generating-missing-data-preliminary

Tony Ni

9/18/2020

Playing around with how to make vectors and datasets with missing values...

Some websites/sources:

https://cran.r-project.org/web/packages/missMethods/vignettes/Generating-missing-values.html https://rmisstastic.netlify.app/how-to/generate/misssimul

Look through this website: https://www.itrcweb.org/gsmc-1/Content/GW%20Stats/5%20Methods%20in%20indiv%20Topics/5%207%20Nondetects.htm#:~:text=Robust%20ROS%20is%20semi%2Dparametric,are%20made%20for%20t

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(missMethods) #package for generating missing data
library(NADA) #package for mle for left censored data
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'NADA'
```

```
## The following object is masked from 'package:stats':
##
##     cor
```

```
library(fitdistrplus) #package for mle imputation method
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

Generating missing data

Assuming log-normal distribution, which is the most common distribution for censored water data, (refer to technotes pdf file) for contaminant values in groundwater data...

Ok let's just try to pull random numbers from a lognormal(1, 1) distribution for a "artificial" dataset.

```
set.seed(7271999)

num <- 1000

m <- 1
s <- 1

location <- log(m^2 / sqrt(s^2 + m^2))
shape <- sqrt(log(1 + (s^2 / m^2)))

print(paste("location:", location))
```

```
## [1] "location: -0.346573590279973"
```

```
print(paste("shape:", shape))
```

```
## [1] "shape: 0.832554611157698"
```

```
id <- seq(1, num, by = 1)
value <- rlnorm(num, location, shape)

my_df <- as.data.frame(cbind(id, value))
```

Now, to play around with the methods in the `missMethods` package. These methods let us generate missing values in a dataset, in our case – the artificial dataset we generated above.

```
my_df2 <- delete_MAR_censoring(ds = my_df, #dataframe
                      p = 0.3, #probability that a value is missing
                      cols_mis = "id",
                      cols_ctrl = "value")

glimpse(my_df2)
```

```
## Rows: 1,000
## Columns: 2
## $ id    <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, NA, 12, NA, 14, 15, 16, 17, 18...
## $ value <dbl> 1.5557094, 0.9067474, 0.8399910, 1.0081827, 0.7199531, 1.9305...
```

Uh... I don't think the methods in this package are what we're looking for... This `delete_MAR_censoring` "generate MAR values using a censoring mechanism. This leads to a missing value in `id`, if the value is below the 30% quantile of `value`" We really only have 1 numeric variable of interest here...

Maybe we can just do it by hand...

Let's just say something like, for any value below `some number`, we make a new column and mark it as being censored/below limit of detection. I just chose 0.5 as a completely arbitrary value...

```
threshold <- 0.3

my_df3 <- my_df %>%
  mutate(below_detection = case_when(value <= threshold ~ "T",
                                     value > threshold ~ "F"))
tibble(my_df3)
```

```
## # A tibble: 1,000 x 3
##       id value below_detection
##    <dbl> <dbl> <chr>
##  1     1 1.56  F
##  2     2 0.907 F
##  3     3 0.840 F
##  4     4 1.01  F
##  5     5 0.720 F
##  6     6 1.93  F
##  7     7 1.14  F
##  8     8 1.20  F
##  9     9 0.842 F
## 10    10 0.847 F
## # ... with 990 more rows
```

Great! This is basically the general format of how missing values are encoded in the groundwater data – there is a variable called called `<` which is `<` is the value is below the LOD and left blank if not. This is exactly what we want! This is a bit weird because it doesn't make sense to have negative concentrations, but it's not worth it to try to fix it since we just want some artificial dataset to work with at the moment (may be in the future though)...

Playing around with missing data

Now, we want to try playing around with this dataset, let's try seeing what our sample mean and sd are – as a comparison point.

```
mean(my_df3$value)
```

```
## [1] 0.9599192
```

```
sd(my_df3$value)
```

```
## [1] 0.9049119
```

Now, let's make the values for which the observations in which `below_detection` is `T` – NA.

```
#have a 4th column of those actual values if above and just LOD if below

my_df4 <- my_df3 %>%
  mutate(value = if_else(
    below_detection == "F", value, NA_real_))

glimpse(my_df4)
```

```
## Rows: 1,000
## Columns: 3
## $ id              <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, ...
## $ value           <dbl> 1.5557094, 0.9067474, 0.8399910, 1.0081827, 0.71995...
## $ below_detection <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "...
```

Substitution Approach

Ok, let's true our substitution approach where we impute in the missing values with value/2 and value/sqrt(2). Um... for this artificial dataset, we don't really have a LOD but let's make it an arbitrary value 9.

First, replacing all our missing values with LOD/2

```
lod_div_2 <- my_df4 %>%
  mutate(value = if_else(
```

```
  below_detection == "T", 9/2, value))

mean(lod_div_2$value)
```

```
## [1] 1.562124
```

```
sd(lod_div_2$value)
```

```
## [1] 1.459535
```

Next, replacing all our missing value with LOD/sqrt(2)

```
lod_div_sqrt2 <- my_df4 %>%
  mutate(value = if_else(
  below_detection == "T", 9/sqrt(2), value))

mean(lod_div_sqrt2$value)
```

```
## [1] 1.823078
```

```
sd(lod_div_sqrt2$value)
```

```
## [1] 2.020845
```

We know the true population mean and sd of the distribution we pulled samples from is lognormal(1, 1)

The sample mean and sd of our artificial dataset is 0.95 and 0.905 respectively.

The mean and sd we get from LOD/2 substitution method yields 1.56 amd 1.46 respectively.

The mean and sd we get from LOD/sqrt(2 substitution method yields 1.82 and 2.02 respectively.

We can see that this method is bad... It doesn't really capture the true mean and sd very well...

MLE Approach

Now using the MLE approach, we can use the `cen_mle` function to compute statistics when the data contains left-censored values.

```
my_df5 <- my_df4 %>%
  mutate(below_detection = as.logical(below_detection))

mle_res = cenmle(my_df5$value, my_df5$below_detection, conf.int=0.95, dist = "gaussian")

mean(mle_res)
```

```
##      mean        se   0.95LCL   0.95UCL
## 1.0838646 0.0312669 1.0225826 1.1451466
```

```
sd(mle_res)
```

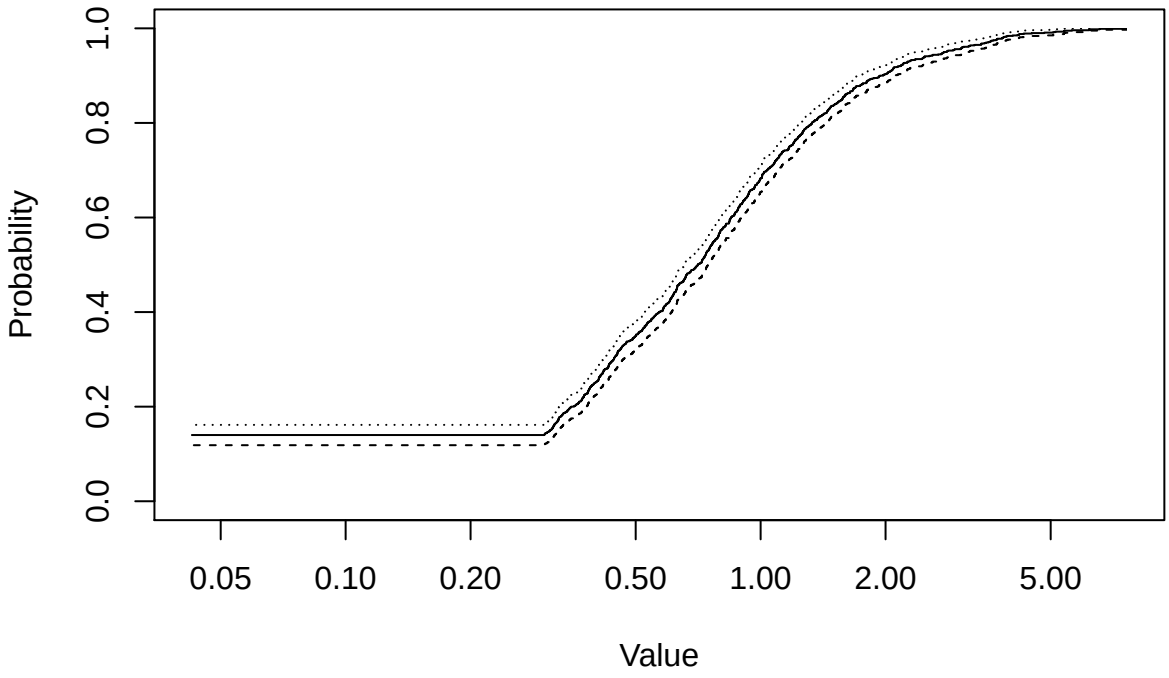
```
## [1] 0.9169254
```

We obtain a mean of 1.08 and sd of 0.92. This method performs much better than the substitution method!

Kaplan Meier Approach

This method doesn't have any distributional assumptions, which is an advantage it has over the MLE approach:

```
km_res = cenfit(my_df$value, my_df5$below_detection, conf.int=0.95)
plot(km_res) #plots cdf of value
```




```
mean(km_res)
```

```
##       mean         se    0.95LCL    0.95UCL
## 0.97425809 0.02824486 0.91889918 1.02961701
```

```
sd(km_res)
```

```
## [1] 0.893181
```

We obtain a mean of 0.97 and sd of 0.89. This method is comparable to the MLE method!

Imputation Method with MLE

```
#use fitdistcens from fitdistrplus package
#first arg df of 2 cols named left, right -- left is NA for censored obs, or value if not censored -- r

my_df5 <- my_df3 %>%
  mutate(left = if_else(
    below_detection == "F", value, NA_real_),
         right = value) %>%
  dplyr::select(left, right)

fitdistcens(my_df5, "lnorm")
```

```
## Fitting of the distribution ' lnorm ' on censored data by maximum likelihood
## Parameters:
##           estimate
## meanlog -0.4362249
```

```
## sdlog    0.9368073
```

```
"location: -0.346573590279973"
```

```
## [1] "location: -0.346573590279973"
```

```
"shape: 0.832554611157698"
```

```
## [1] "shape: 0.832554611157698"
```

We obtain a meanlog of -0.43 and sdlog of 0.93 (which compared to the original meanlog of -0.43 and sdlog of 0.937) is VERY close!

Imputation Method with KM

Feedback

- want to preserve the truth, dont want to overwrite value column
- BUILD UP dataset instead of making new one each time so we can compare in the end
- want to repeat this process multiple times when i set up simulation study (do this at a larger scale)
- read up on how to do simulation studies (prof baileys github) here: https://github.com/bebailey/research-tutorials/tree/master/04_coding-simulation
- what do we the structure to be like in the end?
- THINKING OF EXPLCIIT NAMING SCHEMES INSTEAD OF JUST MY_DF# AND 'value' (think of what the names should be)
- when we are generating data, the ideal simulation set up will be setting up a df, and then that df should be able to be put into the functions w/o any changes
- rerun these functions MULTIPLE times repeatedly with new datasets (does it do better if we assume lognormal dist vs. normal dist? how closely do these methods get to the actual statistics?)
- create a function to apply all of these functions to the generated data
- FOR NEXT WEEK: focus on understanding what a simulation study IS – put on pause the codework
- read the morris paper on simulation studies and then thinking about what it is i want to show with this simulation study (there is the big picture: comparison of how these methods perform vs the truth BUT what are we making these comparisons based off of, how do we know substitution method is better/worse etc. HOW DO WE COMPARE THE ACTUAL DISTRIBUTION OF VALUES – usually just summary statistics... we might look at overall mean and median of values or summary statistics based off on whether or not if they are LOD values – how does avg values compare btwn methods)
- after getting better sense of larger simulation study might look like – will have to go back into code (go back into lnorm and making sure that what you're inputting is what the function is asking for, the parameters that r takes in is not always what wikipedia takes in – verify parameters)
- verifying what the functions are outputting, can you get the rawdata, if so how? if not, are there anyways where we can?

ADEMPS

Aims

- Q: what are the aims of the study?
- A: (1) investigate effectiveness (accuracy?) of methods designed to estimate summary statistics (mean and sd) of left-censored data

Data-Generating Mechanisms

- Q: resampling vs. simulation from parametric dist?
- Q: how simple/complex should the model be?
- Q: should it be based on real data?
- Q: which factors to vary? which levels of factors to use?
- A: resampling from a simple lognormal distribution (not based on real data, for ease of calculations/comprehension)
- A: data generated from $n_{obs} = 1000$ wells that represent a possible collection of chemical concentration measurements (unrealistic, but simple – i personally think drawing it from the actual data itself might be not as useful? we could also vary the sample size to see how the methods perform when different % of the data is censored (refer to Bolks2014 – "different methods work better depending on small/large values of n and small/large percent of data censored))
- A: values talked about in paper are $n < 50$ compared to $n \geq 50$ and $< 50\%$ data censored compared to $50 - 80\%$ censored ($> 80\%$ is too high to compute summary statistics)
- A: let X_i be the concentration of an arbitrary chemical at a well with $X_i \in \text{Lognormal}(\mu, \sigma^2)$
- A: factors we might consider varying is the sample size n and the percent of data censored
- A: $\mu = 1$ and $\sigma^2 = 1$

Estimand

- Q: define estimands of the simulation study
- A: the estimands (variable which is to be estimated in a statistical analysis) of our study are the mean, and variance from which we generated our lognormal distribution form
- A: we want to observe the mean estimator $\bar{X}$, which is the estimator of μ
- A: we want to observe the variance estimator s^2, which is the estimator of σ^2

Methods

- Q: identify methods to be evaluated; are they appropriate?
- A: each simulated dataset(s) (with left censored data) is to be used in the following 5 methods (substitution, MLE, kaplan-meier. imputation with MLE, imputation with kaplan-meier) in order to compute the estimators s^2 and $\bar{X}$

Performance Measures

- Q: list all performance measures to be estimated
- Q: talk about how relevant they are to the estimands?
- Q: choose value of n_sim which achieves acceptable Monte Carlo SE (?)
- A: we will assess the following two performance measures for each method...
- A: root mean square error (RMSE): measures the difference in sample and population values predicted by an estiamtor/model, RMSE $= \sqrt{\sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}}$ where $\hat{y}_i$
- A: bias which is defined as: Bias $= E[\hat{\theta}] - \theta$ where θ is the estimator and $\hat{\theta}$ is the estimated value