`rrLogit`: Multinomial Logistic Regression with Randomized Response

Joseph L. Schafer*

February 22, 2024

Abstract

The R package `rrLogit` provides tools for analyzing categorical variables that have been perturbed by the post-randomization method (PRAM) for statistical disclosure control. The primary function `rrLogit()` fits a baseline-category logistic model to predict the noisy variable from one or more non-noisy ones whose relationships to the noisy variable may be of interest. Using this predictive model, the user generates multiple imputations of the unseen true responses. Each version of the imputed data is analyzed in a standard fashion as if there were no added noise, and the results are combined using Rubin's rules for multiple-imputation inference to yield measures of uncertainty that account for the noise infusion. Model-fitting procedures include mode-finding algorithms (EM, Newton-Raphson) and Bayesian posterior simulation by Markov chain Monte Carlo. Additional methods are provided for extracting fitted values and residuals, assessing model fit, and generating posterior predictions. The package includes example datasets and simple utilities for noise infusion, which may be used by statistical agencies to explore potential uses of PRAM and the tradeoffs between data utility and privacy protection.

Keywords: multiple imputation, post-randomization method, randomized response, response error, statistical disclosure limitation.

*Office of the Associate Director for Research and Methodology, United States Census Bureau, Washington, DC 20233, `joseph.l.schafer@census.gov`. This article is intended to inform interested parties of ongoing research and to encourage discussion. Views expressed are those of the author and not necessarily those of the U.S. Census Bureau.

Contents

1 Introduction

The R package `rrLogit` helps users to analyze categorical data that have been perturbed with random noise by the post-randomization method (PRAM). PRAM, a technique for statistical disclosure control developed by Kooiman et al. (1997) and Gouweleeuw et al. (1998), replaces some of the values of categorical variables in a microdata file with new values according to probabilistic mechanism implemented by a statistical agency before the data are released. PRAM is typically applied to key variables (e.g., age, race or ethnicity) that could be used to identify individuals or units represented in the data file. Depending on the mechanism, PRAM may alter the statistical properties of the data in three ways: (a) Distributions of key variables that have been PRAMed may look substantially different from their original versions. (b) Relationships between these key variables and other variables in the dataset may be distorted. (c) The additional noise added by PRAM, if not taken into account by the analyst, may cause levels of uncertainty to be understated; standard errors may be too small, and evidence against null hypotheses may appear too convincing.

Functions in this package address concerns (a)–(c) at the same time. With `rrLogit`, a user fits models to the PRAMed dataset, supplying information on the probabilistic noise mechanism that was applied by the agency. Coefficients from a fitted model are consistent estimates of what we would see if we could apply the same model to the unPRAMed data in the full population, and standard errors from the fitted model account for both the ordinary sampling variability and the extra noise added by PRAM. When a suitable model has been determined, `rrLogit` can generate multiple versions of the original, unaltered data within the framework of multiple imputation (Rubin, 1987). Each imputed dataset represents a plausible version of the unPRAMed sample, and the variation among them reflects the extra uncertainty introduced by PRAM. Analyzing each imputed dataset and combining the results with Rubin's rules for repeated-imputation inference produces estimates and measures of uncertainty for a wide variety of population quantities.

`rrLogit` may also be applied to data on sensitive topics collected by randomized response (RR) (Warner, 1965; Chaudhuri, 2010; Blair et al., 2015a; Chaudhuri et al., 2016). In RR, outcomes of coin flips or other random events are used to

channel participants into different response options. For example, a woman may be instructed to roll a six-sided die. If the result is 1, 2, 3, or 4, she is to answer the question, "Have you *ever* had an abortion?", but if the result is 5 or 6, she is to answer the opposite question, "Have you *never* had an abortion?" Because only the participant observes the outcome of the roll, her true history remains hidden from the data collectors regardless of whether she answers "Yes" or "No." In RR, unlike PRAM, the unperturbed response is unavailable to the agency, and the noise mechanism must be fully determined before data collection begins. From a data analyst's perspective, however, the two situations are virtually identical; techniques for analyzing PRAMed data may be applied seamlessly to RR.[1]

PRAM and RR are also related to problems of misclassification or response error (RE) in surveys, censuses and administrative lists (Kuha and Skinner, 1997). As with PRAM and RR, RE may distort the distribution of a categorical variable and its relationships to other variables. In typical RE scenarios, however, the error mechanisms are unknown and must be estimated, e.g. by linking to additional data sources or conducting validation studies to uncover true values for some units. Examples of RE methodology for categorical variables are given by Chen (1979), Chua and Fuller (1987), Greenland (1988), Kuha and Skinner (1997), Magnac and Visser (1999) and De Waal et al. (2019). In general, valid inferences with RE will require that uncertainty about the error mechanism be taken into account. The `rrLogit` package was not specifically designed for RE, because its functions presume the mechanism is known. However, it may be possible to apply `rrLogit` to RE problems within Bayesian framework, repeatedly conditioning on simulated values of error-mechanism parameters drawn from a posterior distribution. For extended discussion on the similarities and differences among PRAM, RR and RE, see van den Hout and van der Heijden (2007).

With this release of `rrLogit`, we focus on situations where a single key variable has been infused with noise.[2] We also suppose that there may be additional categorical or continuous covariates that do not have added noise, and whose relationships to the key variable may be of interest. The main function in this package, `rrLogit()`, fits a multinomial logistic regression (Agresti, 2013) to predict the key variable from the covariates. To fit this model, the user supplies the noisy key variable, the covariates, and a matrix of probabilities that describes the noise-infusion mechanism. The default model-fitting method in `rrLogit()` is an expectation-maximization (EM) algorithm, a generalization of one previously described by several authors for the case where the key variable is binary (Van den Hout, 1999;

[1] The `rr` in `rrLogit` is a nod to the extensive literature on randomized response and to existing software packages developed for that context, all of whose names contain `rr` or `RR`.

[2] This assumption is not as limiting as it may initially seem. If multiple variables have been PRAMed, we may recode them as a single variable whose categories consist of all possible combinations of the categories of the individual variables. We discuss this further in Section 2.4.

Woo and Slavković, 2012; Blair et al., 2015a). Supporting functions are provided for extracting fitted values, computing residuals, and comparing the fit of alternative models. The `rrLogit()` function also implements Markov chain Monte Carlo (MCMC) procedures for simulating draws from a Bayesian posterior distribution for the model coefficients. MCMC can also compute posterior predictions and generate multiple imputations of the unseen true key variable, which expands the range of possible analyses to situations where the key variable later assumes the role of a predictor for one or more covariates.

`rrLogit` appears to be the first package specifically intended for analyzing data from PRAM. Functions for handling variables perturbed by RR are found in the R packages `rr` (Blair et al., 2015b), `RRreg`, (Heck and Moshagen, 2021), `GLMMRR` (Fox et al., 2021) and the Stata module RRLOGIT (Jann, 2005). Those programs assume that the noise-infused variable is binary, but `rrLogit` allows an arbitrary number of response categories.

Although `rrLogit` was designed primarily for data users, researchers at statistical agencies may also find it useful for exploring potential uses of PRAM and the assessing the impact of varying levels and patterns of noise on the utility of data products before they are released. To that end, `rrLogit` includes a simple utility for infusing categorical variables with noise using R's internal pseudorandom number generator. A more extensive toolkit for experimenting with PRAM and other methods of statistical disclosure control is provided by the R package `sdcMicro` (Templ et al., 2015; Templ, 2017).

In Section 2 of this article, we introduce our notation and review some basic concepts of PRAM. In Section 3, we describe multinomial logistic regression in the conventional setting where the categorical response variable does not have added noise. Readers who are already familiar with multinomial modeling may still find it helpful to skim that section to understand the basic features of the `rrLogit()` function, for example, how it expects input data to be provided. In Section 4, we begin to apply the function to noise-infused variables. Bayesian features of `rrLogit()` are described in Section 5, and topics related to multiple imputation are covered in Section 6.

2 Overview of the post-randomization method

2.1 PRAM in the context of statistical disclosure control

PRAM is one of many available techniques for statistical disclosure control (SDC), also known as statistical disclosure limitation or disclosure avoidance. SDC reduces the chance that individuals or units represented in published data files and

summary statistics can be pinpointed and linked to external information, even after direct identifiers (e.g., names, addresses, Social Security Numbers) have been removed. For overviews of SDC, see Matthews and Harel (2011), Duncan et al. (2011), Willenborg and De Waal (2012), and Hundepool et al. (2012). In practice, an agency implementing SDC must strike a balance between data utility and disclosure risk (Karr et al., 2006; Cox et al., 2011). Risk of disclosure can be described in many different ways; some measures specifically related to categorical microdata are described by Skinner and Elliot (2002), Reiter (2005), Shlomo and Skinner (2010), and Zhang and Nayak (2021).

A major SDC paradigm that has been growing in popularity is differential privacy (DP) (Dwork, 2006; Dwork et al., 2014). DP provides mathematically rigorous definitions of risk, strong privacy guarantees, and general-purpose prescriptions on how to achieve them by infusing data with random noise. DP also allows the statistical agency releasing the data to be highly transparent, fully disclosing all details of the methodology (apart from the actual random numbers used) without incurring additional disclosure risk. Notably, the U.S. Census Bureau adopted DP for data products published from the 2020 Decennial Census (Abowd, 2018; Hawes, 2020; Abowd et al., 2022). DP implementations are either global or local. In global DP, the agency stores confidential data and releases tabulated summaries with noise added to the tabulations. In local DP, the agency stores a protected version of the microdata to which noise has already been added and releases portions or summaries of that database. RR and PRAM naturally align with local DP. Applications of PRAM to satisfy local DP criteria are discussed by Wang et al. (2016), Ayed et al. (2020), and Nayak (2021).

2.2 The perturbation matrix

Consider a categorical variable Y taking possible values $1, 2, \ldots, C$. Suppose the distribution of Y in the population is

$$Y \sim \text{Categorical}(\boldsymbol{\pi}),$$

where $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_C)^\top$, $\pi_y = P(Y = y)$, and the superscript $^\top$ denotes transpose. (Throughout this document, vectors, matrices and arrays are displayed in boldface, scalars are shown in lightface, and where it matters, all vectors are taken to be columns.) With PRAM, the values of Y in a dataset are kept confidential and withheld from data users. Instead, the agency releases a perturbed version Y^* that has been randomly generated from Y. The probabilistic relationship between Y

and Y^* is described by the $C \times C$ perturbation matrix

$$\boldsymbol{T} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1C} \\ t_{21} & t_{22} & \cdots & t_{2C} \\ \vdots & \vdots & \ddots & \vdots \\ t_{C1} & t_{C2} & \cdots & t_{CC} \end{bmatrix},$$

where $t_{rs} = P(Y^* = r \,|\, Y = s)$. Note that the rows of this matrix refer to the levels of the noisy variable Y^*, and its columns refer to levels of the underlying true variable Y. As values on the diagonal of $\boldsymbol{T}$ become smaller, amounts of perturbation become larger. We will assume that $\boldsymbol{T}$ has full rank; if $\boldsymbol{T}$ were singular, then it would not be possible to identify the distribution of Y from the distribution of Y^*. $\boldsymbol{T}$ is guaranteed to have full rank if every element on the diagonal t_{rr} is greater than $1/2$, a condition known as strict diagonal dominance (Horn and Johnson, 2013, p. 352). That condition is not necessary, however, and many non-singular examples exist with smaller diagonal entries.

In the literature on PRAM, $\boldsymbol{T}$ has occasionally been called a transition matrix, because it resembles the transition probability matrix of a Markov chain. It has also been called a distortion matrix. Depending on how the matrix is chosen, however, it may not actually distort the distribution of the variable; it is possible to have $Y^* \sim$ Categorical($\boldsymbol{\pi}$) even when $\boldsymbol{T} \neq \boldsymbol{I}$ ($\boldsymbol{I}$ denotes the identify matrix). $\boldsymbol{T}$ As long as $\boldsymbol{T} \neq \boldsymbol{I}$, Y^* will sometimes differ from Y, and summaries computed from the perturbed data may not exactly reproduce those from the original data.

PRAM was originally proposed by Kooiman et al. (1997) and Gouweleeuw et al. (1998) who soon recognized its similarity to RR. In PRAM, perturbation is carried out on each unit or record in a dataset independently of all other records; if Y_i and Y_i^* denote the values of the true and noisy variables for individual i in a sample of size N, then Y_i^* is simulated from Y_i independently for $i = 1, \ldots, N$. This may create inconsistencies if the data file has a nested or hierarchical structure where units are grouped into clusters and certain logical relationships must be satisfied among the units within a cluster. For example, persons may be nested within households, and no person within a household may have more than one spouse. One possible solution to this problem is to redefine the variable to be PRAMed as an attribute of the cluster rather than the lower-level unit; for discussion on this point, see de Wolf et al. (1998).

2.3 Moment estimators

From the law of total probability $P(Y^* = r) = \sum_{s=1}^{C} P(Y^* = r \,|\, Y = s)\, P(Y = s)$, it immediately follows that the distribution of the noisy variable is

$$Y^* \sim \text{Categorical}(\boldsymbol{\pi}^*), \quad \boldsymbol{\pi}^* = \boldsymbol{T}\boldsymbol{\pi}.$$

If $\boldsymbol{f}^* = (f_1^*, f_2^*, \ldots, f_C^*)^\top$ denotes the tabulated frequencies of $Y^* = y$ for $y = 1, \ldots, C$ and $\hat{\boldsymbol{\pi}}^* = \boldsymbol{f}^*/N$ denotes the sample proportions,[3] a simple method-of-moments (MOM) estimate for $\boldsymbol{\pi}$ is

$$\tilde{\boldsymbol{\pi}} = \boldsymbol{T}^{-1}\hat{\boldsymbol{\pi}}^*. \tag{1}$$

The MOM estimate is unbiased for $\boldsymbol{\pi}$, and an unbiased estimate of its covariance matrix is

$$\hat{V}(\tilde{\boldsymbol{\pi}}) = \frac{1}{(N-1)}\, \boldsymbol{T}^{-1} \left[\text{Diag}(\hat{\boldsymbol{\pi}}^*) - \hat{\boldsymbol{\pi}}^*(\hat{\boldsymbol{\pi}}^*)^\top \right] \left(\boldsymbol{T}^{-1}\right)^\top. \tag{2}$$

The true covariance matrix for $\tilde{\boldsymbol{\pi}}$ can be decomposed as $V(\tilde{\boldsymbol{\pi}}) = \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2$, where

$$\boldsymbol{\Sigma}_1 = \frac{1}{N} \left[\text{Diag}(\boldsymbol{\pi}) - \boldsymbol{\pi}\boldsymbol{\pi}^\top \right] \tag{3}$$

represents the variability due to sampling the true variable from its population, and

$$\boldsymbol{\Sigma}_2 = \frac{1}{N}\, \boldsymbol{T}^{-1} \left[\text{Diag}(\boldsymbol{\pi}^*) - \boldsymbol{T}\,\text{Diag}(\boldsymbol{\pi})\,\boldsymbol{T}^\top \right] \left(\boldsymbol{T}^{-1}\right)^\top \tag{4}$$

represents the extra noise added by PRAM (van den Hout and van der Heijden, 2007).

Although the MOM estimate is unbiased, it may sometimes stray outside the parameter space, producing negative estimated probabilities for one or more categories. This tends to happen more frequently when N is small, when some categories of Y are rare, and when the amounts of added noise are high. Aside from that occasional glitch, it may work well enough to be an adequate solution if we only care about the marginal distribution of Y. The problem, of course, is that we may also want to describe relationships between Y and other variables. Formulas for some simple measures of association that correct for response error are available; for example, Greenland (1988) gives an estimate and standard error for an odds ratio. For more complicated analyses, however, it is helpful to have a set of general-purpose strategies and computational tools to account for the noise that has been added to Y^*.

2.4 PRAM for multiple variables

If we have two key variables, Y_1 and Y_2, with C_1 and C_2 categories, respectively, we may compound them into a single variable Y with categories $y = 1, \ldots, C$ representing the $C = C_1 \times C_2$ possible combinations of Y_1 and Y_2. Suppose we order the

[3] For this discussion, we assume simple random sampling with replacement, which implies that $\boldsymbol{f}^* \sim \text{Multinomial}(N, \boldsymbol{\pi}^*)$. For complex sample designs, we may redefine $\hat{\boldsymbol{\pi}}^*$ as a vector of weighted proportions computed using the survey weights.

categories of this compounded variable so that Y_1 varies more slowly,

$$\begin{aligned} Y = 1 &\Leftrightarrow (Y_1 = 1, Y_2 = 1), \\ Y = 2 &\Leftrightarrow (Y_1 = 1, Y_2 = 2), \\ &\vdots \\ Y = C - 1 &\Leftrightarrow (Y_1 = C_1, Y_2 = C_2 - 1), \\ Y = C &\Leftrightarrow (Y_1 = C_1, Y_2 = C_2). \end{aligned}$$

If we independently apply PRAM to Y_1 and Y_2 using perturbation matrices $\boldsymbol{T}^{(1)}$ and $\boldsymbol{T}^{(2)}$, then

$$P(Y_1^* = a, Y_2^* = b \,|\, Y_1 = c, Y_2 = d) \;=\; t_{ac}^{(1)}\, t_{bd}^{(2)},$$

and the implied $C \times C$ perturbation matrix for Y becomes

$$\boldsymbol{T} \;=\; \begin{bmatrix} t_{11}^{(1)}\,\boldsymbol{T}^{(2)} & t_{12}^{(1)}\,\boldsymbol{T}^{(2)} & \cdots \\ t_{21}^{(1)}\,\boldsymbol{T}^{(2)} & t_{22}^{(1)}\,\boldsymbol{T}^{(2)} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \;=\; \boldsymbol{T}^{(1)} \otimes \boldsymbol{T}^{(2)},$$

where $\otimes$ denotes the Kronecker product. If Y_1 and Y_2 are not PRAMed independently, then $\boldsymbol{T}$ loses this special structure, but we may still regard Y^* as a single PRAMed variable perturbed by $\boldsymbol{T}$.

The extension of this discussion to three or more variables is immediate. Without loss of generality, we may combine any number of PRAMed variables and apply `rrLogit` to the compounded variable Y^*. This approach does have drawbacks. In particular, the class of multinomial logistic models currently implemented in `rrLogit` would not take into account the cross-classified nature of Y^*. As the number of PRAMed variables grows, the total number categories C and the number of parameters needed to relate Y^* to covariates would increase rapidly, leading to overfitted models and numerical instability. One possible solution to this problem would be to extend the model framework to allow constraints consistent with a multivariate response, as in the multinomial logistic models for multivariate categorical responses proposed by Bock (1975).

2.5 Choosing a perturbation matrix

For users of a PRAMed dataset, $\boldsymbol{T}$ has already been determined by the agency supplying the data. Before the data are released, however, researchers within the agency must select a $\boldsymbol{T}$ that balances the needs of data users against the risks of unintentional disclosure.

For any specific population, one can create a PRAMed variable that has the same distribution as its unPRAMed counterpart. That is, if $Y \sim$ Categorical($\boldsymbol{\pi}$), it is possible to find non-identity perturbation matrices that lead to $Y^* \sim$ Categorical($\boldsymbol{\pi}^*$)

with $\boldsymbol{\pi}^* = \boldsymbol{\pi}$. Methods that have this property are called invariant PRAM. To compute a $\boldsymbol{T}$ for invariant PRAM, we need an estimate of $\boldsymbol{\pi}$ based on the observed values of Y in the sample. Kooiman et al. (1997) and Gouweleeuw et al. (1998) suggested the following version. Let $Y_1, \ldots, Y_N$ denote observations of the unPRAMed variable in the dataset, and let $f_y = \sum_{i=1}^{N} I(Y_i = y)$ denote the observed frequency in category y. Denote the smallest of these frequencies by $f_{(\min)} = \min(f_1, \ldots, f_C)$. Elements of the perturbation matrix are

$$t_{rs} = \begin{cases} 1 - (\theta f_{(\min)}/f_r) & \text{if } r = s, \\ \theta f_{(\min)}/((C-1)f_c) & \text{otherwise,} \end{cases}$$

where $\theta \in (0, 1)$ is a tuning parameter controlling the amount of noise. Additional methods for invariant PRAM are described by Marés and Shlomo (2014), Nayak et al. (2018), and Zhang and Nayak (2021). With invariant PRAM, tabulations of Y^* computed by users will be close to those that would be obtained from Y, but other statistical properties will still change; measures of uncertainty will be too small unless the added noise is taken into account, and relationships between the key variable and other variables will be attenuated. To preserve important relationships, invariant PRAM could be performed independently within subsets of the sample defined by covariates. In practice, however, this may lead to unacceptably high risk of disclosure as the number of subsets grows and the sample sizes within them shrink. Invariant PRAM may be attractive when users are unwilling or unable to adjust their analyses for added noise. Marés and Shlomo (2014, Sec. 2) write:

> *Placing the condition of invariance on the transition matrix... releases the users of the protected file of the extra effort to obtain an unbiased estimate of the original data... This is an important property to instill in the protected data since data providers will generally not release the transition matrix.*

However, if the statistical agency does publish $\boldsymbol{T}$, and if tools for analysing PRAMed data are available to users, these arguments in favor of invariant PRAM become less convincing.

Within the framework of differential privacy, Wang et al. (2016) proposed a simple non-invariant method that satisfies local DP conditions. Their perturbation matrix has equal values on the diagonal and off the diagonal,

$$\boldsymbol{T} = \begin{bmatrix} p & \frac{1-p}{C-1} & \cdots & \frac{1-p}{C-1} \\ \frac{1-p}{C-1} & p & \cdots & \frac{1-p}{C-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-p}{C-1} & \frac{1-p}{C-1} & \cdots & p \end{bmatrix}, \tag{5}$$

with the value of p set by

$$p = \left[1 + (C-1)\, e^{-\epsilon}\right]^{-1},$$

where $\epsilon \in (0, \infty)$ is a privacy-loss parameter. More versions of $\boldsymbol{T}$ consistent with DP are discussed by Ayed et al. (2020) and Nayak (2021), and some additional non-DP choices were proposed by de Wolf and van Gelder (2004) and Domingo-Ferrer and Torra (2001).

3 Multinomial logistic regression without added noise

This section provides an overview of multinomial logistic regression in conventional situations where the true response is seen by the analyst. Readers who are already familiar with multinomial modeling should still review this section to learn the basic features of the `rrLogit()` *function.*

3.1 A simple example

Baseline-category logistic regression extends the well known logistic model for binary responses to $C \geq 2$ outcome categories. Before applying this model to noise-infused data, we illustrate it on a small dataset published by Agresti (2013) which has no added noise and no obvious connection to PRAM or RR. We chose this example only because it is compact and easy to understand. In Section 4, we will switch to much larger dataset taken from an actual survey.

The data shown in Table 1 come from a study of the primary food choices of alligators in four Florida lakes. Researchers classified the stomach contents of 219 captured alligators into five categories: Fish (the most common primary food choice), Invertebrate (snails, insects, crayfish, etc.), Reptile (turtles, alligators), Bird, and Other (amphibians, plants, household pets, stones, and other debris). The alligators were also classified by Lake, Sex and Size. We regard food choice as the outcome variable and Lake, Sex and Size as potential predictors.

Denote the primary food choice for alligator i ($i = 1, \ldots, 209$) by Y_i, which we regard as

$$Y_i \sim \text{Categorical}(\boldsymbol{\pi}_i),$$

where $\boldsymbol{\pi}_i = (\pi_{i1}, \pi_{i2}, \pi_{i3}, \pi_{i4}, \pi_{i5})^\top$. In this notation, π_{i1} is the probability that the alligator primarily consumes fish; π_{i2} is the probability that it primarily consumes invertebrates; and so on. The subscript i indicates that these probabilities may vary among alligators, and this variation may be at least partly explained by covariates. To relate $\boldsymbol{\pi}_i$ to the covariates, we first select one of the food-choice categories to

Table 1: Frequency table for 219 alligators captured in Florida, classified by lake, sex, size, and primary food choice. Source: Agresti (2013)

			Primary Food Choice				
Lake	*Sex*	*Size*	Fish	Inv.	Rept.	Bird	Other
Hancock	M	small	7	1	0	0	5
		large	4	0	0	1	2
	F	small	16	3	2	2	3
		large	3	0	1	2	3
Oklawaha	M	small	2	2	0	0	1
		large	13	7	6	0	0
	F	small	3	9	1	0	2
		large	0	1	0	1	0
Trafford	M	small	3	7	1	0	1
		large	8	6	6	3	5
	F	small	2	4	1	1	4
		large	0	1	0	0	0
George	M	small	13	10	0	2	2
		large	9	0	0	1	2
	F	small	3	9	1	0	1
		large	8	1	0	0	1

serve as a baseline and define the log-odds of every other category versus that one. The choice of baseline does not affect the model fit; if the baseline were changed, the meaning and values of the regression coefficients would change, but the estimated probabilities $\boldsymbol{\pi}_i$ would remain the same. Selecting category 1 (Fish) as the baseline, we suppose that

$$\begin{aligned}
\eta_{i2} &= \log\left(\frac{\pi_{i2}}{\pi_{i1}}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_2, \\
\eta_{i3} &= \log\left(\frac{\pi_{i3}}{\pi_{i1}}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_3, \\
\eta_{i4} &= \log\left(\frac{\pi_{i4}}{\pi_{i1}}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_4, \\
\eta_{i5} &= \log\left(\frac{\pi_{i5}}{\pi_{i1}}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_5,
\end{aligned}$$

where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^\top$ is a $p \times 1$ vector of known predictors describing alligator i, and $\boldsymbol{\beta}_y = (\beta_{1y}, \ldots, \beta_{py})^\top$, $y = 2, 3, 4, 5$ are vectors of coefficients to be estimated. The first element of $\boldsymbol{x}_i$ is 1, and the remaining elements are dummy indicators or codes that characterize main effects for Lake, Sex, and Size and possibly interactions among them.

More generally, if a response variable Y_i has categories $y = 1, \ldots, C$ and we select b as the baseline, the baseline-category logit model is

$$\eta_{iy} = \log\left(\frac{\pi_{iy}}{\pi_{ib}}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_y \tag{6}$$

for $y = 1, \ldots, C$. The coefficients for the baseline category, $\boldsymbol{\beta}_b$, are constrained to be $\mathbf{0} = (0, \ldots, 0)^\top$. The inverse transformation implied by (6) is

$$\pi_{iy} = \frac{\exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_y)}{\sum_{y'=1}^{C} \exp(\boldsymbol{x}_i^\top \boldsymbol{\beta}_{y'})}. \tag{7}$$

Depending on the context, we may write the full set of coefficients as a $p \times C$ matrix

$$\boldsymbol{B} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_C],$$

which includes the column of zeros, or we may remove the zeros and stack the remaining columns into a vector $\boldsymbol{\beta}$ of length $D = p(C-1)$.

3.2 Data formats

Wide format. Although our model is expressed in terms of the outcome Y_i for a single alligator, Table 1 does not display any individual Y_i values. Rather, it groups

alligators by covariate patterns. Each row of the table represents a different combination of Lake, Sex, and Size, and the frequencies for the response categories appear in different columns. This method of data storage is attractive when the number of distinct covariate patterns is small relative to the overall sample size. A wide-format version of the alligator dataset is included with `rrLogit` as a data frame called `alligatorWide`.

```
> # attach the rrLogit library
> library(rrLogit)
> # display Agresti's allligator data in wide format
> alligatorWide
```

```
       Lake Sex  Size Fish Inv Rept Bird Other
1   Hancock   M small    7   1    0    0     5
2   Hancock   M large    4   0    0    1     2
3   Hancock   F small   16   3    2    2     3
4   Hancock   F large    3   0    1    2     3
5  Oklawaha   M small    2   2    0    0     1
6  Oklawaha   M large   13   7    6    0     0
7  Oklawaha   F small    3   9    1    0     2
8  Oklawaha   F large    0   1    0    1     0
9  Trafford   M small    3   7    1    0     1
10 Trafford   M large    8   6    6    3     5
11 Trafford   F small    2   4    1    1     4
12 Trafford   F large    0   1    0    0     0
13   George   M small   13  10    0    2     2
14   George   M large    9   0    0    1     2
15   George   F small    3   9    1    0     1
16   George   F large    8   1    0    0     1
```

Narrow format with frequencies. In narrow format, each row represents a group of units that are identical with respect to the covariates and the response. The response is a single categorical variable (a factor in R) appearing in its own column, and the frequencies are stored as a numeric variable in another column. The narrow-format version of the alligator dataset is called `alligatorNarrow`.

```
> # display a few rows of alligator data in narrow format with frequencies
> head(alligatorNarrow)
```

```
    Lake Sex  Size  Food Freq
1 Hancock   M small  Fish    7
2 Hancock   M small   Inv    1
3 Hancock   M small  Rept    0
4 Hancock   M small  Bird    0
5 Hancock   M small Other    5
6 Hancock   M large  Fish    4
```

```
> # display the total number of alligators
> sum(alligatorNarrow$Freq)
```

```
[1] 219
```

When using narrow format, you may omit lines with frequencies of zero, because these contribute nothing to the loglikelihood. Whether those lines are included or not, you will get the same results.

Microdata. With microdata, each row represents a single observational unit. As with narrow format, the response is a factor appearing in its own column, but a frequency variable is not needed, because the frequency associated with each row is assumed to be one. This version of the dataset is called `alligatorMicro`.

```
> # display a few rows of alligator data in micro format
> head(alligatorMicro)

     Lake Sex  Size Food
1 Hancock   M small Fish
2 Hancock   M small Fish
3 Hancock   M small Fish
4 Hancock   M small Fish
5 Hancock   M small Fish
6 Hancock   M small Fish

> # display the total number of alligators
> NROW(alligatorMicro)

[1] 219
```

Survey weights. In microdata format, you have the option of supplying a numeric variable containing survey weights. Survey weights play a role similar to frequencies in the model-fitting process, but special procedures (e.g., linearization methods or refitting with replicate weights) must be applied afterward to compute standard errors that take into account the complex sample design (Lumley, 2010). Those procedures have not yet been implemented in `rrLogit()`, but the current version does accept survey weights. If weights are supplied, they are automatically scaled to sum to the overall sample size. This scaling has no effect on parameter estimates.

3.3 Fitting a model

Model formula. When calling the function `rrLogit()` to fit a new model, the first argument must be a model formula. The formula takes a different form for data in wide format versus data in narrow or microdata formats. With wide-format data, the left-hand side of the model formula (the part before the $\sim$) should be a matrix of frequencies. In other words, the left-hand side should list the frequency variables for response categories $1, \ldots, C$, separated by commas and enclosed by `cbind()`. The right-hand side should list the predictors, separated by the operators `+`, `*`, or `:` in the standard modeling notation for R developed by Chambers and Hastie (1992), and as described in the help pages `?formula` and `?lm`. For example, suppose that there are $C = 3$ response categories whose frequency variables are named `f1`, `f2` and `f3`. The model formula `cbind(f1,f2,f3)` $\sim$ `x1 + x2*x3` specifies a model with main effects for `x1`, `x2`, and `x3` and the `x2:x3` interaction. In the example below, we fit a model to `alligatorWide` with main effects for Lake, Sex, and Size.

```
> # example in wide format
> fitA <- rrLogit( cbind(Fish, Inv, Rept, Bird, Other) ~ Lake + Sex + Size,
+    data=alligatorWide)
```

If the data are arranged in narrow format with frequencies, the model formula should have the response variable on the left-hand side, and the variable containing the frequencies should be supplied through the argument `freq`.

```
> # same model as before, but in narrow format with frequencies
> fitB <- rrLogit( Food ~ Lake + Sex + Size,
+    data=alligatorNarrow, freq=Freq)
```

For microdata, we again have the response variable on the left-hand side of the formula, but the argument `freq` is omitted.

```
> # same model as before, but with microdata
> fitC <- rrLogit( Food ~ Lake + Sex + Size,
+    data=alligatorMicro)
```

Model matrix. As with other regression procedures in R, the numeric predictor vectors $\boldsymbol{x}_i$ are automatically generated from the right-hand side of model formula. A constant is included unless the formula contains `0` or `-1`. Factor variables are automatically expressed as dummy codes, effect codes, etc. as determined by the `contrasts` attribute of each factor or by the global setting `options("contrasts")`. To see these numeric predictors, apply `model.matrix()` to the result from `rrLogit()`.

```
> # display the model matrix from the wide-format model fit
> model.matrix(fitA)
```

	(Intercept)	LakeOklawaha	LakeTrafford	LakeGeorge	SexF	Sizelarge
1	1	0	0	0	0	0
2	1	0	0	0	0	1
3	1	0	0	0	1	0
4	1	0	0	0	1	1
5	1	1	0	0	0	0
6	1	1	0	0	0	1
7	1	1	0	0	1	0
8	1	1	0	0	1	1
9	1	0	1	0	0	0
10	1	0	1	0	0	1
11	1	0	1	0	1	0
12	1	0	1	0	1	1
13	1	0	0	1	0	0
14	1	0	0	1	0	1
15	1	0	0	1	1	0
16	1	0	0	1	1	1

The matrix returned by `model.matrix()` has one row of predictors $\boldsymbol{x}_i^\top$ for each row of the input dataset.

Choosing the baseline. By default, `rrLogit()` uses the first response category (in this example, Fish) as the baseline. The `baseline` argument allows you to select the baseline by providing an integer or a character string. For example, to change the baseline to Other, you could say `baseline=5` or `baseline="Other"`.

Summarizing results. A call to `rrLogit()` produces an object of class `"rrLogit"`, a special kind of list that holds summaries of the data and results from the model-fitting procedure. Components of this object may be examined directly, but users are strongly encouraged to extract information using `summary()` and other methods specifically provided for that purpose.

```
> # display results from the wide-format fit
> summary(fitA, showCoef=TRUE)

cbind(Fish, Inv, Rept, Bird, Other) ~ Lake + Sex + Size
Saturated option: FALSE
          Prior: none

               Data format: wide
      Frequencies supplied: TRUE
     Rows of supplied data: 16
  Total N in supplied data: 219
Distinct covariate patterns: 16
   Empty covariate patterns: 0

Response categories: Fish Inv Rept Bird Other
  Baseline category: Fish

Assumed perturbation matrix: Identity

Number of estimated parameters = 24
           Degrees of freedom = 40

Newton-Raphson procedure
Starting values: default
Converged at iteration 6
Gradient length = 0.000000

  Final logP = -268.9327
Final loglik = -268.9327

Estimated coefficients with Hessian-based SEs
  (dispersion parameter taken to be 1.000000)
, , Response = Fish

             coef SE zstat pval
(Intercept)     0  0   NaN  NaN
LakeOklawaha    0  0   NaN  NaN
LakeTrafford    0  0   NaN  NaN
LakeGeorge      0  0   NaN  NaN
SexF            0  0   NaN  NaN
Sizelarge       0  0   NaN  NaN

, , Response = Inv

               coef     SE zstat   pval
(Intercept)  -2.074 0.6117 -3.39 0.0007
LakeOklawaha  2.694 0.6693  4.02 0.0001
LakeTrafford  2.936 0.6874  4.27 0.0000
```

```
LakeGeorge    1.781 0.6232  2.86 0.0043
SexF          0.463 0.3955  1.17 0.2418
Sizelarge    -1.336 0.4112 -3.25 0.0012

, , Response = Rept

               coef     SE zstat   pval
(Intercept)  -2.9141 0.8856 -3.29 0.0010
LakeOklawaha  1.4008 0.8105  1.73 0.0839
LakeTrafford  1.9316 0.8253  2.34 0.0193
LakeGeorge   -1.1295 1.1928 -0.95 0.3437
SexF          0.6276 0.6853  0.92 0.3598
Sizelarge     0.5570 0.6466  0.86 0.3890

, , Response = Bird

               coef     SE zstat   pval
(Intercept)  -2.4633 0.7739 -3.18 0.0015
LakeOklawaha -1.1256 1.1924 -0.94 0.3452
LakeTrafford  0.6617 0.8461  0.78 0.4341
LakeGeorge   -0.5753 0.7952 -0.72 0.4694
SexF          0.6064 0.6888  0.88 0.3787
Sizelarge     0.7302 0.6523  1.12 0.2629

, , Response = Other

               coef     SE zstat   pval
(Intercept)  -0.9167 0.4782 -1.92 0.0552
LakeOklawaha -0.7405 0.7422 -1.00 0.3184
LakeTrafford  0.7912 0.5879  1.35 0.1784
LakeGeorge   -0.7666 0.5686 -1.35 0.1776
SexF          0.2526 0.4663  0.54 0.5881
Sizelarge    -0.2906 0.4599 -0.63 0.5275
```

The last part of the printed summary displays each of the estimated coefficients $\hat{\beta}_{jy}$ along with its standard error (SE), Wald z-statistic, and p-value for testing the null hypothesis $\beta_{jy} = 0$ against the two-sided alternative $\beta_{jy} \neq 0$. As the number of response categories and covariates grows, this section can become tediously long. The `summary()` method has an optional argument `showCoef` that determines whether this table of coefficients is displayed, with the default setting `showCoef=FALSE`. To extract the coefficients as a $p \times C$ matrix without SEs or test statistics, use `coef()`.

```
> # extract the estimated coefficients as a matrix
> coef(fitA)

             Fish        Inv       Rept       Bird      Other
(Intercept)     0 -2.0744513 -2.9141377 -2.4632747 -0.9167261
LakeOklawaha    0  2.6936942  1.4007966 -1.1256172 -0.7405175
LakeTrafford    0  2.9363342  1.9315865  0.6617240  0.7911874
LakeGeorge      0  1.7805123 -1.1294628 -0.5752664 -0.7665752
SexF            0  0.4629629  0.6275587  0.6064286  0.2525695
Sizelarge       0 -1.3362610  0.5570360  0.7302394 -0.2905828

> # verify that the estimates are identical regardless of how the
> # input data were formatted
> all.equal( coef(fitA), coef(fitB) )

[1] TRUE
```

```
> all.equal( coef(fitA), coef(fitC) )
[1] TRUE
```

`coef()` can also be used to retrieve the full $D \times D$ covariance matrix for the vectorized $\hat{\boldsymbol{\beta}}$; see `?coef.rrLogit` for details.

Fitting procedure. The `method` argument for `rrLogit()` allows you to select the fitting procedure. For computing maximum-likelihood estimates, the available methods are `"EM"` (expectation-maximization), `"NR"` (Newton-Raphson), and `"FS"` (Fisher scoring), with `"EM"` as the default. Without noise infusion, these three algorithms are identical, so for this example it doesn't matter which one you choose. Details of the fitting procedure are given in A.

Interpreting coefficients. Some of the estimated effects for Lake are large and highly significant. Consider the coefficient for `LakeOklawaha` and `Response = Inv`, with a value of 2.694 $\approx$ 2.7. Because the baseline category is Fish, this coefficient alters the log-odds of Invertebrates versus Fish. The model has dummy indicators for Oklawaha, Trafford, and George, but not for Hancock, so Hancock is the reference lake. Therefore, our model estimates that, as we move from Lake Hancock to Lake Oklawaha, the log-odds of an alligator choosing Invertebrates over Fish increases by 2.7. In terms of odds, moving from Hancock to Oklawaha multiplies the odds of Invertebrates versus Fish by $\exp(2.7) \approx 15$. One of the effects for Size (`Response = Inv`) is large and significant, but all three effects for Sex are small and insignificant, which suggests that we may be able to remove Sex without harming the fit.

3.4 Fitted values, residuals, predictions

Grouping by covariate patterns. Regardless of how the input data are formatted, `rrLogit()` processes the data prior to model fitting, grouping the lines of the data frame by their covariate patterns. Covariate patterns, which we index by $g = 1, \ldots, G$, are the unique combinations of the values of the predictor variables appearing in the dataset. Units within a covariate pattern share the same covariate vector $\boldsymbol{x}_i$ and the same vector of probabilities $\boldsymbol{\pi}_i$, so we will sometimes write these as $\boldsymbol{x}_g$ and $\boldsymbol{\pi}_g$. The grouped data can be expressed as $(\boldsymbol{x}_g, \boldsymbol{f}_g)$, $g = 1, \ldots, G$, where $\boldsymbol{f}_g = (f_{g1}, \ldots, f_{gC})^\top$, and f_{gy} is the frequency of $Y_i = y$ within group g. Regarding the group totals $N_g = \sum_{y=1}^{C} f_{gy}$ as fixed, $\boldsymbol{f}_g$ has a multinomial distribution

$$\boldsymbol{f}_g \,|\, \boldsymbol{\pi}_g \;\sim\; \text{Mult}(N_g, \boldsymbol{\pi}_g)$$

independently for $g = 1, \ldots, G$. The automatic grouping by covariate patterns speeds up the model-fitting procedures quite dramatically, especially when all of

the predictors are categorical. This grouping is also key to interpreting fitted values and residuals, which we discuss next.

Fitted values. Applying `fitted()` to an `rrLogit` object returns a matrix of fitted values with rows corresponding to covariate patterns and columns corresponding to response categories. Fitted values come in three flavors: `type="prob"` (the default), which gives fitted probabilities,

$$\hat{\pi}_{gy} \;=\; \frac{\exp(\boldsymbol{x}_g^\top \hat{\boldsymbol{\beta}}_y)}{\sum_{y'=1}^{C} \exp(\boldsymbol{x}_g^\top \hat{\boldsymbol{\beta}}_{y'})},$$

`type="link"`, which gives fitted logits,

$$\hat{\eta}_{gy} \;=\; \boldsymbol{x}_g^T \hat{\boldsymbol{\beta}}_y,$$

and `type="mean"`, which gives estimated expected frequencies,

$$\hat{\mu}_{gy} \;=\; N_g\, \hat{\pi}_{gy}.$$

```
> # display fitted probabilities from model fit to wide data;
> # results from narrow data and microdata will be the same
> fitted(fitA)
```

```
          Fish        Inv        Rept        Bird      Other
 [1,] 0.6006519 0.07545711 0.032585844 0.051148899 0.24015620
 [2,] 0.6236514 0.02059152 0.059056220 0.110228583 0.18647229
 [3,] 0.5070799 0.10120825 0.051526161 0.079187829 0.26099782
 [4,] 0.5157652 0.02705582 0.091478923 0.167175631 0.19852444
 [5,] 0.3033992 0.56357055 0.066800362 0.008382596 0.05784725
 [6,] 0.4825217 0.23556992 0.185438035 0.027670680 0.06879971
 [7,] 0.2146066 0.63334325 0.088501922 0.010873661 0.05267456
 [8,] 0.3591707 0.27859037 0.258540225 0.037772217 0.06592652
 [9,] 0.2088104 0.49438251 0.078169179 0.034462644 0.18417528
[10,] 0.3050767 0.18984067 0.199347211 0.104506783 0.20122864
[11,] 0.1449086 0.54508886 0.101606803 0.043859046 0.16453672
[12,] 0.2132250 0.21080481 0.260966314 0.133949814 0.18105409
[13,] 0.5008713 0.37331044 0.008782392 0.023994106 0.09304172
[14,] 0.6826673 0.13372796 0.020893605 0.067877589 0.09483358
[15,] 0.3930855 0.46547165 0.012909783 0.034532980 0.09400006
[16,] 0.5781269 0.17992793 0.033141546 0.105416633 0.10338697
```

Covariate patterns are listed in the order in which they first appear in the input data frame. It is often helpful to know the values of the predictor variables associated with the covariate patterns. For this purpose, `fitted()` has an argument `include.predvars` whose default value is `FALSE`. Calling `fitted()` with `fitted()` with `include.predvars=TRUE` returns a data frame that includes the predictor variables.

```
> fitted(fitA, include.predvars=TRUE)

       Lake Sex  Size      Fish        Inv        Rept        Bird      Other
1   Hancock   M small 0.6006519 0.07545711 0.032585844 0.051148899 0.24015620
2   Hancock   M large 0.6236514 0.02059152 0.059056220 0.110228583 0.18647229
3   Hancock   F small 0.5070799 0.10120825 0.051526161 0.079187829 0.26099782
4   Hancock   F large 0.5157652 0.02705582 0.091478923 0.167175631 0.19852444
5  Oklawaha   M small 0.3033992 0.56357055 0.066800362 0.008382596 0.05784725
6  Oklawaha   M large 0.4825217 0.23556992 0.185438035 0.027670680 0.06879971
7  Oklawaha   F small 0.2146066 0.63334325 0.088501922 0.010873661 0.05267456
8  Oklawaha   F large 0.3591707 0.27859037 0.258540225 0.037772217 0.06592652
9  Trafford   M small 0.2088104 0.49438251 0.078169179 0.034462644 0.18417528
10 Trafford   M large 0.3050767 0.18984067 0.199347211 0.104506783 0.20122864
11 Trafford   F small 0.1449086 0.54508886 0.101606803 0.043859046 0.16453672
12 Trafford   F large 0.2132250 0.21080481 0.260966314 0.133949814 0.18105409
13   George   M small 0.5008713 0.37331044 0.008782392 0.023994106 0.09304172
14   George   M large 0.6826673 0.13372796 0.020893605 0.067877589 0.09483358
15   George   F small 0.3930855 0.46547165 0.012909783 0.034532980 0.09400006
16   George   F large 0.5781269 0.17992793 0.033141546 0.105416633 0.10338697
```

For information on how to map the covariate patterns to rows of the input data, see `?fitted.rrLogit`.

Residuals. By default, residuals() computes the Pearson residuals (`type="Pearson"`),

$$R_{gy} = \frac{f_{gy} - \hat{\mu}_{gy}}{\sqrt{\hat{\mu}_{gy}}}. \tag{8}$$

Selecting `type="response"` returns the raw differences $f_{gy} - \hat{\mu}_{gy}$. If the fitted means are not too small, Pearson residuals are roughly comparable to $N(0, 1)$ variates, but they are not independent; in particular, residuals for different responses within the same covariate pattern can be highly correlated.

```
> # examine Pearson residuals,  including the predictor variables in the output
> residuals(fitA, include.predvars=TRUE)

       Lake Sex  Size       Fish         Inv       Rept        Bird       Other
1   Hancock   M small -0.2893234  0.01924183 -0.6508579 -0.81543588  1.06284497
2   Hancock   M large -0.1749598 -0.37965856 -0.6429569  0.26001584  0.60804725
3   Hancock   F small  0.7755248  0.22721841  0.5704975 -0.04103724 -1.45334592
4   Hancock   F large -0.7620718 -0.49345962  0.1947285  0.40389196  0.90768009
5  Oklawaha   M small  0.3921555 -0.48720975 -0.5779289 -0.20472660  1.32159652
6  Oklawaha   M large  0.1283005  0.35363226  0.5367653 -0.84819672 -1.33745742
7  Oklawaha   F small -0.1221163 -0.16226839 -0.2842677 -0.40386249  1.36112124
8  Oklawaha   F large -0.8475502  0.59323730 -0.7190831  3.36344866 -0.36311574
9  Trafford   M small  0.3122499  0.43823698  0.0639841 -0.64307988 -0.81398457
10 Trafford   M large -0.1854958  0.29687611  0.1770439  0.04314838 -0.26726406
11 Trafford   F small  0.1979997 -0.99355457 -0.1985867  0.65294310  1.44152747
12 Trafford   F large -0.4617629  1.71887453 -0.5108486 -0.36599155 -0.42550452
13   George   M small -0.1423618 -0.02500372 -0.4869544  1.67993948 -0.32311453
14   George   M large  0.2823010 -1.26678158 -0.5007227  0.20550261  0.80804229
15   George   F small -1.0670568  0.97282673  1.9270805 -0.69531412 -0.27546104
16   George   F large  0.9227690 -0.59586692 -0.5756869 -1.02672602 -0.03331025
```

Examining these residuals, we see one cell where the observed frequency is a bit higher than we would expect if the model were true ($R_{8,4} = 3.63$); residuals for all other cells look reasonable.

Predictions. Predicted values returned by `predict()` are similar to fitted values, except that they are computed for rows of a dataset rather than its unique covariate patterns. The optional argument `newdata` allows us to provide a new data frame where the function looks for predictor values. With this feature, we can predict for out-of-sample units that were not used in the model fitting, which is useful for cross-validation.

```
> # fit model to microdata with alligators #101 and 102 removed, then
> # use the fitted model to get predicted probabilities for those alligators
> fitFrame <- alligatorMicro[ -(101:102), ]
> predFrame <- alligatorMicro[ 101:102, ]
> fit <- rrLogit( Food ~ Lake + Sex + Size, data=fitFrame )
> predict( fit, newdata=predFrame )

        Fish       Inv       Rept       Bird      Other
101 0.2234349 0.6362147 0.09672538 0.01137081 0.03225419
102 0.3692054 0.2544882 0.29371967 0.03937133 0.04321541
```

If we specify `se.fit=TRUE`, then `predict()` computes standard errors by the delta method (Agresti, 2013), as described in A; in that case, the result is a named list with three components:

`fit`: the matrix of fitted values,

`se.fit`: a matrix with same dimensions as `fit` containing standard errors, and

`cov.fit.array`: a three-dimensional array holding the estimated covariance matrices for the fitted values.

The dimensions of `cov.fit.array` are `c(n,C,C)`, where `n` is the number of rows in the prediction frame (either the original data frame or `newdata`) and `C` is the number of response categories. Each covariance matrix is rank-deficient due to the constraint that the predicted probabilities for each unit must sum to one.

```
> result <- predict( fit, newdata=predFrame, se.fit=TRUE )
> # display fitted probs for first alligator in predFrame
> result$fit[1,]

     Fish       Inv      Rept      Bird      Other
0.22343488 0.63621474 0.09672538 0.01137081 0.03225419

> # display standard errors for those fitted probs
> result$se.fit[1,]

     Fish       Inv      Rept      Bird      Other
0.07659614 0.09710536 0.05629158 0.01368318 0.02602959

> # display estimated covariance matrix for those fitted probs
> result$cov.fit.array[1,,]
```

```
              Fish           Inv          Rept          Bird         Other
Fish   5.866968e-03 -0.0056447340 -3.633956e-04  5.573571e-05  8.542577e-05
Inv   -5.644734e-03  0.0094294506 -2.781232e-03 -2.531274e-04 -7.503567e-04
Rept  -3.633956e-04 -0.0027812325  3.168742e-03 -6.713709e-07 -2.344228e-05
Bird   5.573571e-05 -0.0002531274 -6.713709e-07  1.872294e-04  1.083365e-05
Other  8.542577e-05 -0.0007503567 -2.344228e-05  1.083365e-05  6.775396e-04
```

```
> # show that the matrix is rank-deficient
> qr( result$cov.fit.array[1,,] )$rank
```

```
[1] 4
```

Interval estimates for predicted probabilities. Using the delta method, inferences for predicted probabilities could be based on the large-sample approximation

$$\hat{\pi}_{gy} \sim N\left(\pi_{gy}, [\text{SE}(\hat{\pi}_{gy})]^2\right),$$

where $\text{SE}(\hat{\pi}_{gy})$ is the standard error from calling `predict()` with `se.fit=TRUE`. By this approximation, a $100(1-\alpha)$% confidence interval for π_{gy} would be

$$\hat{\pi}_{gy} \pm z_{1-\alpha/2}\,\text{SE}(\hat{\pi}_{gy}),$$

where z_p is the pth quantile of $N(0,1)$. Unfortunately, the normal approximation for $\hat{\pi}_{gy}$ may be implausible when π_{gy} is close to zero or one. We see ample evidence of this with alligator #1, for whom the predicted probability of Bird is `0.0114` with a standard error of `0.0137`; the 95% interval

```
> piHat <- result$fit[1,4]
> piSE <- result$se.fit[1,4]
> piLower <- piHat - qnorm(.975) * piSE
> piUpper <- piHat + qnorm(.975) * piSE
> c(piLower, piUpper)
```

```
[1] -0.01544773  0.03818935
```

strays into the negative range. A better approach is to apply a monotonic transformation $\psi_{gy} = \psi(\pi_{gy})$ that maps $(0,1)$ to the real line, compute a symmetric interval on the ψ scale as

$$\psi(\hat{\pi}_{gy}) \pm z_{1-\alpha/2}\,\psi'(\hat{\pi}_{gy})\,\text{SE}(\hat{\pi}_{gy}), \tag{9}$$

$\psi'(\pi) = d\psi/d\pi$, and then translate the interval endpoints back to probabilities by the inverse transformation $\pi_{gy} = \psi^{-1}(\psi_{gy})$. Taking

$$\begin{aligned}
\psi(\pi) &= \log\left(\frac{\pi}{1-\pi}\right), \\
\psi'(\pi) &= \frac{1}{\pi\,(1-\pi)}, \\
\psi^{-1}(\psi) &= \frac{\exp(\psi)}{1+\exp(\psi)},
\end{aligned}$$

the approximate 95% interval no longer covers zero.

```
> logit <- function(pi) log( pi / (1-pi) )
> logitDeriv <- function(pi) 1 / ( pi * (1-pi ) )
> logitInv <- function(psi) exp(psi) / ( 1 + exp(psi) )
> psiHat <- logit( piHat )
> psiLower <- psiHat - qnorm(.975) * logitDeriv( piHat ) * piSE
> psiUpper <- psiHat + qnorm(.975) * logitDeriv( piHat ) * piSE
> logitInv( c( psiLower, psiUpper ) )
```

```
[1] 0.001057343 0.111095404
```

3.5 Assessing model fit

Pearson goodness-of-fit test. Squaring the Pearson residuals and summing them over covariate patterns gives the Pearson goodness-of-fit statistic

$$X^2 = \sum_{g=1}^{G} \sum_{y=1}^{C} R_{gy}^2, \tag{10}$$

which measures the overall lack of fit. More specifically, it compares the fit of the current model to that of a saturated model which estimates $\hat{\boldsymbol{\pi}}_g$ independently for each covariate pattern $g = 1, \ldots, G$, with no restrictions other than $\sum_{y=1}^{C} \pi_{gy} = 1$. If the current model holds, then X^2 is asymptotically distributed as χ^2, with degrees of freedom equal to the number of free parameters in the saturated model minus the number of free parameters in the current model. For this example, the saturated model has $C - 1 = 4$ free parameters within each of the $G = 16$ covariate patterns for a total of 64 parameters. The current model has $C - 1 = 4$ vectors of non-zero coefficients, and each vector has length $p = 6$, for a total of 24 parameters. Therefore, the nominal degrees of freedom associated with this test are $64 - 24 = 40$.[4]

```
> # compute the Pearson goodness-of-fit statistic and its approximate p-value
> X2 <- sum( residuals(fitA)^2 )
> 1 - pchisq(X2, 40)
```

```
[1] 0.08802714
```

The p-value suggests some lack of fit, but not enough to conclusively reject the current model. For this example, however, the p-value should not be taken seriously, because the fitted means $\hat{\mu}_{gy}$ are too small. A common rule-of-thumb is that the χ^2 approximation works well if no more than 20% of the fitted means are less than 5.0 and none are less than 1.0 (Agresti, 2013, Sec. 3.2).

[4]The nominal degrees of freedom do not include any adjustments to account for estimates that fall on a boundary of the parameter space, as they do here for the alternative saturated model; we discuss this further in Section 3.6.

```
> # check the sizes of the fitted means to see if the chi-squared
> # approximation is reasonable according to Agresti's rule-of-thumb
> muhat <- fitted(fitA, type="mean")
> mean( muhat < 5 )

[1] 0.775

> mean( muhat < 1 )

[1] 0.425
```

In this case, 77% fall below 5.0 and 42% fall below 1.0, so we should not put much stock in the apparently poor fit. Rather, we should focus on the model's predictors to see whether important interactions have been omitted or if any main effects could be dropped.

Fitting a saturated model. The Pearson goodness-of-fit test is asymptotically equivalent to a likelihood-ratio test (LRT) that compares the current model to the saturated model. To fit the saturated model, use `saturated=TRUE`. Under this option, `rrLogit()` enumerates all of the distinct combinations of predictors on the right-hand-side of the model formula that appear in the dataset with frequencies greater than zero, then it independently fits a multinomial model to each one.

```
> # fit the saturated model to wide-format data with saturated=TRUE;
> # this option works in the same way for narrow format and microdata
> fitSat <- rrLogit( cbind(Fish, Inv, Rept, Bird, Other) ~ Lake + Sex + Size,
+    data=alligatorWide, saturated=TRUE)
> summary(fitSat)

cbind(Fish, Inv, Rept, Bird, Other) ~ Lake + Sex + Size
Saturated option: TRUE
           Prior: none

               Data format: wide
       Frequencies supplied: TRUE
      Rows of supplied data: 16
   Total N in supplied data: 219
Distinct covariate patterns: 16
   Empty covariate patterns: 0

Response categories: Fish Inv Rept Bird Other
  Baseline category: Fish

Assumed perturbation matrix: Identity

Number of estimated parameters = 64
            Degrees of freedom =  0

Expectation-Maximization (EM) algorithm
Starting values: default
Converged at iteration 2
Estimate at or near boundary

  Final logP = -243.8009
Final loglik = -243.8009
```

Notice that the printed summary includes the message, "Estimate at or near boundary," which we will explain shortly. The LRT statistic is twice the difference in the loglikelihoods achieved by the two model fits. To compute the LRT statistic, you can extract the loglikelihood values from the fitted model objects using `loglik()`.

```
> # compute the LRT test statistic
> 2 * ( loglik(fitSat) - loglik(fitA) )

[1] 50.26369
```

Note the similarity between this value and the Pearson X^2. As with X^2, the fitted means in this example are not large enough for the LRT's p-value to be trustworthy. With a saturated model, the fitted means under the ML solution are equal to the corresponding response frequencies, so the residuals are exactly zero.

```
> residuals(fitSat)

      Fish Inv Rept Bird Other
 [1,]    0   0    0    0     0
 [2,]    0   0    0    0     0
 [3,]    0   0    0    0     0
 [4,]    0   0    0    0     0
 [5,]    0   0    0    0     0
 [6,]    0   0    0    0     0
 [7,]    0   0    0    0     0
 [8,]    0   0    0    0     0
 [9,]    0   0    0    0     0
[10,]    0   0    0    0     0
[11,]    0   0    0    0     0
[12,]    0   0    0    0     0
[13,]    0   0    0    0     0
[14,]    0   0    0    0     0
[15,]    0   0    0    0     0
[16,]    0   0    0    0     0
```

If you try to extract the coefficients from a model fit with `saturated="TRUE"`, you won't find any.

```
> coef(fitSat)

NULL
```

When `saturated="TRUE"`, the estimated probabilities are computed directly from the frequencies using $\hat{\boldsymbol{\pi}}_g = \boldsymbol{f}_g/N_g$; no model matrix is applied during the model fit, so the coefficients are not defined.

Comparing nested models. An easier way to perform the LRT is to compare fitted model objects with `anova()`. The default setting `method="lrt"` automatically computes the LRT statistic and its degrees of freedom (df). With the argument `pval=TRUE`, it also displays a p-value based on the χ^2 approximation.

```
> anova( fitA, fitSat, pval=TRUE )

Model 1: cbind(Fish, Inv, Rept, Bird, Other) ~ Lake + Sex + Size
Model 2: cbind(Fish, Inv, Rept, Bird, Other) ~ Lake + Sex + Size (saturated)
  nParams -2*loglik df change   pval
1      24    537.87
2      64    487.60 40 50.264 0.1282
```

The LRT is appropriate for comparing models that are nested in the sense that the smaller model is a special case of the larger. Any model fit with `saturated=FALSE` is nested within the version fit with `saturated=TRUE`. If models M_1 and M_2 are both fit with `saturated=FALSE`, then M_1 is nested within M_2 if every term in M_1 also appears in M_2. If `anova()` is applied to a sequence of nested models, the smallest model should be listed first. This function does not automatically check the models to see whether they are nested, so when using `method="lrt"`, it is up to the user to make sure that they are. In the example below, we assess a sequence of nested models starting with no predictors and ending with the saturated model.

```
> M1 <- rrLogit( Food ~ 1, data=alligatorMicro )
> M2 <- rrLogit( Food ~ Lake, data=alligatorMicro )
> M3 <- rrLogit( Food ~ Lake + Size, data=alligatorMicro )
> M4 <- rrLogit( Food ~ Lake + Size + Sex, data=alligatorMicro )
> M5 <- rrLogit( Food ~ Lake + Size + Sex, data=alligatorMicro, saturated=TRUE )
> anova( M1, M2, M3, M4, M5, pval=TRUE )

Model 1: Food ~ 1
Model 2: Food ~ Lake
Model 3: Food ~ Lake + Size
Model 4: Food ~ Lake + Size + Sex
Model 5: Food ~ Lake + Size + Sex (saturated)
  nParams -2*loglik df change   pval
1       4    604.36
2      16    561.17 12 43.195 0.0000
3      20    540.08  4 21.087 0.0003
4      24    537.87  4  2.215 0.6963
5      64    487.60 40 50.264 0.1282
```

As we discussed earlier, the p-value for the comparison between models `M4` and `M5` may not be trustworthy, because the estimated expected means are too small. The p-values for the other model comparisons may be more reasonable, because for any fixed sample size, the χ^2 approximation works better for tests with smaller df. Based on these results, there is little reason to keep `Sex` in the model.

Comparing non-nested models. For non-nested model comparisons, `anova()` implements the Akaike information criteron (`method="AIC"`) and Bayesian information criterion (`method="BIC"`), which include penalty functions to discourage the inclusion of unnecessary predictors. Smaller values of these criteria correspond to better models. With `showRank=TRUE`, the models are ranked from best to worst.

```
> # model with no predictors
> M1 <- rrLogit( Food ~ 1, data=alligatorMicro )
> # models with one main effect
> M2 <- rrLogit( Food ~ Lake, data=alligatorMicro )
> M3 <- rrLogit( Food ~ Sex, data=alligatorMicro )
> M4 <- rrLogit( Food ~ Size, data=alligatorMicro )
> # models with two main effects
> M5 <- rrLogit( Food ~ Lake + Sex, data=alligatorMicro )
> M6 <- rrLogit( Food ~ Lake + Size, data=alligatorMicro )
> M7 <- rrLogit( Food ~ Sex + Size, data=alligatorMicro )
> # model with three main effects
> M8 <- rrLogit( Food ~ Lake + Sex + Size, data=alligatorMicro )
> # compare and rank these models using AIC
> anova( M1, M2, M3, M4, M5, M6, M7, M8, method="AIC", showRank=TRUE )
```

```
Model 1: Food ~ 1
Model 2: Food ~ Lake
Model 3: Food ~ Sex
Model 4: Food ~ Size
Model 5: Food ~ Lake + Sex
Model 6: Food ~ Lake + Size
Model 7: Food ~ Sex + Size
Model 8: Food ~ Lake + Sex + Size
  nParams -2*loglik    AIC rank
1       4    604.36 612.36    7
2      16    561.17 593.17    3
3       8    602.26 618.26    8
4       8    589.21 605.21    5
5      20    555.47 595.47    4
6      20    540.08 580.08    1
7      12    588.18 612.18    6
8      24    537.87 585.87    2
```

3.6 Boundary problems and numerical exceptions

Estimate at or near boundary. Recall that when we fit the model with `saturated=TRUE`, we saw this message in the printed summary:

```
Estimate at or near boundary
```

"Boundary" refers to outer limits of the parameter space. For a saturated model, the parameter space is the set of probability vectors $\boldsymbol{\pi}_g = (\pi_{g1}, \ldots, \pi_{gC})^\top$, $g = 1, \ldots, G$ satisfying $\pi_{gy} \geq 0$ and $\sum_{y=1}^{C} \pi_{gy} = 1$, and we fall on a boundary if one or more of the estimated $\hat{\pi}_{gy}$'s is zero. The loglikelihood function for this model,

$$l = \sum_{g=1}^{G} \sum_{y=1}^{C} f_{gy} \log \pi_{gy}, \tag{11}$$

is defined at the boundary, provided that we regard $0 \log 0$ as 0 and do not observe any impossible events where $\pi_{gy} = 0$ but $f_{gy} > 0$. For a saturated model, the ML

estimate will fall on a boundary whenever a frequency of zero appears in a non-empty covariate pattern.[5]

When `saturated=FALSE`, however, the issues are more subtle. From the perspective of $\boldsymbol{\beta}$, the parameter space is $\mathbb{R}^d$ with $d = p(C-1)$, which has no boundaries. However, if individual coefficients become large in magnitude, some of the π_{gy}'s become indistinguishable from zero. So even though the $\boldsymbol{\beta}$-space is unbounded, with respect to the π_{gy}'s we may be at or near a boundary. For example, see what happens when we fit a model with `Lake:Size` interactions.

```
> M <-  rrLogit( Food ~ Lake + Size + Lake:Size, data=alligatorMicro )

Estimate at or near boundary; standard errors may be unreliable
```

The Newton-Raphson algorithm is deemed to have converged if the largest observed change in any of the estimated π_{gy}'s from one iteration to the next falls below a threshold.[6] In this example, the procedure did converge, but the final value of or more of the fitted π_{gy}'s was very small. When this happens, the Hessian matrix containing second derivatives of the loglikelihood may be nearly singular, making the SEs unstable. A quick glance at the table of coefficients shows that some of the estimated interactions are unusually large and their SEs are enormous.

```
> # display a portion the table of coefficients with SEs
> summary(M)$coefficients[,,2:3]

, , Response = Inv

                            coef           SE zstat   pval
(Intercept)             -1.749200    0.5417363 -3.23 0.0012
LakeOklawaha             2.537657    0.7644523  3.32 0.0009
LakeTrafford             2.537657    0.7644523  3.32 0.0009
LakeGeorge               1.921050    0.6392260  3.01 0.0027
Sizelarge              -17.278587 5120.1478700  0.00 0.9973
LakeOklawaha:Sizelarge  16.004622 5120.1479181  0.00 0.9975
LakeTrafford:Sizelarge  16.356599 5120.1479245  0.00 0.9975
LakeGeorge:Sizelarge    14.273524 5120.1479846  0.00 0.9978

, , Response = Rept

                             coef           SE zstat   pval
(Intercept)             -2.4423470    0.7372098 -3.31 0.0009
LakeOklawaha             0.8329091    1.3204084  0.63 0.5282
LakeTrafford             1.5260563    1.1151136  1.37 0.1711
LakeGeorge              -0.3302417    1.2672720 -0.26 0.7944
Sizelarge                0.4964369    1.2985898  0.38 0.7022
LakeOklawaha:Sizelarge   0.3398111    1.7691594  0.19 0.8477
LakeTrafford:Sizelarge   0.1321718    1.6364602  0.08 0.9356
LakeGeorge:Sizelarge   -18.1987761 6774.0695743  0.00 0.9979
```

[5] If a covariate pattern appears in the dataset but the frequencies for response categories $1, \ldots, C$ are all zero, the pattern is considered empty and excluded from the model when `saturated=TRUE`.

[6] This threshold is a control parameter named `critConverge`, has a default value of 10^{-8}. For information on how to change this value, see `?rrLogit` and `?rrLogitControl`.

This dataset provides too little information to get stable estimates for some `Lake:Size` interactions. However, the model's predictions exhibit the opposite problem; some appear to have too much precision.

```
> # show predicted probs and their SEs for alligator #219, rounded to
> # four decimal places
> result <- predict(M, se.fit=TRUE)
> round( result$fit[219,], 4 )

  Fish    Inv   Rept   Bird  Other
0.7727 0.0455 0.0000 0.0455 0.1364

> round( result$se.fit[219,], 4 )

  Fish    Inv   Rept   Bird  Other
0.0893 0.0444 0.0000 0.0444 0.0732

> alligatorMicro[219,]

     Lake Sex  Size  Food
219 George   F large Other
```

This model claims it's virtually impossible for an alligator like #219, a large specimen from Lake George, to primarily consume Reptiles. This demonstrates a fundamental problem with ML: the loglikelihood may rise sharply near a boundary, creating an illusion of high precision where there should be substantial uncertainty.

Numerical exceptions and aborted procedures. Boundary estimates are likely with sparse data where many of the f_{gy}'s are zero. They can also happen if just one frequency is zero, if that zero falls in a covariate pattern that gets singled out by a predictor or by a linear combination of predictors. These conditions are known as complete separation or quasi-complete separation (Albert and Anderson, 1984; Lesaffre and Albert, 1989). In these situations, coefficients and SEs may become large as the estimate approaches a boundary. In some cases, the fitting procedure may abort because of a numerical exception such as attempted division by zero, logarithm of a non-positive number, or square root of a negative number. Numerical exceptions often occur when a Hessian (second derivative) matrix is being factored or inverted. For example, see what happens when we try to include all two- and three-way interactions.[7]

```
> M <-  rrLogit( Food ~ Lake*Sex*Size, data=alligatorMicro )

Note: Matrix not positive definite
OCCURRED IN: cholesky_saxpy in MOD matrix_methods
Newton-Raphson aborted
```

[7] In this example, the model with terms `Lake*Sex*Size` happens to be equivalent to the saturated model that we fit earlier with `saturated=TRUE`, because all possible combinations of of `Lake`, `Sex` and `Size` are represented in the dataset. With `saturated=TRUE`, we didn't encounter any problems, because under that option the ML procedure does not attempt to compute or invert the Hessian.

```
Iteration 14
Attempted logarithm of non-positive number
OCCURRED IN: compute_logP_score_hess_rrlogit in MOD rrlogit_engine
Hessian-based SEs not available


Warning message:
In rrLogit.formula(Food ~ Lake * Sex * Size, data = alligatorMicro) :
  Procedure failed to converge by iteration 14
```

Coefficients in logistic models represent log-odds ratios. A single zero frequency can produce an odds ratio of $0/x$ or $x/0$, sending the estimated coefficient toward $-\infty$ or $+\infty$. Multiple zero frequencies may lead to odds ratios of $0/0$, which are uninformative. When the fitting procedure reports a numerical exception and aborts before it has converged, it means that one or more odds ratios were veering sharply toward $0/x$, $x/0$, or $0/0$ before the fitted probabilities stabilized.

Boundary estimates and model comparisons. When zero frequencies send ML estimates to a boundary, some of the Pearson residuals are undefined, and we cannot compute the Pearson goodness-of-fit statistic (10). If we compare the loglikelihoods of nested models by their loglikelihood values, and if one or both of the ML solutions lies on a boundary, the degrees of freedom calculated in the usual way can be misleading. The nominal degrees of freedom is the number of parameters that we are attempting to estimate under one model, minus the the number that we are attempting to estimate under the other model. The problem is that some of parameters that we are attempting to estimate may not actually be estimable from the given data, as when an odds ratio becomes $0/0$. A traditional way to handle this is to set aside the cells with fitted values of zero and adjust for parameters that cannot be estimated (Bishop et al., 1975; Clogg et al., 1991). Rules for making these adjustments are complicated and difficult to implement, and the `anova()` method does not attempt to adjust degrees of freedom for estimates on a boundary.

Possible remedies. Problems due to zero frequencies can sometimes be fixed by removing unnecessary predictors. For this alligator dataset, avoiding models with two- and three-way interactions seems reasonable. Removing predictors is not a panacea, however, and it may lead us to unwittingly discard variables because their associations with the outcome are *too strong*. Imagine a world where female alligators never eat fish; any model that includes `Sex` would produce ML estimates on a boundary, but models without `Sex` would fail to describe this important phenomenon. As an alternative to removing predictors, `rrLogit()` allows you to smooth estimates away from the boundary by applying an informative prior distribution.

3.7 Prior distributions

Data-augmentation prior (DAP). To address problems associated with zero frequencies, Clogg et al. (1991) suggest maximizing the objective function

$$\log P = l + \sum_{g=1}^{G}\sum_{y=1}^{C} f_{gy}^{\dagger} \log \pi_{gy}, \tag{12}$$

where l is the loglikelihood (11) and $f_{gy}^{\dagger}$ is a small positive constant applied to response category y in covariate pattern g. These constants, which are not necessarily integers, play the role of imaginary prior observations added to the observed frequencies. The extra term $\sum_g \sum_y f_{gy}^{\dagger} \log \pi_{gy}$ can be regarded either as a penalty function that penalizes estimates for getting too close to a boundary, or as a log-prior density for $\boldsymbol{\beta}$. Under the latter interpretation, the maximizer of (12) becomes a posterior mode. A prior distribution with this form, called a data-augmentation prior (DAP) by Bedrick et al. (1996), has certain advantages over the more traditional priors that are formulated as densities for $\boldsymbol{\beta}$. DAP's can be easier to interpret, and they do not complicate the model fitting procedure; an algorithm for ML estimation can maximize (12) if we simply replace each f_{gy} with the augmented frequency $f_{gy} + f_{gy}^{\dagger}$.

Choosing prior frequencies. Let $N_g^{\dagger} = \sum_{y=1}^{C} f_{gy}^{\dagger}$ denote the total number of imaginary prior observations added to covariate pattern g, and let $N^{\dagger} = \sum_{g=1}^{G} N_g^{\dagger}$ denote the total prior sample size. As recommended by Clogg et al. (1991), we divide this total equally among covariate patterns, taking $N_g^{\dagger} = N^{\dagger}/G$ for $g = 1, \ldots, G$, and then allocate each $N_g^{\dagger}$ to the response categories according to $f_{gy}^{\dagger} = N_g^{\dagger}\tilde{\pi}_y$, where $\tilde{\pi}_y$ is a prior guess of the marginal probability $P(Y = y)$.

To apply the data-augmentation prior, call `rrLogit()` with `prior="DAP"`. The default is `prior="none"`, which sets all imaginary prior frequencies to zero. When `prior="DAP"`, the prior frequencies are determined by two additional arguments,

`priorFreqTot`: the overall prior sample size $N^{\dagger}$, and

`priorAlloc`: a numeric vector holding the prior proportions $\tilde{\pi}_1, \ldots, \tilde{\pi}_C$.

If these are not provided, we follow the suggestions of Clogg et al. (1991) and take $N^{\dagger} = p(C-1)$, the total number of coefficients being estimated, and $\tilde{\pi}_y = \sum_g f_{gy} / \sum_g N_g$, the observed proportion of $Y = y$ in the sample.

As a demonstration, recall what happened when we tried to estimate `Lake:Size` interactions.

```
> fitML <-  rrLogit( Food ~ Lake*Size, data=alligatorMicro )
```

```
Estimate at or near boundary; standard errors may be unreliable
```

Now let's refit the model with `prior="DAP"`.

```
> # apply a DAP with default settings
> fitDAP <-  rrLogit( Food ~ Lake*Size, data=alligatorMicro, prior="DAP" )
```

With this prior, the boundary message has disappeared, because the smallest fitted probabilities have been pulled away from zero. The estimated coefficients and SEs have also stabilized.

```
> # show some of the estimated coefficients and SEs without the prior
> summary(fitML)$coefficients[,,2,drop=FALSE]

, , Response = Inv

                           coef           SE zstat   pval
(Intercept)            -1.749200    0.5417363 -3.23 0.0012
LakeOklawaha            2.537657    0.7644523  3.32 0.0009
LakeTrafford            2.537657    0.7644523  3.32 0.0009
LakeGeorge              1.921050    0.6392260  3.01 0.0027
Sizelarge             -17.278587 5120.1478700  0.00 0.9973
LakeOklawaha:Sizelarge 16.004622 5120.1479181  0.00 0.9975
LakeTrafford:Sizelarge 16.356599 5120.1479245  0.00 0.9975
LakeGeorge:Sizelarge   14.273524 5120.1479846  0.00 0.9978

> # now show them with the prior
> summary(fitDAP)$coefficients[,,2,drop=FALSE]

, , Response = Inv

                           coef        SE zstat   pval
(Intercept)           -1.5754748 0.4857920 -3.24 0.0012
LakeOklawaha           2.1652234 0.6836814  3.17 0.0015
LakeTrafford           2.1652234 0.6836814  3.17 0.0015
LakeGeorge             1.7023799 0.5849388  2.91 0.0036
Sizelarge             -0.4816918 1.1172532 -0.43 0.6664
LakeOklawaha:Sizelarge -0.5872242 1.2873809 -0.46 0.6483
LakeTrafford:Sizelarge -0.2883128 1.3060765 -0.22 0.8253
LakeGeorge:Sizelarge   -1.8259846 1.3714386 -1.33 0.1830
```

Adjusting the strength of the prior. When using a prior distribution to stabilize a model, the prior ought to be strong enough to achieve the desired objective, but not so strong that it exerts undue influence and prevents the data from speaking for themselves. Basic information about the prior can be viewed with `summary()`.

```
> summary(fitDAP)

Food ~ Lake * Size
Saturated option: FALSE
           Prior: DAP

               Data format: narrow
      Frequencies supplied: FALSE
```

```
      Rows of supplied data: 219
   Total N in supplied data: 219
Distinct covariate patterns: 8
   Empty covariate patterns: 0

Response categories: Fish Inv Rept Bird Other
  Baseline category: Fish

Assumed perturbation matrix: Identity

Number of estimated parameters = 32
            Degrees of freedom =  0

Data-augmentation prior (DAP)
Prior effective sample size = 32
        Prior N per pattern =  4
Proportions for allocating prior counts:
    Fish 0.42922
     Inv 0.27854
    Rept 0.08676
    Bird 0.05936
   Other 0.14612

Newton-Raphson procedure
Starting values: default
Converged at iteration 7
Gradient length = 0.000000

  Final logP = -313.7247
Final loglik = -264.0331
```

This DAP added the equivalent of $N^\dagger = 32$ imaginary observations, which is about 15% of the actual sample size $N = 219$. We cannot offer any general rule to determine when $N^\dagger/N$ has become too large, but we do recommend examining the loglikehood function to see how much it drops when we apply the prior.

```
> # difference in loglikelihoods, ML minus posterior mode
> loglik( fitML ) - loglik( fitDAP )

[1] 2.53291
```

Exponentiating this difference gives us a quantity known as a Bayes factor.

```
> # Bayes factor comparing the ML estimate to the posterior mode
> exp( loglik( fitML ) - loglik( fitDAP ) )

[1] 12.59009
```

This Bayes factor, the likelihood at the ML estimate divided by the likelihood at the posterior mode, summarizes the data's evidence about the relative plausibility of these two estimates. Using rules-of-thumb for interpreting Bayes factors given by Jeffreys (1961) and Kass and Raftery (1995), a value between 1.0 and $\sqrt{10} \approx 3.2$ means that the evidence for preferring the ML estimate to the posterior mode is weak and "not worth more than a bare mention"; a value between 3.2 and 10 means

that the evidence is "substantial"; and a value greater than 10 is "strong." These guidelines suggest that for this example, the default version of `prior="DAP"` might be too aggressive. To weaken the prior, reduce the value of `priorFreqTot`. Note that `priorFreqTot` doesn't have to be an integer.

```
> # set priorFreqTot to 2.19, which corresponds to one percent
> # of the sample size
> fitDAP <-  rrLogit( Food ~ Lake*Size, data=alligatorMicro,
+    prior="DAP", priorFreqTot=2.19 )
```

Under this weaker prior, the coefficients and SEs are still stable,

```
> summary(fitDAP)$coefficients[,,2,drop=FALSE]

, , Response = Inv

                           coef        SE zstat   pval
(Intercept)           -1.7354125 0.5371972 -3.23 0.0012
LakeOklawaha           2.5075496 0.7578073  3.31 0.0009
LakeTrafford           2.5075496 0.7578073  3.31 0.0009
LakeGeorge             1.9039509 0.6348593  3.00 0.0027
Sizelarge             -2.8008819 3.6801948 -0.76 0.4466
LakeOklawaha:Sizelarge 1.5437252 3.7456079  0.41 0.6802
LakeTrafford:Sizelarge 1.8914667 3.7542021  0.50 0.6144
LakeGeorge:Sizelarge  -0.1342751 3.8269924 -0.04 0.9720
```

and the Bayes factor shows that there is almost no evidence whatsoever to prefer the ML solution to this new result.

```
> # Bayes factor comparing the ML estimate to the new posterior mode
> exp( loglik( fitML ) - loglik( fitDAP ) )

[1] 1.171978
```

When applying a DAP, it is good practice to try to keep the Bayes factor small. This is not always possible, especially with larger datasets. If a model is not unnecessarily complicated, and the mildest prior that stabilizes the fit produces a Bayes factor greater than 3.2, the prior may still be defensible on the grounds that the $f^{\dagger}_{gy}$'s portray a reasonable state of foreknowledge or ignorance. Note that a Bayes factor will not be computable if the ML fitting procedure aborts due to a numerical exception.

Comparing alternative models fit with the same prior. Unlike traditional prior densities for $\boldsymbol{\beta}$, the same DAP can be applied to models with different predictors and $\boldsymbol{\beta}$'s of different sizes. Nested models can be compared with `anova()` either by differences in the loglikelihood (`method="lrt"`) or by differences in the $\log P$ function (`method="logP"`), and non-nested models can be compared using `method="AIC"` or `method="BIC"`. A test of nested models based on $\log P$ has the same asymptotic properties as the corresponding LRT, because as the sample size increases, the impact of any fixed prior distribution is eventually overwhelmed by the influence of

the data. In the example below, we test the significance of the `Lake:Size` interactions by fitting nested models with the same DAP.

```
> # fit model without the Lake:Size interactions
> M1 <- rrLogit( Food ~ Lake + Size, data=alligatorMicro,
+    prior="DAP", priorFreqTot=2.19 )
> # refit model with Lake:Size interactions
> M2 <- rrLogit( Food ~ Lake*Size, data=alligatorMicro,
+    prior="DAP", priorFreqTot=2.19 )
> # compare using the default method="lrt"
> anova( M1, M2, pval=TRUE )

Model 1: Food ~ Lake + Size
Model 2: Food ~ Lake * Size
  nParams -2*loglik df change   pval
1      20    540.09
2      32    523.32 12 16.776 0.1582

> # compare using method="logP"
> anova( M1, M2, method="logP", pval=TRUE )

Model 1: Food ~ Lake + Size
Model 2: Food ~ Lake * Size
  nParams -2*logP df change   pval
1      20  547.01
2      32  531.03 12 15.972 0.1925
```

If the prior is weak enough to allow the data to speak for themselves, then tests with `method="lrt"` and `method="logP"` should lead to similar conclusions.

Adjusting the prior allocation. When `prior="DAP"` and `priorAlloc` is not specified, the prior frequencies are distributed across response categories according to the marginal distribution of Y in the sample. If some response categories are rare, this default setting might not be effective at pulling estimates away from the boundary unless `priorFreqTot` is unreasonably large. When this happens, a better choice for `priorAlloc` is the vector of uniform probabilities `rep(1/C, C)` where `C` is the number of response categories. Uniform values for `priorAlloc` can be especially helpful when working with data that have been infused with noise, to which we now turn our attention.

4 Modeling noise-infused responses

4.1 A simple example

Two randomized responses with no predictors. We begin this section with a small dataset involving randomized response (RR). Because this example has no predictor variables, we might not need to use logistic regresssion; we can possibly get by

Table 2: Frequencies from a survey on rule violations in unemployment benefits collected by randomized response: Y_1^* = had respondent deliberately avoided work by rejecting a job offer or sabotaging a job application; Y_2^* = had respondent applied for fewer jobs than required. Source: van den Hout and van der Heijden (2007)

	Y_2^*		
Y_1^*	1 = Yes	2 = No	Total
1 = Yes	68	52	120
2 = No	103	189	292
Total	171	241	412

with the simple method-of-moments (MOM) procedures described in Section 2.3. Nevertheless, this example will introduce some of the basic features of `rrLogit()` that pertain to noise-infused responses.

van den Hout and van der Heijden (2007) describe a survey conducted in the Netherlands to detect rule violations in claims for unemployment benefits. One question, Y_1, asked participants whether they had ever avoided work by turning down a job offer or deliberately sabotaging a job application. Another question, Y_2, asked whether they had applied for fewer jobs than the program required. To elicit more honest answers on these sensitive topics, each participant was given two decks of shuffled cards. The deck on the right contained 80% red cards and 20% black cards, and the deck on the left had 20% red and 80% black. If the participant's true answer was "Yes," they were to draw a card from the right and report its color, and if the answer was "No," they were to do the same from the left. Treating red as a noisy version of "Yes" and black as a noisy version of "No," each question had the same perturbation matrix

$$\boldsymbol{T} = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}.$$

The cross classification of $N = 412$ participants by the noisy versions Y_1^* and Y_2^* is shown in Table 2.

Univariate analyses. To begin, let's examine each response variable separately. We first create a data frame with a single row containing the marginal frequencies for Y_1^*. There are no covariates, so we fit an intercept-only model using the wide-format syntax. The `rrLogit()` function needs to know the perturbation matrix $\boldsymbol{T}$, so we provide it through the argument `pertMat`.

```
> # create a data frame containing the marginal frequencies for first question
> df.1 <- data.frame( Yes=120, No=292 )
> # enter the perturbation matrix
```

```
> Tmat <- matrix( c( .8, .2, .2, .8 ), 2, 2 )
> rownames(Tmat) <- colnames(Tmat) <- c("Yes","No")
> # fit the logistic model with no predictors and display the estimated beta
> fit.1 <- rrLogit( cbind(Yes, No) ~ 1, data=df.1, pertMat=Tmat )
> coef( fit.1 )

             Yes       No
(Intercept)   0 1.718197
```

Estimated probabilities for the response categories are available from `fitted()`. This method has an optional argument `noisy` that becomes relevant when $\boldsymbol{T} \neq \boldsymbol{I}$. When `noisy=TRUE`, it returns fitted values for the noisy response Y^*, and when `noisy=FALSE`, it gives fitted values for the true response Y. The default setting is `FALSE`.

```
> # fitted probs for the noisy response
> fitted( fit.1, noisy=TRUE )

          Yes       No
[1,] 0.2912621 0.7087379

> # fitted probs for the true response
> fitted( fit.1 )

          Yes       No
[1,] 0.1521036 0.8478964
```

Notice that the estimated probability of "Yes" for the true Y_1 is only 0.152, compared to 0.291 for the noisy Y_1^*. This behavior is common with noise-infused data. Unless a perturbation matrix is specifically designed to maintain the marginal proportions as with invariant PRAM (Section 2.5), the added noise tends to flatten the response distribution, making rare categories more common and common categories more rare. To get standard errors for the estimated probabilities, use `predict()` with `se.fit=TRUE`.

```
> # standard errors for fitted probs, true response
> predict( fit.1, se.fit=TRUE )$se.fit

        Yes        No
1 0.03730654 0.03730654
```

We now repeat the analysis for the second question.

```
> # univariate analysis for question Y2
> df.2 <- data.frame( Yes=171, No=241 )
> fit.2 <- rrLogit( cbind(Yes, No) ~ 1, data=df.2, pertMat=Tmat )
> fitted( fit.2 )

          Yes       No
[1,] 0.3584142 0.6415858
```

```
> predict( fit.2, se.fit=TRUE )$se.fit

        Yes        No
1 0.04045847 0.04045847
```

The estimate of $P(Y_2 = \texttt{"Yes"})$ (0.358) is more than double the estimate of $P(Y_1 = \texttt{"Yes"})$, suggesting that rule violations of the second kind are more common than the first. To formally examine this hypothesis, we need to analyze both responses together to account for the possibility that Y_1 and Y_2 are related.

Comparing to the method of moments. Before moving on to a joint analysis, let's pause to compare our results from `rrLogit()` to those from the MOM procedure in Section 2.3.

```
> # enter freqs from the noisy version of Y1 and compute the proportions
> noisyFreq <- c( Yes=120, No=292 )
> N <- sum( noisyFreq )
> piStarHat <- noisyFreq / N
> # MOM estimates for true proportions
> Tinv <- solve(Tmat)
> piHat.MOM <- as.vector( Tinv %*% piStarHat )
> piHat.MOM

[1] 0.1521036 0.8478964
```

The differences between the MOM estimates and those from `rrLogit()` are extremely small.

```
> # show the differences between rrLogit and MOM estimates
> abs( piHat.MOM - predict(fit.1) )

          Yes          No
1 1.387779e-16 1.110223e-16
```

This is no accident, because whenever the MOM estimate lies in the interior of the parameter space (i.e., there are no negative estimated probabilities), MOM is identical to ML, with differences only due to rounding errors (van den Hout and van der Heijden, 2007). Standard errors from the MOM covariance matrix (2) are shown below.

```
> # compute unbiased estimate of covariance matrix for MOM and the
> # corresponding standard errors
> Vhat.MOM <- Tinv %*% ( diag(piStarHat) - piStarHat %*% t(piStarHat) ) %*%
+    t(Tinv) / ( N - 1 )
> se.MOM <- sqrt( diag( Vhat.MOM ) )
> se.MOM

     Yes       No
0.0373519 0.0373519
```

These are slightly different from ML because the MOM covariance matrix uses a denominator of $N - 1$ to correct for bias. If we replace the MOM denominator with N, the two methods become equal except for rounding error.

```
> # replace denominator of N-1 with N, then compare the resulting SEs to
> # those from rrLogit
> se.MOM.biased <- sqrt( diag( Vhat.MOM * (N-1)/N ) )
> abs( se.MOM.biased - predict(fit.1, se.fit=TRUE)$se.fit )

          Yes          No
1 1.387779e-17 1.387779e-17
```

Joint analysis of two noisy responses. To analyze the two variables together, we compound Y_1^* and Y_2^* into a single variable with four categories.

```
> # create a one-row data frame with frequencies for the compounded variable
> df.12 <- data.frame( Yes.Yes=68, Yes.No=52, No.Yes=103, No.No=189 )
```

The perturbation matrix for the compounded variable is the Kronecker product of the individual matrices (Section 2.4).

```
> bigT <- kronecker(Tmat, Tmat)
> rownames(bigT) <- colnames(bigT) <- c("Yes.Yes", "Yes.No", "No.Yes", "No.No")
> bigT

       Yes.Yes Yes.No No.Yes No.No
Yes.Yes    0.64   0.16   0.16  0.04
Yes.No     0.16   0.64   0.04  0.16
No.Yes     0.16   0.04   0.64  0.16
No.No      0.04   0.16   0.16  0.64

> # fit model to compounded variable, then display estimated probabilities
> # and SEs for the true proportions
> fit.12 <- rrLogit( cbind(Yes.Yes, Yes.No, No.Yes, No.No) ~ 1, data=df.12,
+    pertMat=bigT )
> fitted( fit.12 )

      Yes.Yes       Yes.No    No.Yes     No.No
[1,] 0.1649982 2.133668e-07 0.1901168 0.6448848

> predict( fit.12, se.fit=TRUE )$se.fit

     Yes.Yes       Yes.No     No.Yes     No.No
1 0.03227335 0.0001072887 0.04451019 0.0400154
```

Notice that the printed estimate of $P(Y_1 = \texttt{"Yes"}, Y_2 = \texttt{"No"})$ is essentially zero. Here the ML estimate for that probability is actually zero, but the fitting procedure stopped when changes in the estimated probabilities from one iteration to the next

fell below a threshold.[8] This is an example where ML and MOM give different results; the MOM estimated probability for that category is negative.

```
> # compute MOM estimates for the compounded variable
> noisyFreq <- c( Yes.Yes=68, Yes.No=52, No.Yes=103, No.No=189 )
> N <- sum( noisyFreq )
> piStarHat <- noisyFreq / N
> # MOM estimates for true proportions
> bigTinv <- solve(bigT)
> piHat.MOM <- as.vector( bigTinv %*% piStarHat )
> piHat.MOM

[1]  0.17718447 -0.02508091  0.18122977  0.66666667
```

How common are boundary estimates? Because the ML estimate for $\pi_{12} = P(Y_1 = \texttt{"Yes"}, Y_2 = \texttt{"No"})$ is zero, and the reported standard error for $\hat{\pi}_{12}$ is also very small, it is tempting to conclude that this category in the population must be empty or extremely rare. That would be a mistake. Noise infusion by RR or PRAM can make small subpopulations hard to detect even when they are not very rare. To illustrate, imagine a population in which the true proportions for the four categories are

$$\boldsymbol{\pi} = (\pi_{11}, \pi_{12}, \pi_{21}, \pi_{22})^\top = (0.17, 0.02, 0.21, 0.60)^\top.$$

If we drew a random sample of $N = 412$ individuals from this population, it is very likely that the sample would have at least one person from cell $(1, 2)$; the chance of getting no one from that category is only $(1 - 0.02)^{412} = 0.000243$ or about one in 4,000. Now imagine applying the noise mechanism from this example to the true responses. The table of noisy frequencies from the sample would be distributed as Mult$(N, \boldsymbol{\pi}^*)$ with $N = 412$ and

$$\boldsymbol{\pi}^* = \begin{bmatrix} 0.64 & 0.16 & 0.16 & 0.04 \\ 0.16 & 0.64 & 0.04 & 0.16 \\ 0.16 & 0.04 & 0.64 & 0.16 \\ 0.04 & 0.16 & 0.16 & 0.64 \end{bmatrix} \begin{bmatrix} 0.17 \\ 0.02 \\ 0.21 \\ 0.60 \end{bmatrix} = \begin{bmatrix} 0.1696 \\ 0.1444 \\ 0.2584 \\ 0.4276 \end{bmatrix}.$$

Using R, let's draw 10,000 samples from this population and see how often the ML estimates for elements of $\boldsymbol{\pi}$ are zero. We could do this by applying `rrLogit()` to each sample, but it's easier and faster to simply compute the MOM estimates and observe how many estimated probabilities are negative.

[8] Alert readers may wonder why `rrLogit()` did not issue a message about estimates on the boundary for this example. Proximity to a boundary is judged by the control parameter `critBoundary`, whose default value is `1e-08`. When the estimation procedure halted, the fitted probability for the `Yes.No` cell was a tiny bit larger than this, so no message was given.

```
> piTrue <- c( .17, .02, .21, .60 )
> piStarTrue <- bigT %*% piTrue
> N <- 412
> nSamp <- 10000
> set.seed(30966) # for reproducibility
> samples <- rmultinom(nSamp, N, piStarTrue)
> piStarHat.Samples <- samples / N
> piHat.MOM.Samples <- bigTinv %*% piStarHat.Samples
> apply( piHat.MOM.Samples < 0, 1, FUN=mean )

Yes.Yes  Yes.No  No.Yes   No.No
 0.0000  0.2998  0.0000  0.0000
```

About 30% of the samples produced negative estimates for $P(Y_1 = \texttt{"Yes"}, Y_2 = \texttt{"No"})$. With these rates of noise infusion and a sample size of $N = 412$, negative estimates from MOM and zero estimates from ML are very common, and neither of those answers makes sense. In situations like these, we recommend applying `rrLogit()` with a DAP.

Data-augmentation prior with a noise-infused response. In Section 3.7, we described DAPs for a non-noisy response. With a noisy response, a DAP works the same way, with the caveat that we are now seeding the dataset with imaginary observations of the unseen *true* response variable. To illustrate, let's refit our model using a DAP with `priorFreq=2` and `priorAlloc=rep(1/4,4)`. This adds information equivalent to seeing the true responses for $N^\dagger = 2$ imaginary persons, allocating them uniformly across the four cells of the true response table, 0.5 imaginary persons per cell.

```
> # apply a data-augmentation prior and refit
> fit.12.DAP <- rrLogit( cbind(Yes.Yes, Yes.No, No.Yes, No.No) ~ 1, data=df.12,
+    pertMat=bigT, prior="DAP", priorFreqTot=2, priorAlloc=rep(1/4, 4) )
> # Bayes factor comparing the two estimates
> exp( loglik(fit.12) - loglik(fit.12.DAP) )

[1] 1.481715
```

The Bayes factor comparing the two solutions is only 1.48, so there is almost no evidence in the data to prefer the ML estimate to the posterior mode. With this DAP, the estimate of π_{12} is much more plausible, and its standard error is larger.

```
> # show estimated probs for true response and their SEs
> piHat <- as.vector( fitted( fit.12.DAP ) )
> piHat

[1] 0.1585551 0.0162299 0.1968988 0.6283162

> piSE <- as.vector( predict( fit.12.DAP, se.fit=TRUE )$se.fit )
> piSE
```

```
[1] 0.03296247 0.01979087 0.04438030 0.04276111
```

An approximate 95% confidence interval computed as $\hat{\pi}_{12} \pm 1.96\,\text{SE}(\hat{\pi}_{12})$ would stray into the negative range. A better approach is to compute the interval on the logit scale and transform its endpoints back to the probability scale.

```
> # compute 95\% intervals for each pi on the logit scale, then transform
> # endpoints back to probabilities
> logit <- function(pi) log( pi / (1-pi) )
> logitDeriv <- function(pi) 1 / ( pi * (1-pi ) )
> logitInv <- function(psi) exp(psi) / ( 1 + exp(psi) )
> psiHat <- logit( piHat )
> psiLower <- psiHat - qnorm(.975) * logitDeriv( piHat ) * piSE
> psiUpper <- psiHat + qnorm(.975) * logitDeriv( piHat ) * piSE
> endpoints.95 <- data.frame( piLower=logitInv(psiLower),
+    piUpper=logitInv(psiUpper))
> rownames( endpoints.95 ) <- names( df.12 )
> endpoints.95
```

```
           piLower   piUpper
Yes.Yes 0.104027032 0.2341944
Yes.No  0.001451127 0.1577438
No.Yes  0.123914730 0.2982360
No.No   0.541435578 0.7076249
```

Inferences for other parameters. When we examined the two questions separately, our estimate for $P(Y_2 = \texttt{"Yes"})$ was higher than our estimate for $P(Y_1 = \texttt{"Yes"})$, suggesting that rule violations of the second kind are more common than the first. Using results from the joint analysis, we can now test the null hypothesis that $d = P(Y_2 = \texttt{"Yes"}) - P(Y_1 = \texttt{"Yes"}) = 0$. Because

$$d = (\pi_{11} + \pi_{21}) - (\pi_{11} + \pi_{12}) = \pi_{21} - \pi_{12},$$

this is equivalent to testing whether π_{21} is different from π_{12}. The estimated variance of $\hat{d} = \hat{\pi}_{21} - \hat{\pi}_{12}$ is $\boldsymbol{a}^\top \hat{V}(\hat{\boldsymbol{\pi}})\,\boldsymbol{a}$, where $\boldsymbol{a} = (0, -1, 1, 0)^\top$. In the code below, we use `predict()` with `se.fit=TRUE` to extract $\hat{V}(\hat{\boldsymbol{\pi}})$ and then compute the approximate 95% interval for d.

```
> # extract estimated cov matrix for fitted probs
> vHatPi <- predict( fit.12.DAP, se.fit=TRUE )$cov.fit.array[1,,]
> vHatPi
```

```
             Yes.Yes        Yes.No        No.Yes         No.No
Yes.Yes  0.0010865242 -0.0001780295 -0.0007399069 -0.0001685878
Yes.No  -0.0001780295  0.0003916785  0.0001082857 -0.0003219347
No.Yes  -0.0007399069  0.0001082857  0.0019696115 -0.0013379902
No.No   -0.0001685878 -0.0003219347 -0.0013379902  0.0018285126
```

```
> # compute a CI for d
> d <- piHat[3] - piHat[2]
```

```
> a <- c(0,-1,1,0 )
> dSE <- sqrt( t(a) %*% vHatPi %*% a )
> dLower <- d - qnorm(.975) * dSE
> dUpper <- d + qnorm(.975) * dSE
> c( dLower, dUpper )

[1] 0.08990081 0.27143701
```

The interval does not include zero, so we have strong evidence that $d \neq 0$. To investigate whether Y_1 and Y_2 are related, we could apply the delta method to compute a normal-theory interval for the odds ratio

$$\omega \;=\; \frac{\pi_{11}\pi_{22}}{\pi_{12}\pi_{21}}.$$

However, because the loglikelihood function rises as $\pi_{12} \to 0$, the estimate of ω is unstable and the normal approximation is poor. A better way to address this would be to simulate random draws from the Bayesian posterior distribution of ω using Markov chain Monte Carlo (MCMC), which does not rely on large-sample normal approximations. We will demonstrate the use of MCMC in Section 5.

4.2 Some details about the model

Model definition. For a noise-infused response, `rrLogit()` groups rows of the input dataset by their covariate patterns and tabulates the noisy response Y^* within each group. Let $\boldsymbol{x}_g$ $(p \times 1)$ denote the vector of covariates and $\boldsymbol{f}_g^* = (f_{g1}^*, \ldots, f_{gC}^*)^\top$ the noisy frequencies for group $g = 1, \ldots, G$. Treating the within-group sample size $N_g = \sum_{y=1}^{C} f_{gy}^*$ as fixed, we assume $\boldsymbol{f}_g^*$ is distributed as

$$\boldsymbol{f}_g^* \;\sim\; \text{Mult}(N_g, \boldsymbol{\pi}_g^*)$$

independently for $g = 1, \ldots, G$. The noisy response probabilities $\boldsymbol{\pi}_g^*$ relate to the true ones $\boldsymbol{\pi}_g = (\pi_{g1}, \ldots, \pi_{gC})^\top$ by

$$\boldsymbol{\pi}_g^* \;=\; \boldsymbol{T}\boldsymbol{\pi},$$

where $\boldsymbol{T}$ is a known full-rank perturbation matrix, and $\boldsymbol{\pi}_g$ depends on the covariates through the baseline-category logits

$$\eta_{gy} \;=\; \log\left(\frac{\pi_{gy}}{\pi_{gb}}\right) \;=\; \boldsymbol{x}_g^\top \boldsymbol{\beta}_y,$$

with $\boldsymbol{\beta}_b = \mathbf{0}$ for the baseline category b. The transformation from $\boldsymbol{\eta}_g = (\eta_{g1}, \ldots, \eta_{gC})^\top$ to $\boldsymbol{\pi}_g$ is

$$\pi_{gy} \;=\; \frac{\exp(\eta_{gy})}{\sum_{y'=1}^{C} \exp(\eta_{gy'})}.$$

The stacked vector of coefficients with $\boldsymbol{\beta}_g$ removed is denoted by $\boldsymbol{\beta}$, which has length $d = p(C-1)$.

Model fitting. By analogy to MOM, it might be tempting to estimate $\boldsymbol{\beta}$ by fitting a conventional baseline-category logit model to the noisy response Y^* and transforming the distorted coefficients to correct for the noise. Except in special cases, the relationship between $\boldsymbol{\pi}^*_g$ and $\boldsymbol{\eta}_g$ is not logistic, so that strategy does not work; we need fitting techniques specially designed for this situation.

The loglikelihood for this model is

$$l \;=\; \sum_{g=1}^{G} \sum_{y=1}^{C} f^*_{gy} \log \pi^*_{gy}. \tag{13}$$

Despite a superficial resemblance to the loglikelihood for a non-noisy response (11), this function can behave very differently over the parameter space $\boldsymbol{\beta} \in \mathbb{R}^d$. It may be oddly shaped and non-concave, especially in regions far from the maximum. If the perturbation matrix is nearly singular, which tends to happen if the rates of noise infusion from RR or PRAM are high, this function becomes nearly constant over a lower-dimensional subspace of $\mathbb{R}^d$. These unusual features may cause Newton-Raphson (NR) to fail. Fisher scoring (FS), a close cousin of NR, doesn't fare much better.

For maximizing (13), our default method is an expectation-maximization (EM) algorithm that treats the noise-free response Y as missing data. The procedure alternates between an expectation or E-step and a maximization or M-step. In the E-step, we replace each vector of unseen frequencies $\boldsymbol{f}_g$ by its conditional expected value given the observed noisy frequencies,

$$\hat{\boldsymbol{f}}_g \;=\; E(\boldsymbol{f}_g \,|\, \boldsymbol{f}^*_g, \boldsymbol{\beta}), \tag{14}$$

fixing the unknown $\boldsymbol{\beta}$ at its most recent estimate. In the M-step, we update the estimate for $\boldsymbol{\beta}$ by maximizing the loglikelihood function (11) for the non-noisy response, with the unseen $\boldsymbol{f}_g$'s replaced by their expected values $\hat{\boldsymbol{f}}_g$'s from the E-step. The M-step is carried out by NR, which for this purpose tends to work well. Alternating between the E- and M-steps creates a sequence of $\boldsymbol{\beta}$'s that converges to a local maximum of (13). This EM algorithm was previously described for binary response variables by Van den Hout (1999), Woo and Slavković (2012), and Blair et al. (2015a). Details of the EM implementation are given in B.

Predictive distribution for the true frequencies given the noisy ones. The expected true frequencies $\hat{\boldsymbol{f}}_g$ play a key role in model fitting and interpretation, so it is worthwhile to understand how we compute them. Imagine a two-way table as shown in Table 3 that cross-classifies the units from a single covariate pattern g by their values of Y^* and Y. The row totals $\boldsymbol{f}^*_g \;=\; (f^*_{g1}, \ldots, f^*_{gC})^\top$ in the last column are

Table 3: Frequencies in the hypothetical two-way table that classifies observational units in covariate pattern g by their values of the noisy response Y^* and the true response Y.

	Y				
Y^*	1	2	$\cdots$	C	Total
1	F_{g11}	F_{g12}	$\cdots$	F_{g1C}	f^*_{g1}
2	F_{g21}	F_{g22}	$\cdots$	F_{g2C}	f^*_{g2}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
C	F_{gC1}	F_{gC2}	$\cdots$	F_{gCC}	f^*_{gC}
Total	f_{g1}	f_{g2}	$\cdots$	f_{gC}	N_g

seen, but the interior frequencies F_{grs} and the column totals $\boldsymbol{f}_g = (f_{g1}, \ldots, f_{gC})^\top$ in the bottom row are unknown. Conditioning on the row totals and a given value for $\boldsymbol{\beta}$, the interior frequencies in this table have a product-multinomial distribution. Specifically, let $\boldsymbol{F}_{gr\cdot} = (F_{gr1}, \ldots, F_{grC})^\top$ denote the interior frequencies from row r. These frequencies are distributed as

$$\boldsymbol{F}_{gr\cdot} \,|\, f^*_{gr}, \boldsymbol{\beta} \sim \text{Mult}(f^*_{gr}, \boldsymbol{\phi}_{gr})$$

independently for $r = 1, \ldots, C$, and the probabilities $\boldsymbol{\phi}_{gr} = (\phi_{gr1}, \ldots, \phi_{grC})^\top$ come from Bayes' Theorem,

$$\phi_{grs} = \frac{t_{rs}\,\pi_{gs}}{\sum_{s'=1}^{C} t_{rs'}\,\pi_{gs'}}. \tag{15}$$

Therefore, given $\boldsymbol{f}^*_g$ and $\boldsymbol{\beta}$, the vector of true frequencies $\boldsymbol{f}_g$ is distributed as the sum of independent multinomial vectors; in a slight abuse of notation,

$$\boldsymbol{f}_g \,|\, \boldsymbol{f}^*_g, \boldsymbol{\beta} \sim \sum_{r=1}^{C} \text{Mult}(f^*_{gr}, \boldsymbol{\phi}_{gr}). \tag{16}$$

This distribution has expected value

$$E(\boldsymbol{f}_g \,|\, \boldsymbol{f}^*_g, \boldsymbol{\beta}) = \boldsymbol{\Phi}_g^\top \boldsymbol{f}^*_g,$$

where $\boldsymbol{\Phi}_g$ is the $C \times C$ matrix with (r, s)th element given by (15). In the PRAM literature, $\boldsymbol{\Phi}_g^\top$ has been called a calibration matrix and is sometimes written as $\boldsymbol{T}^{\leftarrow}$ (van den Hout and Elamir, 2006). Like $\boldsymbol{T}^{-1}$, it has the ability to transform observed frequencies or proportions from the noisy response into consistent estimates for the true response. In general, it is not an inverse of $\boldsymbol{T}$, and it depends both on $\boldsymbol{T}$ and the unknown true response prevalences $\boldsymbol{\pi}_g$ specific to covariate pattern g.

4.3 A hypothetical application of PRAM to real survey data

For this example, we rely on data from the National Health Interview Survey (NHIS), an annual cross-sectional study sponsored by the National Center for Health Statistics and conducted by the U.S. Census Bureau. NHIS collects information on health status and behaviors in a large representative sample of the non-institutionalized U.S. population. The dataset `cig2019`, which is distributed with the `rrLogit` package, includes items pertaining to self-reported cigarette smoking for nearly 32,000 adults interviewed in the 2019 NHIS. Basic demographic and geographic characteristics (age, sex, race/ethnicity, urbanicity, region) are included as well, along with a few variables that summarize the sampling design. For more information on this dataset, see `?cig2019`.

Key variable. This set of microdata was released to the public without added noise. Using the simple noise-infusion utilities included with `rrLogit`, we will apply PRAM to one key variable, a classifier of race and Hispanic origin with seven levels.[9] In the code below, we change the variable's name to `raceEth` and change its levels to short character strings.

`1="Hispanic"`: Hispanic, any race

`2="White"`: non-Hispanic White as sole race

`3="Black"`: non-Hispanic Black/African American as sole race

`4="Asian"`: non-Hispanic Asian as sole race

`5="AIAN"`: non-Hispanic American Indian/Alaska native as sole race

`6="AIAN+`: non-Hispanic American Indian/Alaska native plus at least one other race

`7="Other"`: all other single and multiple race categories

```
> # copy cig2019 into the local workspace and show variable names
> data(cig2019)
> names(cig2019)

 [1] "wtia_a"     "wtfa_a"     "urbrrl"     "region"     "pstrat"     "ppsu"
 [7] "agep_a"     "sex_a"      "hispallp_a" "smkev_a"    "smknow_a"   "cignow_a"
[13] "smk30d_a"   "cig30d_a"
```

[9] This illustrates an important practical difference between RR and PRAM. In RR, perturbed variables tend to be questions on sensitive topics that the respondent might be reluctant to answer truthfully. With PRAM, the perturbed variables tend to be common demographic and geographic descriptors that might be combined with external information to uniquely identify individuals.

```
> # rename the variable hispallp_a to raceEth and assign short
> # character strings for the levels
> raceEth <- cig2019$hispallp_a
> levels(raceEth) <- c("Hispanic", "White", "Black", "Asian",
+    "AIAN", "AIAN+", "Other")
> # display marginal frequencies
> table(raceEth)

raceEth
Hispanic    White    Black    Asian     AIAN    AIAN+    Other
    4152    21915     3483     1648      212      248      339
```

Notice that three of the categories are relatively rare, each comprising one percent or less of the sample. Disclosure risks could be reduced by collapsing this variable into fewer categories, but the data would then become useless for investigating certain questions, e.g., how patterns of tobacco use vary among the original groups. The minority categories are important for scientific, social and policy purposes, so we want to retain them if possible.

Other variables. The first question on cigarette use was, "Have you smoked at least 100 cigarettes in your ENTIRE LIFE?" If the response was "Yes," the participant was asked additional questions about past and current smoking; if "No," those questions were skipped. Here we combine the smoking items into a single variable `smokStat` classifying respondents as never smokers, former smokers, current non-heavy smokers, or current heavy smokers.

```
> # create a four-level categorization of smoking status
> #  1: never smoker
> #  2: former smoker
> #  3: current smoker, non-heavy
> #  4: current smoker, heavy
> # A heavy smoker is defined as someone who, on average, smokes 25 or more
> # cigarettes per day.
> smokStat <- integer( NROW(cig2019) )
> smokStat[] <- NA
> smokStat[ cig2019$smkev_a == "No"  ] <- 1L   # Never smoker
> smokStat[ ( cig2019$smkev_a == "Yes" ) &
+    ( cig2019$smknow_a == "Not at all" )   ] <- 2L   # Former smoker
> smokStat[ ( cig2019$smknow_a == "Every day" ) &
+    ( cig2019$cignow_a < 25 ) ] <- 3L        # Current smoker, non-heavy
> smokStat[ ( cig2019$smknow_a == "Some days" ) &
+    ( ( cig2019$smk30d_a / 30 ) * cig2019$cig30d_a < 25 ) ] <- 3L
> smokStat[ ( cig2019$smknow_a == "Every day" ) &
+    ( cig2019$cignow_a >= 25 ) ] <- 4L        # Current smoker, heavy
> smokStat[ ( cig2019$smknow_a == "Some days" ) &
+    ( ( cig2019$smk30d_a / 30 ) * cig2019$cig30d_a >= 25 ) ] <- 4L
> smokStat <- factor( smokStat )
> levels(smokStat) <- c("Never", "Former", "Current non-heavy",
+    "Current heavy" )
```

Other covariates that we will consider are

`metro`: urbanicity, with levels `1="Large central metro"`, `2="Large fringe metro"`, `3="Med/small metro"`, `4="Nonmetropolitan"`,

`region`: census region, with levels `1="Northeast"`, `2="Midwest"`, `3="South"`, `4="West"`,

`age`: in years,

`catAge`: categorized age, with levels `1="[18,25]"`, `1="(25,35]"`, `1="(35,45]"`, `1="(45,55]"`, `1="(55,65]"`, `1="(65,75]"`, `1="(75,85]"`, and

`sex`: with levels `1="Male"` and `2="Female"`.

```
> # get demographic and geographic variables
> metro <- cig2019$urbrrl
> region <- cig2019$region
> age <- cig2019$agep_a
> catAge <- cut( age, breaks=c(18,25,35,45,55,65,75,85), include.lowest=TRUE)
> sex <- cig2019$sex_a
```

Simplifications. Before proceeding, we will quickly bypass two issues that, if we handled them in a more careful way, would add unnecessary details and distract from our intended purpose. The first issue is missing values that appear in some variables. To handle these, we delete the incomplete cases, slightly reducing the sample size.

```
> # put variables into a new data frame
> df <- data.frame( raceEth, smokStat, metro, region, catAge, sex)
> # show percentage of missing values per variable
> round( 100 * apply( is.na(df), 2, FUN=mean ), 1 )

 raceEth smokStat    metro   region   catAge      sex
     0.0      2.8      0.0      0.0      0.3      0.0

> # eliminate rows with missing values
> incomplete <- apply( is.na(df), 1, any )
> df <- df[ ! incomplete, ]
> # get rid of the rownames that correspond to the original row numbers
> rownames(df) <- NULL
> # see how many rows there were originally, and how many are left
> NROW(cig2019)

[1] 31997

> NROW(df)

[1] 31032
```

The second issue pertains to the complex sample design. The `cig2019` dataset includes weights to adjust estimates for unequal probabilities of selection and nonresponse, and stratum and cluster identifiers for variance estimation. Incorporating those features would not be difficult, but to keep this presentation focused, we will ignore the design and treat the $N = 31{,}032$ respondents in our reduced sample as if they are a simple random sample from the target population. Applications with complex sample designs may be addressed in future versions of `rrLogit`.

Adding noise. To create a PRAMed version of `RaceEth`, we will use the perturbation matrix given by (5), setting the local DP privacy-loss parameter to $\epsilon = 1.5$. This matrix can be computed automatically using the function `rrPertMat()`.

```
> # perturbation matrix proposed by Wang, Wu and Hu (2013) with a
> # privacy-loss parameter of 1.5
> Tmat <- rrPertMat( privLoss=1.5, nLevels=7 )
> round(Tmat, 4)
```

```
      [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]
[1,] 0.4276 0.0954 0.0954 0.0954 0.0954 0.0954 0.0954
[2,] 0.0954 0.4276 0.0954 0.0954 0.0954 0.0954 0.0954
[3,] 0.0954 0.0954 0.4276 0.0954 0.0954 0.0954 0.0954
[4,] 0.0954 0.0954 0.0954 0.4276 0.0954 0.0954 0.0954
[5,] 0.0954 0.0954 0.0954 0.0954 0.4276 0.0954 0.0954
[6,] 0.0954 0.0954 0.0954 0.0954 0.0954 0.4276 0.0954
[7,] 0.0954 0.0954 0.0954 0.0954 0.0954 0.0954 0.4276
```

This noise mechanism discards a majority of the true values; it keeps the recorded category with probability $p \approx 0.43$, and otherwise replaces it with a category drawn from the other six with equal probability. To generate the noisy variable, we supply the true version to `rrPerturbResponse()` along with the perturbation matrix. This function expects the user to supply microdata, not a dataset with frequencies, because the noise is being applied at the level of individuals.

```
> # create a noisy version of raceEth and add it to the data frame
> set.seed(52274866)  # for reproducibility
> df$raceEth.noisy <- rrPerturbResponse( df$raceEth, Tmat )
```

Here we display the marginal percentages for the true and noisy versions, along with the MOM estimates from the noisy version.

```
> # compute percentages from original and noisy versions
> pct.orig <- 100 * table( df$raceEth ) / NROW(df)
> pct.noisy <- 100 * table( df$raceEth.noisy ) / NROW(df)
> # estimate original percentages from noisy version using MOM
> Tinv <- solve(Tmat)
> pct.MOM <- Tinv %*% pct.noisy
> # display percentages rounded to two decimal places
> result <- cbind( pct.orig, pct.noisy, pct.MOM )
> colnames(result) <- c("orig", "noisy", "MOM" )
> round( result, 2 )
```

```
           orig noisy   MOM
Hispanic 12.91 13.59 12.19
White    68.81 32.40 68.83
Black    10.66 12.93 10.19
Asian     5.12 11.20  4.99
AIAN      0.66  9.98  1.32
AIAN+     0.79  9.79  0.75
Other     1.05 10.12  1.73
```

Notice how the noise severely flattened the variable's distribution, but MOM brought it back to something close to the original; the difference between the MOM estimate and the original percentages reflects noise added by PRAM.

By repeatedly calling `rrPerturbResponse()`, it would be easy to run a simulation verifying that, over repetitions of PRAM, the average of the MOM estimates approximates the original sample percentages. Also, we could show that the variability of the MOM estimates over repeated PRAM is approximated by the covariance matrix (4). Simulations with repeated PRAM help a statistical agency study the impact of noise mechanisms and understand tradeoffs between disclosure risk and data utility. We will not pursue those here. Instead, we assume the perspective of a data user who has access to `raceEth.noisy` and the perturbation matrix but not `raceEth`, and we will show how to use this information and functions in `rrLogit` to make inferences about the population.

4.4 Building predictive models

Response and predictors. In an `rrLogit` model, the noise-infused variable always plays the role of response, and non-noisy variables serve as predictors. If we consider the ultimate goal of the data analysis, however, a variable distorted by PRAM is usually some characteristic like `raceEth` that will eventually become a predictor for one or more non-perturbed variables like `smokStat`. Using the data from `cig2019`, we may ultimately want to examine rates and patterns of smoking within classifications by age, sex and race, or look for differences in smoking by race and region. We will implement this role reversal in Section 6 through multiple imputation.

For now, we pursue the intermediate goal of using `raceEth.noisy` to build predictive models for `raceEth`, models that capture its relationships with `smokStat`, and other variables that will be examined later. For example, if we intend to analyze smoking by race and sex, we should consider predictive models that include `smokStat`, `sex`, and `smokStat:sex`. If we intend to look at smoking by race and region, the predictive models should have `smokStat`, `region`, and `smokStat:region`. Future analyses that do not involve race (e.g., smoking by urbanicity and region) are not impacted by noise infusion and do not need `rrLogit`. Given the wide range of possible future analyses, it is usually impractical and undesirable to create one comprehensive predictive model that can accommodate all of them; it is often better

to use smaller, more focused and well tuned models designed with specific analyses in mind.

Aggregating the data. Before proceeding, let's collapse our microdata with $N =$ 31,032 rows into narrow format with frequencies.

```
> # rename the data frame to dfMicro
> dfMicro <- df
> # aggregate df to narrow format with frequencies; each row will consist
> # of a unique combination of the noisy response and the predictors
> df$Freq <- 1
> df <- aggregate( Freq ~ raceEth.noisy + smokStat + metro + region +
+    catAge + sex, data=df, FUN=sum )
> # see how many rows there are now, and examine the first few rows
> NROW(df)
[1] 4237

> head(df)
  raceEth.noisy smokStat              metro    region  catAge  sex Freq
1      Hispanic    Never Large central metro Northeast [18,25] Male    7
2         White    Never Large central metro Northeast [18,25] Male   10
3         Black    Never Large central metro Northeast [18,25] Male   13
4         Asian    Never Large central metro Northeast [18,25] Male    8
5          AIAN    Never Large central metro Northeast [18,25] Male    7
6         Other    Never Large central metro Northeast [18,25] Male    6
```

This aggregation step is optional and has little impact on computational speed, because `rrLogit()` always groups data by covariate and response patterns before fitting any model. The main advantage of aggregating is that it sorts the rows of data by the levels of the covariates, which will make the results from `fitted()` and `residuals()` appear in their natural order for easier interpretation.

Handling failure to converge. Because `smokStat` is the future outcome variable, let's begin with a model that has `smokStat` as its only predictor.

```
> # fit model to aggregated data; don't forget to include the 'freq' argument
> fit.1 <- rrLogit( raceEth.noisy ~ smokStat, data=df, freq=Freq, pertMat=Tmat )
Estimate at or near boundary; standard errors may be unreliable
Warning message:
In rrLogit.formula(raceEth.noisy ~ smokStat, data = df, freq = Freq,  :
  Procedure failed to converge by iteration 1000
```

The boundary message tells us that some of the estimated $\hat{\pi}_{gy}$'s have gotten very close to zero. The warning about failure to converge means that, even after 1,000 iterations, some of the $\hat{\pi}_{gy}$'s have still not stabilized. EM can be painfully slow to converge, especially when noise-infusion rates are high. To achieve convergence, we can re-run the procedure while increasing the control parameter `iterMaxEM` from its default value of 1,000 to something larger, like this.

```
> fit.1 <- rrLogit( raceEth.noisy ~ smokStat, data=df, freq=Freq, pertMat=Tmat,
+    control=list(iterMaxEM=5000) )

Estimate at or near boundary; standard errors may be unreliable
```

The convergence warning has disappeared, and the output from `summary(fit.1)` (not shown) tells us that EM converged at iteration 1,324. A disadvantage of this method is that EM restarts from the beginning, sacrificing the progress already made in the first 1,000 steps. A more efficient strategy would have been to take the object created by `rrLogit()` and feed it to the function again, like this:

```
> fit.1 <- rrLogit( fit.1 )
```

In this approach, `rrLogit()` takes the parameter estimates in `fit.1` as starting values and continues to run EM, reaching convergence at iteration 324. If you call `rrLogit()` with an `rrLogit` object as its first argument, the data, model, prior distribution, parameter estimates, control parameters, etc. are automatically transferred to the new call, and unless a new `method` is provided, whatever procedure was being done in the old call is continued in the new one. Roughly speaking, calling `rrLogit()` on an `rrLogit` object means, "Carry on; do more of the same, unless I tell you otherwise." If EM fails to converge, you can also feed the `rrLogit` object back into `rrLogit()` but change the method to Newton-Raphson, as shown below.

```
> fit.1 <- rrLogit( fit.1, method="NR" )
```

NR converges much faster than EM, but it is less stable and can easily fail unless its starting values are already close to the maximum. For more discussion about this, see B.

Handling estimates at the boundary. In conventional logistic regression with a non-noisy response, estimates on a boundary happen when covariate patterns contain zero frequencies. For this example, however, the frequency table for `smokStat` by `raceEth.noisy` has no zeros.

```
> # display the contingency table for smokStat by raceEth.noisy
> xtabs( Freq ~ smokStat + raceEth.noisy, data=df)

                 raceEth.noisy
smokStat          Hispanic White Black Asian AIAN AIAN+ Other
  Never               2795  5662  2526  2220 1865  1828  1988
  Former               944  2944   933   808  799   802   730
  Current non-heavy    455  1350   521   427  405   389   399
  Current heavy         23    99    31    20   28    19    22
```

The nature of the problem becomes apparent when we examine the fitted values for the unseen true response.

```
> # display fitted means, rounded to one decimal place
> fitted( fit.1, type="mean", include.predvars=TRUE, digits=1 )

          smokStat Hispanic   White  Black  Asian  AIAN AIAN+ Other
1            Never   2990.6 11621.7 2180.8 1259.5 190.8  79.5 561.1
2           Former    544.1  6540.7  511.1  136.4 109.4 118.4   0.0
3 Current non-heavy    236.4  2930.8  435.1  152.1  85.9  37.7  67.8
4     Current heavy      0.0   212.8   18.9    0.0  10.3   0.0   0.0
```

The dataset has relatively few current heavy smokers, and the fitted model estimates that none of them belong to the `Hispanic`, `Asian`, `AIAN+` or `Other` categories. Also, the fitted model is finding no former smokers in the category `Other`. We could try to resolve these problems by combining the current heavy smokers with current non-heavy smokers, or by collapsing `Other` into another racial group, but that would preclude future analyses targeting those subpopulations. Instead of collapsing, we will keep the existing categories and apply a prior distribution. With a little experimentation, we found that a very mild DAP with `priorFreqTot=7` and `priorAlloc=rep(1/7,7)` moves the estimate away from the boundary with very little change in the loglikelihood relative to the ML solution. This prior adds just one fictitious person to each of the seven categories of `raceEth`.

```
> # apply a mild prior that moves the estimates away from the boundary
> fit.2 <- rrLogit( raceEth.noisy ~ smokStat, data=df, freq=Freq, pertMat=Tmat,
+    prior="DAP", priorFreqTot=7, priorAlloc=rep(1/7,7) )
> # Bayes factor comparing posterior mode to the ML estimate
> exp( loglik(fit.1) - loglik(fit.2) )

[1] 3.037522

> # display new fitted means
> fitted( fit.2, type="mean", include.predvars=TRUE, digits=1 )

          smokStat Hispanic   White  Black  Asian  AIAN AIAN+ Other
1            Never   2981.4 11599.9 2173.2 1254.6 204.0 109.9 561.1
2           Former    539.4  6514.8  506.7  141.8 117.4 125.4  14.5
3 Current non-heavy    234.5  2913.3  430.7  152.6  90.5  50.1  74.5
4     Current heavy      4.3   199.8   17.8    2.7  11.3   2.4   3.7
```

Fitting richer predictive models. The current model has `smokStat` as its only predictor. If we intend to analyze rates of smoking by race/ethnicity and other variables, those variables should be included with sufficient interactions to support the future analyses. For example, a model with terms `smokStat*sex` allows the relationship between race/ethnicity and smoking status to vary by sex, and can therefore support estimation of smoking rates within a classification by race/ethnicity and sex. For illustration, we now fit two predictive models:

Model A: `smokStat*sex*catAge`, which can support estimation of smoking rates within classes of race/ethnicity $\times$ sex $\times$ age, and

Model B: `smokStat*region*metro`, which can support estimation of smoking rates within classes of race/ethnicity $\times$ region $\times$ urbanicity.

For both of these models, the EM algorithm converges slowly. In each case, we run EM for 1,000 iterations to get close to the mode, and then switch to NR which reaches a solution very quickly.

```
> # fit Model A, using the same data-augmentation prior as before
> fit.A <- rrLogit( raceEth.noisy ~ smokStat * sex * catAge,
+    data=df, freq=Freq, pertMat=Tmat,
+    prior="DAP", priorFreqTot=7, priorAlloc=rep(1/7,7) )

Warning message:
In rrLogit.formula(raceEth.noisy ~ smokStat * sex * catAge, data = df,  :
  Procedure failed to converge by iteration 1000

> # switch to Newton-Raphson for faster convergence
> fit.A <- rrLogit(fit.A, method="NR")
> # fit Model B, using the same data-augmentation prior as before
> fit.B <- rrLogit( raceEth.noisy ~ smokStat * region * metro,
+    data=df, freq=Freq, pertMat=Tmat,
+    prior="DAP", priorFreqTot=7, priorAlloc=rep(1/7,7) )

Warning message:
In rrLogit.formula(raceEth.noisy ~ smokStat * region * metro, data = df,  :
  Procedure failed to converge by iteration 1000

> # switch to Newton-Raphson for faster convergence
> fit.B <- rrLogit(fit.B, method="NR")
```

Fitting predictive models with the saturated option. Both of these models include the full set of interactions, making them the richest possible models that could be fit with their respective sets of predictors. In each case, we were able to fit the model successfully because there happened to be no zero frequencies in the table that cross-classifies the individuals by the predictors.

```
> # see if there are any zeros in the full cross-classification by
> # predictors in Model A: smokStat, sex, and catAge
> any( xtabs( Freq ~ smokStat + sex + catAge, data=df ) == 0 )

[1] FALSE

> # see if there are any zeros in the full cross-classification by
> # predictors in Model B: smokStat, region and metro
> any( xtabs( Freq ~ smokStat + region + metro, data=df ) == 0 )

[1] FALSE
```

If any zeros had been present, the model fitting would have failed and given the following error message,

```
modelMatrix does not have full rank due to empty covariate patterns
```

which implies that some of the elements of β are inestimable. When that happens, we can apply the argument `saturated=TRUE`, as described in Section 3.5. With `saturated=TRUE`, `rrLogit()` skips over the empty covariate patterns and fits an unrestricted multinomial model to each non-empty pattern, bypassing the model matrix and β entirely.

```
> # fit the same models as before, but this time use the option saturated=TRUE
> fit.A.sat <- rrLogit( raceEth.noisy ~ smokStat * sex * catAge,
+    data=df, freq=Freq, pertMat=Tmat,
+    prior="DAP", priorFreqTot=7, priorAlloc=rep(1/7,7), saturated=TRUE )

Warning message:
In rrLogit.formula(raceEth.noisy ~ smokStat * sex * catAge, data = df,  :
  Procedure failed to converge by iteration 1000


> # continue running EM to convergence
> fit.A.sat <- rrLogit( fit.A.sat )
> fit.B.sat <- rrLogit( raceEth.noisy ~ smokStat * region * metro,
+    data=df, freq=Freq, pertMat=Tmat,
+    prior="DAP", priorFreqTot=7, priorAlloc=rep(1/7,7), saturated=TRUE )

Warning message:
In rrLogit.formula(raceEth.noisy ~ smokStat * region * metro, data = df,  :
  Procedure failed to converge by iteration 1000


> # continue running EM to convergence
> fit.B.sat <- rrLogit(fit.B.sat)
```

Using `saturated=TRUE` has no effect on model fit.

```
> # show that the models fit with saturated=TRUE achieve the same
> # loglikelihood as those fit with saturated=FALSE
> all.equal( loglik(fit.A), loglik(fit.A.sat) )

[1] TRUE

> all.equal( loglik(fit.B), loglik(fit.B.sat) )

[1] TRUE
```

4.5 Model diagnostics with a noisy response

Comparing the fit of alternative models. In Section 3.5, we showed how to use `anova()` for model comparisons when the response variable has no added noise. With a noisy response, `anova()` works in exactly the same way. To illustrate, let's compare the fit of our Model B (`smokStat*sex*catAge`) to a reduced version that includes all two-way interactions but eliminates the three-way terms.

```
> # fit a reduced model that eliminates the three-way interaction
> fit.B.all2way <- rrLogit( raceEth.noisy ~ smokStat*region +
+   smokStat*metro + region*metro,  data=df, freq=Freq, pertMat=Tmat,
+   prior="DAP", priorFreqTot=7, priorAlloc=rep(1/7,7) )

Warning message:
In rrLogit.formula(raceEth.noisy ~ smokStat * region + smokStat *  :
  Procedure failed to converge by iteration 1000

> # switch to NR for faster convergence
> fit.B.all2way <- rrLogit( fit.B.all2way, method="NR")
> # compare the fit
> anova( fit.B.all2way, fit.B.sat, pval=TRUE )

Model 1: raceEth.noisy ~ smokStat * region + smokStat * metro + region *
Model 2:     metro (saturated)
Model 1: raceEth.noisy ~ smokStat + region + metro
  nParams -2*loglik  df change pval
1     222    112973
2     384    112922 162 51.124    1
```

The three-way interactions add 162 parameters to the model but only decrease the deviance by 51.1, suggesting that these terms are unnecessary. It may be tempting to remove them, but in this case, that should be discouraged. Collectively discarding all of these terms might omit some that are actually important. The purpose of predictive modeling is not to find a parsimonious description of the data but to preserve relationships for subsequent analyses after multiple imputation. Noise infusion reduces the chance that real effects are deemed statistically significant, increasing the risk of Type-2 errors if the terms are trimmed away. As a general principle, it is wise to keep interactions in predictive models even when they are not statistically significant, because eliminating them could make it impossible to detect effects of interest later.

Fitted values. With the `fitted()` function, we can extract fitted values for either the true response Y (`noisy=FALSE`) or the noisy response Y^* (`noisy=TRUE`). Definitions of the various types of fitted values are shown in Table 4.

```
> # extract fitted means for the true response (noisy=FALSE) from Model A
> fits.A <- fitted( fit.A, include.predvars=TRUE, type="mean")
> # print the first few rows
> print( head(fits.A), digits=2)
```

Table 4: Definitions of fitted-value vectors returned by the `fitted()` method for a given covariate pattern g.

	noisy=FALSE	noisy=TRUE
type="prob"	$\hat{\boldsymbol{\pi}}_g = \boldsymbol{x}_g^\top \hat{\boldsymbol{\beta}}$	$\hat{\boldsymbol{\pi}}^*_g = \boldsymbol{T}\hat{\boldsymbol{\pi}}_g$
type="mean"	$\hat{\boldsymbol{\mu}}_g = N_g \hat{\boldsymbol{\pi}}_g$	$\hat{\boldsymbol{\mu}}^*_g = N_g \hat{\boldsymbol{\pi}}^*_g$
type="link"	$\hat{\boldsymbol{\eta}}_g = \boldsymbol{x}_g^\top \hat{\boldsymbol{\beta}}$	—

```
           smokStat  sex  catAge Hispanic White   Black   Asian   AIAN AIAN+  Other
1             Never Male [18,25]   218.42 490.2 152.724  90.055  0.958  2.74 21.862
2            Former Male [18,25]     3.72  60.0   9.435   6.525 15.340  1.42  6.525
3 Current non-heavy Male [18,25]     7.80 104.9   2.440   4.986  2.440  0.41  4.986
4     Current heavy Male [18,25]     0.71   2.5   0.032   0.032  0.032  0.71  0.032
5             Never Male (25,35]   259.65 803.8 151.461 160.473  7.121 25.89 43.579
6            Former Male (25,35]    63.73 296.8  14.509  25.978 14.509  2.26  0.175
```

Residuals. With noise infusion, the Pearson residuals are

$$R^*_{gy} = \frac{f^*_{gy} - \hat{\mu}^*_{gy}}{\sqrt{\hat{\mu}^*_{gy}}}. \tag{17}$$

Note that these residuals are defined with respect to the noisy response Y^*. There are no analogous residuals for the true response variable Y, because the true frequencies f_{gy} are unseen. Let's examine the Pearson residuals for Model B.

```
> # extract residuals from Model B
> resids <- residuals( fit.B )
> # display the first few rows
> round( head(resids), digits=2)
```

```
     Hispanic White Black Asian  AIAN AIAN+ Other
[1,]     0.08  0.10  0.08  0.07 -0.34 -0.16  0.05
[2,]     0.27  0.37  0.27  0.24 -1.56  0.18 -0.20
[3,]     0.14  0.20  0.15  0.13  0.08 -1.06  0.12
[4,]    -0.45  0.70  0.70 -0.45 -0.45 -0.45 -0.45
[5,]     0.28  0.47  0.28  0.27 -2.31  0.22  0.25
[6,]     0.56  1.15  0.51  0.09  0.53 -1.52 -2.60
```

Curiously, these residuals are not zero even though the model is saturated. There are two reasons for this. One reason is that we applied a DAP, which gently pulls the parameters away from their ML estimates. The second, and more important, reason is that the ML estimate for this example lies on the boundary of the parameter space. It would be impossible for this fitted values $\hat{\mu}^*_{gy}$ to exactly reproduce the noisy frequencies f^*_{gy} unless some of the $\hat{\pi}^*_{gy}$'s strayed into the negative range, which the fitting procedure does not allow them to do.

4.6 Measures of missing information

The EM algorithm alternates between an E-step, which predicts the true response frequencies from the noisy ones, and an M-step, which maximizes a hypothetical loglikelihood function that supposes the true frequencies are known. If we compare the curvature of the hypothetical loglikelihood being maximized within the M-step to that of the actual loglikelihood, we can compute diagnostics that describe the impact of added noise on measures of uncertainty for the model coefficients and predicted values. These diagnostics, which are defined in Appendix C, include the fraction of missing information (FMI) and the width-inflation factor (WIF).

To compute these diagnostics for β, we must extract the estimated "complete-data covariance matrix," which approximates the covariance matrix that would have resulted if there were no added noise. To do this, we call `coef()` with the options `covMat=TRUE` and `completeData=TRUE`.

```
> # extract the complete-data covariance matrix from Model A
> tmp <- coef( fit.A, covMat=TRUE, completeData=TRUE )

Error in coef.rrLogit(fit.A, covMat = TRUE, completeData = TRUE) :
  completeData = TRUE may only be used when method="EM"
```

The error message appeared because we had sped the convergence of our model by switching from EM to NR. To fix the problem, we can restart the EM algorithm by calling `rrLogit()` again with `method="EM"`. This new run of EM stops after one iteration, because the procedure has already converged.

```
> # restart EM, which will automatically stop after a single iteration
> fit.A <- rrLogit( fit.A, method="EM")
> # extract the complete-data covariance matrix and corresponding standard errors
> tmp <- coef( fit.A, covMat=TRUE, completeData=TRUE )
> covMatCompleteData <- tmp$covMat
> SEsCompleteData <- sqrt( diag( covMatCompleteData ) )
> # extract the actual covariance matrix and corresponding standard errors
> tmp <- coef( fit.A, covMat=TRUE )
> covMatActual <- tmp$covMat
> SEsActual <- sqrt( diag( covMatActual ) )
```

We now compute and summarize the WIFs, which are equal to the actual SEs divided by the complete-data SEs.

```
> WIF <- SEsActual / SEsCompleteData
> summary(WIF)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.155   2.560   4.495   4.483   6.069  10.167
```

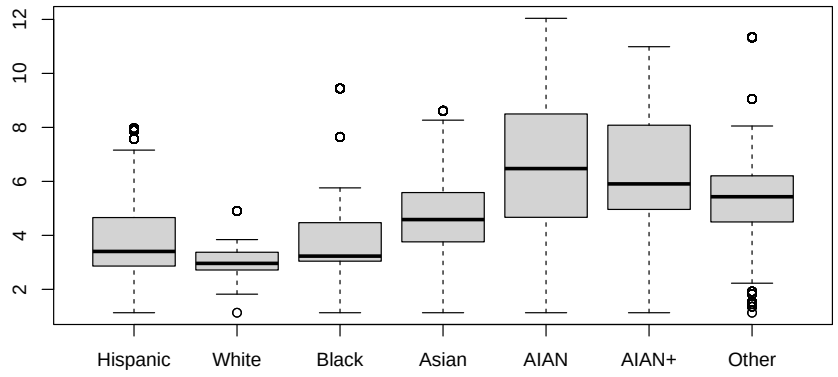

Figure 1: Boxplots of the width-inflation factors (WIFs) for predicted probabilities from Model A

Among the 336 coefficients in this model, noise infusion inflated the standard errors by an average factor of 4.5, with a minimum of 1.2 and a maximum of 10.2.

The `completeData=TRUE` option can also be used with `predict()`, which allows us to compute WIFs for the predicted probabilities.

```
> # compute WIFs for the predicted probabilities
> SEFitCompleteData <- predict(fit.A, se.fit=TRUE, completeData=TRUE)$se.fit
> SEFitActual <- predict(fit.A, se.fit=TRUE)$se.fit
> WIF <- SEFitActual / SEFitCompleteData
> FMI <- 1 - 1 / WIF^2
```

A boxplot of these WIFs, produced with

```
> boxplot(WIF)
```

and displayed in Figure 1, shows that the impact of noise infusion on measures of error is lowest in categories of race/ethnicity that are common and highest in the categories that are rare.

5 Bayesian methods

6 Multiple imputation

References

Abowd, J. M. (2018). The US Census Bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867.

Abowd, J. M., Ashmead, R., Cumings-Menon, R., Garfinkel, S., Heineck, M., Heiss, C., Johns, R., Kifer, D., Leclerc, P., Machanavajjhala, A., Moran, B., Sexton, W., Spence, M., and Zhuravlev, P. (2022). The 2020 census disclosure avoidance system topdown algorithm. *Harvard Data Science Review*. Special Issue 2: Differential Privacy for the 2020 U.S. Census.

Agresti, A. (2013). *Categorical Data Analysis, Third Edition*. John Wiley & Sons, Hoboken, NJ.

Albert, A. and Anderson, J. L. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, 71:1–10.

Ayed, F., Battiston, M., and Camerlenghi, F. (2020). An information theoretic approach to post randomization methods under differential privacy. *Statistics and Computing*, 30(5):1347–1361.

Bedrick, E. J., Christensen, R., and Johnson, W. (1996). A new perspective on priors for generalized linear models. *Journal of the American Statistical Association*, 91(436):1450–1460.

Bishop, Y. M., Fienberg, S. E., and Holland, P. W. (1975). *Discrete Multivariate Analysis: Theory and Practice*. MIT Press, Cambridge, MA.

Blair, G., Imai, K., and Zhou, Y.-Y. (2015a). Design and analysis of the randomized response technique. *Journal of the American Statistical Association*, 110:1304–1319.

Blair, G., Imai, K., and Zhou, Y.-Y. (2015b). *rr: Statistical Methods for the Randomized Response*. R package version.

Bock, R. D. (1975). *Multivariate statistical methods in behavioral research*. McGraw-Hill, New York.

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA.

Chaudhuri, A. (2010). *Randomized response and indirect questioning techniques in surveys*. CRC Press, Boca Raton, FL.

Chaudhuri, A., Christofides, T. C., and Rao, C. R. (2016). *Data gathering, analysis and protection of privacy through randomized response techniques: Qualitative and quantitative human traits*. Elsevier.

Chen, L. and Savalei, V. (2021). Three sample estimates of fraction of missing information from full information maximum likelihood. *Frontiers in Psychology*, 12:667802.

Chen, T. T. (1979). Analysis of randomized response as purposively misclassified data. In *Proceedings of the Section on Survey Research Methods*, pages 158–163. American Statistical Association.

Chua, T. C. and Fuller, W. A. (1987). A model for multinomial response error applied to labor flows. *Journal of the American Statistical Association*, 82(397):46–51.

Clogg, C. C., Rubin, D. B., Schenker, N., Schultz, B., and Weidman, L. (1991). Multiple imputation of industry and occupation codes in census public-use samples using Bayesian logistic regression. *Journal of the American Statistical Association*, 86(413):68–78.

Cox, L. H., Karr, A. F., and Kinney, S. K. (2011). Risk-utility paradigms for statistical disclosure limitation: How to think, but not how to act. *International Statistical Review*, 79(2):160–183.

De Waal, T., van Delden, A., and Scholtus, S. (2019). Quality measures for multisource statistics. *Statistical Journal of the IAOS*, 35(2):179–192.

de Wolf, P.-P., Gouweleeuw, J. M., Kooiman, P., and Willenborg, L. C. (1998). Reflections on PRAM. Technical report, Department of Statistical Methods, Statistics Netherlands.

de Wolf, P.-P. and van Gelder, I. (2004). An empirical evaluation of pram. Technical Report Discussion paper 04012, Statistics Netherlands, Voorburg/Heerlen.

Domingo-Ferrer, J. and Torra, V. (2001). A quantitative comparison of disclosure control methods for microdata. In Lane, J. I., Doyle, P., Zayatz, L., and Theeuwes, J., editors, *Confidentiality, disclosure and data access: theory and practical applications for statistical agencies*, pages 111–134. North-Holland, Amsterdam.

Duncan, G. T., Elliot, M., and Salazar, G. J.-J. (2011). *Statistical confidentiality: principles annd practice*. Springer Science+Business Media, New York.

Dwork, C. (2006). Differential privacy. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*, pages 1–12. Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407.

Fox, J.-P., Klotzke, K., and Veen, D. (2021). *GLMMRR: Generalized Linear Mixed Model (GLMM) for Binary Randomized Response Data*. R package version 0.5.0.

Gouweleeuw, J. M., Kooiman, P., and de Wolf, P.-P. (1998). Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14(4):463–478.

Greenland, S. (1988). Variance estimation for epidemiologic effect estimates under misclassification. *Statistics in Medicine*, 7(7):745–757.

Hawes, M. B. (2020). Implementing differential privacy: Seven lessons from the 2020 united states census. *Harvard Data Science Review*, 2(2).

Heck, D. W. and Moshagen, M. (2021). *RRreg: Correlation and Regression Analyses for Randomized Response Data*. R package version 0.7.3.

Horn, R. A. and Johnson, C. R. (2012). *Matrix Analysis, Second Edition*. Cambridge University Press, New York.

Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E. S., Spicer, K., and De Wolf, P.-P. (2012). *Statistical disclosure control*. John Wiley & Sons, Chichester, UK.

Jann, B. (2005). *RRLOGIT: Stata module to estimate logistic regression for randomized response data*. revides 12 May 2011.

Jeffreys, H. (1961). *The Theory of Probability, Third Edition*. Oxford University Press, Oxford, UK.

Jennrich, R. I. and Sampson, P. (1976). Newton-raphson and related algorithms for maximum likelihood variance component estimation. *Technometrics*, 18(1):11–17.

Karr, A. F., Kohnen, C. N., Oganian, A., Reiter, J. P., and Sanil, A. P. (2006). A framework for evaluating the utility of data altered to protect confidentiality. *The American Statistician*, 60(3):224–232.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.

Kooiman, P., Willenborg, L. C., and Gouweleeuw, J. M. (1997). PRAM: A method for disclosure limitation of microdata. Technical report, Department of Statistical Methods, Statistics Netherlands.

Kuha, J. and Skinner, C. J. (1997). Categorical data analysis and misclassification. In Lyberg, L. E., Biemer, P., Collins, M., De Leeuw, E. D., Dippo, C., Schwarz, N., and Trewin, D., editors, *Survey Measurement and Process Quality*, pages 633–670. John Wiley & Sons, New York.

Lesaffre, E. and Albert, A. (1989). Partial separation in logistic discrimination. *Journal of the Royal Statistical Society: Series B (Methodological)*, 51(1):109–116.

Little, R. J. and Rubin, D. B. (2019). *Statistical analysis with missing data, Third edition*, volume 793. John Wiley & Sons.

Lumley, T. (2010). *Complex Surveys: A Guide to Analysis Using R*. John Wiley & Sons, Hoboken, NJ.

Magnac, T. and Visser, M. (1999). Transition models with measurement errors. *Review of Economics and Statistics*, 81(3):466–474.

Marés, J. and Shlomo, N. (2014). Data privacy using an evolutionary algorithm for invariant PRAM matrices. *Computational Statistics & Data Analysis*, 79:1–13.

Matthews, G. J. and Harel, O. (2011). Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Surveys*, 5:1–29.

Nayak, T. K. (2021). A review of rigorous randomized response methods for protecting respondent's privacy and data confidentiality. In Arnold, B. C., Balakrishnan, N., and Coelho, C. A., editors, *Methodology and Applications of Statistics: A Volume in Honor of C.R. Rao on the Occasion of his 100th Birthday*, pages 319–341. Springer International Publishing.

Nayak, T. K., Zhang, C., and You, J. (2018). Measuring identification risk in microdata release and its control by post-randomisation. *International Statistical Review*, 86(2):300–321.

Orchard, T. and Woodbury, M. A. (1972). A missing information principle: theory and applications. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*, volume 6, pages 697–716. University of California Press.

Reiter, J. P. (2005). Estimating risks of identification disclosure in microdata. *Journal of the American Statistical Association*, 100(472):1103–1112.

Rubin, D. B. (1987). *Multiple imputation for nonresponse in surveys*. John Wiley & Sons, New York.

Savalei, V. and Rhemtulla, M. (2012). On obtaining estimates of the fraction of missing information from full information maximum likelihood. *Structural Equation Modeling: A Multidisciplinary Journal*, 19(3):477–494.

Shlomo, N. and Skinner, C. (2010). Assessing the protection provided by misclassification-based disclosure limitation methods for survey microdata. *The Annals of Applied Statistics*, 4(3):1291–1310.

Skinner, C. J. and Elliot, M. J. (2002). A measure of disclosure risk for microdata. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):855–867.

Templ, M. (2017). *Statistical disclosure control for microdata*. Springer, Vienna.

Templ, M., Kowarik, A., and Meindl, B. (2015). Statistical disclosure control for micro-data using the r package sdcmicro. *Journal of Statistical Software*, 67:1–36.

Van den Hout, A. (1999). *The analysis of data perturbed by PRAM*. WBBM Report Series 45, IOS Press, Amsterdam.

van den Hout, A. and Elamir, E. A. (2006). Statistical disclosure control using post randomisation: Variants and measures for disclosure risk. *Journal of Official Statistics*, 22(4):711–731.

van den Hout, A. and van der Heijden, P. G. (2007). Randomized response, statistical disclosure control and misclassification: a review. *International Statistical Review*, 70(2):269–288.

Wang, Y., Wu, X., and Hu, D. (2016). Using randomized response for differential privacy preserving data collection. In *Proceedings of EDBT/ICDT Workshops*, volume 1558.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60:63–69.

Willenborg, L. and De Waal, T. (2012). *Elements of statistical disclosure control*. Springer Science & Business Media, New York.

Woo, Y. M. J. and Slavković, A. B. (2012). Logistic regression with variables subject to post randomization method. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2012. Proceedings*, pages 116–130. Springer.

Zhang, C. and Nayak, T. K. (2021). Post-randomization for controlling identification risk in releasing microdata from general surveys. *Journal of Applied Statistics*, 48(3):455–470.

A Model fitting with no added noise

Grouping by covariate patterns. Regardless of how the data are provided (wide format, narrow format, or microdata), the `rrLogit()` function processes the data prior to model fitting by grouping the lines of the dataset into covariate patterns, classifying them into groups $g = 1, \ldots, G$ where the units within a group are identical with respect to their covariate vectors $\boldsymbol{x}_i$. The grouped data can be written as $(\boldsymbol{x}_g, \boldsymbol{f}_g)$ for $g = 1, \ldots, G$, where $\boldsymbol{f}_g = (f_{g1}, \ldots, f_{gC})^\top$, and f_{gy} is the frequency of $Y_i = y$ within group g. Regarding the group totals $N_g = \sum_{y=1}^{C}$ as fixed, $\boldsymbol{f}_g$ has a multinomial distribution

$$\boldsymbol{f}_g \,|\, \boldsymbol{\pi}_g \;\sim\; \text{Mult}(N_g, \boldsymbol{\pi}_g)$$

independently for $g = 1, \ldots, G$. The loglikelihood function with grouped data is $l(\boldsymbol{\beta}) = \sum_{g=1}^{G} l_g$, where

$$l_g \;=\; \sum_{y=1}^{C} f_{gy} \, \log \pi_{gy} \tag{18}$$

is the loglikelihood contribution for group g.

Newton-Raphson (NR). One iteration of NR updates the current estimate $\hat{\boldsymbol{\beta}}^{(t)}$ by

$$\hat{\boldsymbol{\beta}}^{(t+1)} \;=\; \hat{\boldsymbol{\beta}}^{(t)} \;+\; \left[-\left(\frac{\partial^2 l}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \right) \right]^{-1} \left(\frac{\partial l}{\partial \boldsymbol{\beta}} \right), \tag{19}$$

where the derivatives on the right-hand side are evaluated at $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}^{(t)}$. The first derivatives are $\partial l / \partial \boldsymbol{\beta} = \sum_{g=1}^{G} \partial l_g / \partial \boldsymbol{\beta}$, where

$$\frac{\partial l_g}{\partial \boldsymbol{\beta}} \;=\; \left(\frac{\partial \boldsymbol{\pi}_g}{\partial \boldsymbol{\beta}^\top} \right)^\top \left(\frac{\partial l_g}{\partial \boldsymbol{\pi}_g} \right), \tag{20}$$

and the second derivatives are $\partial^2 l / \partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top = \sum_{g=1}^{G} \partial^2 l_g / \partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top$, where

$$\frac{\partial^2 l_g}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \;=\; \sum_{y=1}^{C} \left(\frac{\partial l_g}{\partial \pi_{gy}} \right) \left(\frac{\partial^2 \pi_{gy}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \right) \;+\; \left(\frac{\partial \boldsymbol{\pi}_g}{\partial \boldsymbol{\beta}^\top} \right)^\top \left(\frac{\partial^2 l_g}{\partial \boldsymbol{\pi}_g \partial \boldsymbol{\pi}_g^\top} \right) \left(\frac{\partial \boldsymbol{\pi}_g}{\partial \boldsymbol{\beta}^\top} \right). \tag{21}$$

The derivatives of l_g with respect to $\boldsymbol{\pi}_g$ have elements $\partial l_g / \partial \pi_{gy} = f_{gy} / \pi_{gy}$ and

$$\frac{\partial^2 l_g}{\partial \pi_{gy} \partial \pi_{gy'}} \;=\; -\left(\frac{f_{gy}}{\pi_{gy}^2} \right) I(y = y'),$$

where $I(\cdot)$ is the indicator function equal to one if its argument is true and zero

otherwise. The derivatives of $\boldsymbol{\pi}_g$ with respect to $\boldsymbol{\beta}$ have elements

$$\frac{\partial \pi_{gy}}{\partial \beta_{jk}} \;=\; \pi_{gy}\,[\,I(k=y)\;-\;\pi_{gk}\,]\,x_{gj}, \tag{22}$$

$$\begin{aligned}\frac{\partial^2 \pi_{gy}}{\partial \beta_{jk}\partial \beta_{lm}} \;=\; &-\pi_{gy}\,\Big\{\pi_{gk}\,[\,I(k=m)\;-\;\pi_{gm}\,] \\ &-\;[\,I(k=y)\;-\;\pi_{gk}\,]\,[\,I(m=y)\;-\;\pi_{gm}\,]\Big\}\,x_{gj}\,x_{gl}.\end{aligned} \tag{23}$$

Plugging these expressions into (20)–(21) and simplifying, one can show that the first derivatives become

$$\frac{\partial l_g}{\partial \beta_{jk}} \;=\; (f_{gk}\;-\;N_g\pi_{gk})\;x_{gj}, \tag{24}$$

and the second derivatives become

$$\frac{\partial^2 l_g}{\partial \beta_{jk}\partial \beta_{lm}} \;=\; -\,N_g\,\pi_{gk}\,(1-\pi_{gk})\,x_{gj}\,x_{gl} \tag{25}$$

if $k = m$, and

$$\frac{\partial^2 l_g}{\partial \beta_{jk}\partial \beta_{lm}} \;=\; N_g\,\pi_{gk}\,\pi_{gm}\,x_{gj}\,x_{gl} \tag{26}$$

if $k \neq m$. In the `rrLogit()` function, convergence is judged not by changes in $\boldsymbol{\beta}$, but with respect to the fitted probabilities; when the largest absolute difference in the estimated π_{gy}'s from one iteration to the next falls below the control parameter `critConverge`, NR is deemed to have converged.

Fisher scoring. For statistical modeling, a popular alternative to NR is Fisher scoring (FS), which replaces the Hessian matrix $\partial^2 l/\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^\top$ by its expected value. In this case, the second derivatives (25)–(26) do not depend on the response frequencies $\boldsymbol{f}_g$ (except through N_g, $g = 1,\ldots,G$ which are treated as fixed), so the actual and expected Hessian are equal, and FS is equivalent to NR.

Data augmentation prior. When applying the data-augmentation prior (DAP) described in Section 3.7, the NR procedure proceeds in the same way; the only change is that each f_{gy} is replaced by $f_{gy} + f^{\dagger}_{gy}$, and the function being maximized becomes $\log P$.

Starting values. If starting values are not supplied by the user, we begin with a naive estimate for $\boldsymbol{\beta}$ that produces equal fitted-probability vectors $\boldsymbol{\pi}_g$ across the covariate patterns $g = 1,\ldots,G$. These initial fitted probabilities are set to the marginal proportions for $Y = 1,\ldots,C$ observed in the sample.

Step halving. Netwon-Raphson is not guaranteed to increase the loglikelihood or $\log P$ at each step. Depending on where the current parameter value is located, a

full step of NR could propel the next value into extreme regions of the parameter space where some of the fitted probabilities are numerically zero or $\log P$ becomes $-\infty$. To stabilize the procedure, we apply the technique of step halving (Jennrich and Sampson, 1976). With this modification, (19) is replaced by

$$\hat{\boldsymbol{\beta}}^{(t+1)} \;=\; \hat{\boldsymbol{\beta}}^{(t)} \;+\; (0.5)^s \left[-\left(\frac{\partial^2 l}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \right) \right]^{-1} \left(\frac{\partial l}{\partial \boldsymbol{\beta}} \right), \tag{27}$$

where s is initially set to zero, and then increased to 1, 2, ... until the $\log P$ function at the updated parameter becomes at least as high as it was under the previous one.

Estimated variances and standard errors. When the NR procedure converges, the final value of $\left[-\partial l^2/\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top\right]^{-1}$ is an estimated covariance matrix for $\hat{\boldsymbol{\beta}}$. This matrix, which we denote by $\hat{V}(\hat{\boldsymbol{\beta}})$, becomes the basis for standard errors computed by `predict()` when `se.fit=TRUE`. Let r denote a row of the dataset under consideration, either the original one used to fit the model or a new one specified by `newdata`, and let $\boldsymbol{x}_r$ denote the vector of predictors for that row. The vector of fitted logits is $\hat{\boldsymbol{\eta}}_r \;=\; (\hat{\eta}_{r1}, \ldots, \hat{\eta}_{rC})^\top$ with $\hat{\eta}_{ry} \;=\; \boldsymbol{x}_r^\top \hat{\boldsymbol{\beta}}_y \;=\; \sum_{j=1}^{p} x_{rj} \hat{\beta}_{jy}$, and the estimated covariance matrix for $\hat{\boldsymbol{\eta}}_r$ is

$$\hat{V}(\hat{\boldsymbol{\eta}}_r) \;=\; \left(\frac{\partial \boldsymbol{\eta}_r}{\partial \boldsymbol{\beta}^\top} \right) \hat{V}(\hat{\boldsymbol{\beta}}) \left(\frac{\partial \boldsymbol{\eta}_r}{\partial \boldsymbol{\beta}^\top} \right)^\top, \tag{28}$$

with the elements of $\partial \boldsymbol{\eta}_r / \partial \boldsymbol{\beta}^\top$ given by

$$\frac{\partial \eta_{ry}}{\partial \beta_{jk}} \;=\; \begin{cases} x_{rj} & \text{if } k = y \text{ and } y \neq b, \\ 0 & \text{otherwise.} \end{cases}$$

Applying the delta method, the estimated covariance matrix for $\hat{\boldsymbol{\pi}}_r \;=\; (\hat{\pi}_{r1}, \ldots, \hat{\pi}_{rC})^\top$, $\hat{\pi}_{ry} \;=\; \exp(\hat{\eta}_{ry}) / \sum_{y'=1}^{C} \exp(\hat{\eta}_{ry'})$ is

$$\hat{V}(\hat{\boldsymbol{\pi}}_r) \;=\; \left(\frac{\partial \boldsymbol{\pi}_r}{\partial \boldsymbol{\beta}^\top} \right) \hat{V}(\hat{\boldsymbol{\beta}}) \left(\frac{\partial \boldsymbol{\pi}_r}{\partial \boldsymbol{\beta}^\top} \right)^\top, \tag{29}$$

with the elements of $\partial \boldsymbol{\pi}_r / \partial \boldsymbol{\beta}^\top$ given by $\partial \pi_{ry} / \partial \beta_{jk} \;=\; \hat{\pi}_{ry} \left[I(k = y) \;-\; \hat{\pi}_{rk} \right] x_{rj}$.

B Model fitting with a noisy response

EM algorithm. Using the notation from Section 4.2, the loglikelihood function for modeling the noise-infused response is is $l \;=\; \sum_{g=1}^{G} l_g$, where

$$l_g \;=\; \sum_{y=1}^{C} f_{gy}^* \log \pi_{gy}^* \tag{30}$$

is the contribution from covariate pattern g. The default procedure (`method="EM"`) maximizes this function by repeatedly maximizing the loglikelihood for a non-noisy response (18), with the unknown vectors of true frequencies $\boldsymbol{f}_g$ replaced by predictions $\hat{\boldsymbol{f}}_g = E(f_{gy} \mid \boldsymbol{f}^*_g, \boldsymbol{\beta})$. One iteration of EM proceeds as follows.

E-step: From the current estimate $\boldsymbol{\beta}^{(t)}$, compute $\boldsymbol{\pi}_g$, $\boldsymbol{\Phi}_g$, and $\hat{\boldsymbol{f}}_g = \boldsymbol{\Phi}^\top \boldsymbol{f}^*_g$ for $g = 1, \ldots, G$.

M-step: Compute a new estimate $\boldsymbol{\beta}^{(t+1)}$ by the NR procedure from A, replacing the unknown f_{gy}'s by their predicted values from the E-step.

Convergence is judged by examining changes in the fitted probabilities; the procedure halts when the maximum absolute difference in the π_{gy}'s from one iteration to the next falls below `critConverge`, or when the number of iterations reaches `iterMaxEM`.

EM for a saturated model. When fitting a model with `saturated=TRUE`, the E-step remains the same, and the M-step changes to $\hat{\boldsymbol{\pi}}_g = \hat{\boldsymbol{f}}_g / N_g$ for $g = 1, \ldots, G$.

Standard errors after EM. Standard errors are not an automatic byproduct of the EM algorithm. Upon convergence, we compute $\hat{V}(\hat{\boldsymbol{\beta}}) = [-\partial^2 l / \partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top]^{-1}$ using the derivative formulas below. When `saturated=TRUE`, coefficents are not defined and standard errors are omitted.

Newton-Raphson. EM's convergence can be slow, especially when the perturbation matrix $\boldsymbol{T}$ has high error rates. If EM has not converged by `iterMaxEM` iterations, a user may continue with EM or switch to `method="NR"`. The convergence behavior of NR in the vicinity of a mode is quadratic, compared to linear convergence for EM. If we are near a solution, switching to NR can greatly speed up the process of getting there, but if we are too far from the solution, NR may fail due to the loglikelihood's ususual shape. For NR, we compute derivatives of l_g with respect to $\boldsymbol{\beta}$ and sum them for $g = 1, \ldots, G$. Applying the chain rules (20)–(21) to the new version of l_g, and noting that

$$\frac{\partial l_g}{\partial \boldsymbol{\pi}_g} = \boldsymbol{T}^\top \left(\frac{\partial l_g}{\partial \boldsymbol{\pi}^*_g} \right)$$

and

$$\frac{\partial^2 l_g}{\partial \boldsymbol{\pi}_g \partial \boldsymbol{\pi}_g^\top} = \boldsymbol{T}^\top \left(\frac{\partial^2 l_g}{\partial \boldsymbol{\pi}^*_g \partial \boldsymbol{\pi}^{*\top}_g} \right) \boldsymbol{T},$$

the results are

$$\frac{\partial l_g}{\partial \boldsymbol{\beta}} = \left(\frac{\partial \boldsymbol{\pi}_g}{\partial \boldsymbol{\beta}^\top} \right)^\top \boldsymbol{T}^\top \left(\frac{\partial l_g}{\partial \boldsymbol{\pi}^*_g} \right) \tag{31}$$

and

$$\frac{\partial^2 l_g}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} = \sum_{y=1}^{C} \left(\frac{\partial l_g}{\partial \pi_{gy}} \right) \left(\frac{\partial^2 \pi_{gy}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \right) \tag{32}$$

$$+ \left(\frac{\partial \boldsymbol{\pi}_g}{\partial \boldsymbol{\beta}^\top} \right)^\top \boldsymbol{T}^\top \left(\frac{\partial^2 l_g}{\partial \boldsymbol{\pi}_g^* \partial \boldsymbol{\pi}_g^{*\top}} \right) \boldsymbol{T} \left(\frac{\partial \boldsymbol{\pi}_g}{\partial \boldsymbol{\beta}^\top} \right). \tag{33}$$

The derivatives of l_g with respect to $\boldsymbol{\pi}_g^*$ have elements

$$\frac{\partial l_g}{\partial \pi_{gy}^*} = \frac{f_{gy}^*}{\pi_{gy}^*} \quad \text{and} \quad \frac{\partial^2 l_g}{\partial \pi_{gy}^* \partial \pi_{gy'}^*} = - \left(\frac{f_{gy}^*}{\pi_{gy}^{*2}} \right) I(y = y'),$$

and the derivatives of $\boldsymbol{\pi}_g$ with respect to $\boldsymbol{\beta}$ are given by (22)–(23).

Fisher scoring. With a noise-infused response, the second derivatives of l are functions of the observed frequencies. Fisher scoring (`method="FS"`) proceeds like NR but replaces the second derivatives with their expected values. In (32), $\partial l_g / \partial \pi_{gy}^*$ is replaced by

$$E \left(\frac{\partial l_g}{\partial \pi_{gy}^*} \right) = N_g,$$

and in (33), $\partial^2 l_g / \partial \pi_{gy}^* \partial \pi_{gy'}^*$ is replaced by

$$E \left(\frac{\partial^2 l_g}{\partial \pi_{gy}^* \partial \pi_{gy'}^*} \right) = - \left(\frac{N_g}{\pi_{gy}^*} \right) I(y = y').$$

In regions of the parameter space where l is not concave, FS occasionally succeeds when NR fails. If `method="FS"` reaches a solution, `rrLogit()` attempts to compute an estimated covariance matrix $\hat{V}(\hat{\boldsymbol{\beta}})$ based on the actual (not expected) Hessian.

Starting values. If starting values are not supplied by the user, default starting values are obtained by setting $\boldsymbol{\pi}_g = \tilde{\boldsymbol{\pi}}$ for $g = 1, \ldots, G$, where $\tilde{\boldsymbol{\pi}}$ is the method-of-moments (MOM) estimate of the marginal probabilities for Y. If that procedure fails because one or more of the MOM probabilities are negative, the vector of uniform probabilities $(1/C, \ldots, 1/C)^\top$ is used instead. The starting value for $\boldsymbol{\beta}$ is then computed by converting $\boldsymbol{\pi}_g$ to $\boldsymbol{\eta}_g$ and regressing each of the non-zero logits on $\boldsymbol{x}_g$ using ordinary least squares.

Standard errors for predictions. When `predict()` is called with `noisy=FALSE`, standard errors for $\hat{\eta}_{ry}$'s and $\hat{\pi}_{ry}$'s for rows $r = 1, \ldots, R$ of the dataset are computed as in (28)–(29). When `noisy=TRUE`, standard errors for the $\hat{\pi}_{ry}^*$'s come from

$$\hat{V}(\hat{\boldsymbol{\pi}}_r^*) = \boldsymbol{T} \left(\frac{\partial \boldsymbol{\pi}_r}{\partial \boldsymbol{\beta}^\top} \right) \hat{V}(\hat{\boldsymbol{\beta}}) \left(\frac{\partial \boldsymbol{\pi}_r}{\partial \boldsymbol{\beta}^\top} \right)^\top \boldsymbol{T}^\top. \tag{34}$$

Boundary issues and data-augmentation priors. When modeling a noise-infused response, it is quite common for some of the fitted probabilities $\hat{\pi}_{gy}$ to approach zero, even when the $\hat{\pi}^*_{gy}$'s are all far from zero. When this happens, some estimated coefficients and their standard errors may become unstable, and standard errors for the smallest $\hat{\pi}_{gy}$'s become tiny, creating the illusion of high precision. In extreme cases, the NR procedure used in the M-step of EM may fail because the Hessian is numerically singular. These problems can often be alleviated by a well chosen DAP. When `prior="DAP"`, the function being maximized changes from l to

$$\log P \;=\; \sum_{g=1}^{G}\sum_{y=1}^{C} f^*_{gy} \log \pi^*_{gy} \;+\; \sum_{g=1}^{G}\sum_{y=1}^{C} f^\dagger_{gy} \log \pi_{gy}, \tag{35}$$

where the $f^\dagger_{gy}$'s represent imaginary prior observations of the true response variable Y within covariate patterns. These frequencies are computed as

$$f^\dagger_{gy} \;=\; \left(\frac{N^\dagger}{G}\right) \tilde{\pi}_y,$$

where $N^\dagger$ is the total number of imaginary prior observations from `priorFreqTot`, and $\tilde{\boldsymbol{\pi}} \;=\; (\tilde{\pi}_1, \ldots, \tilde{\pi}_C)^\top$ is the estimate of the marginal proportions of $Y = 1, \ldots, Y = C$ from `priorAlloc`. The default value for `priorFreqTot` is $d = p(C-1)$, the number of free parameters in the model. If `priorAlloc` is not specified, then `rrLogit()` sets $\tilde{\boldsymbol{\pi}}$ to the MOM estimate of the marginal probabilities for Y. If any of the MOM probabilities are negative, the procedure fails and a warning message is given. If that happens, a reasonable setting for `priorAlloc` is the vector of uniform probabilities `rep(1/C,C)` where `C` is the number of response categories. Even when the MOM procedure succeeds, the default `priorAlloc` might not be as effective as we would like at pulling the solution away from the boundary, especially if some of the MOM probabilities are small; in that case, `rep(1/C,C)` may be a better choice. With a DAP, the E-step of the EM algorithm remains the same, and the M-step becomes

M-step: Compute a new estimate $\boldsymbol{\beta}^{(t+1)}$ by the NR procedure from Appendix A, with each unknown frequency f_{gy} replaced by $\hat{f}_{gy} + f^\dagger_{gy}$.

When performing NR or FS with a DAP, the first and second derivatives of the loglikelihood are computed as before, but we need to add to them the first and second derivatives of the second term in (35). That penalty term has the same functional form as the loglikelihood for non-noisy response, and its derivatives are given by the formulas in A with the $f^\dagger_{gy}$'s playing the role of the f_{gy}'s.

C Measures of missing information

For this discussion, we momentarily switch to a generic notation that applies to a broad set of missing-data problems. Let $\boldsymbol{X} = (\boldsymbol{Y}, \boldsymbol{Z})$ denote a generic dataset to be modeled, where $\boldsymbol{Y}$ is a portion of $\boldsymbol{X}$ that is seen, and $\boldsymbol{Z}$ is a portion of $\boldsymbol{X}$ that is hidden. We refer to $\boldsymbol{X}$ as the complete data, $\boldsymbol{Y}$ as the observed data, and $\boldsymbol{Z}$ is the missing data. Let $\boldsymbol{\theta}$ denote a vector of unknown parameters in a population model for $\boldsymbol{X}$, a model that we assume has been correctly specified. Let

$$L(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \;\propto\; P(\boldsymbol{X} \,|\, \boldsymbol{\theta})$$

denote the likelihood function that conditions on the complete data, and let

$$L(\boldsymbol{\theta} \,|\, \boldsymbol{Y}) \;\propto\; \int P(\boldsymbol{X} \,|\, \boldsymbol{\theta})\, d\boldsymbol{Z}$$

denote the likelihood function based on the observed data ignoring the missing-data mechanism. The relationship between these two likelihood functions is

$$L(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \;=\; L(\boldsymbol{\theta} \,|\, \boldsymbol{Y})\, P(\boldsymbol{Z} \,|\, \boldsymbol{Y}, \boldsymbol{\theta}),$$

where $P(\boldsymbol{Z} \,|\, \boldsymbol{Y}, \boldsymbol{\theta})$ is the predictive distribution of the missing data given the observed data. Taking logarithms and differentiating twice gives

$$\left[-\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \log L(\boldsymbol{\theta} \,|\, \boldsymbol{X}) \right] \;=\; \left[-\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \log L(\boldsymbol{\theta} \,|\, \boldsymbol{Y}) \right] + \left[-\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \log P(\boldsymbol{Z} \,|\, \boldsymbol{Y}, \boldsymbol{\theta}) \right],$$

or "the complete information equals the observed information plus the missing information,"

$$\mathcal{I}_X(\boldsymbol{\theta}) \;=\; \mathcal{I}_Y(\boldsymbol{\theta}) \;+\; \mathcal{I}_{Z|Y}(\boldsymbol{\theta}). \tag{36}$$

This decomposition, originally due to Orchard and Woodbury (1972), is commonly known as the missing information principle (Little and Rubin, 2019).

In applying this missing information principle to the `rrLogit` model, the true responses Y play the role of missing data, the noisy responses Y^* play the role of observed data, and the logistic coefficients $\boldsymbol{\beta}$ play the role of $\boldsymbol{\theta}$. (If Y becomes known, then Y^* provides no additional information about $\boldsymbol{\beta}$, so Y also plays the role of complete data.) Making these substitutions in (36), rearranging terms, and evaluating at $\boldsymbol{\beta} = \hat{\boldsymbol{\beta}}$, the missing-information matrix becomes

$$\begin{aligned} \mathcal{I}_{Y|Y^*}(\hat{\boldsymbol{\beta}}) \;&=\; \mathcal{I}_Y(\hat{\boldsymbol{\beta}}) \;-\; \mathcal{I}_{Y^*}(\hat{\boldsymbol{\beta}}) \\ &=\; I \;-\; \mathcal{I}_{Y^*}(\hat{\boldsymbol{\beta}})\, \mathcal{I}_Y^{-1}(\hat{\boldsymbol{\beta}}). \end{aligned} \tag{37}$$

The observed-information matrix $\mathcal{I}_{Y^*}(\hat{\boldsymbol{\beta}})$ is the inverse of the estimated covariance matrix $\hat{V}(\hat{\boldsymbol{\beta}})$ returned by the `rrLogit()` function, which is computed using

the derivative formula (32)–(33). When `method="EM"`, an estimate of the inverse complete-information matrix $\mathcal{I}_Y^{-1}(\hat{\boldsymbol{\beta}})$ is available from the final M-step of the EM algorithm, which uses Newton-Raphson to maximize the expected complete-data loglikelihood function. The missing-information matrix (37) plays a key role in the convergence behavior of the EM algorithm. EM's rate of convergence is given by the largest eigenvalue of this matrix, which is the called the "worst fraction of missing information"; it pertains to the particular linear combination of the parameters for which the rate of missing information is highest.

For diagnostic purposes, it is generally more useful to examine fractions of missing information for individual parameters rather than the global upper bound for all possible parameters. Let $\hat{V}(\hat{\boldsymbol{\beta}}) = \mathcal{I}_{Y^*}^{-1}(\hat{\boldsymbol{\beta}})$ denote the usual estimated covariance matrix for $\hat{\boldsymbol{\beta}}$, which can be extracted from the result of an `rrLogit()` model fit by calling `coef( ..., covMat=TRUE)`. Let $\hat{V}_Y(\hat{\boldsymbol{\beta}}) = \mathcal{I}_Y^{-1}(\hat{\boldsymbol{\beta}})$ denote the estimated complete-data covariance matrix, which can be extracted by calling `coef( ..., covMat=TRUE, completeData=TRUE)`. Let β_j denote an individual element of $\boldsymbol{\beta}$. By analogy to (37), the fraction of missing information (FMI) for β_j is

$$\mathit{FMI}(\beta_j) = 1 - \frac{\left[\hat{V}_Y(\hat{\boldsymbol{\beta}})\right]_{jj}}{\left[\hat{V}(\hat{\boldsymbol{\beta}})\right]_{jj}}. \tag{38}$$

A closely related quantity is relative increase in variance (RIV),

$$\mathit{RIV}(\beta_j) = \frac{\left[\hat{V}(\hat{\boldsymbol{\beta}})\right]_{jj}}{\left[\hat{V}_Y(\hat{\boldsymbol{\beta}})\right]_{jj}} = \frac{1}{1 - \mathit{FMI}(\beta_j)}. \tag{39}$$

The RIV, which is greater than or equal to one, is the factor by which the estimated variance of $\hat{\beta}_j$ has been inflated because the true response Y was replaced with the noisy version Y^*. For easier interpretation, Savalei and Rhemtulla (2012) and Chen and Savalei (2021) recommend using the width inflation factor (WIF),

$$\mathit{WIF}(\beta_j) = \sqrt{\mathit{RIV}(\beta_j)} = \sqrt{\frac{1}{1 - \mathit{FMI}(\beta_j)}}, \tag{40}$$

which measures the inflation in standard errors and width of confidence intervals. Solving (40), we can compute FMI from WIF as

$$\mathit{FMI}(\beta_j) = 1 - \frac{1}{[\mathit{WIF}(\beta_j)]^2}.$$

The diagnostics (38)–(40) describe the impact of noise infusion on measures of error for logistic coefficients. We can also obtain analogous diagnostics for predicted

values of the response. By calling `predict()` with the options `se.fit=TRUE` and `completeData=TRUE`, the resulting standard errors will approximate the uncertainty that we would have seen without added noise. These standard errors are computed by the delta methos, as in (34), with $\hat{V}(\hat{\boldsymbol{\beta}})$ replaced by $\hat{V}_Y(\hat{\boldsymbol{\beta}})$.

Note that these missing-information diagnostics come from the M-step of the EM algorithm, so they are only available from a model that was fit with `method="EM"`. If a model was fit to convergence using `method="NR"`, we can apply `method="EM"`, which will cause `rrLogit()` to take one step of EM before stopping.

D Bayesian simulation and MCMC

(describe Data Augmentation (DA) and random-walk Metropolis (RWM))