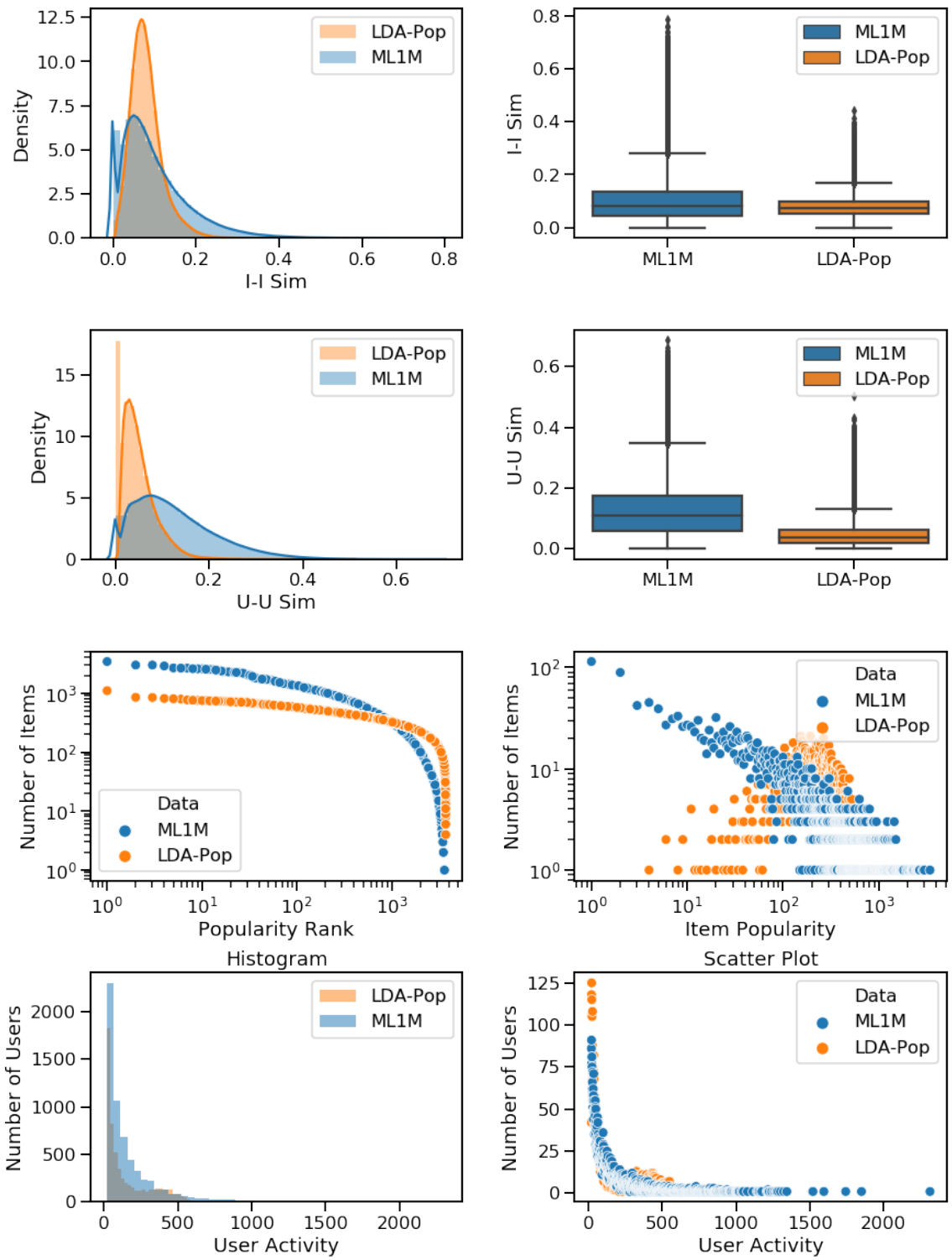

Figure 44: LDA-Pop optimized for User Act on ML1M.

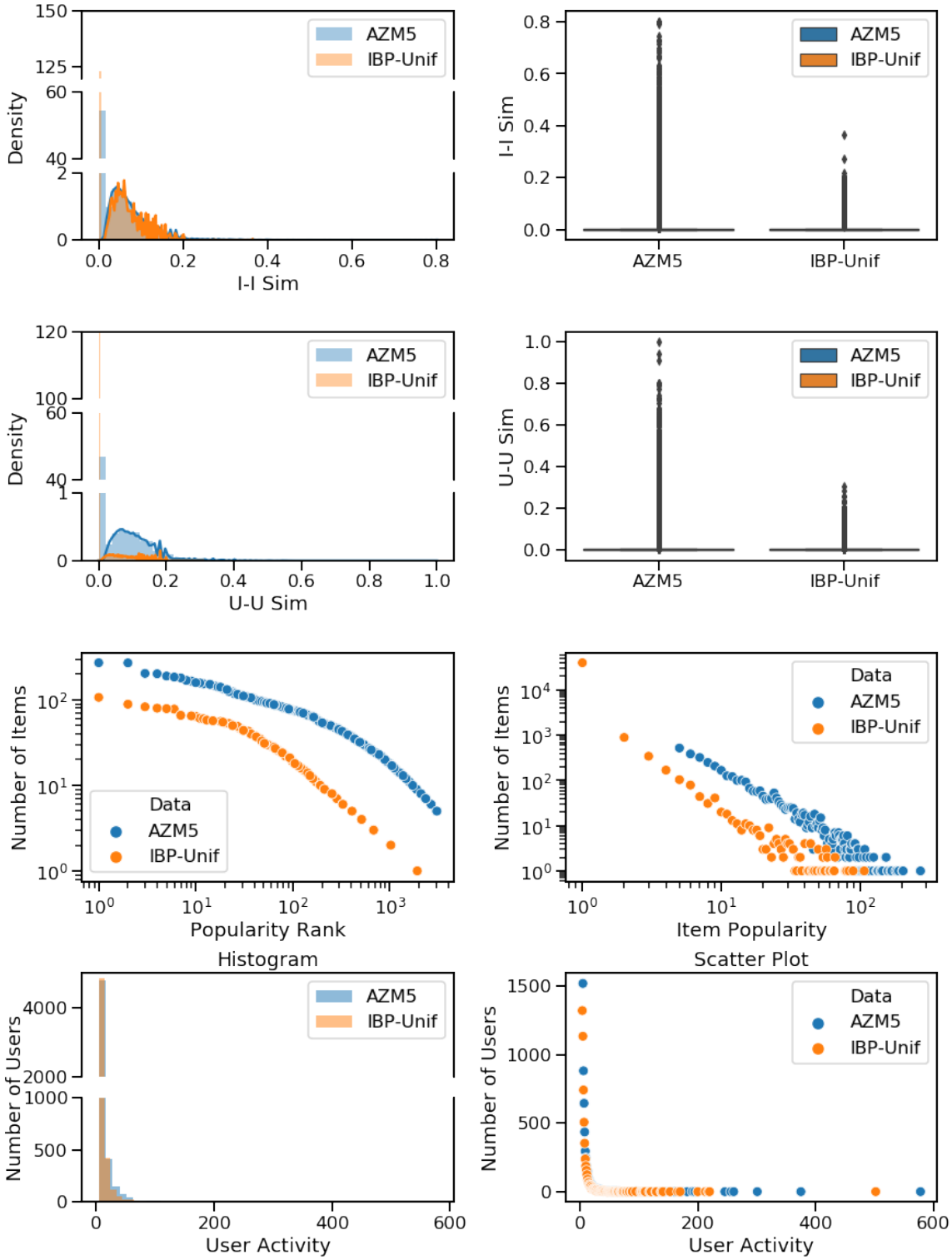

Figure 45: IBP-Unif optimized for I-I Sim on AZM5.

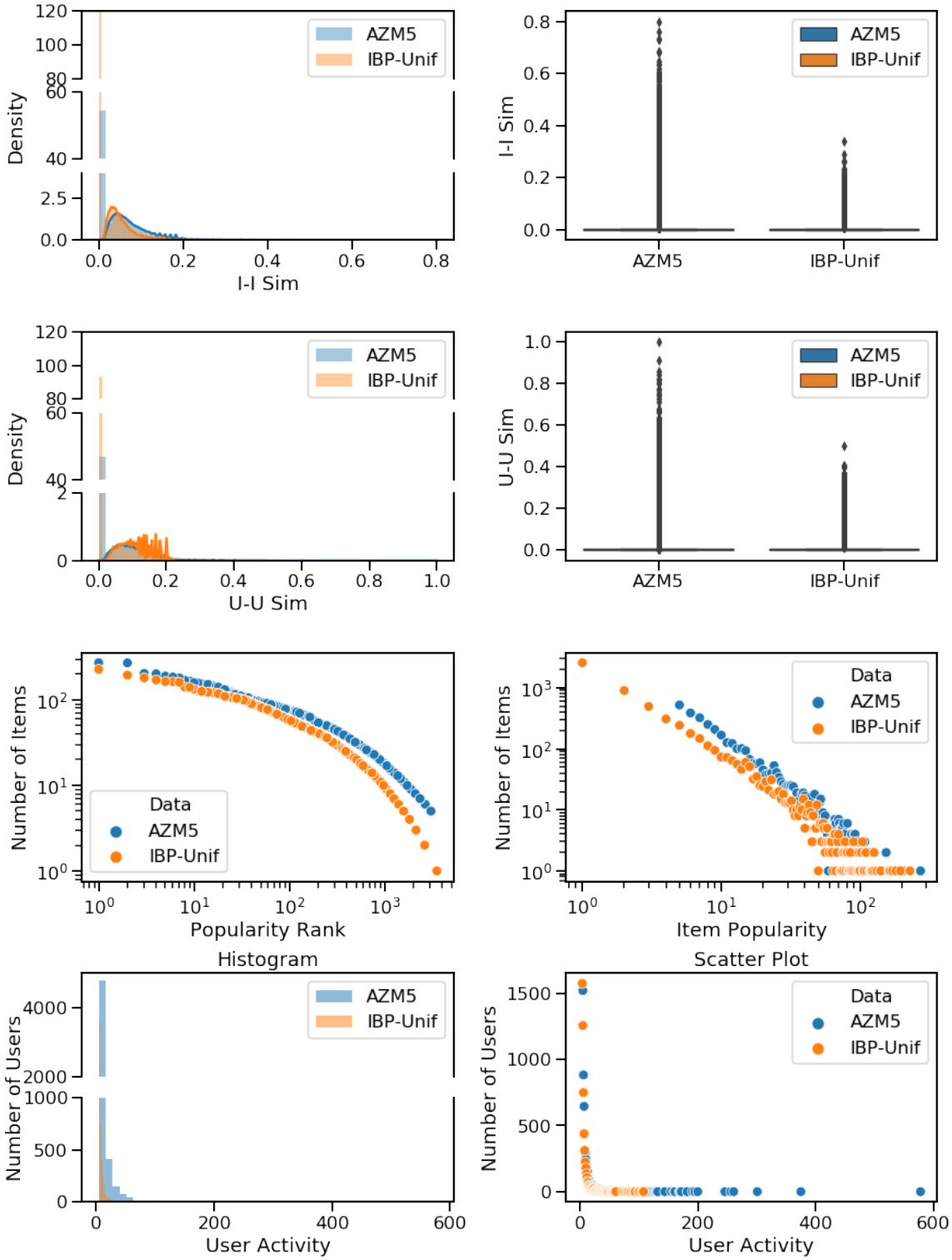

Figure 46: IBP-Unif optimized for U-U Sim on AZM5.

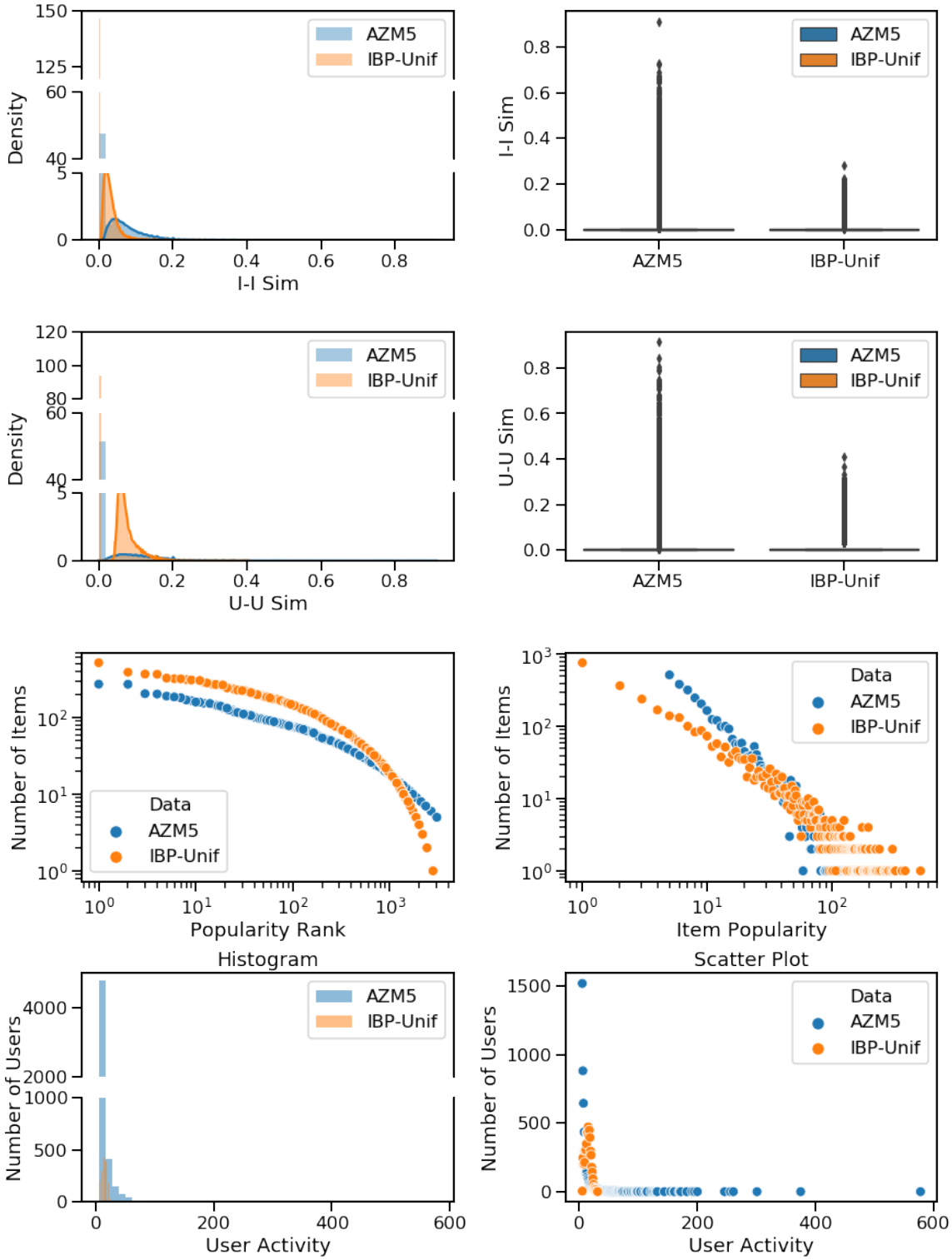

Figure 47: IBP-Unif optimized for Item Pop on AZM5.

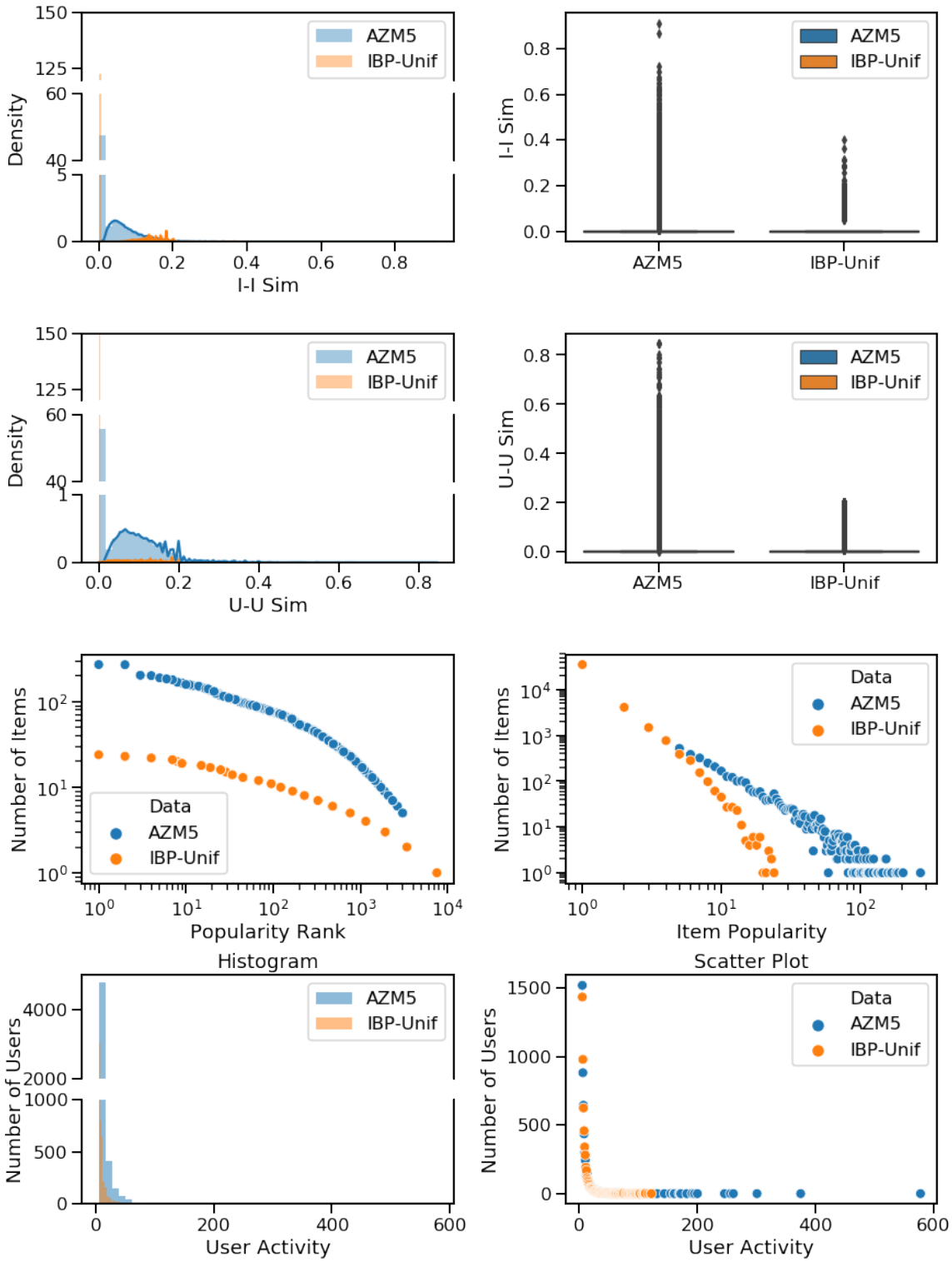

Figure 48: IBP-Unif optimized for User Act on AZM5.

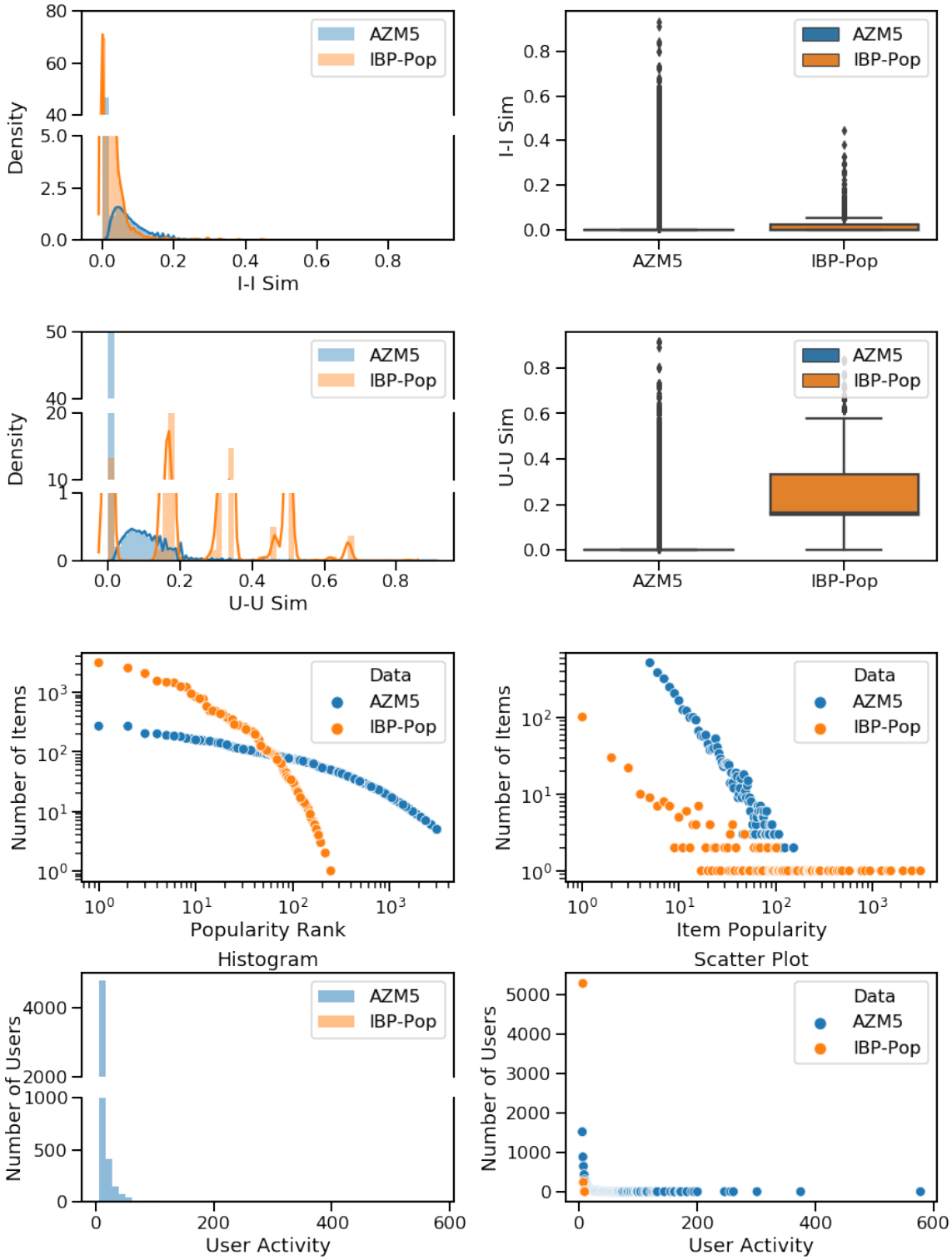

Figure 49: IBP-Pop optimized for I-I Sim on AZM5.

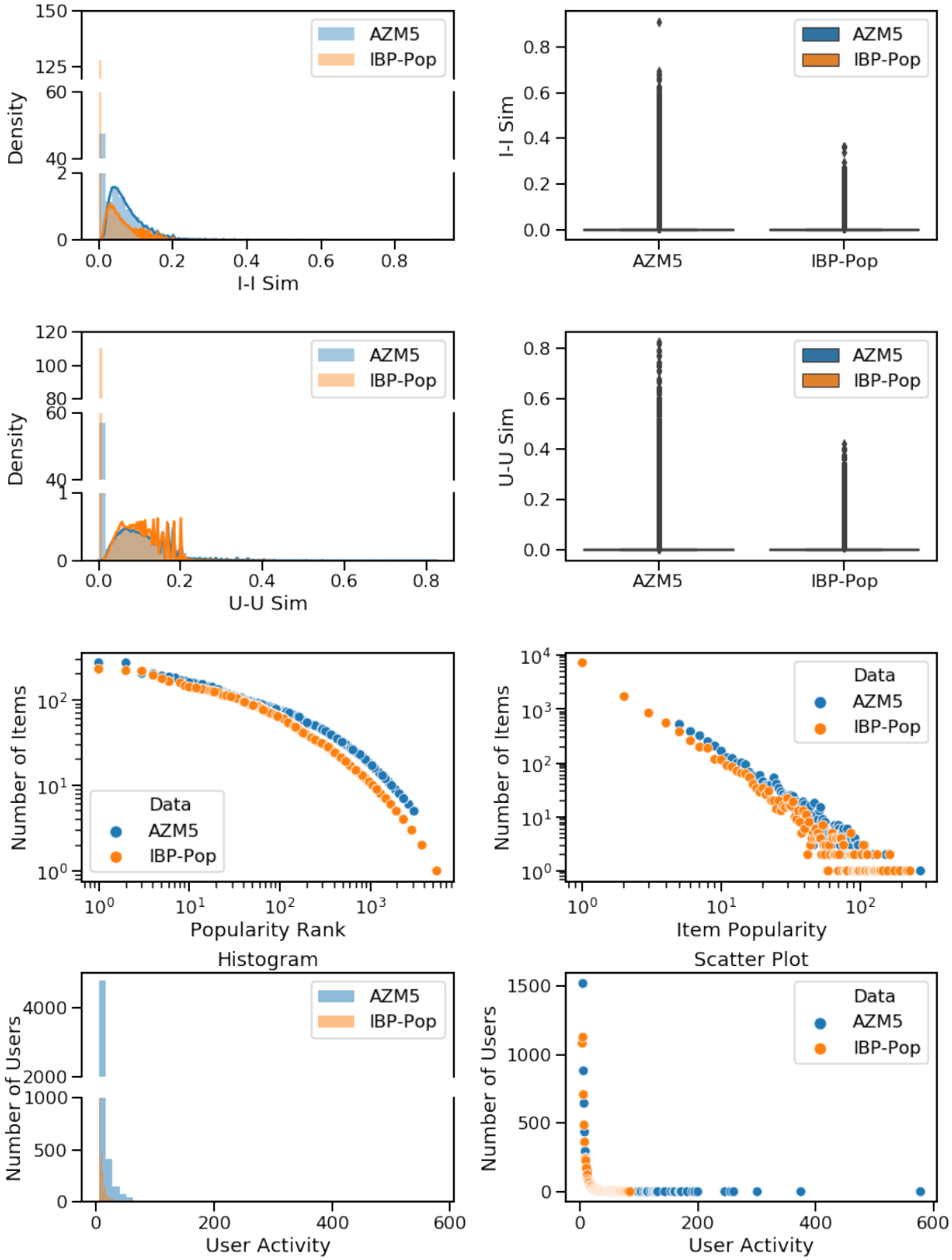

Figure 50: IBP-Pop optimized for U-U Sim on AZM5.

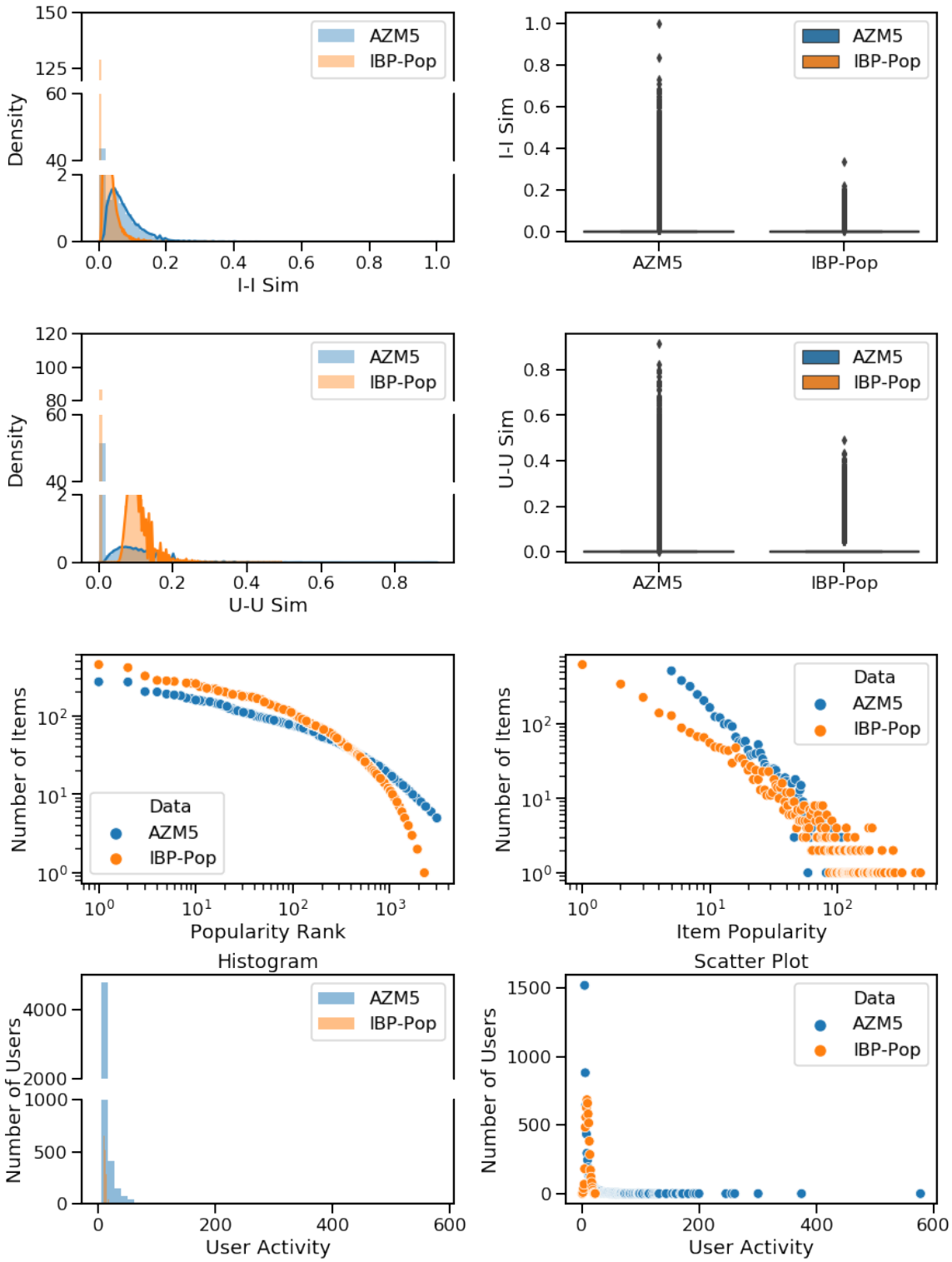

Figure 51: IBP-Pop optimized for Item Pop on AZM5.

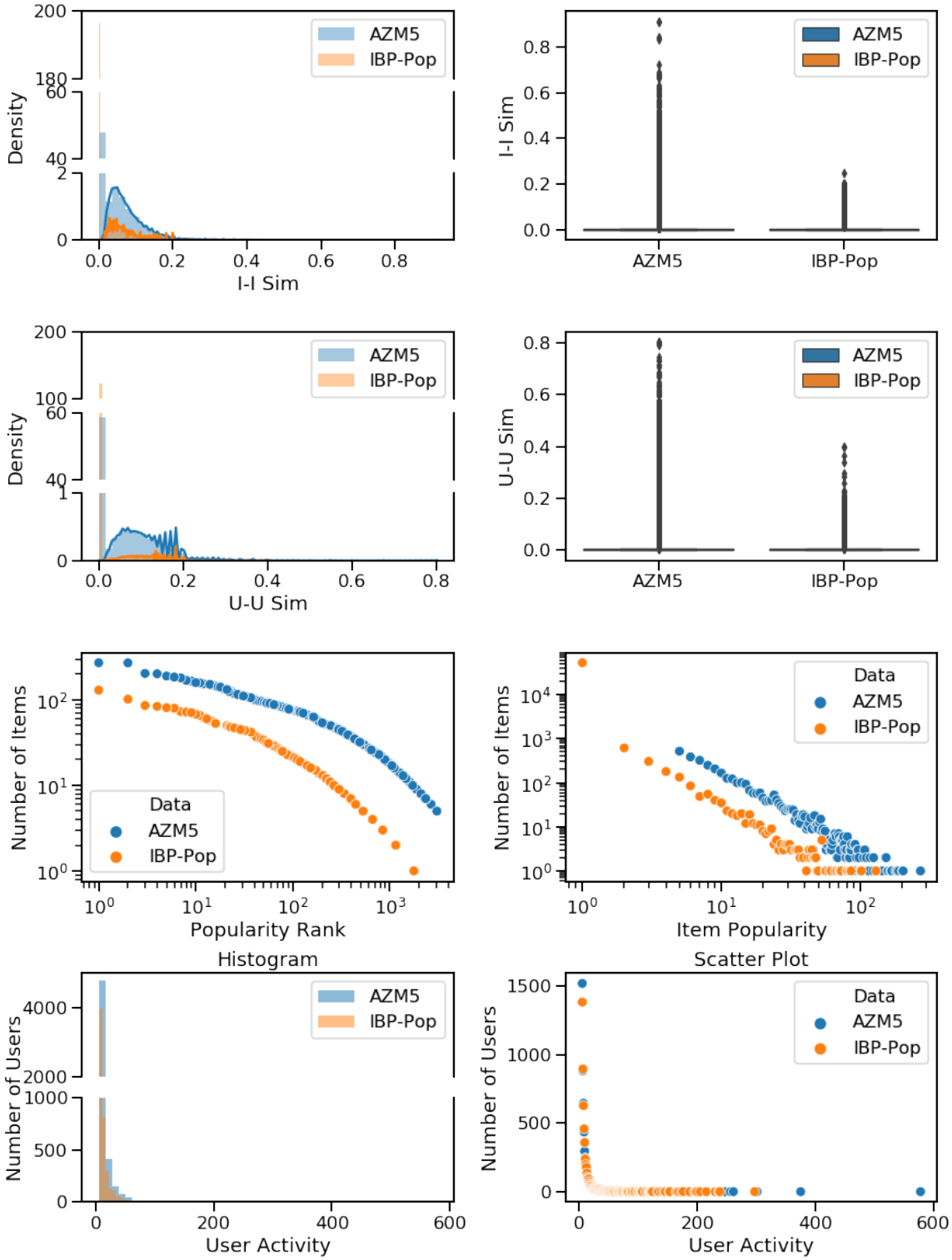

Figure 52: IBP-Pop optimized for User Act on AZM5.

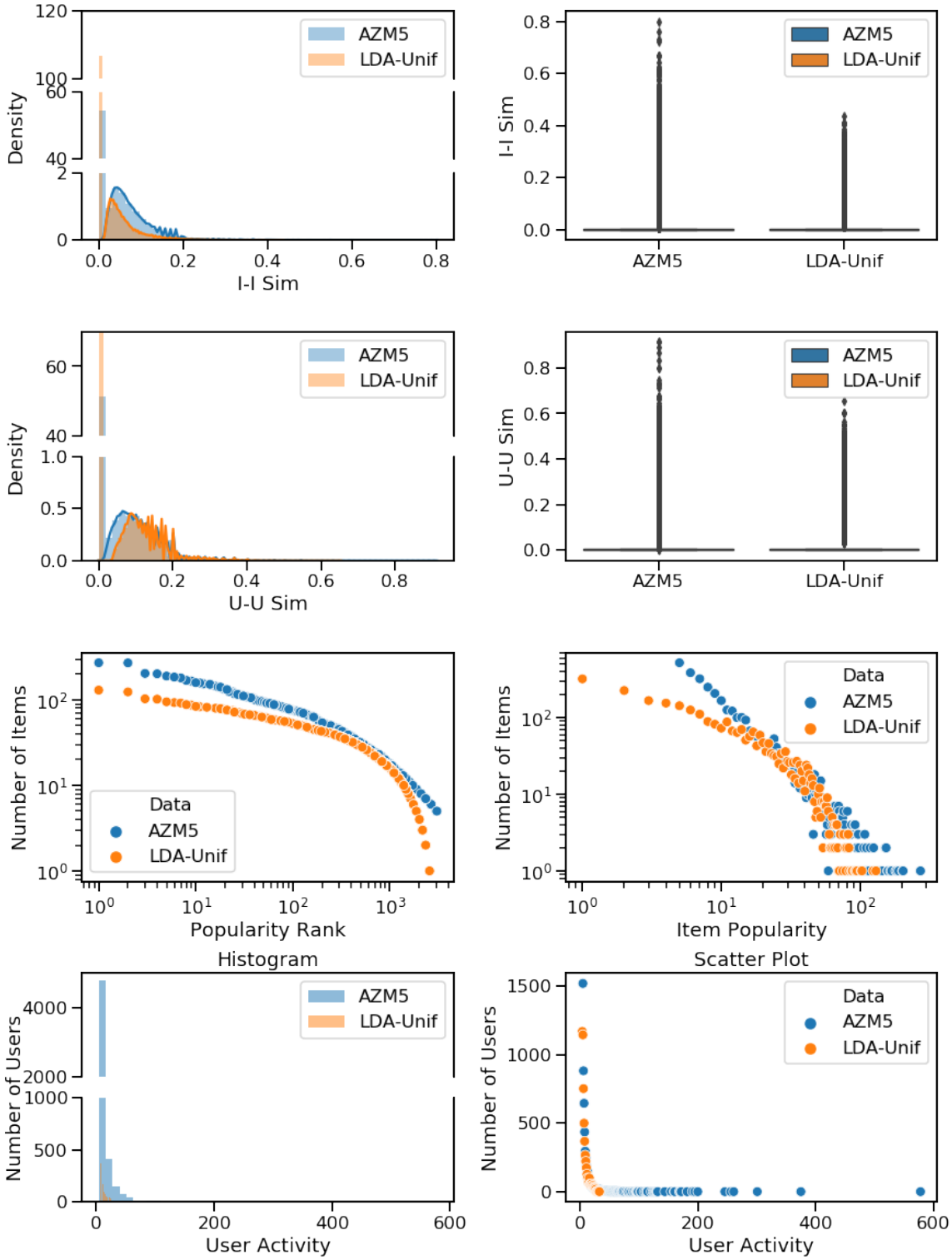

Figure 53: LDA-Unif optimized for U-U Sim on AZM5.

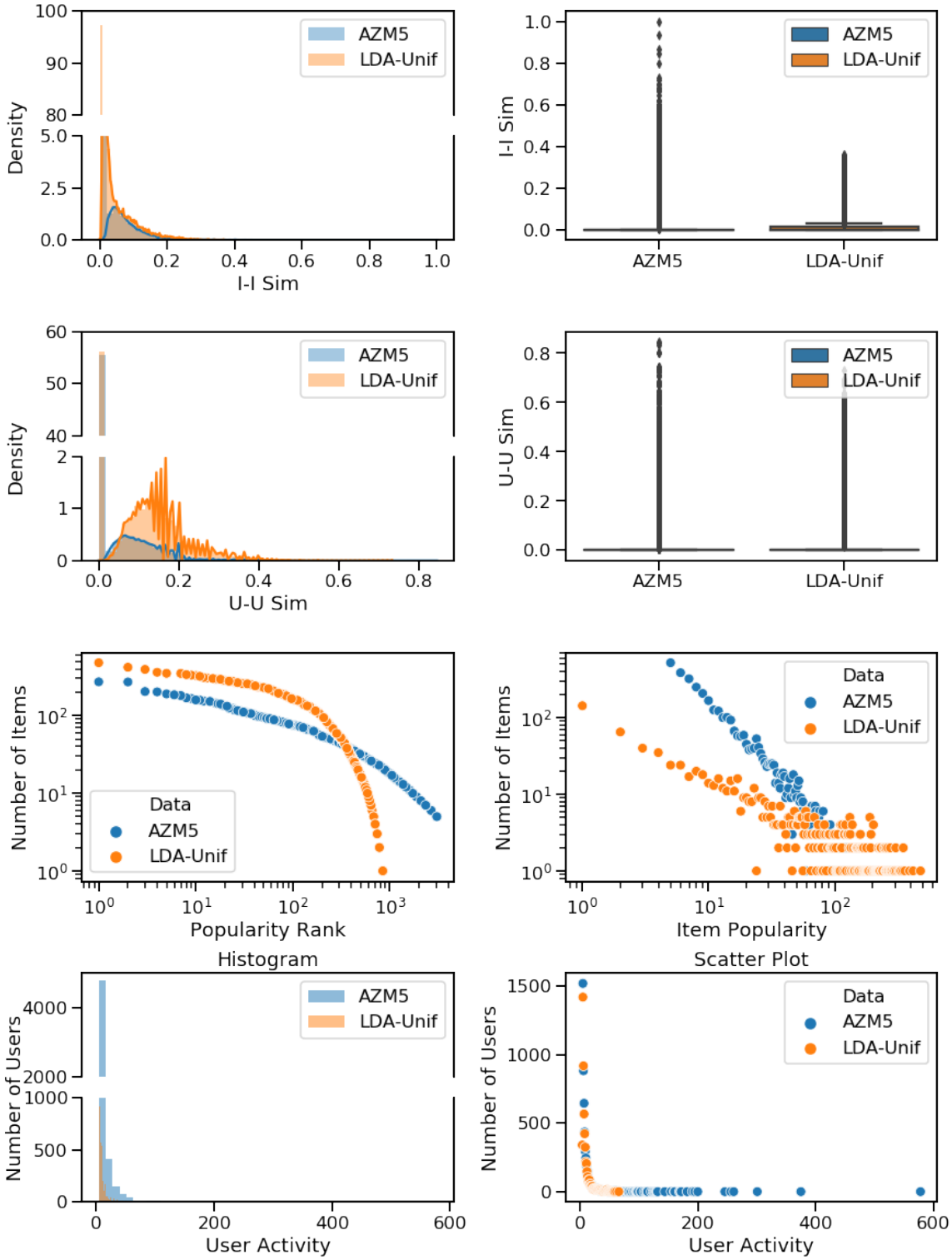

Figure 54: LDA-Unif optimized for I-I Sim on AZM5.

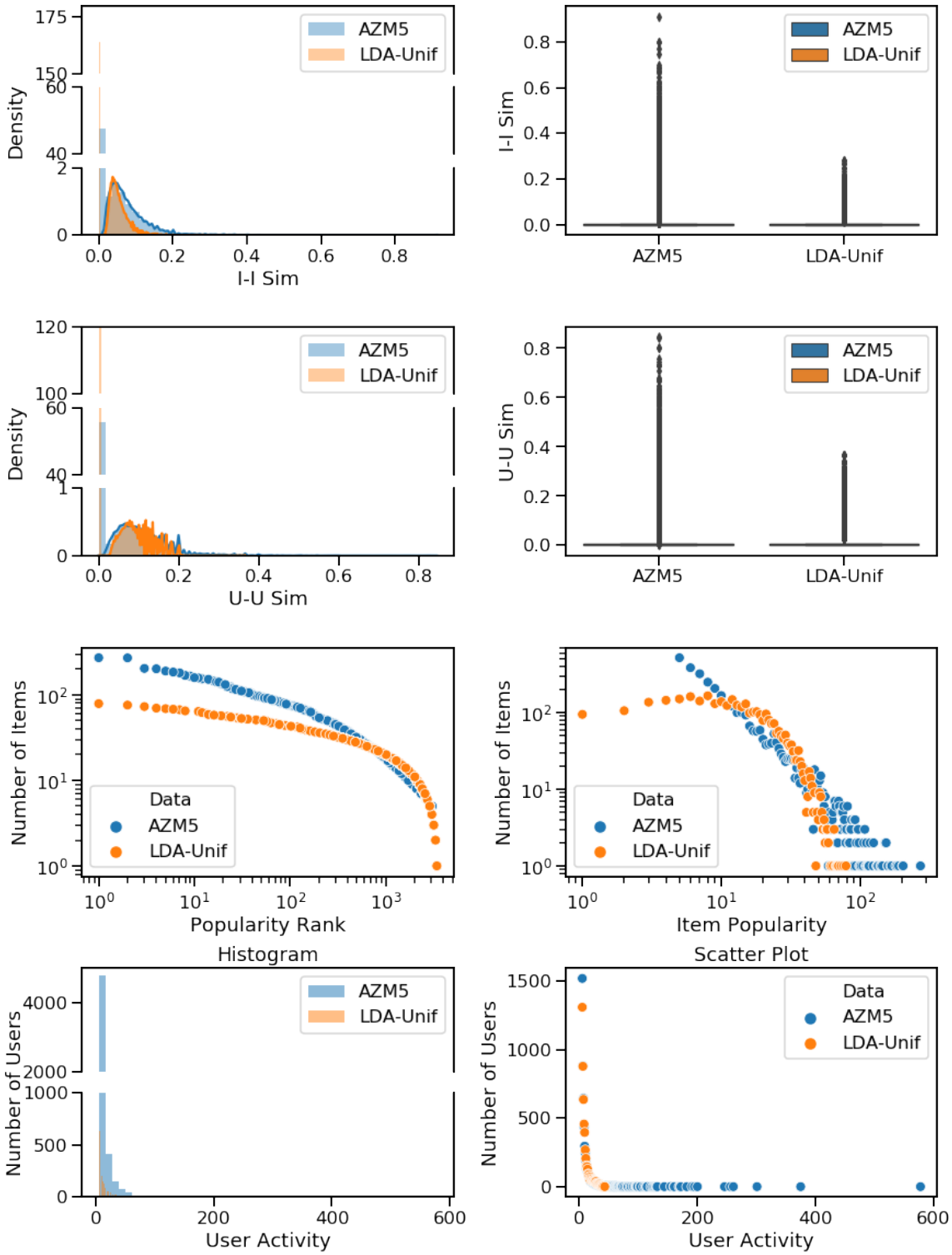

Figure 55: LDA-Unif optimized for Item Pop on AZM5.

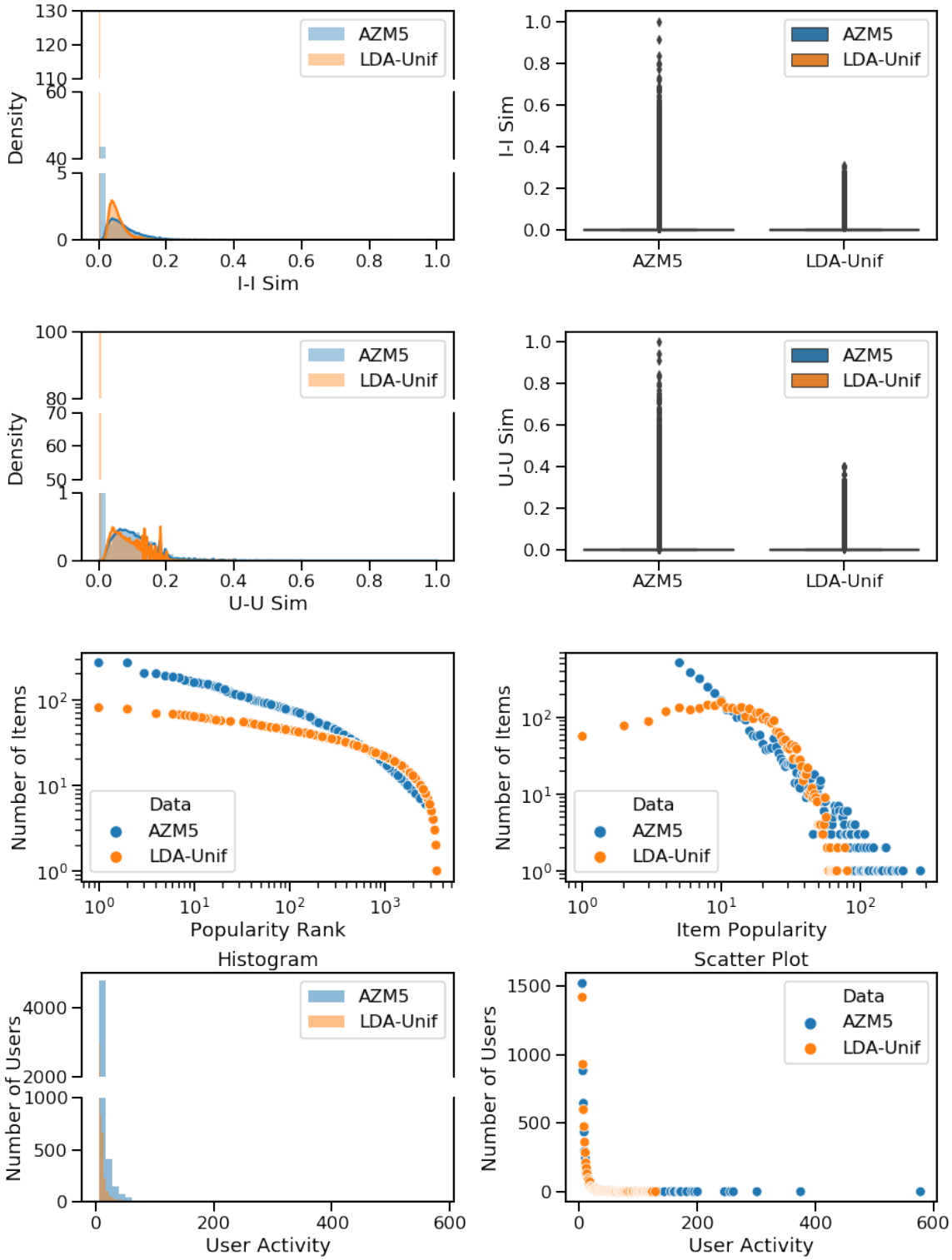

Figure 56: LDA-Unif optimized for User Act on AZM5.

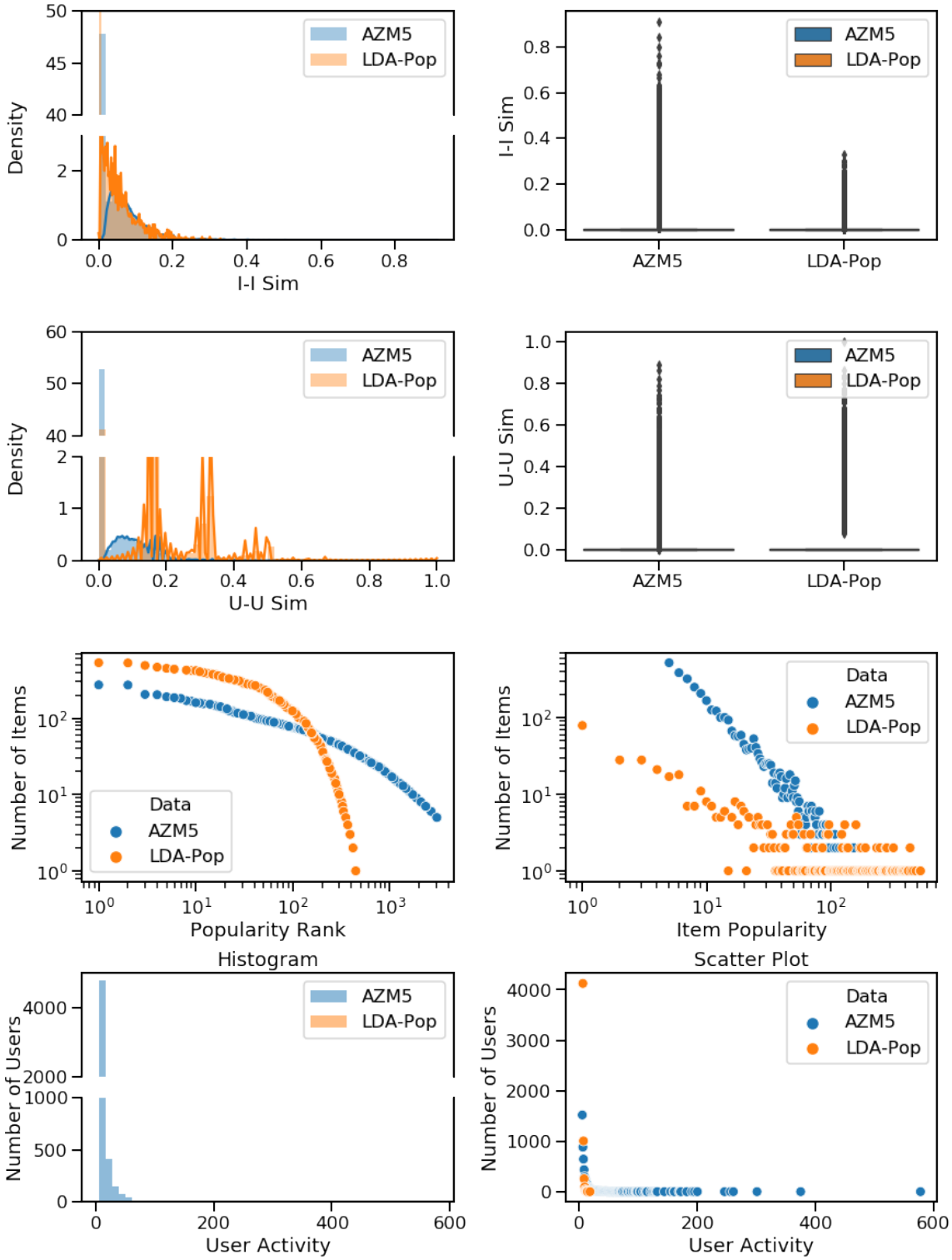

Figure 57: LDA-Pop optimized for I-I Sim on AZM5.

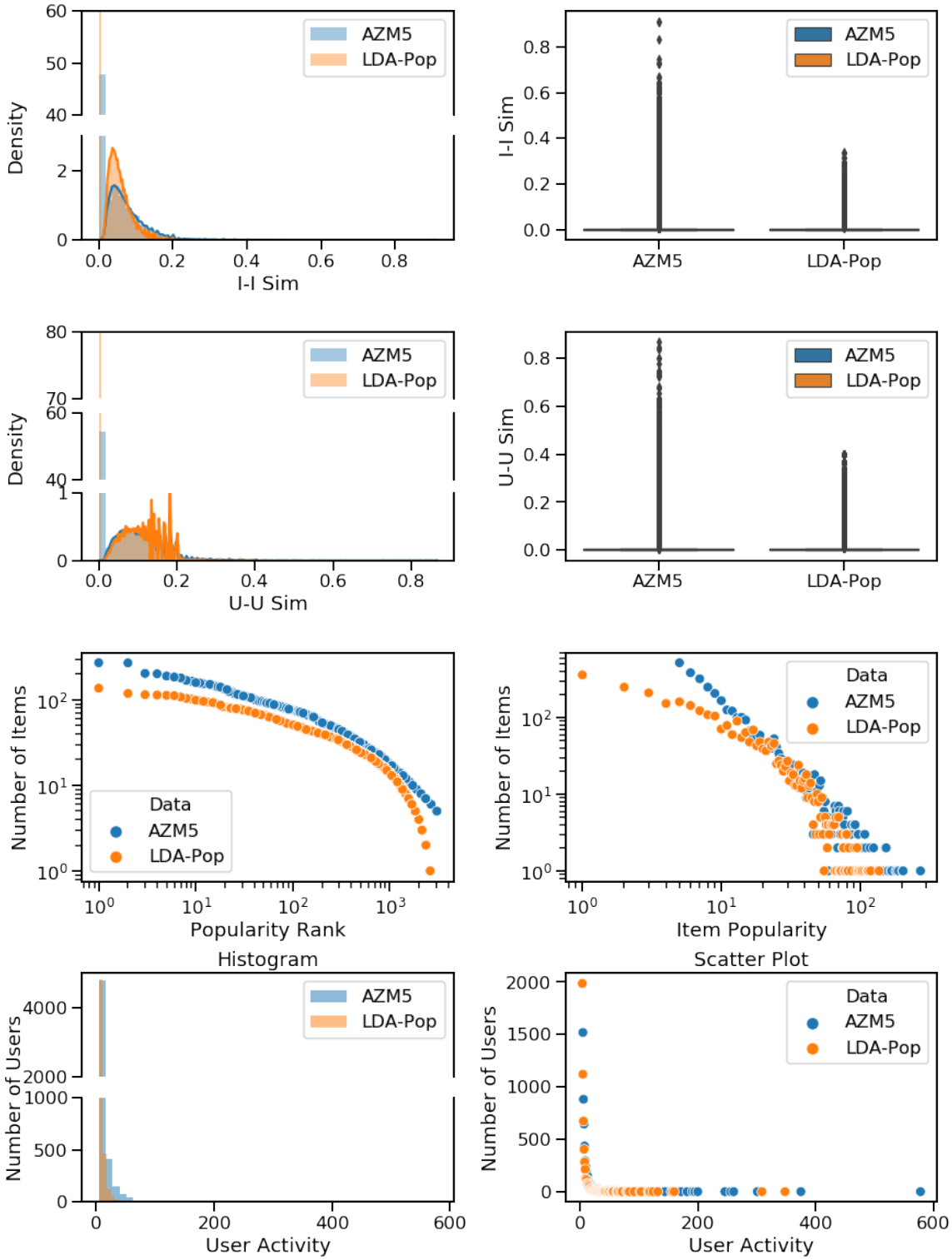

Figure 58: LDA-Pop optimized for U-U Sim on AZM5.

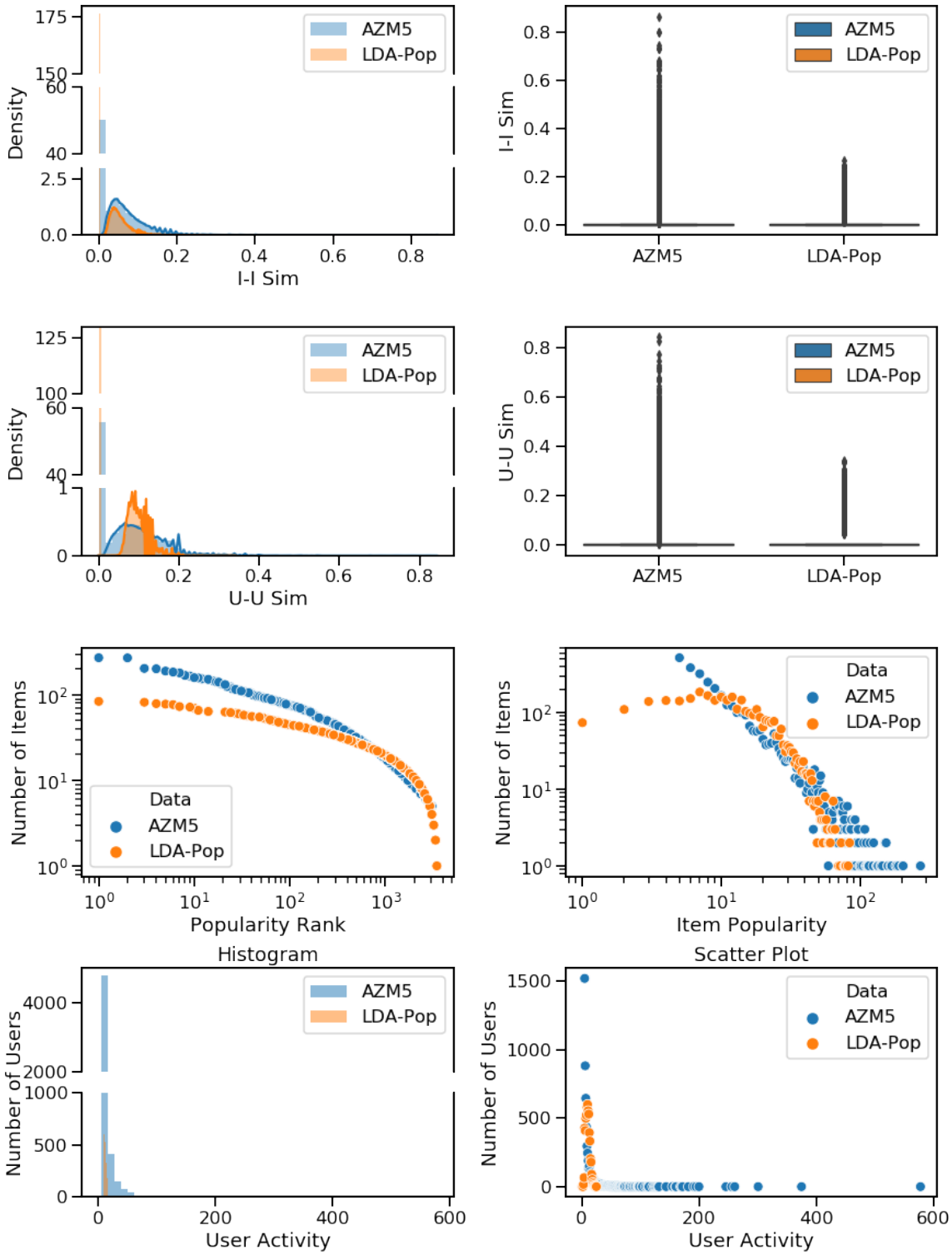

Figure 59: LDA-Pop optimized for Item Pop on AZM5.

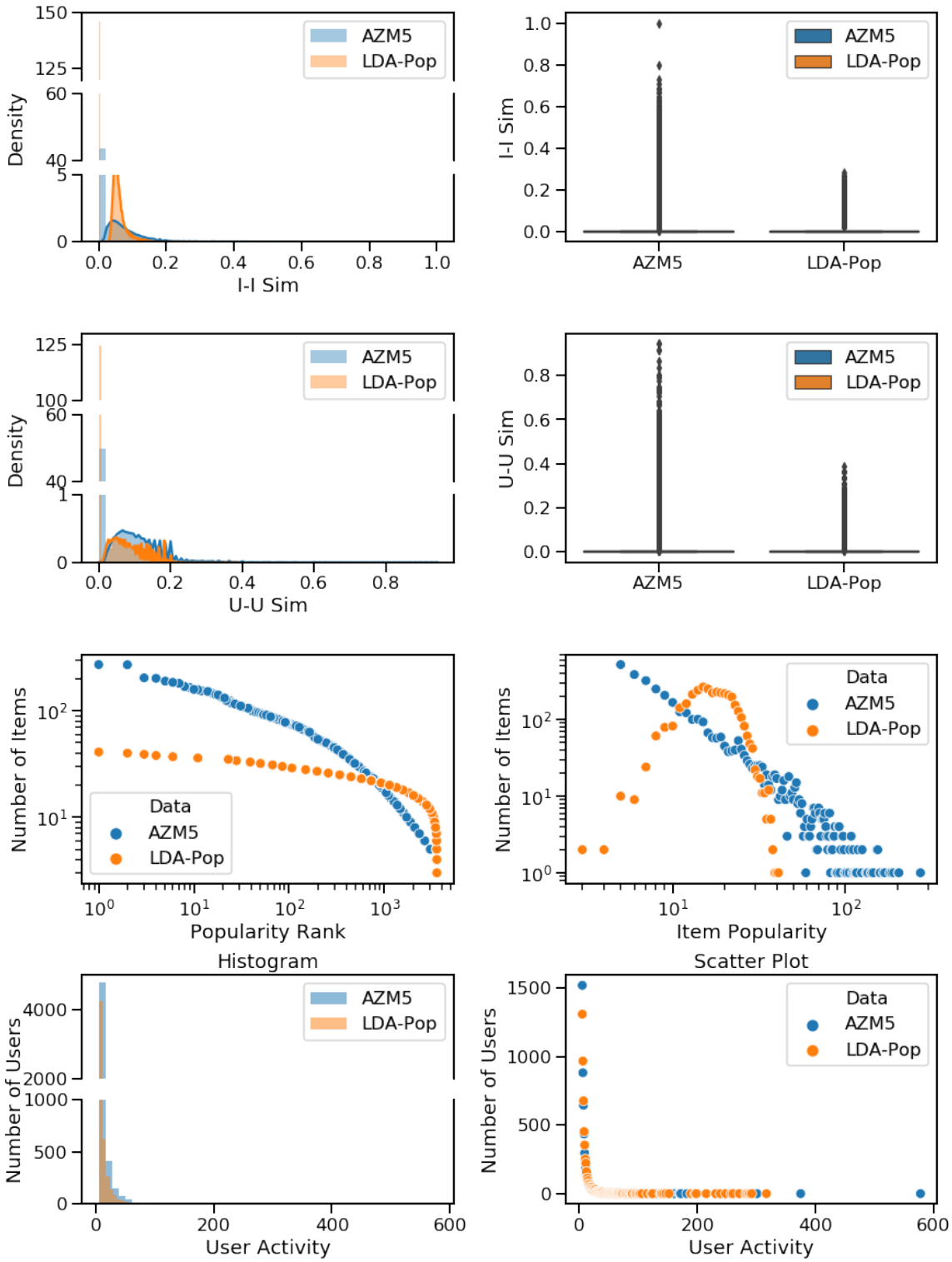

Figure 60: LDA-Pop optimized for User Act on AZM5.

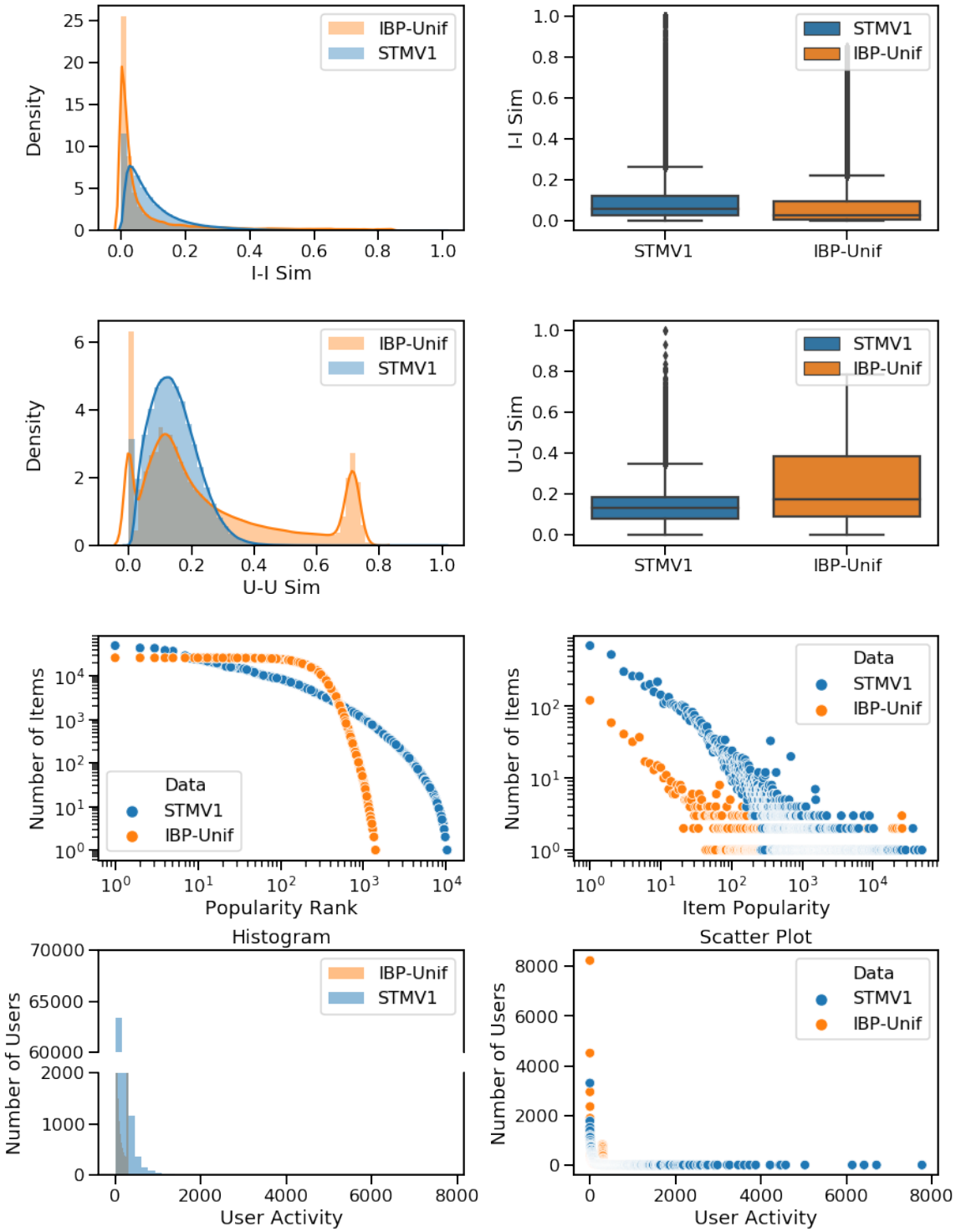

Figure 61: IBP-Unif optimized for I-I Sim on STMV1.

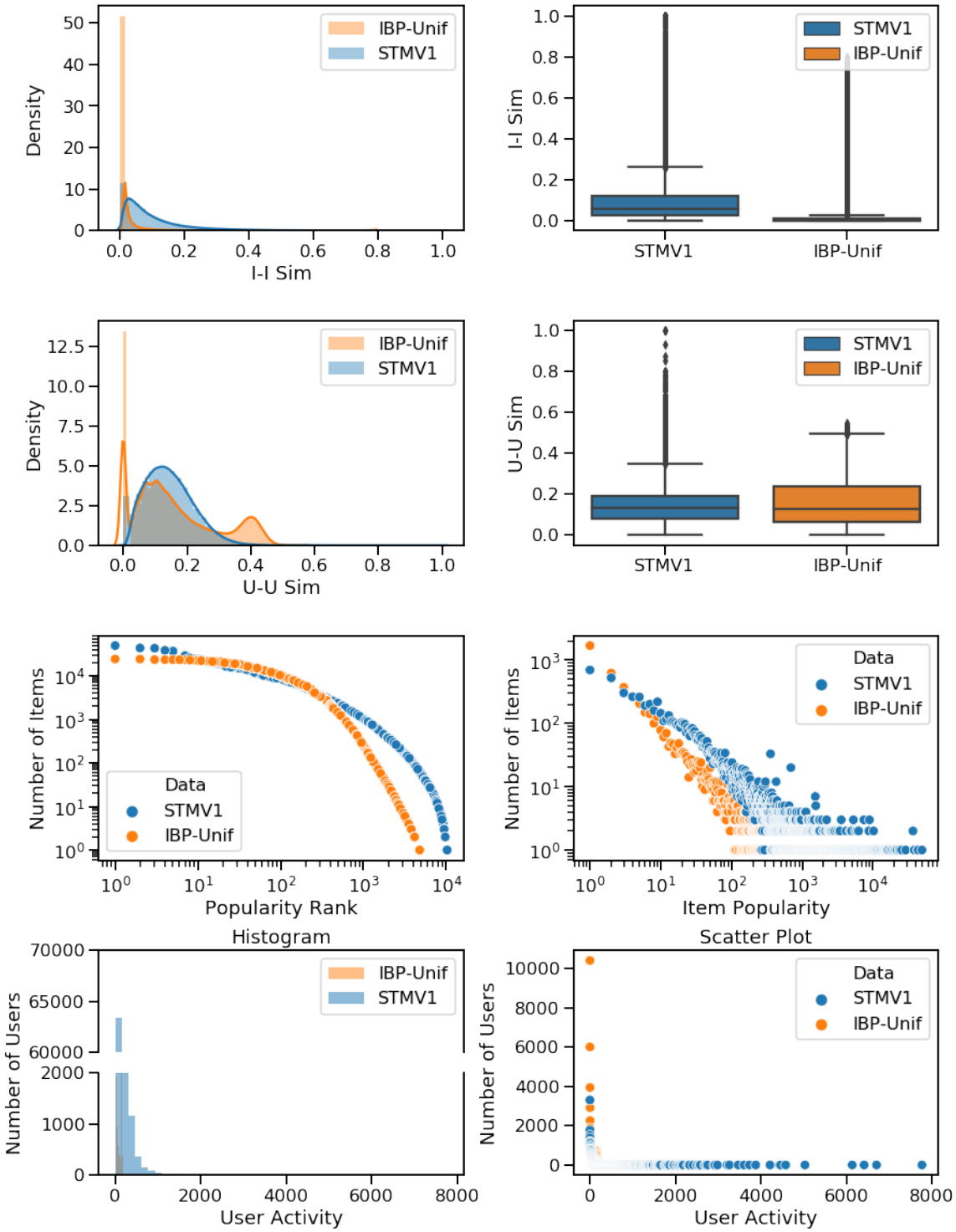

Figure 62: IBP-Unif optimized for U-U Sim on STMV1.

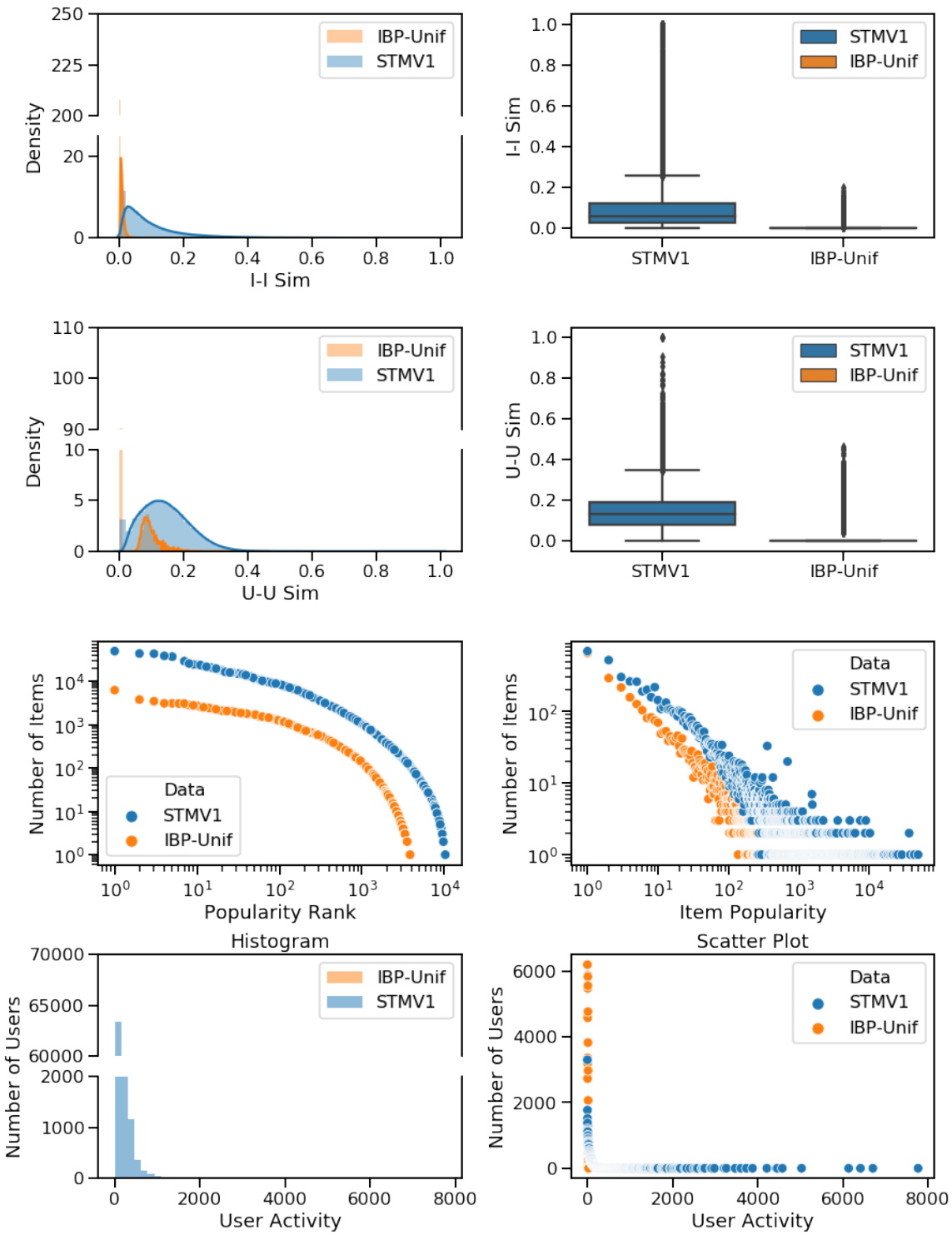

Figure 63: IBP-Unif optimized for Item Pop on STMV1.

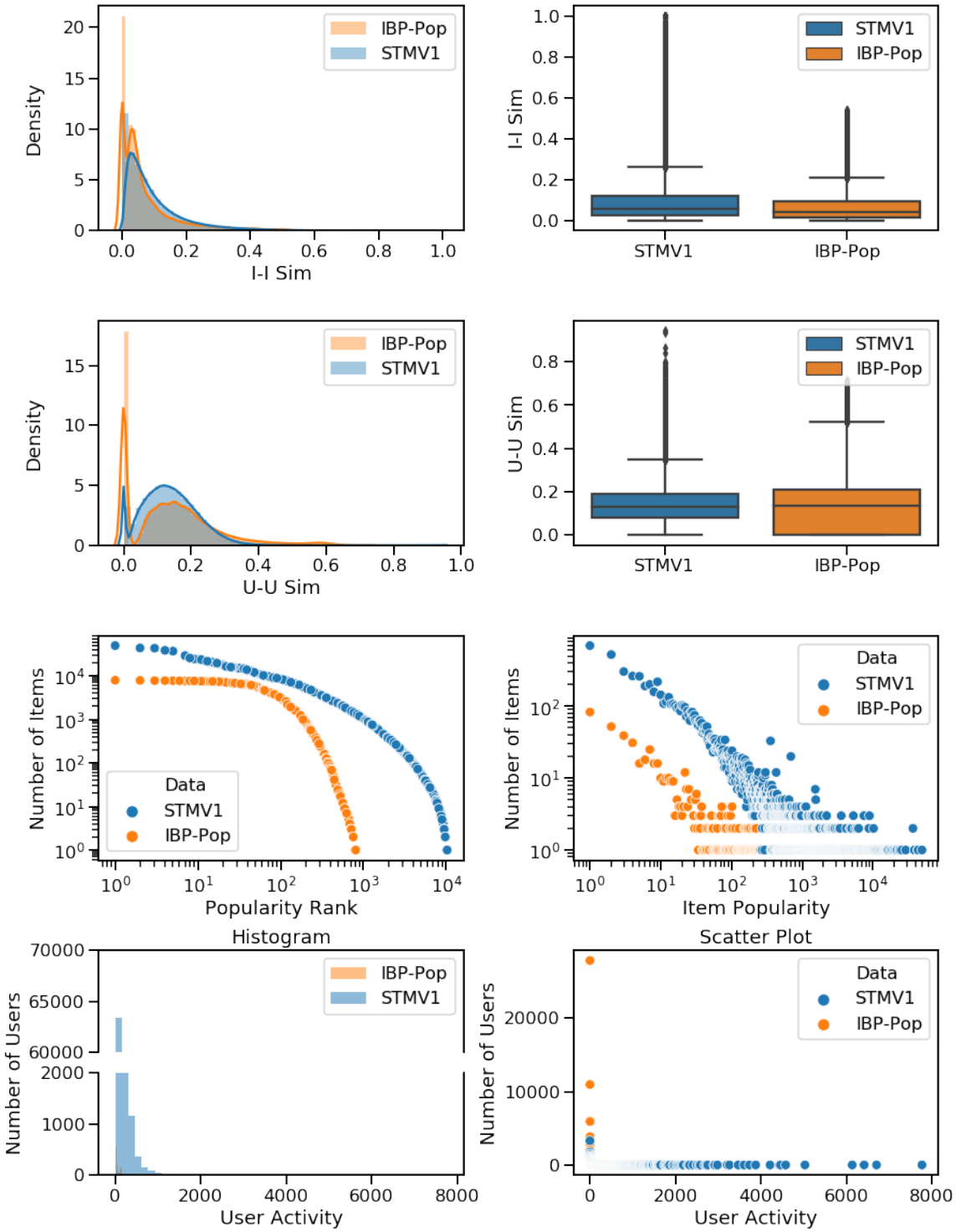

Figure 64: IBP-Pop optimized for I-I Sim on STMV1.

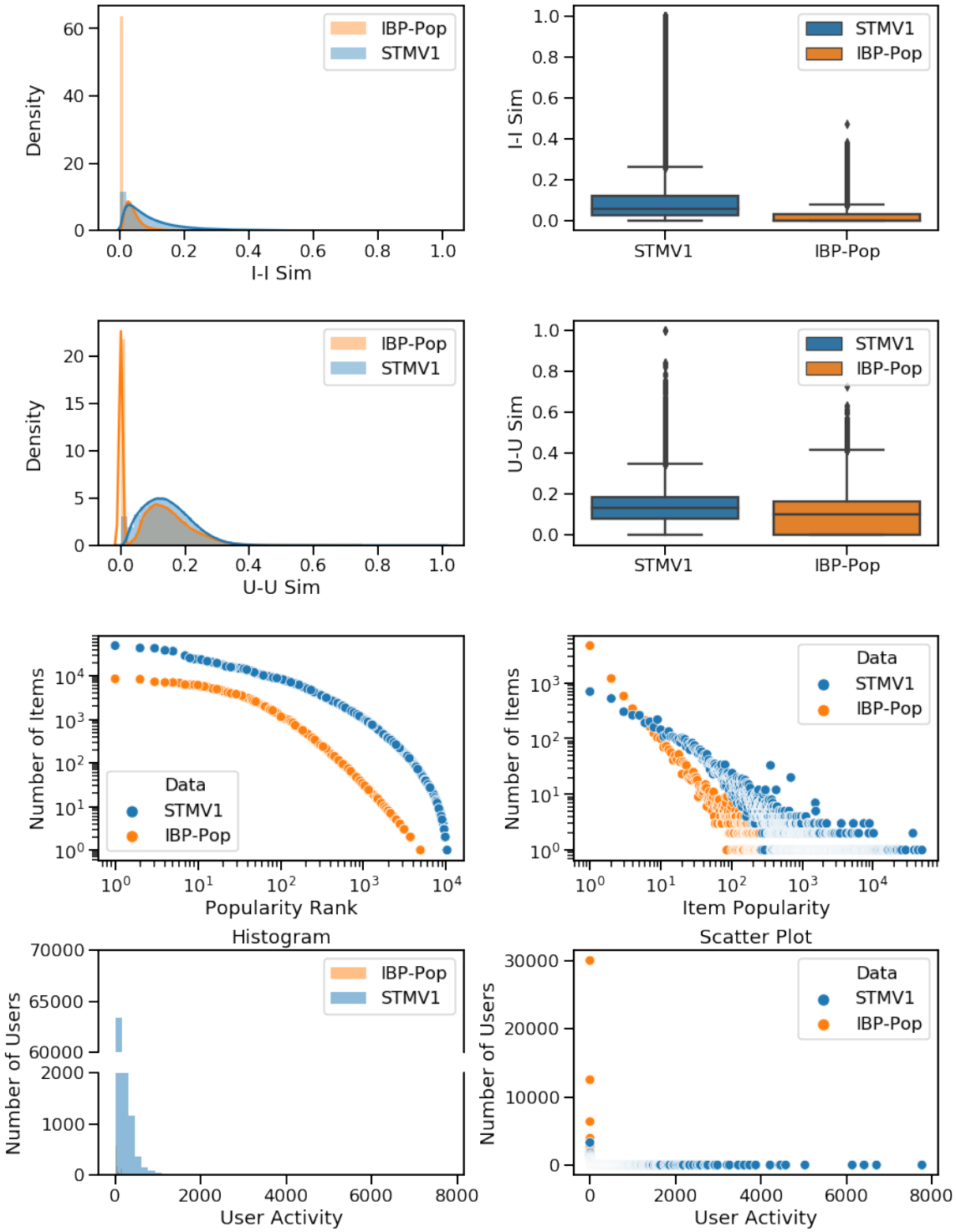

Figure 65: IBP-Pop optimized for U-U Sim on STMV1.

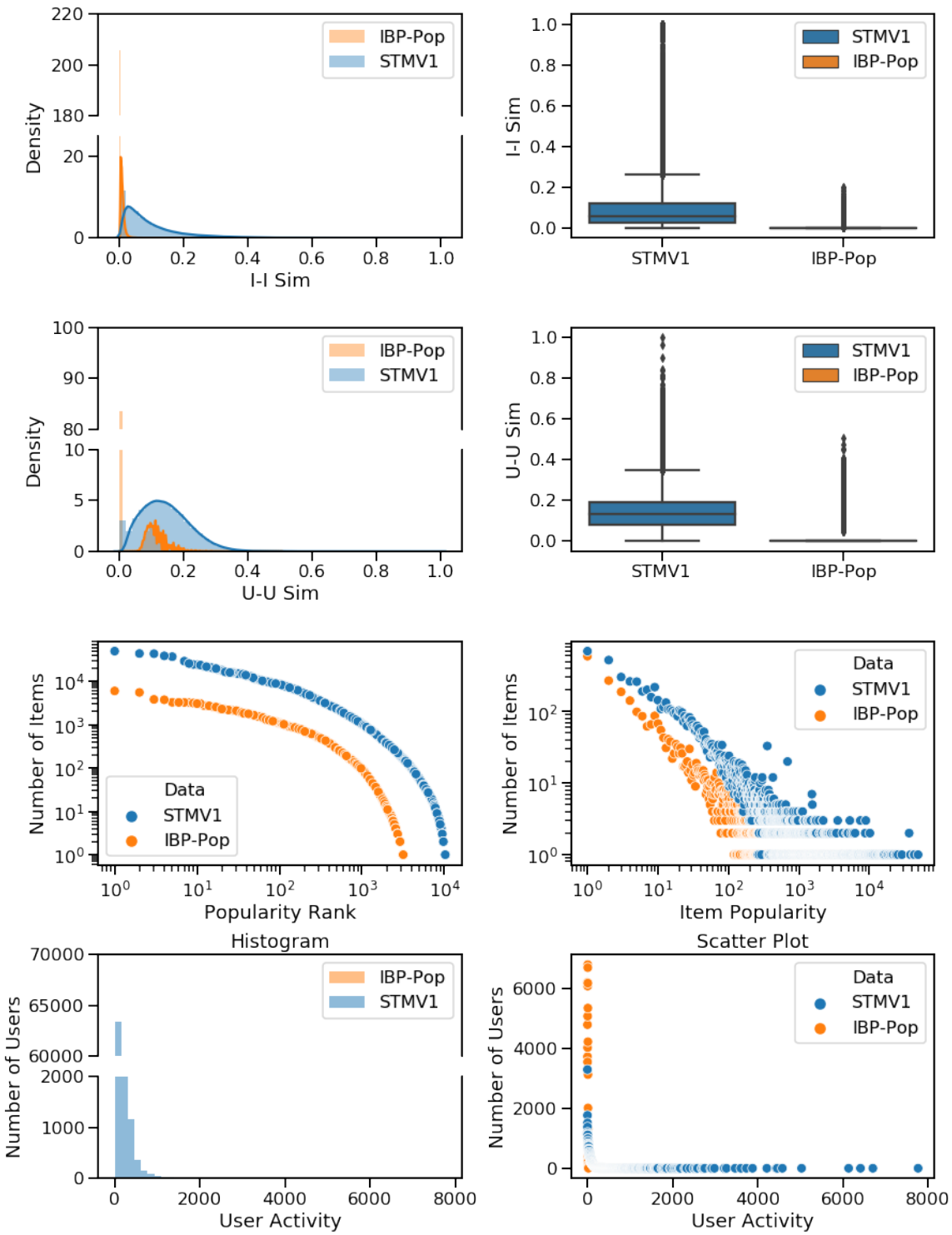

Figure 66: IBP-Pop optimized for Item Pop on STMV1.

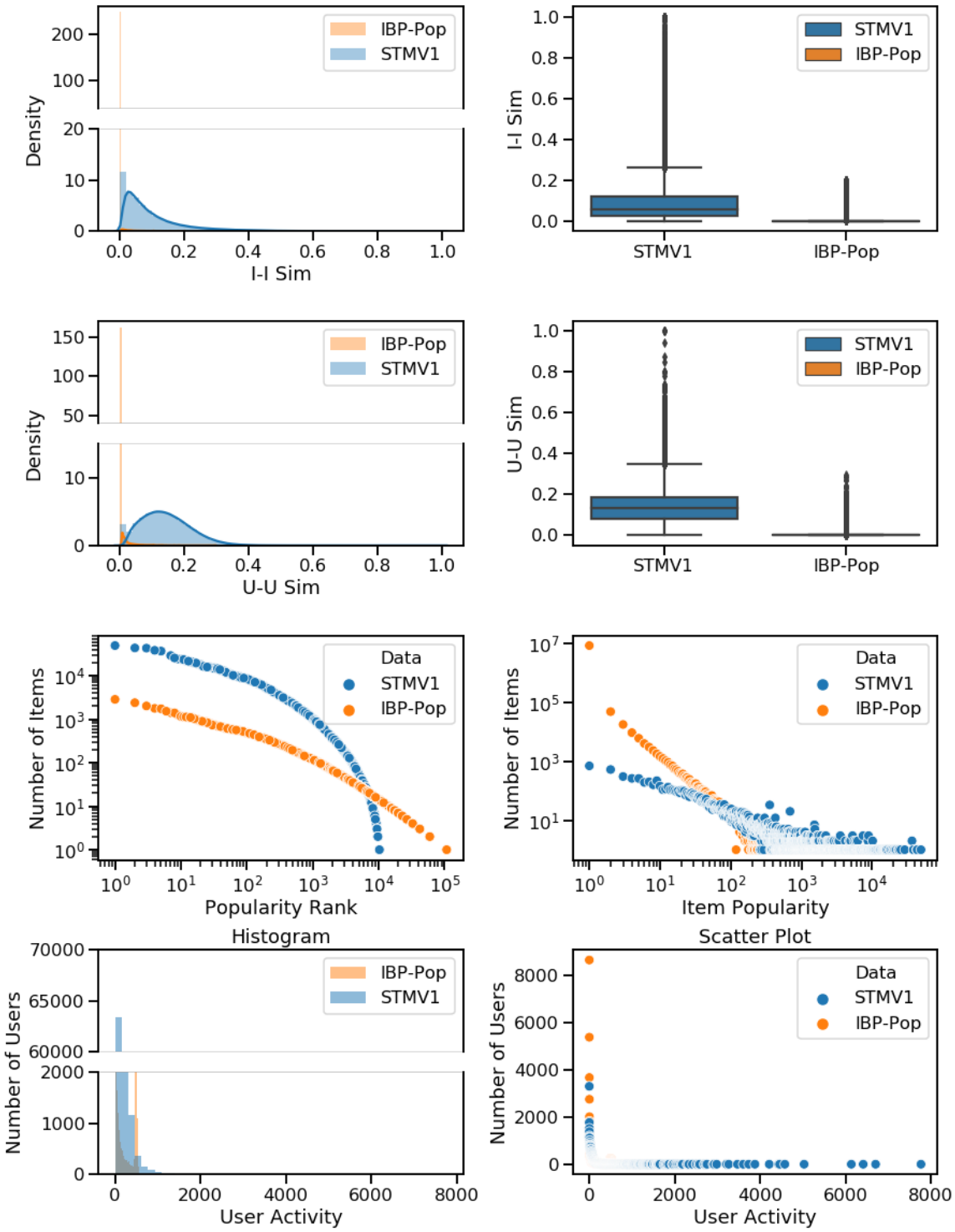

Figure 67: IBP-Pop optimized for User Act on STMV1.

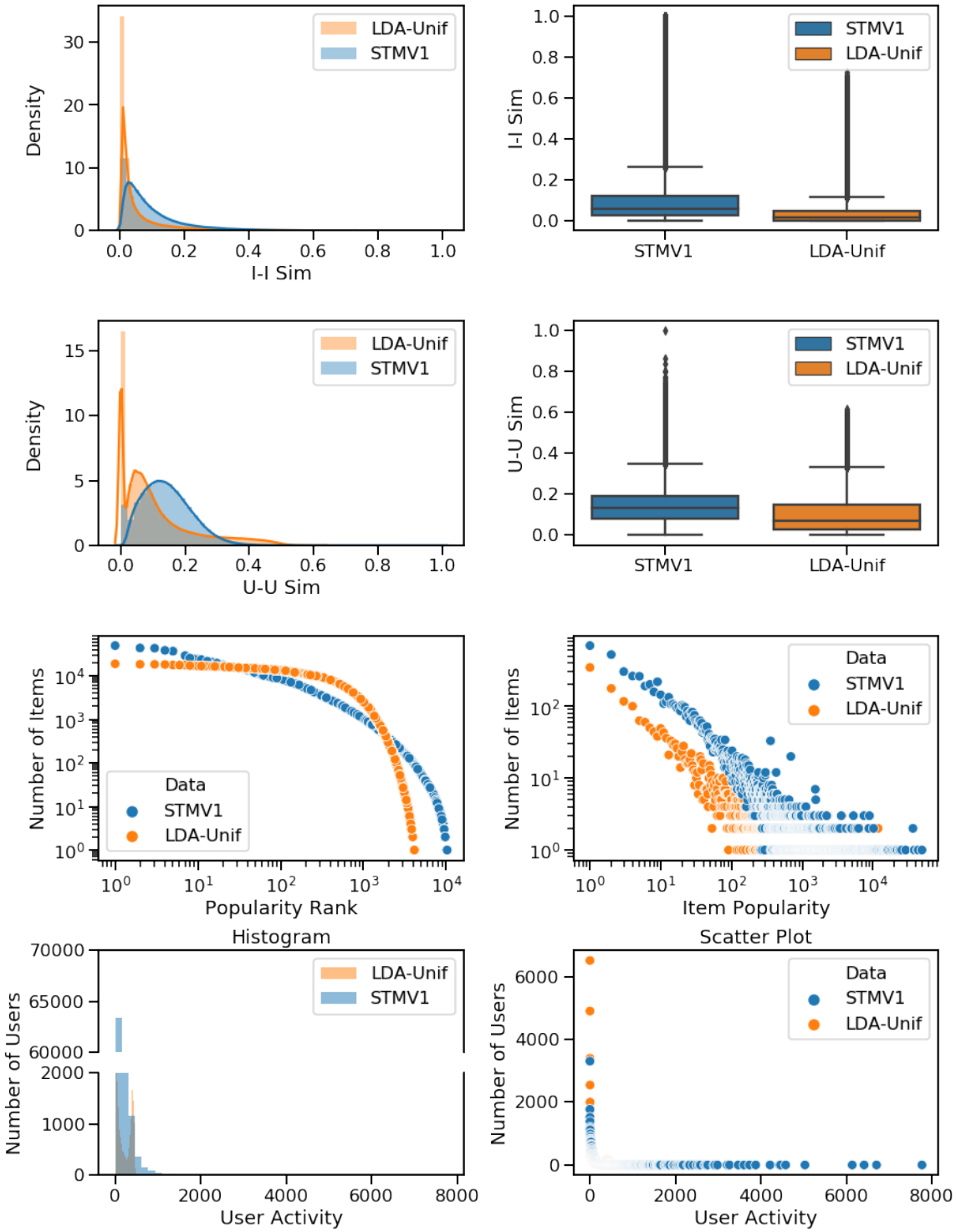

Figure 68: LDA-Unif optimized for U-U Sim on STMV1.

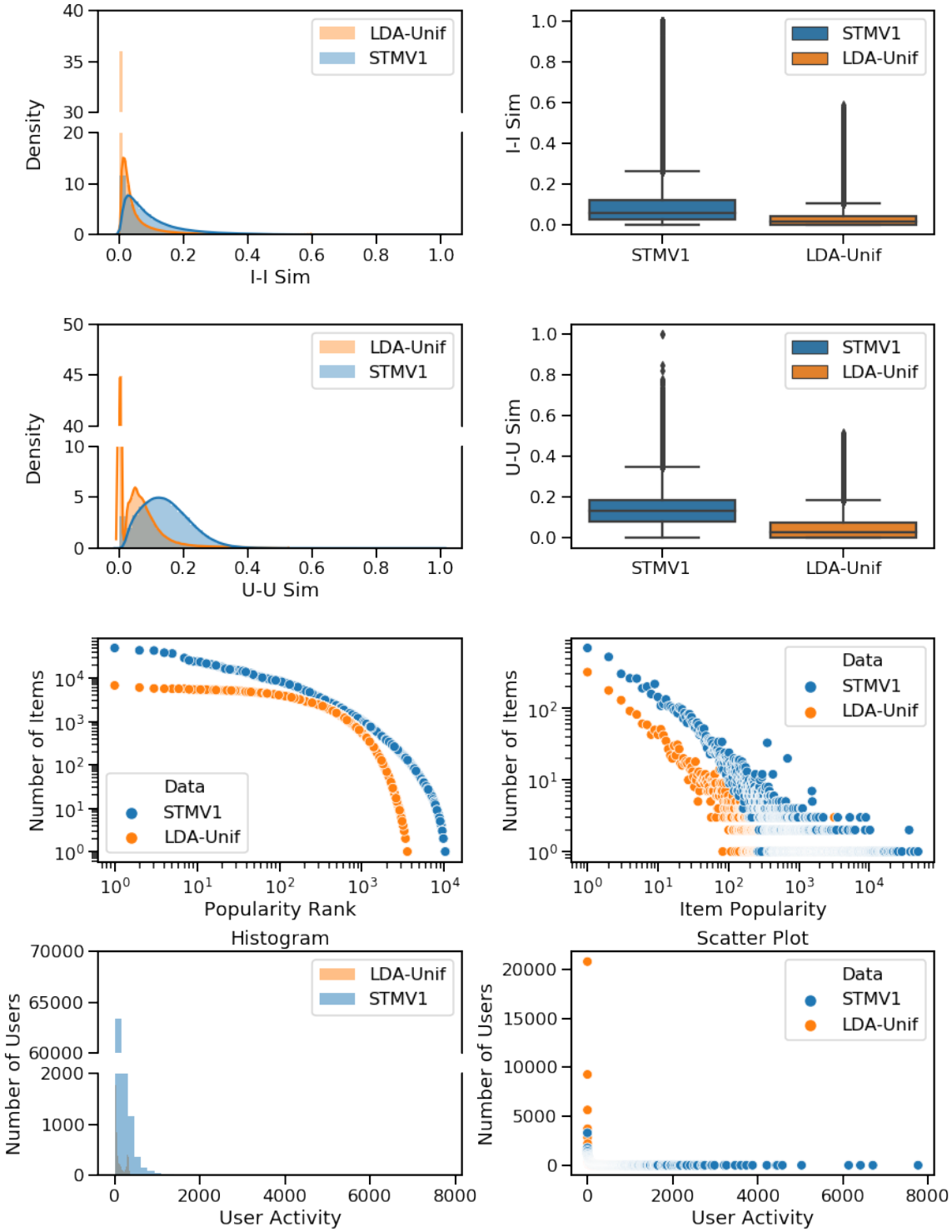

Figure 69: LDA-Unif optimized for I-I Sim on STMV1.

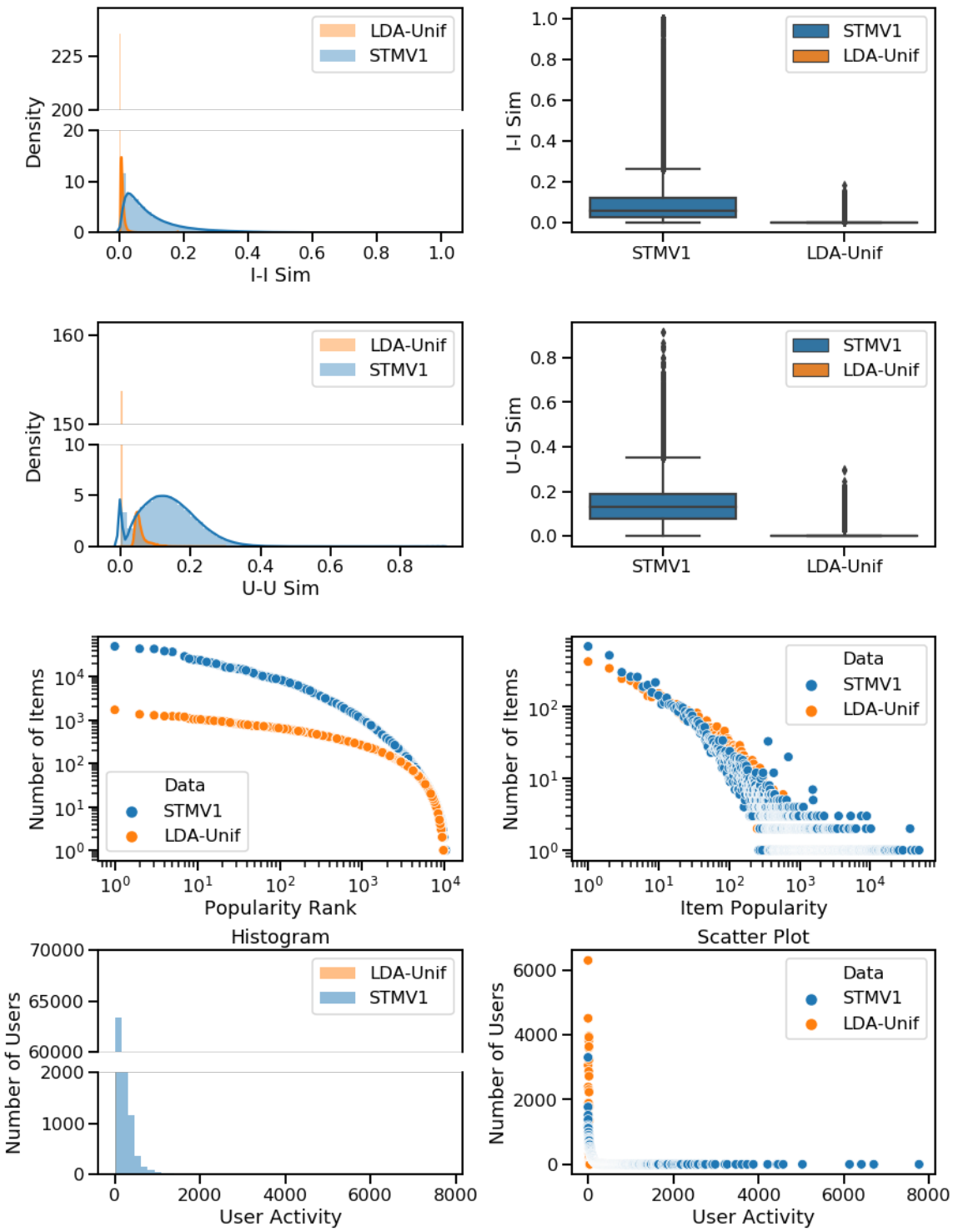

Figure 70: LDA-Unif optimized for Item Pop on STMV1.

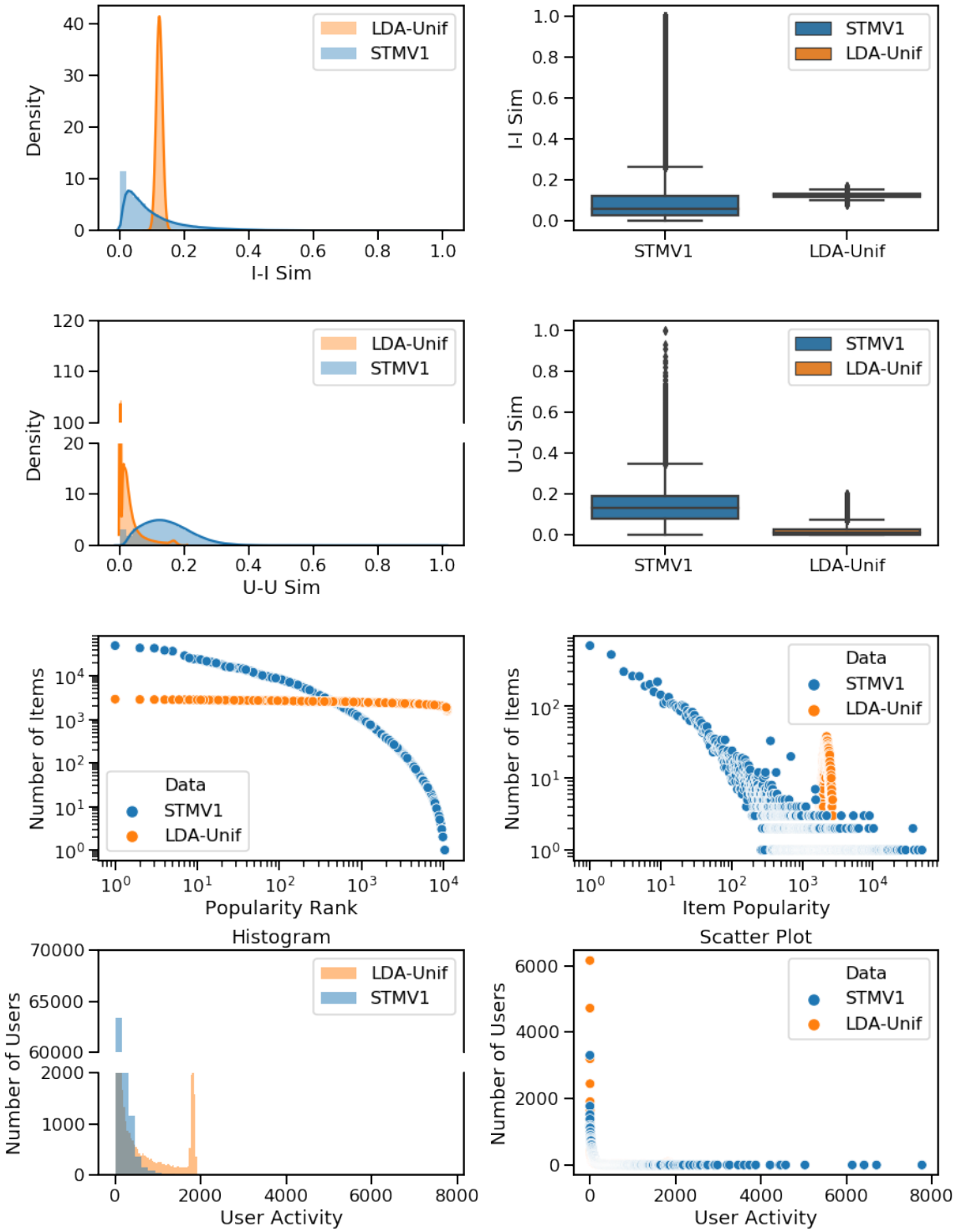

Figure 71: LDA-Unif optimized for User Act on STMV1.

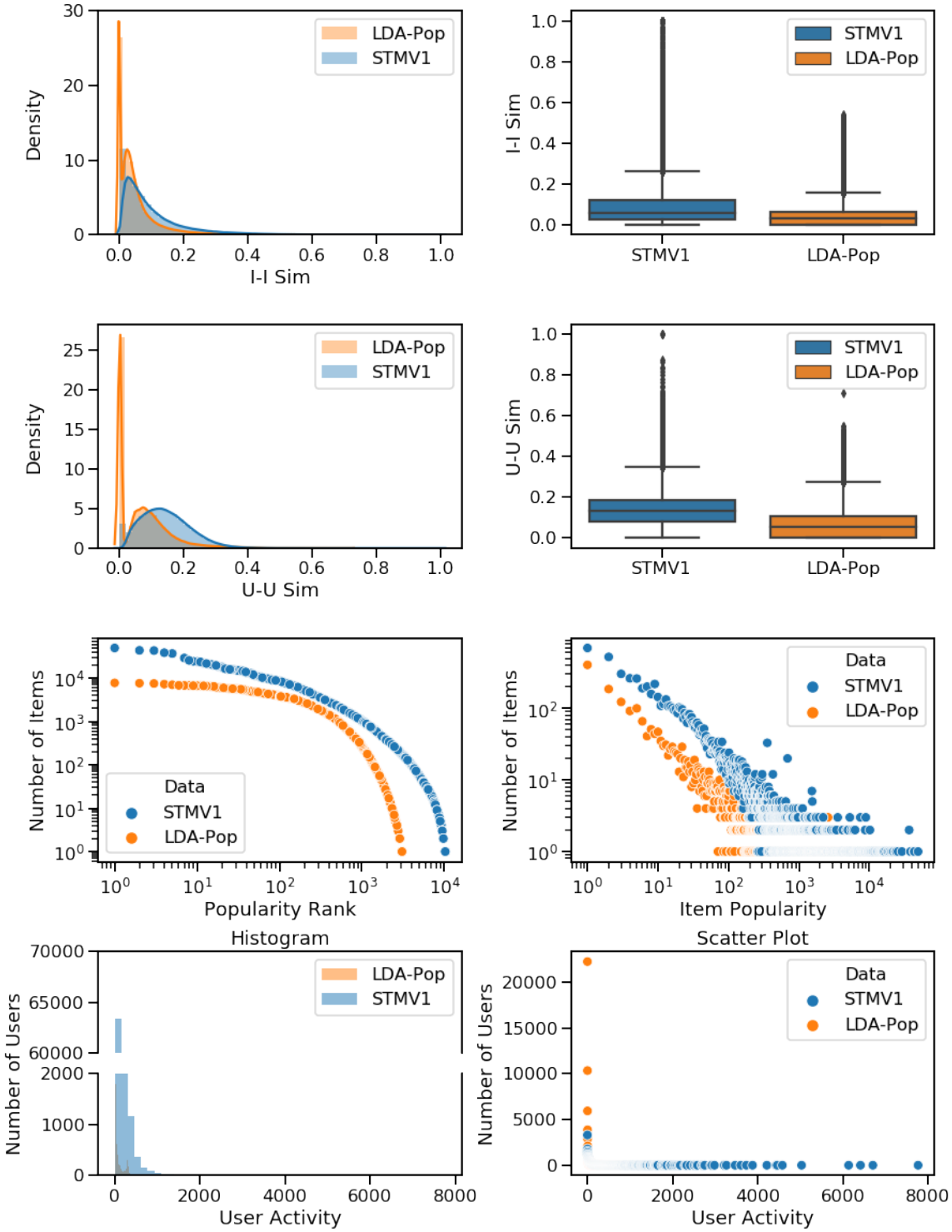

Figure 72: LDA-Pop optimized for I-I Sim on STMV1.

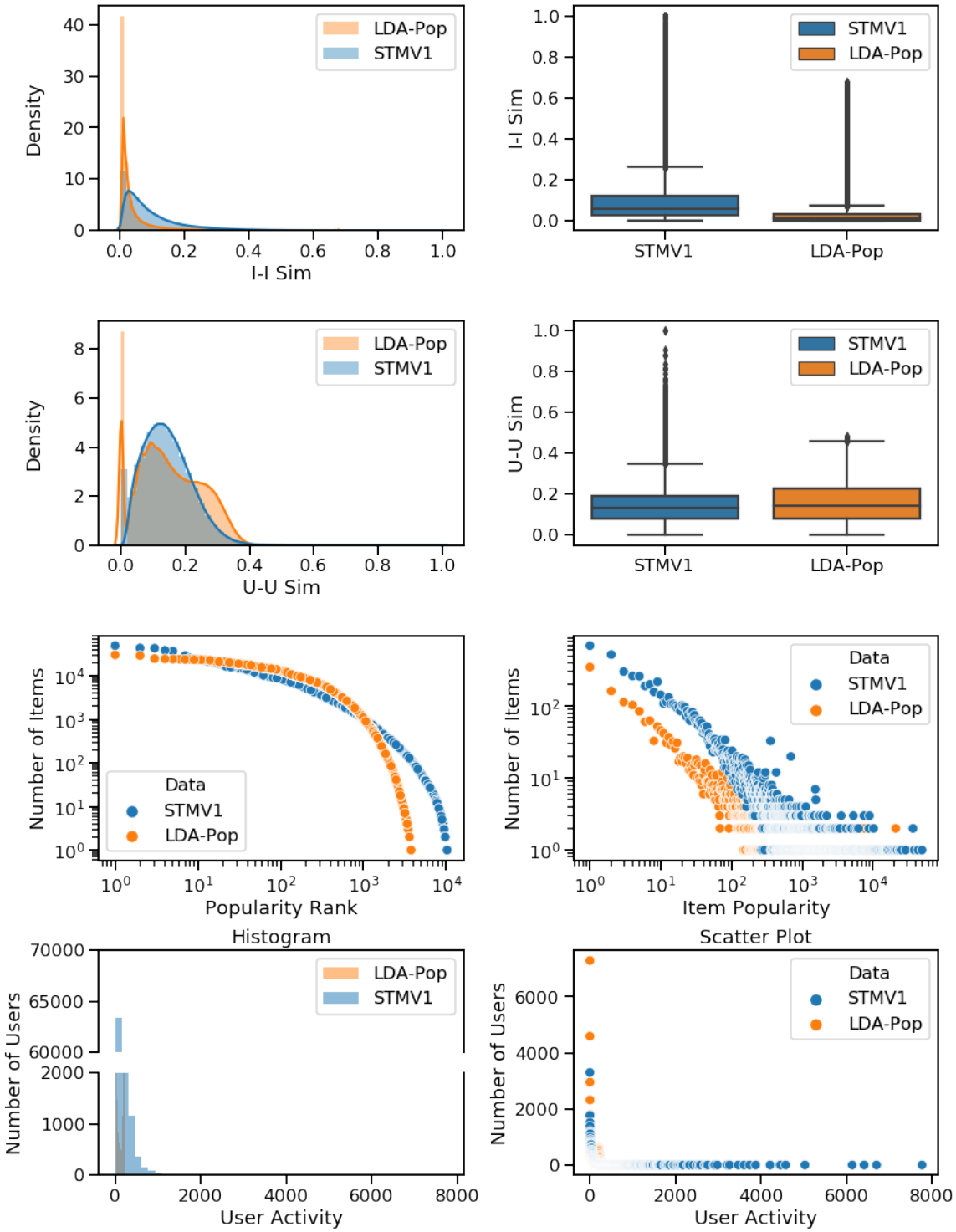

Figure 73: LDA-Pop optimized for U-U Sim on STMV1.

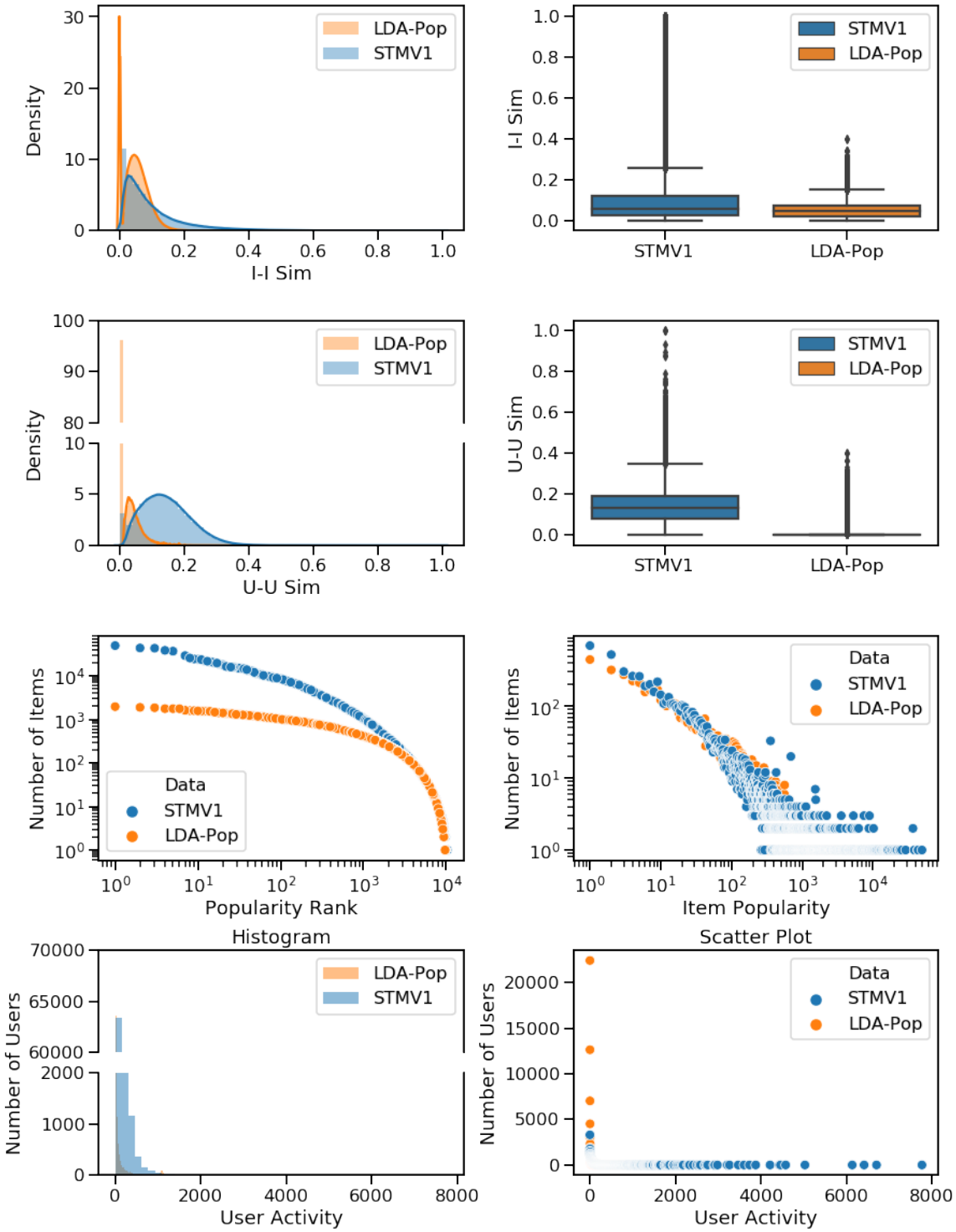

Figure 74: LDA-Pop optimized for Item Pop on STMV1.

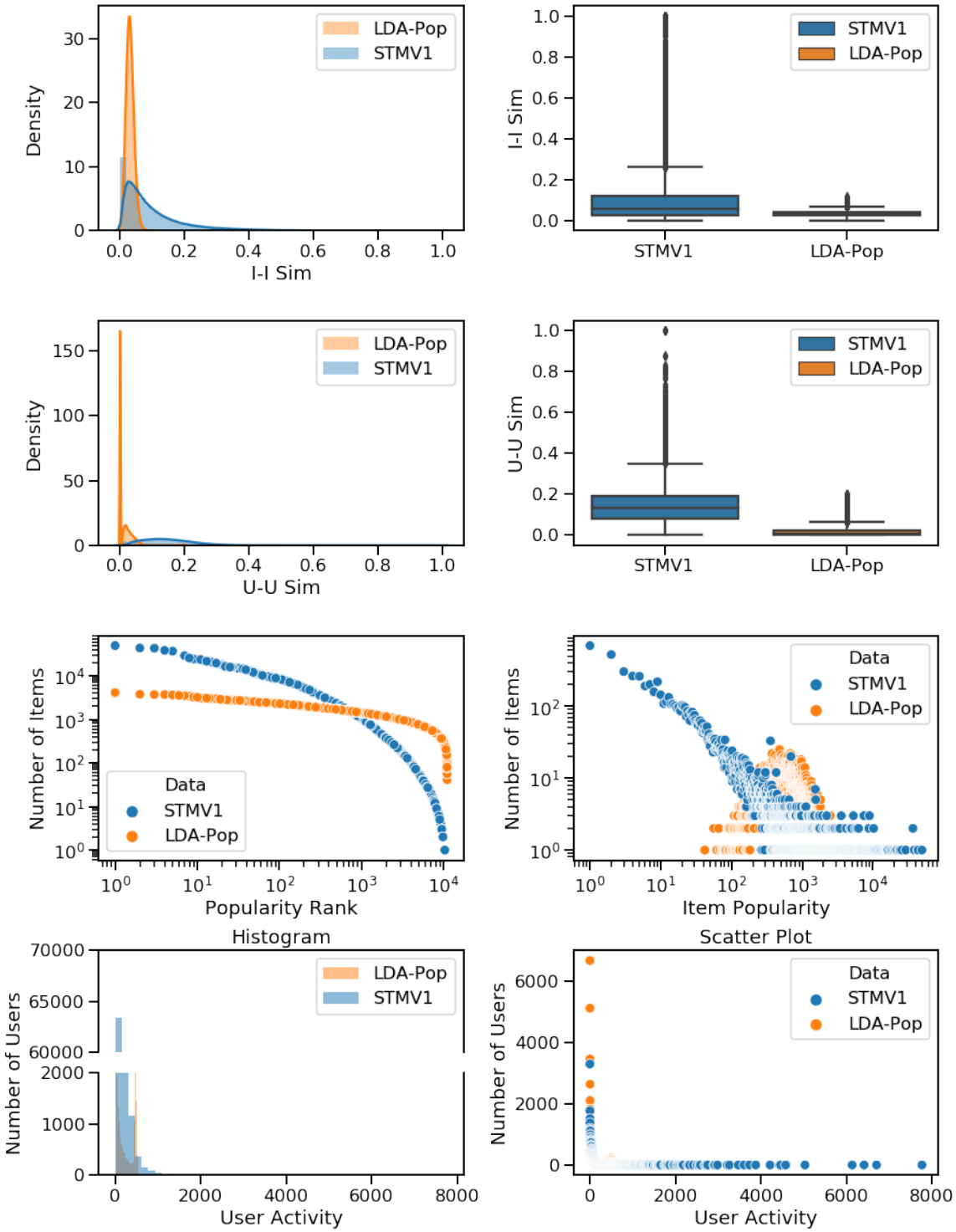

Figure 75: LDA-Pop optimized for User Act on STMV1.