Ordinal Variables and Missing Data

April 7, 2021

General

- This paper should be an intersection of ordinal variables and missing data. Jeff thinks there is room there for a contribution
- The general idea is that general treatments of missing data (listwise deletion, multiple imputation etc.) lead to different results – depending on the type of variable. How you treat `Don't Know` and `Refused` and how this treatment affects the results depends greatly on the type of variable that you use to impute the missing data. The idea is that you would need to approach things differently depending on whether you use a nominal, interval, or ordinal variable to fill in values for `Don't Know/Refused`. So I am developing a method to specifically treat `Don't Know/Refused` with ordinal variables that improves over current uses that are generic for all types of variables
- This method should be a customized form of multiple imputation that is closer to the data than general multiple imputation. Jeff and Skyler developed affinity scores and hot decking in their BJPS paper. They used the number of exact matches (in the form of other participants) to calculate the affinity score. Instead, I should use a weighted distance solution between ordinal variable categories. I would use the OP model from the blocking paper to weight the distances between the categories in matches (in the form of other participants). In other words, I would use the underlying ordered probit numbers to create the weights. So this would be a specific ordinal variable adjustment of the affinity score building
- Set up chapter structure

Code

- My code functions for one variable with NAs, for 10,000 iterations
  - I have saved results for `inc`, `age`, `Dem`, `Rep` for `hd.ord` and `hd.norm`
  - For all 4, `na.omit` performs the best. This isn't surprising, since deleting observations with missing values shouldn't be a problem with MCAR
- My code does not function When run for several variables with NAs
  - It throws up a replacement error somewhere down the line when run for 10,000 iterations. It's never in the same place, so it must be something random
  - Always the same: # of NAs overall. Not always the same between some runs: # of NAs per column, # of rows with NAs
  - Find out why my code doesn't work for NAs in several variables
    - The number of NAs inserted wasn't actually the issue. It was in the `OPMord` code
    - I painfully ran 1,000 iterations for two variables, gradually adding one line at a time to the function. I discovered the error was where I assign values to `df.cases`, specifically where the columns are all combined and the resulting vector added to the data as `educ.new`

- I saved the vector to an empty vector, ran everything for 1,000 iterations, then looked at the vectors and the corresponding versions of `df.cases`
- There were rows with all NAs in `df.cases`, which makes the vector shorter than the data, which means it can't be assigned to it as a column. I reran the `df.cases` creation code to find the culprit
- I had overlooked that `int.df$Values` has one value fewer than there are variable levels, since it lists cutpoints between the levels. For 6 levels, I had assigned `int.df$length(levels(...))`, which picks the 6th value of `int.df$Values` – but it only goes to 5. I had to add `-1`. Now it's working

- Method to insert NAs
  - I tried `prodNA`, which works for `MCAR` but gives me nothing for `MAR`. And `MCAR` works well for `na.omit`, since by definition it's a random sample of missing data, which doesn't matter
  - Use `mice` `ampute()`
    - It has `MCAR` and `MAR` options. `MAR` is what I need
    - It doesn't work on just one variable; it needs at least two
    - `ampute()` works very well
- With the code adjusted (`-1`), `OPMord` now works. However, some of the `int.df`s don't have all the education levels: `int.df` with all levels has 6 rows (one fewer than the levels, since each row shows a cutoff), but some have 5. This means `OPMcut` now doesn't work
  - I could adjust `OPMcut` to work with fewer levels, but then I would be, in the end, taking means of most data with all levels and some data with fewer levels. That's problematic
  - It's better to discard the iterations where `int.df` has 5 rows after `OPMord` has run and then continue with the other functions. I set the code up to do that. However, after about 40 percent of running 12,500 iterations, there was an error: I hadn't factored in that the `int.df`s could also have fewer rows than that. A few of them had 4 rows, which caused the error. I adjusted the code to now discard the iterations where `int.df` has anything other than 6 rows
- The data ran for 12,500 iterations, 80 percent NAs, `Rep` and `inc`, `hot.deck.ord`, `hot.deck.norm`, `na.omit`. The results looked promising: `hd.ord` performed better than `hd.norm` and `na.omit`
- More methods
  - Include `mice`
  - Include `Amelia`
  - Use the defaults for both since that's what over 90% of people do (even if they shouldn't)
  - Include `hot.deck.norm` on the original education values (I ran it on the cutpoints)
- `hot.deck impContinuous`
  - No need to use it, since no variable is continuous (age has more values than `sdCutoff` but it's nominal, not continuous)
- `hot.deck sdCutoff`
  - No need to change the default
- Add more variables/NAs

  - Add NAs to 6 of the variables
  - Demographics I still have: Age(numeric), employment (5 levels), ideology (liberal, conservative, neither), following public affairs (ordinal, 4 levels), media interest (numeric, accumulative count of activities), participation (numeric, accumulative count of activities)
  - I can't add them all, since `mice` and `Amelia` don't work when there is collinearity
  - I used code to find and discard variables with high correlation
- Percentage of NAs
  - I'm currently using 80 percent. Jeff used 20, 50, and 80
  - I did one run each for 20, 50, and 80 percent
- Preliminary results
  - Across the board, `hd.ord` performs better than normal `hot.deck`, but worse than `Amelia` and `mice`
  - Across the board, `Amelia` is better than `mice` for `inc` and `age`. It's also better for `interest`, except for 20 percent NAs, where `mice` wins
  - For `Dem`, `Female`, and `White`, `Amelia` also edges first, thought `mice` sometimes is ahead by one unit or so in the fourth decimal
  - The difference is less pronounced for the binary variables and most visible in the nominal ones. It is worst for "age", which has the most unique values
  - As the percentage of NAs increases, the estimates are further off from the true means. This is to be expected for all methods, but it affects `hd.ord` and `hot.deck` much more than `Amelia` and `mice`
- Jeff likens this to the search for the pot of gold at the end of the rainbow. There will be something there, somewhere. I just need to find one specific scenario, one specific set of circumstances where I do particularly well or better. He gave me a bunch of pointers/ideas where to go from here. I'm using `ampute()` to generates `NAs` `MAR`. `ampute()` is part of `mice` – it's not a great surprise that `mice` then performs really well. `amelia()` performs even better – Jeff thinks `amelia()` calls `mice` at some point. `ampute()` has lots of default settings that are not immediately obvious. Jeff says to mess with those
- Run it with `bycases = FALSE` (`TRUE` introduces a very specific pattern of missingness), `cont = FALSE`, `type = "MID"`
  - I had to take out `White` because `ampute()` complained there were too many variables for `bycases = FALSE`
  - I ran it for 20 percent `NAs` and saved `framing.nas` and `ampute()` output – no significant change in results: `hd.ord` is better than `hot.deck` across the board. `Amelia` is better than `mice` for `inc`, `age`, and `interest`. `mice` is better for `Dem` and `Female`
  - The same for 10 percent
- Runnning it with ANES data on Code Ocean
  - Previous small tests have shown that the results should not be different when compared to the framing data
  - I have results from running things for 2,363 iterations, but the `amelia` runtime results were much faster than expected here. I did this run with `lapply`. I want to rerun this with a loop, as before, before I write anything up here

- Runtimes
  - `hd.ord()` outperforms `hotdeck()` (minimally) and massively outperforms `mice` and `Amelia` on the framing data
  - I have some ANES runtime results for fewer iterations (1,307)
  - I also have ANES runtime results for more iterations (2,363) obtained through `lapply`. Weirdly, `amelia` is almost as fast as `hd.ord` here ... I want to rerun this with a loop before I write anything up here
- Own function to create NAs as MAR
  - Jeff: "It's MAR if you 'delete' values in column 1 based on values in column 2", e.g. 'delete' the values for age where income is 1 and where income is 5
  - I wrote a function that samples the ID numbers for a specified percentage of each selected variable's unique values, then replaces the corresponding values of a different variable with NAs, based on the sampled ID numbers
  - I have run this function on the framing data
  - Seems to be more MCAR than MAR, since `na.omit` performs really well now
- Run `ampute` with MNAR for framing and ANES data
  - I've written everything up. `hd.ord` is doing well for binary variables overall, but less so for nominal and ordinal ones
  - MNAR or MAR makes no significant difference
- I've written up the `own.NA.rows` results
- It is not possible to run the whole ANES data for 10,000 iterations on Jeff's machine or on Code Ocean. It exhausts the memory. After endless trials, I finally managed to do it by sampling 1,000 observations, then running my code on that smaller ANES data set
- I added CCES to my data sets for analysis and set up several coding scripts
- With each data set sampled for 1,000 observations (old framing, CCES, ANES), I got great running times where `hd.ord` consistently performs far better than `amelia`
- What doesn't work so far
  - Increasing percentage of NAs == worse `hd.ord` performance
  - Ordinal and nominal variables == worse `hd.ord` performance
  - High number of observations == reduced number of iterations (crashes and/or RAM maxing out)
  - High number of observations == makes `amelia` faster relative to `hd.ord`
  - 17 ANES education levels == increases needed number of iterations
  - 17 ANES education levels == causes `amelia` to stop on CO and Jeff
  - `ampute()` with `bycases=FALSE` and `cont=FALSE` == worse `hd.ord` performance, good `na.omit` performance
  - `own.NA()` == `na.omit` performs best
  - `own.NA.rows()`== pretty much everything is zero, incl. `na.omit` (only `mice` is awful)
  - Running `hd.ord` with `method = p.draw` == worse `hd.ord` performance
  - Running `hd.ord` with `method = p.draw` == only works with only binary vars in the data
  - Increasing `sdCutoff` == only does something with only binary vars in the data
  - Only binary vars in data == no gain in `hd.ord` performance

- What works so far
  - Results for binary variables == equal performance of `hd.ord`, `mice`, `amelia`
  - Increasing number of variables with NAs (all, not just binary) == better `hd.ord` performance
  - 1000 observations in data sets == allows 10,000 iterations
  - 1000 observations in data sets == increases `amelia` running time
  - MNAR == MAR in terms of `hd.ord` performance
- I ran CCES for 14 NA variables for 10,000 iterations on CO. It took FOREVER because `mice` took more than 20 hours (!) on its own. I then ran the same code for 100 iterations on my machine. The results are virtually identical, so we really don't need all these thousands of runs. They're good to get good running time numbers, but not for other things. From now on I can test ideas like variables to add etc. on at most 1,000 iterations. No need for any more
- In addition, `mice` is at best identical to `amelia` regarding the results. Most of the times, `amelia` is better. If I do need to run things for 10,000-ish iterations again for some reason, consider leaving `mice` out of it, if possible. That massively cuts down the runtime of the code
- I selected and included more variables for ANES and framing, then ran those for 1,000 iterations. The results are getting even closer to zero across the board, including for `na.omit`
- I set up versions of all my functions to work with multiple ordinal variables: `OPMordMult`, `OPMcutMult`, `hot.deck.ord.mult`. I ran this for 1,000 iterations for CCES, ANES, and framing. It produced good results for all binary variables, though not better than before
- So far, I have really good results that are on par with `amelia` for lots of binary variables. I also have great running times where `hd.ord` is consistently at least twice as fast as `amelia` for all data sets with 1,000 observations and 10,000 iterations. That's a good step forward
- I don't think there is a scenario where `hd.ord` performs better. I'm reaching the stage where I would state that the theory of the latent underlying variable didn't pan out
- Emailed Jeff: I'm at the end of my research. I can't beat `amelia`, no matter the circumstances. But I get the same results for binary variables, with tons better running time. It looks like the theory of `polr` to estimate ordinal variables doesn't pay dividends. I want that to be enough for my diss
- Jeff responded: "Go ahead and write it up focusing on the speed and the rigor of your testing of all of these packages. Something like 'Quality Comparison of Major Missing Data Solutions for Discrete Data with a Proposed New Method'. That seems to reflect your extensive testing and understanding of software solutions"

Chapter write-up

- Write up the chapter focusing on the performance and speed of the packages under the different circumstances I tested
- Reorganize headings and general outline
- Identify where things are missing with +++
- Gradually keep filling missing sections

- Work in Jeff's feedback
- When I have implemented all of his feedback, Jeff doesn't need to see it again. I can send it out directly to Ryan and Liz
- Perhaps move one or two tables to the appendix

Pre-defense feedback Jeff

- Page 29, first sentence of S3.1: include "refusal" in the list of missingness
- Page 29, change the sentence that starts with "This poses a big problem" to "Missingness in datasets poses a big problem since without pre-processing such data cannot be appropriately analyzed with statistical software."
- Page 29-30 mention that deletion is also called case-wise deletion, list-wise deletion, and complete case analysis
- Page 31-32, why is this matrix **Y** instead of **X**? This is unnecessarily confusing. Say that **R** is an "indicator variable" matrix. Note that I've bolded these! And $\boldsymbol{\beta}$ too. Page 33 bold too. And throughout the document. Conventions like this exist because it makes the technical material easier to read. This is also a good reminder to say that you are concerned only with missingness in the explanatory variables. Or are you wrapping all the variables into one imputation matrix? The letter is the correct procedure, BTW. See:

  ```
  \bibitem    Graham, John W. 2009.
              \href{http://rhowell.ba.ttu.edu/graham2009.pdf}%
              {``Missing Data Analysis: Making it Work in the Real World.''}
              \emph{Review of Psychology} 60, 549-576.
  ```
- Page 32: "It is not possible to test respective data on the mechanism underlying their missingness." There is at least one weak test for MCAR vs. MAR by Rod Little. Restated on page 33.

  ```
  \bibitem    Little, Roderick J. A.  1988b.
              \href{http://www.jstor.org/stable/2290157}%
              {``A Test of Missing Completely At Random for Multivariate Data with Mi
              \emph{Journal of the American Statistical Association} 83, 1198-1202.

  \bibitem    {von Hippel}, Paul T.  2007.
              \href{http://smx.sagepub.com/content/37/1/83.full.pdf}%
              {``Regression With Missing Ys: An Improved Strategy for Analyzing Multi
              \href{http://smx.sagepub.com/content/37/1/83.full.pdf}%
              {Data.''}
              \emph{Sociological Methodology} 37, 83-117.
  ```
- Page 32-33: The discussion of the three types of missing data is too wordy at the expense of more precise equations
- Page 33: "often induces bias" contradicts "In the case of data MCAR, listwise deletion is not biased." Also, the first sentence is wrong. A better way to say all of this is that listwise deletion is not biased with MCAR data but does reduce the degrees of freedom and my hinder sub-group analysis. Listwise deletion is always biased with MAR and MNAR, the question is how much
- Page 34: "it biases standard errors and confidence intervals", you should say "down-

ward biases”

- Page 35: ”Mean imputation, sometimes also called unconditional mean imputation, means” should be ”Mean imputation, sometimes also called unconditional mean imputation, involves” (too many ”means” for one sentence)
- Everywhere: never use the phrase ”independent variables” because this never happens in the social sciences and especially with survey data
- Pages 38-89: Somewhere state that the last step of MI is really an ANOVA calculation
- Maybe you say it somewhere in this chapter but if you don’t then state the MI is advised with MCAR data too so as to not lose DF
- Pages 39-40: you can go to CRAN and find out how many times mice and Amelia have been downloaded
- Page 40: One of the original authors of mice has a book

```
\bibitem    van Buuren, Stef.  2007.
            \href{http://smm.sagepub.com/content/16/3/219.short}%
            {‘‘Multiple Imputation of Discrete and Continuous Data by Fully Conditi
            \emph{Statistical Methods in Medical Research} 16, 219-242.
```

- Pages 42-43 and 45: Why do you have β in the mice and Amelia processes? You’re not doing regression yet. This is pure imputation, before the regression
- Page 63: The information in Table 3.8 would be much more intuitive and clear as some graphs. Use the layout package to put them together
- Page 44: ”EM functions to a large part like IP”. Not at all: one is MCMC whereas EM is just a mode finder. Also, ”The Amelia package circumvents this issue with expectation-maximization importance sampling (EMi) (Gelfand & Smith, 1990; Rubin, 1987; Wei & Tanner, 1990),”. This is wrong. Gelfand and Smith introduce Gibbs sampling. Wei and Tanner introduce data augmentation, which is really a preview of MCMC. If you citing the Rubin book this is okay. But just about everybody cites DLR 1977 here
- Page 48: ”.4” should be ”0.4” and ”.6” should be ”0.6”. Check elsewhere. Page 57, 59, 61, 62, 63 too
- Tangential: do you want to put hd.ord into hot.deck and be one of the three coauthors?
- Page 53: ”Establishing whether real-life missing data is MNAR is a difficult feat.” I have found that sometimes it’s actually quite easy
- Page 54: ”Following Collins et al. (2001) and Honaker & King (2010), as many relevant pre- dictor variables as possible were used to impute each of the data sets.” Do these papers actually user the word ”predictor”? They really are predictor variables in way that word is used in statistics. I’ve seen ”helper,” ”observed,” and ”complete” much more often
- Page 65: change ”dependent variable” to ”outcome variable”
- Page 68: fourth line from the bottom, put a space before amelia