Maximum Likelihood Methods for the Social Sciences
POLS 510 · CSSS 510

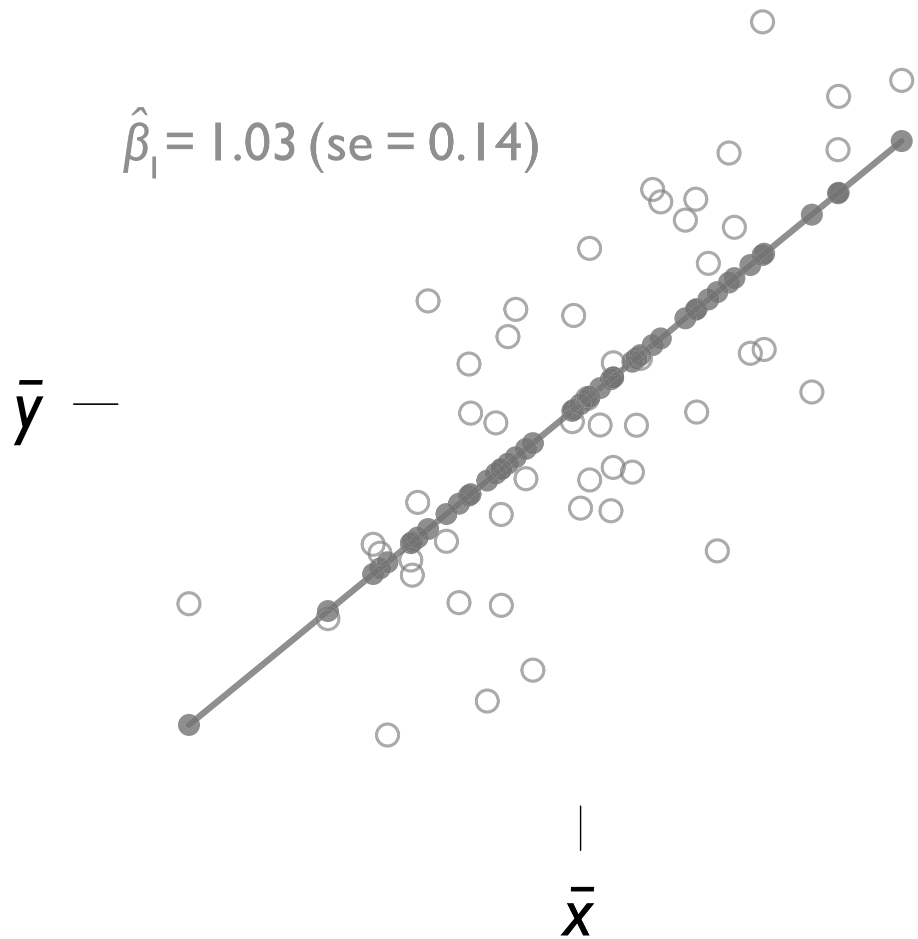
Missing Data and Multiple Imputation

Christopher Adolph

Political Science *and* CSSS
University of Washington, Seattle

Vincent van Gogh *The Bedroom* 1888

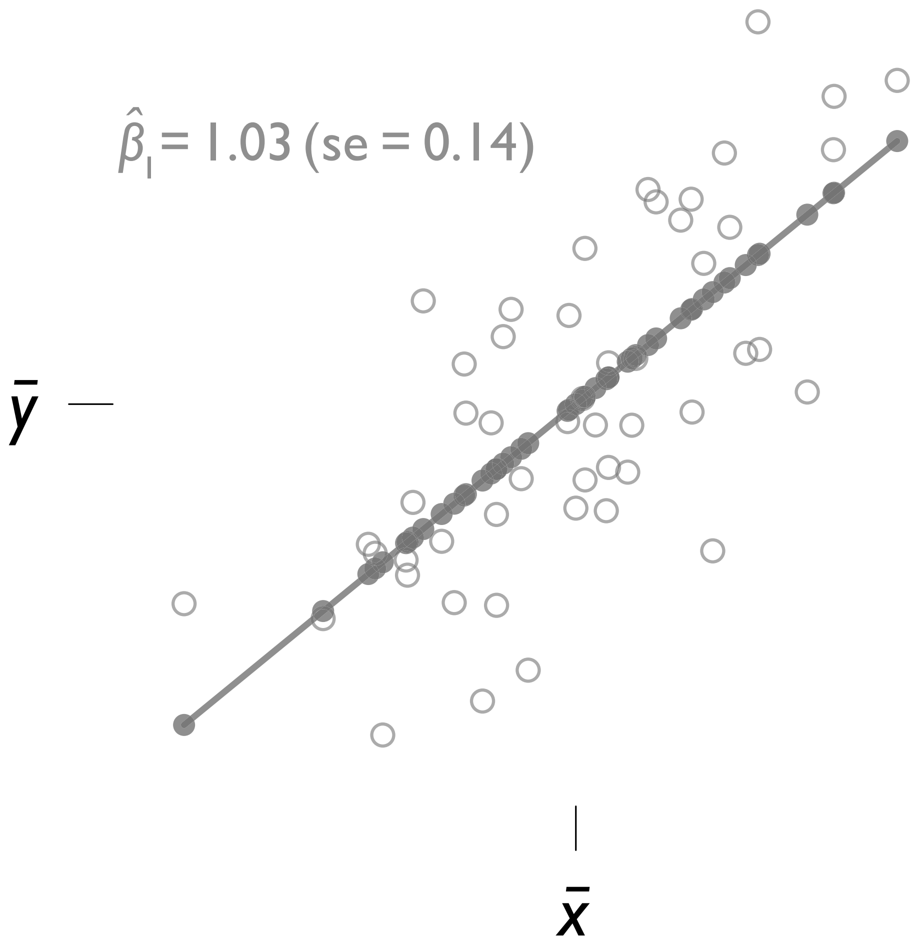
Using a random sample



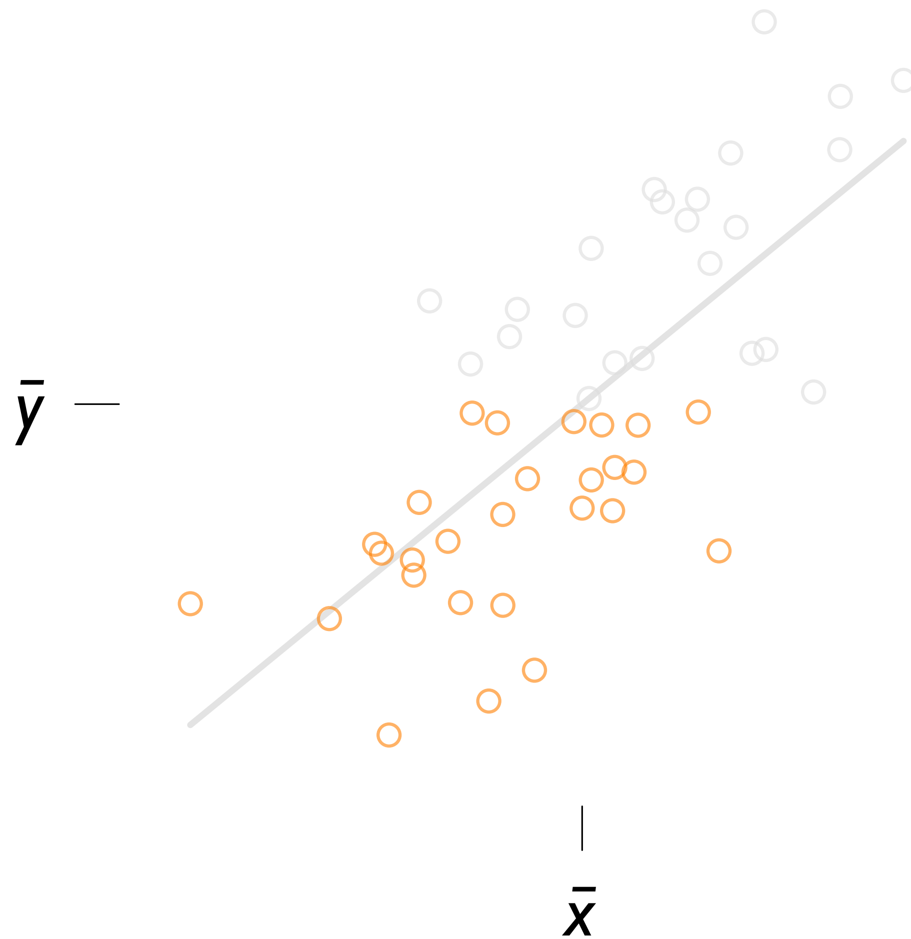
Suppose the population relationship between x and y is $y = x + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, 1)$

If we randomly sample 50 cases, we recover $\hat{\beta}_1$ close to the true value of 1

Using a random sample

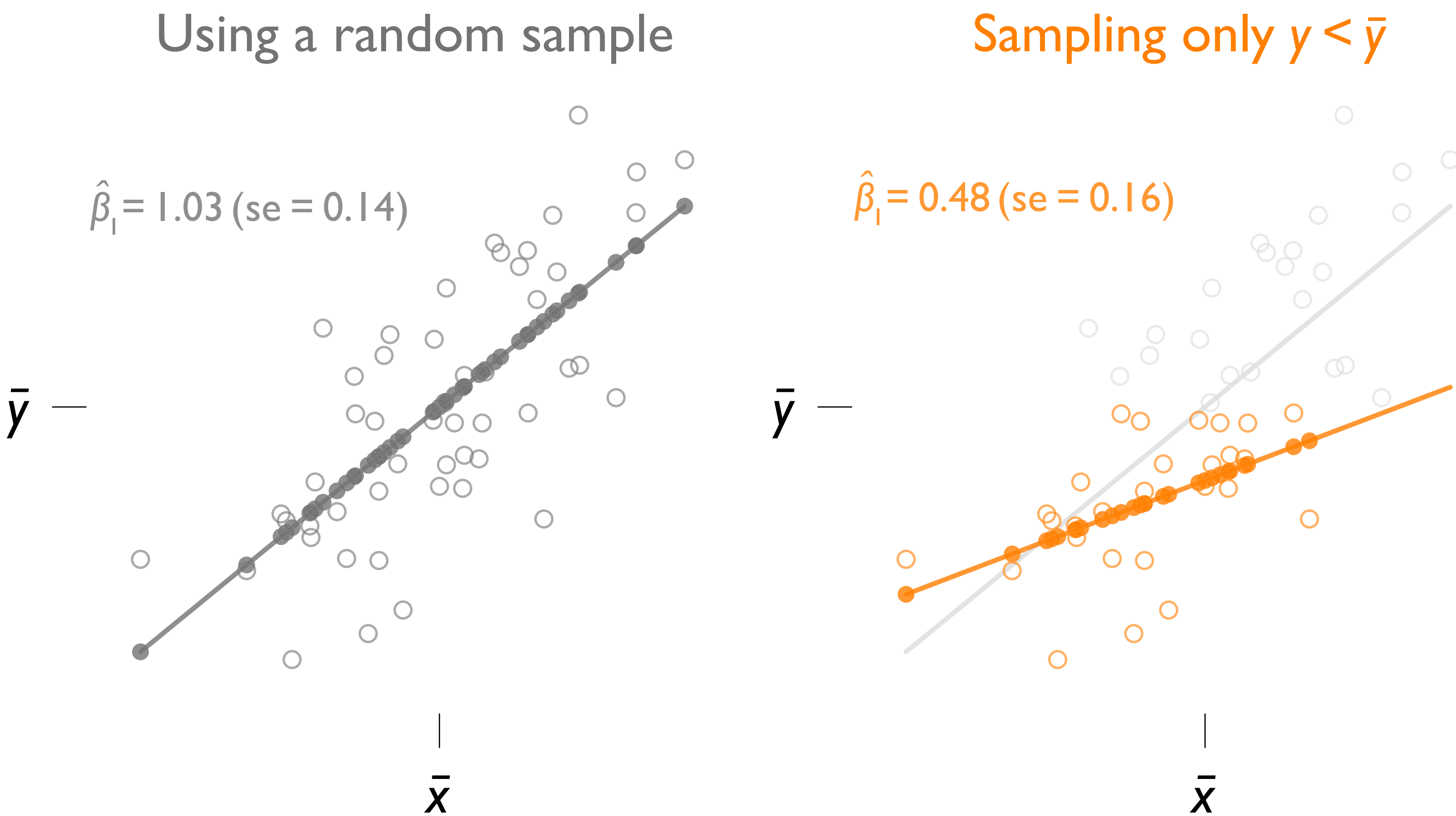


Sampling only $y < \bar{y}$



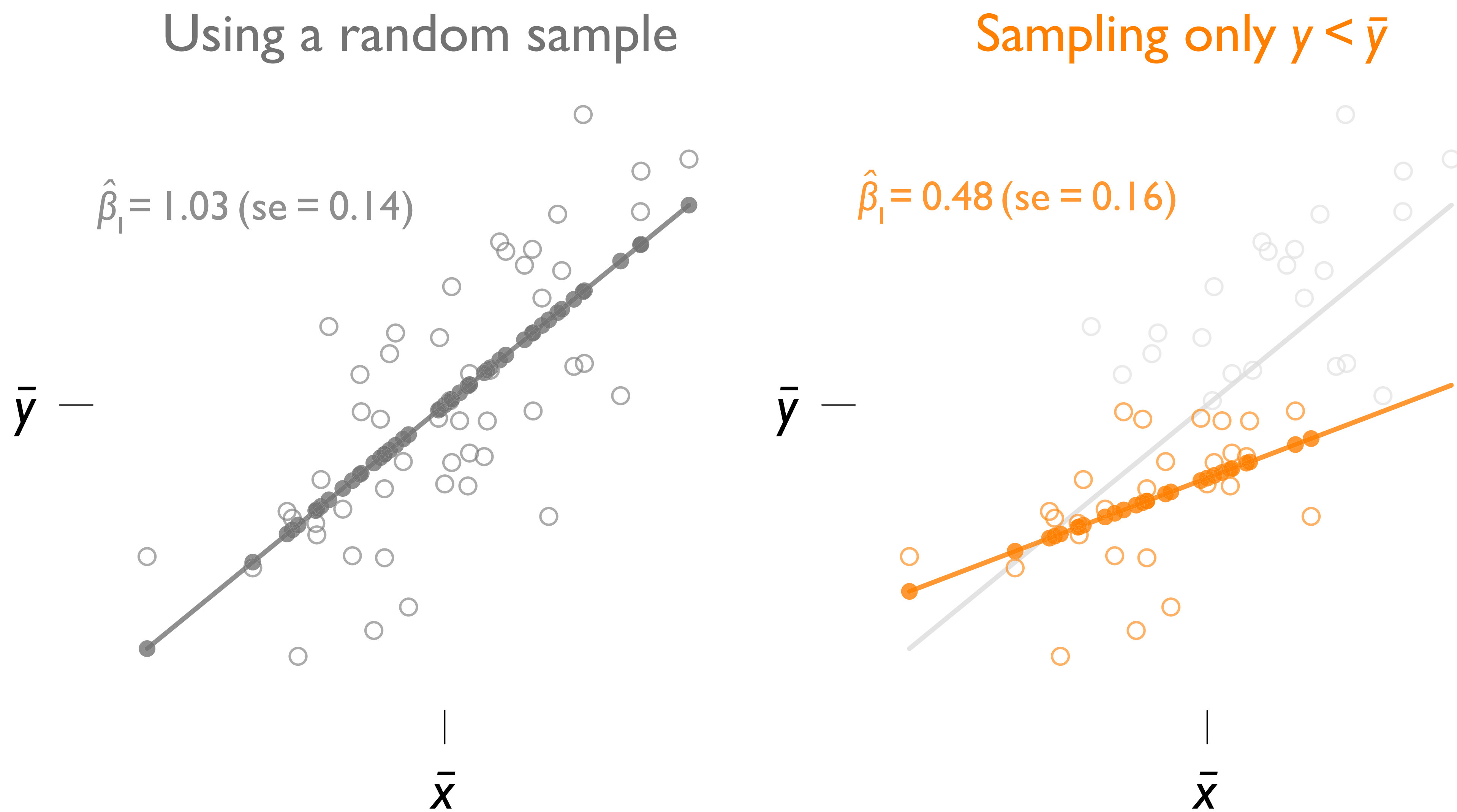
Suppose we have sample selection bias: we can only collect cases with low y

What happens if we run a regression on the orange dots only?



This pattern of missingness biased our result biased towards 0,
whether we selected cases intentionally or had them selected for us by accident

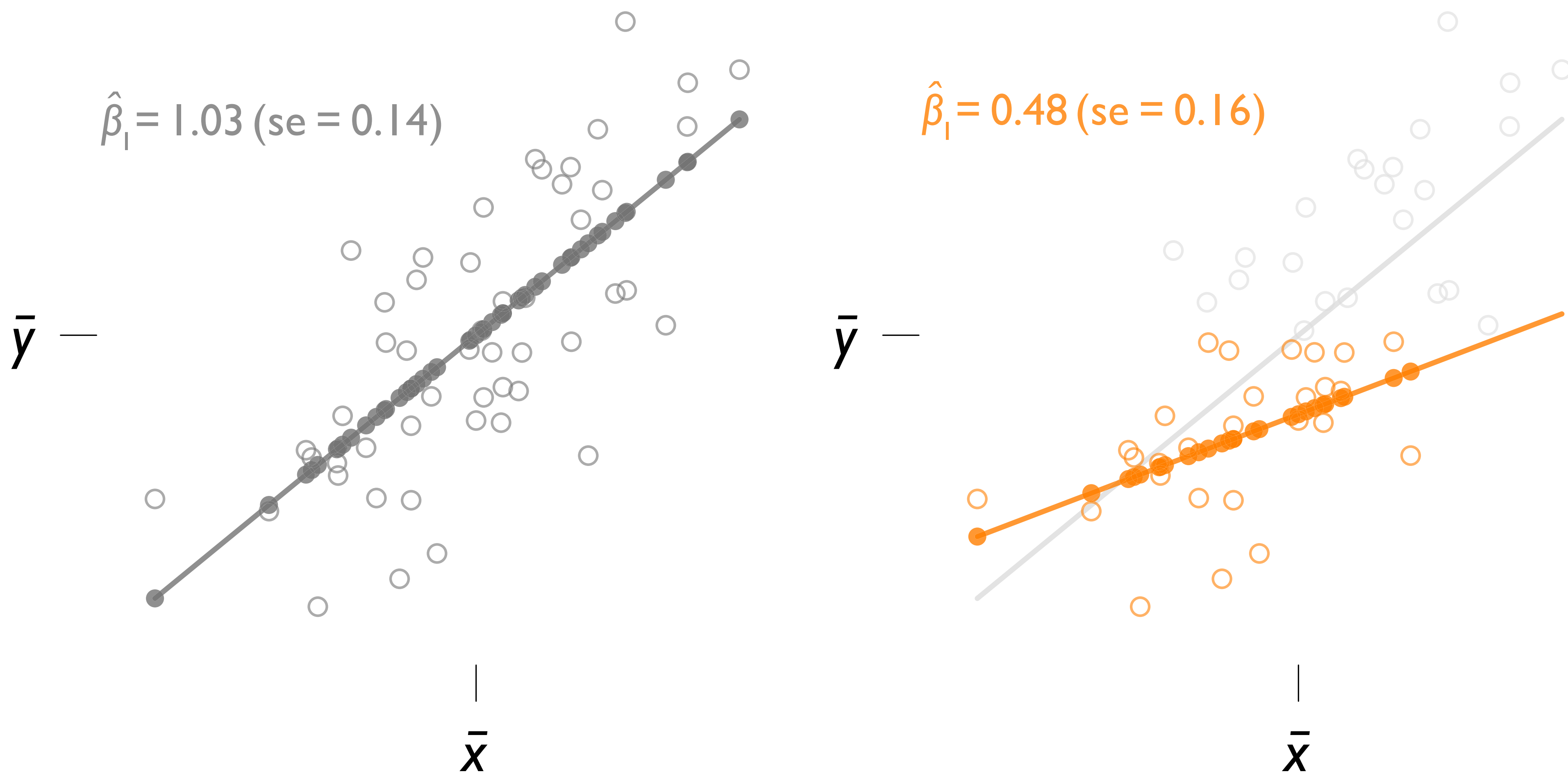
Why? Selecting on y truncates the variation in outcomes, but not in covariates



If I call this *sample selection bias* or *compositional bias*, all would agree I have a serious problem

Using a random sample

Sampling only $y < \bar{y}$



If I call this *sample selection bias* or *compositional bias*,
all would agree I have a serious problem

If I say "I had some missing data, so I listwise deleted," would you object as strongly?

Agenda

Why listwise deletion can be harmful

Why crude methods of imputation are no cure

A generic approach to multiple imputation

When multiple imputation is most needed

Alternative methods of multiple imputation

Practical considerations

Sources

The methods and ideas emphasized here come from:

Gary King et al (2001) "Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation", *American Political Science Review*

James Honaker and Gary King (2010) "What to Do about Missing Values in Time-Series Cross-Section Data", *American Journal of Political Science*

Stef van Buuren and Karin Groothuis-Oudshoorn (2011) "`mice`: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software*

while the classic source on missing data imputation is

Roderick Little and Donald Rubin (2002), *Statistical Analysis with Missing Data,* 2nd Ed., Wiley.

From a certain point of view, all inference problems are missing data problems; we could just treat unknown parameters as "missing data"

For today, we will just consider missingness in the data itself

A Monte Carlo experiment

$$
\begin{aligned}
y_i &= -1x_i + 1z_i + \varepsilon_i \\
\left[\begin{array}{c} x_i \\ z_i \end{array}\right] &\sim \text{Multivariate Normal}\left(\left[\begin{array}{c} 0 \\ 0 \end{array}\right], \left[\begin{array}{cc} 1 & -0.5 \\ -0.5 & 1 \end{array}\right]\right) \\
\varepsilon &\sim \text{Normal}(0, 4)
\end{aligned}
$$

We will create some data using this model, then delete some of it,
and compare the effectiveness of different methods of coping with missing data

In our data, y and z_i are always observed, but x_i is sometimes missing

In our setup, we allow this to happen 3 different ways. . .

A Monte Carlo experiment

$$y_i = -1x_i + 1z_i + \varepsilon_i$$

$$\begin{bmatrix} x_i \\ z_i \end{bmatrix} \sim \text{Multivariate Normal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \right)$$

$$\varepsilon \sim \text{Normal}(0, 4)$$

Missing at random given z_i: Probability of missingness a function of quantile of z_i: 60% at $\min z_i$, 30% at 25th percentile of z_i, 0% at median and above

Missing at random given y_i: Probability of missingness a function of quantile of y_i: 60% at $\min y_i$, 30% at 25th percentile of y_i, 0% at median and above

Missing completely at random: In addition to the above conditional missingness, 20% of the time, x_i is missing regardless of the values of z_i and y_i

A Monte Carlo experiment

$$
\begin{aligned}
y_i &= -1x_i + 1z_i + \varepsilon_i \\
\begin{bmatrix} x_i \\ z_i \end{bmatrix} &\sim \text{Multivariate Normal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \right) \\
\varepsilon &\sim \text{Normal}(0, 4)
\end{aligned}
$$

Net effect of three patterns of missingness: x_i missing about 60% of the time

In our experiments, we will simulate 200 observations:

about 120 will be missing, and about 80 will be full observed

Exact number of missing cases will vary randomly from dataset to dataset

A Monte Carlo experiment

$$\begin{aligned}
\text{Democracy}_i &= -1 \times \text{Inequality}_i + 1 \times \text{GDP}_i + \varepsilon_i \\
\begin{bmatrix} \text{Inequality}_i \\ \text{GDP}_i \end{bmatrix} &\sim \text{Multivariate Normal}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \right) \\
\varepsilon &\sim \text{Normal}(0, 4)
\end{aligned}$$

It may help to imagine some context, but remember this example is fictive:

Imagine democracy is hampered by inequality and aided by development,

Inequality tends to be lower in developed countries,

Poorer countries & non-democracies less likely to collect/publish inequality data,

And sometimes even rich democracies fail to collect such complex data

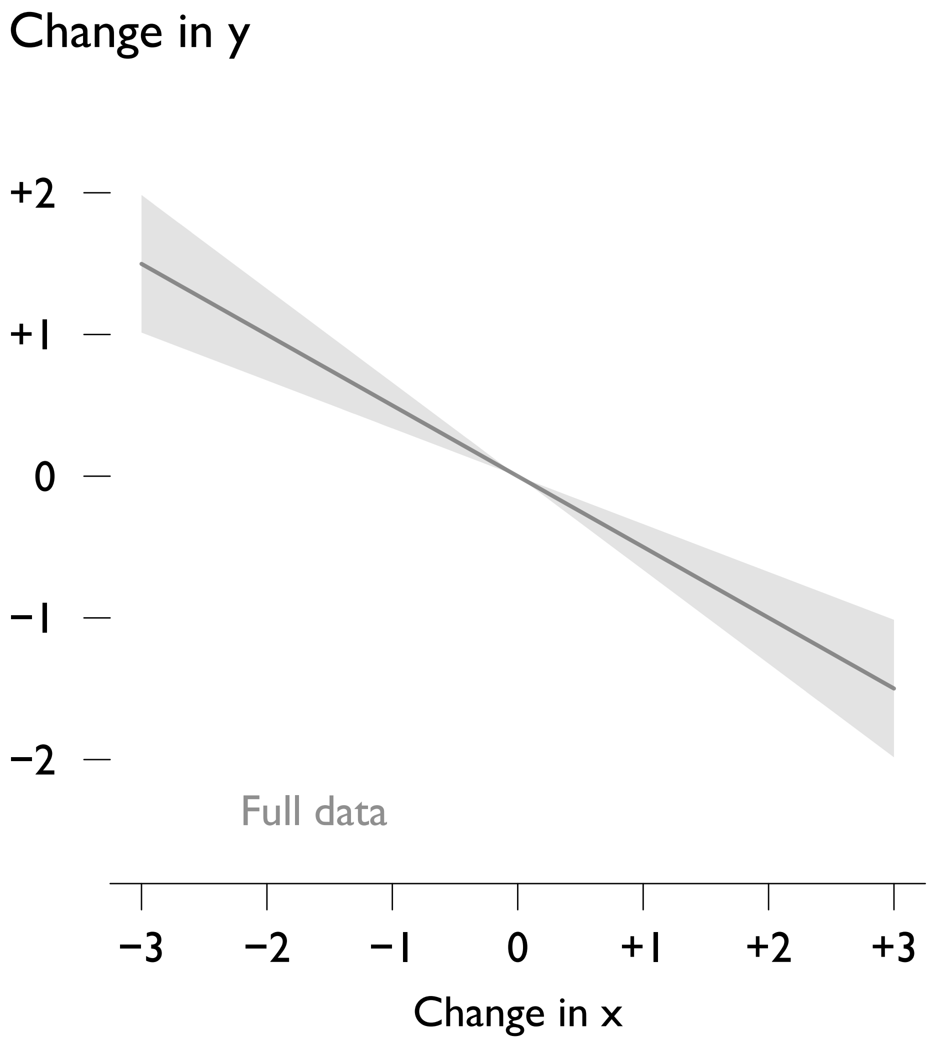
Monte Carlo run 1, fully observed

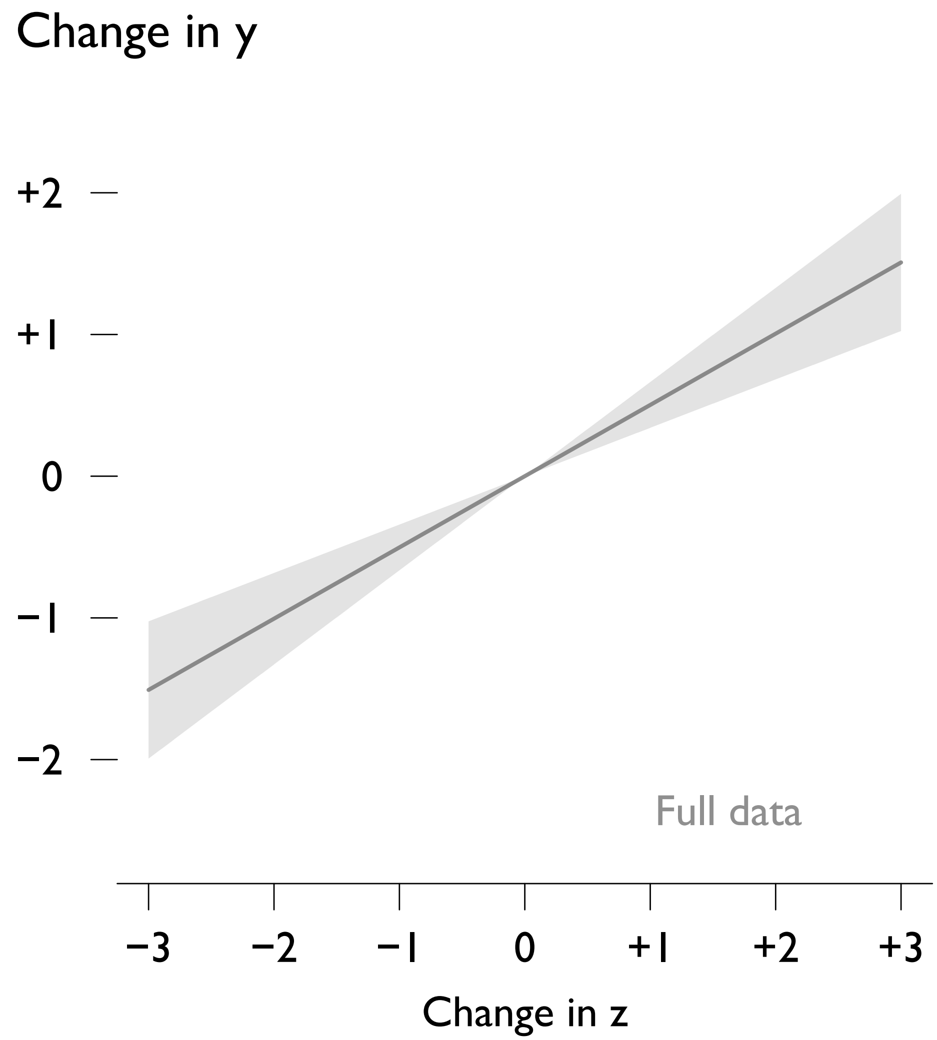
i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	0.27	-0.21
[3]	0.97	-0.05	-0.66
[4]	0.17	-0.46	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	-1.16	0.28
⋮	⋮	⋮	⋮

I will generate many datasets from this true model
as part of the Monte Carlo experiement

But to illustrate how data goes missing and get imputed,
I'll show what happens to the first 6 cases of the first Monte Carlo dataset

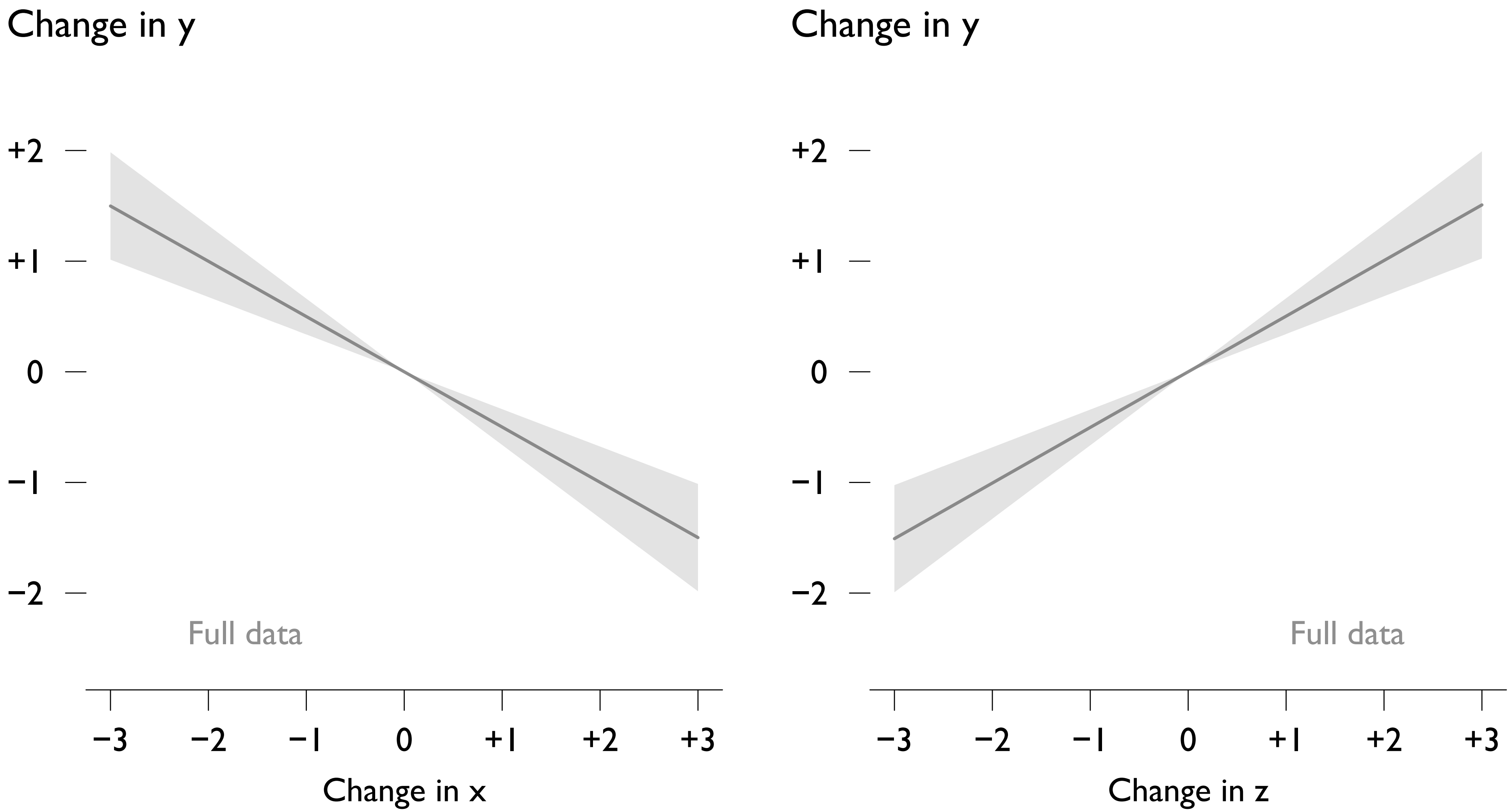
First, let's establish a baseline:
what we would find if we could use the full dataset. . .





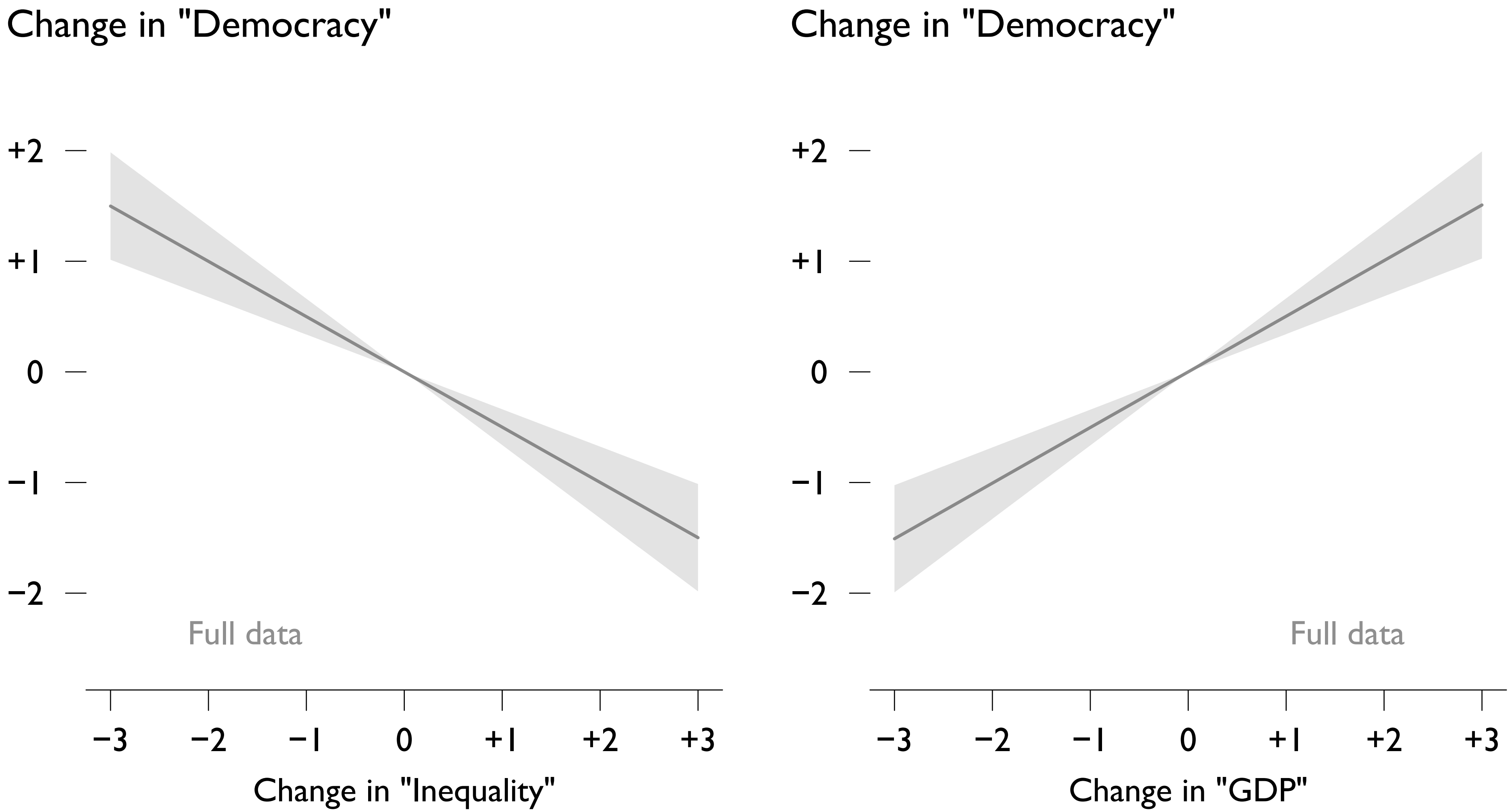
Above shows the first differences we'd get if we fully observed our 200 cases

Our goal henceforth is to reproduce these effects & 95% CIs as closely as possible



For all first difference plots, I've actually averaged results after running the whole experiment (creating a dataset, then estimating the model) 1000×

This eliminated Monte Carlo error, and shows us what will happen on average for each missing data strategy



To make the example easier to follow,
I've replaced x, y, and z with our fictive variable names

Of course, we don't have any real evidence on this hypothetical research question;
all the data are made up

Costs of listwise deletion

Our dataset contains 3 variables and 200 cases

But for about 120 of our cases, a single variable has a missing value

This means that only $120/(3 \times 200) = 20\%$ of our cells are missing

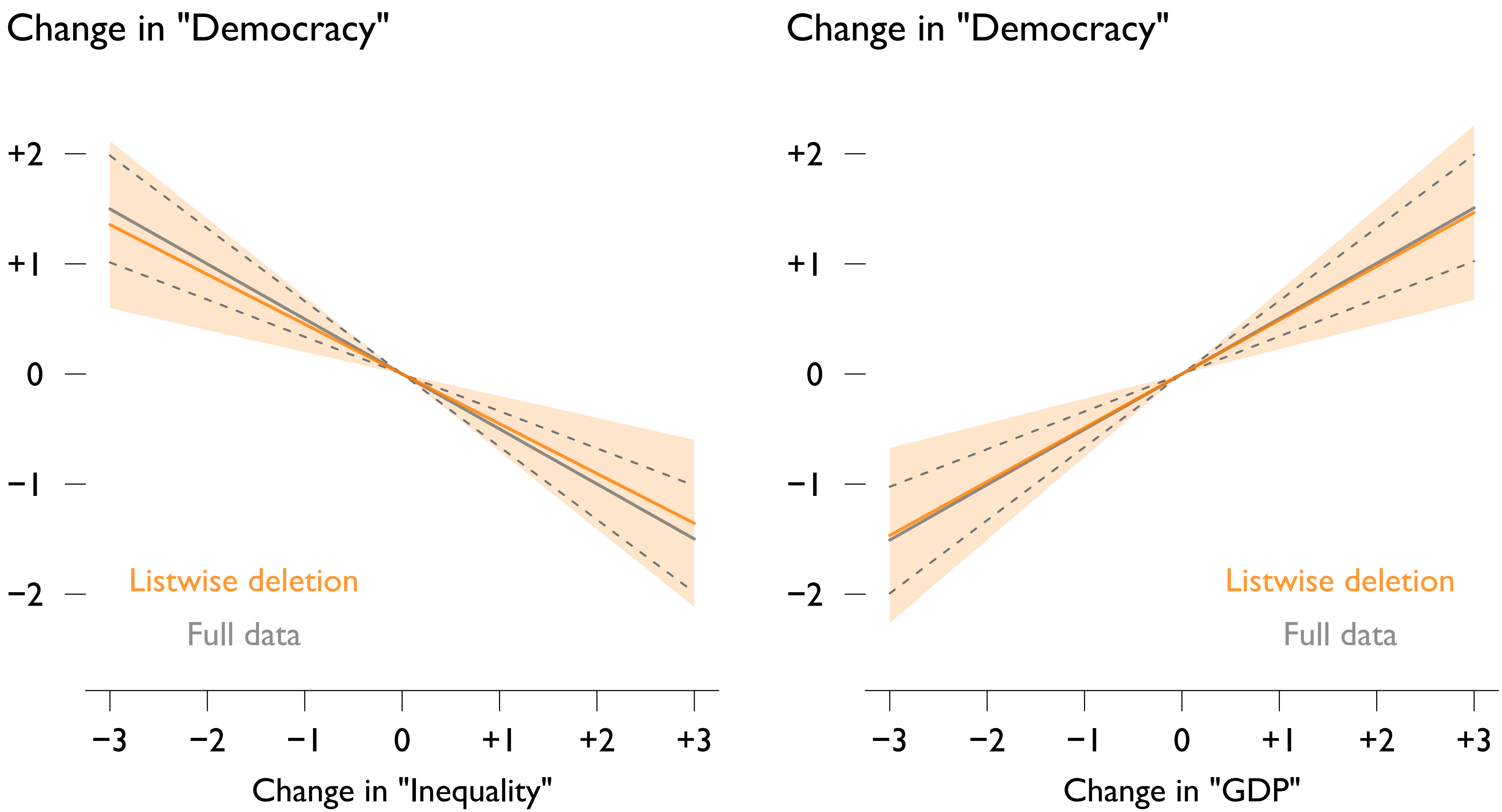
But listwise deletion will remove 60% of our cases,
increasing standard errors considerably

We've thrown away 240 cells containing actual data – *half* the observed cells

Imagine collecting your dataset by hand, then tossing half of it the trash

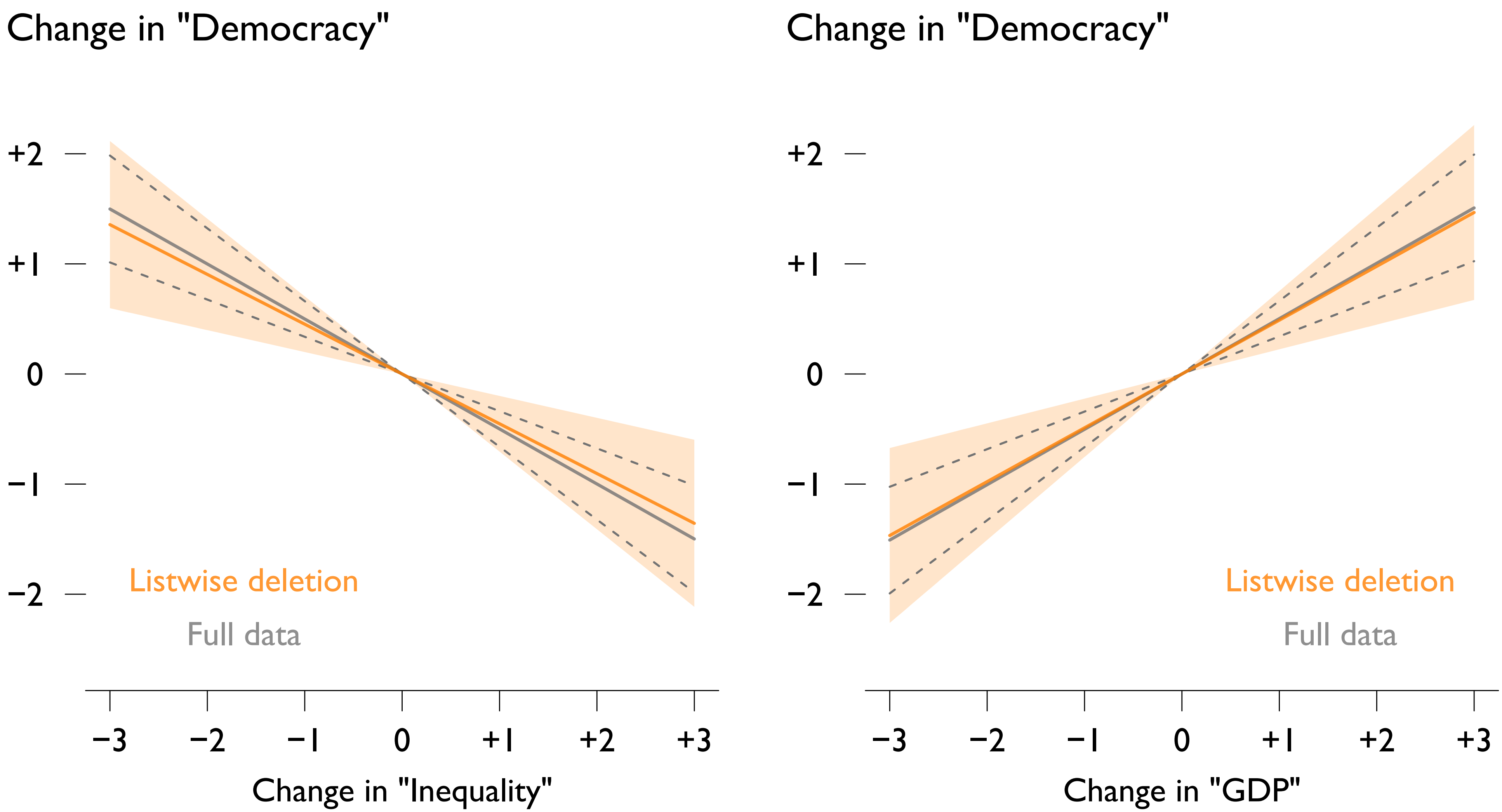
But this isn't just wasted data collection effort:

listwise deletion is statistically inefficient
and often creates statistical bias



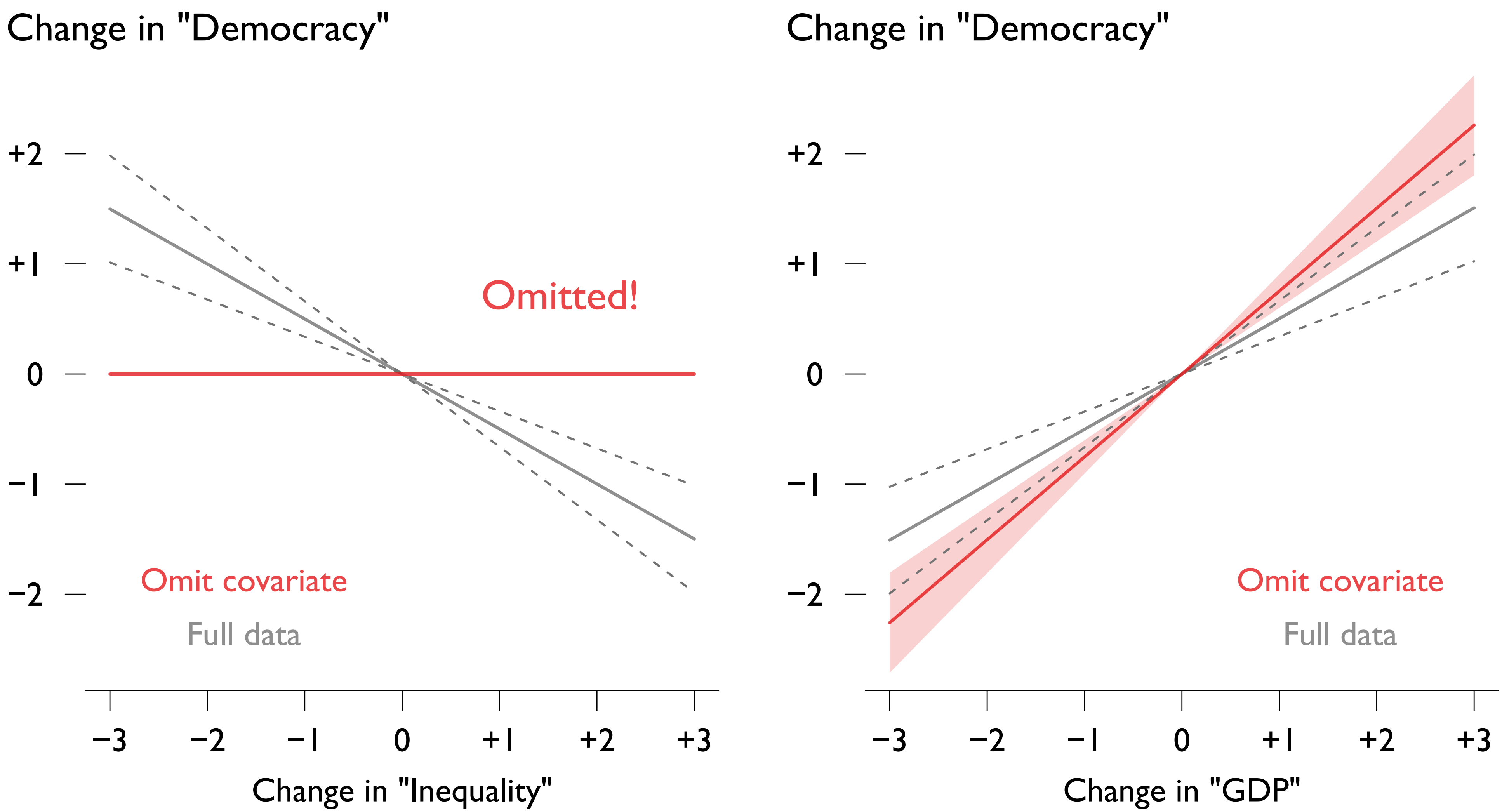
In our hypothetical example, listwise deletion is biased:
the relationship between Democracy & Inequality is attenuated

It's also inefficient: CIs are wider than they should be,
so we might fail to detect significant relationships because of missingness



Why did we listwise delete?

Why not drop Inequality from the model instead?



Even if we didn't care about estimating the relationship between Inequality and GDP, we still need it in the model

Including Inequality is necessary to get unbiased estimates of the effect of GDP, because it is correleted with both Inequality & Democracy

Crude imputation methods don't help

Listwise deletion just trades one problem – omitted variable bias – for another – inefficiency and possible bias from sample selection

The latter occurs, as in the introductory example, when the missingness of a covariate is correlated with the value of the outcome

If both approaches are statistically flawed, what about filling in the missing data?

Crude imputation methods don't help

Listwise deletion just trades one problem – omitted variable bias – for another – inefficiency and possible bias from sample selection

The latter occurs, as in the introductory example, when the missingness of a covariate is correlated with the value of the outcome

If both approaches are statistically flawed, what about filling in the missing data?

This approach called *imputation*, and there are obvious crude methods:

Mean imputation Fill in missing x_i's with unconditional expected values, $\bar{x}_i$

Single imputation Fill in missing x_i's with conditional expected values, $\mathbb{E}(x_i|y_i, z_i)$

Neither crude approach works

Both are *worse* than listwise deletion most of the time

Monte Carlo run 1, with missing values

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	NA	-0.21
[3]	0.97	NA	-0.66
[4]	0.17	NA	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	NA	0.28
⋮	⋮	⋮	⋮

Above are the first six observations, now showing the effects of missing data

Mean imputation says to replace each `NA` with the observed mean of that variable

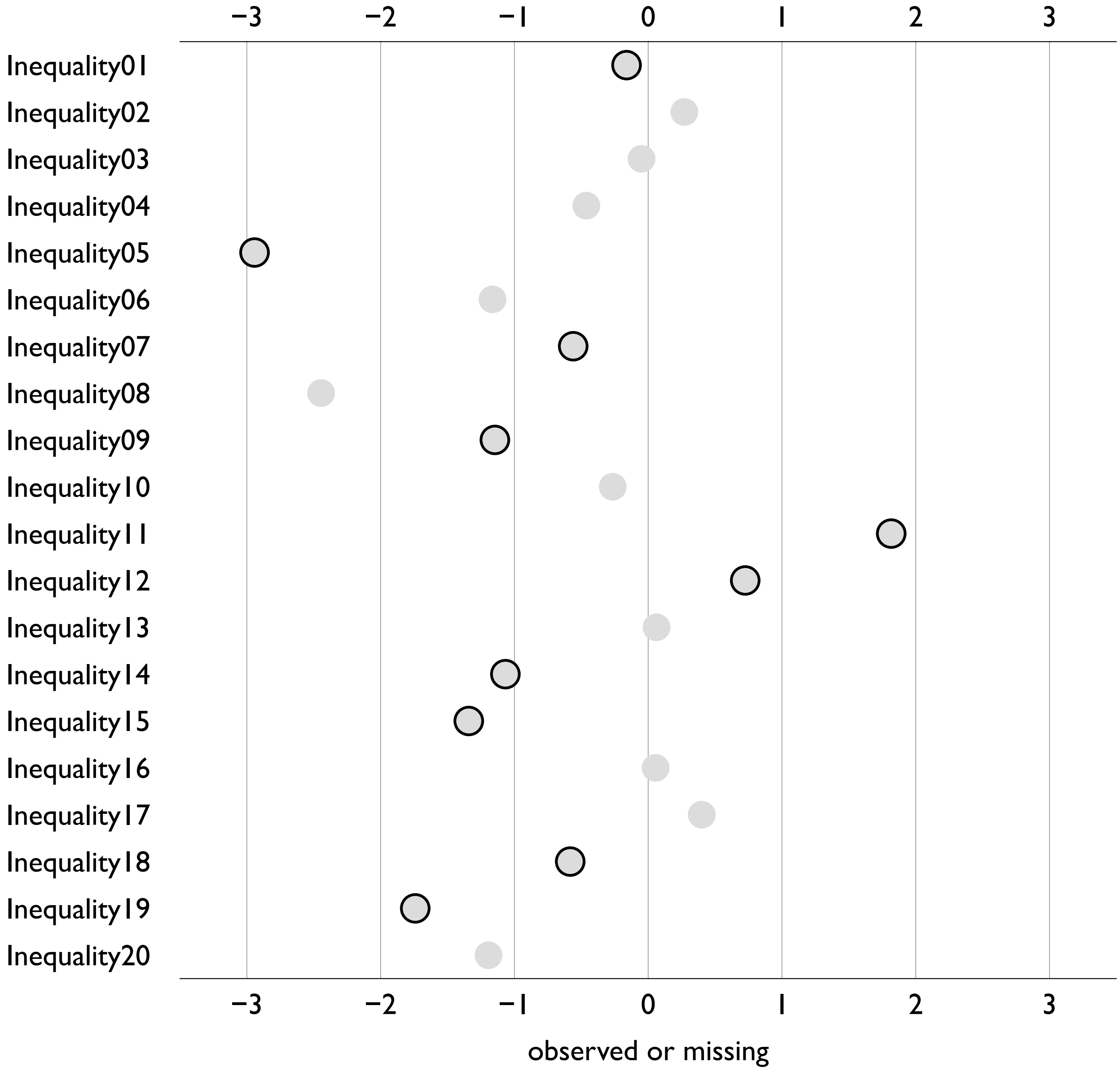
Monte Carlo run 1, mean imputation

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	-0.23	-0.21
[3]	0.97	-0.23	-0.66
[4]	0.17	-0.23	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	-0.23	0.28
⋮	⋮	⋮	⋮

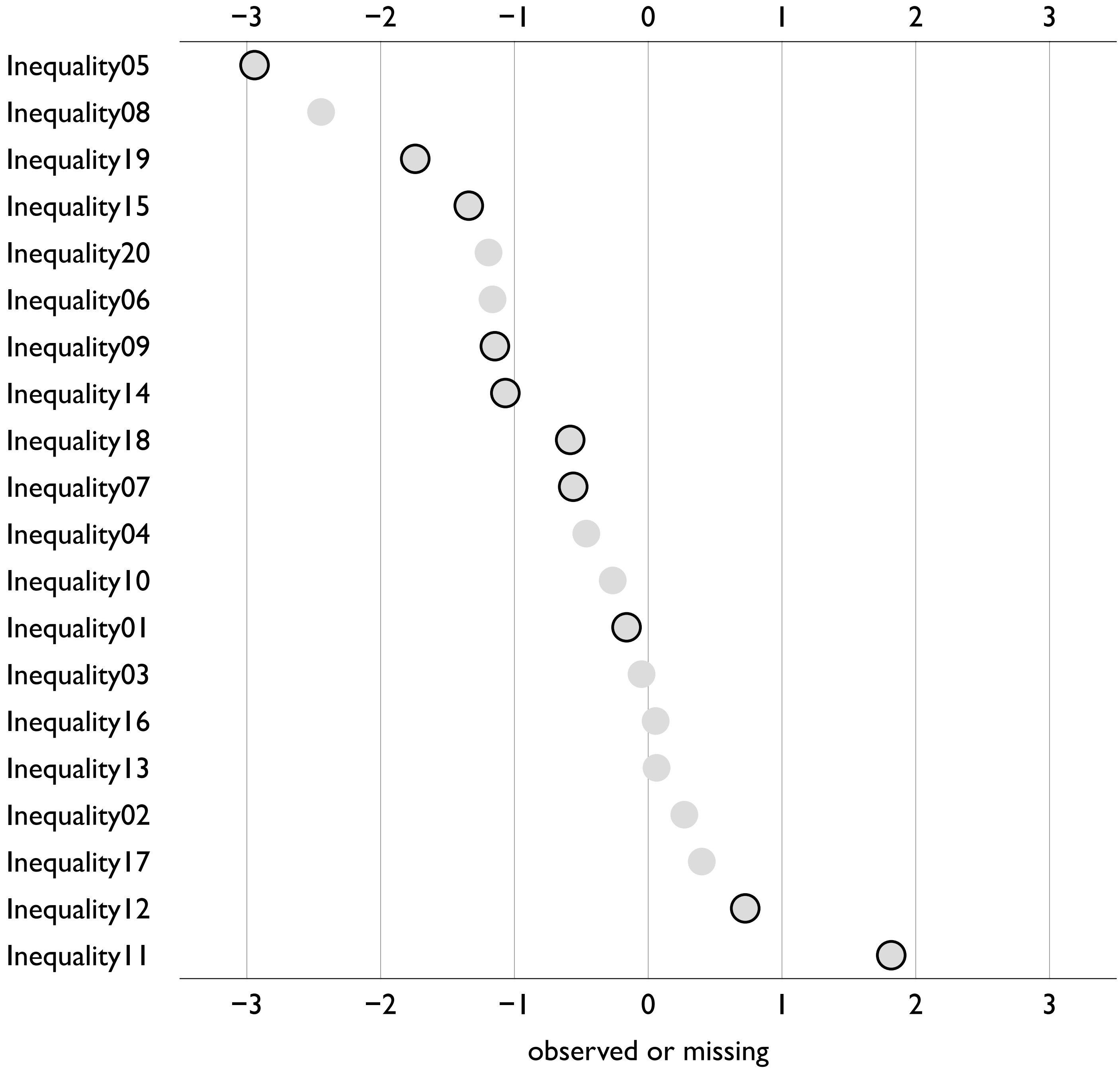
Above are the first six observations, now showing the effects of missing data

Mean imputation says to replace each `NA` with the observed mean of that variable

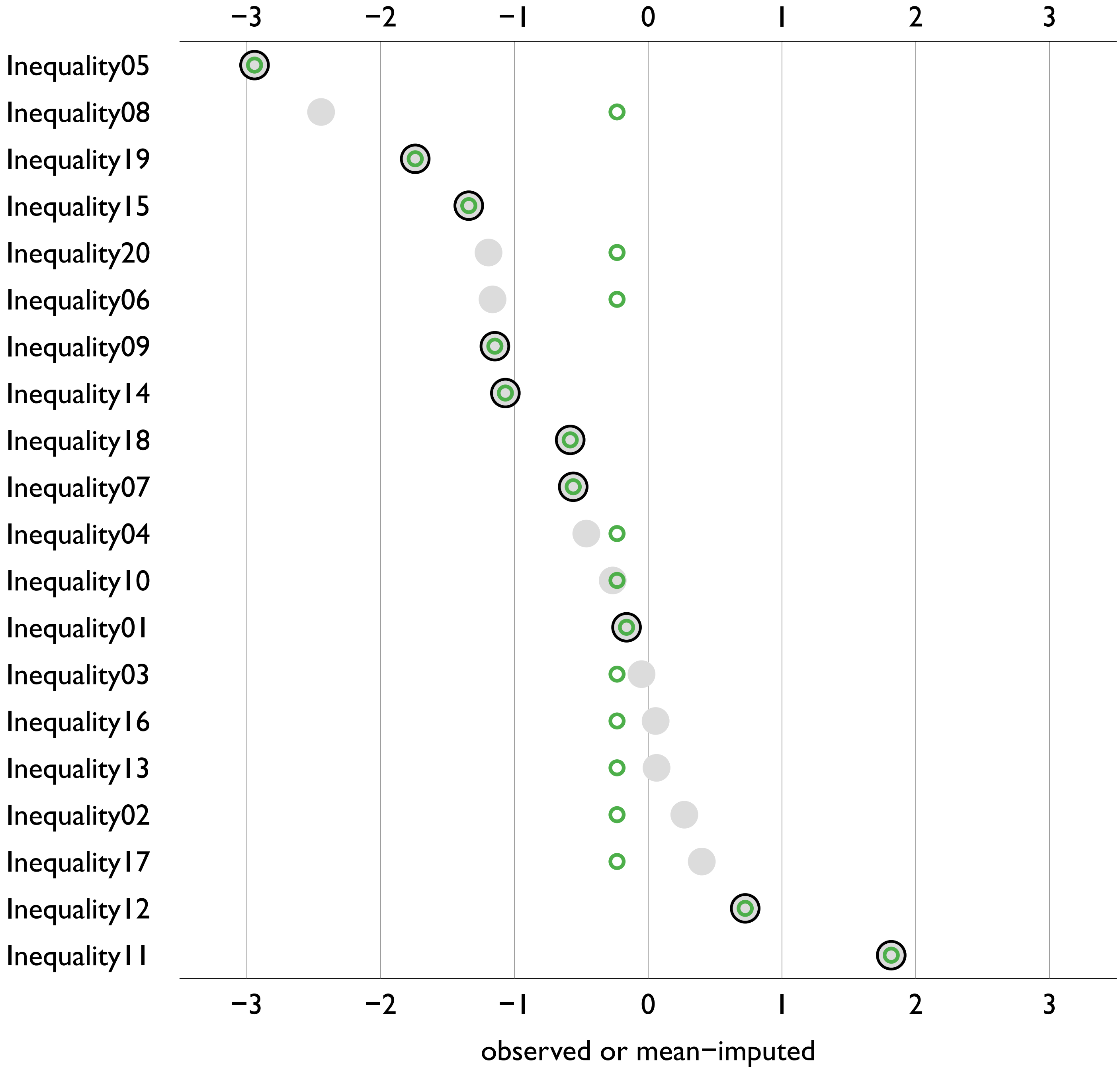
The observed mean of Inequality is -0.23



A visual representation of the first 20 cases, with non-missing cases ringed in black

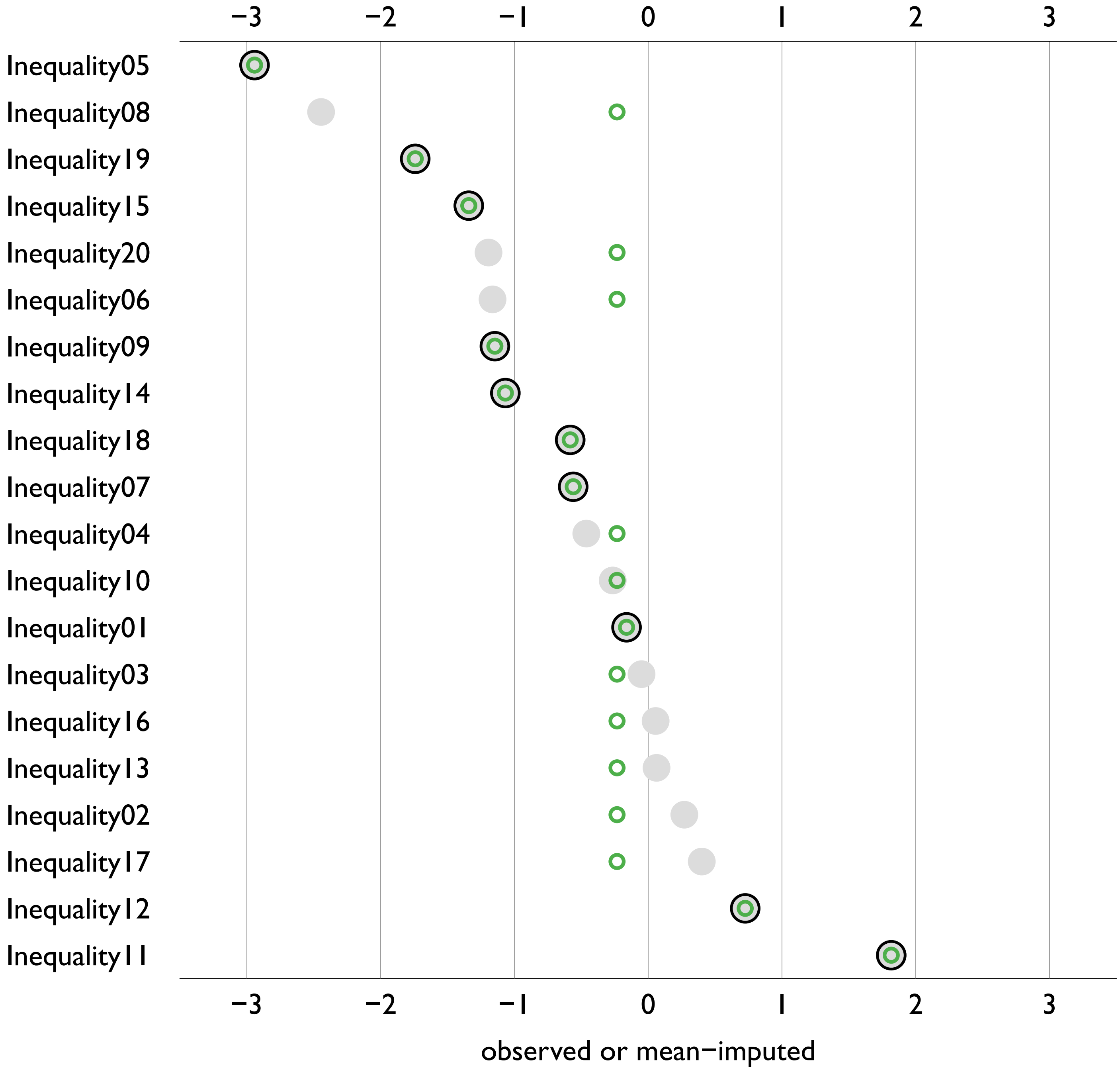


Sorting the cases by level of Inequality will aid comparison across methods

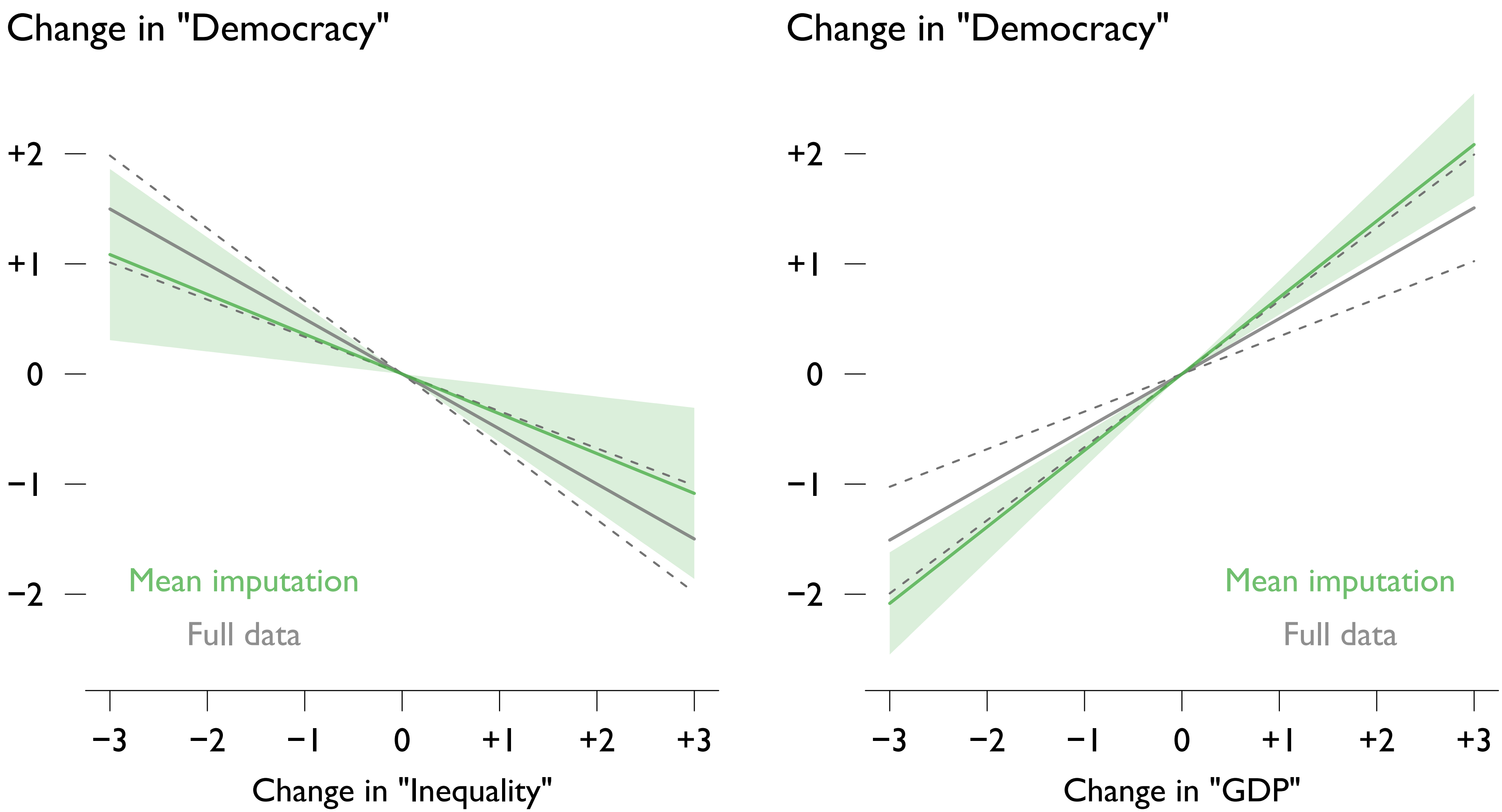


The mean-imputation completed dataset

Remind you of anything?



We've created a mixed distribution: half real data, half very different!



Mean imputation biases coefficients for missing variables downwards

And biases correlated observed variables upwards

Why did this happen?

Why mean imputation doesn't work

1. *Filling in missings with the mean* assumes
there's no relationship among our variables

But the whole reason for the model is to *measure* the conditional relationship!

Why mean imputation doesn't work

1. *Filling in missings with the mean* assumes
there's no relationship among our variables

But the whole reason for the model is to *measure* the conditional relationship!

For example, we to fill in the sixth observation, we need
$\mathbb{E}(\text{Inequality}_6|\text{Democracy}_6, \text{GDP}_6)$, not the unconditional $\mathbb{E}(\text{Inequality})$

If Democracy is low in case 6,
and if Democracy is inversely correlated with Inequality,
we should fill in a high value, not an average one

Filling in the unconditional mean biases $\hat{\beta}_{\text{Democracy}}$ towards zero

Why mean imputation doesn't work

1. *Filling in missings with the mean* assumes
there's no relationship among our variables

But the whole reason for the model is to *measure* the conditional relationship!

For example, we to fill in the sixth observation, we need
$\mathbb{E}(\text{Inequality}_6|\text{Democracy}_6, \text{GDP}_6)$, not the unconditional $\mathbb{E}(\text{Inequality})$

If Democracy is low in case 6,
and if Democracy is inversely correlated with Inequality,
we should fill in a high value, not an average one

Filling in the unconditional mean biases $\hat{\beta}_{\text{Democracy}}$ towards zero

2. *Missing data has also biased our estimate of the mean,*
and we've translated that bias into our imputations

The true sample mean of Inequality in the fully observed data is -0.03, not -0.23

Monte Carlo run 1, with missing values

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	NA	-0.21
[3]	0.97	NA	-0.66
[4]	0.17	NA	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	NA	0.28
⋮	⋮	⋮	⋮

Mean imputation failed because we didn't take the model into account

If our variables are correlated – and we think they are –
we need to condition on that correlation when imputing

Monte Carlo run 1, with missing values

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	NA	-0.21
[3]	0.97	NA	-0.66
[4]	0.17	NA	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	NA	0.28
⋮	⋮	⋮	⋮

Suppose that we fit the following model for our fully observed cases:

$$\text{Inequality}_i = \gamma_0 + \gamma_1 \text{GDP}_i + \gamma_2 \text{Democracy}_i + \nu_i$$

Monte Carlo run 1, with missing values

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	NA	-0.21
[3]	0.97	NA	-0.66
[4]	0.17	NA	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	NA	0.28
⋮	⋮	⋮	⋮

Suppose that we fit the following model for our fully observed cases:

$$\text{Inequality}_i = \gamma_0 + \gamma_1 \text{GDP}_i + \gamma_2 \text{Democracy}_i + \nu_i$$

And then use the fitted values to fill-in missing values of Inequality j:

$$\mathbb{E}(\text{Inequality}_j) = \hat{\gamma}_0 + \hat{\gamma}_0 \text{GDP}_j + \hat{\gamma}_2 \text{Democracy}_j$$

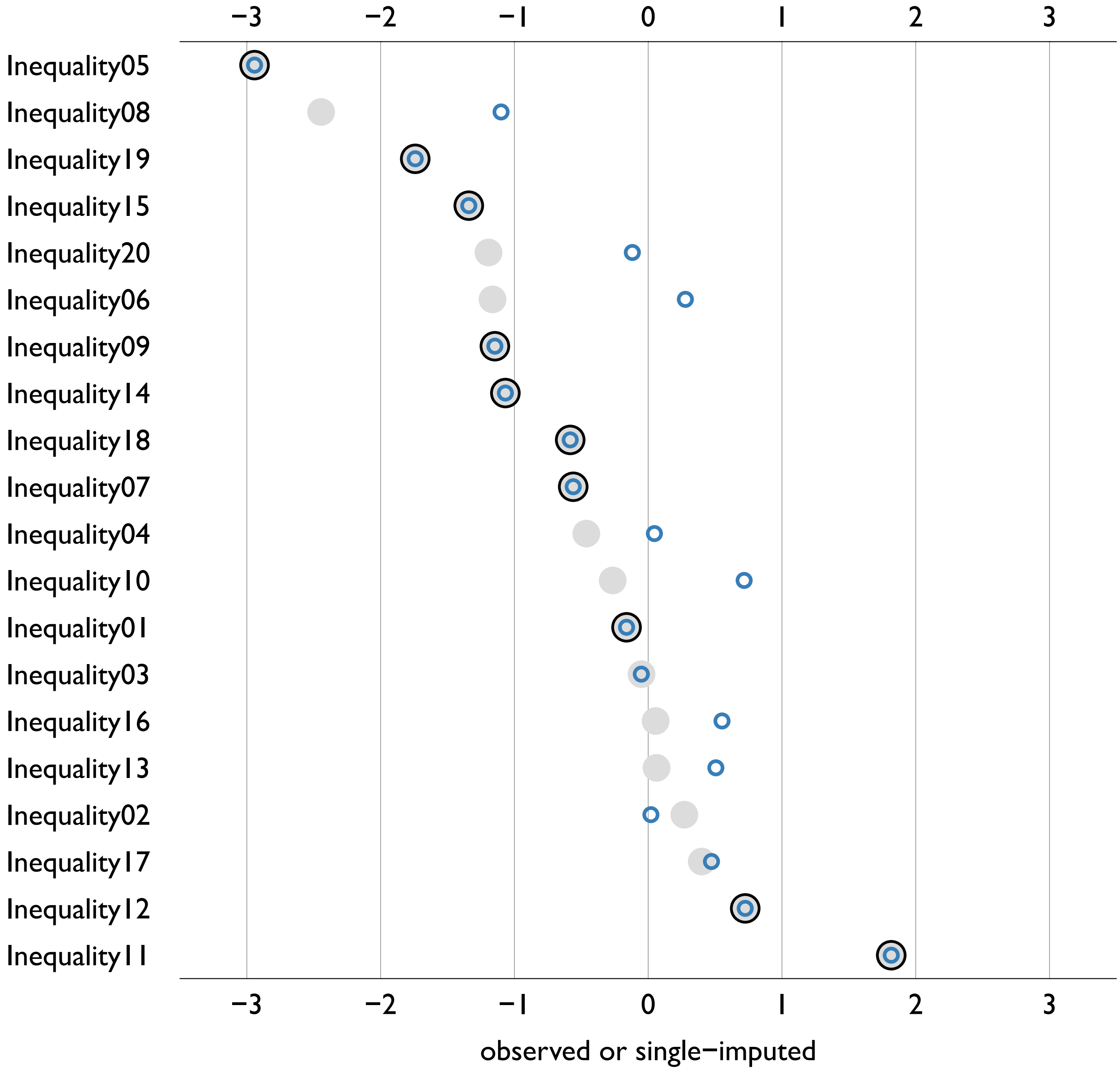
Monte Carlo run 1, single imputation

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	0.02	-0.21
[3]	0.97	-0.05	-0.66
[4]	0.17	0.05	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	0.28	0.28
⋮	⋮	⋮	⋮

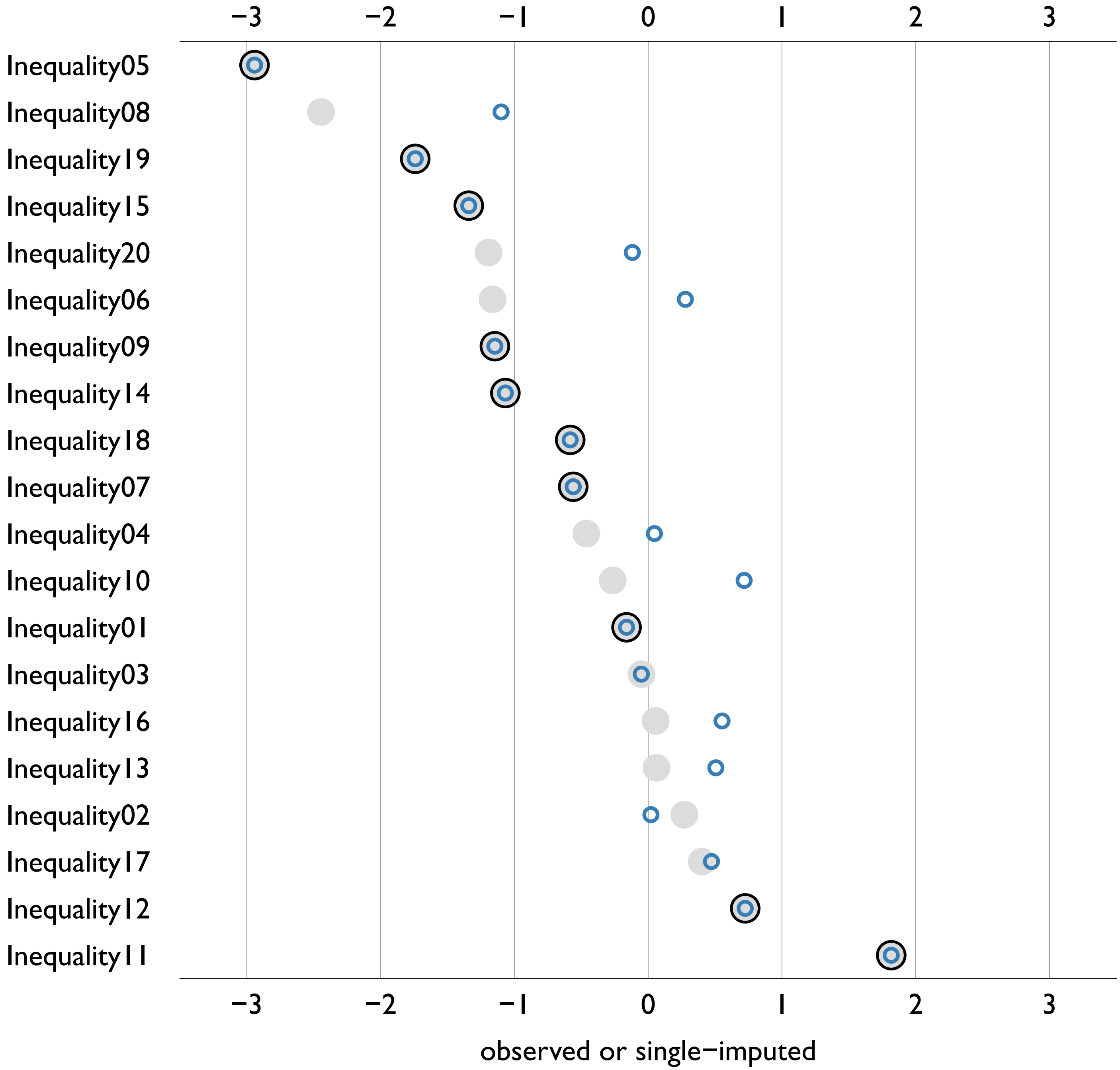
This seems better:
the imputed Inequality values at least seem consistent with the rest of the data

As noted, observation 6 has low democracy and is imputed to have higher inequality

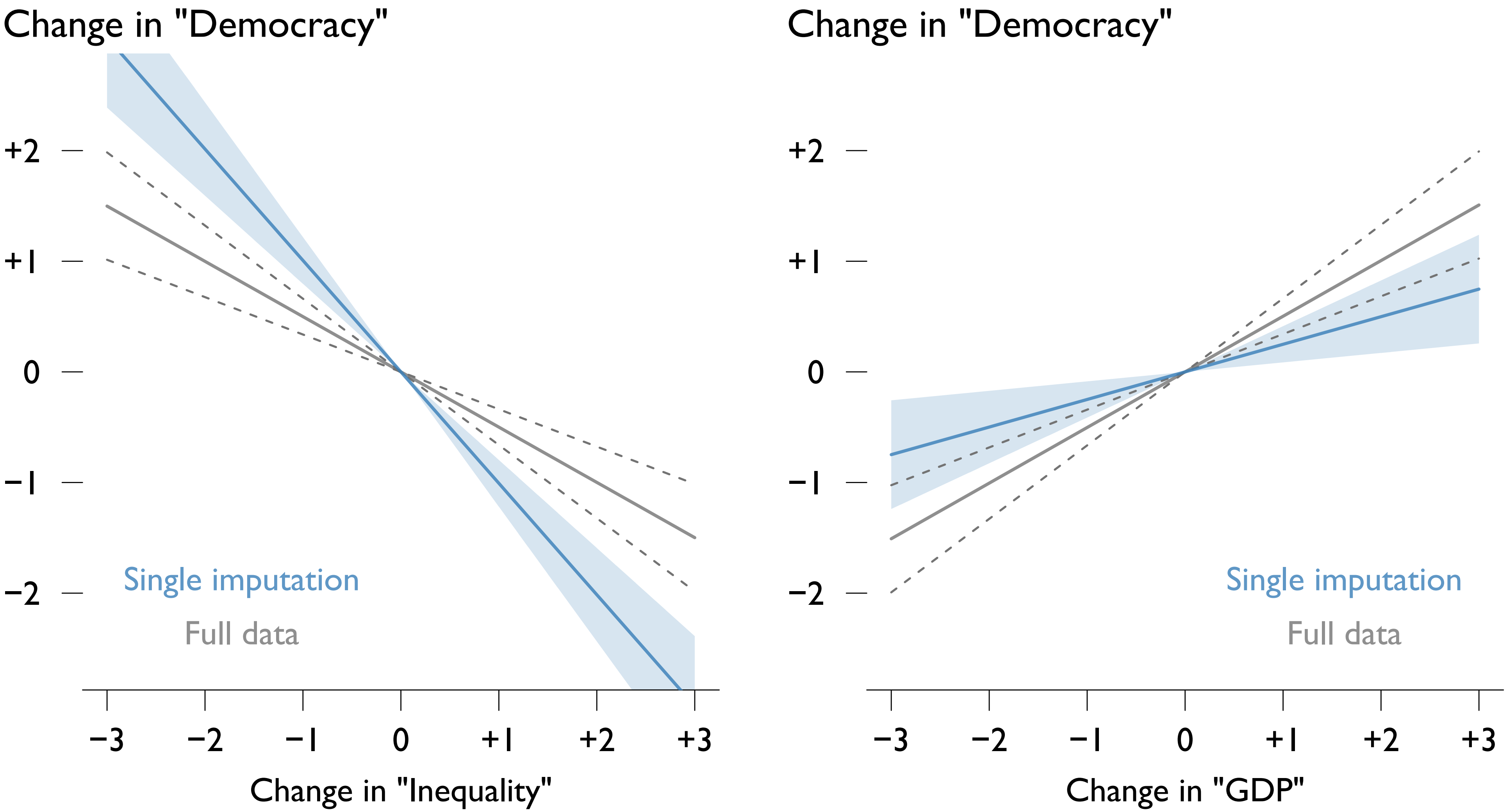
But actually, what we've done is *worse* than before



Our imputations still miss by a lot – yet we treat them as *data*



For example, case 6 had a large random error – it's much lower than expected



Single imputation biases imputed variables upwards

And biases correlated observed variables downwards

Why did this happen?

Why single imputation doesn't work

1. *We assumed any missing values were exactly equal to their conditional expected values, with* no error

But randomness is fundamental to all real world variables – none of our other variables are deterministic functions of covariates

$\rightarrow$ we've assumed that the cases we didn't see are more consistent with our model than the cases we did see!

This leads to considerable overconfidence, and biases our β's upwards

Why single imputation doesn't work

1. *We assumed any missing values were exactly equal to their conditional expected values, with* no error

But randomness is fundamental to all real world variables – none of our other variables are deterministic functions of covariates

$\rightarrow$ we've assumed that the cases we didn't see are more consistent with our model than the cases we did see!

This leads to considerable overconfidence, and biases our β's upwards

2. *How would we implement this approach consistently across cases if different or multiple variables are missing?*

Why single imputation doesn't work

1. *We assumed any missing values were exactly equal to their conditional expected values, with* no error

But randomness is fundamental to all real world variables – none of our other variables are deterministic functions of covariates

$\rightarrow$ we've assumed that the cases we didn't see are more consistent with our model than the cases we did see!

This leads to considerable overconfidence, and biases our β's upwards

2. *How would we implement this approach consistently across cases if different or multiple variables are missing?*

3. *The linear model of Inequality is still estimated using listwise deletion, so the bias from LWD still passes on to our imputations*

This last objection suggests an infinite regress – how do we escape it?

Multiple imputation

Goals: (1) treat all observed values in our original data as known with certainty; (2) summarize the *uncertainty* about missing values implied by the observed data

Multiple imputation

Goals: (1) treat all observed values in our original data as known with certainty; (2) summarize the *uncertainty* about missing values implied by the observed data

Specifically, the method should

1. Impute our missing values conditional on the structure of the *full* dataset

Multiple imputation

Goals: (1) treat all observed values in our original data as known with certainty;
(2) summarize the *uncertainty* about missing values implied by the observed data

Specifically, the method should

1. Impute our missing values conditional on the structure of the *full* dataset

2. Include the uncertainty in our estimation of the missings,
as we'll never be sure we have the right estimates

Multiple imputation

Goals: (1) treat all observed values in our original data as known with certainty;
(2) summarize the *uncertainty* about missing values implied by the observed data

Specifically, the method should

1. Impute our missing values conditional on the structure of the *full* dataset

2. Include the uncertainty in our estimation of the missings,
as we'll never be sure we have the right estimates

3. Includes the randomness of real world variables,
which can't be exactly predicted even by the true model

Multiple imputation is a family of methods that achieve these goals

Unless stringent assumptions are met, MI improves on listwise deletion

We start with the King, Honaker *et al* method known as Amelia

How Amelia works

Take all the data – the outcome, covariates, even "auxilliary variables" correlated with them but not part of the model – and place them in a matrix $\mathbf{D}$

How Amelia works

Take all the data – the outcome, covariates, even “auxilliary variables” correlated with them but not part of the model – and place them in a matrix $\mathbf{D}$

Call the known elements of this matrix $\mathbf{D}_{\text{obs}}$, and the missing elements $\mathbf{D}_{\text{miss}}$

How Amelia works

Take all the data – the outcome, covariates, even "auxilliary variables" correlated with them but not part of the model – and place them in a matrix $\mathbf{D}$

Call the known elements of this matrix $\mathbf{D}_{\text{obs}}$, and the missing elements $\mathbf{D}_{\text{miss}}$

Key assumption of Amelia: all these variables are jointly multivariate normal

$$\mathbf{D} \overset{\text{iid}}{\sim} \text{Multivariate Normal}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

To impute missing elements of $\mathbf{D}$, we first need to estimate $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

The iid MVN assumption implies this likelihood for the joint distribution of the data

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{D}) = \prod_{i=1}^{N} f_{\mathcal{MVN}}(\mathbf{d}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

where $\mathbf{d}_i$ refers to the ith observation in the dataset $\mathbf{D}$

How Amelia works

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{D}) = \prod_{i=1}^{N} f_{\mathcal{MVN}}(\mathbf{d}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

If we knew the true $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we could use them to draw several predicted values of the missing values $\mathbf{D}_{\text{miss}}$ and fill them into several new predicted “copies” of our dataset $\tilde{\mathbf{D}}$

How Amelia works

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma} | \mathbf{D}) = \prod_{i=1}^{N} f_{\mathcal{MVN}}(\mathbf{d}_i | \boldsymbol{\mu}, \boldsymbol{\Sigma})$$

If we knew the true $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we could use them to draw several predicted values of the missing values $\mathbf{D}_{\text{miss}}$ and fill them into several new predicted "copies" of our dataset $\tilde{\mathbf{D}}$

Each copy of the dataset would contain the known values for $\mathbf{D}_{\text{obs}}$,
but a different set of predicted draws for $\tilde{\mathbf{D}}_{\text{miss}}$

How Amelia works

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{D}) = \prod_{i=1}^{N} f_{\mathcal{MVN}}(\mathbf{d}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

If we knew the true $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we could use them to draw several predicted values of the missing values $\mathbf{D}_{\text{miss}}$ and fill them into several new predicted "copies" of our dataset $\tilde{\mathbf{D}}$

Each copy of the dataset would contain the known values for $\mathbf{D}_{\text{obs}}$,
but a different set of predicted draws for $\tilde{\mathbf{D}}_{\text{miss}}$

Variation across $\tilde{\mathbf{D}}_{\text{miss}}$ would summarize uncertainty about these imputations,

while the mean value of $\tilde{\mathbf{D}}_{\text{miss}}$ would capture the expected value the missing data

Often even a small number of imputed datasets is enough to summarize uncertainty

How Amelia works

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{D}) = \prod_{i=1}^{N} f_{\mathcal{MVN}}(\mathbf{d}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

But we *don't* know the true $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$

If we try to estimate them from $\mathbf{D}_{\text{obs}}$ only using listwise deletion, we will have biased estimates, as in single imputation

How Amelia works

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{D}) = \prod_{i=1}^{N} f_{\mathcal{MVN}}(\mathbf{d}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Instead, we use a method called *Expectation Maximization* (EM)
which iterates back and forth between two steps:

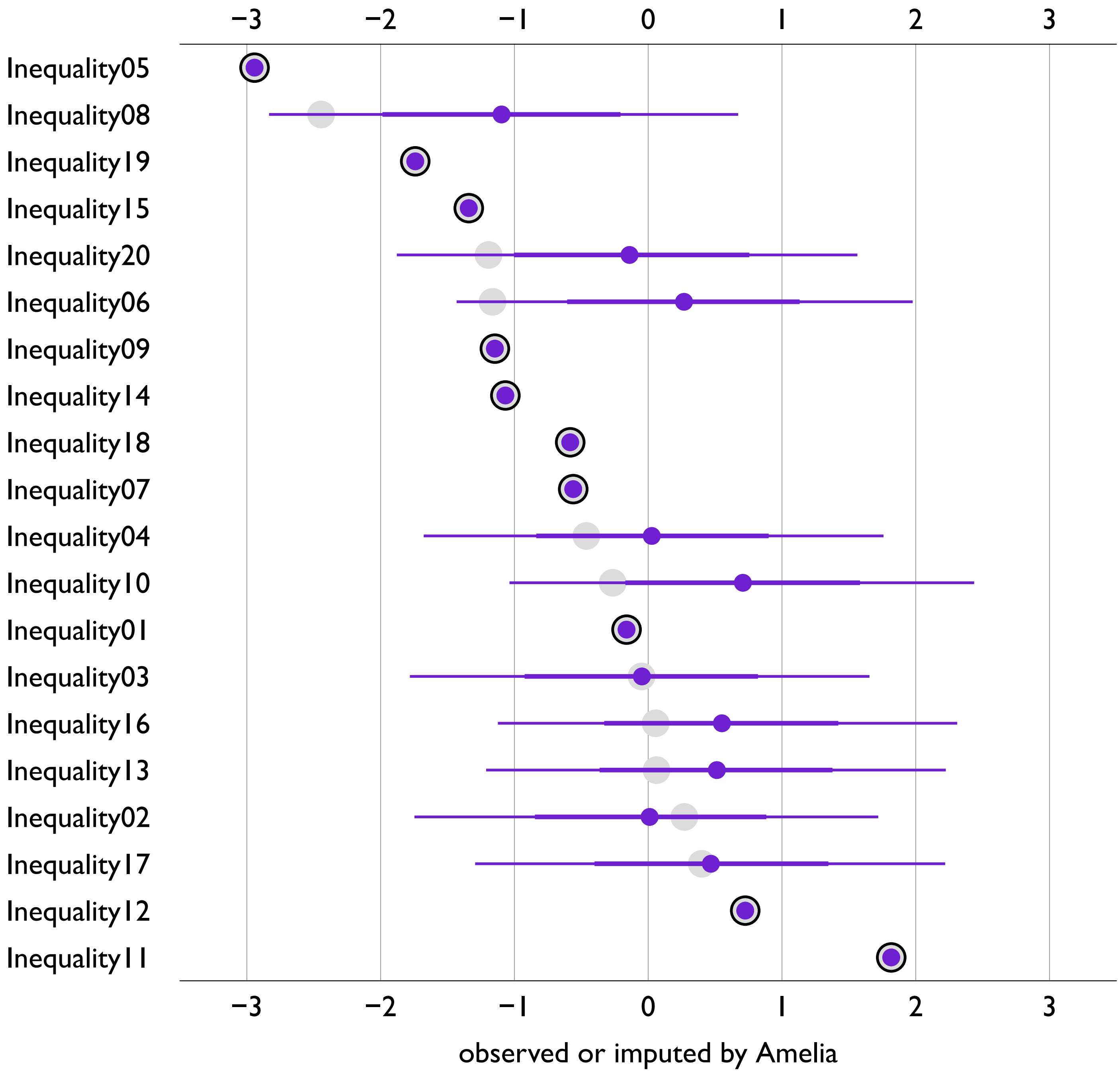
Expectation step Use the estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ to fill in missing data $\mathbf{D}_{\text{miss}}$

Maximization step Use the filled-in matrix $\mathbf{D}$ to estimate $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$

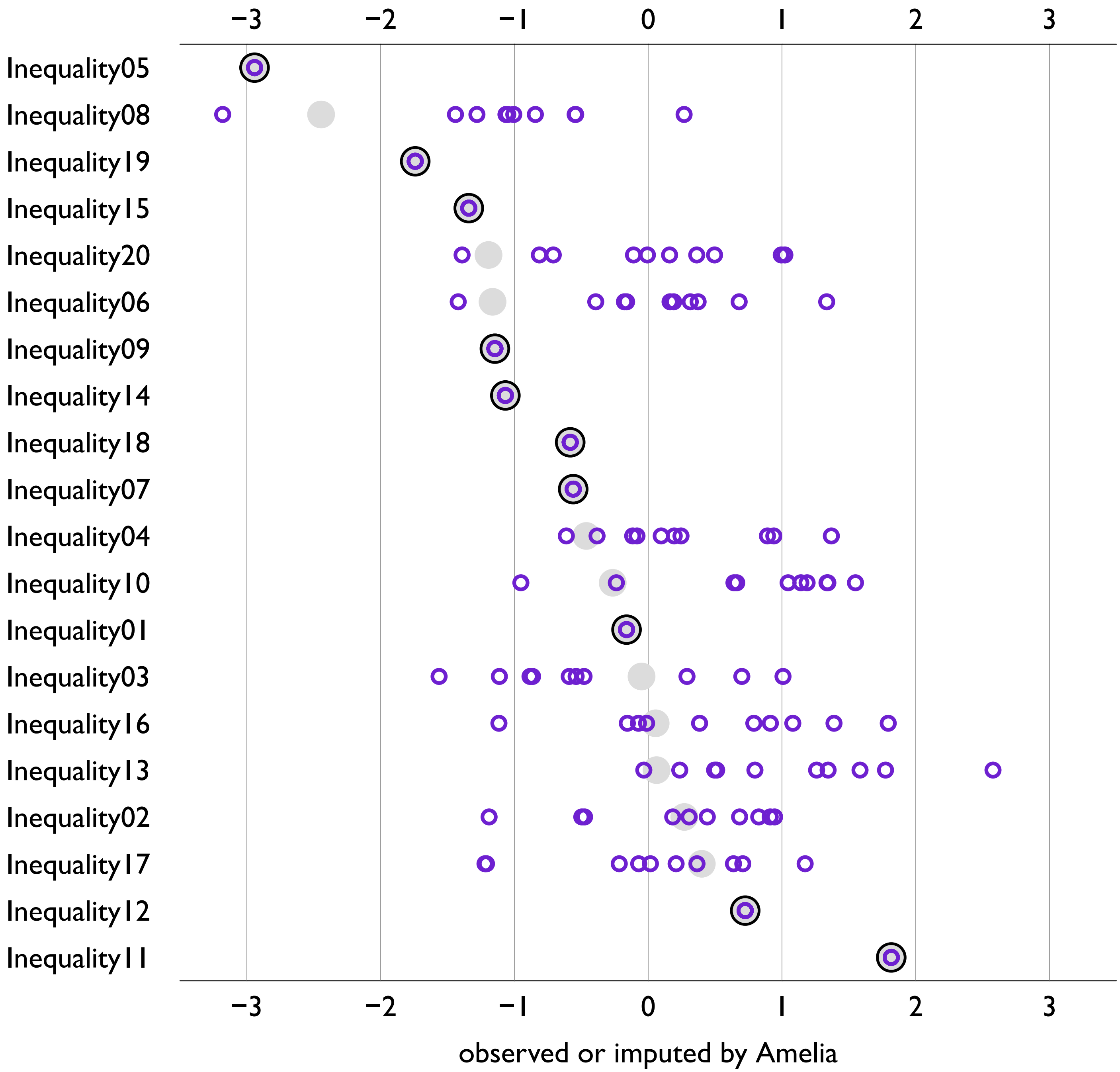
To get this "chicken-and-egg" process rolling, we supply starting values for $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$

Then we iterate back-and-forth until convergence
and never need to delete any rows with missing data

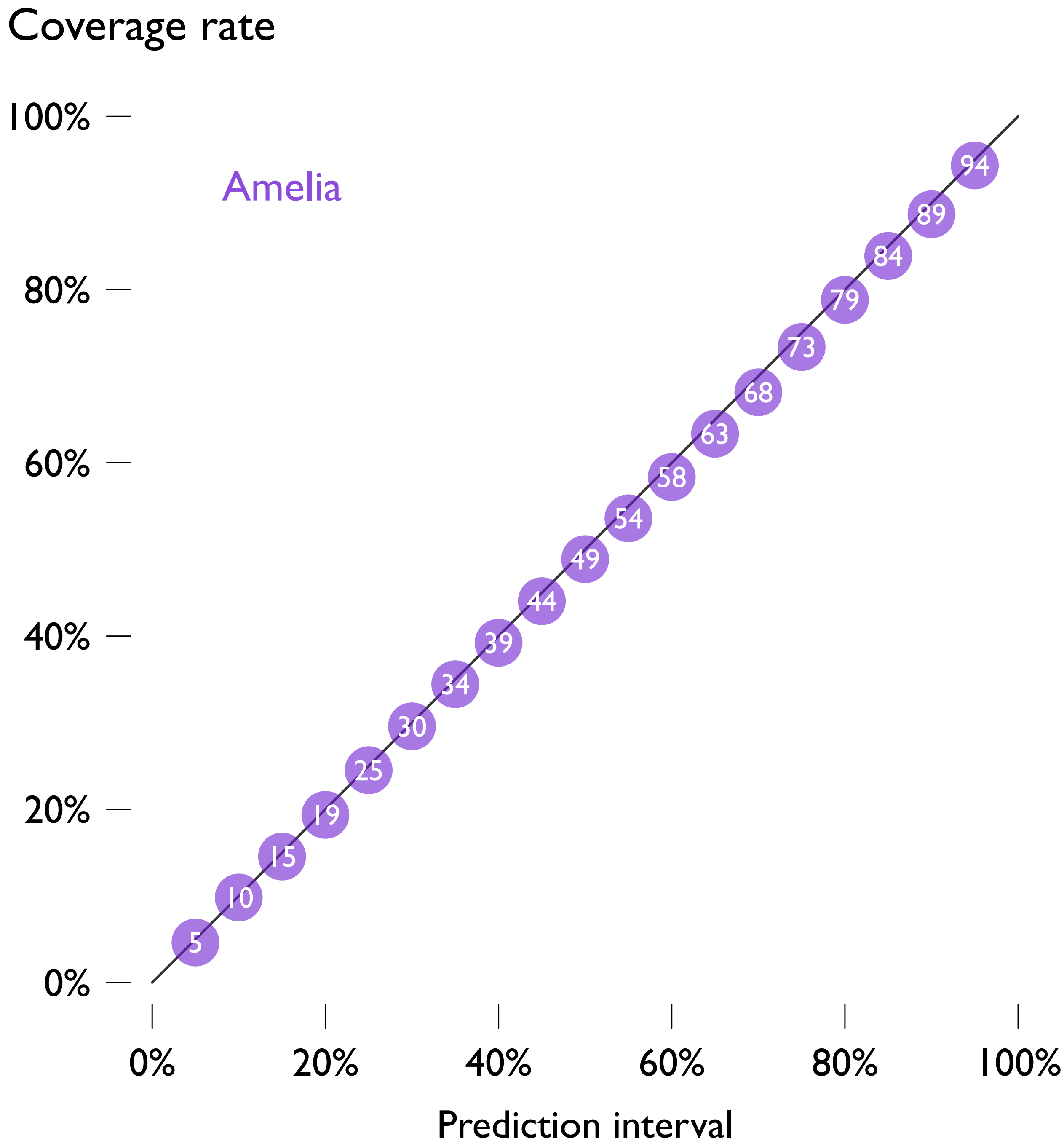
Naturally, there are a few extra pieces to the model
Bayesian priors, empirical priors, etc.



$\hat{\mu}$ and $\hat{\Sigma}$ allow us to compute posterior distributions over each missing datum



We summarize uncertainty with 5 (or 10, or more) draws from these posteriors



Across MC runs, Amelia's posteriors over missing values have correct *coverage*

Monte Carlo run 1, multiple imputation I

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	0.91	-0.21
[3]	0.97	-0.54	-0.66
[4]	0.17	0.10	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	-0.18	0.28
⋮	⋮	⋮	⋮

Imputed dataset 1

We end up with not one but five or more imputed datasets

Collectively, these datasets provide the central tendency *and* uncertainty of the missing cases

Monte Carlo run 1, multiple imputation 2

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	0.68	-0.21
[3]	0.97	-1.56	-0.66
[4]	0.17	0.89	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	-0.39	0.28
⋮	⋮	⋮	⋮

Imputed dataset 2

We need to run all our analyses in parallel on the five datasets, then combine the results using simulation

Monte Carlo run 1, multiple imputation 3

i	Democracy$_i$	Inequality$_i$	GDP$_i$
[1]	1.94	-0.16	1.28
[2]	0.26	0.44	-0.21
[3]	0.97	0.29	-0.66
[4]	0.17	-0.61	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	1.33	0.28
⋮	⋮	⋮	⋮

Imputed dataset 3

Specifically, take one-fifth of your simulated $\hat{\beta}$'s from each of your five analyses, then `rbind()` them together before computing counterfactual scenarios

Monte Carlo run 1, multiple imputation 4

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	0.94	-0.21
[3]	0.97	-0.88	-0.66
[4]	0.17	0.25	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	-0.16	0.28
⋮	⋮	⋮	⋮

Imputed dataset 4

`zelig()` in the `Zelig` package can automate this for you,

but it only works for certain statistical models

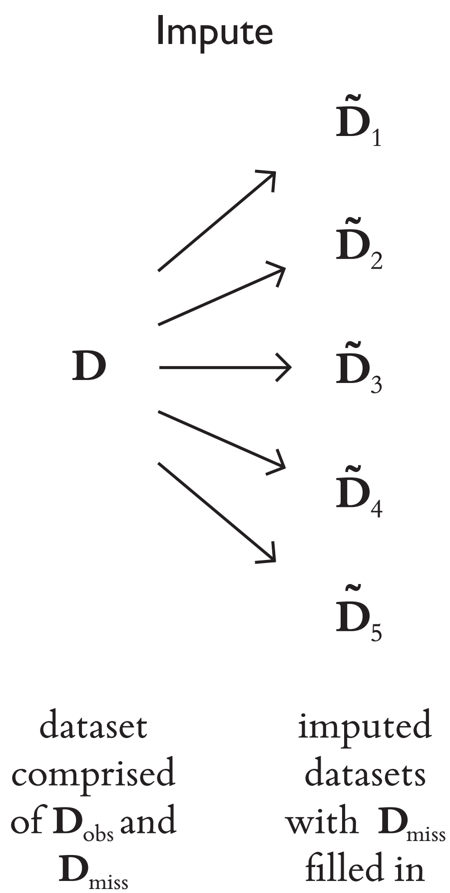
Monte Carlo run 1, multiple imputation 5

i	Democracy_i	Inequality_i	GDP_i
[1]	1.94	-0.16	1.28
[2]	0.26	0.18	-0.21
[3]	0.97	1.01	-0.66
[4]	0.17	0.94	-0.31
[5]	3.17	-2.94	0.96
[6]	-1.56	0.19	0.28
⋮	⋮	⋮	⋮

Imputed dataset 5

Instead, I recommend you write your own code, which is more flexible

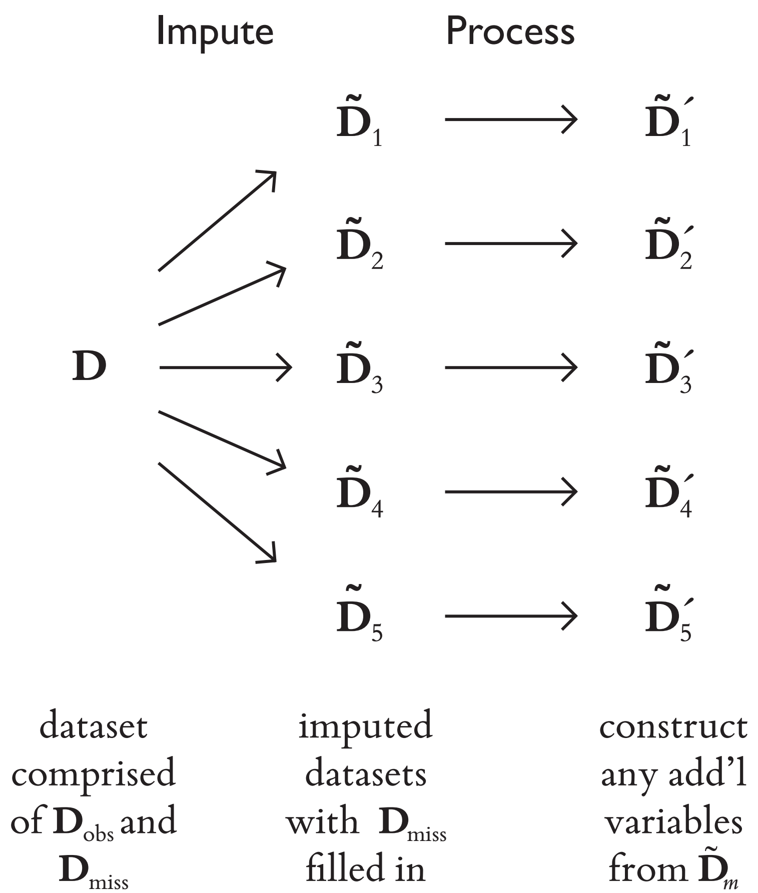
Here's the multiple imputation workflow. . .



Step 1: Perform multiple imputation to create $m = 5$ or more imputation datasets

(Very time consuming, especially if run multiple times under different assumptions)

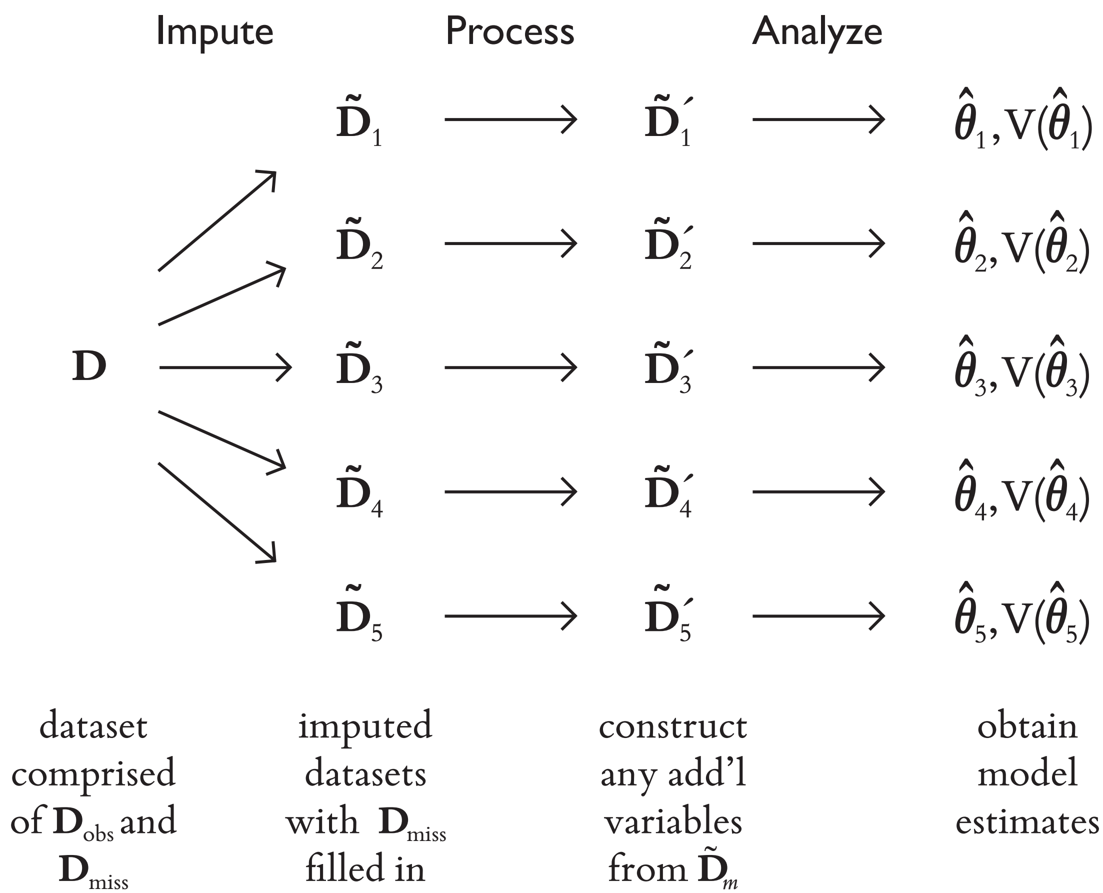
Imputing splits the analysis into M streams,
so it helps to loop over the imputed datasets for each subsequent step



Step 2: Construct additional variables from the imputed datasets

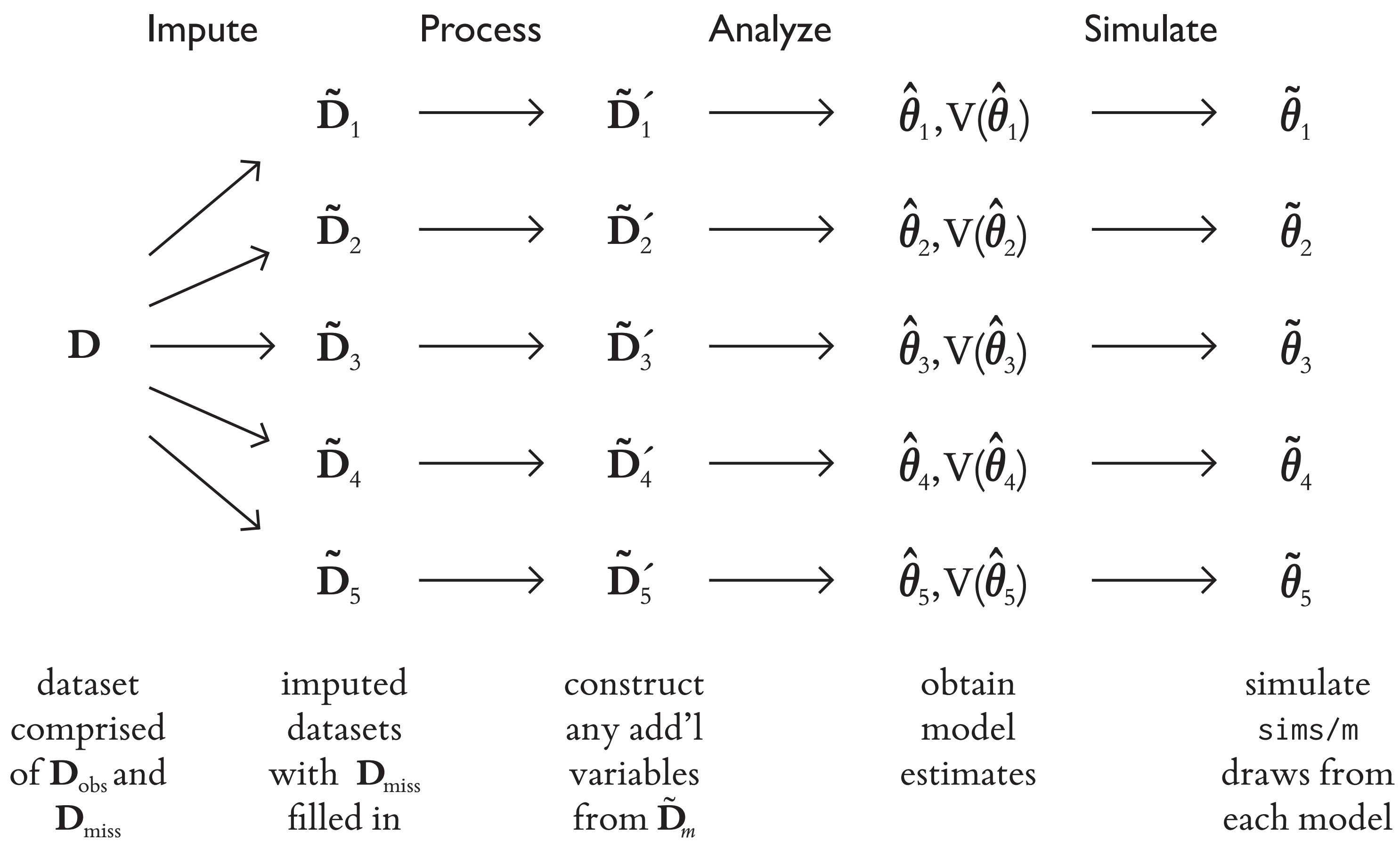
E.g., interaction terms, sums of components, or other products and sums

(e.g., if you impute GDP and popuation,
construct GDP per capita *after* all missings in either are imputed)



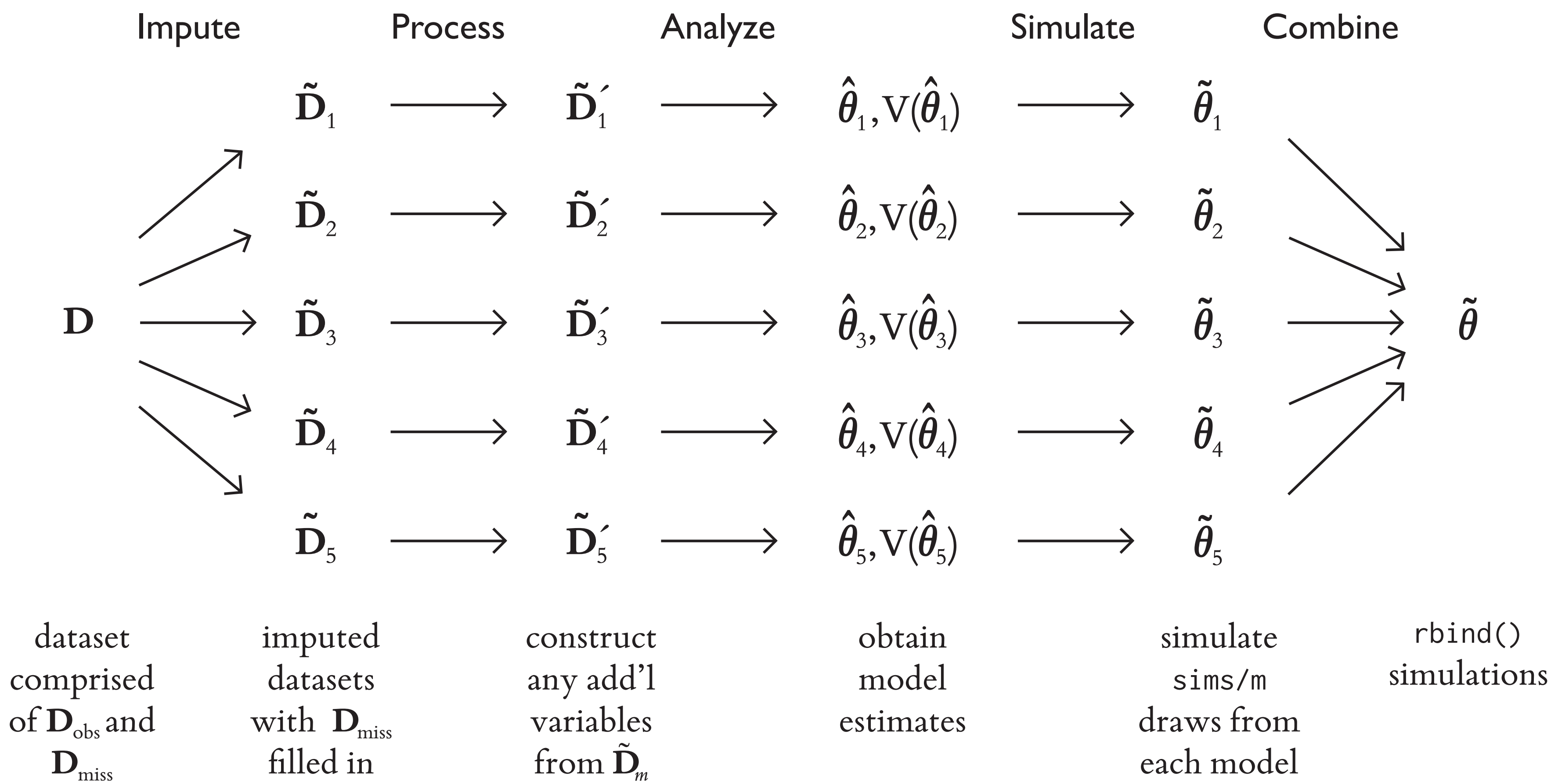
Step 3: Estimate the analysis model separately on each dataset m, and save each set of estimates $\boldsymbol{\theta}_m$ and variance-covariance matrix $\mathrm{V}(\hat{\boldsymbol{\theta}}_m)$

Each model should be the *same*, so use a loop or `lapply()`



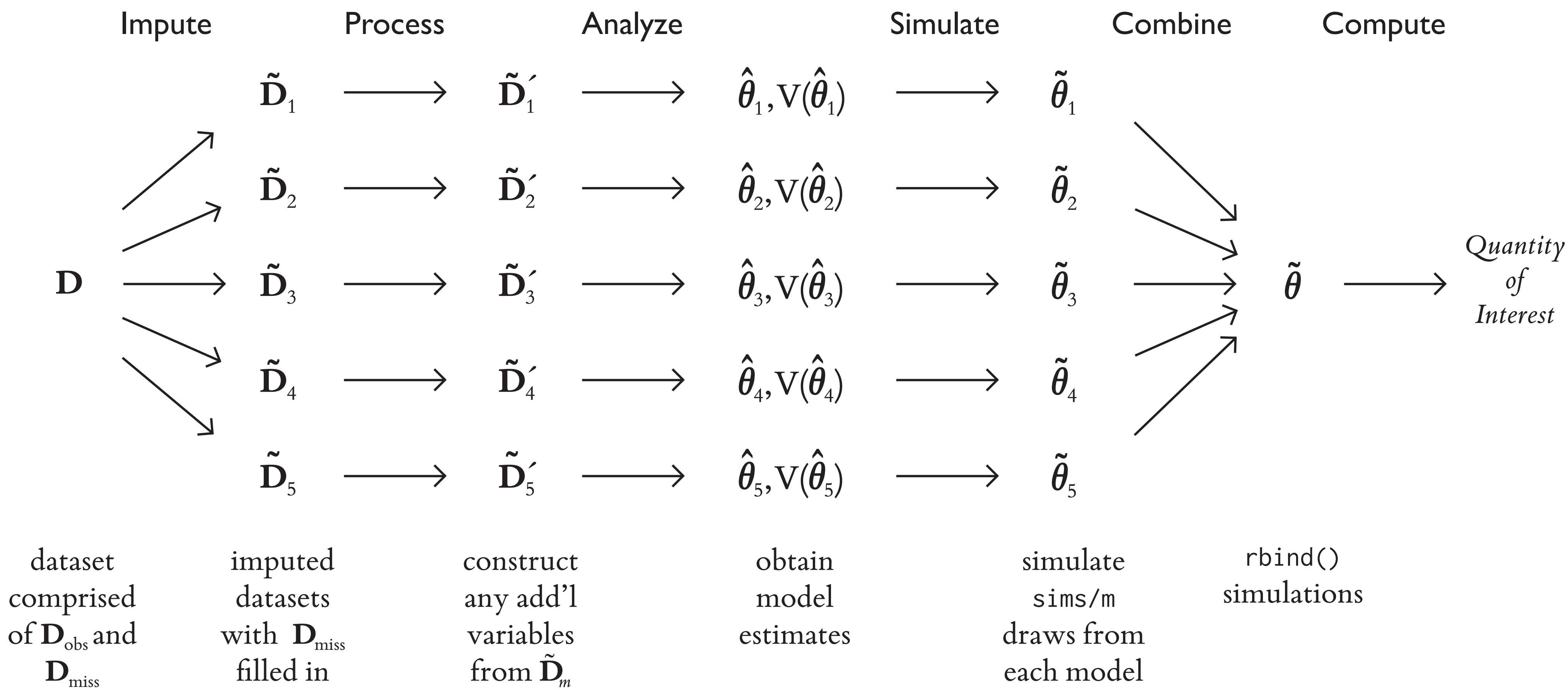
Step 4: Draw `sims/M` sets of simulated parameters from each of the M analyses

Use `mvrnorm()` as usual for this step, but in a loop over the M analysis runs



Step 5: Combine the M sets of simulated parameters into a single matrix using `rbind()`

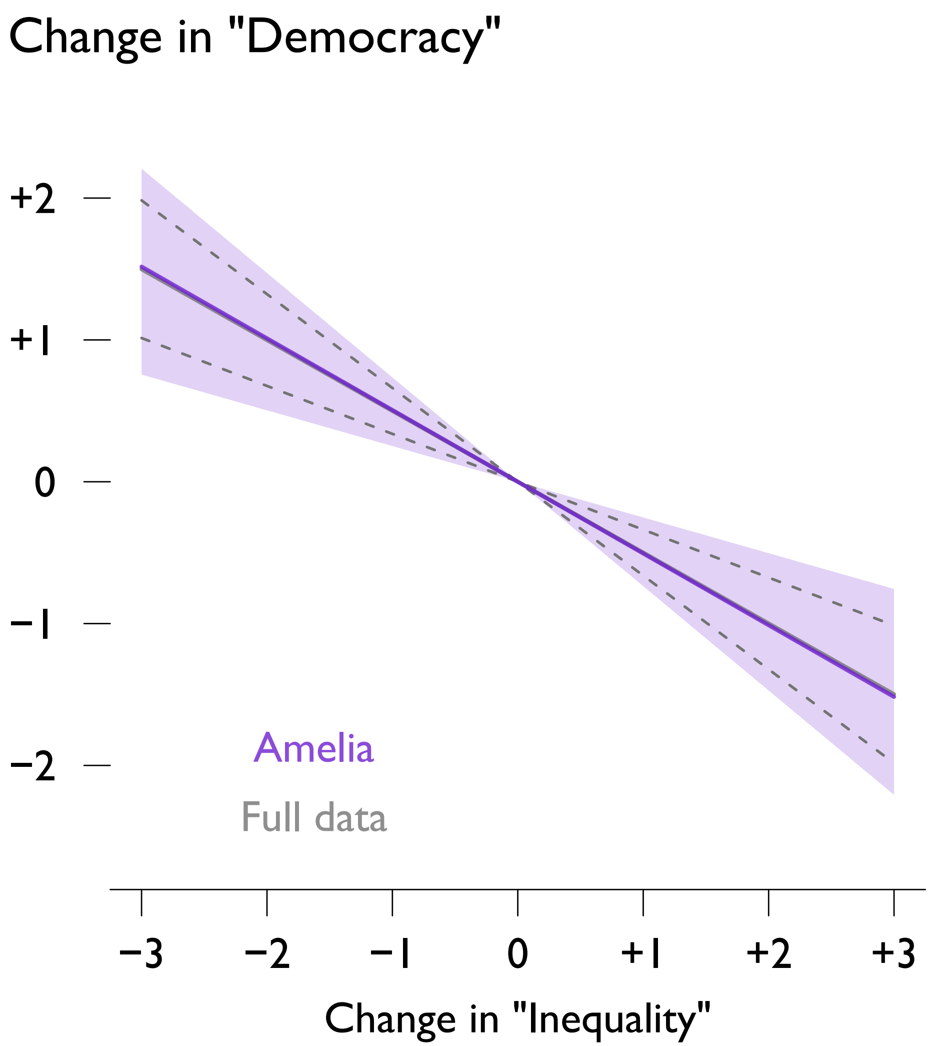
This brings the $M = 5$ streams of the analysis back together;
after this step, we only need to do things *once* for the whole analysis

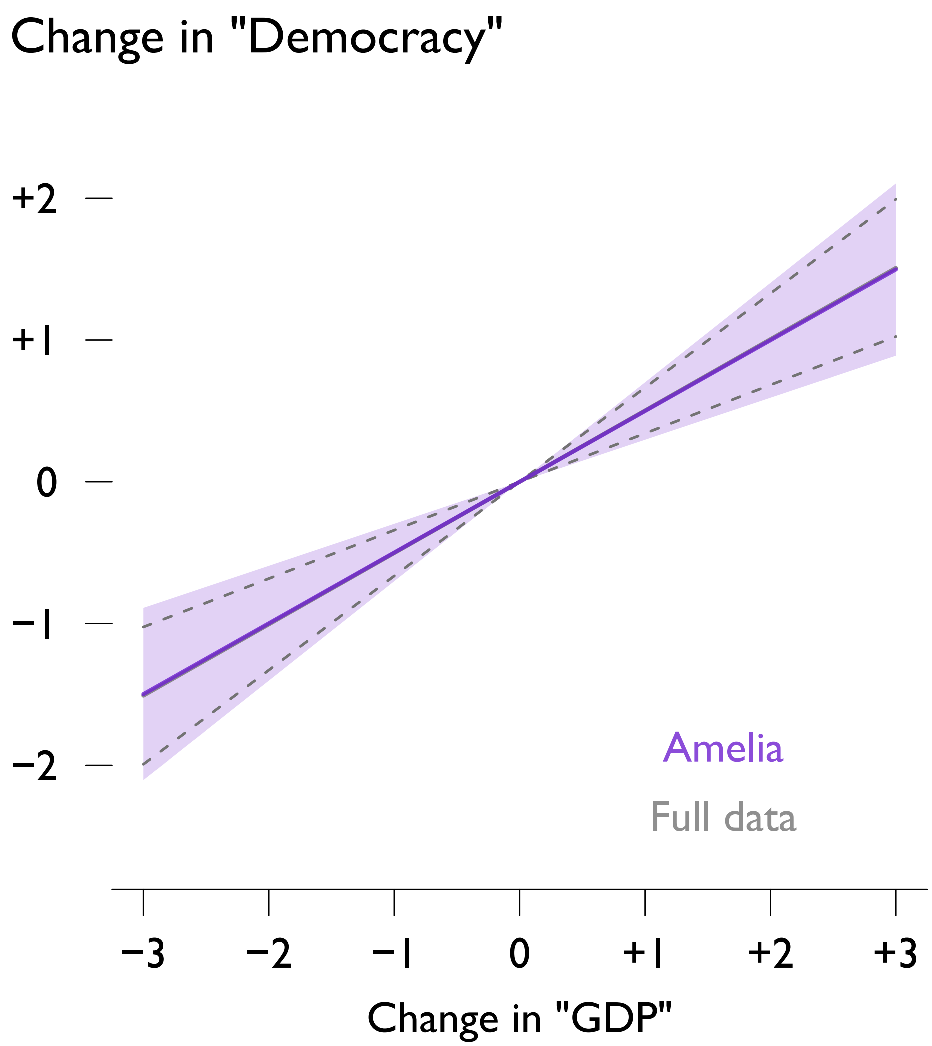


Step 6: Produce counterfactual scenarios and graphics as usual

The code for this step can be exactly the same as for a non-imputation analysis

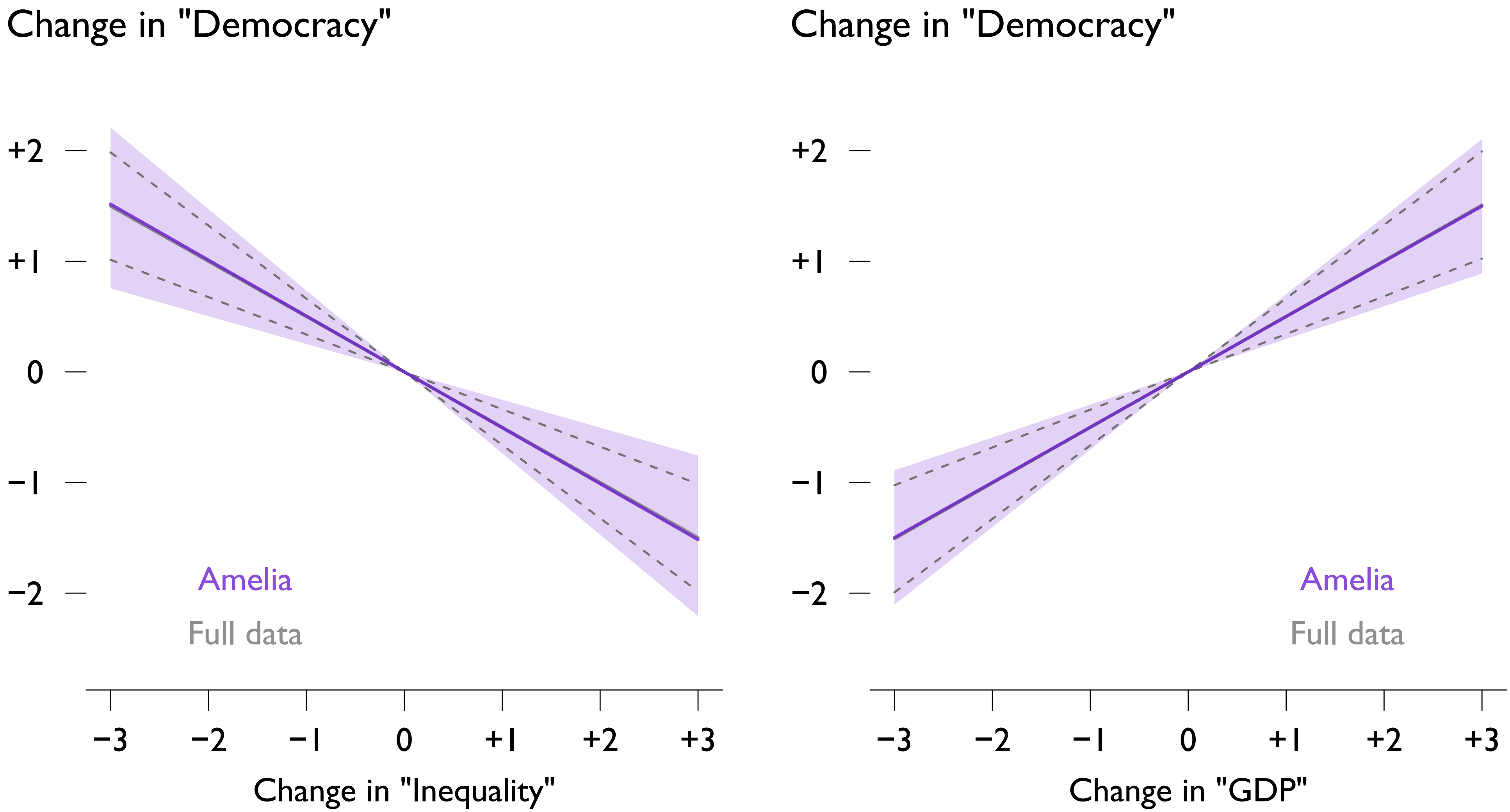
You may wish to average the $M = 5$ datasets at this stage for computing factual and counterfactual values of the covariates





Success! We have closely matched the original full data results

We've gotten more information & precision out of our data than with LWD, and not added any bias despite imputing



Will multiple imputation always work this well?

Should we ever listwise delete instead?

Outcome y is missing as a function of...

	Itself NI	*Covariate x* MAR	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased*			Inefficient	
MI	Biased				

Covariate x is missing as a function of...

	Outcome y MAR	*Itself* NI	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased	Inefficient†	Inefficient	Inefficient	Inefficient‡
MI		Biased			

Choose the row with your method for dealing with missing data: either listwise deletion or multiple imputation

Each column describes a potential mechanism by which missingness occurs

Your method has all the problems listed in the relevant cells

If you have all blank cells, your method is unbiased and efficient

Outcome y is missing as a function of...

	Itself	*Covariate x*	*Covariate z*	*Auxilliaries*	*None of these*
	NI	MAR	MAR	MAR	MCAR
LWD	Biased*			Inefficient	
MI	Biased				

Covariate x is missing as a function of...

	Outcome y	*Itself*	*Covariate z*	*Auxilliaries*	*None of these*
	MAR	NI	MAR	MAR	MCAR
LWD	Biased	Inefficient†	Inefficient	Inefficient	Inefficient‡
MI		Biased			

Non-ignorable (NI) missingness. After controlling for observables, whether a datum is missing depends on the missing datum. Unbiased imputation impossible

Missing at random (MAR). Pattern of missingness is related to observed values in dataset, and seemingly purely random once that pattern is controlled for

Missing completely at random (MCAR). Pattern of missingness is uncorrelated with all variables in the model, and thus seemingly purely random

Outcome y is missing as a function of...

	Itself NI	*Covariate x* MAR	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased*			Inefficient	
MI	Biased				

Covariate x is missing as a function of...

	Outcome y MAR	*Itself* NI	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased	Inefficient†	Inefficient	Inefficient	Inefficient‡
MI		Biased			

$*$ Logit unbiased in this case if missingness does not depend on covariates

† It's complicated: unbiased if missingness of x only depends on x (!) or other covariates; biased if also depends on y

‡ Assumes you have multiple covariates, ≥ 1 of which is observed when x is missing

Can you identify cases/assumptions where LWD is superior to MI?

Outcome y is missing as a function of...

	Itself NI	*Covariate x* MAR	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased*			Inefficient	
MI	Biased				

Covariate x is missing as a function of...

	Outcome y MAR	*Itself* NI	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased	Inefficient†	Inefficient	Inefficient	Inefficient‡
MI		Biased			

Most applications of LWD have efficiency costs: MI can produce more efficient results

If pattern of missinging in y depends on x, or vice versa, then LWD will be biased and MI can repair the bias – provided missingness can be predicted using observed data

If the pattern of missingness in y (or x) depends on the values of y (or x) that are missing, no method can eliminate bias, but careful use of MI may help sometimes

Outcome y is missing as a function of...

	Itself NI	*Covariate x* MAR	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased*			Inefficient	
MI	Biased				

Covariate x is missing as a function of...

	Outcome y MAR	*Itself* NI	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased	Inefficient†	Inefficient	Inefficient	Inefficient‡
MI		Biased			

Common misconception: *"you can't impute missing values of an outcome variable"*

1. No benefit to MI if only y has missings & no auxiliary variables present
2. Shouldn't impute if only y has missings in a logistic regression & no aux help
3. *Should* impute y as needed for imputation models of missing covariates, or any time helpful auxillary variables correlated with y are available

Outcome y is missing as a function of...

	Itself NI	*Covariate x* MAR	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased*			Inefficient	
MI	Biased				

Covariate x is missing as a function of...

	Outcome y MAR	*Itself* NI	*Covariate z* MAR	*Auxilliaries* MAR	*None of these* MCAR
LWD	Biased	Inefficient†	Inefficient	Inefficient	Inefficient‡
MI		Biased			

Finally, multiple imputation is not magical

1. MI can't help if all of your covariates and auxilliaries are missing for a case

2. May fail if you try to impute a dataset that has a very high percentage of missing values, or some variables which are almost never observed

You may need to give up on some variables in this case (exclude from your study)

Special considerations for effective use of Amelia

Key issue: managing assumption data are jointly Multivariate Normal

- transform continous variables to be as close to Normal as possible, e.g., through log, logit, or quadratic transformations
- tell your imputation model which variables are ordered or categorical – note King et al recommend treating *binary* variables as MVN
- check available diagnostics to make sure imputation worked

Special considerations for effective use of Amelia

Key issue: managing assumption data are jointly Multivariate Normal

- transform continous variables to be as close to Normal as possible, e.g., through log, logit, or quadratic transformations
- tell your imputation model which variables are ordered or categorical – note King et al recommend treating *binary* variables as MVN
- check available diagnostics to make sure imputation worked

Two additional best practices for all multiple imputation methods:

- include in the imputation as many well-observed variables related to your partially observed variables as you can find

  These auxilliary variables don't need to be in the analysis model later

- *every* variable in the analysis model *must* also be in the imputation model

Multiple Imputation beyond Amelia

Multiple imputation can be generic, like Amelia, or purpose-built

The latter is often superior, if you have theoretical insights into the nature of your missing data

But Amelia isn't the only generic imputation method

Two approaches to generic multiple imputation

Joint modeling Specifies a joint distribution of all data

Work well – and firmly grounded statistically – to the extent assumptions fit

Examples: Amelia and other fully Bayesian MI methods

Fully conditional specification Allow *ad hoc* models for each variable

Avoids blanket assumptions like Amelia's Multivariate Normal

Disadvantages: lacks clear statistical foundations;

can be slower than Amelia;

doesn't handle time series or time series cross-section as well

Examples: MICE (discussed here), mi, Hmisc

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\text{miss}}$ with starting values, such as the unconditional column means

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\mathrm{miss}}$ with starting values, such as the unconditional column means

Step 2. Cycle through the columns k of $\mathbf{X}$:

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\text{miss}}$ with starting values, such as the unconditional column means

Step 2. Cycle through the columns k of $\mathbf{X}$:

Step 2i. Reset filled-in missings in $\mathbf{x}_k$ to `NA`

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\text{miss}}$ with starting values, such as the unconditional column means

Step 2. Cycle through the columns k of $\mathbf{X}$:

Step 2i. Reset filled-in missings in $\mathbf{x}_k$ to `NA`

Step 2ii. Fit a regression of $\mathbf{x}_k^{\text{obs}}$ on (some subset of) $\mathbf{x}_{\neg k}$ using an appropriate MLE: e.g., MNL for categories; Quasipoisson for counts

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\text{miss}}$ with starting values, such as the unconditional column means

Step 2. Cycle through the columns k of $\mathbf{X}$:

Step 2i. Reset filled-in missings in $\mathbf{x}_k$ to `NA`

Step 2ii. Fit a regression of $\mathbf{x}_k^{\text{obs}}$ on (some subset of) $\mathbf{x}_{\neg k}$ using an appropriate MLE: e.g., MNL for categories; Quasipoisson for counts

Step 2iii. Draw predicted values from this model to fill in $\mathbf{x}_k^{\text{miss}}$

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\text{miss}}$ with starting values, such as the unconditional column means

Step 2. Cycle through the columns k of $\mathbf{X}$:

Step 2i. Reset filled-in missings in $\mathbf{x}_k$ to `NA`

Step 2ii. Fit a regression of $\mathbf{x}_k^{\text{obs}}$ on (some subset of) $\mathbf{x}_{\neg k}$ using an appropriate MLE: e.g., MNL for categories; Quasipoisson for counts

Step 2iii. Draw predicted values from this model to fill in $\mathbf{x}_k^{\text{miss}}$

Step 3. Repeat (2) p times (e.g., $p = 10$) to construct one imputed dataset

Multiple Imputation by Chained Equations (MICE)

The MICE algorithm

Step 1. Fill in $\mathbf{X}^{\text{miss}}$ with starting values, such as the unconditional column means

Step 2. Cycle through the columns k of $\mathbf{X}$:

- Step 2i. Reset filled-in missings in $\mathbf{x}_k$ to `NA`
- Step 2ii. Fit a regression of $\mathbf{x}_k^{\text{obs}}$ on (some subset of) $\mathbf{x}_{\neg k}$ using an appropriate MLE: e.g., MNL for categories; Quasipoisson for counts
- Step 2iii. Draw predicted values from this model to fill in $\mathbf{x}_k^{\text{miss}}$

Step 3. Repeat (2) p times (e.g., $p = 10$) to construct one imputed dataset

Step 4. Repeat (3) m times (e.g., $m = 10$) to construct m imputed datasets

MICE offers user flexibility in step 2ii: choosing appropriate MLEs for each variable

Variable type	Default MLE in MICE
Binary	Logistic regression
Ordered categories	Ordered logit
Unordered categories	Multinomial logit
Numeric	Predictive mean matching

MICE will try to guess the type of variable based on `R` data types

Specifically, it will only deviate from "predictive mean matching"
if the data is a `factor`

Because data types can be other than expected,
I strongly recommend setting the MLE for each column of data yourself

You can even provide MICE a custom MLE for a data column or variable type

Variable type	Default MLE in MICE
Binary	Logistic regression
Ordered categories	Ordered logit
Unordered categories	Multinomial logit
Numeric	Predictive mean matching

Predictive mean matching is a semiparameteric technique

Step 2ii. for PMM has four parts:

Step a. For column k, regress $\mathbf{x}_k^{\text{obs}}$ on the other columns in $\mathbf{X}$

Variable type	Default MLE in MICE
Binary	Logistic regression
Ordered categories	Ordered logit
Unordered categories	Multinomial logit
Numeric	Predictive mean matching

Predictive mean matching is a semiparameteric technique

Step 2ii. for PMM has four parts:

Step a. For column k, regress $\mathbf{x}_k^{\text{obs}}$ on the other columns in $\mathbf{X}$

Step b. Draw a set of parameters $\tilde{\gamma}$'s from this regression's predictive distribution

Variable type	Default MLE in MICE
Binary	Logistic regression
Ordered categories	Ordered logit
Unordered categories	Multinomial logit
Numeric	Predictive mean matching

Predictive mean matching is a semiparameteric technique

Step 2ii. for PMM has four parts:

Step a. For column k, regress $\mathbf{x}_k^{\text{obs}}$ on the other columns in $\mathbf{X}$

Step b. Draw a set of parameters $\tilde{\gamma}$'s from this regression's predictive distribution

Step c. Use $\tilde{\gamma}$ to compute predicted values $\tilde{\mathbf{x}}_k$ for each observed and missing $\mathbf{x}_k$

Variable type	Default MLE in MICE
Binary	Logistic regression
Ordered categories	Ordered logit
Unordered categories	Multinomial logit
Numeric	Predictive mean matching

Predictive mean matching is a semiparameteric technique

Step 2ii. for PMM has four parts:

Step a. For column k, regress $\mathbf{x}_k^{\mathrm{obs}}$ on the other columns in $\mathbf{X}$

Step b. Draw a set of parameters $\tilde{\gamma}$'s from this regression's predictive distribution

Step c. Use $\tilde{\gamma}$ to compute predicted values $\tilde{\mathbf{x}}_k$ for each observed and missing $\mathbf{x}_k$

Step d. For each $\tilde{\mathbf{x}}_k^{\mathrm{miss}}$, sample observed cases with similar predicted values

Then use a corresponding $\mathbf{x}_k^{\mathrm{obs}}$ (selected randomly) as the new imputation of $\mathbf{x}_k^{\mathrm{miss}}$

Variable type	Default MLE in MICE
Binary	Logistic regression
Ordered categories	Ordered logit
Unordered categories	Multinomial logit
Numeric	Predictive mean matching

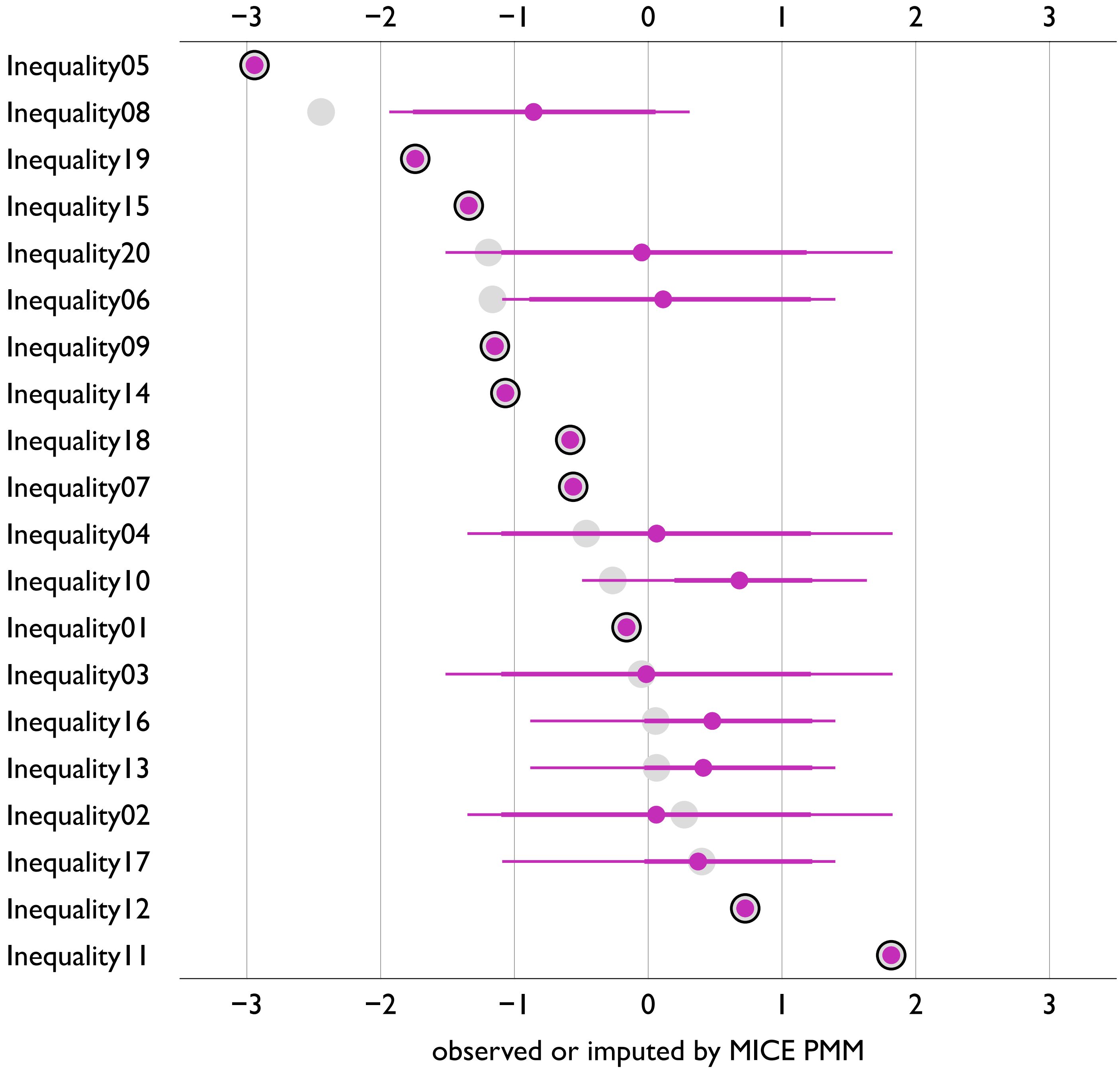
Is predictive mean matching superior to assuming a Normal distribution?

Virtues: Produces predicted values that look like the distribution of $\mathbf{x}_k^{\text{obs}}$

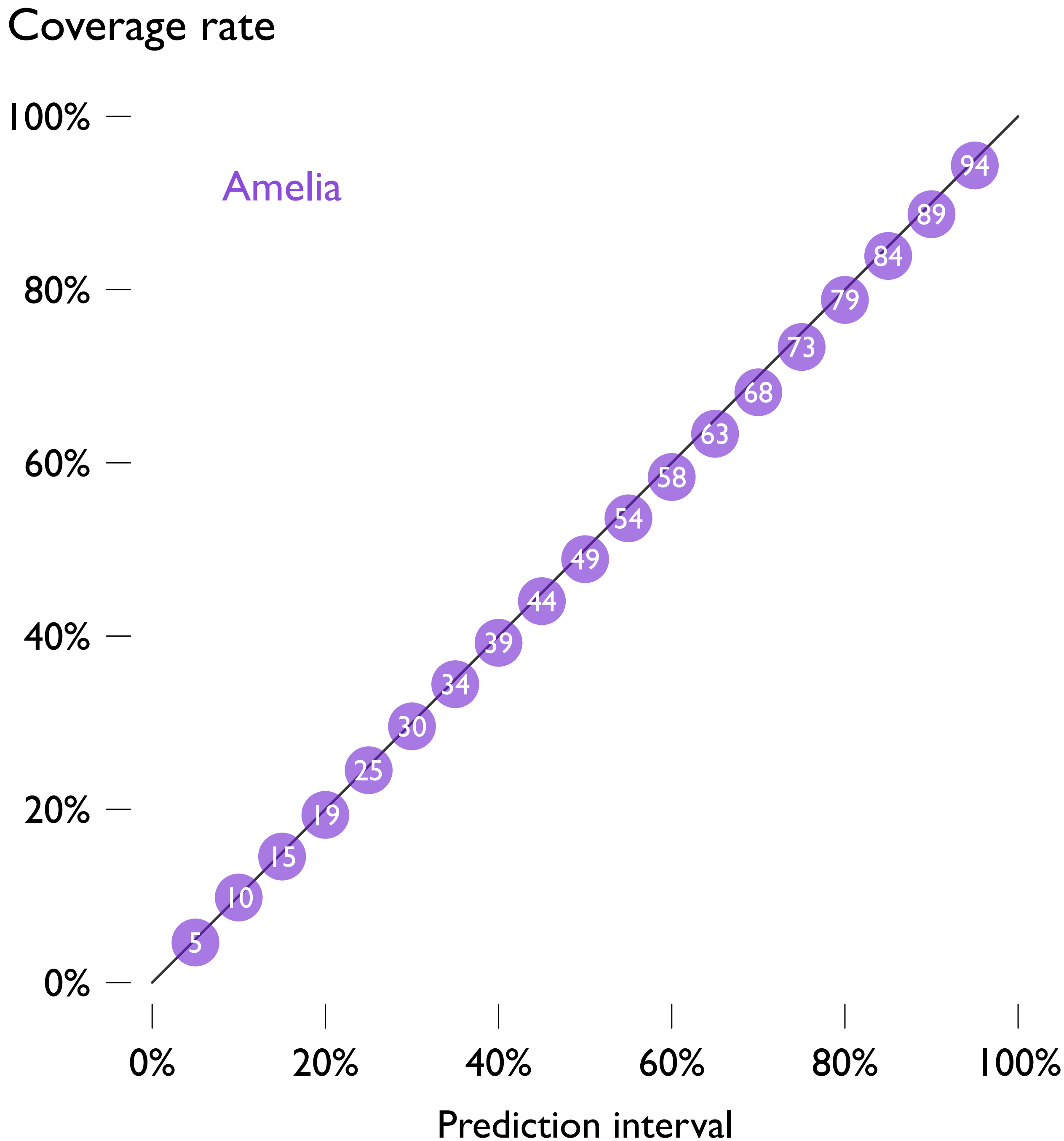
More robust to misspecification, heteroskedasticity, deviations from simple transformations (or from linearity, if none are provided)

Downsides: Statistical properties of this procedure unknown (unknowable?); may be overconfident when imputing missing values far from mean

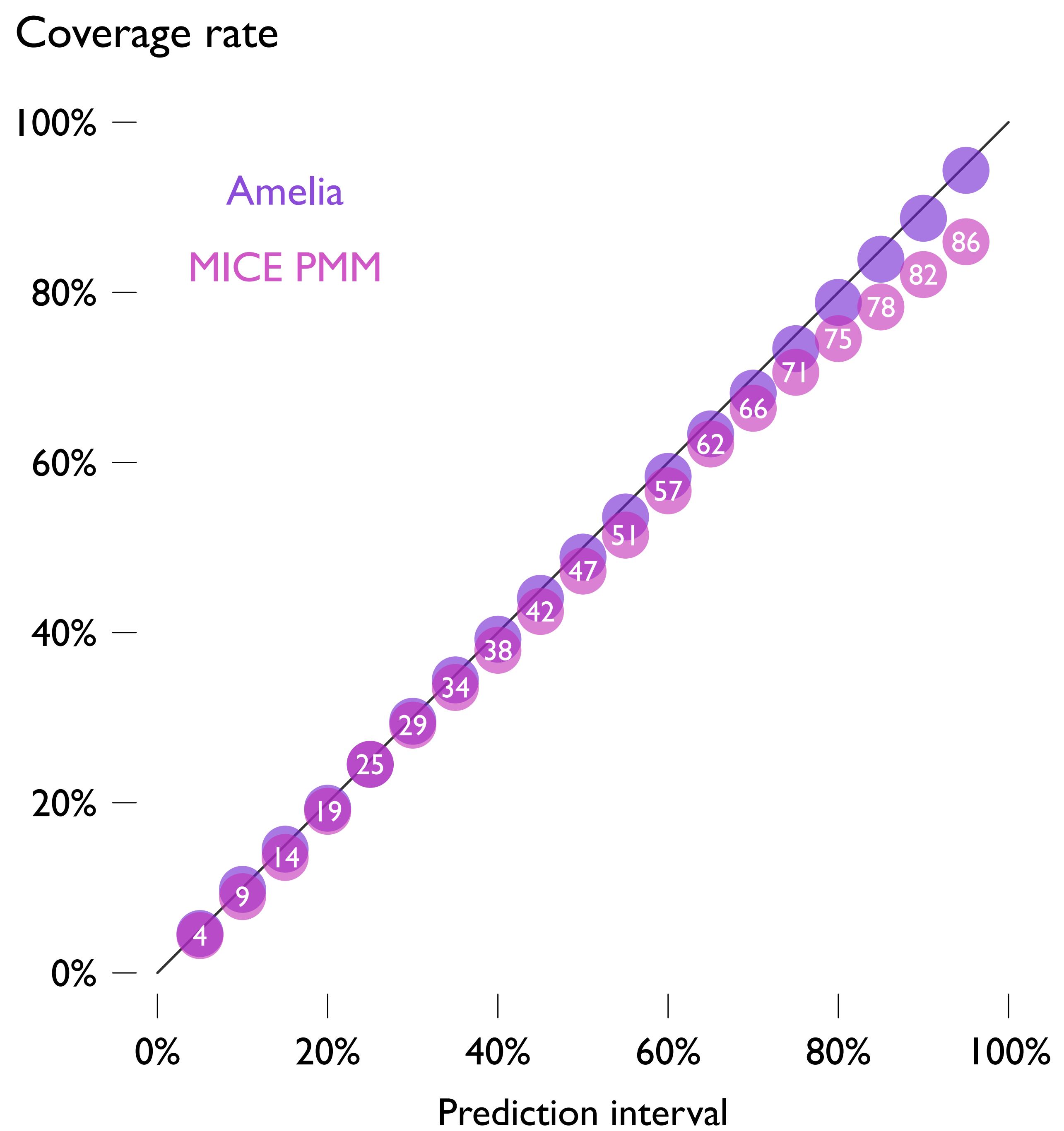
What does MICE with predictive mean matching make of our data?



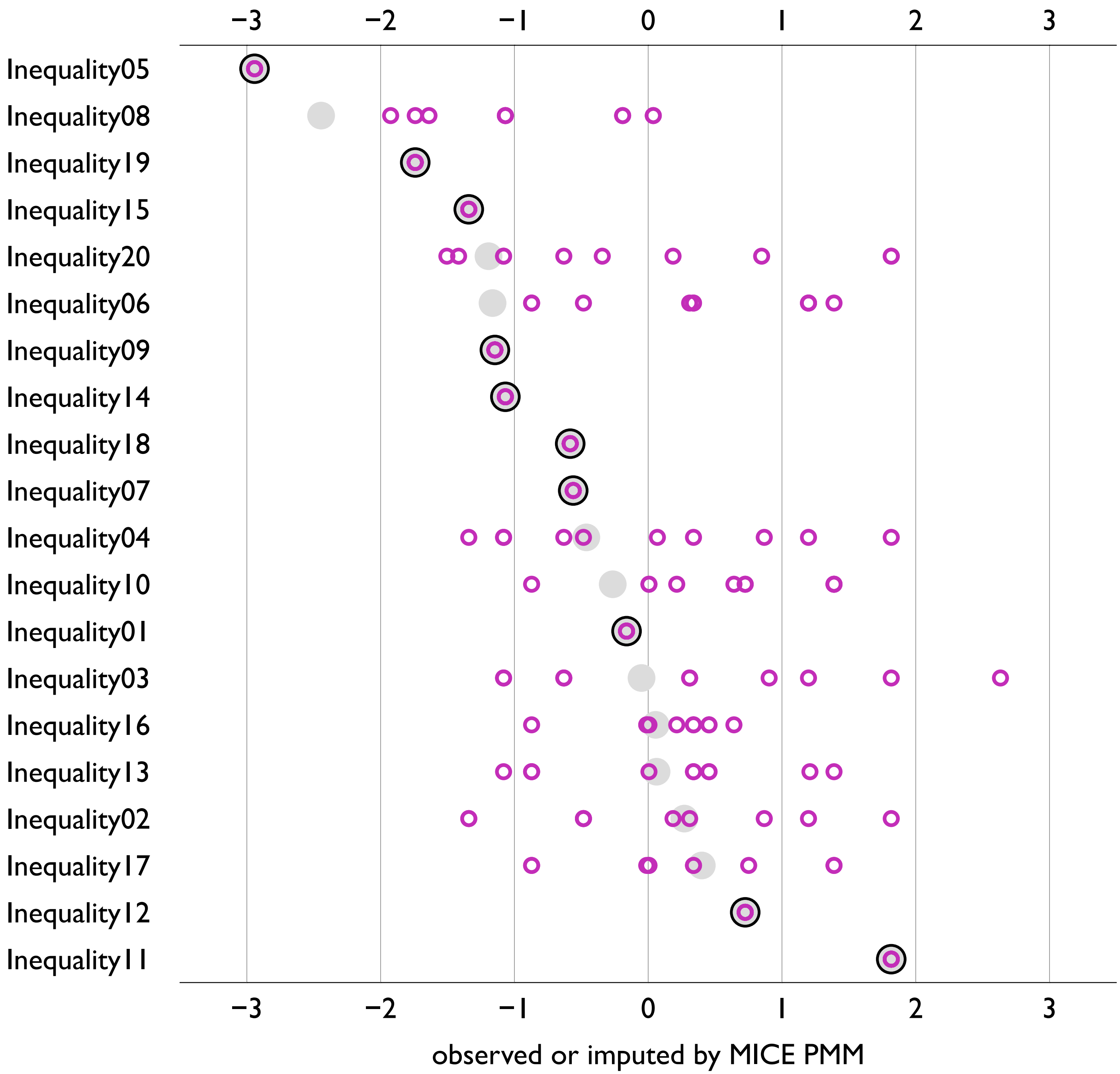
Compared to Amelia, PMM produces similar but smaller dist's of each missing datum



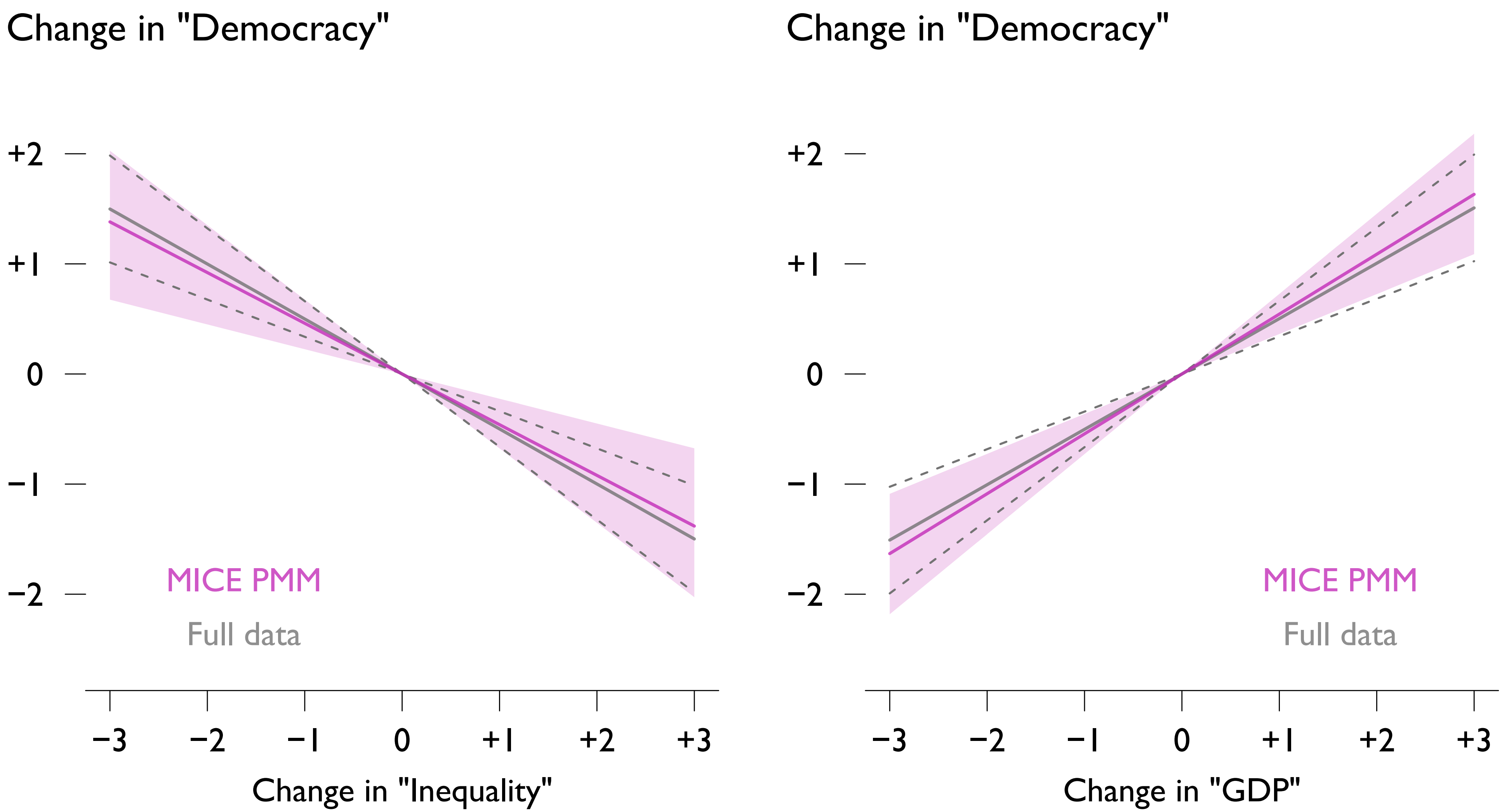
Recall that Amelia's imputations were drawn from intervals with appropriate coverage



The MICE PPM prediction intervals are too narrow, especially at high certainty



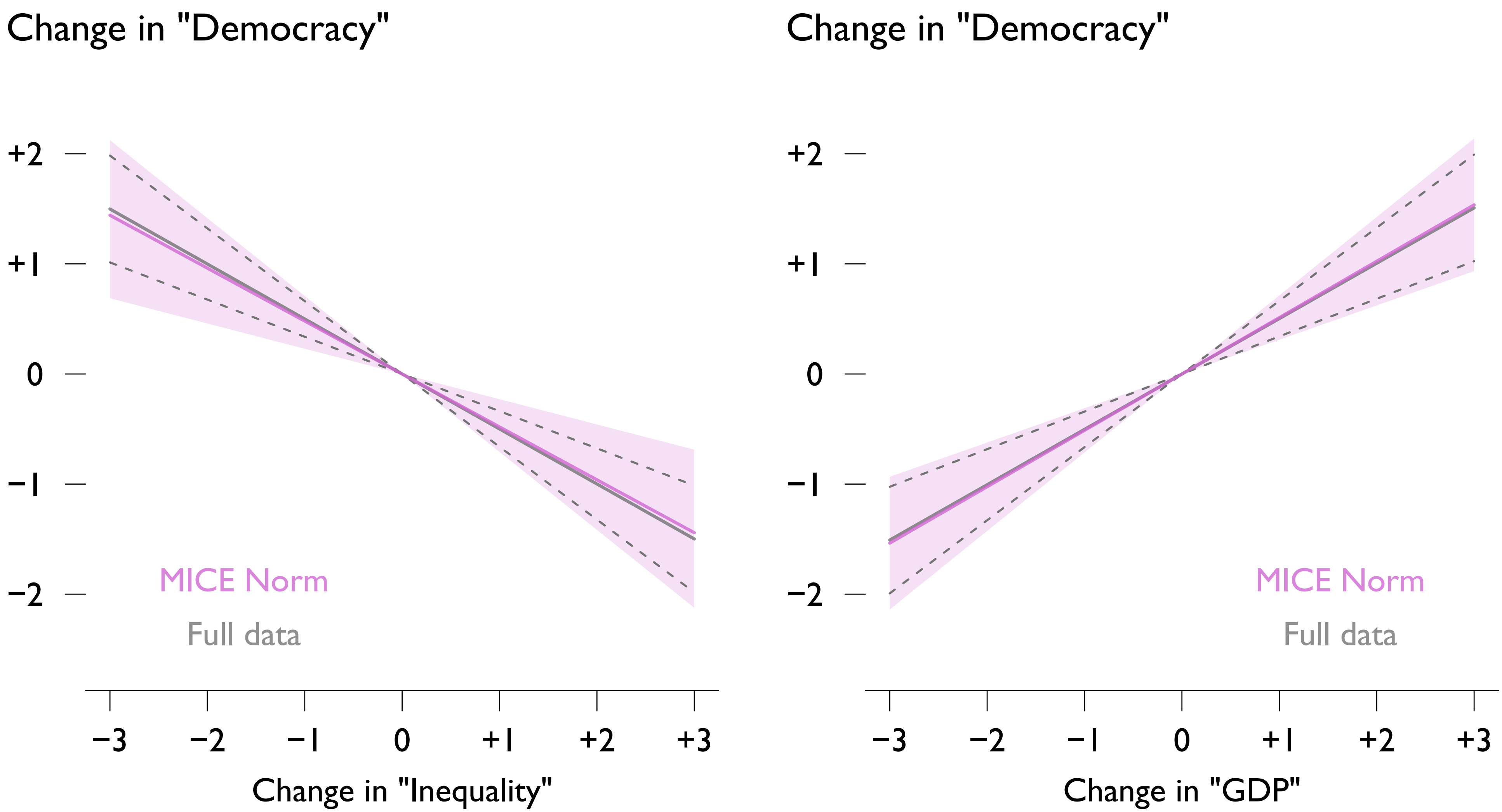
This leads to slightly more confident – or concentrated – imputations than Amelia



The extra confidence appears misplaced – MICE PMM is biased in our case

Why? PMM relies on the existence of close matches in the observed data

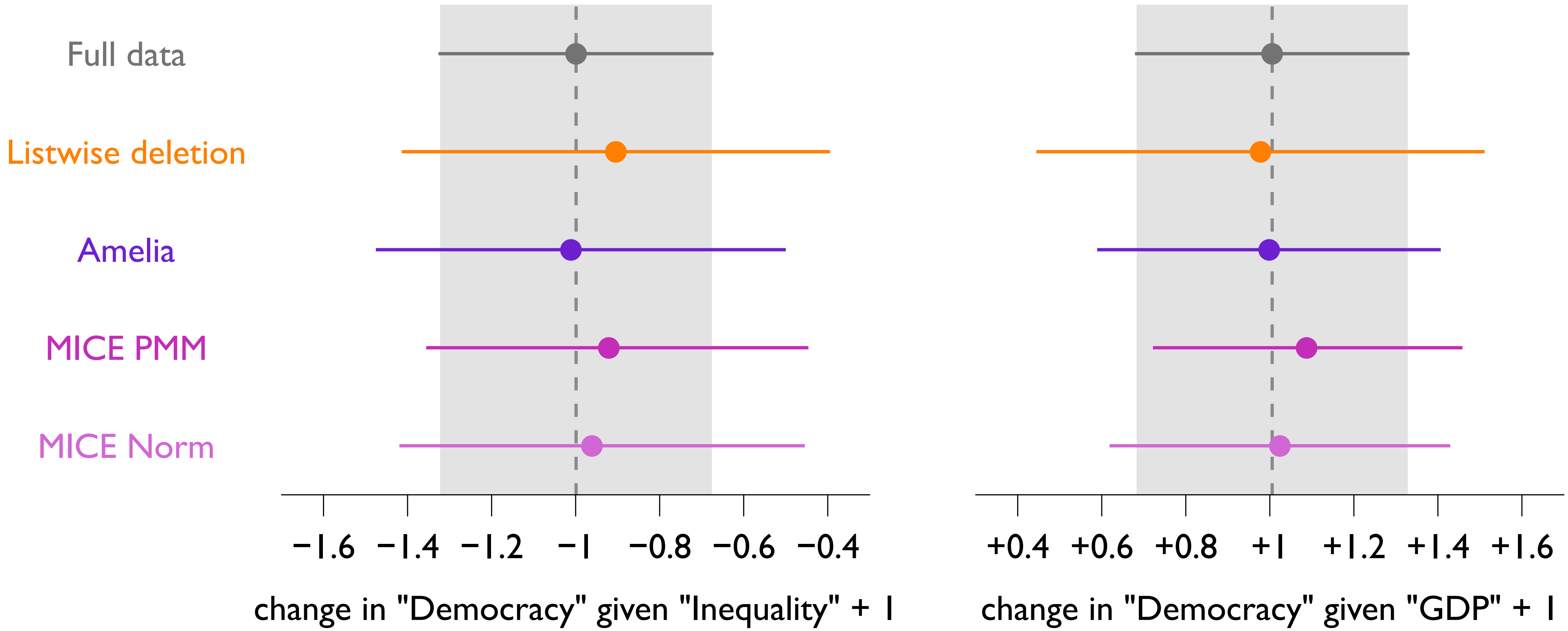
Here, extremely high values of inequality are scarce



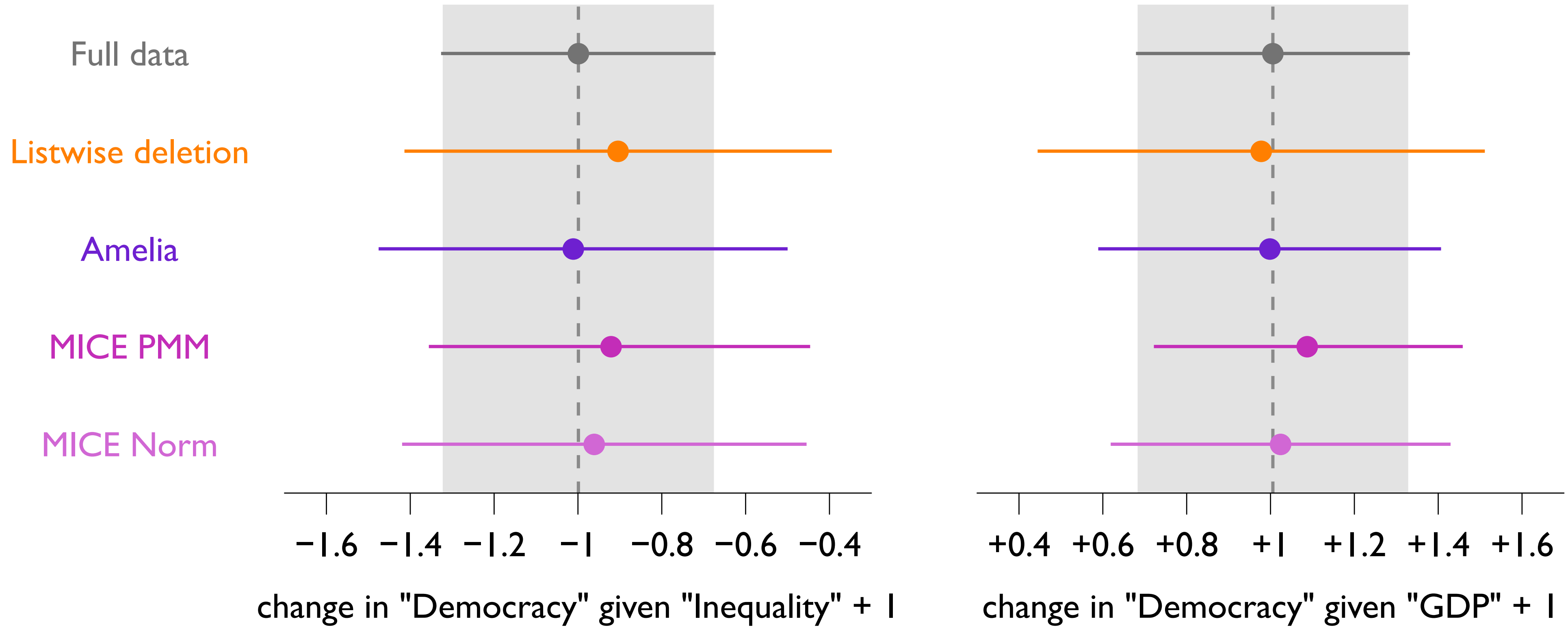
What if we use MICE, but again assume Inequality is Normally distributed?

Using the correct model reduces the bias –

though in real data analysis, we don't usually know the correct model



We have four options for coping with missing data: how do they stack up?

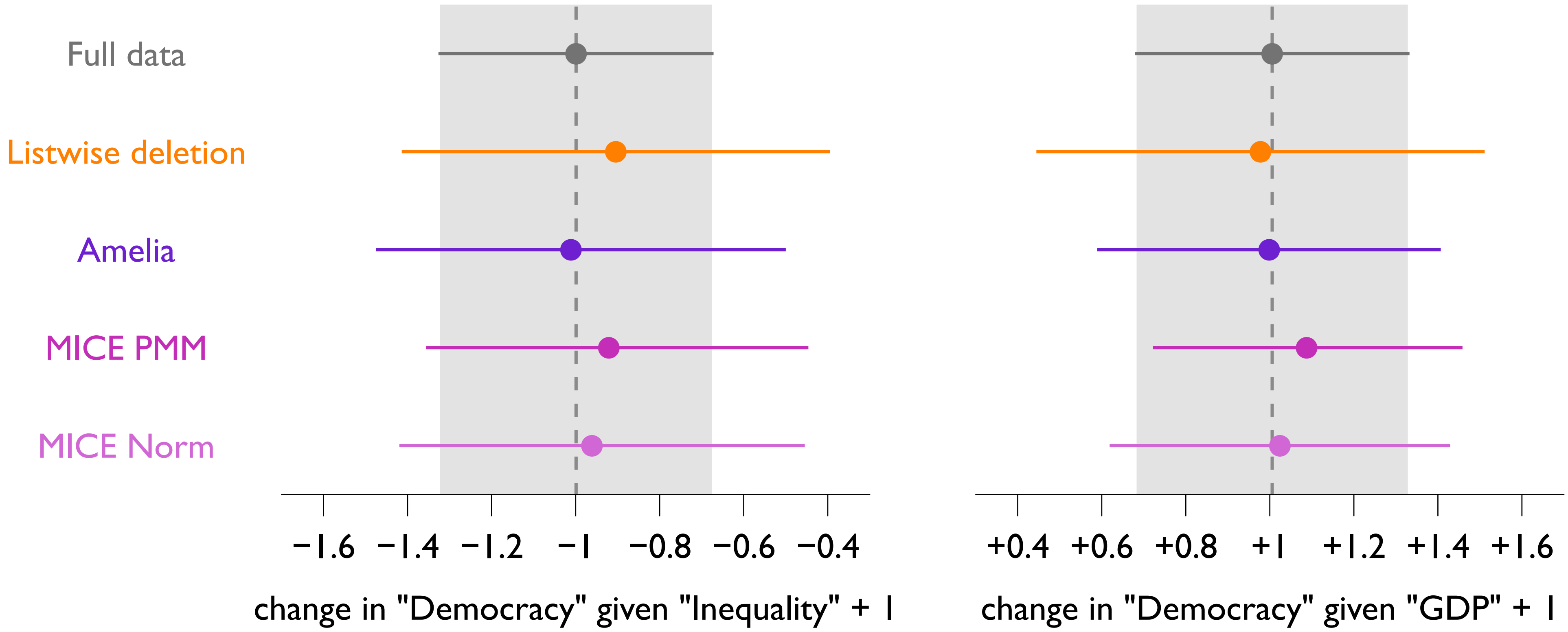


We have four options for coping with missing data: how do they stack up?

All three imputation techniques improve on listwise deletion, especially for estimating coefficients of variables *less* often missing

In data that are truly multivariate normal, Amelia outperforms MICE

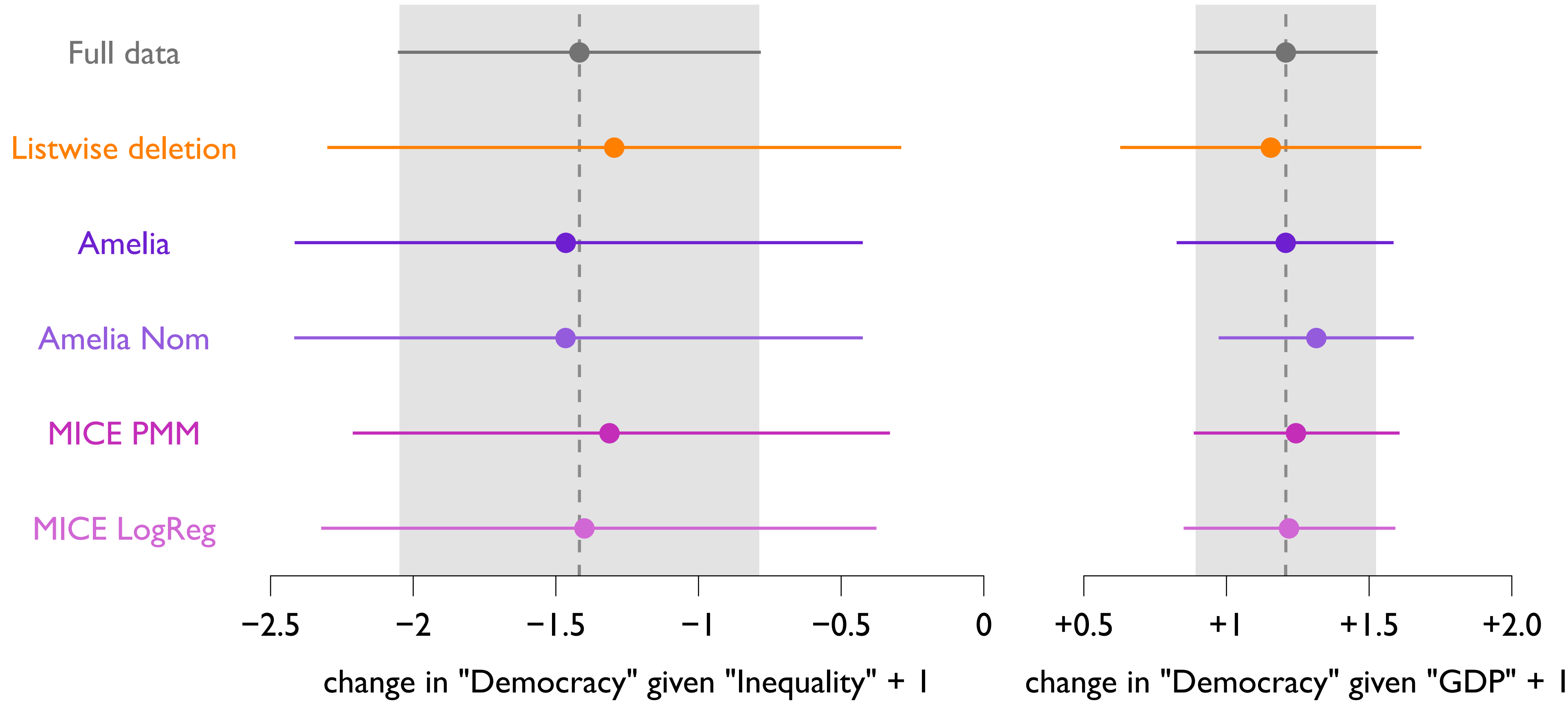
PMM does relatively poorly – but perhaps this was an unfair test?



MICE is often recommended for datasets with binary or categorical data

Let's dichotomize Inequality into "high" and "low",
treating the current x as a latent variable with a cutpoint at 0

The pattern of missingingness stays the same



We now consider four imputation schemes:
(1) Amelia, (2) Amelia for nominal variables,
(3) MICE PMM, (4) MICE with logistic regression

Amelia and MICE logreg have similar good performance

MICE PMM and Amelia for nominal variables fare worse –
note Amelia's authors recommend treating *binary* variables as MVN

Which MI method to use with real data?

Perhaps this latest Monte Carlo experiment still stacks the deck in favor of Amelia

The data were originally Multivariate Normal before Inequality was dichotomized

A fairer test of Amelia vs MICE would be a real-world dataset with an unknown DGP

. . . Such as *your* dataset

But then how could we know which method worked better?

Which MI method to use with real data?

Perhaps this latest Monte Carlo experiment still stacks the deck in favor of Amelia

The data were originally Multivariate Normal before Inequality was dichotomized

A fairer test of Amelia vs MICE would be a real-world dataset with an unknown DGP

. . . Such as *your* dataset

But then how could we know which method worked better?

Overimputation!

1. Propose a missingness model for your data
2. Delete some of the observed data using this model
3. See whether Amelia or MICE recovers the deleted data better

Application: 2004 Washington Governor's Race

Recall, again, our binomial distribution example:
the number of voters who turned out in each of the 39 Washington counties in 2004

Our outcome variable

voters – the count of registered voters who turned out

non-voters – the count of registered voters who stayed home

Our covariates

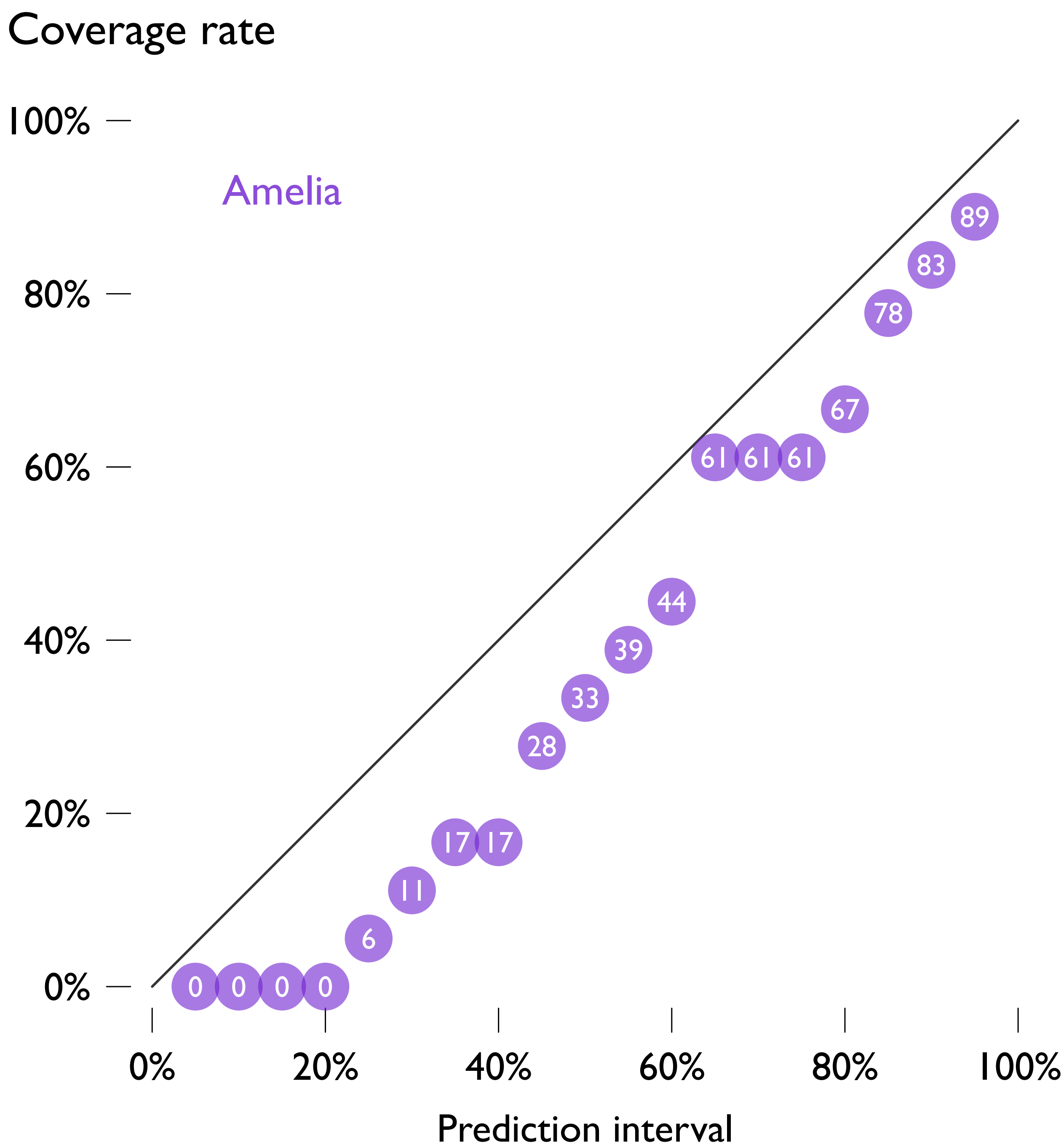
income – the median household income in the county in 2004

college – the % of residents over 25 with at least a college degree in 2005

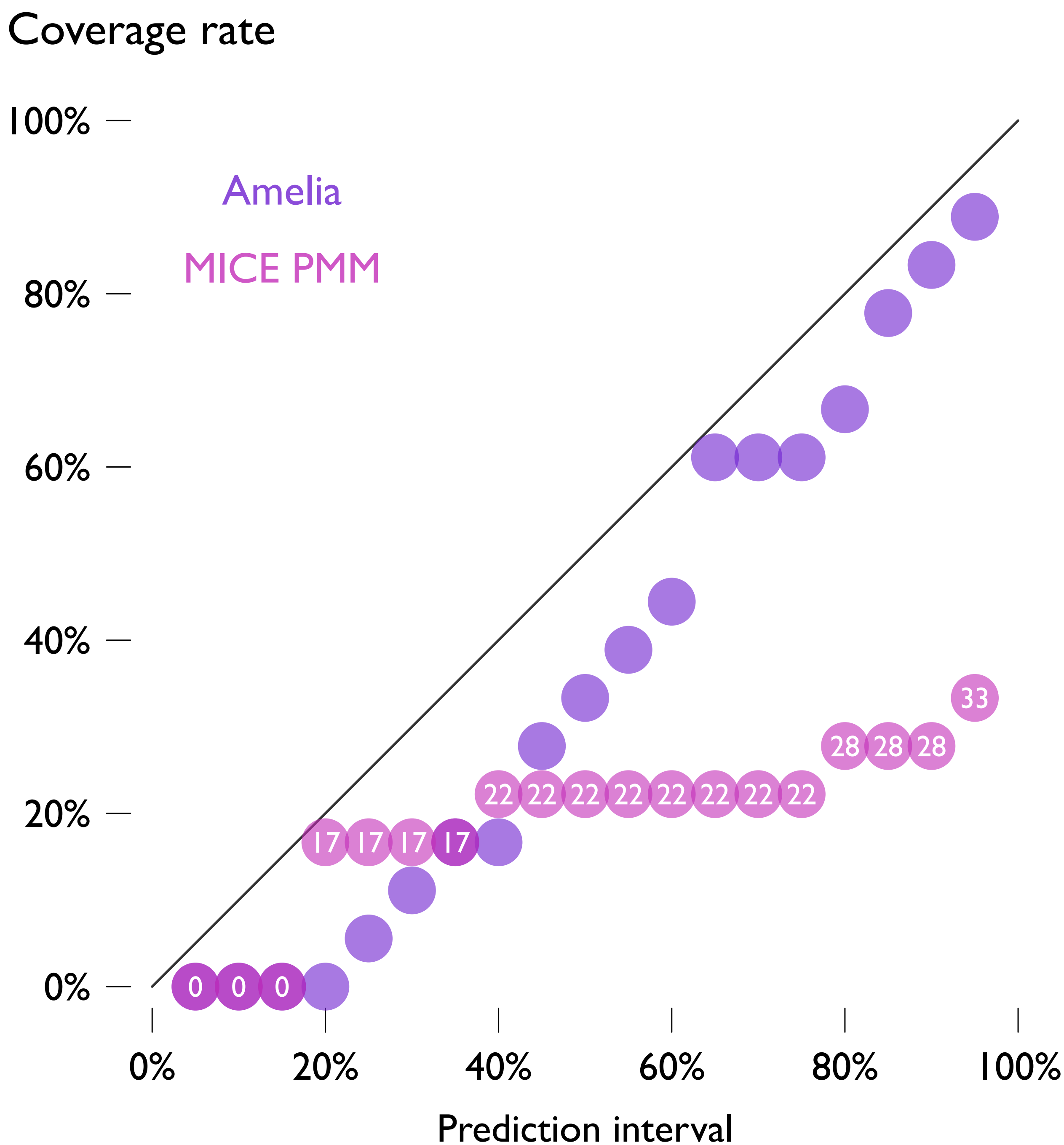
College is only available for the 18 largest counties; the rest are fully observed

I use multiple imputation by Amelia to fill in the missings

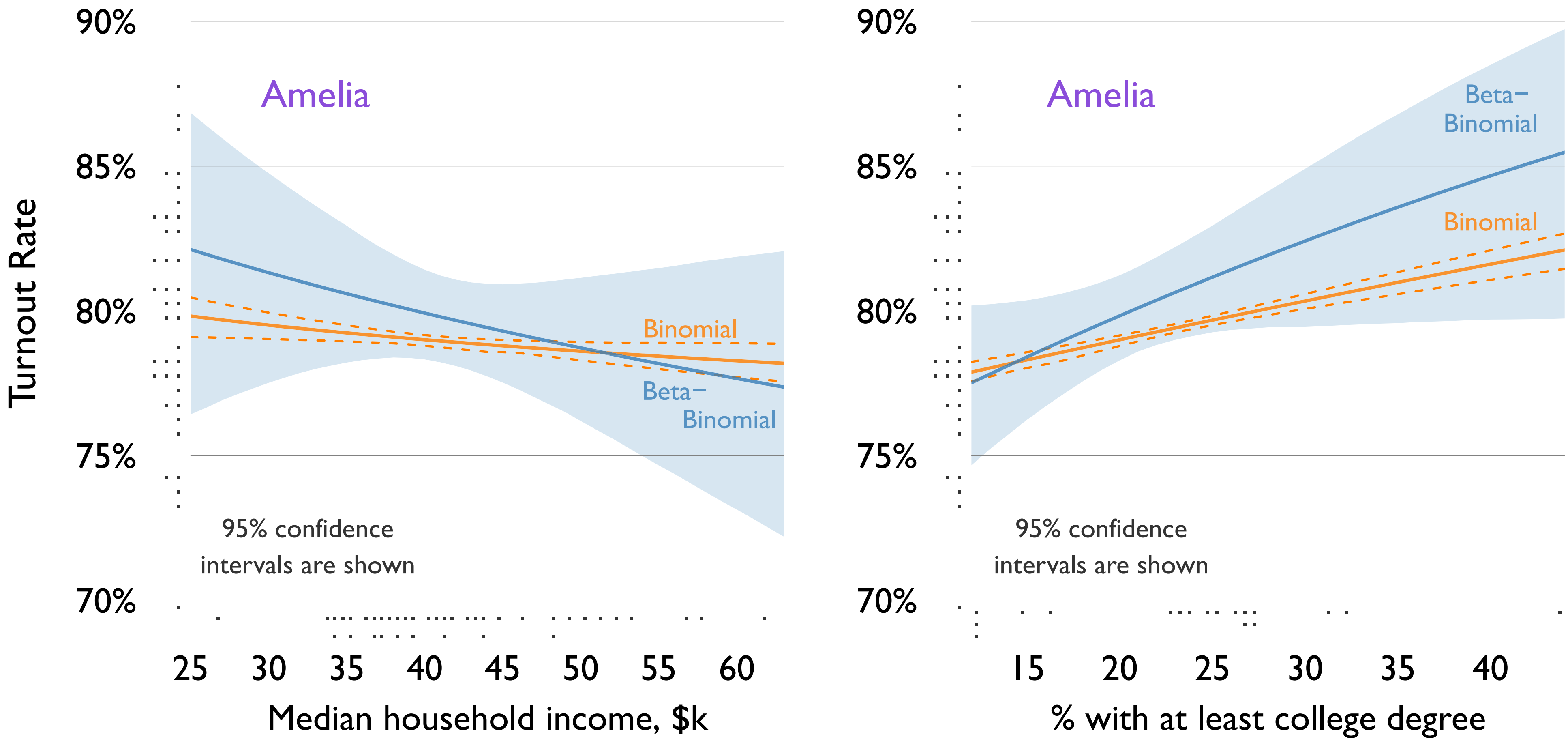
Would it have mattered if I used MICE instead?



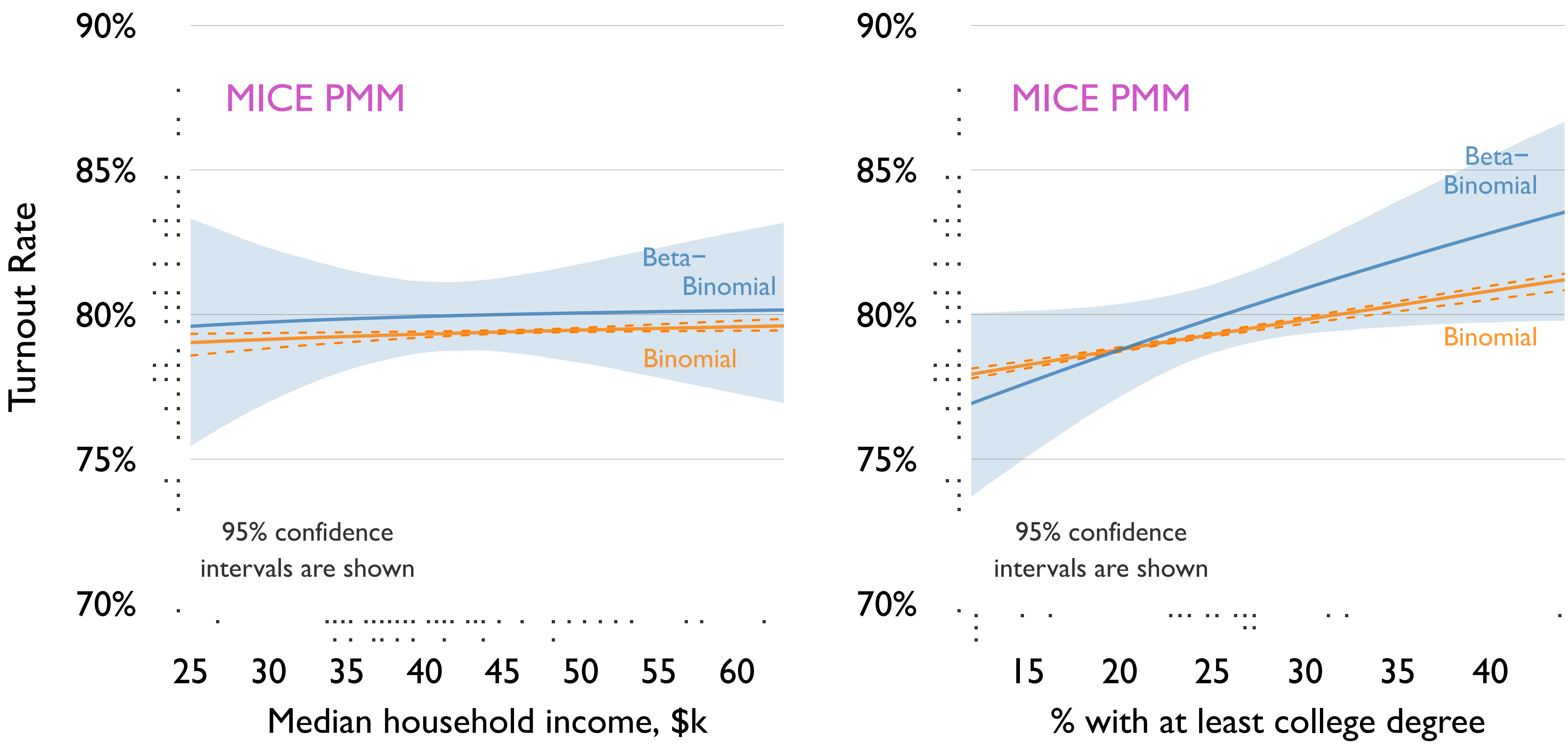
Amelia is a bit overconfident – pretty good given only 18 datapoints!



MICE PMM is worryingly overconfident *How much substantive difference?*



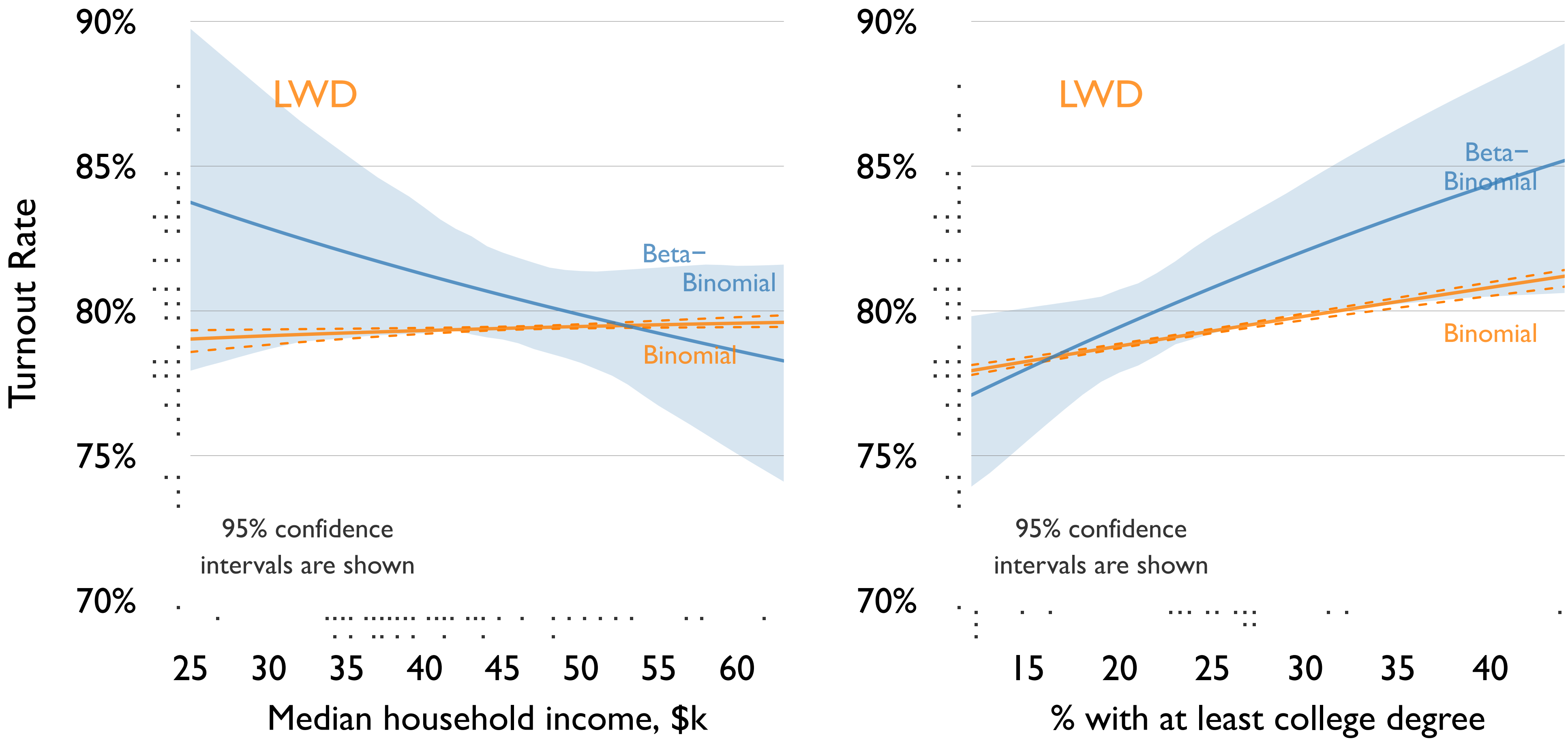
Above are the Amelia-based results from several weeks ago. . .



and these are the MICE PMM results for the same models

Substantively different, but you might use same words to describe them

"Significant, large positive effects of college, especially in the Beta-Binomial; insignificant effects of income controlling for college in the Beta-Binomial"



For the sake of comparison, the listwise deletion expected values

While this doesn't look very different from Amelia,

the t-statistic for College has shrunk from 2.2 to 2.0

Nudge both down another tenth, and MI would have be the difference

between a significant result and a non-significant one

Concluding thoughts

Multiple imputation using generic methods is usually more efficient and less biased than listwise deletion

Imputation methods vary in assumptions and techniques, and work best when assumptions are closely met

But even if the assumptions are a bit off or unverifiable, MI is still usually a better bet than LWD

With a good set of observed covariates and auxilliaries, even different MI techniques can lead to the same results

Concluding thoughts

Multiple imputation using generic methods is usually more efficient and less biased than listwise deletion

Imputation methods vary in assumptions and techniques, and work best when assumptions are closely met

But even if the assumptions are a bit off or unverifiable, MI is still usually a better bet than LWD

With a good set of observed covariates and auxilliaries, even different MI techniques can lead to the same results

Auxilliaries can be *critical*

In the turnout example, I used the 2005 high school graduation rate – available in all counties – as an auxilliary variable

Improved imputation considerably *and* led Amelia and MICE to agree

Implementing Amelia for cross-sectional data

In `R`, the `amelia()` function in the `Amelia` package does multiple imputation for cross-sectional, time series, and TSCS data

For cross-sectional data, it's usually very easy to make your imputed datasets:

```
library(Amelia)

# Run Amelia and save imputed data, and number of imputed datasets
nimp <- 5   # Use nimp=5 at minimum; 10 often a good idea
amelia.res <- amelia(observedData, m=nimp)
miData <- amelia.res$imputations
```

`MiData` is a list object with `nimp` elements, each of which is a complete dataset

Implementing Amelia for cross-sectional data

In `R`, the `amelia()` function in the `Amelia` package does multiple imputation for cross-sectional, time series, and TSCS data

For cross-sectional data, it's usually very easy to make your imputed datasets:

```
library(Amelia)

# Run Amelia and save imputed data, and number of imputed datasets
nimp <- 5   # Use nimp=5 at minimum; 10 often a good idea
amelia.res <- amelia(observedData, m=nimp)
miData <- amelia.res$imputations
```

`MiData` is a list object with `nimp` elements, each of which is a complete dataset

Then run your analysis `nimp` times in a loop, saving each result in a list object:

```
# Run least squares on each imputed dataset,
# and save results in a list vector
mi <- vector("list", nimp)
for (i in 1:nimp) {
  mi[[i]] <- lm(y ~ x + z, data=miData[[i]])
}
```

Implementing MICE for cross-sectional data

In `R`, the `mice()` function in the `mice` package does multiple imputation for cross-sectional data

The usage is slightly different from Amelia

```
library(mice)

# Run mice and save imputed data, and number of imputed datasets
nimp <- 5   # Use nimp=5 at minimum; 10 often a good idea
miceData <- mice(observedData, m=nimp, method="pmm")
            # method can be a vector w/ diff method for each var
```

`miceData` is a list object with many elements; see `?mice`

Then run your analysis `nimp` times in a loop, saving each result in a list object:

```
# Run least squares on each imputed dataset,
# and save results in a list vector
mi <- vector("list", nimp)
for (i in 1:nimp) {
  mi[[i]] <- lm(y ~ x + z, data=complete(miceData, i))
}
```

Multiple imputation for cross-sectional data

Regardless of imputation method, combine the results by drawing one-`nimp`th of your simulated β's from each model, like so:

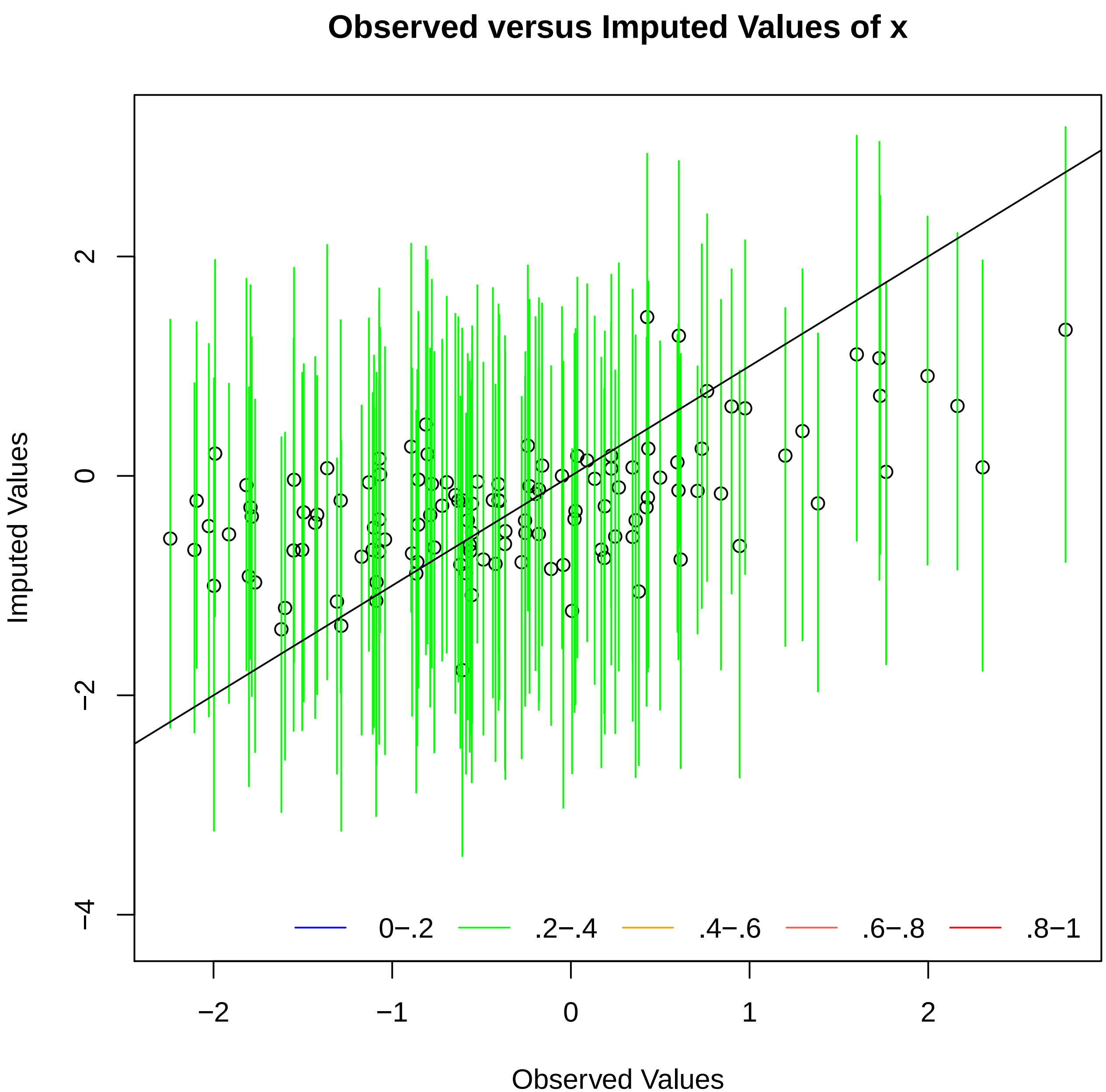
```
# Draw 1/nimp of the beta simulations from each run of least squares
sims <- 10000
simbetas <- NULL
for (i in 1:nimp) {
  simbetas <- rbind(simbetas,
                    mvrnorm(sims/nimp, coef(mi[[i]]), vcov(mi[[i]]))
                    )
}
```

From this point, you can simulate counterfactuals as normal using `simcf`

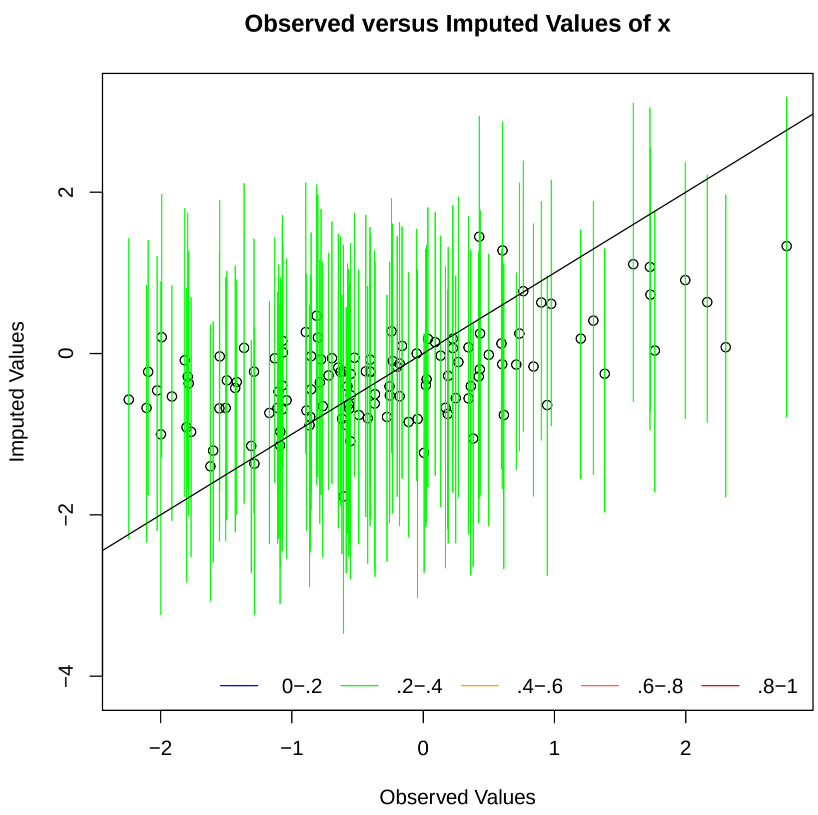
NB: you will need to either select an imputed dataset for computing means of variables, or average them all

Alternatively, you could have `zelig()` automate all of this, as Zelig knows what to do with Amelia objects

But it's usually best to write your own code for flexibility



Overimputation diagnostic: 90% of colored lines should cross the black line




```
pdf("overimputeX.pdf")
overimpute(amelia.res, var="x")
dev.off()
```

We did something similar earlier using MC data;
you could cook up your own version if you like