The Stata Journal (2017)
17, Number 3, pp. 723–735

Evaluating the maximum MSE of mean estimators with missing data

Charles F. Manski
Department of Economics
Northwestern University
Evanston, IL
cfmanski@northwestern.edu

Max Tabord-Meehan
Department of Economics
Northwestern University
Evanston, IL
mtabordmeehan@u.northwestern.edu

Abstract. In this article, we present the `wald_mse` command, which computes the maximum mean squared error of a user-specified point estimator of the mean for a population of interest in the presence of missing data. As pointed out by Manski (1989, *Journal of Human Resources* 24: 343–360; 2007, *Journal of Econometrics* 139: 105–115), the presence of missing data results in the loss of point identification of the mean unless one is willing to make strong assumptions about the nature of the missing data. Despite this, decision makers may be interested in reporting a single number as their estimate of the mean as opposed to an estimate of the identified set. It is not obvious which estimator of the mean is best suited to this task, and there may not exist a universally best choice in all settings. To evaluate the performance of a given point estimator of the mean, `wald_mse` allows the decision maker to compute the maximum mean squared error of an arbitrary estimator under a flexible specification of the missing-data process.

Keywords: st0494, wald_mse, maximum mean squared error

1 Introduction

In this article, we present the `wald_mse` command, which computes the maximum mean squared error (MSE) of a user-specified point estimator of the mean for a population of interest in the presence of missing data. As pointed out by Manski (1989, 2007), the presence of missing data results in the loss of point identification of the mean, unless one is willing to make strong assumptions about the nature of the missing data. Despite this, decision makers may be interested in reporting a single number as their estimate of the mean as opposed to an estimate of the identified set. It is not obvious which estimator of the mean is best suited to this task, and there may not exist a universally best choice in all settings. To evaluate the performance of a given point estimator of the mean, `wald_mse` allows the decision maker to compute the maximum MSE of an arbitrary estimator under a flexible specification of the missing-data process.

The choice of maximum MSE as the criteria with which to evaluate a point estimator of the mean is well founded. With "maximum" MSE, we take an explicitly ex-ante perspective in that we evaluate the performance of the estimator over all possible data-generating processes that the decision maker thinks they may encounter and use the largest of these to evaluate the estimator. The choice of MSE as our measure of perfor-

mance is not only historically grounded but also can be formalized in Wald's statistical decision theory as the minimax-regret solution when solving the problem of best point prediction under square loss (see Dominitz and Manski [Forthcoming] for details).

This article proceeds as follows: section 2 formalizes our discussion and explains the algorithm that we use to compute maximum MSE. Section 3 introduces the syntax of `wald_mse`, explains the options available to the user for specifying the missing-data process, and documents several implementation details. Section 4 illustrates the command through a series of examples.

2 Statistical background

2.1 The MSE of estimators of the mean without missing data

Let $P(y)$ be the distribution of some random variable Y with known bounds. Because the bounds for Y are known, we normalize it to lie in $[0, 1]$ (for example, in the empirical setting of section 4.2, we use the normalization of dividing the data by 100). Recall that the purpose of `wald_mse` is to compute the maximum MSE of a user-specified point estimator of $\mu := E(Y)$, given a sample from $P(y)$ with potentially missing data. To clarify the exposition, we first introduce the problem without missing data.

In this context, an estimator is a function $\delta(\cdot)$ that maps the sample data into a real number, where the sample data are simply a random sample $\psi := \{y_i\}_{i=1}^N$ from $P(y)$. Let S index the set of all possible distributions $P_s(y)$ for Y; then for each $s \in S$, we can compute the MSE of $\delta(\cdot)$ in state s, which is given by

$$\text{MSE}(s) := E_s\left\{(\delta(\psi) - \mu_s)^2\right\}$$

Decomposing $\text{MSE}(s)$ into its bias and variance components, we get

$$\text{MSE}(s) = V_s\left\{\delta(\psi)\right\} + \left[\mu_s - E_s\left\{\delta(\psi)\right\}\right]^2$$

Our goal is to compute the maximum MSE, given by

$$\max_{s \in S} \text{MSE}(s)$$

In this setting, Hodges and Lehmann (HL; 1950) show that the estimator that achieves *minimal* maximum MSE is given by

$$\delta_{\text{HL}}(\psi) := \frac{\widehat{\mu}\sqrt{N} + 0.5}{\sqrt{N} + 1}$$

where $\widehat{\mu}$ denotes the sample average of the N observations of Y. Hence, we see that in this simple setting, there is an analytic expression for the estimator with minimal maximum MSE. Our interest is in the problem with potentially missing data, for which such an analytic solution seems intractable. `wald_mse` is designed to provide numerical approximations to the maximum MSE for estimators of the mean under a flexible specification of the missing-data process.

2.2 Simulation of maximum MSE with missing data

Now, we consider the problem with missing data. As before, because we assume Y has known bounds, we have normalized it to lie in $[0, 1]$. By the law of total probability, we can decompose $P(y)$ as follows:

$$P(y) = P(y|z = 1)P(z = 1) + P(y|z = 0)P(z = 0)$$

where $z = 1$ if a person's outcome is observable and $z = 0$ if not. Thus, we see that a state s is characterized by the three distributions: $[P_s(y|z = 1), P_s(y|z = 0), P_s(z)]$. As pointed out in Manski (1989, 2007), random sampling with nonresponse point identifies the distributions $P_s(y|z = 1)$ and $P_s(z)$ but is uninformative about $P_s(y|z = 0)$. `wald_mse` allows the user to specify what assumptions they would like to maintain about $P_s(y|z = 0)$ and its relationship to $P_s(z)$ and $P_s(y|z = 1)$. Then, it computes the resulting maximum MSE of a user-specified point estimator of the mean under these assumptions. To compute maximum MSE with missing data, we take advantage of the decomposition of maximum MSE into variance and squared bias, and proceed as follows:

Outline of the algorithm

- Fix a state $\{P_s(y|z = 1), P_s(y|z = 0), P_s(z)\}$, $s \in S$.
- Given s, compute $\mu_s = E_s(y|z = 1)P_s(z = 1) + E_s(y|z = 0)P_s(z = 0)$ analytically.
- Given s, draw N observations $\{z_i\}_{i=1}^N$ from the distribution $P_s(z)$. Let $k = \sum_{i=1}^N z_i$ and draw k observations $\{y_i\}_{i=1}^k$ from $P_s(y|z = 1)$ (note that it is possible that $k = 0$, and hence the estimator $\delta(\cdot)$ must be defined for this case as well). Compute $\delta(\cdot)$ using $\left(\{y_i\}_{i=1}^k, \{z_i\}_{i=1}^N\right)$. Call the result δ_1.
- Repeat the above step T times, and use the values $\{\delta_t\}_{t=1}^T$ to approximate $E_s(\delta)$ and $V_s(\delta)$ by $\widehat{E}_s(\delta)$ and $\widehat{V}_s(\delta)$, their sample averages.
- Repeat this process over all states $s \in S$.
- Return $\max_s[\widehat{V}_s(\delta) + \{\mu_s - \widehat{E}_s(\delta)\}^2]$, the computed maximum MSE.

Despite the conceptual simplicity of the above procedure, the state space is still too large to be tractable. To make the computation feasible, we parameterize the distributions $P_s(y|z = 1)$, $P_s(y|z = 0)$, and $P_s(z)$ and specify the state space as a finite grid over the parameter space. We focus on two cases of interest: when y is binary on $\{0, 1\}$ and when y is continuous on $[0, 1]$. When y is binary, $P_s(y|z = 1)$, $P_s(y|z = 0)$, and $P_s(z)$ are all Bernoulli distributions, and the state space is a finite grid over the possible Bernoulli parameters for each distribution. When y is continuous, we specify $P_s(y|z = 1)$ and $P_s(y|z = 0)$ as beta distributions, while $P_s(z)$ is a Bernoulli distribution. This parameterization is obviously not without loss, but we treat it as a flexible approximation. The state space in the continuous case is then a finite grid over the possible shape parameters of the beta distributions and the Bernoulli distribution.

The main feature of `wald_mse` is the ability to specify the state space S flexibly in the computation of the maximum MSE. For example, the user may specify that they believe that nonresponse will be no higher than 80% or that the mean value of the outcome for nonresponders will be no lower than 0.5. In section 3, we describe in detail the syntax of the command and the options available to the user for specifying the state space.

3 The wald_mse command

3.1 Syntax

The syntax for `wald_mse` is as follows:

`wald_mse` *command_name*`, samp_size(#) dist(`*string*`)` [`miss_l(#) miss_r(#) rdgp_l(#) rdgp_r(#) mdgp_l(#) mdgp_r(#) h_distance(#) r_shape(#) m_shape(#) mon_select(#) mc_iter(#) grid(#) user_def true_beta`]

command_name specifies the name of the estimator to be used. It must be either a built-in estimator (for a list of currently supported estimators, see section 3.3) or any user-specified estimator of the mean that is written as an e-class command that returns to `e(b)` (for more information about user-defined commands, see section 3.4).

3.2 Options

`samp_size(#)` specifies the size of the sample. `samp_size()` is required.

`dist(`*string*`)` specifies the type of data being considered. The two options supported are `bernoulli` for binary $\{0, 1\}$ outcomes and `continuous` for continuous outcomes bounded in the unit interval. The continuous distributions used in the computations are beta distributions (see section 3.4 for more information). `dist()` is required.

In what follows, $z = 1$ if an outcome y is observable, and $z = 0$ if it is missing. $P(y|z = 1)$ and $P(y|z = 0)$ are the population distributions of observable and missing outcomes, while $E(y|z = 1)$ and $E(y|z = 0)$ are their respective means. $P(z = 1)$ and $P(z = 0)$ are the probabilities of response and nonresponse.

`miss_l(#)` specifies a lower bound on the nonresponse probability $P(z = 0)$. For example, setting `miss_l(0.5)` specifies that $P(z = 0) \geq 0.5$. The default is `miss_l(0)`.

`miss_r(#)` specifies an upper bound on the nonresponse probability $P(z = 0)$. The default is `miss_r(1)`.

`rdgp_l(#)` specifies a lower bound on the mean $E(y|z = 1)$ of the observable outcomes. For example, setting `rdgp_l(0.5)` specifies that $E(y|z = 1) \geq 0.5$. The default is `rdgp_l(0)`.

`rdgp_r(#)` specifies an upper bound on the mean $E(y|z = 1)$. The default is `rdgp_r(1)`.

`mdgp_l(#)` specifies a lower bound on the mean $E(y|z=0)$ of the missing outcomes. The default is `mdgp_l(0)`.

`mdgp_r(#)` specifies an upper bound on the mean $E(y|z=0)$. The default is `mdgp_r(1)`.

`h_distance(#)` specifies an upper bound on the Hellinger distance between the distributions $P(y|z=1)$ and $P(y|z=0)$. For example, setting `h_distance(0)` forces the distribution of missing data and observable data to be identical, which is equivalent to assuming missingness at random. The default is `h_distance(1)`, equivalent to making no assumptions linking missing and observable data.

`r_shape(#)`, when `dist(continuous)` is also specified, specifies the shape of the observable data's density. If `r_shape(1)` is specified, then only distributions with modes in the interior of the unit interval are considered. If `r_shape(2)` is specified, then only distributions with U-shaped densities are considered. If `r_shape(0)` (the default) is specified, then both types are considered.

`m_shape(#)`, when `dist(continuous)` is also specified, specifies the shape of the missing data's density. See `r_shape()` for details that equally apply to `m_shape()`.

`mon_select(#)` specifies whether to make a "monotone selection" type assumption. If `mon_select(1)` is specified, only data-generating processes where $E(y|z=0) \geq E(y|z=1)$ are considered. If `mon_select(2)` is specified, only data-generating processes where $E(y|z=1) \geq E(y|z=0)$ are considered. If `mon_select(0)` (the default) is specified, then both options are considered.

`mc_iter(#)` specifies the number of Monte Carlo iterations used to compute the MSE of the estimator in repeated samples. The default for built-in estimators is 3,000, and the default for user-defined commands is 500 (see section 3.4 for details).

`grid(#)` specifies the number of evenly spaced grid points used to generate the distributions. For example, if the outcome is binary and `grid(6)` is specified, the resulting grid of Bernoulli parameters for the distributions would be $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. The default for binary outcomes is 25, and the default for continuous outcomes is 5 (see section 3.4 for details).

`user_def` specifies that the estimator being used is user defined (not built in). See section 3.4 for details.

`true_beta` specifies to approximate certain U-shaped continuous distributions by beta distributions instead of by the default Bernoulli distributions (see section 3.4 for more information). In general, specifying `true_beta` will slow down the program but may increase accuracy.

3.3 Built-in estimators

Note that when all the data are missing, all the estimators below return a result of 0.5.

`mean` is simply the sample mean.

`midmean` is an estimator that first estimates the identified interval under no assumptions on the missing data, as in Manski (1989), and then selects the midpoint of the interval. See Dominitz and Manski (Forthcoming) for details.

`MMRzero` is the minimal-maximum-MSE estimator of the mean with no missing data, as described in section 2.1 and as derived in Hodges and Lehmann (1950).

3.4 Implementation details

Details on built-in versus user-defined estimators

`wald_mse` allows use of the built-in estimators available (see section 3.3) or of any user-defined estimator of the mean if it is an e-class Stata command that returns to `e(b)`. In general, user-defined estimators will be much slower than built-in estimators because a user-defined estimator requires a call to Stata in each Monte Carlo iteration, whereas the built-in estimators are implemented directly in Mata. For example, the commands we run in the empirical application of section 4.2 take five seconds for a built-in estimator and five minutes for a user-defined estimator on a personal computer. This difference is reflected in the default options used for the accuracy of the computation: for user-defined estimators, we use six times fewer Monte Carlo iterations by default. When using a user-defined estimator with sample sizes larger than 1,000, we recommend first running the computation with binary outcomes, because this will be much faster and is frequently a very close approximation of the maximum MSE for continuous outcomes. The `user_def` option must be specified when using user-defined estimators.

For advanced users, it is possible to implement the estimators directly in Mata by adding them to the `wald_mse` command. See the comments in the code for more details.

Details on the approximation of continuous-valued random variables

Because the state space S is prohibitively large when dealing with continuous variables, we implement a parametric approximation. For a continuous distribution in the unit interval, we iterate over the (α, β) parameters of a beta distribution. For each parameter, we consider an equally spaced grid from 0.01 to $\sqrt{\texttt{grid()}} + 2$, so that both the density of the grid and its upper bound increase with `grid()`. Because the state space is two-dimensional compared with the one-dimensional state space when the distribution is binary, the default is to set `grid(5)` when `dist(continuous)` is set.

Next, we explain the use of `true_beta` and the default behavior of `wald_mse` to use Bernoulli distributions to approximate some continuous distributions. Note that the beta random-number generator in Mata is restricted to $\alpha > 0.05$ and $\beta > 0.15$, but in our experience, maximum MSE is frequently achieved by distributions for which α and β are very small, which result in very U-shaped densities. To deal with this issue, we have provided the user with two options: the default is to approximate beta distributions for which the random generator in Mata is unavailable by a mean-preserving Bernoulli distribution. This approximation seems to be adequate in practice. If the user would

instead like to avoid this approximation, specifying `true_beta` in the command will force `wald_mse` to use a custom-built beta random-number generator for those cases in which the random-number generator in Mata is unavailable. This option is much slower than the default option. If the user does not want to consider the possibility of these heavily U-shaped densities and instead would prefer to consider only distributions with a mode in the interior of the unit interval, this can be done by setting `r_shape()` and `m_shape()` to 1.

3.5 Stored results

`wald_mse` stores the following in `r()`:

Scalars

`r(N)`	specified sample size	
`r(MSE)`	computed maximum MSE	
`r(rmeanval)`	value of $E(y\|z=1)$ where maximum MSE is achieved	
`r(mmeanval)`	value of $E(y\|z=0)$ where maximum MSE is achieved	
`r(missval)`	value of $P(z=0)$ where maximum MSE is achieved	
`r(missr)`	specified upper bound on $P(z=0)$	
`r(missl)`	specified lower bound on $P(z=0)$	
`r(mdgpr)`	specified upper bound on $E(y\|z=0)$	
`r(mdgpl)`	specified lower bound on $E(y\|z=0)$	
`r(rdgpr)`	specified upper bound on $E(y\|z=1)$	
`r(rdgpl)`	specified lower bound on $E(y\|z=1)$	
`r(hd)`	specified bound on Hellinger distance between $P(y\|z=0)$ and $P(y\|z=1)$	
`r(mshape)`	specified shape option for $P(y\|z=0)$	, if applicable
`r(rshape)`	specified shape option for $P(y\|z=1)$	, if applicable

Macros

`r(cmd)`	`wald_mse`
`r(est)`	estimator used
`r(d)`	specified distribution type

4 Examples and empirical illustration

In this section, we illustrate the use of `wald_mse` through a series of examples. In section 4.1, we present a series of fictional examples to explain the functionality of the command. In section 4.2, we walk through the use of the command with an application.

4.1 Some fictional examples

First, we will compare the maximum MSE of the simple sample analog estimator of the mean, call it $\delta_M(\cdot)$, with that of the minimax-MSE estimator $\delta_{\text{HL}}(\cdot)$ defined above, without missing data. It is straightforward to compute, for a sample of size N, that the maximum MSE of $\delta_M(\cdot)$ is given by $1/4N$. For $\delta_{\text{HL}}(\cdot)$, it can be shown that the maximum MSE for a sample of size N is given by $1/\{4(\sqrt{N}+1)^2\}$. We will verify our command by comparing it against these analytic expressions. The following command returns the computed maximum MSE for $\delta_M(\cdot)$ for $N = 10$, where we consider binary distributions of the outcome and we set the upper bound on the probability of nonresponse to be 0%:

```
. wald_mse mean, samp_size(10) dist(bernoulli) miss_r(0)
Working.............
Maximum MSE: .025148378
```

The following command returns the computed maximum MSE for $\delta_{\text{HL}}(\cdot)$ under the same specification:

```
. wald_mse MMRzero, samp_size(10) dist(bernoulli) miss_r(0)
Working.............
Maximum MSE: .014904072
```

Next, we consider what happens when we allow for the possibility of missing data. The following command returns the computed maximum MSE for $\delta_{\text{HL}}(\cdot)$ with $N = 10$, for binary distributions of the outcome, where we allow the probability of nonresponse to be between 0% and 50%, and we make no assumptions about the relationship between the distribution of realized and missing data. Again, we do this by specifying the `miss_r()` option:

```
. wald_mse MMRzero, samp_size(10) dist(bernoulli) miss_r(0.5)
Working..........
Maximum MSE: .113844727
```

Notice the significant increase in computed maximum MSE to the case with no missing data.

Let's compare this with the maximum MSE of the midpoint mean estimator described in Dominitz and Manski (Forthcoming):

```
. wald_mse midmean, samp_size(10) dist(bernoulli) miss_r(0.5)
Working..........
Maximum MSE: .073496388
```

We see that $\delta_{\text{HL}}(\cdot)$ no longer necessarily achieves minimax MSE in the presence of missing data.

Let's perform the same exercise, where now we make the assumption that the data are missing at random. We can do this by forcing that the Hellinger distance between the distributions of realized and missing data is 0 by specifying the `h_distance()` option:

```
. wald_mse MMRzero, samp_size(10) dist(bernoulli) miss_r(0.5) h_distance(0)
Working..........
Maximum MSE: .026664216
. wald_mse midmean, samp_size(10) dist(bernoulli) miss_r(0.5) h_distance(0)
Working..........
Maximum MSE: .067955161
```

We see that, as might be expected, maximum MSE is reduced when we assume that the data are missing at random, and the computed maximum MSE of $\delta_{\text{HL}}(\cdot)$ is smaller than the computed maximum MSE of the midpoint estimator. Once we add the assumption that the data are missing at random, the missing data simply have the effect of reducing sample size so that $\delta_{\text{HL}}(\cdot)$ is again the minimax MSE estimator.

4.2 The wald_mse command in practice

In this section, we illustrate the use of `wald_mse` in an empirical application. Suppose we wish to compute an estimate of the mean of individuals' expectations of stock market health for the upcoming year, using survey data elicited similarly to the Survey of Economic Expectations (SEE; a previous empirical analysis of this dataset in this context can be found in Dominitz and Manski [2011]). For example, the SEE asks respondents to state their subjective probabilities, expressed as a number between 0 and 100, that a $1,000 investment in a diversified stock fund will grow above a specified threshold in the following year. Nonresponse to this type of question is as high as 30% in the SEE, so consideration of alternative estimators of the mean could be important.

Before illustrating the use of `wald_mse` for this application, we present different possible estimates of the mean of the variable `expect` in `SEE_example.dta`, which is a cleaned up version of the variable described above.[1] First, let's consider two different estimators: the standard sample mean (computed using the `mean` command) and `midmean`. First, because `midmean` assumes that we have outcomes that lie in the unit interval, we normalize our data:

```
. use see_example
. generate expect_norm = expect/100
(420 missing values generated)
```

Next, we compute the estimated mean by using the `mean` command (while suppressing the rest of the output):

```
. quietly mean expect_norm
. matrix list e(b)
symmetric e(b)[1,1]
    expect_norm
y1    .71090171
```

Let's compare this with the estimated mean by using `midmean`:

```
. quietly midmean expect_norm
. matrix list e(b)
symmetric e(b)[1,1]
           c1
y1  .65725015
```

Given these results, we might worry that the standard sample mean overstates the optimism in stock market returns, but practitioners have no formal guidance on which estimator is appropriate for this setting. The purpose of `wald_mse` is to help practitioners evaluate various point estimators through their maximum MSE, under a specification of the missing data that they deem appropriate.

1. The original dataset is available at http://faculty.wcas.northwestern.edu/~cfm754/. Here, we consider waves 12 through 14, which were the waves in which this question was asked. For this example, we use the data column `rvdm05`. For the purposes of our example, we code the response "do not know" as missing (which affects 19 observations).

For this example, we evaluate the maximum MSE of `mean` and `midmean`, as well as a user-defined estimator that we call `monotone_mean`. Let's suppose that the average optimism of those who did not respond is lower than the average optimism of those who did respond; that is,

$$E(y|z = 1) \geq E(y|z = 0)$$

Then, in analogy to the `midmean` estimator, we could consider the estimator formed using the midpoint of the identified interval under this assumption. The resulting estimator is given by the `monotone_mean` command (included with the supplementary materials). Running `monotone_mean` on our dataset yields

```
. quietly monotone_mean expect_norm
. matrix list e(b)
symmetric e(b)[1,1]
    expect_norm
y1    .62047811
```

which is an even lower estimate than that given by `midmean`.

When applying `wald_mse`, we assume that the practitioner plans to survey 650 people, which is a representative size for a survey wave in the SEE. As a first exercise, we evaluate the maximum MSE of our estimators under the following two assumptions: that the proportion of nonresponse will not exceed 50%, which we deem to be a reasonable assumption in our setting, and that the monotonicity assumption that we imposed when deriving `monotone_mean` does in fact hold. To make the computations between our built-in estimators and the user-defined estimator `monotone_mean` comparable, we additionally specify that the number of Monte Carlo iterations used in all of the computations be set to 1,000.

Let's first compute the maximum MSE for the `mean` command:

```
. drop expect expect_norm
. wald_mse mean, samp_size(650) dist(continuous) miss_r(0.5)
> mon_select(2) mc_iter(1000)
Working.......
Maximum MSE: .247711421
```

We stress again that the evaluation of maximum MSE is an ex-ante concept: `wald_mse` computes the MSE of the specified estimator for all data-generating processes that satisfy the specifications outlined in the command options.

Next, we compute the maximum MSE of the `midmean` command:

```
. wald_mse midmean, samp_size(650) dist(continuous) miss_r(0.5) mon_select(2)
> mc_iter(1000)
Working.......
Maximum MSE: .062231883
```

We see that, as was the case in the fictional examples of section 4.1, `midmean` has a much lower maximum MSE in the presence of missing data than the standard sample mean estimator.

Finally, we compute the maximum MSE of the `monotone_mean` command; note the addition of `user_def` to specify that this is a user-written command. As we alluded to in section 3.4, evaluating user-defined estimators will be much slower than evaluating the estimators built in to `wald_mse`. For this particular example, running our command on `monotone_mean` takes around five minutes on a personal computer:

```
. wald_mse monotone_mean, samp_size(650) dist(continuous) miss_r(0.5)
> mon_select(2) mc_iter(1000) user_def
Working.......
Maximum MSE: .062223066
```

Perhaps surprisingly, we see that the gain over `midmean` is so small as to be within the Monte Carlo error of the simulation. This exercise shows that, at least under the assumptions on missing data that we have specified, the performance gain of `monotone_mean` from the perspective of maximum MSE is negligible.

We could also consider adding more assumptions on the missing-data process. For example, we could study the performance of our estimators under the monotonicity assumption while also adding a restriction on the Hellinger distance between the observed- and missing-data distributions. As we saw in section 4.1, setting `h_distance(0)` enforces the assumption that the data are missing at random. In general, for Hellinger distances between 0 and 1, it is difficult to get a sense of how the state space changes for small changes in the Hellinger distance. Here, we will set `h_distance(0.3)`, which corresponds to a setting where we allow only small deviations between the observed- and missing-data distributions. We repeat our computations:

```
. wald_mse mean, samp_size(650) dist(continuous) miss_r(0.5) mon_select(2)
> mc_iter(1000) h_distance(0.3)
Working.......
Maximum MSE: .007234028
. wald_mse midmean, samp_size(650) dist(continuous) miss_r(0.5) mon_select(2)
> mc_iter(1000) h_distance(0.3)
Working.......
Maximum MSE: .062231883
. wald_mse monotone_mean, samp_size(650) dist(continuous) miss_r(0.5)
> mon_select(2) mc_iter(1000) h_distance(0.3) user_def
Working.......
Maximum MSE: .062223066
```

We see that if we restrict ourselves to data-generating processes where we only allow small deviations between the observed- and missing-data distributions, the standard sample mean vastly outperforms both the `midmean` and `monotone_mean` estimators.

Finally, in an attempt to gauge the robustness of our estimators to a misspecification of our assumptions, we consider what would happen if we drop both the monotonicity and the Hellinger distance restrictions:

```
. wald_mse mean, samp_size(650) dist(continuous) miss_r(0.5) mc_iter(1000)
Working.......
Maximum MSE: .247711421
. wald_mse midmean, samp_size(650) dist(continuous) miss_r(0.5) mc_iter(1000)
Working.......
Maximum MSE: .062425221
. wald_mse monotone_mean, samp_size(650) dist(continuous) miss_r(0.5)
> mc_iter(1000) user_def
Working.......
Maximum MSE: .248327823
```

Once we drop the monotonicity assumption and the Hellinger distance assumption, the midmean estimator has a very similar maximum MSE, whereas both the sample mean and the monotone mean estimator suffer from a large increase in maximum MSE.

Performing these exercises gives us a clearer picture of which estimator would be appropriate for our application depending on the assumptions the researcher is willing to maintain. Under the assumption that the missing data are close to missing at random, the standard sample mean estimator outperforms both the midmean estimator and the monotone mean estimator. When we drop the assumption that the data are close to missing at random but maintain our monotonicity assumption, we see that the maximum MSE of the sample mean estimator increases drastically, while the maximum MSEs of the monotone mean and midmean estimators stay mostly the same, with the monotone mean estimator performing only slightly better than the midmean estimator. Finally, we see that when we drop both assumptions, the midmean estimator maintains its maximum MSE, whereas the maximum MSE of the other two estimators increases drastically. From this, we can conclude that if the researcher is willing to assume that the data are close to missing at random in this application, the standard mean estimator is most appropriate. Otherwise, the midmean estimator would be most appropriate.

5 References

Dominitz, J., and C. F. Manski. 2011. Measuring and interpreting expectations of equity returns. *Journal of Applied Econometrics* 26: 352–370.

———. Forthcoming. More data or better data: A statistical decision problem. *Review of Economic Studies*.

Hodges, J. L., Jr., and E. L. Lehmann. 1950. Some problems in minimax point estimation. *Annals of Mathematical Statistics* 21: 182–197.

Manski, C. F. 1989. Anatomy of the selection problem. *Journal of Human Resources* 24: 343–360.

———. 2007. Minimax-regret treatment choice with missing outcome data. *Journal of Econometrics* 139: 105–115.

About the authors

Charles F. Manski is the Board of Trustees Professor in Economics at Northwestern University.

Max Tabord-Meehan is a graduate student in economics at Northwestern University.