Multiple Imputation using IVEware in SAS

Cody Olsen, MS
Biostatistics seminar
October 13, 2011

Last Time:
• The missing value problem
• Types of missing
• Simple approaches
  – Mean, median, mode, indicator, Hotdeck
• Multiple Imputation
  – Multivariate Normal
  – Chained Equations

Today:
• Multiple Imputation in SAS:
  – PROC MI
  – IVEware in SAS (with example)
  – PROC MIANALYZE
• Example using IVEware

Today:
• Multiple Imputation in SAS:
  – PROC MI
  – IVEware in SAS (with example)
  – PROC MIANALYZE
• Example using IVEware

Review of Multiple Imputation

Typical Multiple Imputation Analysis in SAS

Typical Multiple Imputation Analysis in SAS
Typical Multiple Imputation Analysis

Analysis Procedure

Data set with missing values

Set 1
Results 1

Set 2
Results 2

Set ...
Results ...

Set k
Results k

Combined Results

PROC MI

• Multivariate Normal Distribution
• Sounds great for continuous variables, but what about categorical variables?
  – Linear model—no rounding
  – Linear model—round to 0 or 1
  – Experimental CLASS statement**
  – Experimental DISCRIM statement**
• Simulation study:
  http://www2.sas.com/proceedings/sugi30/113-30.pdf

Monotone Missing: example

Monotone Missing

<table>
<thead>
<tr>
<th>ID</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not Monotone Missing

<table>
<thead>
<tr>
<th>ID</th>
<th>Y1</th>
<th>Y2</th>
<th>Y3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IVEware is another method of obtaining multiple imputed datasets

IVEware

Data set with missing values

Set 1
Results 1

Set 2
Results 2

Set ...
Results ...

Set k
Results k

Combined Results

IVEware

• Imputation and Variance Estimation Software
• Developed by University of Michigan Survey Research Center
  http://www.isr.umich.edu/src/smp/ive/
• Launched from SAS
• Creates a sequence of regression models
• Imputes by drawing values from posterior predictive distributions
Why IVEware?

- Practical issues we've had with PROC MI
  - Assumes multivariate normality
  - Crashes a lot
  - Upper and lower bound issues
  - Required to sort/manipulate the data and run a preliminary imputation to create a monotone missing pattern

Why IVEware?

- Monotone missing pattern not required
- You choose the assumed distribution: normal, categorical, count, normal/binary mixture
- Specifying upper and lower bounds for imputed values works well
- Can restrict imputation of a variable to a subset of cases

THE METHOD

IVEware

Sequential Regression Multivariate Imputation (SRMI)

- $Y_1, Y_2, \ldots, Y_k$ are variables with missing data
- $Y_1$ has the least amount of missing, $Y_2$ next least and so on
- $X$ are the variables with complete data, and an intercept
- $Y_1^{*}, Y_2^{*}, \ldots, Y_k^{*}$ are imputed variables

Sequential Regression Multivariate Imputation (SRMI)

- $Y_1 \sim X$
  Impute missing data in $Y_1$ from the corresponding posterior predictive distribution
- $Y_2 \sim X + Y_1^{*}$
  Impute missing data in $Y_2$ from the corresponding posterior predictive distribution
- Repeat for $Y_3, \ldots, Y_k$
- Repeat Process: $Y_1 \sim X + Y_2^{*} + \ldots + Y_k^{*}$
- Finished after a pre-specified number of rounds, or when the imputed values are stable

THE APPLICATION

IVEware
Installation

1. Download a zipped folder
   - http://www.isr.umich.edu/src/smp/ive/
2. Unzip files into a location that you create
3. Edit your SAS configuration file or point to those files using an options statement
4. Verify that it works using an example SAS program

Basics

- IVEware is similar to using a SAS macro
- You must submit code using the Program Editor

IVEware Modules

- IMPUTE: imputes missing values and outputs imputed data set(s)
- DESCRIBE: produces various estimates
- REGRESS: analyzes multiple datasets and combines results (6 model types)
- SASMOD: calls SAS procedures when analyzing data (for more complex analyses)

IVEware: IMPUTE module

```sas
%impute(name=impute, dir = ., setup = new)
DATAIN <input data set>;
DATAOUT <imputed data set name>;
CONTINUOUS <variable list>;
CATEGORICAL <variable list>;
MIXED <variable list>;
COUNT <variable list>;
DROP <variable list>;
TRANSFER <variable list>;
DEFAULT <variable type>;
RESTRICT <variable(logical expression)>
BOUNDS <variable(logical expression)>;
INTERACT <variable*variable…>
MAXPRED…; MINRSQD <R2>; …
ITERATIONS <n>; MULTIPLES <k>;
SEED; NOBS; PRINT; TITLE;
run;
```

• Required/Standard Statements

```
%impute(name=impute, dir = ., setup = new)
DATAIN <input data set>;
DATAOUT <imputed data set name>;
CONTINUOUS <variable list>;
CATEGORICAL <variable list>;
MIXED <variable list>;
COUNT <variable list>;
DROP <variable list>;
TRANSFER <variable list>;
DEFAULT <variable type>;
RESTRICT <variable(logical expression)>;
BOUNDS <variable(logical expression)>;
INTERACT <variable*variable…>
MAXPRED…; MINRSQD <R2>; …
ITERATIONS <n>; MULTIPLES <k>;
SEED; NOBS; PRINT; TITLE;
run;
```

• Declaring Variable Types
IVEware: IMPUTE module

%impute(name=impute, dir = ., setup = new)
DATAIN <input data set>
DATAOUT <imputed data set name>
CONTINUOUS <variable list>
CATEGORICAL <variable list>
MIXED <variable list>
COUNT <variable list>
DROP <variable list>
TRANSFER <variable list>
RESTRICT <variable(logical expression)>;
BOUNDS <variable(logical expression)>;
INTERACT <variable*variable…>
MAXPRED…; MINRSQD <R2>; …
ITERATIONS <n>; MULTIPLES <k>
SEED; NOBS; PRINT; TITLE 
run;
Example: Crash data

• Includes occupants and pedestrians involved in motor vehicle crashes
• Crash data from the scene
• Injury outcomes from the Emergency Department and/or Hospital

Example: Crash data

• Age: age in years
• Seatbelt: 1=belted, 0=not
• Injury: per crash report: 1=none, ..., 5=fatal
• Pertype: 1=driver, 2=occupant, 3=pedestrian
• BAC: blood alcohol content from crash report
• PostedSpeed: speed limit per crash report
• LOS: length of stay in days from hospital record

Define Variable Types

• Example:
  DEFAULT transfer;
  CONTINUOUS age;
  CATEGORICAL seatbelt injury pertype;
  COUNT los;
  MIXED bac;
  DROP postedspeed;

Optional Statements: RESTRICT

• Restrict imputation to certain observations
  – a logical expression identifies eligible cases
• When the restriction is not met, the variable will be set to a Not Applicable value
  – Continuous: zero
  – Categorical: one higher than the highest observed level

Optional Statements: RESTRICT

• Example:
  RESTRICT seatbelt(pertype = 1,2);
  – Imputes seatbelt (1=Yes, 0=No) to only pertype = 1(drivers) and 2 (passengers)
  – Sets seatbelt=2 for pedestrians
  – Before, seatbelt had levels: 0, 1, .
  – After, seatbelt has levels: 0, 1, 2

Optional Statements: BOUNDS

• Restrict the range of imputed values
• Does not change observed values which fall out-of-bounds
• Works for continuous models
• Does not work for categorical models
Optional Statements: BOUNDS

- Example: BOUNDS age (>0, <90);
  - Will not impute an age over 90 years-old
  - Will not change an observed age of 98 years-old

Model-Building Statements

- INTERACT: interactions that might be useful
- MAXPRED: maximum # of predictors to use in stepwise regression models
- MINRSQD: minimum marginal r-squared for stepwise regression
- OFFSETS: specify offsets for count models
- MAXLOGI: maximum number of iterative algorithms to be performed in a logistic/multilogit regression model

Model-Building Statements

- Example: Interact Age*Age Age*Age*Age;
  MAXPRED 4;
  MAXPRED Seatbelt(2);
  MINRSQD 0.0005;
  MAXLOGI 100

Output/Control Statements

- ITERATIONS: number of cycles you would like the imputation program to carry out for each data set
- MULTIPLES: number of output data sets
  - Total # of cycles = ITERATIONS x MULTIPLES
- SEED: random seed
- NOBS: number of observations to use
- PRINT: specifies what to print

Output/Control Statements

- Example: ITERATIONS: 5;
  MULTIPLES: 5;
  SEED 12345;
  NOBS 10000;
  PRINT details coef all;

EXAMPLE:

- Example to show how the process works, what output you get, etc. (CARS modified)
### IVEware gives you:

- Log also stored as name.log
  - Warnings, Errors, Notes
- Output also stored as name.lst
  - Code, coefficients, dataset summary, variable summaries
- Other setup files
  - Can be re-used if you don’t use setup=new
- Data set (saved where specified)

### LOG

- Look for any warnings, errors, or notes
- Common log messages:
  - Warning: Too many iterations, probably due to collinearity with dependent variable
  - You have multicollinearity among predictors in a model
  - Reduce the number of predictors with MAXPRED, or MINRSQD

### LOG

- Common log messages:
  - Warning: Only one code imputed for…
  - This one may be fine: if one level is very prevalent, and there are few missing
  - In other cases, it may indicate something is wrong: collinearity or an error in the data step
  - Check the distribution in the original and imputed data
  - Use MAXPRED or MINRSQD if collinearity could be the issue

### LOG

- Common log messages:
  - Warning: No degrees of freedom left for perturbations
  - Either your dataset is too small, or your number of variables too large
  - Reduce the number of predictors in each model using MAXPRED

### LOG

- Common log messages:
  - NOTE: Variable _cimpl_ is uninitialized.
  - This one is fine

### OUTPUT

- In your output window you will get:
  - The submitted code
  - Coefficients: one set for each model
    - # variables x # iterations x #multiples
  - Data Summary: # observed, imputed, and double counted for each variable
  - Summary of each variable
    - Mean, SD, Min, Max for continuous
    - Frequency Distribution for categorical
Coefficients
• Do the models make sense?
• Are there too few or too many predictors in your models?
• If you have a collinearity message, can you see which predictors may be causing it?

Summary
• Number observed and imputed
• Number double counted: the original data was changed! Usually because of a restriction
  – Let’s say we restrict imputation of seatbelt to drivers. What would happen to value of seatbelt for passengers?
  – It would become the N/A value, seatbelt=2, instead of 1 or 0 if either was observed

Summary
• Distributions of each imputed variable
  – Do you have N/A values where you should?
  – What if the distributions don’t look similar?
  – Sometimes it is hard to compare because of N/A values
  – Sometimes the imputed distribution will be different, and you can see why using the coefficients

Example
• Evaluate the imputation: log, coefficients, summary

Log and coefficients
• LOG: there seems to be a multicollinearity issue with type1
• Coefficients: a ton of coefficients…
• Decision: limit the number of predictors for type1

Update and %impute
• Add MAXPRED for type1
Run and Repeat…

• Try to get rid of all colinearity log messages
• Make sure the models are not always 'intercept only models'
• Make sure imputed variable distributions make sense

Analyzing the data

• IVEware modules:
  – DESCRIBE, REGRESS, SASMOD
• PROC MIAnalyze

Example

• How do vehicle type, origin, and engine size relate to retail mark up on a vehicle (MSRP – invoice)?

Summary

• IVEware:
  – Supports many types of regression models
  – Allows restrictions and bounds
  – Does not require monotone missingness
  – Runs in SAS and is more flexible and less error-prone than PROC MI
• Try it out!

References

• A multivariate technique for multiply imputing missing values using a sequence of regression models.
  – Raghunathan, Lepkowski, Van Hoewyk and Solenberger, Survey Methodology, June 2001
• Installation: http://www.isr.umich.edu/src/smp/ive/

References continued

• Advanced statistics: missing data in clinical research—part 1: an introduction and conceptual framework
• Advanced statistics: missing data in clinical research—part 2: multiple imputation
Questions/Discussion

Contact information:
Cody Olsen
cody.olsen@hsc.utah.edu
801 581-5755
Program: IVEWARE Imputation--CARS.sas
Author: Cody Olsen
Date: October, 2011
Description: 1) Creates a cars dataset with a restricted variable (AWD) and missing data to be imputed.
    2) runs a couple of imputations

**********************************************************************************;
** creating a dataset to impute: mycarsMISSING ***************************************;
**************************************************************************************;
** first of all, i need to run this, because my CONFIG file isn't set up;
options set = SRCLIB "C:\iveware\software\srclib" sasautos = ('!SRCLIB' sasautos) mautosource;
** Character variables will need to be converted to coded variables;
** Will use these formats to create and format coded versions of type, origin, and drivetrain;
proc format;
    value mytype 1='Hybrid' 2='SUV' 3='Sedan' 4='Sports' 5='Truck' 6='Wagon';
    value myorigin 1='Asia' 2='Europe' 3='USA';
    value mydrive 1='All' 2='Front' 3='Rear';
    value $mytype 'Hybrid'=1 'SUV'=2 'Sedan'=3 'Sports'=4 'Truck'=5 'Wagon'=6;
    value $myorigin 'Asia'=1 'Europe'=2 'USA'=3;
    value $mydrive  'All'=1 'Front'=2 'Rear'=3;
run;
** The Mazda RX-8 doesn't have 'cylinders' per se, and is missing;
**** Will make an indicator: nocylinders=1 if RX-8, 0 otherwise;
*** AWD;
**** let's add an extra variable, AWD, that only applies to drivetrain='All';
**** it will be 1 if the car is All Wheel Drive, and 0 if it is true 4-Wheel Drive;
**** I am just going to make this up. let's say it depends on type:
**** sedan, sports, & wagon: (80%), SUV: (50%), Truck: (20%);
data mycars;
    set sashelp.cars;
    call streaminit(12345);
    id = _N_;
    ** make character variables coded;
    type1 = input(put(type, $mytype.), 8.); format type1 mytype.;
    origin1 = input(put(origin, $myorigin. ), 8.); format origin1 myorigin.;
    drivetrain1 = input(put(drivetrain, $mydrive. ), 8.); format drivetrain1 mydrive.;
    ** Create AWD;
    y = rand('UNIFORM');
    if type = 'Sedan' and y< .8 then AWD = 1; else if type='Sedan' then AWD=0;
    if type = 'Sports' and y < .8 then AWD = 1; else if type='Sports' then AWD=0;
    if type = 'Truck' and y < .2 then AWD = 1; else if type='Truck' then AWD=0;
    if type = 'Wagon' and y < .8 then AWD = 1; else if type='Wagon' then AWD=0;
    if type = 'SUV' and y < .5 then AWD = 1; else if type='SUV' then AWD=0;
    ** AWD only applies to drivetrain = 'All';
    if drivetrain1 ne 1 then AWD = .;
    ** Mazda RX-8: nocylinders flag;
    if length(left(trim(model)))<4 then nocylinders=0;
    else if substr(left(trim(model)),1,4) = "RX-8" then nocylinders =1;
    else nocylinders=0;
    keep  id make model type1 origin1 drivetrain1 msrp invoice enginesize cylinders horsepower mpg_city mpg_highway weight wheelbase length awd nocylinders ;
run;
** creating missing randomly. varying between 10 and 50% missing;

data mycarsMISSING;
    set mycars;
    call streaminit(678910);

    t=rand('UNIFORM'); o=rand('UNIFORM');
    d=rand('UNIFORM'); m=rand('UNIFORM');
    i=rand('UNIFORM'); e=rand('UNIFORM');
    c=rand('UNIFORM'); h=rand('UNIFORM');
    mpg1=rand('UNIFORM'); mpg2=rand('UNIFORM');
    w=rand('UNIFORM'); wh=rand('UNIFORM');
    l=rand('UNIFORM'); a=rand('UNIFORM');

    if t<.1 then type1= .;
    if o<.2 then origin1= .;
    if d<.3 then drivetrain1= .; if m<.3 then msrp= .;
    if i<.4 then invoice= ;
    if c<.4 then cylinders= .; if h<.4 then horsepower= .;
    if mpg1<.2 then mpg_city= .; if mpg2<.3 then mpg_highway= .;
    if e<.4 then enginesize= .; if wh<.5 then wheelbase= .;
    if l<.2 then length= .;

    **if drivetrain is missing, AWD should be missing too;
    if drivetrain1 = . then awd = .;
    else if drivetrain1=1 and a < .20 then awd = .;
    else awd=awd;

    keep id make model type1 origin1 drivetrain1 msrp invoice enginesize cylinders horsepower mpg_city mpg_highway weight wheelbase length awd nocylinders ;
run;

** check out the missingness of mycarsMISSING;
proc means data = mycarsMISSING print n nmiss min max;
    var type1 origin1 drivetrain1 msrp invoice enginesize cylinders horsepower mpg_city mpg_highway weight wheelbase length awd;
run;

**************************************************************************************;
*** Impute the missing data!! ********************************************************;
**************************************************************************************;
** this has to be run in the program editor (go to VIEW->PROGRAM EDITOR);
** depending on your setup, you may need to run some options first (see setup guide);
/*options set = SRCLIB "\programs\srclib" sasautos = ('!SRCLIB' sasautos) mautosource; */

%impute(name=cars, dir =C:\iveware\Teaching , setup = new)
    DATAIN mycarsmissing;
    DATAOUT mycarsimputed ALL;
    CONTINUOUS MSRP INVOICE enginesize mpg_city mpg_highway weight wheelbase length cylinders;
    CATEGORICAL AWD typel origin1 drivetrain1;
    TRANSFER make model id;
    RESTRICT AWD(drivetrain1=1)
        cylinders(nocylinders=0)
    ;
    BOUNDS MSRP(>9000, < 200000) invoice(>9000, <200000)
        cylinders(>3, <=12) horsepower(>70, <550)
        mpg_city(>8, <=60) mpg_highway(>8, <=80)
        weight(>1500, <8000) wheelbase(>80, <150)
        length(>125, <250);
    INTERACT msrp*msrp invoice*invoice
        length*length length*length length
        weight*weight weight*weight weight
        horsepower*horsepower enginesize*enginesize;
    MINRSDQ 0.0001;
    ITERATIONS 5;
    MULTIPLES 5;
    SEED 12345;
    PRINT coef;
    TITLE Imputing Missing Cars Data;
run;
/** run 2: set maxpred for type1  */

%impute(name=cars, dir =P:\Users\colsen\Teaching\IVeware Multiple Imputation , setup = new)
    DATAIN mycarsmissing;
    DATAOUT mycarsimputed ALL;
    CONTINUOUS  MSRP INVOICE enginesize mpg_city mpg_highway weight wheelbase
               length cylinders;
    CATEGORICAL AWD type1 origin1 drivetrain1;
    TRANSFER make model id;
    RESTRICT AWD(drivetrain1=1)
               cylinders(nocylinders=0)
    ;
    BOUNDS  MSRP(>9000, < 200000)   invoice(>9000, <200000)
           cylinders(>=3, <=12)   horsepower(>70, <550)
           mpg_city(>8, <=60)    mpg_highway(>8, <=80)
           weight(>1500, <8000)  wheelbase(>80, < 150)
           length(>125, <250);
    INTERACT msrp*msrp invoice*invoice
           length*length length*length*length
           weight*weight weight*weight*weight
           horsepower*horsepower enginesize*enginesize;

    MINRSQD 0.0001;
    MAXPRED type1(10);
    ITERATIONS 5;
    MULTIPLES 5;
    SEED 12345;
    PRINT coef;
    TITLE Imputing Missing Cars Data;
run;

** question: How do vehicle type, origin, and engine size relate to retail mark up on a vehicle (MSRP - invoice)?;

data analysis;
    set mycarsimputed;
    Difference = MSRP-Invoice; **outcome;
    asia = origin1=1;    ** usa is ref;
    europe = origin1=2;
    all = drivetrain1=1;
    rear=drivetrain1=3;  ** front is ref;
    hybrid=type1=1;
    SUV=type1=2;
    Sports=type1=4;
    Truck=type1=5;
    Wagon=type1=6;      ** sedan is ref;
    _Imputation_ = _mult_;   ** mianalyze likes this to be named _Imputation_;
    drop _mult_;
run;

** sort by the imputation number;
proc sort data = analysis; by _Imputation_; run;

** run a procedure, output the parameter estimates;
proc reg data=analysis outest=outreg covout;
    model difference = asia europe  all rear hybrid svg sports truck wagon enginesize;
    by _Imputation_;
run;

** combine the parameter estimates in mianalyze procedure;
proc mianalyze data=outreg;
    modeleffects asia europe  all rear hybrid svg sports truck wagon enginesize;
run;