# Region Sampling and Estimation of GeoSocial Data with Dynamic Range Calibration

Yanhua Li [#], Moritz Steiner [*], Jie Bao [‡], Limin Wang [†], and Ting Zhu [§]

[#] *HUAWEI Noah's Ark Lab, China,* [*] *Akamai Inc., CA, USA*
[‡] *University of Minnesota, Twin Cities, MN, USA,* [†] *Bell Labs, NJ, USA*
*Binghamton University, NY, USA*
li.yanhua1@huawei.com, moritz@akamai.com, baojie@cs.umn.edu,
liminwang@bell-labs.com, tzhu@binghamton.edu

***Abstract*—Location based social networks (LBSNs) are becoming increasingly popular with the fast deployment of broadband mobile networks and the growing prevalence of versatile mobile devices. This success has attracted great interest in studying and measuring the characteristics of LBSNs, such as Facebook Places, Yelp, and Google+ Local. However, it is often prohibitive, and sometimes too costly, to obtain a detailed and complete snapshot of a LBSN due to its usually massive scale. In this work, taking Foursquare as an example, we focus on sampling and estimating restricted geographic regions in LBSNs, such as a city or a country. By exploiting the application programming interfaces (APIs) provided by Foursquare for geographic search, we first introduce how to obtain the "ground truth", namely, a complete set of all venues (i.e., places) in a specified region. Then, we propose random region sampling algorithms that allow us to draw representative samples of venues, and design unbiased estimators of regional characteristics of venues. We validate the efficiency of our sampling algorithms on Foursquare using complete datasets obtained from 12 regions, such as Switzerland, New York City and Los Angeles. Our results are applicable to perform sampling and estimation in all GeoDatabases, such as Facebook Places, Yelp, and Google+ Local, which have similar venue search APIs as Foursquare. These location service providers can also benefit from our results to enable efficient online statistic estimation.**

## I. Introduction

The fast development of broadband mobile networks and the increasing prevalence of versatile mobile devices, e.g., smart phones and tablets, help to boost the popularity of location based services. For example, Foursquare [2], one of the most popular location based social networks (LBSNs), had more than 10 million registered users with 1 billion check-ins in September 2011 [8], and the number of check-ins doubled to 2 billion in only half a year [5]. Location based services have also been embedded into other leading online social networks, such as Facebook Places [1] and Google+ Local [7], as an important feature.

This success of LBSNs has generated great interest in studying and measuring their characteristics, e.g.,[28], [31], [32], [34], [24], [13]. By collecting and investigating large scale datasets, these studies provide useful insights into the understanding of different aspects of LBSNs, such as popular route discovery and user mobility prediction. The datasets used in these investigations are collected either by performing exhaustive search from geographic regions, e.g., a few US cities, or random walk (or breath first search, in short BFS) to traverse the entire network via users' friendship relations and check-ins to venues. However, exhaustive search is in general costly, and thus can only be applied to relatively small regions. On the other hand, random walk and BFS type of traversal sampling algorithms have no control of the locality of the sampled users and venues. Therefore it would cause inefficiency when the objective is to sample restricted regions instead of entire networks. For example, if the region being sampled is California, random walk or BFS would waste many sampling steps to visit non-California venues or users. Moreover, the users and venues are often disconnected, which further limits the capability of these traversal sampling techniques.

In this paper, we make the first attempt to study sampling and estimation in LBSNs, and aim to obtain a representative sample set of venues (i.e., places) in a restricted region from a large scale geospatial dataset of a LBSN. The obtained sample dataset can be used in many applications to estimate various statistics, including total number of venues, check-in distributions, etc.

Taking Foursquare as an example, we study the functionality of its application programming interfaces (APIs) for venue search, and develop an exhaustive search algorithm that can extract a complete list of venues from a geographic region. The exhaustive search algorithm as a building block allows us to design random region sampling algorithms and unbiased estimators for many venue properties, such as total number of venues and the distributions of the number of check-ins, tips and users. We highlight and summarize our contributions as follows.

- Foursquare provides a venue search API that returns a list of venues in a given geographic region (Section III). However, the rate limit (500 API queries per hour per authorized user) and the return limit (up to 50 venues per query) restrict the query capacity of the venue search API and make large scale searches costly. By exploiting the unique properties of this API, we develop an exhaustive search algorithm that allows us to efficiently obtain a complete list of venues within a geographic bounding box. (Section IV.)
- Utilizing the exhaustive search mechanism as a building block, we first develop a simple random region sampling algorithm, where the objective region is divided into pre-

determined equal-sized boxes, and the sampling process uniformly at random chooses from these boxes to draw samples via the exhaustive search. The unbiasedness of the simple sampling algorithm allows us to design unbiased estimators for various regional characteristics, such as number of venues, check-in and tip distributions. (Section V.)

• By taking the venue density distribution into consideration, we introduce a dynamic random region sampling algorithm that can adaptively adjust the sizes of the sampled boxes based on the obtained samples so as to achieve more accurate estimation results under the same query budget constraints. (Section VI.)

• We run these two random region sampling algorithms on 12 regions all over the world. By comparing their estimation results with the "ground truth" obtained via exhaustive search algorithm, our random region sampling algorithms exhibit outstanding performance for estimating the total number of venues and label distribution. (Section VII.)

• Our results are applicable to all GeoDatabases, such as Facebook Places, Yelp, and Google+ Local, which have similar venue search APIs as Foursquare, and these location service providers can also benefit from our estimation to enable efficient online statistic estimation (Section VIII.)

## II. Related work

To the best of our knowledge, we are the first to study the geographic sampling and estimation in LBSNs. In this section, we discuss two topics that are closely related to our work, including (1) location-based social networks and (2) network sampling techniques.

**Location-based social networks.** LBSNs have attracted great interest to investigate and analyze such large-scale data and their implications in improving the location-based online services, e.g., user mobility prediction, friendship recommendations, etc. In [31], [32], the authors studied the correlation between the friendship relations and the user location relevance, and designed social link prediction system by incorporating the users' location information. In [21], [35], the authors used location history to infer user similarity. In [37], [36], the authors developed efficient algorithms for personalized location recommendation and advertising in location based social networks. In [13], [24], [34], [11], [38], the authors examined large-scale (uncertain) trajectory data, with applications in revealing user mobility patterns and providing personalized routes recommendations. All these efforts rely on mining and understanding massive LBSN data, and the representativeness (or biasedness) of the dataset may significantly impact the results. As we will discuss in this paper, it is challenging to retrieve a representative sample set from large scale LBSNs, due to various practical constraints, e.g., query rate limit and unknown API return principle. Below we provide a comprehensive introduction on the state-of-the-art network sampling techniques and illustrate why they are not applicable to sampling in LBSNs.

**Network sampling techniques.** There is a wide range of sampling algorithms in the literature [40], [29], [16], [19], [30], [10], [15], [27]. Here, we present them in three categories, i.e., *non-probability sampling*, *random vertex/edge sampling*, and *random walk sampling* [25].

(i) The non-probability sampling including breadth-first search (BFS) [20] and snow ball sampling[9], is easy to implement, but due to the uncontrollable sampling bias to high degree nodes, it is hard to obtain unbiased estimates of network statistics.

(ii) Considering the online social network (OSN) as a graph, where vertices and edges represent users and friendship relations, respectively, random vertex (resp. edge) sampling technique [30] chooses each vertex (resp. edge) uniformly at random, thus enables unbiased estimations of network properties, such as total number of users, distribution of number of friends. These techniques require known vertex/edge id spaces, e.g., user id. In LBSNs, the id space is often unknown, e.g., hashed id space, or the id space is very sparse, namely, too costly to generate a valid id. As we observed, the venue id in Foursquare consists of $24$ characters, in $[0-9]$ and $[a-f]$, which generates a huge and sparse id space, making it impossible to apply random vertex/edge sampling due to the high cost of generating a valid venue id. Recently, the authors in [40] propose a random prefix sampling to estimate the total number of YouTube videos, which relies on the random prefix search API provided by Google. Unfortunately, LBSNs do not offer such public interface for random prefix search.

(iii) Random walk sampling performs as if a random walker wanders via links between OSN users, with the probability of each vertex being visited equal to its stationary probability. This property enables designing unbiased estimators for network statistics, such as network size [18] and degree distribution [30], [16]. There are many variations of random walk sampling techniques. For example, Metropolis-Hastings random walk (MHRW) [39], [12] realizes sampling with uniform stationary distribution. In [30], [16], the authors utilized multiple parallel random walks to alleviate the problem of disconnectedness in OSNs to prevent the walker from being trapped in a local community. Directed Unbiased Random Walk (DURW) [29] is designed to realize unbiased out-degree estimation in OSNs with both bi- and uni-directional links. Random walk is a simple and powerful sampling tool in many scenarios. In LBSNs, however, when the objective is to sample and estimate a restricted geographic region, the random walk would be very costly and inefficient, due to the frequent visits to entities outside the objective region.

In summary, none of the existing sampling techniques are applicable to collect representative samples from a restricted region in LBSNs. In this paper we provide a systematic study on how to efficiently collect data from LBSNs that preserve the network statistics, such as network size and check-in distributions, with unbiased estimators.

## III. Preliminaries

In this section, we first describe the geosocial objects in a LBSN (using Foursquare [2] as an example). After that, we introduce the search API we used in our sampling framework.

### A. GeoSocial Data

**Venue:** A Foursquare venue is a physical location. It can be a place of business office or private residence where Foursquare users can check in, e.g., a restaurant, train station or movie theater. Foursquare venues are all generated by Foursquare users, and each venue is assigned with a unique venue id, consisting of $24$ characters in $[0-9]$ and $[a-f]$. When creating a venue, the user is required to provide a few attributes of the venue, such as the venue's location, name, address, and category. Other attributes, such as zip code, cross street, twitter contact, and phone number, are optional.

**House Rules:** Since all the venue information is generated by users, people may create venues with inaccurate or wrong information. The Foursquare community establish the House Rules [4] to guide users' behaviors. For example, it suggests "don't check in when you're not at a place" and "don't create venues that do not exist."

**Venue location:** When creating a venue, a user needs to specify the venue's location by both dropping a pin on the map and providing the address, including street number, city, state, and country.

**Venue Category:** Foursquare defines a 3-level hierarchical structure of categories that are applied to venues. There are nine top level categories: Arts & Entertainment, College & University, Food, Professional & Other Places, Nightlife Spot, Great Outdoors, Shop & Service, Travel & Transport and Residence. Users need to specify one of these nine categories and a subcategory when creating a venue. Foursquare launched the category feature on March 10th 2010. Hence, venues created before that date may not have any category information. A special case is "Home (Private)" category, which is a subcategory of Residence. Venues in this category are privacy sensitive, and it can creep people out to see non-friends checking in at these venues. Hence, the House Rules [4] suggest "Don't check into someone else's home if you're not there" and "Only create a Foursquare venue for someone else's home if you have the permission of the resident/owner." Moreover, Foursquare ensures that the sensitive details of a Home (Private) venue will be visible only to its owner and her friends, for example, a zoomed out view on the map instead of its real location will be shown to any non-owner or non-friend. When a home venue is included in the results returned by the Foursquare venue search API, its location field is fuzzed to an approximate area to protect the users privacy, i.e., many "Home (Private)" venues in the same vicinity could exhibit the same location information in API query results. Thus, in our study, we only focus on those venues not in "Home (Private)" category, namely, "non-home" venues, by simply filtering out those venues from our search results. We will use venues instead of "non-home" venues for simplicity.

**Check-in:** Via web or a mobile application, Foursquare allows registered users to explicitly post their presence at a venue that will be displayed on their connected friends' Foursquare sites. Users will be awarded with points and coupons at check-ins.

**Tip:** Foursquare users can add "Tips" to venues that other users can read, which serve as suggestions for great things to do, see or eat at the location.

### B. Foursquare search API

Foursquare provides an application programming interface (API) [3] for the application developers to access the data in Foursquare. Most importantly, the venue search API is able to post location-aware requests, which takes a rectangular geographic region as an input (specified by the south-west and north-east corners in latitude and longitude), and returns a list of venues in that region, with rich information, including venue name, geo-location (in latitude and longitude), address, contact, venue category, check-ins, tips, and number of past users. In our study, we use the Google Geocoding API [6] to retrieve a bounding box (i.e., the south-west and north-east corners) for an objective geographic region, e.g., New York City as $\texttt{ne} = 40.917577, -73.700272$ and $\texttt{sw} = 40.477399, -74.25909$.

However, the Foursquare API is subject to a querying ratio constraint, i.e., 500 queries per hour per authenticated account, and a result size constraint, i.e., up to 50 venues returned for a query. Moreover, the results obtained from the venue search API are repeatable, namely, if there is no venue changes in a bounding box, the return of querying for that bounding box is consistent.

## IV. Exhaustive region search

As seen from last section, the constraints of the Foursquare venue search API limit its capability to complete some much wanted tasks, e.g. returning all venues (potentially a large number) in an arbitrary region. These constraints are in fact not unique to Foursquare, and are supported by justified operational or business rationales, which will be discussed more in later sections. In this section, however, we strive to explore the unique features of this API and design an exhaustive search algorithm to retrieve a complete set of venues from a given geographic region.

### A. Smallest resolution

The Foursquare venue search API requires a pair of south-west and north-east locations, i.e.,$\texttt{sw} = (\texttt{lat}_\texttt{s}, \texttt{lng}_\texttt{w})$ and $\texttt{ne} = (\texttt{lat}_\texttt{n}, \texttt{lng}_\texttt{e})$, to specify the bounding box for the search. Although not specified by Foursquare, by applying the venue search API to different regions all over the world, we observe that there exists a minimal size of the bounding box that the API can handle. To be precise, we define the size of the bounding box as $(\texttt{lat}_\texttt{n} - \texttt{lat}_\texttt{s}) \times (\texttt{lng}_\texttt{e} - \texttt{lng}_\texttt{w})$. By performing venue search queries for 1 million bounding boxes with different sizes, we observe that when the bounding box is as small as $a_q = s_q^2$, where $s_q = (4.8828 \times 10^{-5})^\circ$, queries on any sub-boxes inside the objective box will return the same set of venues as that of the objective box. This indicates that the smallest resolution that the venue search API can handle is about $a_q$, equivalent to roughly $4.8^2 m^2$ at equator, and $2.4^2 m^2$ at $60^\circ$ latitude.

Given such smallest resolution, any (rectangular) region can be "tessellated" into a grid of the smallest squares. The number of such squares is a function of the area of the region. This yields a baseline venue search algorithm to obtain all the venues that can be returned by the venue search API, by completely scanning the region with boxes of side length $s_q$.

### B. Exhaustive venue search algorithm

The baseline scanning algorithm is in fact very costly, due to the query rate limits, and potentially a large number of queries with no venues returned. Taking New York City as an example, in total $2.5378 \times 10^7$ queries are needed to complete, equivalent to $2,203$ days.

We propose a top-to-down exploration algorithm to collect venues by two dimensional binary division, which works as follows. Let $G_0$ be a geographic region, and its size does not exceed the maximum size of venue search API, namely, $\bar{a} = 10,000 km^2$, equivalent to roughly $0.81$ lat-lng area. At level 0, an API query is performed on $G_0$ directly, and it returns a venue set $V_0(G_0)$, where $|V_0(G_0)| \leq \bar{b} = 50$ venues, because it cannot exceed the return limit. Then, at level 1, we divide $G_0$ into four equal size boxes $G_1, \cdots, G_4$, with latitudinal and longitudinal side lengths half of $G_0$, and perform four individual API queries on them. At the following steps, the bounding boxes are divided in the same fasion. We stop dividing a (sub-)bounding box, when the returned venue set on it satisfies the two *stopping conditions*: the API space limitation and the smallest resolution. To be precise, when (1) the number of returned venues in a bounding box is less than 50; or (2) the size of the bounding box hits the smallest resolution, we know that a complete list of venues in the bounding box is returned, and no further division on the box is needed.

We call the above *exhaustive venue search algorithm*, which enables us to collect *all venues* in $G_0$. The pseudo-code is presented in Algorithm. 1, where it is a recursive process with input as $G_0$, the objective region, and $V_0(G_0)$, the returned venue list by querying directly on $G_0$. Its output includes all venues collected from $G_0$ and the number of queries performed.

**Algorithm 1** $\texttt{InSearch}(G_0, V_0(G_0))$

1: **INPUT:** $G_0$ with size less than $\bar{a}$, and $V_0(G_0)$;
2: **OUTPUT:** $V(G_0)$ and # of queries $B_0$;
3: $V(G_0) = \emptyset$ and $B_0 = 1$;
4: **if** $s(G_0) \leq s_q$ or $|V_0(G_0)| < \bar{b} = 50$; **then**
5: Return $V_0(G_0)$ and $B_0$;
6: Divide $G_0 = G_1 \cup G_2 \cup G_3 \cup G_4$ in equal size;
7: Perform 4 API searches to retrieve $V_0(G_1)$, $V_0(G_2)$, $V_0(G_3)$, $V_0(G_4)$;
8: **for** $i = 1 : 4$ **do**
9: $[V(G_i), B_i] = \texttt{InSearch}(G_i, V_0(G_i))$;
10: $B_0 = B_0 + B_i$;
11: $V(G_0) = \cup_{i=1}^{4} V(G_i)$;
12: Return $V(G_0)$ and $B_0$;

Comparing to the simple scanning method, it significantly reduces the number of queries needed. Taking again New York City as an example, only $248,860$ queries are needed, equivalent to $21.6$ days, reducing the running time by two orders of magnitude compared to the simple scanning approach.

TABLE I
COMPLETENESS OF EXHAUSTIVE VENUE SEARCH ALGORITHM

| Regions | Total venues | Collected venues | Missing venues | Missing ratio |
|---|---|---|---|---|
| Orlando | 46945 | 46516 | 429 | 0.91% |
| Cairo | 3391 | 3364 | 27 | 0.80% |
| NYC | 453070 | 449284 | 3786 | 0.84% |
| Singapore | 400520 | 397088 | 3432 | 0.85% |
| Colorado | 166190 | 164908 | 1282 | 0.77% |
| Slovenia | 66155 | 65632 | 523 | 0.79% |
| Sydney | 12721 | 12617 | 104 | 0.82% |
| Switzerland | 141565 | 140595 | 970 | 0.69% |
| Seoul | 329120 | 326253 | 2867 | 0.87% |
| Paris | 96147 | 95353 | 794 | 0.83% |
| Los Angeles | 193262 | 191727 | 1535 | 0.79% |
| Wyoming | 14179 | 14079 | 100 | 0.71% |

**Completeness of exhaustive venue search algorithm.** We applied the exhaustive venue search algorithm on 12 geographic regions listed in Table I, and we compare the venues collected using Algorithm 1 vs the simple scanning method (on 40 clusters). We can see at the worst case, there are $0.91\%$ missing venues observed when using Algorithm 1, where all the missing venues are due to the reasons, such as a) it is a very new venue; b) the venue is reported and removed. Hence, the exhaustive venue search algorithm can retrieve a complete list of venues.

## V. SIMPLE RANDOM REGION SAMPLING

The exhaustive venue search algorithm developed in Section IV enables us to collect a complete set of venues for a given geographic region. However, it is still very costly to perform such an exhaustive search. For example, as stated earlier, crawling New York City requires about 21 days, while going over a larger area such as a US state or a European country needs a few months. Hence, this method is not suitable to timely learn and estimate the statistics of LBSN properties, i.e., the total number of venues, check-in distributions, etc. *Sampling* is a more efficient and practical alternative. In this section, we present a simple random region sampling algorithm for LBSN built on top of Foursquare's venue search API, and design unbiased estimators for the total number of venues and venue label distributions.

### A. Model of simple random region sampling

Given a large geographic region, denoted by $G$, it represents a "rectangular" area bounded by the south-west location $(\texttt{lat}_\texttt{s}, \texttt{lng}_\texttt{w})$, and north-east location $(\texttt{lat}_\texttt{n}, \texttt{lng}_\texttt{e})$. The total number of venues in it is denoted as $N$. The simple random region sampling algorithm *(SRRS)* works as follows.

**Initial stage.** Given an initial side length $s_0$ in lat-lng degree, $G$ is partitioned into $n$ non-overlapping equal-sized boxes, i.e., $G = G_1 \vee \cdots \vee G_n$ and $G_i \wedge G_j = \phi$, for $\forall i \neq j, 1 \leq i, j \leq n$. Let $\mathcal{G} = \{G_1, \cdots, G_n\}$ denote the set of all boxes, with $N_1, \cdots, N_n$ venues, respectively, where $\sum_{i=1}^{n} N_i = N$.

**Sampling stage.** At each sampling step $t \geq 1$, a box $X_t \in \mathcal{G}$ is chosen uniformly at random. $X_t$ is thus considered as a sample of $G$, where various statistics of $X_t$ are represented as functions on $X_t$. For instance, the number of venues can be viewed as a mapping $f : \mathcal{G} \to \mathbb{R}$ from each region $X_t \in \mathcal{G}$ to the number of venues in $X_t$, namely $N_t = f(X_t)$. The value of $N_t$ can be obtained by using LBSN's query API. For example, when a box $X_t$ is chosen from $\mathcal{G}$, an exhaustive search on $X_t$ as described in Section IV is performed to collect all venues located in $X_t$. Each such exhaustive search incurs a cost, i.e., a certain number of API queries, and we are given a sampling budget $B$ as the maximum number of API queries allowed. In our analysis, we assume that $B$ is always sufficiently large, and we defer the detailed discussion on the actual cost $B$ to Section V-C. If $X_t$ has been chosen before, we simply keep this sample by copying previous sampled results for the same box, without performing another exhaustive search.

**Tabulating stage.** The sampling stage repeats until the sampling budget $B$ is exhausted. If the budget runs out while exhaustively searching the last box, that box will be ignored. Then, a total of $m$ independent samples are obtained, i.e., $X_1, \cdots, X_m$, with each $X_t \in \mathcal{G}$. Estimators discussed in next subsection will then be applied to these $m$ samples to generate corresponding estimates of LBSN properties.

The SRRS algorithm has a very nice property that the samples drawn are unbiased, namely, every individual venue in the population has an equal chance of being selected, as stated in Theorem 1.

**Theorem 1 (Unbiased Sample).** *The probability $Pr(v)$ of each venue $v \in G$ being sampled using SRRS sampling is equal, i.e., $Pr(v) = 1/n$.*

*Proof:* Each box $G_i$ $(1 \leq i \leq n)$ has an equal probability $Pr(G_i) = 1/n$ to be chosen. The conditional probability of each venue $v$ being sampled given that $G_i$ is sampled is $Pr(v|G_i) = 1$, if $v \in G_i$; and $Pr(v|G_i) = 0$ otherwise. Applying the law of total probability yields the probability of $v$ being sampled as $Pr(v) = \sum_{i=1}^{n} Pr(G_i)Pr(v|G_i) = \frac{1}{n}$. ■

Given the unbiased simple random region sampling algorithm, in the next subsection we will introduce estimators for various characteristics of a geographic region $G$.

### B. Estimators

An estimator is a function of a sequence of observations that outputs an estimate of an unknown population parameter. Using the unbiased samples obtained by SRRS, in this section, we present unbiased estimators of the characteristics of the objective geographic region $G$, such as the total number of venues, the check-in distribution, and the tip distribution.

**Total number of venues.** We aim to estimate the total number of venues $N$ in a geographic region $G$. Given a total of $n$ non-overlapping partitions $G_1 \vee \cdots \vee G_n$ of $G$, we propose an estimator $\hat{N}$ to estimate $N$ in Theorem 2.

**Theorem 2 (Estimator of the total number of venues).** *With a sampling budget $B$, SRRS collects $m$ samples $X_1, \cdots, X_m$, where each $X_t \in \mathcal{G}$ $(1 \leq t \leq m)$. Then, $\hat{N}$ in eq.(1) is an unbiased estimator of $N$.*

$$\hat{N} = \frac{n}{m} \sum_{t=1}^{m} f(X_t). \tag{1}$$

*Proof:* Since each $X_t$ is chosen independently and uniformly at random from the same population space $\mathcal{G}$, $f(X_t)$'s are all independent and follow the same distribution, and thus have the same expectation as $E[f(X_t)] = \mu = N/n$. Using the linearity of the expectation, we prove $E[\hat{N}] = N$. ■

**Label distribution.** We estimate the label distributions of venues in a geographic region $G$. The label of a venue could be any property, such as the number of check-ins, the number of users who have checked in the venues, the number of tips. Denote $\mathcal{L}$ as the label set and $\ell \in \mathcal{L}$ each individual label. Indicator function $\mathbf{1}(\ell, v)$ denotes if $\ell$ is venue $v$'s label or not, namely, $\mathbf{1}(\ell, v) = 1$, if $\ell$ is $v$'s label, and $\mathbf{1}(\ell, v) = 0$, otherwise. The label distribution is represented by $p_\ell = \frac{1}{N} \sum_{v \in G} \mathbf{1}(\ell, v)$, with $\ell \in \mathcal{L}$. Recall that $G$ is divided into $n$ non-overlapping boxes $G = G_1 \vee \cdots \vee G_n$. Define $g_\ell : \mathcal{G} \to \mathbb{R}$ to be the mapping from each box $X_t \in \mathcal{G}$ to the number of venues with label $\ell$ in $X_t$, i.e., $g_\ell(X_t) = \sum_{v \in G_t} \mathbf{1}(\ell, v)$. To estimate the label distribution, we introduce an estimator $\hat{p}_\ell$ in Theorem 3, which can be proven by the ratio form of the law of large numbers (Theorem 17.2.1 on p.426 in [26]). The complete proof is omitted here for brevity.

**Theorem 3 (Estimator of label distribution).** *With a sampling budget $B$, SRRS collects $m$ samples, $X_1, \cdots, X_m$, each of which represents a box $X_t \in \mathcal{G}$. $\hat{p}_\ell$ in eq.(2) is an (asymptotically) unbiased estimator of $p_\ell$.*

$$\hat{p}_\ell = \frac{\sum_{t=1}^{m} g_\ell(X_t)}{\sum_{t'=1}^{m} f(X_{t'})}. \tag{2}$$

### C. Practical challenges

Although we have established the theoretical foundation for SRRS above, in practice, SRRS' actual sampling cost and estimation performance are complicated by resource constraints and parameter selections. We thus explore these practical challenges in this subsection.

**Sampling cost.** With enough physical computing power and network bandwidth, resource constraints of sampling processes such as SRRS mainly stem from the limitations of the LBSNs' query APIs. To avoid unsustainable heavy load on their server infrastructure and to protect themselves from being abused or attacked via computationally costly requests, LBSN providers, including Foursquare, often limit the maximum number of API queries allowed per time unit for each API user (a time constraint), and limit the number of total venues returned per API query (a space constraint). Although the actual numerical values of these constraints may vary from LBSN to LBSN, or even from user to user (e.g., developer vs. normal user) on the same LBSN, it is unlikely that any LBSN would remove these limits completely.

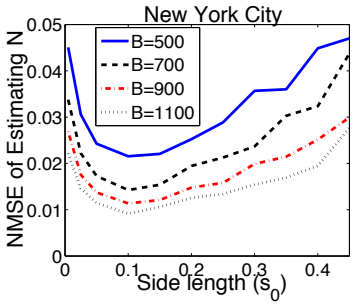



Fig. 1. Impacts of side length $s_0$


Fig. 2. NYC map

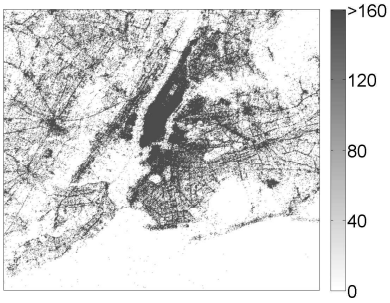



Fig. 3. NYC Venue density extracted from our data

TABLE II
VENUE DENSITY VARIATIONS FOR DIFFERENT $s_0$ AT NYC. THE LATITUDINAL AND LONGITUDINAL LENGTH OF NYC ARE $0.44°$ AND $0.55°$.

| $s_0$ | $0.4°$ | $0.2°$ | $0.1°$ | $0.05°$ | $0.01°$ | $0.005°$ | $0.001°$ | $0.0005°$ |
|---|---|---|---|---|---|---|---|---|
| $n$ | 4 | 9 | 30 | 108 | 2520 | 9968 | 246960 | 985839 |
| $\max f(X)$ | 380176 | 275207 | 168222 | 104153 | 10045 | 3309 | 254 | 164 |
| $E[f(X)]$ | 111401 | 49512 | 14853.7 | 4126.07 | 176.85 | 44.71 | 1.81 | 0.45 |
| $Var[f(X)]$ | $2.43 \times 10^{10}$ | $6.56 \times 10^{9}$ | $9.32 \times 10^{8}$ | $1.22 \times 10^{8}$ | $3.70 \times 10^{5}$ | $2.68 \times 10^{4}$ | 58.99 | 4.88 |

These constraints have direct impacts on the sampling costs. As discussed in Section III, due to the API space constraint–result limit $\bar{b}$ of each API call–obtaining a sample (i.e., all venues of a sampled box) may take different numbers of API queries (or amount of time accordingly), depending on the size and the density of the sample box. Therefore, each sample should not be counted equally as incurring one unit of cost. The actual cost of each sample and the total sampling budget $B$ should use the number of API queries. As a result, choosing the right $s_0$ for SRRS becomes critical. Oversized or undersized $s$ could either incur extra API queries for performing the binary division traversal in dense sample boxes or result in less efficient use of API queries on a large number of mostly empty sample boxes.

**Estimation performance.** With the same sampling budget $B$, a better estimation performance simply means smaller estimation errors. Since choosing $s_0$ impacts the cost of each sample in SRRS, and more samples usually lead to better estimation, $s_0$ selection certainly plays an important role on SRRS estimation performance. To see how $s_0$ affects the sampling accuracy and cost, in Fig. 1, we show the normalized mean square error (NMSE) between our estimation results using SRRS and the "ground truth" in the New York City region, where the "ground truth" was obtained by our exhaustive venue search algorithm. NMSE [30] is defined as

$$NMSE(\hat{N}) = \frac{\sqrt{E[(\hat{N} - N)^2]}}{N}. \quad (3)$$

There are in total $453,070$ venues in New York City. We observe that in SRRS, there exists an optimal initial box side length $s_0 \approx 0.1°$ that leads to the lowest estimation error. This can be explained as follows. Big boxes often lead to large variances in the number of venues inside them (See Table II). When a sample box is big, it may consume a lot of API queries through the binary division traversal to perform the exhaustive venue search. On the other hand, when $s_0$ is too small, a huge number of boxes with no venues are sampled, making SRRS less efficient due to the large number of API queries spent on empty boxes. However, it is in general hard to determine this optimal side length $s_0$ in advance. Even for the optimal side length $s_0$, the estimation error can still be high, due to the high venue density variations across $G$. For instance, in New York City, the downtown area has a much higher venue density than suburb areas, which in turn are denser than remote areas like the ocean or mountains. Fig.2-Fig.3 and Table II illustrate how the venue density varies in NYC.

Given these practical challenges, the estimation performance can greatly benefit from the following two scenarios: 1) each sample box should have roughly the same number of venues to keep the estimation variance small, and 2) the number of venues per box should be close to but not exceed the API return limit to maximize the utilization of each API query. Without any knowledge of the region $G$, it is impossible for SRRS to choose an optimal $s_0$ The above analysis raises the following question: *Is it possible to design a dynamic random region sampling algorithm that gradually builds up the knowledge of the density of the objective region with the cumulative sample information and adaptively calibrates sample boxes to achieve better estimation performance?*

## VI. DYNAMIC RANDOM REGION SAMPLING

In this section, we will present a dynamic random region sampling algorithm that provides a novel and promising solution to address the question posted in the last section. The basic idea behind the dynamic random region sampling is that at every step, a location, denoted by `ll`, is chosen uniformly at random from $G$, and then the size of the sample box around `ll` is determined by two criteria as follows.

• **Box size selection using venue density prediction:** The venue density $d'$ around `ll` is predicted as the weighted

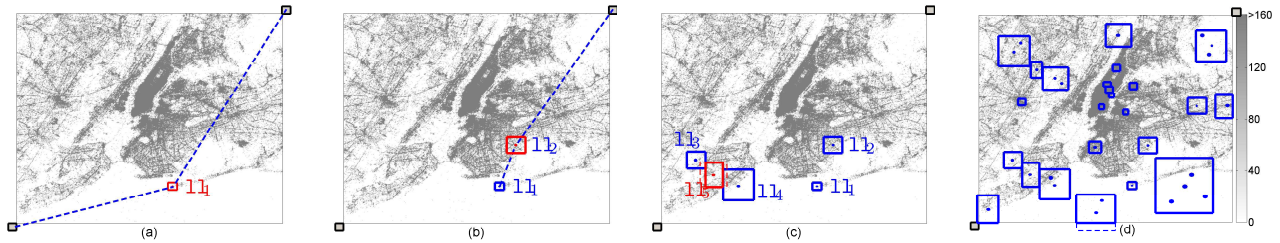



Fig. 4. Illustration of the DRRS algorithm in New York City. (Areas with darker color have higher venue densities as shown on the colorbar. The red point represents the current chosen location, where the red boxes represent the sample boxes generated.) **(a)** The $\texttt{ll}_{\texttt{sw}}$, $\texttt{ll}_{\texttt{ne}}$ as two starting points are assigned with an initial side length $s_0 = 0.02^\circ$ and density $d_0 = 50/0.02^2$. Hence, the predicted density for the first chosen location $\texttt{ll}_1$ will be $d'_1 = (dis^{-1}(\texttt{ll}_1, \texttt{ll}_{\texttt{sw}})d_0 + dis^{-1}(\texttt{ll}_1, \texttt{ll}_{\texttt{ne}})d_0)/(dis^{-1}(\texttt{ll}_1, \texttt{ll}_{\texttt{sw}}) + dis^{-1}(\texttt{ll}_1, \texttt{ll}_{\texttt{ne}})) = d_0$, and the side length will be $s_1 = 0.02^\circ$. As seen from the background color, $\texttt{ll}_1$ locates in a venue sparse area, the actual number of returned venues is 5, indicating its true density as $d_1 = 5/0.02^2$; **(b)** The second location $\texttt{ll}_2$ is closer to $\texttt{ll}_1$ and $\texttt{ll}_{ne}$, with distance $dis(\texttt{ll}_2, \texttt{ll}_1) = 0.08^\circ$ and $dis(\texttt{ll}_2, \texttt{ll}_{\texttt{ne}}) = 0.40^\circ$. Its density is predicted as $d'_2 = (dis^{-1}(\texttt{ll}_2, \texttt{ll}_1)d_1 + dis^{-1}(\texttt{ll}_2, \texttt{ll}_{\texttt{ne}})d_0)/(dis^{-1}(\texttt{ll}_2, \texttt{ll}_1) + dis^{-1}(\texttt{ll}_2, \texttt{ll}_{\texttt{ne}})) = 50/0.04^2$, thus it chooses side length as $s_2 = 0.04^\circ$. The exhaustive search over this box returns 39 venues. **(c)** The fifth location is chosen between two nearby sampled boxes, and with the predicted side length, it overlaps with $\texttt{ll}_3$ and $\texttt{ll}_4$. Hence, it is cut to be non-overlapping with other boxes. **(d)** Over steps, a sequence of sample boxes are formed with more and more accurate predicted densities. Note that some locations falling into already sampled boxes will trigger the corresponding boxes to be re-sampled.

average of the venue densities of its closest sampled boxes. Then, the side length $s$ of a new box centered at $\texttt{ll}$ is computed as $\sqrt{\bar{b}/d'}$, to keep the expected number of venues in this new box close to the API return limit $\bar{b}$. (Section VI-A.)

• **Non-overlapping boxes:** Check whether the new box obtained above collides with any already sampled box, and cut the new box if necessary to keep it containing $\texttt{ll}$ and non-overlapping with those sampled boxes. (Section VI-B.)

Using the above two criteria, a non-overlapping new box $G_{11}$ is determined based on the best knowledge to have an expected total number of venues close to the API return limit. Then, an exhaustive search is performed on this box, and the area it covers is considered as a sampled box. Later if a random location is chosen that falls into this box, it is considered that this box is sampled again, and no actual API queries are needed. Note that keeping the sampled boxes to be non-overlapping ensures each box has an invariant probability to be sampled again once the box has been established in one sampling process. This probability is proportional to the size of its area, and it is an important quantity in designing unbiased estimators for the objective region. Until running out of the API budget $B$, $m$ samples, $X_1, \cdots, X_m$ are drawn from $n$ non-overlapped boxes $G_1, \cdots, G_n$, with $m \geq n$. If the budget runs out while exhaustively searching the last box, that box will be ignored.

### A. Dynamic box size selection

We observe that the venue densities are changing gradually across a geographic region, which means that the closer two nearby regions are to each other, their venue densities are more similar. Hence the venue density $d'$ around $\texttt{ll}$ is predicted based on the venue densities of its closest sampled boxes, where the distance is measured by the Euclidean distance in latitude and longitude.

Given an unknown objective region $G$, we do not have any knowledge of its venue density. To bootstrap a sampling process, initially the south-west ($\texttt{ll}_{\texttt{sw}}$) and north-east ($\texttt{ll}_{\texttt{ne}}$) corners of $G$ are assigned with a density $d_0 = \bar{b}/s_0^2$, namely, a square area of side length $s_0$ with exactly $\bar{b}$ venues. Note that the selection of $s_0$ certainly has some impacts on the estimation performance, but unlike affecting all boxes in SRRS, $s_0$ here serves as an initial estimation on two boxes and its importance fades gradually with more sampled boxes. These two locations as two starting points are used to start the venue density prediction. When the first location $\texttt{ll}_1$ is chosen uniformly at random from $G$, it will be assigned with a side length of $s_1 = \sqrt{\bar{b}/d'_1}$ to form a potential box $G_1$ centered at $\texttt{ll}_1$, where the predicted venue density $d'_1$ is the weighted average of densities $d_{sw}$ and $d_{ne}$, thus $d'_1 = d_0$. Then, the returned exhaustive venue search result on $G_1$ establishes the first true venue density in $G$ as $d_1 = |V^{in}(G_1)|/a(G_1)$, which in turn affects the size selection of following boxes. As the number of sample boxes increases, more true venue densities are collected, gradually improving the accuracy of venue density predictions.

### B. Non-overlapping sample boxes

Every box $G_i$ is always chosen to be non-overlapping with already sampled boxes, which enables us to iteratively build up a partition of the objective region $G$. If a newly selected location falls into a sampled box, it is considered that this sampled box is sampled again. In this way, it keeps locations in each sampled box having an invariant probability to be sampled through the entire sampling process, where the probability $Pr(G_i)$ of each $G_i$ being sampled is $Pr(G_i) = a(G_i)/a(G)$, where $a : \mathcal{G} \to \mathbb{R}$ represents the area of the box in term of latitude and longitude, i.e., the product of the latitudinal and longitudinal side lengths.

### C. Dynamic random region sampling

Given $G$, each run of DRRS algorithm corresponds to a partition of $G$, i.e., $\mathcal{G} = \{G_1, \cdots, G_{n^*}\}$, with $G = G_1 \vee \cdots \vee G_{n^*}$ and $n^*$ as the total number of subgraphs. This partition $\mathcal{G}$ is actually built on-the-fly, which thus forms the population space of each sampling process. The population space $\mathcal{G}$ is

**Algorithm 2** Dynamic random region sampling algorithm

1: **INPUT:** API budget $B$, initial side length $s_0$, API return limit $\bar{b}$, $G$ bounded by $\texttt{ll}_{sw} = (\texttt{lat}_\texttt{s}, \texttt{lng}_\texttt{w})$ and $\texttt{ll}_\texttt{ne} = (\texttt{lat}_\texttt{n}, \texttt{lng}_\texttt{e})$
2: **OUTPUT:** Sample list $\mathcal{X} = [X_1, \cdots, X_m]$, where $X_t = V_t^{in}$.
3: $\mathcal{G}_0 = \emptyset$; $c = 0$; $m = 0$; $\mathcal{X} = \emptyset$
4: Map $\mathcal{M} = \{(\texttt{ll}_\texttt{sw}, d_{sw}), (\texttt{ll}_\texttt{ne}, d_{ne})\}$, $d_{sw} = d_{ne} = \bar{b}/s_0^2$;
5: **while** $c < B$ **do**
6: Choose $\texttt{ll} = (\texttt{lat}, \texttt{lng})$ uniformly at random from $G$;
7: **if** ll is in any box $G_j \in \mathcal{G}_0$ **then**
8: $m = m + 1$; $\mathcal{X} = [\mathcal{X}, X_j]$, ($X_j$: InSearch return on $G_j$);
9: Continue;
10: Choose $\texttt{ll}_1$ and $\texttt{ll}_2$ in $\mathcal{M}$ with smallest distance to ll;
11: Predict the density around ll as $d' = \sum_{i=1}^{2} dis^{-1}(\texttt{ll}, \texttt{ll}_i) d_i / \sum_{i=1}^{2} dis^{-1}(\texttt{ll}, \texttt{ll}_i)$;
12: $s = \sqrt{\bar{b}/d'}$;
13: Create $G_{\texttt{ll}}$ with center $(\texttt{lat}, \texttt{lng})$ and side length $s$;
14: **if** $G_{\texttt{ll}}$ overlaps with any box in $\mathcal{G}_0$ **then**
15: Cut $G_{\texttt{ll}}$ to be non-overlapping;
16: Directly query $G_{\texttt{ll}}$ and obtain $V_0^{in}(G_{\texttt{ll}})$ and $V_0(G_{\texttt{ll}})$;
17: $[V_m^{in}, V_m, B_m] = \texttt{InSearch}(G_{\texttt{ll}}, V_0^{in}(G_{\texttt{ll}}), V_0(G_{\texttt{ll}}))$;
18: $X_m = V_m^{in}$; $d = |V_m^{in}|/a(G_{\texttt{ll}})$;
19: $\mathcal{G}_0 = \mathcal{G}_0 \cup \{G_{\texttt{ll}}\}$;$c = c + B_m + 1$;$m = m + 1$;$\mathcal{X} = [\mathcal{X}, X_m]$;
20: $\mathcal{M} = \mathcal{M} \cup \{(\texttt{ll}, d)\}$;

iteratively established during the sampling process and may vary for different sampling processes, which depends on the sequence of the randomly selected locations, as well as the initial side length $s_0$. Note that given a population space, i.e., a partition $\mathcal{G}$ of $G$, the corresponding sampling process is performed equivalently to running the sampling process on $G$ with predetermined partition $\mathcal{G}$.

At the beginning, let the partition set $\mathcal{G}^{(0)} = \emptyset$. When the first box $G_1$ is formed at step 1, $\mathcal{G}^{(1)} = \{G_1\}$. At every step $t$, $\mathcal{G}^{(t)} = \mathcal{G}^{(t-1)} \cup \{G_t\}$. In a particular sampling process, with a limited budget $B$, a total of $m$ sampled boxes $X_1, \cdots, X_m$ (which in general may have duplications) are drawn representing $n \leq m$ unique non-overlapping boxes $G_1, \cdots, G_n$, which together form a box set $\mathcal{G}^{(m)} = \{G_1, \cdots, G_n\}$, with $\mathcal{G}^{(m)} \subseteq \mathcal{G}$. When it goes through a sufficient number of steps, namely, having a large enough budget $B$, $\lim_{m\to\infty} \mathcal{G}^{(m)} = \mathcal{G}$. Note that the non-overlapping box selection criterion ensures that each box once formed has an invariant probability being sampled again during that sampling process.

The detailed DRRS algorithm is summarized in Algorithm 2, and Fig. 4 visualizes the sampling process. DRRS takes the *inputs* as the total API budget $B$, an initial side length $s_0$, the API return limit $\bar{b}$, and the objective region $G$ bounded by $\texttt{ll}_\texttt{sw} = (\texttt{lat}_\texttt{s}, \texttt{lng}_\texttt{w})$ and $\texttt{ll}_\texttt{ne} = (\texttt{lat}_\texttt{n}, \texttt{lng}_\texttt{e})$. It outputs a list of returned venues obtained from the exhaustive venue search on sampled boxes.

Besides the sampled venues, DRRS maintains a set of already sampled boxes, denoted by $\mathcal{G}_0$, and their densities, denoted by $\mathcal{M}$. At every step, a location ll in $G$ is chosen uniformly at random (line 6). Then, if ll is within one of the already sampled boxes $G_j \in \mathcal{G}_0$, $G_j$ is considered being sampled again, and no API budget is spent (line 7–9). If not, a new box $G_{\texttt{ll}}$ is created based on the two criteria (line 10-15) in Section VI-A and Section VI-B, and an exhaustive search is performed on $G_{\texttt{ll}}$ (line 16-17). Theorem 4 shows that DRRS is an importance sampling algorithm, where the probability of every individual venue being selected in the population is proportional to the box size it belongs to.

**Theorem 4 (Importance Sampling).** *For a DSSR sampling process, the probability $Pr(v)$ of each venue $v \in G$ being sampled is $Pr(v) = a(G_v)/a(G)$, where $v \in G_v \in \mathcal{G}$.*

*Proof:* In DRRS, each location ll has an equal probability of being chosen, thus each box $G_v$ has $a(G_v)/a(G)$ probability to be selected. Using the law of total probability, we prove that the probability of a venue $v$ being sampled using DRRS is

$$Pr(v) = \sum_{G_v \in \mathcal{G}} Pr(G_v) Pr(v|G_v) = \frac{a(G_v)}{a(G)}, \tag{4}$$

where the conditional probability of $v$ being sampled given the box $G_v$ is chosen, is 1 is $v \in G_i$, and 0 otherwise. ■

### D. Estimators

We use the venues sampled with DRRS to estimate the characteristics of objective region $G$, such as the total number of venues, the check-in distribution and tips distribution.

*1) Total number of venues:* Theorem 5 below presents the estimator $\hat{N}$ to estimate $N$.

**Theorem 5 (Estimator of the total number of venues).** *Using DRRS with budget $B$, we obtain $m$ sampled boxes $X_1, \cdots, X_m$, with each $X_t \in \mathcal{G}$. Then, $\hat{N}$ in eq.(5) is an (asymptotically) unbiased estimator of $N$.*

$$\hat{N} = \frac{a(G)}{m} \sum_{t=1}^{m} \frac{f(X_t)}{a(X_t)}. \tag{5}$$

*Proof:* Define the random variable of the venue density of $X_t$ as $d(X_t) = f(X_t)/a(X_t)$. Each $d(X_t)$ independently follows the same distribution. Applying the linearity of the expectation and the ratio form of the law of large numbers (Theorem 17.2.1 on p.426 in [26]) gives

$$\lim_{m\to\infty} \hat{N} \xrightarrow{a.s.} a(G) \frac{\sum_{X\in\mathcal{G}} \frac{f(X)}{a(X)} Pr(X)}{\sum_{X\in\mathcal{G}} Pr(X)} = \sum_{X\in\mathcal{G}} f(X) = N,$$

Hence, $E[\hat{N}] = N$ holds for all values of $m > 0$. ■

*2) Label distribution:* Theorem 6 introduces the unbiased estimator of the label distribution in terms of the DRRS samples, which can be proven by applying the ratio form of the law of large numbers (Theorem 17.2.1 on p.426 in [26]).

**Theorem 6 (Estimator of label distribution).** *Given a budget $B$, DRRS collects $m$ samples, $X_1, \cdots, X_m$, each of which represents a sample box $X_t \in \mathcal{G}$. $\hat{p}_\ell$ in eq.(6) is an unbiased estimator of $p_\ell$.*

$$\hat{p}_\ell = \frac{\sum_{t=1}^{m} g_\ell(X_t)/a(X_t)}{\sum_{t'=1}^{m} f(X_{t'})/a(X_{t'})}. \tag{6}$$

*3) Convergence analysis:* To further understand the estimators, namely, to determine the stopping condition of the sampling process with an ensured estimation accuracy, in this section, we derive the variance and confidence interval of our proposed estimators, including $\hat{N}$ and $\hat{p}_\ell$. We start with the introduction of the confidence interval inequality in Lemma 1 and the ratio form confidence interval in Lemma 2. We then use these lemmas to derive the sufficient conditions to achieve the desired confidence interval of our proposed estimators.

**Lemma 1 (Confidence interval inequality).** *Given a random variable $Y$, $Var[Y] \leq \alpha\epsilon^2 E^2[Y]$ is the sufficient condition to ensure the following confidence interval*

$$Pr(E[Y](1-\epsilon) \leq Y \leq E[Y](1+\epsilon)) \geq 1-\alpha, \quad (7)$$

*where $0 < \alpha \leq 1$ and $0 < \epsilon \leq 1$.*

*Proof:* (Sketch) Comparing the Chebyshev's inequality for $Y$ and the confidence interval (eq.(7)) completes the proof. ■

Lemma 2 below presents the sufficient condition to ensure a confidence interval of a ratio form random variable.

**Lemma 2 (Confidence interval of a ratio form random variable).** *Given two random variables $Y$ and $Z$ ($Z \neq 0$ and $E[Z] \neq 0$), if $E[Y/Z] = E[Y]/E[Z]$ holds[1], $Var[Y] \leq \alpha\epsilon^2 E^2[Y]/18$ and $Var[Z] \leq \alpha\epsilon^2 E^2[Z]/18$ together is the sufficient condition to ensure the following confidence interval*

$$Pr\big(E[\frac{Y}{Z}](1-\epsilon) \leq \frac{Y}{Z} \leq E[\frac{Y}{Z}](1+\epsilon)\big) \geq 1-\alpha, \quad (8)$$

*where $0 < \alpha \leq 1$ and $0 < \epsilon \leq 1$.*

*Proof:* By applying Lemma 1, the sufficient conditions to guarantee an $\epsilon/3$ approximation to $Y$'s and $Z$'s expected values with probability at least $1-\alpha/2$ are

$$Var[Y] \leq \alpha\epsilon^2 E^2[Y]/18, \qquad Var[Z] \leq \alpha\epsilon^2 E^2[Z]/18. \quad (9)$$

Since the probability of random variable $Y$ or $Z$ for deviating from their expected values is at most $\alpha/2$, the probability of one or both of them deviating is at most $\alpha$ (the union bound). This gives us that with probability at least $1-\alpha/2$ both variables are $1-\epsilon/3$ close to their respective expectations. Then using the facts that $\frac{Y/Z}{E[Y/Z]} \leq (1+\epsilon/3)/(1-\epsilon/3) \leq 1+\epsilon$ and $\frac{Y/Z}{E[Y/Z]} \geq (1-\epsilon/3)/(1+\epsilon/3) \geq 1-\epsilon$, we prove that $0 \leq \epsilon \leq 1$. ■

**Convergence analysis of the total number of venues.** Theorem 7 below presents the confidence interval of the estimator $\hat{N}$ for DRRS algorithm.

**Theorem 7.** *The estimator $\hat{N}$ (eq.(5)) guarantees that for any $0 < \epsilon \leq 1$ and $0 < \alpha \leq 1$,*

$$Pr(N(1-\epsilon) \leq \hat{N} \leq N(1+\epsilon)) \geq 1-\alpha, \quad (10)$$

*when $m \geq m_o \in \mathcal{O}(\frac{a^2(G)\sigma_d^2}{\alpha\epsilon^2 N^2})$. $\sigma_d^2$ is the variance of venue density $d(X_t) = f(X_t)/a(X_t)$, and is an constant for the population space—the partition $\mathcal{G}$, governed by a sampling process of DRRS.*

[1]Note that in general this does not necessarily hold for a ratio form random variable.

*Proof:* (Sketch) Computing the variance and expectation of $\hat{N}$ and applying Lemma 1 complete the proof. ■

**Convergence analysis of the label distribution.** Theorem 8 discusses the confidence interval of the estimator $\hat{p}_\ell$ when using DRRS.

**Theorem 8.** *The estimator $\hat{p}_\ell$ (eq.(6)) guarantees that for any $0 < \epsilon \leq 1$ and $0 < \alpha \leq 1$,*

$$Pr(p_\ell(1-\epsilon) \leq \hat{p}_\ell \leq p_\ell(1+\epsilon)) \geq 1-\alpha, \quad (11)$$

*when $m \geq m_\ell \in \mathcal{O}\{\frac{a^2(G)\sigma_d^2}{\alpha\epsilon^2 N^2}, \frac{a^2(G)\sigma_{d,\ell}^2}{\alpha\epsilon^2 N_\ell^2}\}$. $\sigma_{d,\ell}^2$ is the variance of venue density $d_\ell(X_t) = g_\ell(X_t)/a(X_t)$ for venues with label $\ell$. It is an constant for the population space—the partition $\mathcal{G}$, governed by a sampling process of DRRS.*

*Proof:* (Sketch) The label distribution estimator is a ratio form random variable $\hat{p}_\ell = D_{m,\ell}/D_m$, where $D_m = \sum_{t=1}^m d(X_t)$ and $D_{m,\ell} = \sum_{t=1}^m g_\ell(X_t)/a(X_t)$ denote the random variables of the total sum of the venue densities and the density sum of venues with label $\ell$ from the collected regions. Computing the expectations and variances for $D_{m,\ell}$ and $D_m$ and applying Lemma 2 complete the proof. ■

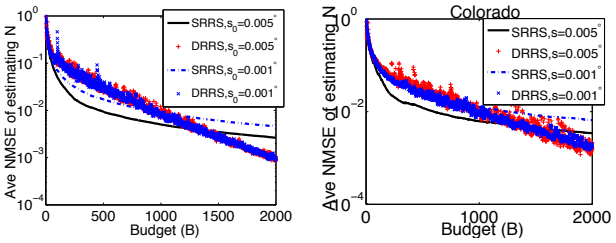
## VII. Experimental results

For the evaluation, we compare our DRRS with a simple static random region sampling algorithm (SRRS). We apply both DRRS and SRRS on 12 geographic regions, with the initial side length $s_0$ ranging from the smallest resolution $4.8 \times 10^{-5^\circ}$ to $0.9^\circ$, i.e., the largest side length that the search API allowed. We take the venue set obtained by exhaustive venue search algorithm as the "ground truth" and compare the estimation accuracy of the total number of venues, and their check-in and tip distributions. Note that the results obtained for all 12 geographic regions are qualitatively similar, and we only present the results of Switzerland and Colorado here for brevity. Moreover, we ran each sampling algorithm 600 times and took the average to remove the impacts of the randomness. We ran our experiments in parallel using 40 machines with quad-core CPUs.

### A. Total number of venues

Fig. 5 shows the errors (in NMSE) of estimating the total number of venues in Switzerland and Colorado. We make the following two observations. First of all, we observe that at the early stage of sampling, DRRS does not perform as well as SRRS, where after a certain number of queries, the cumulated venue density information increases the estimation accuracy of DRRS rapidly and generates much better performance than SRRS. This happens because for DRRS, no pre-knowledge is initially available, and the early estimation errors depend heavily on the selections of the first few locations. As more samples are obtained, DRRS can predict more precise venue densities of the following selected locations, thus significantly

saving sampling budgets and improving the estimation accuracy. Moreover, we found that the estimation errors of SRRS are more sensitive than that of DRRS. For example, when changing the initial side length $s$ for Switzerland from $0.005°$ to $0.001°$, NMSE of SRRS increases from $0.0015$ to $0.0040$, where NMSEs of DRRS are almost the same. This happens because after a few samples, DRRS in fact does not rely on the initial side length any more, where unlike SRRS, a slight change to the side length $s$ may have a significant impact on the accuracy of the estimators. Hence, the above observations also indicate that it is a good idea to combine SRRS and DRRS to take the advantages of both of them by performing SRRS at the early stage and switch to DRRS after a sufficient number of samples are cumulated. We leave this for future work.




(a) Switzerland (b) Colorado

Fig. 5. NMSE of estimating total number of venues in Switzerland and Colorado

### B. Venue label distributions

Now, we are in a position to analyze the estimation results of venue label distributions, such as the distribution of the number of check-ins, the number of users and the number of tips. In this section, we focus on the comparisons and analysis of the overall estimated distributions given a certain initial side length and a sampling budget $B$. We use $B = 2000$ and $s = 0.005°$ for the following results. In Fig 6, we present the results of estimating the distributions of the number of check-ins, users, and tips for Switzerland. Fig 6(a)–Fig 6(c) present comparisons between the "ground truth" distribution and estimated distributions by SRRS and DRRS, respectively. We can see that the estimated label distributions by DRRS match the "ground truth" distributions much better than that by SRRS. Using NMSE, Fig 6(d)–Fig 6(f) quantitatively measure the estimation errors by SRRS and DRRS. An interesting phenomenon we observed is that for smaller label values, SRRS and DRRS exhibit similar estimation accuracies, whereas for higher label values, SRRS performs much worse than DRRS, i.e., NMSE by DRRS on high label venues is smaller than that of SRRS by four orders of magnitude. Moreover, apparently SRRS estimates lower label distributions with higher accuracy than that of higher label distributions. This happens because there are far more lower label venues than higher label venues, and SRRS samples each venue with equal probability, thus lower label venues have higher chances of being sampled. On the other hand, DRRS estimates higher label distributions with better overall accuracies than that of lower label distributions. This happens because DRRS judiciously adjusts the box sizes, which saves many sampling budgets from getting empty returns or querying a region with too many venues, and thus collect far more sampled venues than SRRS. In our evaluation, SRRS and DRRS draw $1995$ and $253$ sampled boxes per run on average, with $772$ and $38443$ sampled venues, respectively. These observations suggest that to estimate the low label distribution, DRRS and SRRS perform equally well, while to get better estimation of high label distribution, DRRS is preferred. Fig 7(a)–Fig 7(f) show comparison results for the distribution estimation of the total number of checkins, users, and tips in Colorado, which echo the observations obtained from Switzerland dataset.

## VIII. Discussion

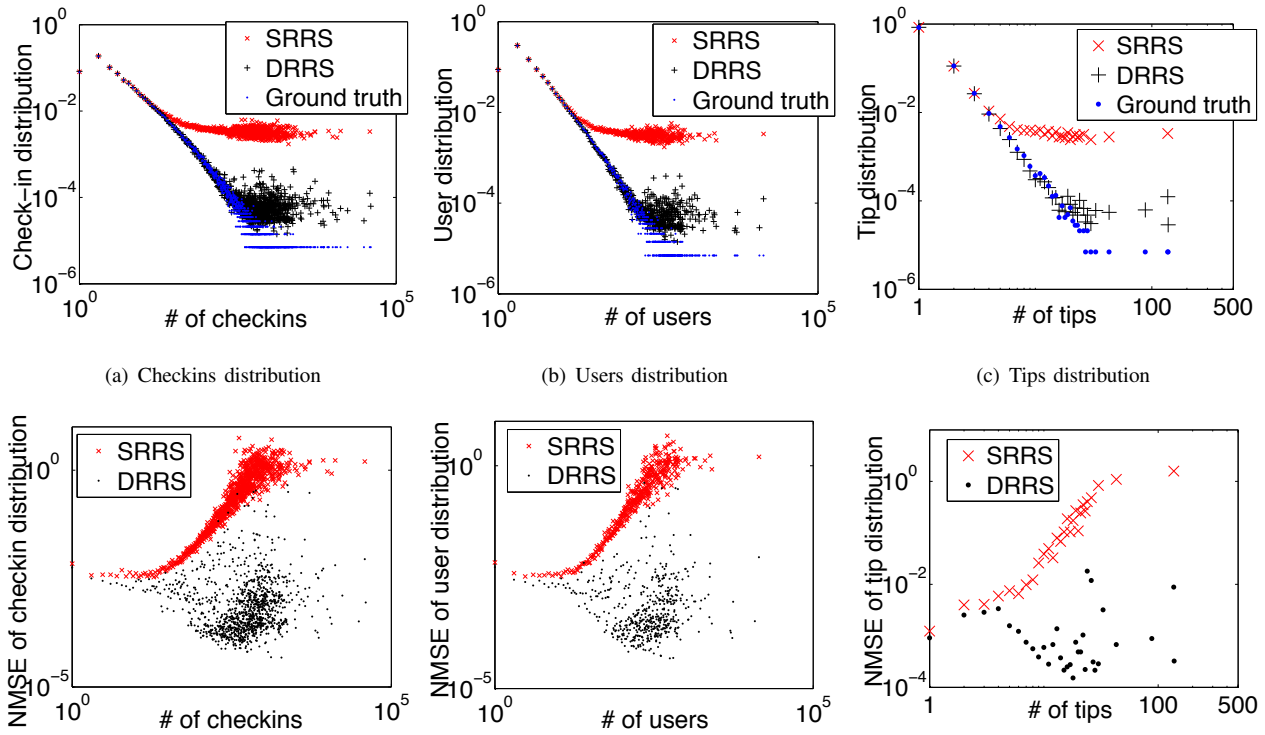
The static and dynamic random region sampling algorithms as building blocks enable us to retrieve representative samples from LBSNs. They can be easily generalized to realize more sampling purposes as follows.

Firstly, our sampling algorithms can be generalized to sample and estimate the dynamics of LBSNs, namely, how the statistics of LBSNs evolve over time. The sampling algorithms can be performed periodically to extract snapshots of the LBSNs, and the samples obtained earlier can be utilized for the following snapshot sampling. For example, SRRS can partition the graph with respect to the venue densities inferred from the earlier samples, while DRRS can use the previous samples as initial starting points instead of the two corner points. Moreover, SRRS can be used to sample and estimate statistics of active users, by collecting all users who left tips to or checked in to locations in a sampled box. The probability of each user being sampled is proportional to the user's degree, i.e., the number of tips or check-ins he left. Thus, the unbiased estimators of the total number of active users or the user's label distributions can be obtained by quantitatively removing such bias.

Secondly, all LBSNs, e.g., Foursquare, Yelp, and Google+ Local, provide similar regional venue search APIs (as shown in Table III), with either a square or circle bounding box, a return limit on the number of venues returned per query, and a limit on the number of queries allowed per day. In this paper we develop and utilize an exhaustive search mechanism for Foursquare to design our sampling algorithms. Exhaustive search mechanisms for other LBSNs may have to be designed in a different way depending on their API return principles. Our sampling algorithms, however, are independent of a specific LBSN.

TABLE III
Regional Venue Search APIs from leading LBSNs.

| LBSNs | Bounding box | Specified by | Return limit | Max query rate (a day) |
|---|---|---|---|---|
| Yelp API 2.0 | Rectangle | sw & ne locations | 20 | 10,000 |
| Yelp API 1.0 | Circle | location & radius | 20 | 10,000 |
| Foursquare | Rectangle | sw & ne locations | 50 | 12,000 |
| Google+ local | Circle | location & radius | 60 | 100,000 |




(a) Checkins distribution (b) Users distribution (c) Tips distribution

(d) Checkin NMSE (e) Users NMSE (f) Tips NMSE

Fig. 6. Label distribution estimation (Switzerland)

Last but not the least, the location service providers, e.g., Foursquare, Yelp, and Google+ Local, can also benefit from our sampling and estimation algorithm to enable efficient online statistic estimation for regions. Currently, location services expedite the range queries to retrieve a complete venues set, by indexing the GeoSocial data with data structures such as B-tree, R-tree, or $R^+$-Tree, etc [14], [17], [33]. However, when the geographic region being estimated is large, e.g., a city or a state, the range query will be too costly to be implemented online. Instead of completely examining the region, our algorithms allow the location service providers to estimate the real-time regional statistics with controllable execution time and estimation accuracy.

## IX. Conclusion

In this paper, we study how to sample and estimate characteristics of location based social networks. Taking Foursquare as an example, we first study the functionality of the venue search API, and develop an exhaustive venue search algorithm, that can extract a complete set of venues for a restricted geographic region. The exhaustive search algorithm as a building block allows us to design random region sampling algorithms and unbiased estimators for many venue properties, such as total number of venues and the distributions of the number of check-ins, tips and users.

We run our random region sampling algorithms on 12 regions all over the world, and compare their estimation results with the "ground truth" obtained by the exhaustive search algorithm. The comparison results show that SRRS algorithm performs well for a small sample budget; it is however outperformed by DRRS algorithm when more budget is available. In LBSNs, most venues have few check-ins and users and only few venues have many check-ins and users. Since SRRS samples each venue with equal probability, the popular venues are very unlikely to be sampled by SRRS. Therefore, SRRS performs much worse than DRRS when estimating the distributions of very popular venues.

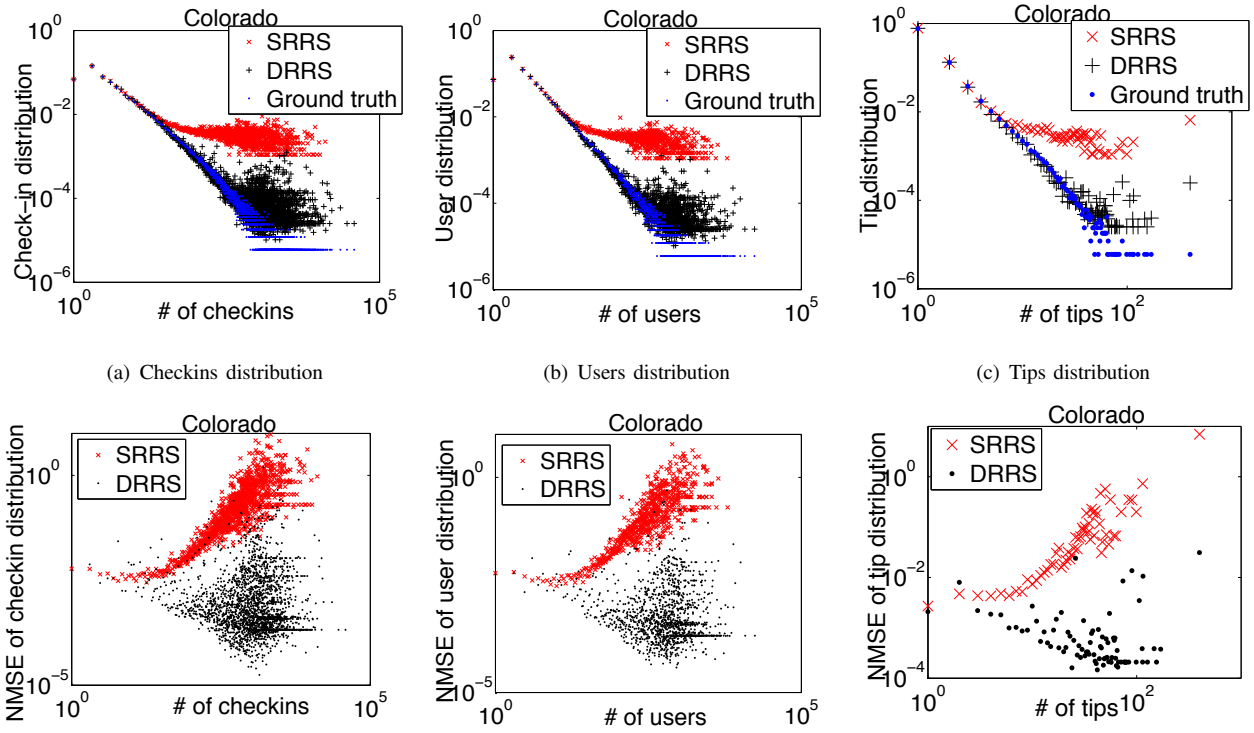
Our proposed region sampling algorithms enable studying large-scale location based social networks via sampled datasets [22], [23]. In the future, we are interested in improving users' experience in LBSN, by developing efficient algorithms for location recommendation, location popularity prediction.

## X. Acknowledgement

We would like to thank the anonymous reviewers for their helpful feedbacks on this paper. Yanhua Li was supported in part by National Grant Fundamental Research (973 Program) of China under Grant 2014CB340304 and US NSF grant 0831734. Ting Zhu was supported in part by US NSF grant CNS-1217791.

## References

[1] Facebook places. *http://touch.facebook.com/*.
[2] Foursquare. *https://foursquare.com/*.
[3] Foursquare API. *https://developer.foursquare.com/*.
[4] Foursquare house rules. *https://foursquare.com/info/houserules*.
[5] Foursquare on wikipedia. *http://en.wikipedia.org/wiki/Foursquare*.
[6] Google Geo-coding API. *https://developers.google.com/maps/documentation/geocoding/*.
[7] Google+ Local. *www.google.com/+/learnmore/local*.




(a) Checkins distribution (b) Users distribution (c) Tips distribution

(d) Checkin NMSE (e) Users NMSE (f) Tips NMSE

Fig. 7. Label distribution estimation (Colorado)


[8] A million check-ins isn't cool. you know what's cool? *http://blog.foursquare.com/2011/09/20/billion/*.

[9] Snowball sampling. *http://en.wikipedia.org/wiki/Snowball_sampling*.

[10] K. Avrachenkov, B. F. Ribeiro, and D. F. Towsley. Improving random walk estimation accuracy with uniform restarts. In *WAW*, 2010.

[11] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *SIGSPATIAL GIS*. ACM, 2012.

[12] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In *WWW*, 2006.

[13] E. Cho, S. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *KDD*, 2011.

[14] D. Comer. Ubiquitous b-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137, 1979.

[15] M. Gjoka, C. T. Butts, M. Kurant, and A. Markopoulou. Multigraph sampling of online social networks. *JSAC*, 2011.

[16] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of OSNs. In *INFOCOM*, 2010.

[17] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*. ACM, 1984.

[18] L. Katzir, E. Liberty, and O. Somekh. Estimating sizes of social networks via biased sampling. In *WWW*, 2011.

[19] M. Kurant, M. Gjoka, C. T. Butts, and A. Markopoulou. Walking on a Graph with a Magnifying Glass. In *ACM SIGMETRICS*, 2011.

[20] M. Kurant, A. Markopoulou, and P. Thiran. On the Bias of Breadth First Search (BFS) and of Other Graph Sampling Techniques. In *ITC*, Amsterdam, 2010.

[21] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W.-Y. Ma. Mining user similarity based on location history. In *SIGSPATIAL GIS*, page 34. ACM, 2008.

[22] Y. Li, M. Steiner, L. Wang, Z.-L. Zhang, and J. Bao. Dissecting foursquare venue popularity via random region sampling. In *CoNEXT student workshop*, pages 21–22, 2012.

[23] Y. Li, M. Steiner, L. Wang, Z.-L. Zhang, and J. Bao. Exploring venue popularity in foursquare. In *NetSciCom*, 2013.

[24] H. Liu, L.-Y. Wei, Y. Zheng, M. Schneider, and W.-C. Peng. Route discovery from mining uncertain trajectories. In *ICDM*, 2011.

[25] S. Lohr. *Sampling: design and analysis*. Thomson, 2009.

[26] S. Meyn and R. L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009.

[27] A. Mohaisen, P. Luo, Y. Li, Y. Kim, and Z.-L. Zhang. Measuring bias in the mixing time of social graphs due to graph sampling. In *MILCOM*, pages 1–6, 2012.

[28] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. An empirical study of geographic user activity patterns in foursquare. *ICWSM'11*.

[29] B. Ribeiro, P. Wang, F. Murai, and D. Towsley. Sampling directed graphs with random walks. In *Proc. IEEE INFOCOM*, 2012.

[30] B. F. Ribeiro and D. F. Towsley. Estimating and sampling graphs with multidimensional random walks. In *IMC*, 2010.

[31] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo. Socio-spatial properties of online location-based social networks. *ICWSM'11*.

[32] S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. In *KDD*, 2011.

[33] T. SELLIS. The R^+ tree: A dynamic index for multi-dimensional objects. In *VLDB*, 1987.

[34] L.-Y. Wei, Y. Zheng, and W.-C. Peng. Constructing popular routes from uncertain trajectories. In *KDD*, 2012.

[35] X. Xiao, Y. Zheng, Q. Luo, and X. Xie. Inferring social ties between users with human location history. *Journal of Ambient Intelligence and Humanized Computing*, 2012.

[36] W. Xu, C.-Y. Chow, and J.-D. Zhang. CALBA: Capacity-Aware Location-Based Advertising in Temporary Social Networks. In *SIGSPATIAL GIS*. ACM, 2013.

[37] J.-D. Zhang and C.-Y. Chow. iGSLR: Personalized Geo-Social Location Recommendation - A Kernel Density Estimation Approach. In *SIGSPATIAL GIS*. ACM, 2013.

[38] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In *WWW*. ACM, 2010.

[39] M. Zhong, K. Shen, and J. Seiferas. The convergence-guaranteed random walk and its applications in peer-to-peer networks. *IEEE Transactions on Computers*, 57(5):619–633, 2008.

[40] J. Zhou, Y. Li, V. K. Adhikari, and Z.-L. Zhang. Counting youtube videos via random prefix sampling. In *IMC*, 2011.