Measuring Cross-Device Online Audiences

Jim Koehler, Evgeny Skvortsov, Sheng Ma, Song Liu

Google Inc.

May 10, 2016

Abstract

We extend the work of Koehler, Skvortsov, and Vos (2013) to measure cross-device online audiences. The method performs demographic corrections in the usual way device-by-device. A new method that converts cross-device cookie counts to user counts is introduced. We provide practical recipes for fitting this transformation function and then demonstrate its use using online panel data from Japan.

1 Introduction

Koehler, Skvortsov, and Vos (2013) [13] (KSV) presents a method for measuring reach and frequency of online ad campaigns by audience attributes for one device (or cookie) type. The method combines ad server logs, publisher provided user data (PPD), census data, and a representative panel to produce corrected cookie and impression counts by these audience attributes. The method corrects for cookie issues such as deletion and sharing, and for PPD issues such as non-representativeness and poor quality of demographic labels. It also proposes a model that converts cookie counts to user counts. However, the method falls short in today's world of multiple device types such as desktop, smartphone, and tablet.

This paper extends the existing method to scenarios where information is available for multiple cookie/device types in the ad server logs. It allows for measurement reports broken out by device types associated with the different cookie types. For example, there could be a browser-cookie with device information on desktop, smartphone, and tablet and another app-cookie broken out by smartphone and tablet. A natural extension of the demographic correction follows from the use of cookie/device specific correction models. That is, for each cookie/device combo, a correction model is developed as specified in KSV.

We propose a new formulation for converting multiple cookie counts to people counts. We introduce the concept of an Activity Distribution Function (ADF), which describes the probability of a person generating cookies of each type. We theoretically relate ADFs to matching cross-device reach functions. Furthermore, we show that ADFs can be approximated by a mixture of Dirac delta functions [4], and estimated empirically using panel data.

In Section 2 we review the single device measurement method, including the demographic correction model and the cookie-to-user mapping. Section 3 extends the demographic correction model to

multiple cookie types, while Section 4 introduces the new cross-device cookie-to-user mapping along with recommendations for fitting these new models. The method is demonstrated, using a Japan online panel, in Section 5.

2 Review of Single-Device Measurement Method

KSV develops a method to measure GRPs that allows for reach and frequency estimates for online audiences to be broken down by audience attributes (e.g., age and gender). However, their approach only considers a single cookie or device type and therefore does not extend to multiple device types. This paper extends this methodology to provide these device type breakouts. However, before showing extentions, we provide an overview of the single-device measurement method.

The method of KSV uses a combination of data from several different sources to compute audience reach metrics: US census data, ad server logs from the ad serving network, publisher-provided self-reported demographic data, and a representative online panel. The number of people exposed to a campaign is inferred from the number of unique cookies exposed to these campaigns. For a subset of these cookies, demographic information is available from publisher provided data (PPD). These demographic labels may be incorrect for some of the cookies, and the cookies with labels may not be representative of all cookies. Demographic correction models adjust for possible inaccurate and biased labels using panel data. Additionally, a user is typically represented by multiple cookies, some of which may be shared with other users on the same device. A method is provided to infer the number of users behind a given number of cookies. These models are trained and evaluated using an online calibration panel for which the true demo/ppd labels and cookie-to-user relationships are known.

2.1 Data Sources

The main data source for this method is ad server logs, recording the impressions served to an associated cookie. Ad server logs provide real-time data broken down by site. Cookies present significant technical challenges:

- a cookie does not identify a person, but a combination of a user account, a computer, and a web browser.
- not all cookies have demographic information attached to them, and when there is demographic information available it may be of questionable quality and biased. In order to obtain the demographic composition of an audience, at least a subset of the cookies for that audience need to have an age and gender label.
- cookie deletion (or cookie churn) can also lead to inaccuracies in audience measurement, such as the overstatement of reach and understatement of frequency.
- the quality of declared demographic information relies on the truthfulness of users and also the extent to which cookies are shared between multiple users.

A probability-recruited online panel provides reliable data to calibrate and validate the demographic correction and cookie-to-user conversion models. The panel plays a key role in adjusting for demographic bias and cookie-sharing effects in PPD, in inferring models to accurately estimate the number of users behind aggregated cookie counts, and in evaluating the accuracy of the method.

The panel should be aligned to census level benchmarks on key demographic variables such as age, gender, household income, and education through the application of demographic weights. Weighting adjustments [1] aim to reduce the bias of estimates by adjusting for demographic differences between the panel and the population it represents. This helps to adjust for the effects of panel attrition, that may cause panels to become less representative of the population over time. This approach - calibrating ad server logs using a smaller high-quality panel - is able to cover a larger part of the long tail of the web than using a panel alone. The benefit is that these reach and frequency estimates get both reduced variance from server logs and reduced bias from the panel.

For our extension of the KSV model it is required that the panel also includes measurement and monitoring across all devices. Furthermore, we need for PPD labels to be recorded, for the panel, in the ad server logs.

2.2 Demographic Correction Model for Single Device

The estimate of a campaign's audience consists of taking the total number of ad impressions, unique cookies exposed to the campaign, a subset of cookies with PPD demographic labels, and then breaking down the impressions and unique cookies into demographic groups. The impressions from each group, when divided by that group's population number and multipled by 100, estimates the GRPs[1] for that group. Finally, these impressions divided by the number of exposed users from that group, estimates the average frequency for that group. This section reviews the models to break either impressions or cookies into demographic groups.

Consider a problem where we have D demographic groups and one publisher providing PPD. Furthermore, assume we have a set of training campaigns[2] broken down by both panel demographics and PPD labels. Let this data consist of N_{train} campaigns each large enough to be confidently measured by the panel. For campaign i, let $\boldsymbol{y_i}$ be the proportion of panelist cookies (or impressions) for each of the panel-measured demographic groups (hence $\boldsymbol{y_i}$ is a vector of length D). Similarly, let $\boldsymbol{x_i}$ be a D-length vector for the PPD proportions. We model the relationship as[3]

$$\boldsymbol{y_i} = (1-\alpha_i)A\boldsymbol{x_i}/||A\boldsymbol{x_i}||_1 + \alpha_i B\boldsymbol{x_i} + \boldsymbol{\epsilon_i} \tag{1}$$

where A is a D x D "correction" matrix, B is a D x D left-stochastic matrix[4], and α_i represents the fraction of cookies (impressions) for the ith campaign either served on the publisher's site or via cookie targeting using the PPD. Hence, $1-\alpha_i$ represents the fraction of unlabeled cookies (or unlabeled impressions) for that campaign.

If the PPD labels are perfect for the publisher's site, then $B = I$. But usually the PPD labels have misclassification issues and hence B should be a left-stochastic matrix. That is, it re-distributes the PPD demographic proportions to better represent the actual population proportions for those cookies with PPD labels exposed to the campaign. It is sometimes possible to estimate B from panel cookies (impression) that have PPD labels directly, rather than through a model fit.

Matrix A is used to measure cookies (impressions) without PPD labels. For these cookies (impressions) it adjusts $\boldsymbol{x}_i$ for both misclassification and non-representiveness between the labeled and

[1] Strictly speaking, this is the target rating points (TRPs) indicating this is the GRPs for this group.

[2] For training data we could use a set of campaigns, site visit data, or a combination of both. Campaigns are required for advertiser-based reports, while site visit data are required for publisher-based reports. We refer to just campaigns here for convenience.

[3] $||\mathbf{z}||_1$ is L_1 distance of $\mathbf{z}$ or $\sum_d |z_d|$.

[4] A left-stochastic matrix is a square matrix with non-negative entries and columns that sum to one.

non-labeled cookies (impression). If B has been fit directly from the panel then the appropriate regression model is

$$\tilde{\boldsymbol{y}}_i = \frac{\boldsymbol{y_i} - \alpha_i B \boldsymbol{x_i}}{1 - \alpha_i} = A\boldsymbol{x_i}/||A\boldsymbol{x_i}||_1 + \boldsymbol{\epsilon_i} \tag{2}$$

and should be fit using either least squares or penalized least squares. KSV found that unconstrained regression provided the best results using simulated data.

2.3 Mapping Cookies to Users

KSV provides an equation for converting cookie counts to people counts. This equation depends on some defined time interval T:

$$u = \frac{c\gamma_T P_T}{C_T + c(\gamma_T - 1)} \tag{3}$$

where u is the estimated people (user) counts, c is the cookie count, P_T is the total active online population count during time interval T, C_T is the total active cookie count during time interval T, and γ_T is a parameter to be estimated. They also found that typically $\gamma_T \approx \kappa C_T/P_T$ for $\kappa \approx 1$. Equation 3 does not generalize well for arbitrary T and doesn't adjust well for a particular demographic group, as C_T is unknown for a any given demographic group. An almost equivalent formulation[5] that is useful for arbitrary T and for a given demographic group d is

$$u_d = \frac{\kappa_d c_d P_d}{P_d + \kappa_d c_d} \tag{4}$$

where P_d is the active online population (defined over a long time period - say 90 days) for group d, c_d is the corrected cookie count for group d (output from Equation 1), and κ_d is a parameter estimated for demographic group d. In practice κ_d could be set to the same κ for all demographic groups and is close to 1.0 for mature cookies and slightly less than 1.0 for younger cookies.

Equation 4 has theoretical justification and has been used in the literature (cf. [9], [3], [11], and [6]). Suppose we have a campaign with c total cookies exposed and that for an arbitrary person the number of his/her cookies exposed, c_i, follows a Poisson distribution with rate parameter λ and that these rate parameters - across people - follow an exponential distribution:

$$\begin{aligned} c_i|\lambda, c &\overset{ind}{\sim}^{6} \text{Poisson}(\lambda) \\ \lambda|c &\overset{iid}{\sim} \text{Exp}(\theta) \end{aligned}$$

Then it is easy to show - by integrating out λ - that

[5] Letting $\kappa = \gamma P/C$ then $\frac{c\gamma P}{C+c(\gamma-1)} = \frac{c\kappa C}{C+c(\kappa C/P-1)} \approx \frac{c\kappa C}{C+c\kappa C/P} = \frac{c\kappa P}{P+c\kappa}$.

[6] The c_i's are not technically independent as $\sum_i c_i = c$, but since the number of people is large, the c_i's are practically independent.

$$\begin{aligned} P(c_i = n|c) &= \int_0^\infty P(c_i = n|\lambda) \cdot f_{\exp}(\lambda|\theta) d\lambda \\ &= \int_0^\infty \frac{e^{-\lambda}\lambda^n}{n!} \theta e^{-\theta\lambda} d\lambda \\ &= \frac{\theta}{n!} \int_0^\infty \lambda^n e^{-\lambda(\theta+1)} d\lambda \\ &= \frac{\theta}{n!} \cdot \frac{\Gamma(n+1)}{(\theta+1)^{(n+1)}} \\ &= \left(\frac{\theta}{\theta+1}\right)\left(\frac{1}{\theta+1}\right)^n \end{aligned}$$

This can be used to calculate

$$P(c_i > 0|c) = 1 - P(c_i = 0|c) = 1 - \frac{\theta}{\theta+1} = \frac{1}{\theta+1}$$

and hence by adding these probabilities over all P people that

$$E[u|c] = \frac{P}{\theta+1} = \frac{P}{P/\kappa_e c + 1} = \frac{\kappa_e c}{P + \kappa_e c}$$

by subsituting $\theta = P/\kappa_e c$ which is the same as Equation 4. We call this cookie-to-user function the *Exponential Bow* model. Note that the derivative of the Exponential Bow model evaluated at the origin ($c = 0$) is κ_e. Hence, κ_e represents the expected number of people reached with the first impression and should be close to 1.0.

Consider another case where every person has the same rate parameter $\lambda_i \equiv \kappa_0 c/P$, then $P(c_i > 0|c) = 1 - e^{-\kappa_0 c/P}$. We call this cookie-to-user function the *Dirac Bow* model. Note that for this model, κ_0 is the slope at the origin so again should be close to 1.0. We will extent these concepts of heterogeneity of the rate parameter to the multi-device situation in Section 4.

Taken together, for a campaign with a total of I impression on c cookies with normalized PPD labels $\boldsymbol{x}$ (a vector of size D) the "Impression" version of Equation 1 is used to get the impression demographic breakdown

$$I_d = y_d^I * I$$

and the "Cookie" version of Equation 1 is used to get the cookie demographic breakdown

$$c_d = y_d^C * c$$

Finally, Equation 4 is used to convert cookies (c_d) to users while the average frequency is calculated as $\bar{f}_d = I_d/u_d$.

3 Demographic Correction Model for Cross-Device Measurement

Section 2.2 reviews the model for estimating GRPs for a single device type. Specifically, first a set of demographic correction models are applied to impressions and cookies, respectively. And second, the results from the cookie-correction model are input into a cookie-to-user model that estimates the unique number of people reached by the campaign. This section generalizes the demographic correction model for multiple device types while Section 4 generalizes the conversion of multiple, post-corrected, cookie types to people.

The generalization of the demographic correction models to multiple device types is straightforward where a unique pair of models (impressions, cookies) is estimated for each device type, as described in Section 1. But first attention is needed to weight the panel properly to reflect the associated target audience. Most panelists will have multiple weights. For example, for two device types there are: weights for device 1, weights for device 2, and then either weights for (device 1 AND 2) or for (device 1 OR 2). For the demographic correction models, the single device weights should be applied for each of the single device correction models. However, the joint weights need to be used in developing the multi-device cookie-to-user models as discussed in Section 4.

For device type j, build the pair of demographic correction models - one for impressions and one for cookies - as prescribed in Section 1. Specifically, find N^j_{train} campaigns each large enough so that device type j activity is confidently measured by the panel. These training data should include cross-device campaigns but could also include single device campaigns. The training data is used to estimate both the impression and correction models (Equation 1) for device type j:

$$\boldsymbol{y^I_{ij}} = (1-\alpha^I_{ij})A^I_j \boldsymbol{x^I_{ij}}/||A^I_j \boldsymbol{x^I_{ij}}||_1 + \alpha^I_{ij} B^I_j \boldsymbol{x^I_{ij}} + \boldsymbol{\epsilon^I_{ij}}$$

$$\boldsymbol{y^C_{ij}} = (1-\alpha^C_{ij})A^C_j \boldsymbol{x^C_{ij}}/||A^C_j \boldsymbol{x^C_{ij}}||_1 + \alpha^C_{ij} B^C_j \boldsymbol{x^C_{ij}} + \boldsymbol{\epsilon^C_{ij}} \tag{5}$$

where the superscript denotes impression or cookie data/parameters, the device specific subscript j indicates the data ($\boldsymbol{y_{ij}}, \boldsymbol{x_{ij}}$, and α_{ij}) for campaign i and device j, and A_j and B_j are the device specific correction and redistribution matrices, respectively. If I_{ij} and c_{ij} are the device j's total impressions and cookies measured for campaign i, respectively, then demographic group d's estimates are $I^d_{ij} = (\boldsymbol{y^I_{ij}})_d \cdot I_{ij}$ and $c^d_{ij} = (\boldsymbol{y^C_{ij}})_d \cdot c_{ij}$. The vector $\boldsymbol{c^d_i} = (c^d_{i1}, c^d_{i2}, \cdots, c^d_{iJ})'$ is the input into the multiple-device reach function for campaign i and demo group d.

4 General Approach for Mapping Cookies to Users

Section 2.3 describes a method for converting cookie counts for a specific demo group into people counts but only handles one device type. Here we extend that model to address the following needs:

- Estimate the number of unique people in a cross-device audience. We need to deal with multiple types of cookie counts instead of just one cookie count. In particular, counts of cookies by device type and potentially app-specific logged-in user ids. We need to treat these cookie types differently, as churning behaviour of desktop and mobile cookies is different. Some people are reachable only through a desktop computer or only through a mobile device.

- Provide modeling flexibility. Equation 7 from KSV depends on one parameter and gives a reasonable first approximation of generic desktop cookie behavior. If sufficient training data is available the accuracy can be improved by a more flexible cookie-to-user model. This can be achieved for both single and multiple cookie types.

We first introduce a simple method involving individual device-specific reach functions. Then we consider a general approach to estimate people counts from multiple cookie types.

4.1 Multiple Device Reach Curves via Independence Assumption

For multiple cookie types we could assume that advertising on different device types reaches people independently from each other. That is, if reach of the j-th cookie type is given by function $R_j(c_j)$[7] then the overall (multi-cookie type) reach function is computed - under this assumption - as

$$R(\boldsymbol{c}) = 1 - \prod_j (1 - R_j(c_j)) \tag{6}$$

This assumption provides a simple and easy method to construct a multi-device reach surface by fitting one dimensional marginals and then joining them. It is often a good working assumption, particularly for campaigns with relatively small reach, and for countries with high percentage of the population using multiple device types. For example, consider a campaign with two device types each having a reach of 10%. This formula would estimate the combined device reach as 19% and the formula estimates that the audience reached by both devices is only 1% (= 10% * 10%). The true overlap would need to be far from this assumption to materially affect the overall reach estimate. However, for large campaigns this isn't true. As another example, consider a campaign with 50% reach for both device types - hence the overall reach estimate is 75% with the overlap estimate of 25% (= 50% * 50%). Now the overlap assumption could materially affect the overall reach estimate. Caution should be exercised to validate this assumption using cross-device campaigns measured by the panel.

This assumption is better for device types with high user pentration. For example, this assumption can't be true in a country where the the single-device ownership is changing over time. As an extreme example, if a country has two types of people - owners of only device type 1 and owners of only device type 2 - then this assumption breaks down as $R(c_1, c_2) = R_1(c_1) + R_2(c_2)$ as the true overlap is zero. There are two easy adjustments to this assumption: adjust the model to accurately account for population device type adoption using census data and possibly adjusting the overlap independence assumption using panel data.

Consider a simplified situation where only two device types are of interest, and let a population have P_1 users of only device 1, P_2 users of only device 2, and P_{12} users of both devices. Suppose we have, based on the single device models, estimates of reach for each device type, R_j. Then a modification of Equation 6 is

$$\begin{aligned} R &= P_1 R_1 + P_2 R_2 + P_{12} \cdot [1 - (1 - R_1)(1 - R_2)] \\ &= (P_1 + P_{12}) R_1 + (P_2 + P_{12}) R_2 - P_{12} R_1 R_2 \end{aligned} \tag{7}$$

[7] $R_j(\cdot)$ is a reach function so its maximum is one ($R(\infty) = 1$)

where the reach estimate for device j is applied directly to the device j subpopulation and then the approach of Equation 6 is only applied to the cross-device subpopulation. This model can further be modified based on information from the panel that the cross-device subpopulation reach overlap doesn't satisfy the independence assumption by applying an additional parameter β_{12}

$$R = (P_1 + P_{12})R_1 + (P_2 + P_{12})R_2 - \beta_{12}P_{12}R_1R_2 \tag{8}$$

Here $\beta_{12} = 1$ matches the independence assumption while $\beta_{12} < 1$ matches positive (> 1 matches negative) correlation between the reach of the two device types.

4.2 Multiple Device Reach Curves via Activity-Distribution Functions

We generalize the modeling of the cookies-to-user mapping function by introducing the concept of an *Activity Distribution Function* (ADF) that models the heterogeneity of the number and type of cookies owned by people. We show that any ADF directly relates to a reach function. We illustrate this for the *Exponential Bow* and *Dirac Bow* models that were introduced in Section 2.3. Finally, we present two particularly useful ADFs: the first based on mixtures of Dirac functions which can model any arbitrary multiple device reach curve; and the second that extends the *Exponential Bow* to allow for more flexibility in modeling the reach curve.

For this section, to generalize to any population, we introduce a new variable, t, that is the average cookie counts rather than raw cookie counts. That is, $t = c/P$ where P is the number of internet users. Particularly, for demographic group d, we convert cookie counts for device j (c_j) by $t_j^d = c_j^d/P_d$ where P_d is the internet population for demographic group d. After dropping the dependence on d, we define $\boldsymbol{t} = (t_1, t_2, \cdots, t_J)'$ as the input into the reach function. We also now model the reach function, $R(\cdot)$, rather than the user function as presented in subsection 2.3. Ultimately, we multiply the output from the reach function by P to yield number of people.

Assume that there is an underlying population of people (P), and each person has a certain probability of generating a cookie of each type. Let the (multivariate) probability distribution $\mathcal{A}$ model the heterogeneity of these probabilities. $\mathcal{A}$ can be converted to a cross-device reach surface using

$$R(\boldsymbol{t}) = \int_{\boldsymbol{x}\in\mathbb{R}^+} \mathcal{A}(\boldsymbol{x}) \cdot (1 - e^{-\boldsymbol{t}\boldsymbol{x}}) d\boldsymbol{x} \tag{9}$$

We have found that cookie-to-user dependencies, that occur in practice, arise from applying Equation 9 using an appropriate distribution. We call the function $\mathcal{A}$ an *Activity Distribution Function* (ADF). Next we illustrate the use of the ADF/Reach function relationshp for one-dimensional reach curves.

4.2.1 Exponential Bow Reach Model

Recall that the Exponential Bow model with $\kappa_e > 0$ is defined in Equation 4. Converting this to a reach function and introducing t yields

$$R(t) = \frac{\kappa_e t}{\kappa t + 1}. \tag{10}$$

This corresponds to an exponential cookie generation probability distribution (ADF) which is defined by

$$\mathcal{A}(x) = \frac{e^{-x/\kappa_e}}{\kappa_e}. \tag{11}$$

Notice that for this ADF, the expected number of cookies per person is κ_e. Interestingly, the exponential ADF has maximum entropy over all ADFs, under the condition that the expected number of cookies is fixed at κ_e.

4.2.2 Dirac Bow Reach Model

Also recall the Dirac Bow model with $\kappa_0 > 0$. In terms of a reach function this is defined as

$$R(t) = 1 - e^{-\kappa_0 t}. \tag{12}$$

The corresponding ADF is a Dirac delta function located at κ_0, i.e.

$$\mathcal{A}(x) = \delta(x - \kappa_0) \tag{13}$$

Note that when assigning c cookies to a set of people, the Dirac ACF corresponds to distributing them according to an uniform distribution (each person has equal probability of being assigned any of the c cookies). Subsequently, the assignment of cookies-to-people has maximum entropy.

4.2.3 Dirac Mixture Models

We can extend the Dirac Bow model to higher dimensions by considering a multivariate Dirac Delta function located at $\boldsymbol{x^0} = (x_1^0, x_1^0, \cdots, x_J^0)'$. That is, we can define the ADF as

$$\mathcal{A}(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x^0}).$$

The assumption of similar device usage across all people does not yield a particularly interesting ADF in itself. However, we can add arbitrary heterogeneity by considering mixtures of multivariate Dirac Delta functions (cf. [8] and [7])

$$\mathcal{A}(\boldsymbol{x}) = \sum_k \alpha_k \delta(\boldsymbol{x} - \boldsymbol{x_k^0}). \tag{14}$$

This ADF has subpopulations of people with each subpopulation having similar device usage. For example, subpopulation k has usage centered at $\boldsymbol{x_k^0}$ and the fraction of the population represented by this group is α_k. For this ADF, we have the associated reach surface represented as a sum of exponents

$$R(\boldsymbol{t}) = \sum_k \alpha_k (1 - e^{\boldsymbol{x_k^0} \cdot \boldsymbol{t}}). \tag{15}$$

If we have training data in the form of $(\boldsymbol{t}_i, r_i)$[8] and choose a set of subpopulations centered at $\boldsymbol{x}_k^0$, then we can easily find the set of coefficients α_k's using constrained linear regression (as the $\alpha_k \geq 0$). The locations of $\boldsymbol{x}_k^0$ can either be picked along a grid or found via local search. This approach is illustrated in Section 4.3 and specific algorithms are discussed in the Appendix.

[8] r_i is an estimate of the reach surface at $\boldsymbol{t}_i$ so $r_i \approx R(\boldsymbol{t}_i)$.

4.2.4 Generalized Exponential Family Distribution

The exponential ADF is often a good approximation of the reach surface, but it is natural to consider its generalization in cases when more flexibility is needed. In this subsection we restrict ourselves to the one dimensional case. Multidimensional generalization is conceptually straightforward, but requires working with more complex indices.

Consider the *Generalized Exponential* ADF of order N - defined for $x > 0$ - as

$$\mathcal{A}(x) = e^{\sum_{n=0}^{N} \lambda_n x^n},$$

for parameters $\lambda_0, \ldots, \lambda_n$[9].

In this case the reach curve has form

$$R(t|\boldsymbol{\lambda}) = \int_0^\infty e^{\sum_{n=0}^{N} \lambda_n x^n} (1 - e^{-xt}) dx.$$

Techniques for finding such parameters by matching first moments of the distribution are well known [15]. Note that the moments of the distribution are equal to the corresponding derivatives of the reach curve evaluated at $t = 0$. (e.g., first moment is equal to the first derivative).

One of the simplest algorithms that can be used for fitting the Generalized Exponential distribution in the context of reach estimation is gradient descent. Indeed, the partial derivative of the reach curve with respect to λ_k has the form

$$\frac{\partial R(t)}{\partial \lambda_k} = \int_0^\infty x^k e^{\sum_{n=0}^{N} \lambda_n x^n} (1 - e^{-xt}) dx \tag{16}$$

which can be calculated by numeric integration. Thus given a set of points of the reach curve $\{(t_i, r_i)\}$ we can calculate the gradient of the reach estimation error

$$\boldsymbol{\nabla}_{\lambda_0,\ldots,\lambda_N} \left(\sum_i (R(t_i) - r_i)^2 \right) = 2 \sum_i \frac{\partial R(t)}{\partial \lambda_k} \cdot (R(t_i) - r_i) \tag{17}$$

and use it in the gradient descent algorithm to optimize the parameters $\lambda_0, \ldots, \lambda_N$.

4.3 Simulations

We illustrate the multiple device reach methods presented above using various simulation scenarios. We begin with three examples demonstrating the performance of the Adaptive Dirac Mixture algorithm for estimating both the underlying ADFs and more importantly the reach surface. We also include a brief example for fitting the Generalized Exponential reach curve.

4.3.1 Adaptive Dirac Mixture

The Adaptive Dirac Mixture (ADM) algorithm is a very general procedure for estimating the Dirac Mixture Model. It estimates the number of Dirac mixtures, their locations, and associated weights. More details on this algorithm can be found in the Appendix. For each of our examples we begin

[9] We require that $\lambda_N < 0$ and note that λ_0 is used as a normalizing constant so that $\mathcal{A}$ integrates to one.

by constructing a true underlying ADF. We next construct a training set of I campaigns. For each campaign, we randomly simulate cookie counts across multiple cookie types (i.e., $\boldsymbol{t_i}$) using a truncated (at 0) Gaussian with mean 0.5 and standard deviation of 1.5. We then use Equation 9 to find the corresponding reach. Hence we construct $(\boldsymbol{t_i}, r_i)$ for $i = 1, \cdots, I$. Finally, we use these as inputs to the ADM algorithm to estimate the ADF and associated reach surface. We assume no error in the r's for our examples. However, our simulations indicate that the algorithm is robust against reasonable noise.

For our first example, we construct an ADF using nine Dirac mixtures located at random positions all with equal weights ($\alpha_k = 1/9$). We randomly generate our $I = 2,000$ campaigns and then estimate the ADF. We initialize the algorithm with one Dirac and it converges with nine clusters of Diracs each with weight very close to 1/9 and with locations indistiguishable from the original ADF (see Figure 1).

In our second example, we increase the number of cookie types to three and use a continuous ADF, specifically a trivariate Gaussian distribution with the mean of $(0.7, 0.8, 1.0)'$ and covariance matrix

$$\begin{pmatrix} 0.05, & 0.05, & 0.00 \\ 0.05, & 0.10, & 0.00 \\ 0.00, & 0.00, & 0.15 \end{pmatrix}$$

For this ADF, analytically solving Equation 9 is impossible and hence we use Monte-Carlo integration, specifically in the form of

$$R(\boldsymbol{t}) = \int_{\boldsymbol{x} \in \mathbb{R}^+} \mathcal{A}(\boldsymbol{x}) \cdot (1 - e^{-\boldsymbol{t}\boldsymbol{x}}) d\boldsymbol{x} \approx \frac{1}{|Sample(\mathcal{A})|} \sum_{\boldsymbol{x}_l \in Sample(\mathcal{A})} (1 - e^{-\boldsymbol{t}\boldsymbol{x}_l}), \tag{18}$$

where $Sample(\mathcal{A})$ is a sample of $|Sample(\mathcal{A})|$ points from the distribution $\mathcal{A}$. In our simulations we use 1,000 points. Note that this integration actually reduces the continuous distribution to a Dirac mixture. Since the difference in estimated reach surfaces using different large samples is very small, the ADM algorithm does not converge to the exact sample, but rather to some other configuration that approximates the underlying continuous ADF.

Figure 2 shows the $Sample(\mathcal{A})$ on the left and the estimated ADF using the ADM algorithm on the right. For this example we increased I to 3,000. The top row shows the three-dimensional centers of the ADFs while the middle and bottom rows show two-dimensional scatterplots. The estimated ADF has mean $(0.711, 0.818, 1.003)$ which is within three digits of the average from $Sample(\mathcal{A})$. The estimated ADF has covariance matrix

$$\begin{pmatrix} 0.046, & 0.045, & -0.001 \\ 0.045, & 0.092, & -0.001 \\ -0.001, & -0.001, & 0.167 \end{pmatrix}$$

which is within two digits of the covariance matrix from $Sample(\mathcal{A})$. Hence the majority of the error is introduced by using Monte-Carlo integration rather than from the ADM algorithm.

Most important is how well we construct the reach surface as this is the ultimate use of this method. Figure 3 shows the scatterplot of $\hat{R}_i$ vs c_i for the I campaigns on the left and the estimate reach $(\hat{R}_i)$ vs. the truth (R_i) on the right. In this example we see that the model almost exactly estimates the true reach surface.

For our third example, we make the ADF more complicated by taking a 50/50 mixture of trivariate Gaussian distributions. The first distribution is the same as in Example 2 while the second distribution has mean $(1.5, 0.5, 0.5)$ and covariance matrix

$$\begin{pmatrix} 0.0\bar{3}, & -0.0\bar{3}, & 0.00 \\ -0.0\bar{3}, & 0.0\bar{6}, & 0.00 \\ 0.00, & 0.00, & 0.15 \end{pmatrix}$$

Figure 4 shows the results in fitting this ADF - analogous to Figure 2. Again, we very closely reconstruct the ADF (we are actually closer to $Sample(\mathcal{A})$) and the reach surface estimates are reconstructed almost exactly (not shown but similar to Figure 3).

These examples (and other simulations we've performed) demonstrate that the ADM algorithm with an appropriate amount of training data, starting from centers sampled uniformly at random from a cube of an appropriate dimension, can closely approximate reasonably complex ADFs and their associated reach surfaces.

4.3.2 Generalized Exponential Distribution

As discussed in Section 4.2.4, gradient descent can be used to estimate parameters of a Generalized Exponential distribution. Since it involves doing computationally expensive numeric integration at each step, we recommend using it with a very few pre-computed reach curve points. For instance, the points can be obtained from counts of cookies and people for weekly and monthly audiences of the network in question. Figure 5 is an example of an Generalized Exponential-based reach curve. We fit it using gradient descent to pass through two points: $(0.05, 0.04)$ and $(1.0, 0.47)$.

5 Case Study

We illustrate the methods using desktop and smartphone campaigns served in Japan by the Google Display Network, DoubleClick for Advertisers, and DoubleClick for Publishers. We first describe the panel and the PPD available in the ad server logs. We then show the performance of the cookie-correction models for both desktop and smartphone. And finally, we show the results for the cross-device cookie-to-user models.

5.1 The panel

Panel data is provided by Intage[10] from their i-SSP (INTAGE Single Source Panel). It includes panelist weights that are calibrated to population benchmarks derived from the Population Census conducted by Japan Statistic Bureau[12] and Intage's propriertary survey. Figure 6 shows the number of panelists by gender and 10-year age groups. It shows that the 13-17 (aka age 13 to 17) and 65-99 age groups have very few panelists. In this case study, we remove all 13-17 panelists from consideration, and merge 55-64 and 65-99 panelists into a common 55-99 age group, separately for each gender.

[10]Intage's web site: https://www.intage.co.jp/english

5.2 Campaign data

This case study focuses on Google ads campaigns in Japan that were active from 2016/02/01 to 2016/02/28, and reached at least 100[11] panelists from desktop or from smartphone. We collected Google ad serving events for these campaigns together with their cookies and their YouTube declared labels when available. YouTube declared labels are provided by users when they create their YouTube accounts. Such labels were merged with Google ad events for logged-in users. Figure 7 shows the histograms of α_{ij}, the fraction of cookies that had YouTube declared labels in a campaign for desktop and smartphone, respectively.

5.3 Corrected cookies by device

KSV describes the root mean squared error (RSME) for measuring the goodness of fit of a model. We introduce the shuffle distance as an additional metric to gain better interpretation and insight of the performance of the model's ability to estimate demographic decompositions. The shuffle distance is very similar to *edit distance* [5]. It measures the difference between two proportion vectors by computing the minimum fraction that needs to be relabelled to achieve an exact match. In our case, shuffle distance is defined as

$$\text{shuffle}_{ij} = \frac{||\boldsymbol{y_{ij}} - \boldsymbol{\hat{y}_{ij}}||_1}{2}. \tag{19}$$

$\boldsymbol{y_{ij}}$ represents the demo proportion of cookies observed from panel data for campaign i and device j. It is regarded as ground truth for training. $\boldsymbol{\hat{y}_{ij}}$ represents our model estimate.

Section 3 describes the methodology for training per device cookie-correction models for both impressions and cookies. This case study only focuses on cookie-correction models for desktop and smartphone. The redistribution matrix B_j, where j indexes device, represents the probabilty of true demo of a cookie given its observed YouTube label. It can be computed directly by counting weighted (using associated panelists weights) cookies by their true demo (rows) and YouTube labels (columns) and then column normalizing. The correction matrix A_j is trained by campaigns using cookies from device j. For these models we require for each training campaign that at least 100 panelists are reached for the desktop (smartphone) models. We use unconstrained linear regression as discussed in KSV.

Table 1 summarizes the performance of our models for demographic proportions. The first and the second rows in the table are for the cookie-correction models for desktop and smartphone, respectively. The third and the fourth rows are for the overall cross-device people demographics, and will be discussed in the next section.

The first column in Table 1 reports the number of campaigns. The second and third columns are RMSE and the average shuffle distance for 10-fold cross validation. The last three columns report the performance of models trained and evaluated using all data samples. Ten-fold cross validation [14][2] is a standard model validation technique for assessing how the results of a statistical model will generalize to an unseen data set. The fact that the performance of 10-fold validation is very close to those using all campaigns confirms that the training procedure does not have any generalization or overfitting issues.

[11] While arbitrary, we require campaigns to have at least 100 panelists reached to maintain precision of the "ground truth" estimated from the panel. Using a lower cutoff allows too many noisy campaigns into the dataset while using a higher cutoff biases the dataset towards large campaigns. With a 100 panelists cutoff, we are able to include campaigns with reach as low as 0.6%.

In practice, we consider 20% shuffle distance to the ground truth is an acceptable difference. The "%within20" (the last column in Table 1) measures the faction of campaigns whose shuffle distances to the respective ground truth are less than 20%, and thus have acceptable performance. As shown in the table, 92.0% of desktop campaigns and 90.5% of smartphone campaigns have their cookie demos estimated within the acceptable distance (20% shuffle distance) to their respective panel truths.

Figures 8 and 9 show the demo proportion comparison for cookies between the "panel" ground truth (y-axis) and the estimate (x-axis) for each demo group for desktop and smartphone, respectively. These plots show that our model fits the training campaigns reasonably well for both desktop and smarpthone.

5.4 Cross-device people demo and reach

The previous section evaluates the per device cookie-correction model. This section evaluates the cross-device cookie-to-user models for the independence model as described in Section 4.1 and the Dirac mixture model as described in Section 4.2.

5.4.1 Cross-device independence model

Following the methodology presented in Section 4.1, we first train Bow models for desktop and smarphone, separately. These Bow models estimate people reached by cookies from a single device. We then dedupe per device reach through the independence assumption (Equation 6).

The Dirac Bow model fits the training data better than the Exponential Bow model for both devices. The fitted kappas are 0.92 and 1.00 for desktop and smartphone, respectively. As expected the desktop model has a lower kappa as a person has more desktop cookies (because of higher chance of cookie churn and multiple browsers) than smartphone cookies. Figures 10 and 11 shows per device reach results for desktop and smartphone, respectively. Overall, the model reach estimates match the panel estimates very closely. The smartphone model shows a close one-for-one matching of cookies to people.

To evaluate the cross-device reach model using the independence assumption, we focus on cross-device campaigns that have reached at least 100 panelists from desktop and at least 100 panelists from smartphone. We evaluate the performance for both cross-device people demo decomposition and cross-device people reach.

The third row in Table 1 shows the summary performance for cross-device people demographic estimates. The results show that the independence cross-device model performs reasonably well: 91.3% of evaluation campaigns are within acceptable distance to their respective panel ground truth. Figure 12 shows detailed comparisons of the cross-device people demographic proportions between the model estimates and the panel ground truth.

The first row in Table 2 shows the summary performance of cross-device people reach for the independence model. The cross-device reach estimates are within 10% of the panel estimates for 88.6% of campaigns. Figure 13 shows the reach performance for the campaigns. The left plot shows the relative difference between estimated reach from the model and the ground truth from the panel while the right plot shows the panel vs. model estimates.

In summary, the independence cross-device cookie-to-user model performs well for estimating both the people-demographic proportions and total reach.

5.4.2 Dirac Mixture Model

The Dirac mixture model described in section 4.2 was trained by the ADM algorithm described in Appendix A.2. The fitted model has three Dirac delta functions (see parameters in Table 3). The first Dirac delta represents people who have only smartphone devices (estimated at 10.6%), the second Dirac delta represents people who have both desktop and smartphone devices (estimated at 47.0%), and the third Dirac delta represents people who have only desktop devices (estimated at 42.4%). While the properties of the Dirac mixtures are interesting, they are ultimately a means to estimate the reach surface and hence should not be over-interpreted.

The fourth row in Table 1 shows the summary performance for cross-device people demographic estimates. The %within20 is 90.9% which is slightly better than the independence model although all metrics are very close between the two models. Figure 14 shows detailed comparisons of the cross-device people demographic proportions between the model estimates and the panel ground truth. The results are almost identical to those for the independence model (Figure 12).

The second row in Table 2 shows the summary performance of cross-device people reach using the Dirac mixture model. The cross-device reach estimates are within 10% of the panel estimates for 88.1% of campaigns - slightly better than the independence model. Figure 15 shows the reach performance by campaigns for the Dirac mixture model. Again, it has very similar results as the independence model.

6 Conclusion

We have developed a generalized methodology for measuring the reach and frequency of online audiences with demographic breakdowns. The method handles cross-device audiences and combinations of cookie types and therefore can measure both signed-in and signed-out users. The method calibrates ad server logs and PPD using a smaller high-quality panel that is itself calibrated to census benchmarks. To measure cross-device audiences, we have introduced an *Activity Distribution Function* that models the joint cookie ownership distribution across a population. We've included algorithms for fitting the ADF and provided simulation results that demonstrates the method provides accurate results given enough training data from the calibration panel.

We demonstrated the method using data from Japan where we fit two reach models:

- one that assumes that campaigns reach desktop and smartphone users independently
- one using the Dirac mixture model and fit using the ADM algorithm

In this example, both models fit the campaign data well with over 90% of campaigns within 20% shuffle distance for demographic breakdowns and over 88% of campaigns within 10% for reach. Apparently, for these data, the independence assumption is not too strict an assumption. However, in general, not all markets nor device combinations considered will adhere to this assumption. The Dirac mixture model, with its added flexibility, fits slightly better and provides a more generalized solution.

References

[1] Bethlehem, J. Selection bias in web surveys. *International Statistical Review*, 78(2), 161-188, 2010.

[2] Cross validation wiki. More information of cross validation can be found in *https://en.wikipedia.org/wiki/Cross-validation_(statistics)*

[3] Danaher, P. Modeling Pageviews across multiple websites with an application to internet reach and frequency prediction. *Marketing Science*, 26(3), 422-437, 2007.

[4] Dirac, P. (1958), The Principles of Quantum Mechanics (4th ed.), Oxford at the Clarendon Press, ISBN 978-0-19-852011-5.

[5] Edit distance. Wiki page for edit distance: *https://en.wikipedia.org/wiki/Edit_distance*

[6] Georg, G. (2014) Estimating reach curves from one data point. Technical report, Google, Inc.
http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/43218.pdf

[7] Hanebeck, U, M Huber, and V Klumpp. Dirac mixture approximation of multivariate Gaussian densities. Joint 48th IEEE conf. on decision and control and the 28th chinese control conference. Shanghai, P R China. Dec. 16-18, 2009. p3851-8.

[8] Hanebeck, U, and O Schrempf. Greedy algorithms for Dirac mixture approximation of arbitrary probability density functions. Proceedings of the 2007 IEEE conference on decision and control (CDC 2007). New Orleans, LA. Dec. 2007. p3065-71.

[9] Huang, C-Y and C-S Lin. Modeling the audience's banner ad exposure for internet advertising planning. J. Advertising, 35(2), 23-37, 2006.

[10] Hormozi, A. Cookies and privacy. *EDPACS*. Vol.32, Iss. 9; pp. 1-13, 2005.

[11] Jin, Y, S Shobowale, J Koehler, and H Case. (2012) The incremental reach and cost efficiency of online video ads over tv ads. Techinical report, Google, Inc.
http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/40426.pdf

[12] Japan Population Census and Statistics Bureau, Japan. 2010 population census. *http://www.stat.go.jp/english/data/kokusei/index.htm*

[13] Koehler, J, E Skvortsov, and W Vos. A method for measuring online audiences. Technical report, Google Inc.
http://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/41089.pdf

[14] Stone M. Cross-validatory choice and assessment of statistical predictions. *J. Royal Stat. Soc.*, 36(2), 111-147, 1974

[15] Wu, X. Calculation of maximum entropy densities with application to income distribution. *Journal of Econometrics*, 115(2), 347-354, 2003.

A Algorithms

In this section we describe practical algorithms for fitting reach surfaces. The first two algorithms show how to fit marginal reach curves for the Exponential and Dirac Bow models, respectively. The third and fourth algorithms generalize this to fit reach surfaces using the Dirac Mixture Model. Algorithm 3 assumes that the Dirac mixture centers are fixed and hence uses least squares to find the population fractions (α's). Algorithm 4 shows the adaptive Dirac mixture algorithm where the number and location of the Dirac mixtures are allowed to vary. Lastly, we describe how to fit the Generalized Exponential ADF in Algorithm 5.

A.1 Building reach surfaces from simple marginals

When we build cross-device reach surface via independence assumption we just need to fit the reach marginals. The Exponential Bow and Dirac Bow models use two coefficients: κ and the maximum population estimate P. Now for the population limit it is natural to use the internet population usually available from census data and hence the models only have one parameter to estimate. When fitting to panel data it is reasonable to set P to be the total number of panelists.

The Intuitive interpretation of the κ parameter is the number of people per cookie in small audiences. We recommend estimating κ via quantile regression by selecting the median κ going through points of your training data.

input : training data $\{(c_i, r_i)\}_{i\in\{1,\dots,n\}}$, imposed limit P
set κ to be an empty array;
for $i \in \{1, \dots, n\}$ **do**
  append element $\frac{r_i P}{c_i P - r_i c_i}$ to the end of κ
end
return median(κ)

Algorithm 1: Exponential Bow Model

input : training data $\{(c_i, r_i)\}_{i\in\{1,\dots,n\}}$, imposed limit P
set κ to be an empty array;
for $i \in \{1, \dots, n\}$ **do**
  append element $\frac{-P\log(1-c_i/P)}{t_i}$ to the end of κ
end
return median(κ)

Algorithm 2: Dirac Bow Model

A.2 Building reach surfaces from Dirac mixtures

As it was mentioned above, when we have a collection of locations of delta functions then the Dirac Mixture can be fit using least squares. We define $[\boldsymbol{x_1^0}, \dots, \boldsymbol{x_K^0}]$ as a matrix composed of vectors $\boldsymbol{x_1^0}, \dots, \boldsymbol{x_K^0}$ as its columns.

If the dimensionality (e.g. number of devices) of your model is low (say 1 or 2) then the activity space can be covered by a grid of reasonably high precision and Algorithm 3 can be used to find weights (α's) of the delta functions located on the grid. As the dimensionality increases above two, the size of a grid with reasonable resolution becomes prohibitively high. For instance, in four

dimensions a grid of 10 points in each dimension has 10,000 points. For this situation the Adaptive Dirac Mixture algorithm is better. It tries to locate the number of delta functions, their optimal positions, and associated weights.

In summary, Algorithm 4 does at each iteration

- adds new centers around existing ones,
- finds optimal weights of the new set of centers and
- removes centers of zero weight.

input : training data $\{(\boldsymbol{c_i}, r_i)\}_{i\in\{1,\dots,n\}}$, imposed maximum population P, locations of delta functions, J-dimensional vectors $\{\boldsymbol{d_k}\}_{k\in\{1,\dots,K\}}$
output : A collection of delta function centers and coefficients $\{(\boldsymbol{d_k}, \alpha_k)\}_{k\in\{1,\dots,K\}}$ defining a Dirac Mixture
let $C_{ik} = 1 - e^{\boldsymbol{c_i}\cdot\boldsymbol{d_k}}$; // C is $n \times m$ matrix
let $\boldsymbol{\alpha} = \mathrm{NNLS}(C, \boldsymbol{r})$; // $\boldsymbol{\alpha}$ is m-dimensional
return $\{(\boldsymbol{d_k}, \alpha_k)\}_{k\in\{1,\dots,K\}}$

Algorithm 3: Dirac Mixture Coefficients

input : training data $\{(\boldsymbol{c_i}, r_i)\}_{i\in\{1,\dots,n\}}$, imposed maximum population P
output : A collection of delta function centers and coefficients $\{(\boldsymbol{d_k}, \alpha_k)\}_{k\in\{1,\dots,K\}}$ defining a Dirac Mixture
parameters: l: number of random centers to add at each step, σ: variance of random centers, N: number of iterations to run
let $m = 1$;
let $\alpha_1 = 1$;
let $\boldsymbol{d_1} = (1, \dots, 1)$; // Starting with $1 - e^{-\sum c_j/P}$
for *iteration* $\in \{1, \dots, T\}$ **do**
  sample $\{\boldsymbol{d}_k\}_{k\in\{m+1,\dots,m+l\}}$ from $\sum_{k=1}^{m} \alpha_k N(\boldsymbol{d_k}, \mathrm{diagonal}(\sigma))$;
  let $m = m + l$;
  let $\boldsymbol{\alpha}$ be result of the call to Algorithm 3 on $\{(\boldsymbol{c_i}, r_i)\}_{i\in\{1,\dots,n\}}$ and $\{\boldsymbol{d_k}\}_{k\in\{1,\dots,m\}}$;
  update $m, \boldsymbol{\alpha}, \{\boldsymbol{d_k}\}_{k\in\{1,\dots,m\}}$ removing $\alpha_k, \boldsymbol{d_k}$ where $\alpha_k = 0$
end

Algorithm 4: Adaptive Dirac Mixtures

A.3 Fitting a Generalized Exponential ADF

Recall that parameters $\lambda_1, \dots, \lambda_N$ of a Generalized Exponential ADF can be fit via gradient descent, where the gradient is computed by formula (16).

input : training data $\{(c_i, r_i)\}_{i\in\{1,\ldots,n\}}$, imposed maximum population P
output : A set of parameters $\lambda_0, \ldots, \lambda_N$ defining a Generalized Exponential ADF that approximates the training data.
let $\lambda_0 = 1, \lambda_1 = -1$ and $\lambda_n = 0$ for $n > 1$; `// starting with an Exponential Bow`
for *iteration* $\in \{1, \ldots, T\}$ **do**
  compute gradient of reach estimation error $\boldsymbol{\nabla}$ using formulas (16) and (17);
  update $\lambda_0, \ldots, \lambda_N$ subtracting $\boldsymbol{\nabla}$;
end

Algorithm 5: Fitting a Generalized Exponential ADF by Gradient Descent

Tables

		10-fold CV		Full samples		
	#campaigns	avg shuffle	RSME	avg shuffle	RSME	%within20
Desktop (indep. model)	4418	0.121	0.0336	0.121	0.0334	92.0%
Smartphone (indep. model)	766	0.128	0.0339	0.125	0.0339	90.5%
X-device (indep. model)	596			0.114	0.0321	91.3%
X-device (Dirac mix model)	596			0.115	0.0324	90.9%

Table 1: Demo performance. The first two rows provide evaluation results for the desktop and the smartphone cookie-correction models using the independence model, respectively. The third and the forth rows are the evaluation for cross-device people demo for the indendence model and the Dirac mixture model, respectively.

	#campaigns	Avg relative diff	%within10	%within20
Independence model	596	0.057	88.6%	96.6%
Dirac mixture model	596	0.053	88.1%	96.6%

Table 2: Cross-device people reach performance for the independence model and the Dirac mixture model. Relative difference is the defined as the absolute difference between the ground truth (reach observed from panel data) and the estimated reach from our model divided by the ground truth. $\%within10$ and $\%within20$ are the fraction of campaigns whose relative differences to their ground truths are less than 10% and 20%, respectively.

Dirac delta index	1	2	3
Weight (α)	0.106	0.470	0.424
Desktop (x_{i1})	0.00	0.922	1.10
Smartphone (x_{i2})	4.64	1.28	0.00

Table 3: Estimated parameters of Dirac mixture model using the ADM algorithm. It has three Dirac deltas shown as columns. Each Diract delta is paramerized by weight (α), desktop activity and smartphone activity.

Figures

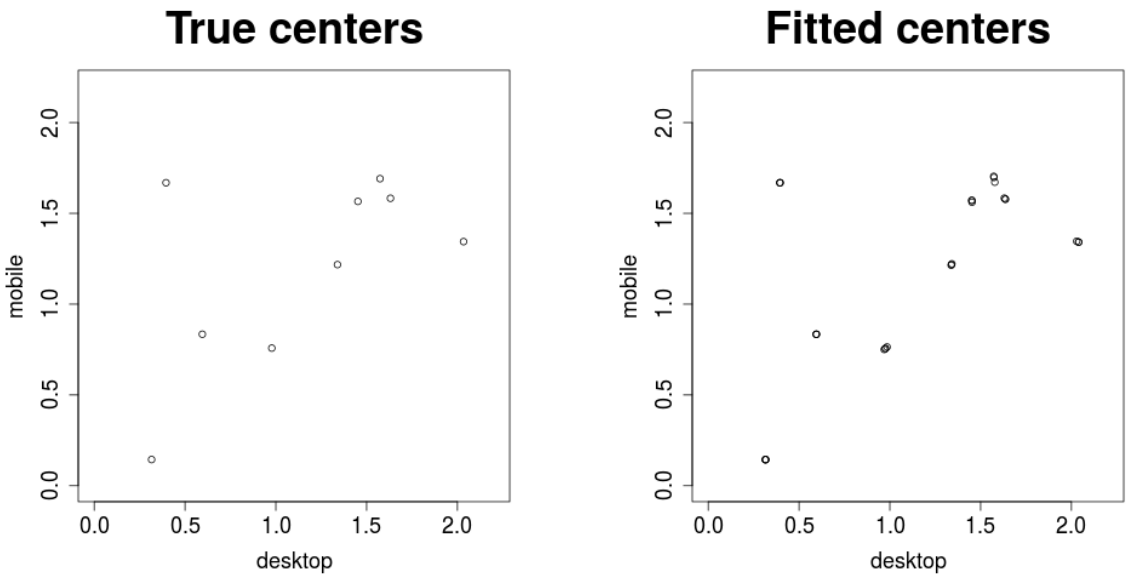

Figure 1: True locations of the ADF Dirac Mixture (left) and the result of the ADM algorithm (right) using the simulated training data from Example 1.

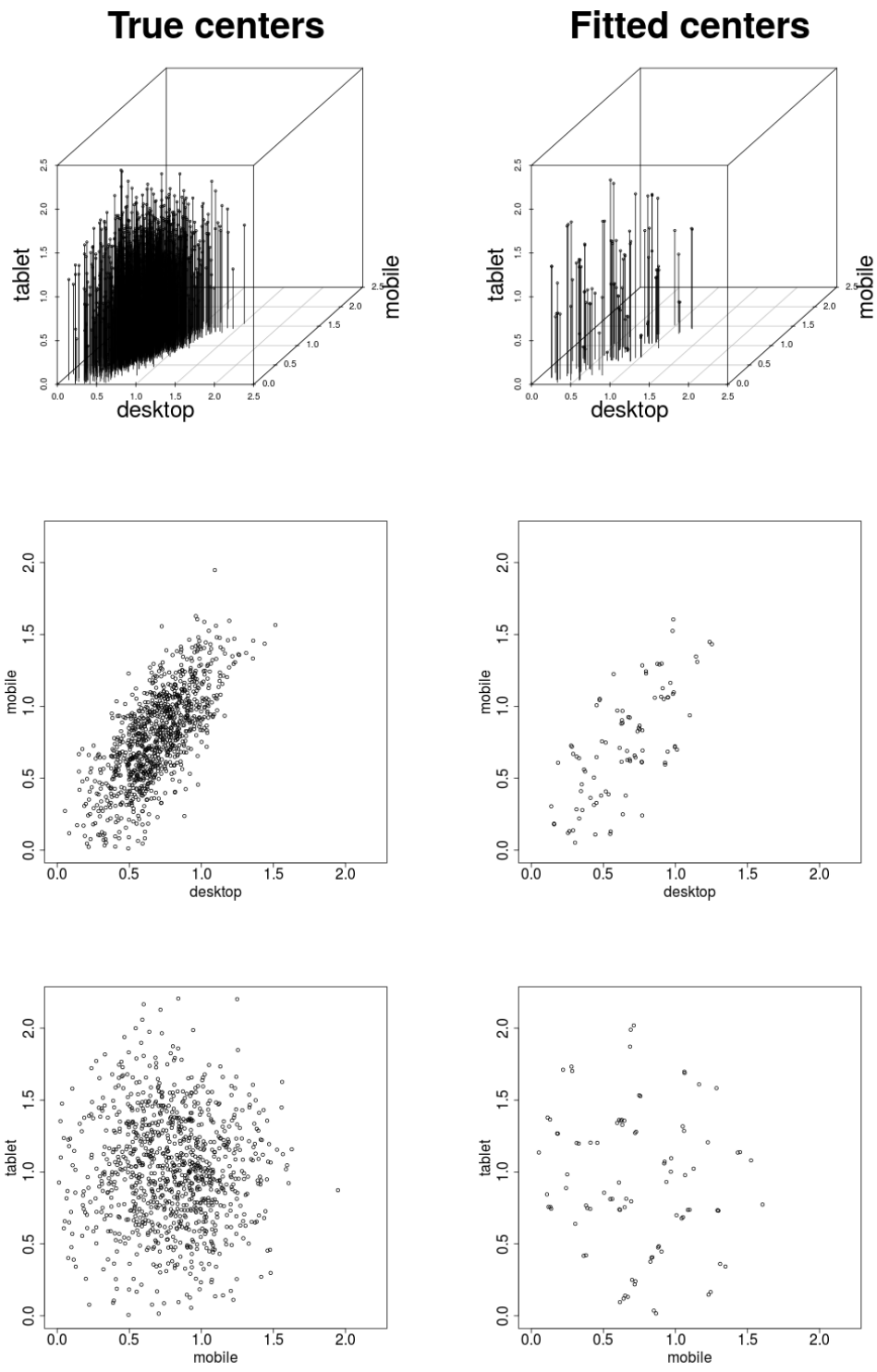

Figure 2: True locations of the ADF Dirac Mixture (left) and the result of the ADM algorithm (right) using the simulated training data corresponding to a 3-dimensional Gaussian ADF from Example 2.

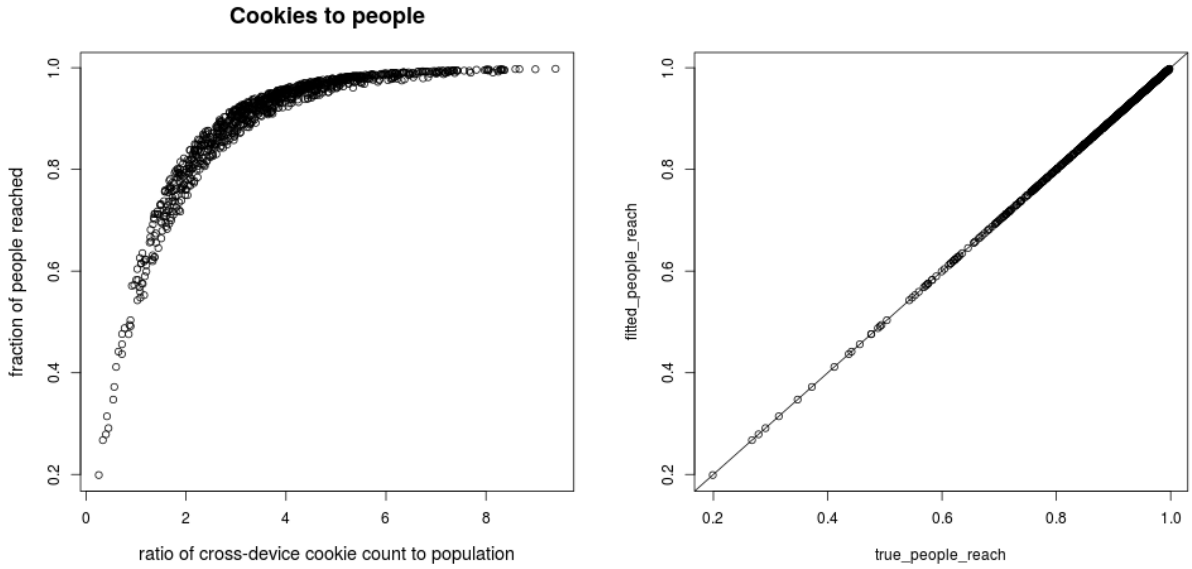

Figure 3: Cookies-to-people (left) and reach truth-to-estimate (right) scatterplots for the simulated three-dimensional normal ADF from Example 2 (see also Figure 2).

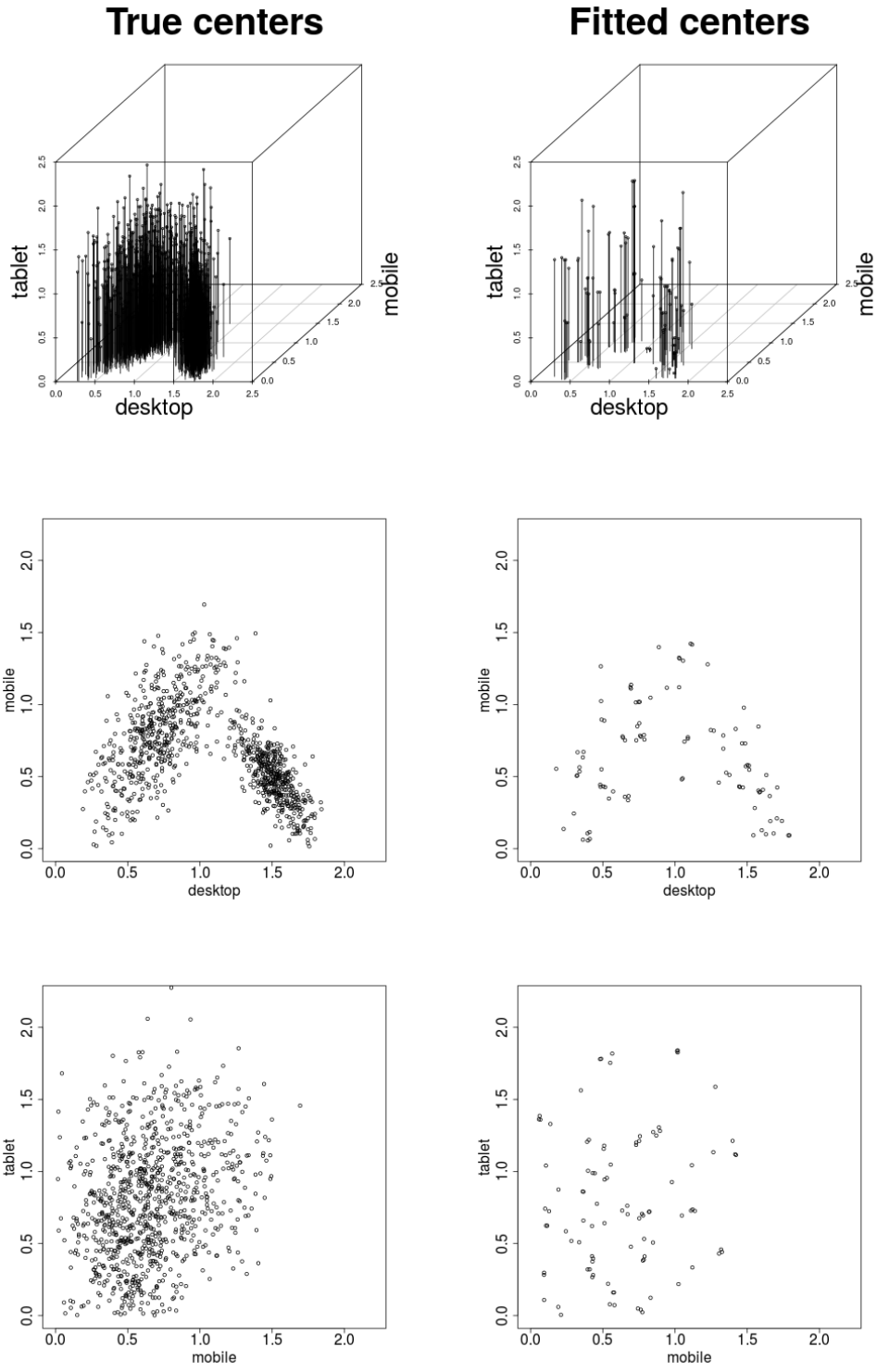

Figure 4: True locations of the ADF Dirac Mixture (left) and the result of the ADM algorithm (right) using the simulated training data corresponding to a 3-dimension Gaussian mixture ADF from Example 3.

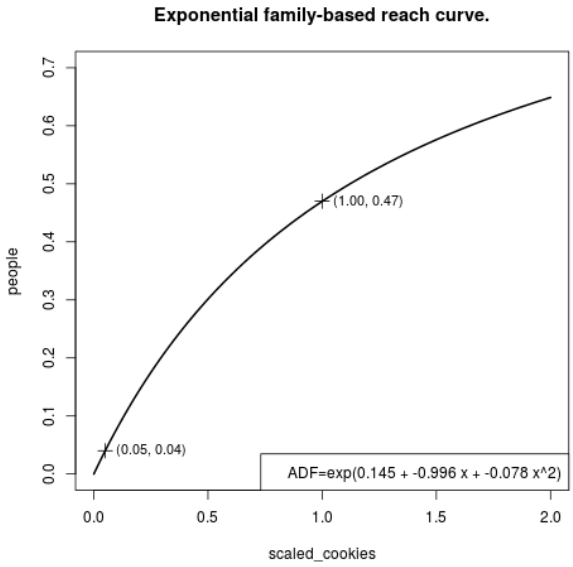

Figure 5: Generalized Exponential based curve.

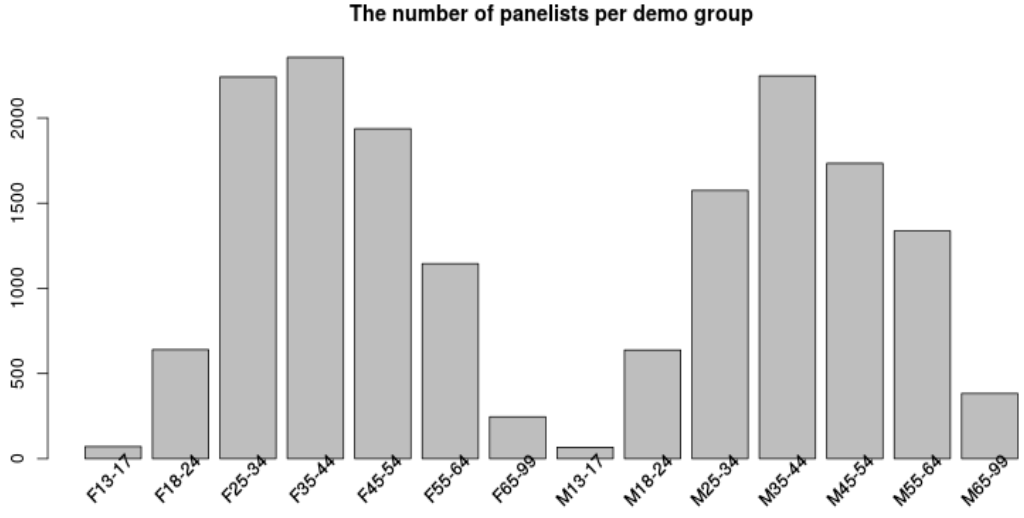

Figure 6: The number of panelists by gender and 10-year age demo group.

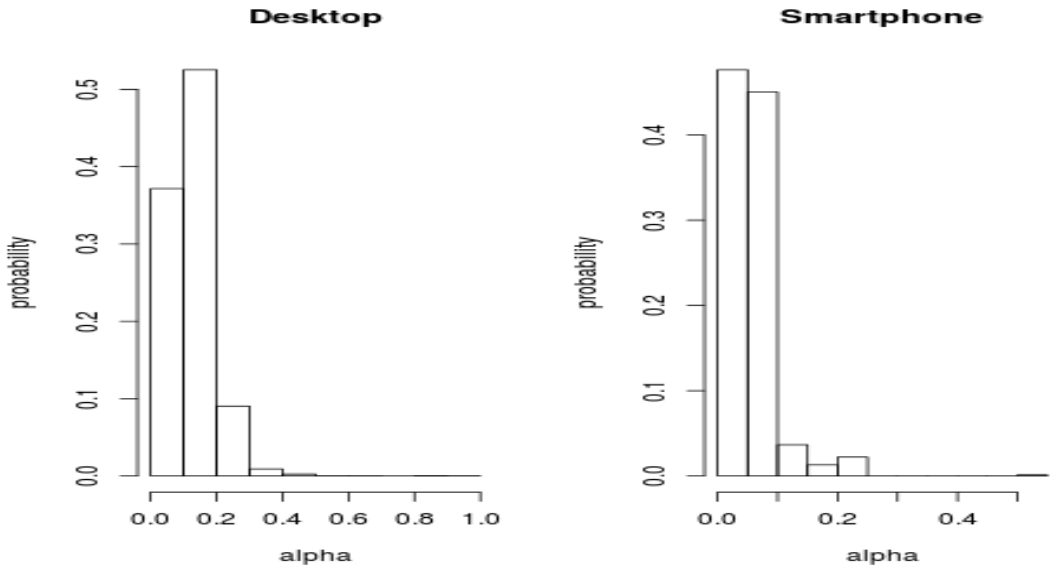

Figure 7: Distribution of cookie proportions with YouTube labels across campaigns, split by device.

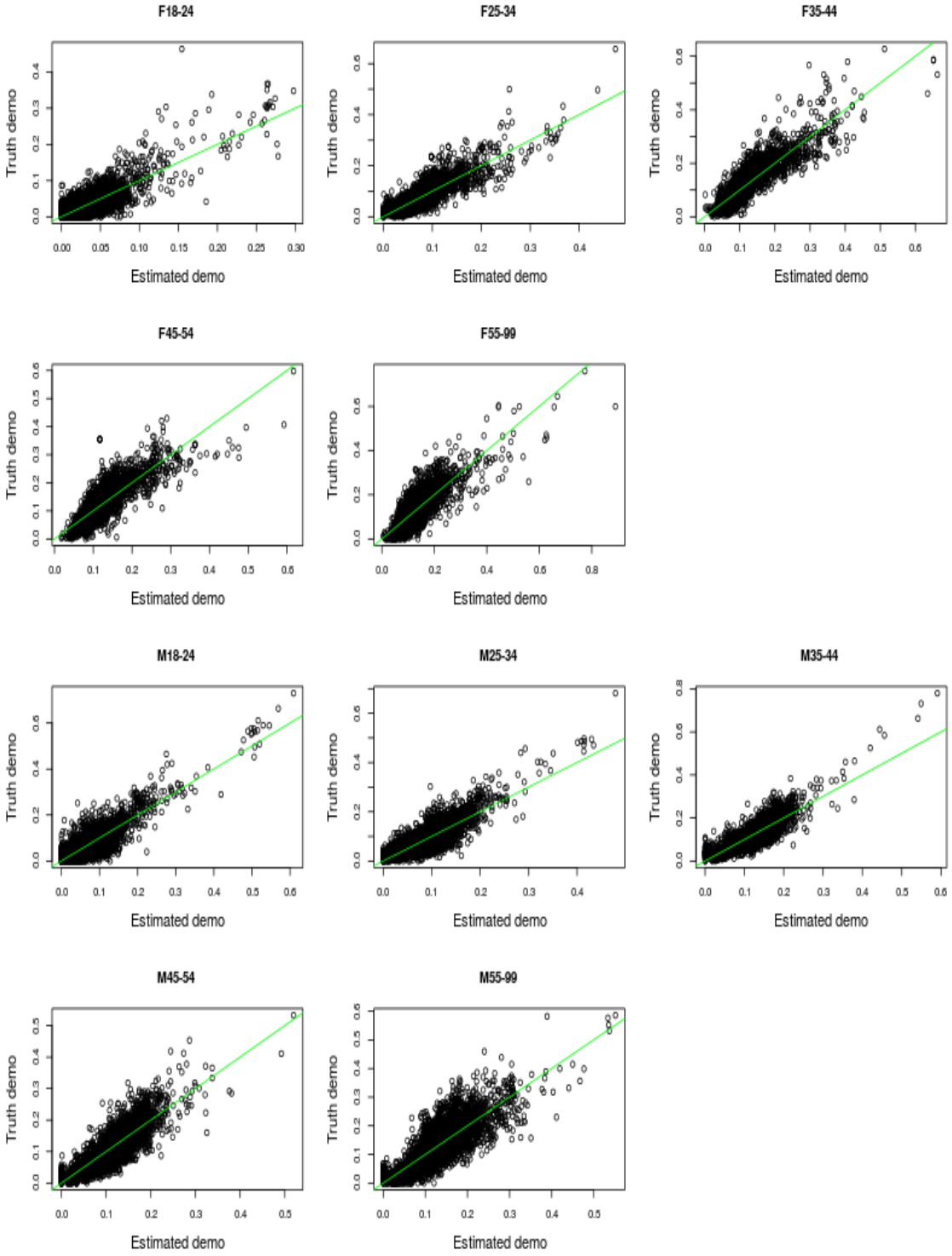

Figure 8: Demographic performance of the **desktop cookie-correction model**. For each demo group, the scatter plot compares the proportion of cookies in a campaign observed from panel data (truth demo in y-axis) to that estimated based on our cookie-correction model (estimated demo in x-axis). The green line marks the identity line.

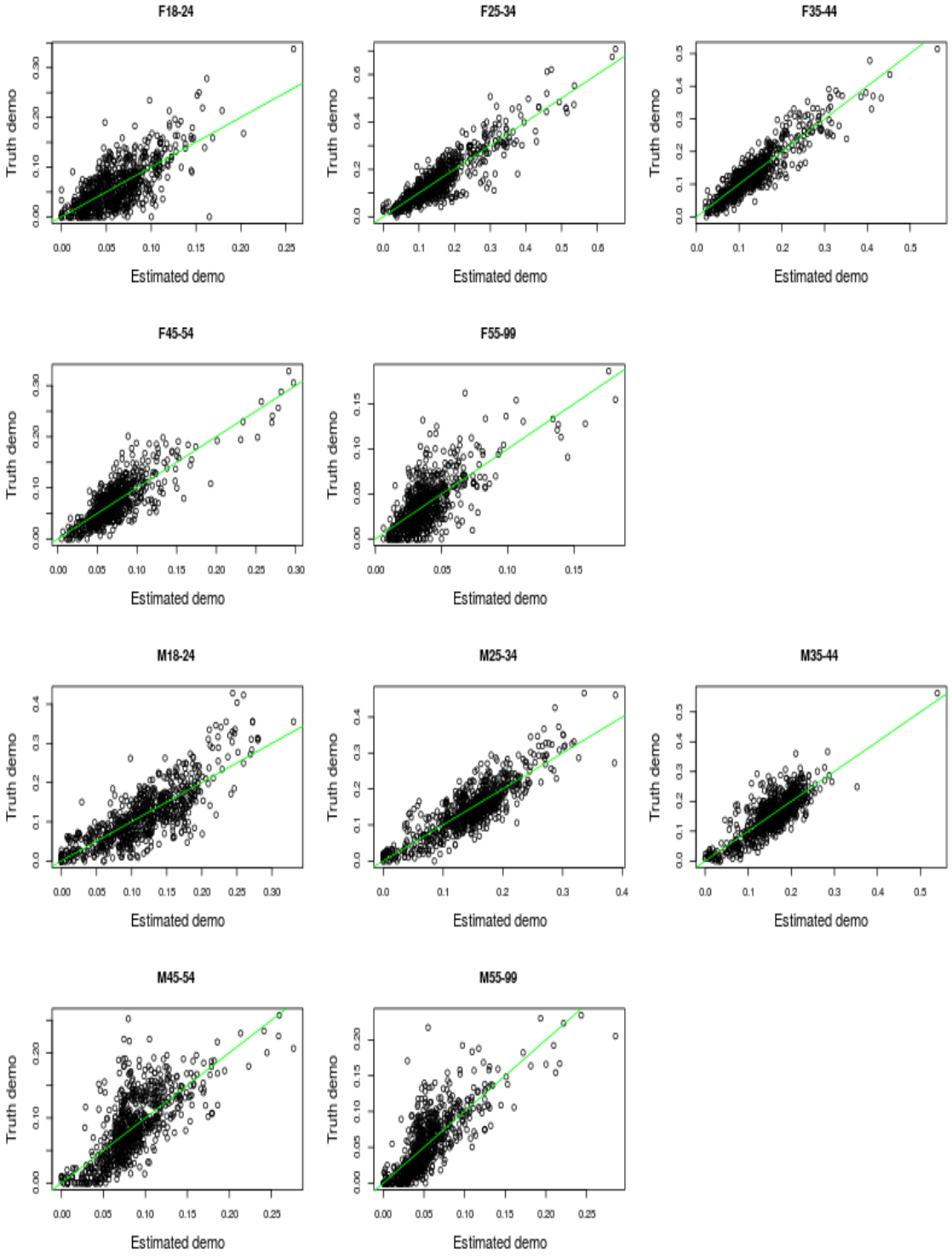

Figure 9: Demographic performance of the **smartphone cookie-correction model**. For each demo group, the scatter plot compares the proportion of cookies in a campaign observed from panel data (truth demo in y-axis) to that estimated based on our cookie-correction model (estimated demo in x-axis). The green line marks the identity line.

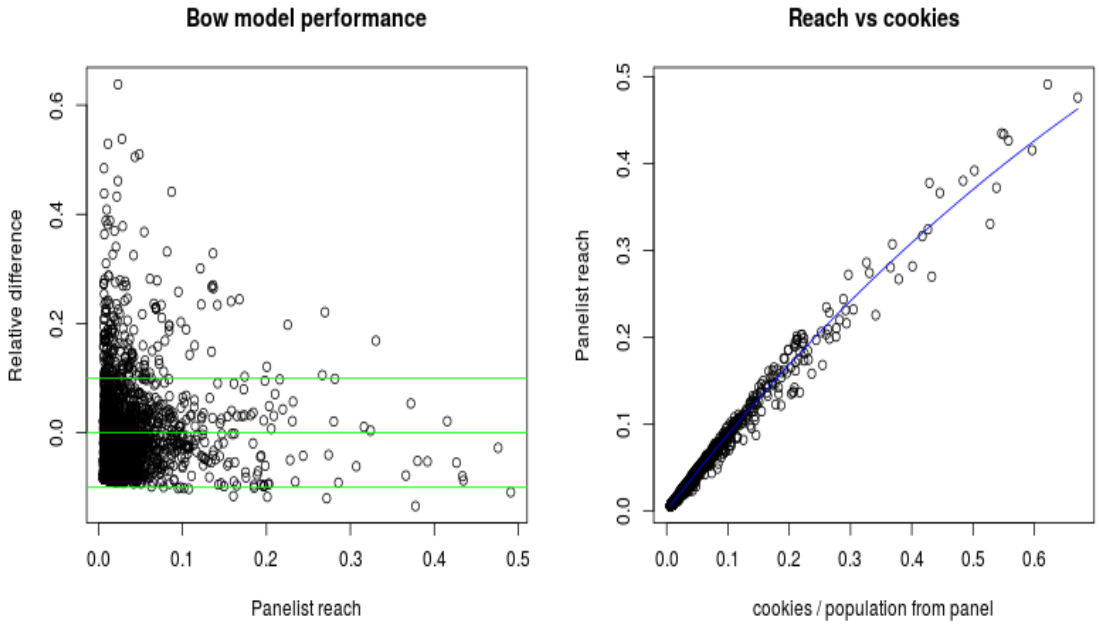

Figure 10: Desktop Reach performance of the Dirac Bow model with $\kappa_0 = 0.92$. The left plot shows the relative difference between the truth (observed reach from panel data) and the model estimate (y-axis) vs. the truth (x-axis) for campaigns. The horizontal lines mark zero and $\pm 10\%$ relative differences. The right plot is the panel reach (i.e. the number of people reached by a campaign divided by total population) vs. the normalized cookie reach (i.e. the number of cookies divided by total population). The circles represent the panal reach for a campaign. The smoothed line is the reach prediction by the Dirac Bow model.

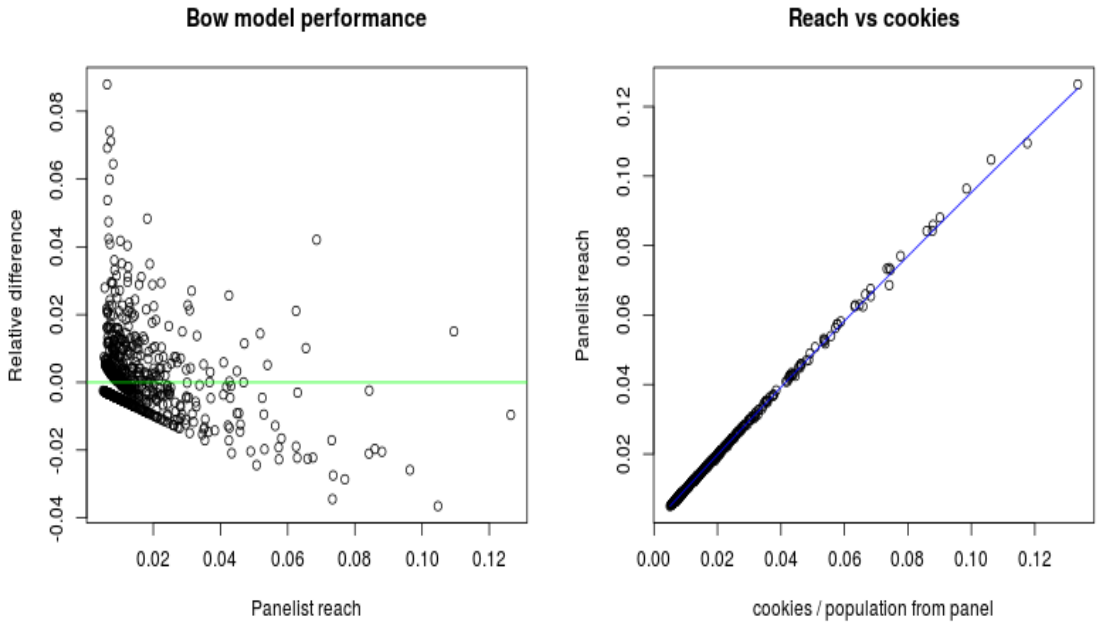

Figure 11: Smartphone reach performance of the Dirac Bow model with $\kappa_0 = 1.00$. The left plot shows the relative difference between the truth (observed reach from panel data) and the model estimate (y-axis) vs. the truth (x-axis) for campaigns. The horizontal lines mark zero and $\pm 10\%$ relative differences. The right plot is the panel reach (i.e. the number of people reached by a campaign divided by total population) vs. the normalized cookie reach (i.e. the number of cookies divided by total population). The circles represent the panal reach for a campaign. The smoothed line is the reach prediction by the Dirac Bow model.

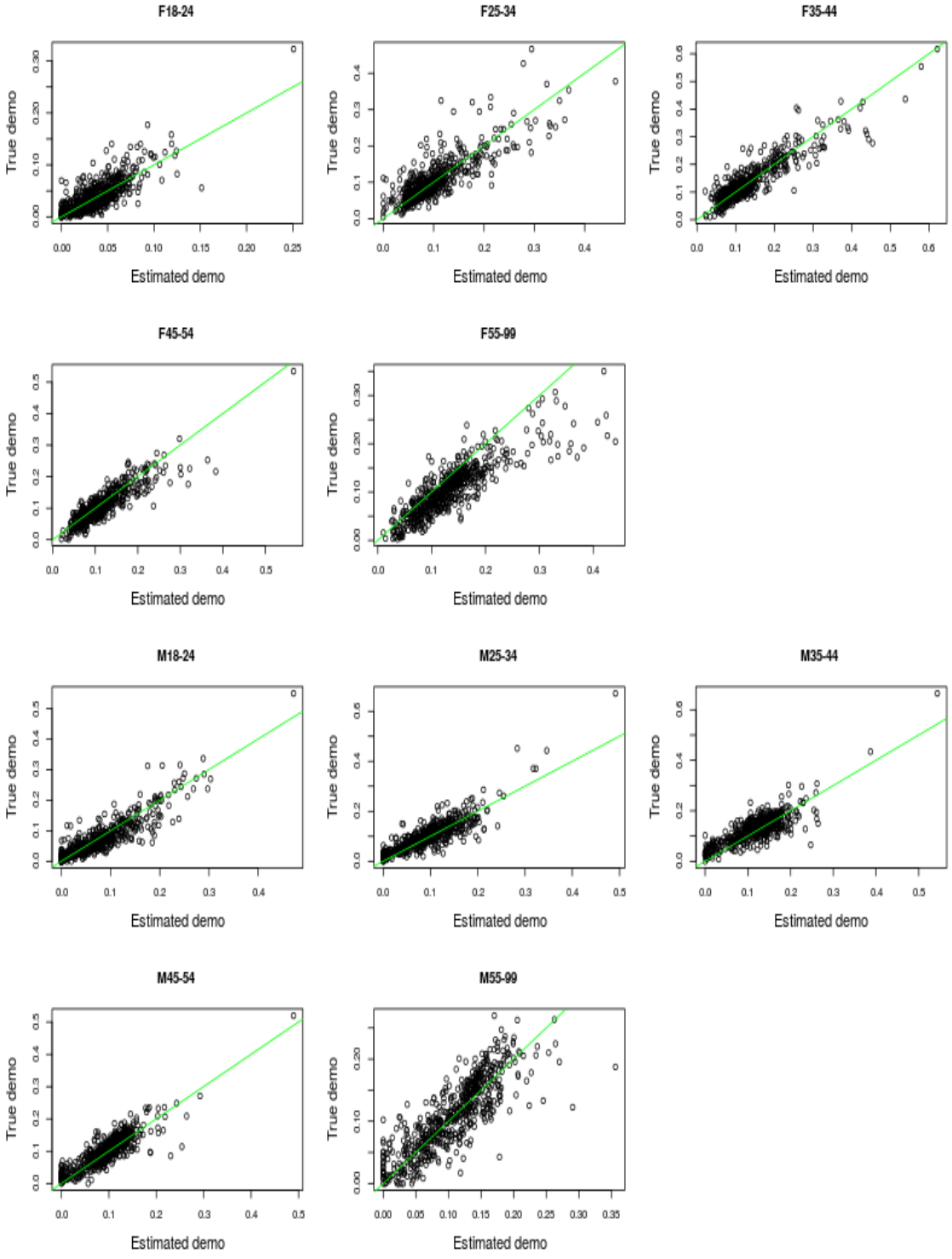

Figure 12: Demographic performance of the **independence cross-device model** for people demographic decomposition using cross-device campaigns.

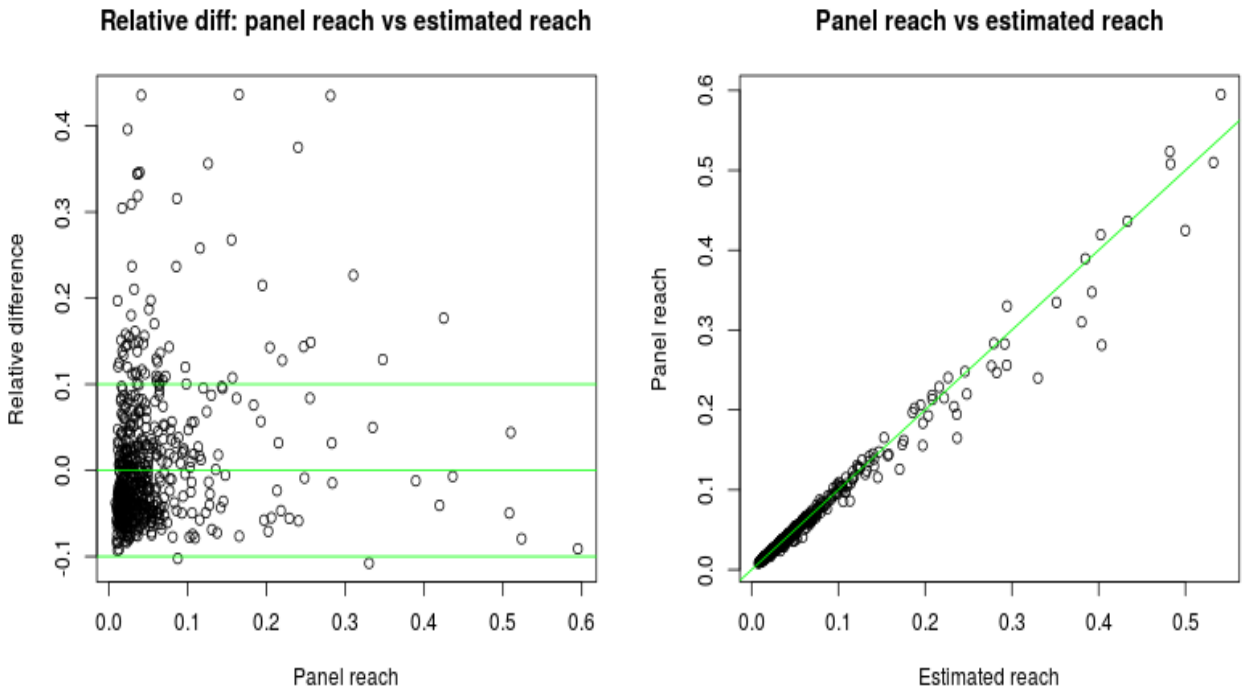

Figure 13: Reach performance for cross-device people using the **independence cross-device model**. The left panel shows the relative difference between truth (observed reach from the panel data) and its estimate. The right panel shows the truth vs. estimated reach.

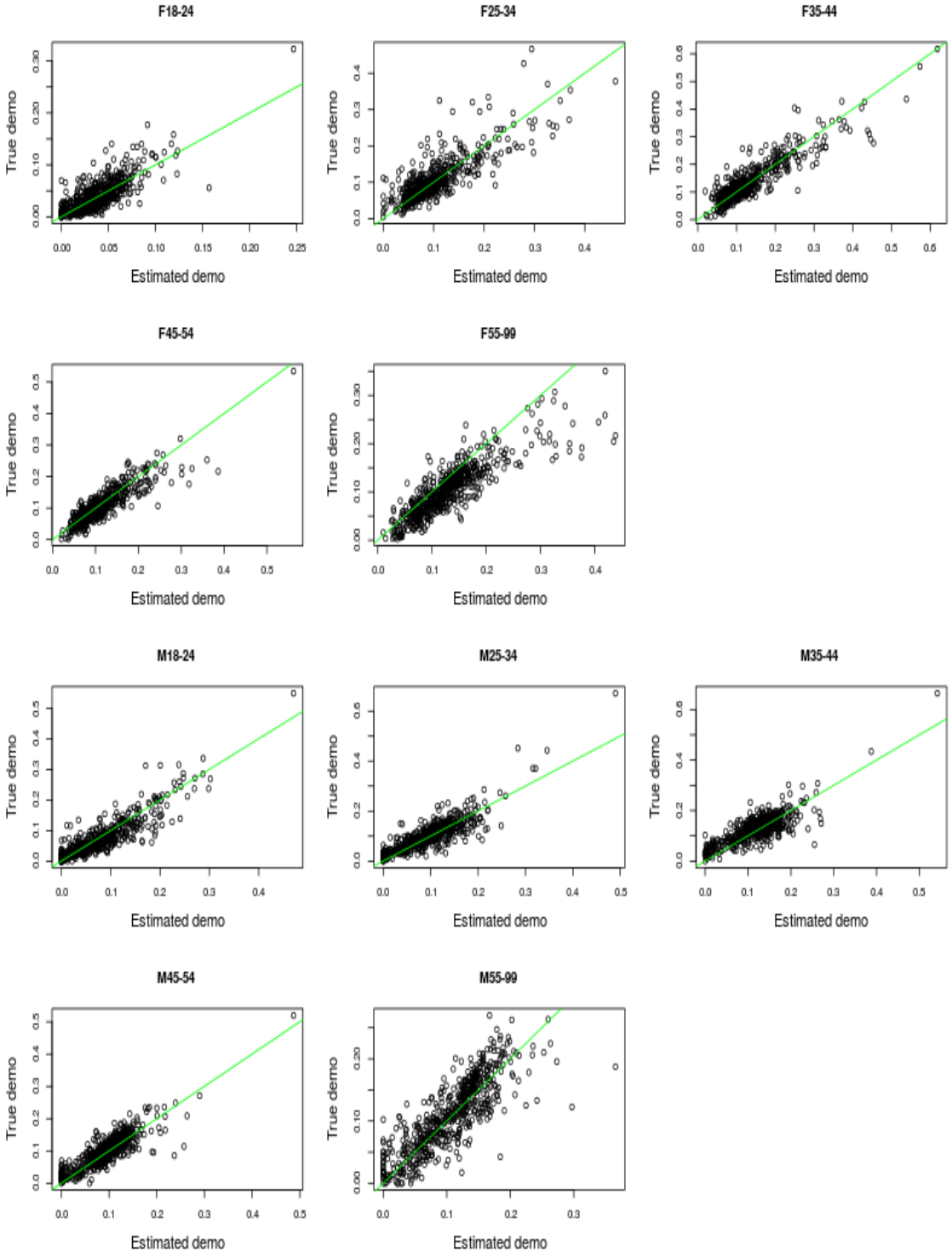

Figure 14: Demographic performance of **Dirac mixture model** for people demographic decomposition using cross-device campaigns.

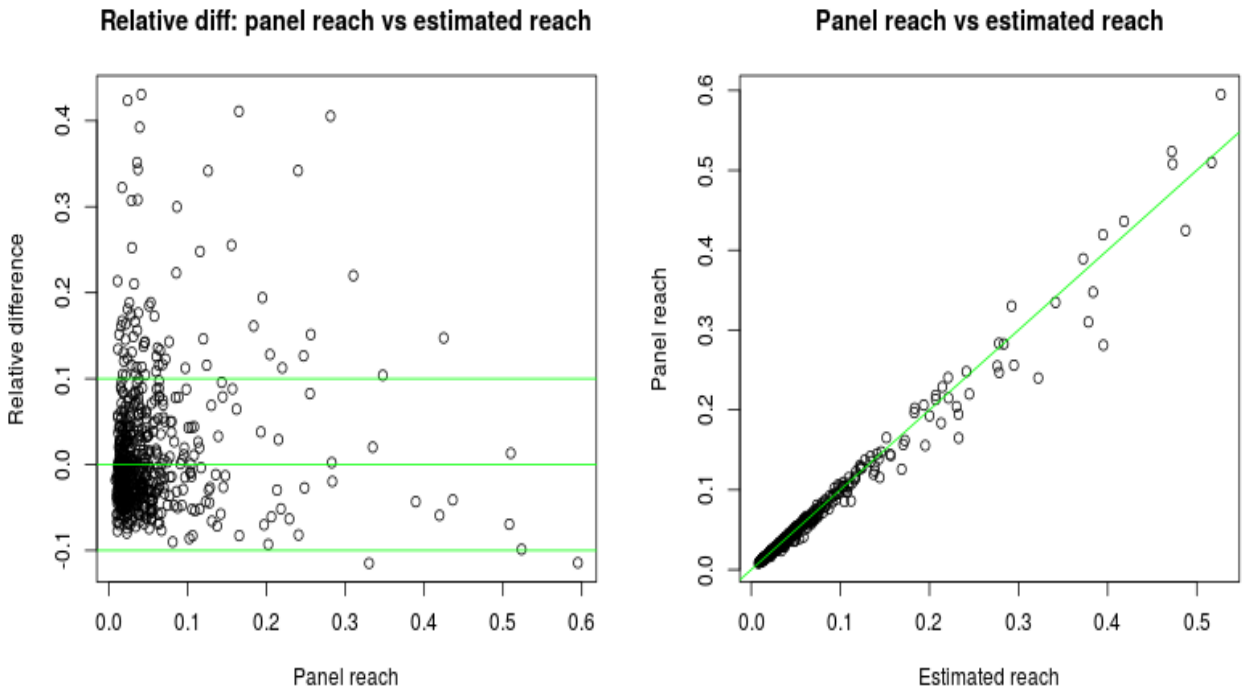

Figure 15: Reach performance for cross-device people using the **Dirac mixture model**. The left panel shows the relative difference between truth (observed reach from panel data) and its estimate. The right panel shows the truth vs. estimated reach.