Lab 6 - Missing Data and Imputation

TA: Isabel Laterzo

March 3, 2021

This week we will be learning ways to handle missing data. Often times you will come across, or collect, data with a significant amount of missingness across variables of interest. If the amount of data is quite small, typically you can ignore it. However, you will likely come across cases where your data has a pretty significant amount of missingness. A common rule of thumb for "too much" missingness - indicating you need to deal with it in some way - is if $> 5\%$ of your data is missing.

Today, we will be examining how we would deal with data that has a significant amount of missingness. We will walk through listwise deletion, mean/median/mode substitution, and something called "imputation."

Data! First, let's get our data set up.

```
#clear global environ
rm(list=ls())

#load in airquality data
data_clean <- airquality
summary(data_clean)
```

```
##      Ozone           Solar.R           Wind             Temp
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
##  NA's   :37       NA's   :7
##      Month            Day
##  Min.   :5.000   Min.   : 1.0
##  1st Qu.:6.000   1st Qu.: 8.0
##  Median :7.000   Median :16.0
##  Mean   :6.993   Mean   :15.8
##  3rd Qu.:8.000   3rd Qu.:23.0
##  Max.   :9.000   Max.   :31.0
##
```

```
#here's our "true model" for future reference
model_true <- lm(Temp~ Ozone + Solar.R + Wind, data = data_clean)

#lets copy this df and then add some missingness
data_miss <- data_clean
data_miss[4:9,3] <- rep(NA,6)
data_miss[1:8,4] <- NA
summary(data_miss) #check out those *~* NAs ~*~
```

```
##      Ozone           Solar.R           Wind             Temp
```

```
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :57.00
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:73.00
##  Median : 31.50   Median :205.0   Median : 9.700   Median :80.00
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.798   Mean   :78.59
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
##  NA's   :37       NA's   :7       NA's   :6        NA's   :8
##      Month            Day
##  Min.   :5.000   Min.   : 1.0
##  1st Qu.:6.000   1st Qu.: 8.0
##  Median :7.000   Median :16.0
##  Mean   :6.993   Mean   :15.8
##  3rd Qu.:8.000   3rd Qu.:23.0
##  Max.   :9.000   Max.   :31.0
##
```

Alright, let's take a look at those NA's more deeply

```
#what does this do?
is.na(data_miss$Ozone)
```

```
##   [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [37]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
##  [49] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [61]  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [73] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
#and this?
which(is.na(data_miss$Ozone))
```

```
##  [1]   5  10  25  26  27  32  33  34  35  36  37  39  42  43  45  46  52  53  54
## [20]  55  56  57  58  59  60  61  65  72  75  83  84 102 103 107 115 119 150
```

```
#and this?
sum(is.na(data_miss$Ozone))
```

```
## [1] 37
```

```
#apply this to other variables!

#An easy way to look at this across the whole DF
colSums(is.na(data_miss))
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##      37       7       6       8       0       0
```

Listwise Deletion Some people just delete the observations with missing values. Although this is definitely a viable approach, depending on how much missing data you have, you might be getting rid of a lot of information!

Below, use the function `na.omit()` to get rid of observations with missing values, then repeat the above `lm()`

model with this new data.

```
# subset with complete.cases to get complete cases
data_listwise <- na.omit(_____)
summary(data_listwise)

#let's create an lm with this model to compare to later iterations
#model Temp as your y, Ozone, Solar.R, and Wind as your x values
model_listwise <- lm(_____________, data = _________)

summary(____________)
```

Mean/Median Recoding

Alternatively, we can replace missing values with other values - such as the mean, median, or mode. Mode is appropriate if you're dealing with categorical variables, but since we're dealing with continuous here, we'll use one of the others.

Although this is a pretty easy approach, mean/median recoding decreases and changes the variance of your data. Especially in the case of time series data, this might not be great - we might be replacing a value with a mean of *all* the existing values, instead of one appropriate for the time period it belongs to! But, let's try it anyways.

```
#make a copy of the data
data_recode <- data_miss

#uses if else statements to recode NA values as the mean of the rest of that column
data_recode$Ozone <- _________________________

data_recode$Solar.R <- _________________________

data_recode$Wind <- _____________________________

data_recode$Temp <- _________________________

#summarize
summary(data_recode)


#linear model (same as above, but rename "model_recode")
model_recode <- lm(Temp~ Ozone + Solar.R + Wind, data = data_recode)

summary(model_recode)
```

Multiple Imputation

Now multiple imputation, a super useful technique. First, we'll discuss `mice`. This package uses multivariate imputations to estimate missing values. What does this mean? Well, if a value is identified as missing, the package will then regress over the other variables and predict the various missing values. This is pretty cool. Further, using *multiple* imputations, or iterations of this process, instead of just one, helps us reduce uncertainty.

Let's take a look!

```
library(mice) #mice package
library(VIM) #visualization

#mice's df visualization
```

```
md.pattern(data_miss)

#vis using aggr() from VIM
aggr_plot <- aggr(data_miss, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE,
                  labels=names(data_miss), cex.axis=.7, gap=3,
                  ylab=c("Histogram of missing data","Pattern"))
#what is this showing us?

#MICE TIME
mice_data <- mice(data_miss, #call our data
                  m=5, #number of multiple imputations
                  maxit=50, #number of iterations
                  # method = ? mice will decide the most appropriate method based on your data
                  seed=500)

summary(mice_data)


#try this out with 25 iterations and 10 imputations
#select animputation method (try pmm) - what does it do differently?
mice_data_2 <- mice(_____________)

summary(mice_data_2)
```

So this is great, but then what do we do if we want to run a model? We have five data sets. One option is to pool the data this way:

```
#pooling
model_mice <- with(mice_data,lm(Temp~ Ozone + Solar.R + Wind))
summary(pool(model_mice))
```

So mice is great. Let's also learn about Amelia II (my personal favorite, named after Amelia Earhart, because she is missing). Amelia is a little different because it assumes that your data is jointly distributed as multivariate normal. It uses the "expectation-maximization with bootstrapping" as an algorithm for imputation. The EM algorithm alternates between something called the expectation (E-step) and maximization (M-step) steps until it converges on values that were missing - it converges when the current and previous values are quite close (sounds a bit like Newton Raphson, no?). It brings in bootstraping by doing this process on multiple bootstrapped samples drawn from your original (and incomplete) data.

Great, let's try it out. Examine the function by calling `?amelia()` - look at the different arguments. Notice that Amelia can handle time series data uniquely. Let's factor that in by creating a running time variable.

```
library(Amelia)
library(lubridate)
library(mice) #for the pool function

#create running time variable, lets just assume we're in 2021
data_miss$time <- make_datetime(month = data_miss$Month,
                                day = data_miss$Day,
                                year = 2021)

#impute
amelia_data <- amelia(x = data_miss,
                  m = 5,
                  time = "time")
```

```
## -- Imputation 1 --
##
##   1  2  3  4  5  6  7  8  9
##
## -- Imputation 2 --
##
##   1  2  3  4  5  6  7  8  9 10 11 12 13
##
## -- Imputation 3 --
##
##   1  2  3  4  5  6  7  8  9 10
##
## -- Imputation 4 --
##
##   1  2  3  4  5  6  7  8  9
##
## -- Imputation 5 --
##
##   1  2  3  4  5  6  7  8  9 10 11
```

```
#let's run the model again, but this time using a for loop
models_amelia <- vector("list", 5)  #vector for our 5 newly imputed data sets

for(i in 1:5){
  models_amelia[[i]] <- lm(Temp ~ Ozone + Solar.R + Wind,
                           data=amelia_data$imputations[[i]]) #why do we index like this?
}

summary(pool(models_amelia))
```

```
##          term    estimate   std.error statistic        df      p.value
## 1 (Intercept) 71.77397685 2.625032902 27.342125 114.51869 0.000000e+00
## 2       Ozone  0.17048137 0.024146623  7.060257  64.11640 1.446086e-09
## 3     Solar.R  0.01000844 0.007166779  1.396505  55.89612 1.680834e-01
## 4        Wind -0.22581559 0.203820857 -1.107912 103.44868 2.704691e-01
```

Compare how models from mice, Amelia, and the above methods perform to our true data:

```
#true model
summary(model_true)

#listwise
summary(model_listwise)

#recoding
summary(model_recode)

#mice
summary(pool(model_mice))

#Amelia
summary(pool(models_amelia))
```

On your own:

Try Amelia again, but this time let's use more of its functionality. Read in data from the `africa` package and

examine its missingness. Then, using Amelia, impute missing values. Note that the dataset is cross-sectional by country, time series by year, and should have the `gdp_pc` variable specified as logged.

After imputing the data, write a linear model that evaluates the effect of `infl`, `trade`, `civlib` and `population` on `gdp_pc`. Report the pooled results!

```
library(Amelia)
data("africa") #read in

#explore missingness



#use Amelia to generate 5 imputed datasets
#use ts and cs so imputation algs knows countries and years
africa_imp <- amelia(_____________________)


models_africa <- vector(__)
for(i in 1:length(africa_imp$imputations)){
  ___________________________________)
}

#report results, including standard errors
```