IVEware: I*mputation and* V*ariance* E*stimation Soft*ware[1]

T. Raghunathan
P. Solenberger
P. Berglund
J. van Hoewyk
University of Michigan
Ann Arbor,Michigan

November 19, 2022

Contents

Chapter 1

Basics

1.1 What is *IVEware*?

IVEware is a collection of routines written under various platforms and packaged to perform multiple imputations, variance estimation and, in general, draw inferences from incomplete data. The software can also be used to perform analysis without any missing data. *IVEware* defaults to assuming a simple random sample, but uses the Jackknife Repeated Replication or Taylor Series Linearization techniques for analyzing data from complex surveys.

IVEware can be run with SAS, Stata, R, SPSS or as stand-alone under the Windows or Linux environment. The R, Stata, SPSS and stand-alone version can also be used with the Mac OS. The stand-alone version has limited capabilities for analyzing the multiply imputed data though the routines for creating imputations are the same across all packages. The command structure is the same across all platforms. *IVEware* can be executed using the built-in XML editor or it can be run using the built-in editor within the four software packages previously mentioned. The user can also mix and match the codes from these software packages through a standard XML toggle-parser (for example, $< SAS\ \ name =$ "$myfile$" $>$ *SAS commands* $< /SAS >$ will execute the *SAS commands* and store the commands in the file "$myfile.sas$".) if the provided XML editor is used to execute *IVEware* commands.

1.2 Download and Setup

Various versions of *IVEware* can be downloaded and installed from www.iveware.org. Installation instructions and setup are slightly different for Windows, Linux and MAC operating systems, therefore, it is very important to follow the instructions for *IVEware* to work properly.

1.2.1 Windows

1. Download the `srclib_windows.msi` installer file. IVEware is downloaded from a University of Michigan Google Shared Drive, which will issue a canned warning: "Google Drive

can't scan this file for viruses. This file is executable and may harm your computer. Download anyway?" You can safely ignore the warning and download the installer.

2. Run the installer. The default directory for the installation is "`C:\Program Files\Srclib\`", but you can put it wherever you want it. The location you choose will replace "`~`/srclib" in the guides for using IVEware with R, SAS, SPSS, Stata and Srcware. The installer will create a desktop icon unless you tell it not to.

3. If the Windows Installer detects an existing installation, it will give you a choice of repairing (updating) the existing installation or removing it. If you want to update IVEware in its current location, click "Repair". If you want to change the location, click "Remove" and repeat the installation process.

4. If you want to save a previous version of IVEware or have it in more than one location, copy the "C:`\Program Files\Srclib`" directory (or wherever you installed it) and paste it where you want it, for example, as "`C:\Program Files\IVEware\Srclib\`". You could create a desktop icon targeting "`C:\Program Files\IVEware\Srclib\srcshell.exe`" or run from that address directly.

5. If you plan to use Srclib with R, SAS, SPSS or Stata and the version you want can't be invoked by its lower-case name (rscript for R), edit the Srclib\settings.xml file to provide the correct path. You can get the correct path from the properties of the desktop icon for the software.

6. To verify that Srclib is installed correctly, download the `srclib_examples_windows.zip` file, extract the Examples directory into an appropriate parent directory. You can put it wherever you want it.

7. Double-click the Srclib desktop icon, click "File" and then "Open", navigate to the Examples directory, open an appropriate setup file, for example, `ive_examples_srcware.xml` and click "Run".

8. Using MS Word or other software, check the *.log files produced by the run to see that there were no errors and compare the *.lst files produced by run with the corresponding *.chk files. They should differ only in the dates.

1.2.2 Linux

1. Download the `srclib_pclinux.tgz` file and extract the srclib directory into an appropriate parent directory, such as, /usr/local/ or `~/`. You can put it wherever you want it. The location you choose will replace `~/`srclib in the guides for using IVEware with R, SAS, SPSS, Stata and Srcware.

2. If you plan to use Srclib with R, SAS, SPSS or Stata and the version you want cannot be invoked by its lower-case name (rscript for R), edit the srclib/settings.xml file to provide

the correct path.

3. To verify that Srclib is installed correctly, download the `srclib_examples_pclinux.tgz` file, extract the examples directory into an appropriate parent directory. You can put it wherever you want it.

4. Navigate to the examples directory and use srcexec to run an appropriate setup file, for example, `~/srclib/bin/srcexec  ive_examples_sas.xml`.

5. OR run emacs with "`-l ~/srclib/srcshell.el`"as a command-line or icon option, click "File" and then "Open", navigate to the Examples directory, open an appropriate setup file, for example, `ive_examples_srcware.xml`, and press F8.

6. Using the Linux cat and diff commands or other software, check the *.log files produced by the run to see that there were no errors and compare the *.lst files produced by run with the corresponding *.chk files. They should differ only in the dates.

1.2.3 Mac OS

1. Download the `srclib_macosx.pkg` installer file. IVEware is downloaded from a University of Michigan Google Shared Drive, which will issue a canned warning:
"Google Drive cannot scan this file for viruses. This file is executable and may harm your computer. Download anyway?" You can safely ignore the warning and download the installer.

2. Run the installer. The default directories for the installation are "/Applications/Srcshell.app" for the app and "/Library/Srclib" for the library. The latter will replace "`~/`srclib" in the guides for using IVEware with R, SPSS, Stata and Srcware.

3. If you plan to use Srclib with R, SPSS or Stata and the version you want can't be invoked by its lower-case name (rscript for R), edit the /Library/Srclib/settings.xml file to provide the correct path.

4. To verify that the software is installed correctly, download the `srclib_examples_macosx.tgz` file, extract the examples directory into an appropriate parent directory. You can put it wherever you want it.

5. Double-click the Srcshell.app icon in the Applications folder, click "File" and then "Open", navigate to the examples directory, open an appropriate setup file, for example, `ive_examples_srcware.xml`, click "Run" and then "Run Srcexec".

6. Using the Mac/Linux cat and diff commands or other software, check the *.log files produced by the run to see that there were no errors and compare the *.lst files produced by run with the corresponding *.chk files. They should differ only in the dates.

1.3 Structure of *IVEware*

IVEware is organized into seven modules to perform various tasks. The six core modules are **IMPUTE**, **BBDESIGN, DESCRIBE**, **REGRESS**, **SYNTHESIZE** and **COMBINE** and the seventh module, **SASMOD,** is specific to SAS.

1. **IMPUTE** uses a multivariate sequential regression approach (Raghunathan et al (2001), Raghunathan (2015)). This approach is also called Chained Equations, (Van Buuren and Oudshoorn (1999)) and Fully Conditional Specification (Van Buuren (2012)) and is used to impute item missing values or unit non-response. IMPUTE can create multiply imputed data sets and can handle continuous, categorical, count and semi-continuous variables.

2. **BBDESIGN** implements the weighted finite population Bayesian Bootstrap approach to generate synthetic populations from complex survey data. The primary goal is to incorporate weighting, clustering and stratification in a nonparametric approach for generating the non-sampled portion of the population from the posterior predictive distribution, conditional on the observed data and the design information. For more details see Zhou, Elliott and Raghunathan (2015, 2016a, 2016b)

3. **DESCRIBE** estimates population means, proportions, subgroup differences, contrasts and linear combinations of means and proportions. A Taylor Series Linearization approach is used to obtain variance estimates appropriate for a user-specified complex sample design. Multiple imputation analysis can also be performed when there is missing data.

4. **REGRESS** fits linear, logistic, polytomous, Poisson, Tobit and proportional hazard regression models. For data resulting from a complex sample design, the Jackknife Repeated Replication technique is used to obtain variance estimates. As in other *IVEware* commands, a multiple imputation analysis can be performed when there are missing values.

5. **SYNTHESIZE** uses the multivariate sequential regression approach to create full or partial synthetic data sets to limit statistical disclosure (See Raghunathan, Reiter and Rubin (2003), Reiter (2002) and Little, Liu and Raghunathan (2004) for more details.) All item missing values are also imputed when creating synthetic data sets. However, DESCRIBE, REGRESS and SASMOD modules cannot be used to analyze synthetic data sets as they DO NOT implement the appropriate combining rules. Examples of implementation of correct combining rules for synthesized data sets are included in later sections of this guide.

6. **COMBINE** is useful for combining information from multiple sources through multiple imputation. Suppose that Data 1 provides variables X and Y, Data 2 provides variables X and Z and Data 3 provides variables Y and Z. COMBINE can be used to concatenate the three data sets and multiply impute the missing values of X, Y and Z to create large data sets with complete data on all three variables. All item missing values in the individual data sets will also be imputed. The multiply imputed combined

data sets can be analyzed using DESCRIBE, REGRESS and SASMOD modules (see Schenker, Raghunathan, and Bondarenko (2010) for an application and Dong, Elliott and Raghunathan (2014a, 2014b) for more details).

7. **SASMOD** (requires SAS) allows users to take into account complex sample design features when analyzing data with selected SAS procedures. Currently the following SAS PROCS can be called: CALIS, CATMOD, GENMOD, LIFEREG, MIXED, NLIN, PHREG, and PROBIT. A multiple imputation analysis can be performed when there are missing values. Unlike the other *IVEware* modules, SASMOD requires SAS.

There are many packages such as R ("with", "mitools", and "pool"), Stata ("mi estimate"), SAS ("PROC MI, PROC MIANALYZE") to analyze multiply imputed data sets. All these packages can be used within the "XML" structure of *IVEware*.

1.4 How to run *IVEware* with software packages?

There are many ways to run *IVEware*. The choice of how to run the program may depend upon whether the data is stored as a text file or as a software specific file (such as a SAS data set) or whether to use the Srcshell XML editor bundled with *IVEware* or use the built-in editors in specific software package (such as program editor in SAS). *IVEware* can also be run in batch mode using the command file. Given this level of flexibility, it is not possible to cover every method in detail. Nevertheless, the next few sections provide various example scenarios which might help users develop code for their own needs and situation. Additional examples of running *IVEware* are provided in later chapters.

This section uses the data from a case control study that was conducted in Seattle and King County to assess the relationship between dietary intake of omega-3 fatty acids (in particular, docosahexaenoic and eicosapentaenoic acids). These fatty acids are mostly derived from eating fish or seafood. Table 1.1 provides a list of variables and a description of the content. The goal of this example is to perform multiple imputation of the missing values.

1.4.1 *IVEware* and SAS

As explained earlier, *IVEware* can be run using the provided XML editor or using the **Regular Program Editor** in SAS (**NOT the Enhanced Editor**). The XML editor approach is described first as it is our most preferred approach. This analysis uses data stored in a text file ("mydata2.txt") with the first row representing the variable names. Create and save an XML file (the default extension is ".xml") with the following structure. The commands are explained whenever needed.

```
<sas name="ive_examples">

/* iveware examples - sas version */

/* import the input datasets */
```

Table 1.1: A list of variables in the data set used in the example

Variable	Description	Remarks
CASECNT	Case-Control Status	1=Case; 0=Control
AGE	Age at the time event (case) or interview (control)	Continuous
GENDER	Gender of the subject	1=Female 0=Male
RACE3	Race of the subject	1=White 0=Non-white
HYPER	Hypertension status	1=Hypertensive 0=Not hypertensive
DIAB	Diabetes Status	1=Diabetic 0=Non-diabetic
SMOKE	Smoking Status	1=Never 2=Former 3=Current
NUMCIG	Number of Cigarettes per week	Continuous, restricted to current and former smokers
YRSSMOKE	Number of years smoked	Continuous, restricted to current and former smokers, must be less than age
FATINDEX	Score measuring total Fat intake	Continuous
FAMMI	History of Family history	1=Yes 0=No
EDUSUBJ3	Education Categories	Less than High school, High school, some College and College
DHA_EPA	Dietary intake of Omega-3 based on Food Frequency Questionnaire	Continuous
REDTOT	Red cell membrane measure of Omega-3	Continuous
CHOLESTH	High cholesterol	1=Yes 0=No
CAFFTOT	Caffeine intake	Semi-continuous or Mixed
WGTKG	Weight in Kilograms	Continuous
TOTLKCAL	Total Kcal spent on physical activity	Continuous
ALCOHOL3	Alcohol Intake	Semi-continuous or Mixed
HGTCM	Height in centimeters	Continuous

```
proc import datafile='mydata2.txt' out=mydata2 dbms=tab replace;
getnames=yes;
run;
```

The first line

```
<sas name="ive_examples">
```

indicates the beginning of SAS commands which are to be stored in a file called "ive_examples.sas" in the current directory (the same directory where the XML file will be stored). The files called ive_examples.log and ive_examples.lst are the corresponding log and list files created by SAS. The command,

```
<sas name="ive_examples",dir="c:\mydir">
```

will store the sas, log and list files in the directory "c:\mydir".

Once the SAS toggle has been invoked, any SAS commands can be inserted including comments. Here, "PROC IMPORT" is used to import a text file and create a data set called "mydata2.sas7bdat" in the SAS work directory. The user can provide a full path for these filenames and also use libname in SAS to point to the directory containing the data files.

```
/* run iveware */

/* multiple imputation */

<impute name="impute">
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0) cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>
```

The command

```
<impute name="impute">
```

now toggles the beginning of the IMPUTE module and stores commands in a file called "impute.set" in the current directory (that is, directory where the XML file is stored). All *IVEware* files (produced by IMPUTE, DESCRIBE, REGRESS, COMBINE and SYNTHESIZE) have a ".set" extension. The filenames in "datain" and "dataout" can follow the SAS convention of "libname.sasname" while a libname can be assigned before invoking IMPUTE. For more detail on IMPUTE keywords, see Chapter 2. The line:

```
</impute>
```

indicates the closure or end of the **IMPUTE** commands. Finally, the

```
</sas>
```

command indicates the end of all SAS commands. Now, click "run" to execute the program. If you already have a setup file named "impute.set" in the directory, it will be overwritten.

Once you have run the Srcshell, you can reuse the SAS setup file it creates, "ive_examples.sas" without Srcshell by issuing the following command

```
sas ive_examples.sas
```

in a command window to run in batch mode. Similarly, in the Linux system, you can use

```
sas ive_examples.sas &
```

to run the program in background. Suppose that you have a setup file called "previous-run.set" from a previous run then you can rerun it by creating a simpler xml file consisting of the single line:

```
<impute name="previous" />
```

and then click "run".

IVEware can be run interactively in Windows SAS by using the Regular Program Editor (**once again, NOT the Enhanced Editor**). First, open or create and save a SAS program (.sas) file and then submit or run it as usual from the editor. This may be an attractive option, especially if you are used to running SAS and an earlier version of *IVEware*. For example, the first line in the SAS command file is the following options statement:

```
options set = SRCLIB '~/srclib/sas' sasautos=('!SRCLIB' sasautos) mautosource;
```

where

```
~\srclib
```

is the *IVEware* installation directory.

This approach (as opposed to the modification of the SAS configuration file used in the previous version of *IVEware*) is easier, especially, when the user does not have write privileges for the configuration file (such as with a network installation). The following commands can be used to perform the same set of tasks as in the XML version discussed above.

```
/* iveware examples - sas version */

/* import the input datasets */

proc import datafile='mydata2.txt' out=mydata2
dbms=tab replace;
getnames=yes;
run;

/* run iveware */

/* multiple imputation */

%impute(name=impute, dir=. setup=new);
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
```

Click "run" to submit the commands. You can modify the program and use the libname and other SAS features to read data from a different directory, store the output in another directory, and save and execute the program in some other directory.

1.4.2 *IVEware* and R

The structure of running *IVEware* in the R-package is very similar to running in the SAS environment as described in the previous section. Using the Srcshell XML editor, you can create the following commands.

```
<R name="ive_examples">

# The above line toggles the beginning of the R commands
```

```
# iveware examples - R version

# import the input datasets

mydata2 <- read.delim("mydata2.txt")
save(mydata2, file="mydata2.rda")

# run iveware

# multiple imputation

<impute name="impute">
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
 edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
 cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

# The line below indicates the end of R-commands
</R>
```

Click "run" to execute the command. This will create a file called "ive_examples.R" with R commands and the "impute.set" file with all needed Impute commands . If you already have the files with those names, then they will be overwritten.

Some users may prefer to run *IVEware* fully in the R environment instead of using the Srcshell XML editor. Use any text editor to create save an Impute setup file (say, "impute.set" or any other file with a ".set" extension) with the following structure. The command file along with the data file should either be in the same directory or you can simply provide the full path in the command file.

```
title Multiple imputation;
```

```
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
 edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
```

Next Start R. In the R editor, type the R commands and execute as usual.

```
# iveware examples - R version

# import the input datasets

mydata2 <- read.delim("mydata2.txt")
save(mydata2, file="mydata2.rda")

# initialize srclib

srclib <<- "~/srclib/R"
source(file.path(srclib, "init.R", fsep=.Platform$file.sep))

# run iveware

# multiple imputation

impute(name="impute")
```

Use the full path of the actual directory where *IVEware* is installed for

```
~/srclib
```

1.4.3 SPSS and *IVEware*

Running *IVEware* with SPSS is slightly different. Use the Srcshell XML editor and type the following commands:

```
<spss name="ive_examples">

/* iveware examples - spss version */

/* import the input datasets */

get translate file="mydata2.txt" /type=tab /fieldnames.
save outfile="mydata2.sav".

/* run iveware */

/* multiple imputation */

<impute name="impute">
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
 edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

</spss>
```

Click "run". When SPSS opens its interactive command window, select "all" and click the "run" icon to execute the commands.

A successful run of Srcshell with SPSS creates an SPSS setup file called ive_examples.sps for this example. You can modify the file in the future and run without Srcshell by issuing the following command:

```
spss ive_examples.sps
```

Some users may prefer to run using the editor in SPSS without using the Srcshell editor. The following commands illustrate the use of the SPSS editor. Change to the working

directory and use the SPSS Syntax Editor to create and save an Impute setup file (say, "impute.set") with the following structure:

```
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
 edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
 cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
```

Using the Syntax Editor, create and save an SPSS setup file (say, "ive_example.sps") with the following commands.

```
begin program.
import sys
sys.path.insert(0, '~/srclib/spss')
import srclib
end program.

/* iveware examples - spss version */

/* import the input datasets */

get translate file="mydata2.txt" /type=tab /fieldnames.
save outfile="mydata2.sav".

/* run iveware */

/* multiple imputation */

begin program.
srclib.impute(name="impute")
end program.
```

Select "all" and click the "run" icon. For

```
~/srclib
```

use the exact path of the installation directory of *IVEware*.

1.4.4 Stata and *IVEware*

To run Stata with the Srcshell XML editor, type the following commands:

```
<stata name="ive_examples">

/* iveware examples - stata version */

/* import the input datasets */

insheet using mydata2.txt, clear names case tab
save mydata2, replace

/* run iveware */

/* multiple imputation */

<impute name="impute">
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

</stata>
```

Click "run". Once you've run Srcshell, a Stata setup file called "ive_examples.do' is created. As with other packages, you can modify the file for future use with any editor and run using the following commands:

```
stata ive_examples.do
```

As in the case of other software, *IVEware* can be run using the Stata built-in editor. To do so, change to the working directory and utilize the Stata do-file editor to create and save an Impute setup file ("impute.set") with the following structure:

```
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
```

In the do-file editor, open or create a Stata setup file with the following structure:

```
global srclib "~/srclib/stata"

/* iveware examples - stata version */

/* import the input datasets */

insheet using mydata2.txt, clear names case tab
save mydata2, replace

/* run iveware */

/* multiple imputation */

global name "impute"
do $srclib/impute
```

Click "run". As before,

```
~\srclib
```

is the name of the *IVEware* installation directory.

1.5 How to run *IVEware* as stand-alone

IVEware can be used as stand-alone software (SRCWARE) for performing multiple imputation (using the **IMPUTE** module) and perform analyses using **DESCRIBE, BBDESIGN, REGRESS, SYNTHESIZE, or COMBINE,** with or without incorporation of complex design features. This section provides example code for reading the data from a text file and performing multiple imputation. See later chapters for additional examples of code and output for other analyses/modules. Like other software packages, SRCWARE can be run using the XML editor or through creation of a setup file using any text editor executed in a command window.

First, click on the Srclib icon and choose "File" and "New" to create a new command file. The **GETDATA** module is used to read a text file containing data, and subsequently attach variable name, type (character or numeric) and formats (optional). The user can specify the delimiter as comma ("csv"), space ("\s"), tab ("\t") etc., and the number of rows to be skipped prior to reading data from the text file. The following commands read the example file "myfile2.txt" provided with the software and also described in Section 1.4:

```
<srcware name=‘‘ive\_examples’’>
/* iveware examples - srcware version */

/* import the input datasets */

<getdata name="mydata2">
data mydata2.txt;
metadata;
delim "\t";
skip 1;
variables
name=STUDYID type=char;
name=CASECNT type=num;
name=AGE;
name=GENDER;
name=RACE3;
name=HYPER;
name=DIAB;
name=SMOKE;
name=NUMCIG;
name=YRSSMOKE;
name=FATINDEX;
name=FAMMI;
name=EDUSUBJ3;
name=DHA_EPA;
name=REDTOT;
name=CHOLESTH;
name=CAFFTOT;
```

```
name=WGTKG;
name=TOTLKCAL;
name=ALCOHOL3;
name=HGTCM;
end;
run;
</getdata>
```

Note that "skip 1;" instructs GETDATA to skip the first row and the "delim "\t" " states that this is tab-delimited data. The keyword "metadata" begins entering of information about the data, and the keyword "variables" indicates the beginning of establishing the name and type of the variables in the columns through use of the keywords "name" and "type". Finally, "end" closes the entering of metadata. The full extent of what can be specified under "metadata" keyword is specified below:

```
metadata
variables
name=gender
type=num
label=‘‘Respondent’s Gender’’
codeframe=sexfmt
location=number
width=number
decimals=number
missing=-9;
codeframe sexfmt 1 male 2 female -9 missing;
end;
```

The above defines the name, type, label and format of the variable "gender". For non-delimited data, ***location number*** indicates the starting location column number for the variable and ***width number*** specifies the width of the given variable. This feature cannot be used with delimited data. The default location number is 1 for the first variable. In general, the location for any variable is the previous variable location plus the width of the previous variable. ***Decimals Number*** is used to specify the number of implicit decimal places for the variable. This defaults to 0 for character and non-consecutive variables, and to the previous number of decimals for non-character variables after the first in a series. The missing data value for the variable is "." or any other missing data type of character(s) such as .N or .J. with a default of none.

Another option is to put the variable names as the first row and use the following code to read the data set:

```
<getdata name=‘‘mydata2’’>
table mydata2.txt;
</getdata>
```

The following code is then used to specify the multiple imputation:

```
/* run iveware */

/* multiple imputation */

<impute name=‘‘impute’’>
title Multiple imputation;
datain mydata2;
dataout impute;
default continuous;
categorical casecnt gender race3 hyper diab smoke fammi
 edusubj3 cholesth;
mixed cafftot alcohol3;
transfer studyid;
restrict numcig(smoke=2,3) yrssmoke(smoke=2,3);
bounds numcig(>0) yrssmoke(>0,<=age-12) fatindex(>0)
cafftot(>=0) alcohol3(>=0);
maxpred redtot(3) wgtkg(2);
minrsqd .01;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

</srcware>
```

To execute, save the file and click “Run’.

1.6 Reading and Writing Other Software Formats

It may be easier to use some other software to write to a file that can be read by *IVEware*. This section provides some useful commands for importing and exporting data to other popular software packages.

1. To write a Gauss data set as a tab-delimited text table that can be read by Srcware, run the following Gauss command:

   ```
   rc = export(mydata, "mydata.txt", mynames)
   ```

   To read a tab-delimited text table created by Srcware, run the following Gauss command:

   ```
   {mydata, mynames} = import("mydata.txt", 0, 1)
   ```

2. To write an R data set as a tab-delimited text table that can be read by Srcware, run the following R command:

```
write.table(mydata, file="mydata.txt", na="", row.names=FALSE,
qmethod="double", sep="\t")
```

To read a tab-delimited text table created by Srcware, run the following R command:

```
mydata<-read.table("mydata.txt", header=TRUE, sep="\t")
```

3. To write a SAS data set as a tab-delimited text table that can be read by Srcware, define the library and run the following SAS program:

```
proc export data=mylib.mydata outfile='mydata.txt'
dbms=tab replace; run;
```

To read a tab-delimited text table created by Srcware, define the library and run the following SAS program:

```
proc import datafile='mydata.txt' out=mylib.mydata dbms=tab
replace; getnames=yes; run;
```

Specify "-noterminal" in the SAS invocation to export/import delimited data sets in non-interactive command-line mode.

4. To write an S-Plus data set as a tab-delimited text table that can be read by Srcware, define the library and run the following S-Plus command:

```
write.table(mydata, file="mydata.txt", dimnames.write="col",
na="", sep="\t")
```

To read a tab-delimited text table created by Srcware, define the library and run the following S-Plus command:

```
mydata<-read.table("mydata.txt", header=T, row.names=NULL,
 sep="\t")
```

5. To write an SPSS data set as a tab-delimited text table that can be read by Srcware, run the following SPSS command:

```
save translate outfile="mydata.txt" /type=tab /fieldnames
 /replace.
```

To read a tab-delimited text table created by Srcware, run the following SPSS command:

```
get translate file="mydata.txt" /type=tab /fieldnames.
```

The "get" translate function exists only in the Windows implementation of SPSS.

6. To write a Stata data set as a tab-delimited text table that can be read by Srcware, open the data set in Stata and run the following command:

   ```
   outsheet using "mydata.txt", replace nolabel
   ```

   To read a tab-delimited text table created by Srcware, run the following Stata command:

   ```
   insheet using "mydata.txt", clear
   ```

7. To write a SUDAAN data set as a tab-delimited text table that can be read by Srcware, save it as a SAS data set and run the following SAS program:

   ```
   proc export data=mylib.mydata outfile='mydata.txt' dbms=tab
   replace; run;
   ```

   To read a tab-delimited text table created by Srcware, read it as a SAS data set after running the following SAS program:

   ```
   proc import datafile='mydata.txt' out=mylib.mydata dbms=tab
   replace; getnames=yes; run;
   ```

Chapter 2

IMPUTE

2.1 Introduction

The IMPUTE module is a general-purpose multivariate imputation procedure that can handle relatively complex data structures when the data are missing at random (Rubin, 1976). Survey data sets often consist of large numbers of variables that have a variety of distributional forms. Typically, such data sets have hundreds of variables, some continuous, others counts, many dichotomous or polytomous, and semi-continuous or limited dependent variables. IMPUTE can handle such complex data structures.

IMPUTE produces imputed values for each individual in the data set conditional on all the values observed for that individual using the sequential regression approach (also called Chained Equations or Flexible Conditional Specifications). The basic strategy is to create imputations through a sequence of multiple regressions, varying the type of regression model by the type of variable being imputed. Covariates include all other variables observed or imputed for that individual. The imputations are defined as draws from the posterior predictive distribution specified by the regression model with a flat or non-informative prior distribution for the parameters in the regression model. The sequence of imputing missing values can be continued in a cyclical manner, each time overwriting previously drawn values, building interdependence among imputed values and exploiting the correlational structure among covariates. To generate multiple imputations, the same procedure can be applied with different random starting seeds or by taking every p^{th} imputed set of values in the cycles mentioned above. For details see Raghunathan et. al. (2001) and Raghunathan (2015).

IMPUTE assumes the variables in the data set are one of the following five types: continuous; binary; categorical (polytomous with more than two categories); counts; and mixed (a continuous variable with a non-zero probability mass at zero). The types of regression models used are linear, logistic, Poisson, generalized logit or mixed logistic/linear, depending on the type of variable being imputed.

IMPUTE can also accommodate two common features of survey data that add to the complexity of the modeling process: (1) the restriction of imputations to sub-populations; and (2) the bounding of imputed values. First, certain restrictions are imperative, requiring the sub-setting of sample individuals to satisfy particular criteria while fitting the regression models. For example, the variable "Number of Years Since Quit Smoking" is defined only

for former smokers; hence, the imputation process for this variable should be restricted only to former smokers. Restrictions also arise due to skip patterns in the questionnaire. For example, certain questions about income from a second job are asked only when the respondent indicates having a second job. The imputation of such variables has to be handled in a hierarchical manner.

Second, there are certain logical or consistency bounds for missing values that must be incorporated in the imputation process. Such interrelationships among the variables make the model specification difficult. For instance, "Years of Smoking" should not only be restricted to current or past smokers but the imputed values might be required to be less than a specified number years, based on other respondent characteristics, such as evidence of smoking as a teen-ager. In such a case, the imputed upper bound for "Year of Smoking" might be the respondent's current age minus 12. This assumes that the respondent may have started smoking at 12 years of age. For a former smoker, "Year of Smoking" would also have take into account years since the respondent stopped smoking. Another example of bounds is discussed in Heeringa, Little and Raghunathan (1997). They address imputation of bracketed response questions in which a respondent is unable or unwilling to provide an exact response (e.g., income and assets), but does define the bounds within which the imputed values must lie. The bounds involve drawing values from a truncated predictive distribution.

Any imputation software package is a tool that needs to be used judiciously. To obtain a valid imputation each regression model needs to carefully developed and specified by the user. Developing such good prediction models requires exploratory data analysis, model building and model checking through residual diagnostics. Thus, if there are p variables in the data set with missing values then p regression models have to be developed appropriately for this software package to produce statistically valid results. There are many good books on regression that discuss model building strategies (for example, Weisberg (2013), Atkinson (1985), Vittinghoff, Glidden, Shiboski and McCulloch (2005) and Gelman and Hill (2006)). Raghunathan (2015) discusses model building and model checking strategies in the context of missing data.

2.2 Required IMPUTE Statements

2.2.1 Input and Output Data Sets

DATAIN *filename*;
This required statement identifies the location and name of the input data set. For example, in a SAS environment, the ***filename*** can be expressed as "libname.sasdata". In other environments, read the data set and include the name of the data set in the ***filename***.

DATAIN *Mylib1.Mydata*;

indicates that the SAS data file **Mydata** is located in the library **Mylib1.** **Mylib1** is the name assigned to a directory with the SAS Libname statement. (See later sections for examples).

DATAOUT *outfile [ALL]*;

This statement identifies the location and name of the output dataset containing the imputed data. The ALL keyword is optional. If it is specified and more than one imputation is generated (see keyword **MULTIPLES**) then the output dataset will be a concatenation of the multiple imputed data sets. The system variable "_MULT_" , automatically added to the output file, can be used to distinguish each imputation.
For example,

DATAOUT Mylib2.Impdata ALL;

will store the SAS file ***Impdata*** in the library ***Mylib2***, a pointer to the directory with appropriate SAS libname statement.

2.2.2 DECLARING VARIABLE TYPES

IMPUTE requires that the SAS data set variables be defined by type. Six types of variables are recognized by the IMPUTE module: continuous, categorical (binary is included as categorical), count, mixed, transfer and drop. If no variable types are specified, all variables will be assumed to be continuous. Variable types should be declared before any BOUNDS, INTERACT, or RESTRICT statements (see below).

CONTINUOUS *variable list*;

Variables declared as CONTINUOUS may take on any value on a continuum. Income is an example of a continuous variable. A normal linear regression model is used to impute the missing values in these variables. You may want to transform the variable to achieve normality and then impute on the transformed scale. After imputation you may re-transform the variable back to its original form.

CATEGORICAL *variable list*;

CATEGORICAL variables have values that represent discrete values. Gender is a categorical variable. A logistic or generalized logistic model is used to impute missing categorical values.

MIXED *variable list*;

Variables declared as MIXED are both categorical and continuous. In a mixed variable a value of zero is treated as a discrete category, while values greater than zero are considered continuous. Alcohol consumption is an example of a mixed variable. A two stage model is use to impute the missing values. First, a logistic regression model is used to impute zero vs. non-zero status. Conditional on imputing a non-zero status, a normal linear regression model is used to impute non-zero values.

COUNT *variable list*;

COUNT variables have non-negative integer values. A Poisson regression model is used to

impute the missing values. The number of annual doctor visits is an example of a COUNT variable.

Sometimes a normal linear regression model is not appropriate because, for example, the distribution of the residuals appear non-normal based on the residual diagnostics. For such variables there are two options (see He and Raghunathan (2006), Bondarenko and Raghunathan (2010) and Raghunathan, Berglund and Solenberger (2017)), **ABB** (Approximate Bayesian Bootstrap) and **GH** (Tukey's *gh*-distribution). These can be specified as

ABB *varlist*;

or

GH *varlist*;

where ***varlist*** are the continuous or mixed variables declared earlier.

DROP variable list;

Variables listed after the DROP keyword will be excluded from the imputation procedure and will not appear in the imputed data set.

TRANSFER *variable list*;

Variables listed after the TRANSFER keyword are carried over to the imputed data set, but are not imputed nor used as predictors in the imputation model. Transfer variables, however, can be used in the RESTRICT and BOUNDS statements (see below). ID is an example of a variable that you might want to treat as a transfer variable or any variables not to be used as predictors in the imputation process (for all the variables being imputed).

DEFAULT *variable type*;

variable type can be Continuous, Categorical, Count, Mixed, Transfer or Drop. This keyword declares that by default all the variables in the data set should be treated as the ***variable type***. The most efficient use of the DEFAULT statement is to declare the most numerous variable type in your data set as the default type, eliminating the need to type a long list of variables. The DEFAULT statement must be given before declaring other variable types.

RUN;

This should be the last statement in your setup file.

2.3 Restrictions and Bounds

RESTRICT *variable(logical expression)*;

This command is used to restrict the imputation of a variable to those observations that satisfy the logical expression. For instance, suppose that the variable ***yrssmoke*** indicates the number of years an individual smoked, and the variable ***smoke*** takes the value 1 for a current smoker, 2 for a former smoker or 3 for someone who never smoked.
Then the declaration,

RESTRICT yrssmoke(smoke=1,2);

will impute ***yrssmoke*** values only for current and former smokers. It will automatically set ***yrssmoke*** equal to 0 for never smokers.
Restrictions on more than one variable may be combined as follows:

RESTRICT ***yrssmoke(smoke=1,2) births(female=1) income(employed=1)***;

When the restriction is not met, the value of the restricted variable will be set to zero for a continuous and count variables. For a categorical variable, a separate category will be created with the response code, one higher than the highest observed code for the restricted categorical variable. For example, the statement,

RESTRICT ***smoke(age>= 13)***;

where smoke has 3 categories as described below, will create a category **4** for those with ***age <= 12***.

BOUNDS ***variable (logical expression)***;

This keyword is useful for restricting the range of values to be imputed for a continuous variable.
For example,

BOUNDS ***yrssmoke (> 0,<= Age-12)***;

will ensure that the imputed values for ***yrssmoke*** are between 0 and the individual's Age minus 12. Smoking is assumed not to begin before the age of 12.
Again, as in the RESTRICT statement more than one variable can be included in the BOUNDS statement.
For example,

BOUNDS ***yrssmoke (>0,<= Age-12) numcig(>0)***;

2.3.1 Model-Building Statements

The fundamental idea behind the sequential regression approach is that the imputation for every variable should be conditional on all other variables as predictors (unless listed under **DROP** or **TRANSFER** statements). There are practical circumstances where this may not be possible. The following two commands are useful to select the predictors based on their predictive power.

MAXPRED *number;* OR **MAXPRED *varlist (number)* ;**
Specifies the maximum number of predictor variables to be included as predictors in the regression model. A step-wise regression procedure is used to select the best predictors subject to the maximum number. Setting MAXPRED to a small number of predictors will greatly reduce the computational time especially for a very large data sets but the imputations will not be fully conditional.
For example,

MAXPRED *5*;

will include the five best predictor variables for every regression model, the five making the largest contribution to the r-square (for linear regression models) and Nagelkerke coefficient of determination for other models.
You can also restrict the number of predictors for selected variables.

MAXPRED *Income (7) Educ (3)*;

will limit the number of predictors of Income to the seven largest contributors to the r-square, while the number of predictors of Educ are limited to the three largest contributors. For other variables, all variables will be used as predictors.

The second option to reduce the number of predictors is use of the minimum additional increase in r-square needed for a variable to be included as a predictor. For example,

MINRSQD *decimal*;

Specifies the minimum marginal r-squared (or generalized r-squared) of ***decimal*** to be included as a predictor. This can reduce computation time. A small decimal number like 0.005 would build very large regression models whereas 0.25 will include a smaller number of predictors in the regression models. If neither MAXPRED nor MINRSQD is set then no variable selection will be performed.

MAXLOGI *number*;

Specifies the maximum number of iterative algorithms to be performed in a logistic or multilogit regression model. The default is 50. This is useful if the Newton-Raphson algorithm used in computing the maximum likelihood estimates does not converge after 50 iterations. This applies to the convergence criterion for the logistic, polytomous and Poisson regression models. You can check whether you have such a non-convergence problem by inspecting the log file (e.g., mysetup.log).

MINCODI *decimal*;

Specifies the minimum proportional change in any regression coefficient to continue the logistic regression iteration process. This applies to the convergence criterion for the logistic, polytomous and Poisson regression models.

Sometimes one may want to include interaction terms as predictors in the model. These

are derived variables. There are two possible options. The first option is to construct the product terms as new variables in the data set and impute them just like any other variable. The product term will be set to missing if either variable is missing. The second option is to impute separately but use the product as the predictor in other regression models. The following options implement this approach.

INTERACT ***variable1*variable2***;

This keyword enables the user to specify interaction terms to be included in the imputation regression model.

For example, a specification

INTERACT ***Income*Income, Age*Race***;

will result in including a square term for Income and an interaction term of Age and Race in the imputation model for all the variables in the data set (except for the variables in the particular interaction term).

OFFSETS ***count variables (offset variable)*** ;

This statement is used to specify an offsets variable when fitting a Poisson regression model. For example,

OFFSETS ***Injuries(Years)***;

will fit a model predicting the number for injuries occurring per year.

Finally, the command,

DIAGNOSE ***variables***/[**all**];

produces imputation diagnostic plots for all the listed variables. This will produce a series of imputation plots used to evaluate the imputation process. For more details about these plots see Bondarenko and Raghunathan (2016). By default, it will produce a set of recommended set of plots and numerical summaries. The optional command "**all**"will produce all the plots generated as a part of the program. Like the "**all**" feature in the **PRINT** command described in Section 2.3.2, the number of output graphs will be voluminous.

2.3.2 Other Commands

ITERATIONS ***number***;

Specifies the number of cycles that the imputation program should iterate for each variable and imputation. You can specify any number greater than or equal to 2. Current investigations show that about 10 cycles are sufficient for most imputations. You may want to experiment with several values and check the differences in the resulting analysis.

MULTIPLES *number*;

Indicates the number of imputations to be performed. By default only a single imputation is generated. Multiples and iterations determine p, the total number of cycles for regression model fitting for each variable. If 5 multiples and 10 iterations were specified then a total of 50 cycles will be performed. After every 10^{th} cycle an imputed data set will be created.

BY *varlist*;

This command can be used to perform imputations separately for the distinct combination of values of the variables in the ***varlist***. For example, if the variable race is coded as White/non-White and the variable gender is coded as Man/Woman then **BY *race gender*;** will create 4 subgroups and separately impute missing values in all other variables in each subgroup. No missing values in variables in the ***varlist*** are allowed.

PERTURB *keyword*;

The keyword **PERTURB** followed by a ***keyword*** (COEF/SIR) allows the user to control perturbations of imputed values. By default, the IMPUTE module will perturb model coefficients using a multivariate normal approximation of the posterior distribution of the parameters in the regression model and the predicted values using the appropriate regression model conditional on the perturbed coefficients. This is equivalent to using the COEF instruction. SIR uses the Sampling-Importance-Resampling algorithm to generate coefficients from the actual posterior distribution of parameters in the logistic, polytomous and Poisson regression models (See Rubin 1987a, Raghunathan and Rubin 1988, Raghunathan 1994, Gelman, et. al 1995). This is appropriate in situations where normal approximation to the posterior distribution is not appropriate. One example of this situation is a logistic regression with a low prevalence of the outcome variable (say, less than 1% or 2%).

One should be able to reproduce the imputed data sets at a later time. The **SEED** option is useful the generate the same random number sequence and, hence, regenerate the same set of imputed values.

SEED *number*;

Specifies a seed for the random draws from the posterior predictive distribution where ***number*** should be greater than zero. A zero seed will result in no perturbations in the regression coefficients or in the predicted values. If the **SEED** keyword is missing from the setup file then the seed will be determined by your computer's internal clock. However, you may not be able to recreate the imputed data set at a later date. For replication of results at a later date, this option must be used and the seed number should be archived.

NOBS *number*;

Specifies the number of observations to be used in the analysis. By default all observations in the data set will be used. You might use NOBS to subset a large data set while testing your setup file.

PRINT *instruction*;

Indicates the printout desired. The options are STANDARD, DETAILS, COEF, and ALL. For the IMPUTE procedure, the STANDARD and DETAILS keywords instruct *IVEware* to print the number and distribution of observed values, imputed values, and combined observed and imputed values for each variable. If the keyword COEF is present, then IMPUTE will also print the unperturbed and perturbed coefficients for each iteration of each multiple imputation. When the ALL keyword is used, in addition to the above, the coefficient covariance matrix for each iteration of each multiple imputation is also printed. IMPUTE also prints a list of the variables used in the imputation model with columns indicating the number of observed cases and the number of imputed cases for each of the variables.

The output from IMPUTE has a column labeled "double counted," and is useful for diagnostic purposes. This entry should be zero. A non-zero entry indicates the actual observations in the data set do not satisfy the restriction specified in **RESTRICT** statement. This has caused the program to count it twice (once satisfying the restriction and once more as not satisfying the restriction). The IMPUTE command, therefore, changes the observed value of a restricted variable according the restriction rule (zero for continuous variables, one higher than the highest observed code for categorical variables; see RESTRICT above) before proceeding with the imputation. In such situations, the data should be checked for consistency with respect to the specified restriction. For example, if the variable SMOKE, indicating whether or not a respondent smokes, is missing and the variable YRSMK, indicating the number of years the respondent has smoked, is observed (say, 10), then logically the respondent should be classified as a smoker. If, however, the value for SMOKE is missing in the data set, this creates inconsistency. The IMPUTE program changes SMOKE to smoker before proceeding with the imputation but then alerts the user to the problem in the data set. The user should correct the data (either 10 for YRSSMK is an error or setting SMOKE to missing is an error) and re-run the imputation. Since the restrictions can be complex, it is possible that for some subjects there could be no resolution. The user should, therefore, be made aware of the problems.

TITLE *text* \n *text*;

Indicates the title(s) to be printed at the top of each page of the printout. A **\n** indicates that the text that follows should be printed on the next line. For example,

TITLE *This is the title on the first line* \n *This is the title on the second line*;

2.4 PUTDATA

The IMPUTE module outputs a single data set, the one specified on the DATAOUT statement of your setup file. If you have requested more than one imputation with the keyword MULTIPLE and have included the keyword ALL in the DATAOUT statement the imputa-

tions are concatenated in the single output file. The imputations can be distinguished by the system variable "_ MULT_". If you request more than one imputation with the keyword MULTIPLE and have not included the keyword ALL in DATAOUT statement only the first imputation will be included in the output file. The additional imputations are stored in an internal file and can be retrieved by submitting the **PUTDATA** statement. For example, suppose that

```
<impute name="myfile">
datain mydata;
dataout myoutdata1;
 /* Other Impute commands are here */
 multiples 5;
 run;
</impute>
```

is the command file executed for creating multiple imputations. The following code uses **PUTDATA** to extract the remaining 4 data sets. These data sets are now available for further analysis simply by calling them into other commands.

```
/* extract the remaining four multiply imputed datasets */
<putdata name="myfile" mult="2" dataout="mydataout2" />
<putdata name="myfile" mult="3" dataout="mydataout3" />
<putdata name="myfile" mult="4" dataout="mydataout4" />
<putdata name="myfile" mult="5" dataout="mydataout5" />
```

Chapter 3

BBDESIGN

3.1 Introduction

The **BBDESIGN** module implements the weighted finite population Bayesian Bootstrap approach to generate synthetic populations from complex survey data. The primary goal is to incorporate weighting, clustering and stratification in a nonparametric approach for generating the non-sampled portion of the population from the posterior predictive distribution, conditional on the observed data and the design information. BBDESIGN assumes a two stage stratified cluster sampling approach with unequal probability of sampling at either or both stages. This approach generates a Bayesian Boostrap sample of non-sampled clusters and then uses a weighted Poly Urn model to sample non-sampled elements within the sampled and non-sampled clusters in each stratum. The details about the procedure are described in Dong, Elliott and Raghunathan (2014a, 2014b) and Zhou, Elliott and Raghunathan(2015, 2016a, 2016b). Once several synthetic populations are generated, the population quantity can be computed from each synthetic population and these can be combined using simple rules to form single inference. If there are missing values, then the synthetic populations are also generated with missing values which can be multiply imputed using the **IMPUTE** module. The combining rules, which differ from standard missing data multiple imputation combining rules, are discussed in the above references and will be illustrated through examples in later chapters.

3.2 BBDESIGN Statements

DATAIN *filename*;
This required statement identifies the location and name of the input data set. For example, in the SAS environment, the ***filename*** can be expressed as "libname.sasdata". In other environments, read the data set and include the name of the data set in the ***filename***. For example,

DATAIN *Mylib1.Mydata*;

indicates that the SAS data file **Mydata** is located in the library **Mylib1.** **Mylib1** is the

name assigned to a directory with the SAS libname statement.

DATAOUT *outfile* ;

This statement identifies the location and name of the output data set containing the synthesized data. If more than one synthetic data set is generated, the output data set will be a concatenation of the multiple synthesized data sets. The system variable _IMPL_ , automatically added to the output file, can be used to distinguish each implicate.

Additional Statements Include:

IMPLICATE *number*; where ***number*** is the number of implicate data sets.

STRATUM *var*;

where ***var*** is the name of the variable in the data set defining the stratum.

CLUSTER *var*;

where ***var*** is the name of the variable in the data set defining the clusters within each stratum.

WEIGHT *var*;

where ***var*** is the name of the variable in the data set defining the unit level weight.

POPSIZE *number*;

where the ***number*** is the number of observations included in each synthetic population generated. The default is 10 times the original sample size.

CSAMPLES *number*;

where ***number*** is the number of Bayesian bootstrap samples to be drawn from the sampled clusters in each stratum. The default is 5.

WSAMPLES *number*;

where ***number*** is the number of times the weighted Polya Urn model to be used to generate replicates of non-sampled units to be added in each of the Bayesian Bootstrap sample of clusters. The default is 5.

The number "CSAMPLES" and "WSAMPLES" determines the number synthetic populations generated. The default is 25 (5×5, the product of the two default numbers). As in the case of any bootstrap based analysis, 250 to 500 synthetic populations may be needed to obtain reliable point and interval estimates.

ID *var*;

where ***var*** is the name of variable indicating a unique subject identifier. If this keyword is

absent an id variable called _OBS_ will be created in the output data set.

VAR ***varlist*****;**

where ***varlist*** is a list of variables to be transferred to the output data set (of synthetic populations). If this keyword is not specified, all variables will be transferred to the output data set.

PRINT ***options*****;**

can be used to print information used in the process for creating synthetic populations.

SEED ***number*****;**

where ***number*** used to initialize the random number sequence for obtaining the draws. This is an important feature to reproduce the same random number sequence at a future time point.

Chapter 4

DESCRIBE

4.1 Introduction

The DESCRIBE module estimates population means, proportions, subgroup differences, and contrasts and linear combinations of means and proportions. A Taylor Series Linearization approach is used to obtain variance estimates for data derived from complex sample designs. Multiple imputation analysis can be performed using the DESCRIBE module.

4.2 DESCRIBE Statements

4.2.1 Required or Standard Statements

DATAIN ***filename*****;**

This keyword identifies the location and name of the data set to be analyzed. See Section 1.4.1 for more information about specifying a filename using the libname statement in SAS or changing the working directory to match the location of the data set. To perform multiple imputation analysis, more than one SAS data set can follow the DATAIN keyword in the DESCRIBE module. When multiple data sets are specified, each is analyzed separately and the inferences–estimates and variances–are combined using the usual multiple imputation combining rules.

RUN;

This should be the last statement in the setup file.

4.2.2 Design Variables

The commands described in this section are relevant only for data from complex sample surveys with stratification, clustering or weighting.

STRATUM ***variable name*****;**

variable name is the name of the stratum variable for the data from a complex survey. No missing values are allowed for this variable. If the statement is missing then the sample is assumed to be non-stratified.

CLUSTER *variable name*;

variable name is the Primary Sampling Unit (PSU) or Sampling Error Computing Unit (SECU) variable for the data from a complex sample survey. No missing values are allowed for the cluster variable. If this statement is missing then the sample is assumed to be un-clustered.

WEIGHT *variable name*;

variable name is the survey weight variable. Survey weights are usually the product of selection, non-response adjustments and post-stratification weights. No missing values are allowed for the weight variable. If this statement is not included then the sample is assumed to be self-weighted.

MODEL *method;*

MODEL indicates the variance estimation method to be used. ***Mult*** (Default) is useful when there are multiple PSUs within a stratum, ***Pair*** employs the paired selection method, and ***Diff*** employs the successive differences method. You can specify different methods for each stratum. For example,

MODEL *Pair(15,16,17) Diff(20,21,27)*;

will use paired differences for strata 15, 16, 17, the successive differences for strata 20, 21,27, and ***Mult*** for the rest.

4.2.3 Analysis Statements

TABLE *variable list*;

This command will produce the weighted proportions and their standard errors for all levels of a variable(s) in the ***variable list***. Some examples are given below.

TABLE *Race*;

for the marginal distribution of the variable ***Race***. Cross-tabulations may be indicated with an asterisk, for example:

TABLE *Race*Gender*;

MEAN *variable list*;

Means, standard errors, and design effects are calculated for the list of variables listed under ***variable list***. For example,

MEAN *BMI Age*;

will compute the means of BMI and Age.

BY *list*;

The BY keyword is used in conjunction with the TABLE or MEAN keyword. The analyses will be performed for each level of the variable(s) specified in the BY statement. For instance,

TABLE *Race*;
BY *Gender*;

will produce the weighted proportion of each Race category for each of the two levels of Gender. If variable Agecat is age in 3 categories then

TABLE *Race*;
BY *Gender Agecat*;

will produce weighted proportions of each Race category for each of the six combinations of Gender and Agecat.

CONTRAST *specifications*;

CONTRAST is used in conjunction with the MEAN keyword to compare or estimate linear combinations of cell means (continuous) or proportions (binary variables). For example,

MEAN *Income*;
CONTRAST *Race*;

will produce all the pairwise comparisons of mean Income defined by Race. If Race has three categories then three pairwise comparisons will be produced. Another example is, **MEAN *Income*;**
CONTRAST *Race*Gender*;

will produce comparisons of Income means for all combinations of Race and Gender.

Linear combinations of means can be estimated using the contrast features. Consider,

MEAN *Income*;
CONTRAST *Race (0.5 0.5 -1)*;

will produce the estimate of the contrast $(\mu_1 + \mu_2)/2 - \mu_3$ of the means for three categories of Race. (If Race has more than three levels then the above statement will produce an error message). The statement,

CONTRAST *Race (0.333333 0.333333 0.333333)*;

will produce an (approximate, due to rounding) estimate of the mean $(\mu_1 + \mu_2 + \mu_3)/3$. Note that this is not technically a contrast. The contrast in *IVEware* can be viewed as a

combination of contrast and estimate features in SAS, for example. You can also specify complicated statements such as

MEAN *Income*;
CONTRAST *Race(-1 0 1)*Gender(-1 1)*;

for contrasting the race differences for one gender group with the race differences for the other. The contrast features can be useful in testing the significance of some pre-planned contrasts in an ANOVA setting.

4.2.4 Missing Data Handling

There are four possible options for handling missing data. Analyze previously multiply imputed data sets, perform multiple imputation analysis concurrently just for variables in the analysis, skip subjects with missing values (that is, perform available case analysis) or stop the analysis and exit.

For previously multiply imputed data sets, list all the data sets in the **DATAIN** statement and omit the **MDATA** command. Other options with only one data set in the **DATAIN** statement are given below.

MDATA *instruction*;

The keyword ***instruction*** options are (STOP/IMPUTE/SKIP). If **MDATA** is not included in your setup, cases with missing data will be excluded from your analysis. This is equivalent to using the SKIP instruction. When the instruction is STOP, the DESCRIBE module stops if missing data are encountered in any analysis variables. If the keyword is IMPUTE then the missing data will be imputed. All of the IMPUTE keywords can be used to specify the models for imputation process.

4.2.5 Other commands

NOBS *number*;

NOBS indicates the number of observations to be used in the analysis. By default, all observations in the data set will be used. Specification of NOBS to subset a large data set might be useful while testing the setup file.

PRINT *instruction*;

Indicates the printout desired. The options are STANDARD (default) and DETAILS. When a DESCRIBE procedure includes the IMPUTE missing-data option (see MDATA above) the DETAILS keyword instructs *IVEware* to print the number and distribution of observed values, imputed values, and combined observed and imputed values for each variable. If the DESCRIBE procedure includes multiple imputations, the DETAILS keyword instructs *IVEware* to print estimates and statistics for each imputed data set as well as combined estimates and statistics across the imputed data sets. The standard DESCRIBE printout does not include imputation results.

TITLE ***text \n text*****;**

Indicates the title(s) to be printed at the top of each page of the printout. A \n indicates that the text that follows should be printed on the next line. For example,

TITLE ***This is the title on the first line \n This is the title on the second line*****;**

Chapter 5

REGRESS

5.1 Introduction

The REGRESS module fits linear, logistic, Poisson, polytomous and proportional hazards regression models. All the keywords for the **DESCRIBE** models are also applicable here. This chapter provides additional commands relevant for performing a regression analysis. One main difference is that **DESCRIBE** uses the Taylor Series Linearization method for variance estimation but **REGRESS** uses the Jackknife Repeated Replication technique to estimate design-based variances (Kish and Frankel 1974).

5.2 REGRESS Statements

5.2.1 Models

DEPENDENT ***variable name*****;**

This statement specifies the name of the dependent variable in the regression model. Dependent variables are assumed to be continuous unless the CATEGORICAL keyword is included as described below.

PREDICTOR ***variable list*****;**

This specifies the right hand side of the regression model. Predictor variables are assumed to be continuous unless they are defined as CATEGORICAL as described below. Interaction terms can be specified by using the "*" notation. For example,

PREDICTOR ***Income Age Income*Age*****;**

LINK ***model*****;**

LINK defines the type of regression model to be fit. Specify ***Linear*** for fitting a multiple linear regression model, ***Logistic*** for fitting a logistic (binary) or generalized logistic (polytomous) regression model, ***Log*** for fitting a Poisson regression model for a count variable, ***Tobit*** for fitting a tobit model or ***Phreg*** for fitting Proportional Hazards model (Cox model).

CENSOR *variable name (number)*;

variable name is the censoring variable, and ***number*** is the code indicating censoring. If the number is omitted then, by default, 1 will be considered as the code indicating censored observation. The Censor statement is required if the **LINK** is specified as ***Phreg***. For example,

LINK *Phreg*;
DEPENDENT *Survivaltime*;
CENSOR *Died (0)*;

In this example, the outcome variable is ***Survivaltime*** and the censoring variable is ***Died*** where ***Died=0*** denotes censored observations.

CATEGORICAL *variable list*;

declares that the listed variables are to be treated as categorical. If a variable with k categories is listed on the CATEGORICAL and PREDICTOR statement then k-1 predictors (dummies) will be included in the regression model. The category with the highest code value will be the reference category. For logistic and multinomial logit models, the dependent variable must also be listed in the ***variable list***.

OFFSETS *count-variable(offset-variable)*;

This statement is used to specify an offsets variable when fitting a Poisson regression model. For example,

OFFSETS *Injuries(Years)*;

will fit a model predicting the number for injuries occurring per year.

ID *variable name*;

Specifies the variable to be used as the unique subject identifier. This allows for linking the PREDOUT file (see below) created by the REGRESS module to other files.

NOINTER;

This keyword will fit regression models without the intercept term.

ESTIMATES *label: specification*;

This is useful for estimating values of the dependent variable for a specific set of covariates or testing hypotheses involving the estimated regression coefficients. For example, suppose that the following regression model is fit:

$$Y = b_o + b_1 x1 + b_2 x2 + b_3 x3$$

and we are interested in predicting Y for $x1 = 1, x2 = 2$ and $x3 = 0$. We can obtain the predicted value and the 95% confidence interval by using the following statement:

ESTIMATES ***Mylabel : Intercept (1) x1(1) x2(2)***;

Several estimates can be requested by separating them with the symbol: "/ " .

5.2.2 Output files

The **REGRESS** module can be used to produce several plots and outputs for later processing. The following are the descriptions of these features.

PLOT ***filename***;

This keyword creates a series of diagnostic plots including residual, leverage, influence and normal probability plots. The plots will be stored in the ***filename*** specified after the **PLOT** keyword. The user can rely on the built-in graphics produced internally or use GNU Plot by downloading this package and including the path in the XML settings file, see Chapter 9 for examples.

PREDOUT ***filename***;

outputs a file containing the predicted values, their standard errors and 95% confidence intervals. If an ID statement is included in the setup, an ID variable is also included in the data set.

ESTOUT ***filename***;

Outputs a file containing estimates and their variances-covariances.

REPOUT ***filename***;

Outputs a file containing estimates for each replicate. Estimated regression coefficients are provided for each combination of STRATUM, CLUSTER and BY variable.

5.2.3 Design Variables

The design features can be specified using the commands **STRATUM, CLUSTER**, and **WEIGHT** as illustrated in the **DESCRIBE** chapter.

1. If the STRATUM, CLUSTER and WEIGHT variable are not specified, then a simple random sample analysis will be performed.

2. If a design based analysis involves only a WEIGHT variable and no STRATUM or CLUSTER variable, then a pseudo-stratification variable and a pseudo-cluster variable should be used. When using pseudo variables, all observations in the data set should have the same value for the pseudo STRATUM variable (e.g., 1), while each observation should have a unique value on the pseudo CLUSTER variable (e.g., observation ID number or SAS system variable _N_). The pseudo variables should be created in the data prior to performing the analysis. Example SAS data step code for creating a pseudo STRATUM variable and a pseudo CLUSTER variable:

```
LIBNAME MYLIB C:\MYINDIR;
 DATA MYLIB.MYDATA;
 SET MYLIB.MYDATA;
 PSEUD_STRAT=1;
 PSEUD_CLUST=_N_;
 RUN;
```

Note that the inclusion of pseudo variables will increase the time REGRESS needs for analysis.

TITLE ***text \n text*****;**

Indicates the title(s) to be printed at the top of each page of the printout. A \n indicates that the text that follows should be printed on the next line. For example,

TITLE ***This is the title on the first line \n This is the title on the second line*****;**

Chapter 6

SASMOD

6.1 Introduction

SASMOD is a SAS macro that provides a framework for performing analysis based upon a collection of SAS procedures. SASMOD includes the ability to perform multiple imputation and/or analysis incorporating complex survey design features. Currently, the following SAS procedures are available: CALIS, CATMOD, GENMOD, LIFEREG, MIXED, NLIN, PHREG, and PROBIT. This particular macro executes user-specified SAS procedure commands for each Jackknife Repeated Replication replicate and then combines the results to compute the proper complex design sampling variance estimate. If multiple data sets are specified as inputs, it performs the analysis for each data set separately and then combines inferences using multiple imputation combining rules.

6.2 SASMOD Statements

The setup is similar to other modules except that multiple imputation must be performed prior to invoking SASMOD. The multiple data sets can be specified in the DATAIN statement. The allowable keywords are DATAIN, BY, STRATUM, CLUSTER, WEIGHT, and TITLE.

SAS commands and optional statements can be used as appropriate for the procedure. However, do not use statements that might lead to more than one model or different models in different replicates or multiples. For example, more than one model statement or specifying a stepwise model is not permitted. Examples of use of SASMOD are presented in later chapters.

Chapter 7

SYNTHESIZE

7.1 Introduction

SYNTHESIZE uses the multivariate sequential regression approach to create full or partial synthetic data sets to limit statistical disclosure (See Raghunathan, Reiter and Rubin (2003), Reiter (2002) and Little, Liu and Raghunathan (2004) for more details). All item missing values will also be imputed when creating synthetic data sets. However, DESCRIBE, REGRESS and SASMOD modules cannot be used to analyze synthetic data sets as they DO NOT implement the appropriate combining rules. See examples in later chapters for demonstration of correct combining rules.

Except for the command IMPLICATES which specifies the number synthesized data sets to be generated, SYNTHESIZE commands are the same as those for IMPUTE.

7.2 SYNTHESIZE Statements

7.2.1 Variable Types

SYNTHESIZE requires that all variables be defined by type. Six types of variables are recognized by the IMPUTE module: continuous, categorical, count, mixed, transfer and drop. If no variable types are specified, all variables will be assumed to be continuous. Variable types should be declared before any BOUNDS, INTERACT, or RESTRICT statements.

CONTINUOUS variable list;

Variables declared as CONTINUOUS may take on any value on a continuum. For example, income is a continuous variable. A normal linear regression model is used to synthesize the missing values in these variables. You may want to transform the variable to achieve normality and then use SYNTHESIZE on the transformed scale. After imputation you may re-transform the variable back to its original form.

CATEGORICAL variable list;

CATEGORICAL variables have values that represent discrete values. For example, gender is a categorical variable. A logistic or generalized logistic model is used to impute missing categorical values.

MIXED variable list;

Variables declared as MIXED are both categorical and continuous. In a mixed variable, a value of zero is treated as a discrete category, while values greater than zero are considered continuous. Alcohol consumption is an example of a mixed variable. A two stage model is use to impute the missing values. First, a logistic regression model is used to impute zero versus non-zero status. Then, conditional on imputing a non-zero status, a normal linear regression model is used to impute non-zero values.

COUNT variable list;

COUNT variables have non-negative integer values. A Poisson regression model is used to impute the missing values. The number of annual doctor visits is an example of a COUNT variable.

DROP variable list;

Variables listed after the DROP keyword will be excluded from the imputation procedure and will not appear in the imputed data set.

TRANSFER variable list;

Variables listed after the TRANSFER keyword are carried over to the imputed data set, but are not imputed nor used as predictors in the imputation model. Transfer variables, however, can be used in the RESTRICT and BOUNDS statements (see below). ID is an example of a variable that you may want to treat as a transfer variable.

DEFAULT variable type;

Variable type can be Continuous, Categorical, Count, Mixed, Transfer or Drop. This keyword declares that by default all the variables in the data set should be treated as the selected variable type. The most efficient use of the DEFAULT statement is to declare the most numerous variable type in your data set as the default type, eliminating the need to type a long list of variables.

Optional Statements

RESTRICT variable(logical expression);

This command is used to restrict the imputation of a variable to those observations that satisfy the logical expression. For instance, suppose that the variable Yrssmoke indicates the number of years an individual smoked, and the variable Smoke takes the value 1 for a current smoker, 2 for a former smoker or 0 for someone who never smoked. Then the declaration:

RESTRICT Yrssmoke(Smoke=1,2);

will impute Yrssmoke values only for current and former smokers. It will automatically set Yrssmoke equal to 0 for those who never smoked. Restrictions on more than one variable may be combined as follows:

RESTRICT Yrssmoke(Smoke=1,2) Births(Gender=2) Income(Employed=1);

When the restriction is not met, the value of the restricted variable will be set to zero for continuous variables and one higher than the highest observed code for categorical variables.

BOUNDS variable (logical expression);

This keyword is useful for restricting the range of values to be imputed for a variable. For example,

BOUNDS Yrssmoke (> 0, <= Age-12);

will ensure that the imputed values for Yrssmoke are between > 0 and the individuals Age minus 12. Smoking is assumed not to begin before the age of 12. Again, as in the RESTRICT statement more than one variable can be included in the BOUNDS statement. For example,

BOUNDS Yrssmoke (>0, <= Age-12) Numcig(>0);

Model-Building Statements The following commands are useful in the specification of the imputation model.

INTERACT variable*variable;

This keyword enables the users to specify interaction terms to be include in the imputation regression model.

INTERACT Income*Income, Age*Race;

In this example, the imputation model for all the variables will include a square term for Income and an interaction term of Age and Race.

Options for Stepwise Regression

MAXPRED number; MAXPRED varlist2 (number);

Specifies the maximum number of predictor variables to be included as predictors in the regression model. A step-wise regression procedure is used to select the best predictors subject to the maximum number. Setting MAXPRED to a small number of predictors will greatly reduce the computational time especially for a very large data sets but the imputations will not be fully conditional.

For example,

MAXPRED 5;

will include the five best predictor variables, that is, the five making the largest contribution to the R-squared statistic. You can also restrict the number of predictors for selected variables.
For example,

MAXPRED Income (7) Educ (3);

will limit the number of predictors of Income to the seven largest contributors to the R-square, while the number of predictors of the variable Educ are limited to the three largest contributors. For other variables, all variables will be used as predictors.

MINRSQD decimal;

Specifies the minimum marginal R-square for a stepwise regression, that is , minimum initial marginal R-square for a logistic regression, and minimum initial R-square for any model being predicted by a polytomous regression. This option can reduce computation time. A small decimal number like 0.005 would build very large regression models whereas 0.25 will include a smaller number of predictors in the regression models. If neither MAXPRED nor MINRSQD is set then no stepwise regression will be performed.

MINRSQD 0.01;

In the above example, only variables with minimum additional R-square of 0.01 or higher will be included as predictors.

MAXLOGI number;

Specifies the maximum number of iterative algorithms to be performed in a logistic or multi-logit regression model. The default is 50. This is useful if the Newton-Raphson algorithm used in producing maximum likelihood estimates does not converge after 50 iterations. This applies to the convergence criterion for the logistic, polytomous and Poisson regression models. You can check whether you have such a non-convergence problem by inspecting the log file (e.g., mysetup.log).

MINCODI decimal;

Specifies the minimum proportional change in any regression coefficient to continue the logistic regression iteration process. This applies to the convergence criterion for the logistic, polytomous and Poisson regression models.

ITERATIONS number;

Specifies the number of cycles you would like the imputation program to carry out for each variable and implicate/multiple. You can specify any number greater than or equal to 2. Current investigations show that about 10 cycles are sufficient for most imputations. You may want to experiment with several values and check the differences in the resulting analysis.

IMPLICATES number;

Indicates the number of synthesized data sets to be created. By default, only a single synthesized data set is generated.

MULTIPLES number;

You can perform imputation within the SYNTHESIZE procedure. The value of the Multiples option indicates the number of imputations to be performed. By default, only a single imputation is generated. Note that IMPUTE is processed prior to SYNTHESIZE. For each multiple, a set of synthesized data sets are created based on the number of implicates specified. For example, if 2 multiples and 5 implicates are specified then 10 synthesized data sets will be created; five for multiple 1 and 5 for multiple 2.

PERTURB instruction;

The keyword PERTURB followed by an instruction of COEF or SIR allows the user to control perturbations of imputed values. By default, the IMPUTE module will perturb model coefficients using a multivariate normal approximation of the posterior distribution and the predicted values using the appropriate regression model conditional on the perturbed coefficients. This is equivalent to using the COEF instruction. SIR uses the Sampling-Importance-Resampling algorithm to generate coefficients from the actual posterior distribution of parameters in the logistic, polytomous or Poisson regression models (See Rubin 1987a, Raghunathan and Rubin 1988, Raghunathan 1994, Gelman, et. al 1995). This is appropriate in situations where normal approximation to the posterior distribution is not appropriate.

SEED number;

Specifies a seed for the random draws from the posterior predictive distribution. This num-

ber should be greater than zero. A zero seed will result in no perturbations of the predicted values or the regression coefficients. If the SEED keyword is missing from the setup file, then the seed will be determined by your computer's internal clock.

NOBS number;

The NOBS option indicates the number of observations to be used in the analysis. By default, all observations in the data set will be used. You might use NOBS to subset a large data set while testing your setup file.

OFFSETS count variables (offset variable) ;

This statement is used to specify an offsets variable when fitting a Poisson regression model. For example,

OFFSETS Injuries(Years);

will fit a model predicting the number for injuries occurring per year.

PRINT instruction;

Indicates the printout desired. The options are STANDARD, DETAILS, COEF, and ALL. For the STANDARD and DETAILS keywords instruct *IVEware* to print the number and distribution of observed values, imputed/synthesized values, and combined observed and imputed/synthesized values for each variable. The keyword COEF instructs additional printing of the unperturbed and perturbed coefficients for each iteration of each imputation/synthesization. When the ALL keyword is used, in addition to the above, the coefficient covariance matrix for each iteration of each multiple imputation is also printed.

A list of the variables used in the imputation/synthesizatio model is also printed with columns indicating the number of observed cases and the number of imputed cases for each of the variables. The third column of the variable list, labeled double counted, is to be used for diagnostic purposes. This entry should be zero. **A non-zero entry indicates that the imputed value of a restricting variable has caused the observed value of a restricted variable to be set to the restricted value (zero for continuous variables, one higher than the highest observed code for categorical variables; see RESTRICT above).** This usually indicates problems with the restriction or an inconsistency in the observed data. In either case, you should run a data step before the imputation to check the appropriateness of the restriction or correct the data inconsistency.

For example, if the variable SMOKE, indicating whether or not a respondent smokes, is missing and the variable YRSMK, indicating the number of years the respondent has smoked, is observed, then logically the respondent should be classified as a smoker. If SMOKE is not given a value indicating the respondent is a smoker in a SAS data step prior to imputation, the missing value could possibly be imputed to a nonsmoker value, causing the IMPUTE/SYNTHESIZE command to change the observed value for YRSMK to zero.

Chapter 8

COMBINE

8.1 Introduction

The Combine data procedure allows the user to concatenate (stack) multiple data sets. The data sets need not contain the same variables. Variables with a shared name will be treated as the same variable in the combined data set. It is important that they have the same value structure. If a variable does not appear in one of the data sets, it is treated as missing data in the combined data set. The user may want to impute the missing values prior to analyzing the combined data set using IMPUTE.

For example, suppose that data set 1 provides variables X and Y, data set 2 provides variables X and Z and data set 3 provides variables Y and Z. COMBINE can be used to concatenate the three data sets and multiply impute the missing values of X, Y and Z to create complete data on all three variables. The multiply imputed combined data sets can then be analyzed using the DESCRIBE, REGRESS or SASMOD modules.

8.2 COMBINE Statements

DATAIN ***filename*****;**

This keyword identifies the location and name of the data set to be analyzed. See Section 1.4.1 for more information about specifying a filename using the libname statement in SAS or changing the working directory to match the location of the data set. To combine multiple data sets, more than one data set can follow the DATAIN keyword.

DATAOUT ***filename*****;**

DATAOUT is used to name an output data set produced by the COMBINE procedure.

RUN;

This should be the last statement in the setup file.

Chapter 9

IVEware and SAS

9.1 Introduction

Chapter 9 presents examples of common imputation and analytic tasks such as multiple imputation of missing data using IMPUTE, use of BBDESIGN to create a complex sample population data set, descriptive analysis of imputed data using DESCRIBE, linear regression analysis with REGRESS, use of SASMOD (with SAS only) for categorical modeling, use of SYNTHESIZE to create data sets that limit statistical disclosure, and use of COMBINE to combine data from multiple sources.

All examples are run using the SRCShell editor with SAS method, that is, with code submitted from an XML editor and enclosed with <sas> and </sas> tags to run *IVEware* from within SAS.

National Comorbidity Survey-Replication data is used in many examples. The NCS-R data is derived from a complex sample survey and thus, each example also demonstrates how to correctly account for the design features. For more information on this data set, see www.hcp.med.harvard.edu/ncs.

Other data sets used include NHANES 2011-2012 data, Health and Retirement Survey data from 2006, 2008, 2010, and 2012, and Primary Cardiac Arrest data. For information on data sets used in this chapter, see Raghunathan, Berglund and Solenberger (2017) or project specific sites.

Each example includes a short description of the purpose of the example, and the code used. The execution method used in these examples can be easily modified for use with other software such as Stata, R, and SPSS (see Chapter 10 for selected examples of this approach).

9.2 IMPUTE Examples

This example uses NCS_R data and begins with use of SAS PROC MI with the NIMPUTE=0 option to produce a missing data pattern grid. The grid allows easy visualization of the missing data pattern, amount of missing data per variable and group means for each group in the data set. This command calls on SAS directly and is not part of *IVEware*.

The second part of the example demonstrates use of IMPUTE and PUTDATA commands to first impute missing data and create M=5 multiples or complete data sets. These imputed

data sets are then extracted from a concatenated MI data set using PUTDATA and are available for subsequent analysis.

The 5 imputed data sets extracted are then used as inputs for a number of descriptive and regression analyses to come in later parts of this chapter. In those later examples, we will be analyzing the imputed data sets and using *IVEware* built-in combining rules appropriate for analysis of multiply imputed data sets as well correct variance estimation for complex sample data.

Syntax

```
<sas name="Impute Example">
libname d 'P:\IVEware 0.3 documentation 2016\SAS XML Examples';

/* check missing data pattern using SAS PROC MI */
title "Missing Data Pattern from SAS PROC MI" ;
proc mi data=d.ncsr_ex1 nimpute=0 ;
run ;

/* Multiple Imputation using %impute */
<impute name="MI Using IVEware">
title Multiple Imputation Using %impute ;
datain d.ncsr_ex1 ;
dataout d.impute_mult1;
default categorical;
continuous bmi intwage ncsrwtsh sestrat ;
transfer caseid ;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

/* Extract remaining 4 data sets */

<putdata name="MI Using IVEware" mult="2" dataout="d.impute_mult2" />
<putdata name="MI Using IVEware" mult="3" dataout="d.impute_mult3" />
<putdata name="MI Using IVEware" mult="4" dataout="d.impute_mult4" />
<putdata name="MI Using IVEware" mult="5" dataout="d.impute_mult5" />

</sas>
```

Selected Output

```
Multiple Imputation Using %impute

Imputation 1

Variable          Observed           Imputed    Double counted
DSM_GAD               9282                 0                 0
REGION                9282                 0                 0
MAR3CAT               9282                 0                 0
ED4CAT                9282                 0                 0
NCSRWTSH              9282                 0                 0
SEX                   9282                 0                 0
SESTRAT               9282                 0                 0
SECLUSTR              9282                 0                 0
bmi                   8285               997                 0
mde                   9112               170                 0
sexf                  9282                 0                 0
sexm                  9282                 0                 0
ald                   9282                 0                 0
racecat               9282                 0                 0
ag4cat                9282                 0                 0
intwage               9282                 0                 0
```

9.2.1 IMPUTE Example with ABB Option

This example uses the ABB option with the IMPUTE command, with the PCA data set. This option permits use of an Approximate Bayesian Bootstrap approach for the imputation model/variable called REDTOT, representing red blood cell total counts.

```
/* Impute Example with ABB using PCA and Omega 3 Fatty Acids Data */
<sas name="IMPUTE with ABB Example">

/* Set libnames */
libname d1 'P:\IVEware_and_MI_Applications_Book\DataSets\PCA and Omega 3 Fatty Acids Data' ;
libname dout 'P:\IVEware 0.3 documentation 2016\SAS XML Examples';

<impute name="Impute with ABB Option Using PCA and Omega3 Data">
datain d1.test;
continuous AGE NUMCIG YRSSMOKE FATINDEX DHA_EPA REDTOT  WGTKG TOTLKCAL HGTCM ;
categorical CASECNT GENDER RACE3 HYPER DIAB SMOKE FAMMI EDUSUBJ3 CHOLESTH ;
mixed CAFFTOT ALCOHOL3 ;
transfer STUDYID ;
/* Declare ABB for REDTOT, assume non-normal residuals */
ABB redtot ;
restrict NUMCIG(smoke=2,3) YRSSMOKE(smoke=2,3) ;
bounds NUMCIG(>0) YRSSMOKE(>0, <age-12) DHA_EPA(>0)  REDTOT(>0)  CAFFTOT(>0)  TOTLKCAL(>0)  ALCOHOL3(>0);
ITERATIONS 3;
MULTIPLES 5;
SEED 2001;
DATAOUT dout.impute_ABB all ;
run;
</impute>

/* Examine Output Data Set*/
proc means data=dout.impute_ABB ;
class _mult_ ;
var redtot ;
run ;

</sas>
```

9.2.2 IMPUTE Example with GH Option

Section 9.2.2 demonstrates use of the GH (Tukey's GH) option for the imputation model/REDTOT variable, again using the PCA data set. Like the ABB method, this method can be used to address situations where linear regression is not appropriate.

```
/* Impute Example with GH using PCA and Omega 3 Fatty Acids Data */
<sas name="IMPUTE with GH Example">

/* Set libnames */
libname d1 'P:\IVEware_and_MI_Applications_Book\DataSets\PCA and Omega 3 Fatty Acids Data' ;
libname dout 'P:\IVEware 0.3 documentation 2016\SAS XML Examples';

<impute name="Impute with GH Option Using PCA and Omega3 Data">
datain d1.test;
continuous AGE NUMCIG YRSSMOKE FATINDEX DHA_EPA  REDTOT WGTKG TOTLKCAL HGTCM ;
categorical CASECNT GENDER RACE3 HYPER DIAB SMOKE FAMMI EDUSUBJ3 CHOLESTH ;
mixed CAFFTOT ALCOHOL3 ;
GH redtot ;
transfer STUDYID ;
restrict NUMCIG(smoke=2,3) YRSSMOKE(smoke=2,3) ;
bounds NUMCIG(>0) YRSSMOKE(>0, <age-12) DHA_EPA(>0)  REDTOT(>0)  CAFFTOT(>0)  TOTLKCAL(>0)  ALCOHOL3(>0);
ITERATIONS 3;
MULTIPLES 5;
```

```
SEED 2001;
DATAOUT dout.impute_gh all ;
run;
</impute>

/* Examine Output Data Set*/
proc means data=dout.impute_gh ;
class _mult_ ;
var redtot ;
run ;
</sas>
```

9.3 BBDESIGN Examples

Section 9.3 uses NHANES 2011-2012 adult data to demonstrate examples of the BBDESIGN command.

```
<sas name="BBDesign Example">

/* BBDesign Example, Uses NHANES 2011-2012 DATA with
 BBdesign and Impute */
libname d 'P:\IVEware_and_MI_Applications_Book\Chapter12Simulations
\Examples\Revised BBDESIGN 12feb2018';

* gather NHANES data where age >=18 and MEC weight > 0
(participated in MEC examination) ;
data nhanes1112_sub_20jan2017  ;
 set d.nhanes1112_sub_4nov2015 ;
 if age >=18 and wtmec2yr > 0 ;
 drop marcat bpxsy1 - bpxsy4 bp_cat pre_hibp  bpxdi1 - bpxdi4
 dmdmartl irregular ;
run ;

proc means nolabels n nmiss mean min max ;
weight wtmec2yr ;
run ;

/* Use BBDesign command to prepare data set using complex
sample design variables and MEC weight:
 25 implicate data sets are generated:
 5 Bootstrap sample of clusters
 5 FPBB using Weighted Polya posterior within each
  bootstrap sample
*/

<bbdesign name="BBdesign">
 datain nhanes1112_sub_20jan2017 ;
 dataout d.bbdesignout ;
 stratum sdmvstra ;
 cluster sdmvpsu ;
 weight wtmec2yr ;
 csamples 5 ;
 wsamples 5 ;
 seed 2001;
run;
</bbdesign>

/* Confirm that there are 10 (sample inflation factor)*5,615
 (original n) *25 (implicates)= 1,403,750 */
proc freq data=d.bbdesignout ;
 tables _impl_ ;
run ;
```

Next, missing data is addressed via use of the IMPUTE command with M=5. Since the data set has already been prepared to represent the population of interest, we impute missing data values for total cholesterol, family income/poverty ratio, BMI, and education within each implicate. Once this is complete, data analysis can be done using simple random sample assumptions, that is, without use of complex sample design variables or probability weights.

```
/* impute missing data within each of 25 implicates
 using M=5 and 5 iterations */
<impute name="Impute_BBDesign"> ;
 datain d.bbdesignout ;
 dataout d.imputed_samples all ;
 default continuous ;
 transfer ridstatr seqn ag1829 ag3044 ag4559 ag60 mex
 othhis white black other _impl_ _obs_ ;
 categorical riagendr ridreth1 edcat ;
 bounds indfmpir (>= 0, <=5) bmxbmi (>=13, <=80)
 lbxtc (>=59, <=523) ;
 by _impl_;
 seed 2016 ;
 multiples 5;
 iterations 5;
 run ;
</impute> ;
```

Two analyses are now demonstrated; one using a linear regression model and another using logistic regression with combining for both models. This is needed because the default combining rules implemented in REGRESS and PROC MIANALYZE are different from the FPBB rules, see Raghunathan (2016) or Zhou, Elliot and Raghunathan (2016b) for details. The linear regression example uses total cholesterol predicted by gender, BMI, and the ratio of family income to poverty thresholds. For the logistic regression example, a binary outcome of obesity status (coded 1 if BMI >=30, and 0 otherwise) is predicted by age, gender and the income/poverty ratio.

```
/* Prepare the imputed synthetic populations for analysis */
data synthpops;
 set d.imputed_samples;
/*
Create 3 indices S, B, L using the fact that wsamples=5 in
the BBDESIGN code above and given the relationships:
****************************************
indexL=_mult_;
indexS=floor((_impl_-1)/wsamples)+1;
indexB=_impl_-(indexS-1)*wsamples;
****************************************
*/

 indexL=_mult_;
 indexS=floor((_impl_-1)/5)+1;
 indexB=_impl_-(indexS-1)*5;
run ;

/* Save imputed data */
data d.imputed_synthpops ;
 set synthpops ;
 male=(riagendr=1) ;
```

```
run ;

proc sort data=d.imputed_synthpops;
 by indexS indexB indexL;
run ;

/* Estimate of the population mean of lbxtc and its
variance involves 2 steps:
Step 1: Average over IndexB and IndexL for each level
of IndexS */

proc means data=d.imputed_synthpops noprint  mean;
var lbxtc;
by indexS;
output out=step1 mean=lbxtcbar;
run ;

/* Step 2: Compute the mean and variance across the
 S synthetic populations */

proc means data=step1 mean var ;
var lbxtcbar;
run ;

/* Linear Regression analysis using PROC REG with
imputed synthetic populations*/
proc reg data=d.imputed_synthpops;
 by indexS indexB indexL;
 model lbxtc = bmxbmi male indfmpir ;
 ods output parameterestimates=outparms ;
run ;

title "Print Out from Linear Regression" ;
proc print data=outparms ;
run ;

/* prepare combined estimates and variance using
 two data steps*/
proc means data=outparms mean ;
 var estimate ;
 where variable ='Intercept' ;
 by indexs ;
 output out=step1_0 mean=bobar ;
run ;

proc print data=step1_0 ;
run ;

proc means data=outparms mean ;
 var estimate ;
 where variable ='BMXBMI' ;
 by indexs ;
 output out=step1_1 mean=b1bar ;
run ;

proc print data=step1_1 ;
run ;

proc means data=outparms mean ;
 var estimate ;
 where variable ='male' ;
 by indexs ;
 output out=step1_2 mean=b2bar ;
run ;
proc print data=step1_2 ;
run ;

proc means data=outparms mean ;
```

```
 var estimate ;
 where variable ='INDFMPIR' ;
 by indexs ;
 output out=step1_3 mean=b3bar ;
run ;

proc print data=step1_3 ;
run ;

* Merge temp data sets into 1 for combining ;
data step1_all ;
 merge step1_0 step1_1 step1_2 step1_3 ;
 by indexs ;
run ;

proc print data=step1_all ;
run ;

* use merged data above for final step ;
proc means data=step1_all mean var noprint;
 var bobar b1bar b2bar b3bar;
 output out=step2 mean=intercept bmxbmi male indfmpir
  var=vintercept vbmxbmi vmale vindfmpir;
run ;

proc print data=step2 ;
run ;

/* Combine results for parameter estimates from above */
data combine_linear  ;
 set step2;

 df=5-1 ; *Min(S-1,C-H);
 tvalue=quantile('T',0.975,df);

 /* Create arrays for estimate variance se and
 lower/upper CI */
 array estimate[4] intercept bmxbmi male indfmpir;
 array variance[4] vintercept vbmxbmi vmale vindfmpir;
 array se[4] se_intercept se_bmxbmi se_male se_indfmpir;
 array lower95[4] l95_intercept l95_bmxbmi
 l95_male l95_indfmpir;
 array upper95[4] u95_intercept u95_bmxbmi
 u95_male u95_indfmpir;
 do i=1 to 4;
  se[i]=sqrt((1+1/5)*variance[i]);  * Note that
  denominator must match the number used for
   "csamples" in the code ;
  lower95[i]=estimate[i]-tvalue*se[i];
  upper95[i]=estimate[i]+tvalue*se[i];
 end;
 drop i;
run ;

options nodate nonumber ;
proc print data=combine_linear ;
 title "Combined Estimates, SE, Lower and Upper CI from
 Linear Regression" ;
 var    intercept se_intercept l95_intercept
  u95_intercept
  bmxbmi se_bmxbmi l95_bmxbmi u95_bmxbmi
        male se_male l95_male u95_male  indfmpir se_indfmpir
         l95_indfmpir u95_indfmpir
        ;
run ;

*******************************************************;
```

```
/* Logistic Regression analysis using PROC LOGISTIC,
outcome is obese predicted by male, family income to poverty
  and age categories*/

data imputed_synthpops2 ;
 set d.imputed_synthpops ;
 if bmxbmi >=30 then obese = 1 ; else obese=0 ;
run ;

/*predict probability of being obese by gender and age in
 categories and family income to poverty ratio */
proc logistic data=imputed_synthpops2;
 by indexS indexB indexL;
 model obese (event='1') = male indfmpir ag3044 ag4559 ag60 ;
 ods output parameterestimates=outparms_log ;
run ;

proc print data=outparms_log ;
run ;

/* prepare combined estimates and variance using
 two data steps*/
/* Create separate mean by IndexS for each variable */

proc means data=outparms_log mean ;
 var estimate ;
 where variable ='Intercept' ;
 by indexs ;
 output out=step1_0 mean=bobar ;
run ;

proc print data=step1_0 ;
run ;

proc means data=outparms_log mean ;
 var estimate ;
 where variable ='male' ;
 by indexs ;
 output out=step1_1 mean=b1bar ;
run ;

proc print data=step1_1 ;
run ;

proc means data=outparms_log mean ;
 var estimate ;
 where variable ='INDFMPIR' ;
 by indexs ;
 output out=step1_2 mean=b2bar ;
run ;
proc print data=step1_2 ;
run ;

proc means data=outparms_log mean ;
 var estimate ;
 where variable ='ag3044' ;
 by indexs ;
 output out=step1_3 mean=b3bar ;
run ;

proc print data=step1_3 ;
run ;

proc means data=outparms_log mean ;
 var estimate ;
 where variable ='ag4559' ;
 by indexs ;
 output out=step1_4 mean=b4bar ;
```

```
run ;

proc print data=step1_4 ;
run ;

proc means data=outparms_log mean ;
 var estimate ;
 where variable ='ag60' ;
 by indexs ;
 output out=step1_5 mean=b5bar ;
run ;

proc print data=step1_5 ;
run ;

data step1_all_log ;
merge step1_0 step1_1 step1_2 step1_3 step1_4 step1_5;
by indexs ;
run ;

proc print data=step1_all_log ;
run ;

/* Prepare combined estimates and variance
using two data steps*/
proc means data=step1_all_log mean var ;
 var bobar b1bar b2bar b3bar b4bar b5bar;
 output out=step2_log mean=intercept male indfmpir
 ag3044 ag4559 ag60
 var=vintercept vmale vindfmpir vag3044 vag4559 vag60;
run ;

/*  Combine results for logistic regression */
data combine_log ;
 set step2_log ;
 df=5-1 ;  *Min(S-1,C-H);
 tvalue=quantile('T',0.975,df);

/* Create arrays to the calculations */
array estimate[6] intercept male indfmpir ag3044
 ag4559 ag60;
array variance[6] vintercept vmale vindfmpir vag3044
vag4559 vag60;
array se[6]       se_intercept se_male se_indfmpir
se_ag3044 se_ag4559 se_ag60 ;
array lower95[6]  l95_intercept l95_male l95_indfmpir
 l95_ag3044 l95_ag4559 l95_ag60;
array upper95[6]  u95_intercept u95_male u95_indfmpir
 u95_ag3044 u95_ag4559 u95_ag60;
array or[6]       or_intercept or_male or_indfmpir
 or_ag3044 or_ag4559 or_ag60;
do i=1 to 6;
  se[i]=sqrt((1+1/5)*variance[i]);
  or[i]=exp(estimate[i]);
  lower95[i]=exp(estimate[i]-tvalue*se[i]);
  upper95[i]=exp(estimate[i]+tvalue*se[i]);
end;
 drop i;
run ;

proc print data=combine_log ;
title "Combined Estimates from Logistic Regression" ;
run ;

</sas>
```

9.4 DESCRIBE Example

The DESCRIBE example, using NCS-R data, presents a descriptive analysis of age at interview and body mass index with a gender contrast. The example uses the 5 imputed data sets from IMPUTE along with MI combining rules and design-based variance estimation using the default TSL method. Use of the STRATUM, CLUSTER, and WEIGHT statements declare the complex sample design variables and weight to the software. The CONTRAST statement requests a linear contrast of mean age at interview and Body Mass Index by gender. The DESCRIBE command with a mean statement produces a means analysis of age at interview and BMI. Missing data is excluded, resulting in a complete case analysis.

```
<sas name="DESCRIBE Example using Imputed Data Sets">
libname d "P:\IVEware 0.3 documentation 2016\SAS XML Examples";

/* Descriptive Analysis of Age at Interview and BMI, Missing Data Imputed by IVEware */
<describe name="DESCRIBE Example Using Imputed
Data Sets with Design Adjusted Imputed Descriptive Analysis of Age and BMI">
title MI Design-based Description;
datain d.impute_mult1 d.impute_mult2 d.impute_mult3 d.impute_mult4 d.impute_mult5 ;
stratum sestrat;
cluster seclustr;
weight ncsrwtsh ;
model mult;
mean intwage bmi;
contrast sexf;
run;
</describe>
</sas>
```

9.5 REGRESS Example

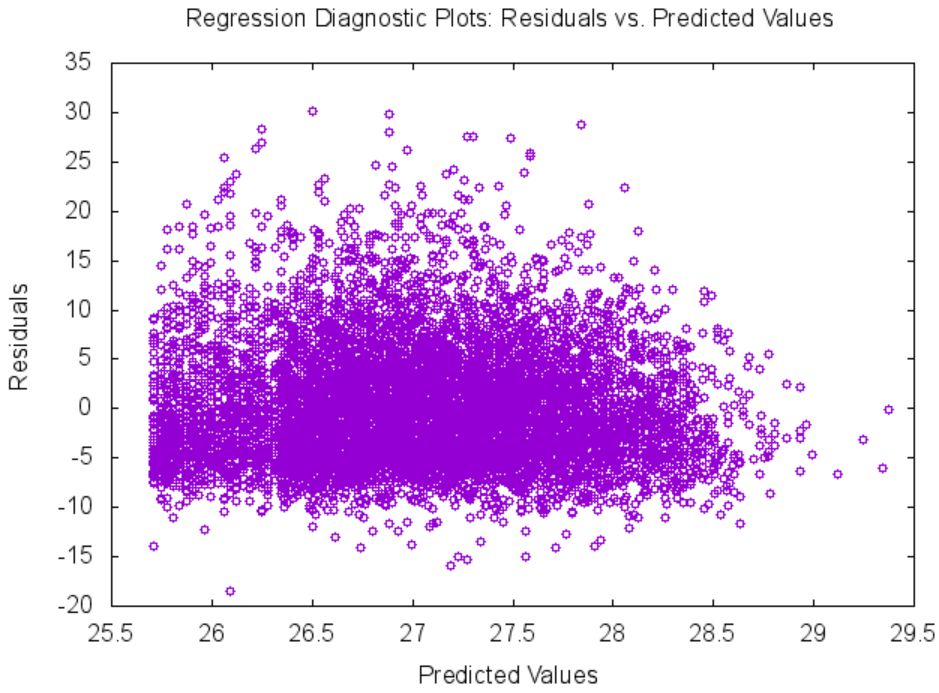
The REGRESS example again uses the previously imputed NCS-R data sets as inputs. This example regresses body mass index (BMI) on lifetime Major Depressive Episode, an indicator of being female, and age at interview. The default variance estimation method in REGRESS is the Jackknife Repeated Replication method. Use of the PLOTS statement will produce a number of diagnostic plots with a .png extension. Note that the gnuplot software must be included in the "settings" file (stored in the SRCLIB folder where software installed) for this to work correctly. One diagnostic plot is included in the selected output given below.

```
<sas name="REGRESS Example using Imputed Data Sets">
libname d "P:\IVEware 0.3 documentation 2016\SAS XML Examples";

/* Analyze Five Previously Imputed Data Sets using Linear Regression with REGRESS*/
/* Example uses Complex Sample Design Variables and Diagnostic Plots */

<regress name="Linear Regression Example">
title Linear Regression using REGRESS with Imputed Data Sets;
datain d.impute_mult1 d.impute_mult2 d.impute_mult3 d.impute_mult4 d.impute_mult5 ;
estout impute_regress;
stratum sestrat;
cluster seclustr;
weight ncsrwtsh;
dependent bmi;
predictor mde sexf intwage ;
link linear ;
plots outplots ;
run;
```

```
</regress>
</sas>
```

Selected Regression Output

```
All imputations

Valid cases              9282
Sum weights        9282.000152

Degr freedom       26.42925062

Sum of squares:
  Model            3954.942524
  Error            299501.0663
  Total            303456.0088
  R-square             0.01303
  F-value              0.08725
  P-value              0.98565

Variable              Estimate         Std Error            T Test        Prob > |T|
Intercept           25.9222187         0.2549001         101.69559           0.00000
mde                  0.8287332         0.1234398           6.71366           0.00000
sexf                -0.6898597         0.1395496          -4.94347           0.00004
intwage              0.0290391         0.0049550           5.86059           0.00000

Variable              Estimate             95% Confidence Interval
                                           Lower              Upper
Intercept           25.9222187        25.3986774        26.4457600
mde                  0.8287332         0.5751993         1.0822672
sexf                -0.6898597        -0.9764816        -0.4032378
intwage              0.0290391         0.0188620         0.0392162

Variable                Design            SRS             % Diff
                        Effect          Estimate          SRS v Est
Intercept               1.48293        26.3002986            1.45852
mde                     0.60113         0.8344926            0.69496
sexf                    1.17056        -0.6735030           -2.37102
intwage                 1.58580         0.0231927          -20.13297
```



9.6 SASMOD Example

The SASMOD command is available with *IVEware* and SAS only. This command is based upon the Jackknife Repeated Replication method for variance estimation and can be used with many SAS procedures (see previous sections of this chapter). In this example, PROC CATMOD is used to execute an MI and design-based log-linear model. The model examines relationships between gender and Major Depressive Episode, using the 5 previously imputed NCS-R data sets as input.

```
<sas name="SASMOD with PROC CATMOD">

libname d 'P:\IVEware 0.3 documentation 2016\SAS XML Examples';

/* Analyze 5 Imputed Data Sets with JRR Variance Estimation PROC CATMOD for Log-Linear Model */
<sasmod name="SASMOD for Log-Linear Model with PROC CATMOD">
title SASMOD Example ;
datain d.impute_mult1 d.impute_mult2 d.impute_mult3 d.impute_mult4 d.impute_mult5 ;
estout modexam;
cluster seclustr ;
stratum sestrat ;
weight ncsrwtsh;

/* SAS statements begin here */
proc catmod;
model mde*sexf =_response_  / pred=freq;
loglin mde sexf mde*sexf ;
run;
</sasmod>
</sas>
```

Selected Output

```
All imputations
Valid cases                9282
Sum weights          9282.000152

Degr freedom            Infinite

-2 LogLike           21833.80757

Variable                Estimate         Std Error         Wald test        Prob > Chi
mde 0                  0.7346635         0.0173232        1798.53648           0.00000
sexf 0                -0.1223741         0.0115333         112.58311           0.00000
mde*sexf 0 0           0.1283257         0.0168590          57.93785           0.00000

Variable                Estimate             95% Confidence Interval
                                             Lower              Upper
mde 0                  0.7346635         0.7006693         0.7686577
sexf 0                -0.1223741        -0.1450064        -0.0997418
mde*sexf 0 0           0.1283257         0.0952424         0.1614090
```

9.7 SYNTHESIZE Examples

9.7.1 Fully Synthesized Data Set

The first SYNTHESIZE example demonstrates multiple imputation of missing data followed by synthesis of each variable in the Primary Cardiac Arrest data set. In the code below, we request 5 multiple imputations with 5 implicates or synthesized data sets per imputation multiple. Use of similar syntax as from IMPUTE is utilized along with the SYNTHESIZE statement to declare the variables to be synthesized. The imputation is done before the synthesis, therefore each of 5 imputation multiples contains 5 synthesized data sets for subsequent analysis. Two key variables are created during this process: _IMPL_ with values of 1,2,3,4,5 denoting synthesized data sets 1-5, and _MULT_ with values of 1,2,3,4,5 denoting imputations. The final part of this example demonstrates correct combining rules for fully synthesized and imputed data (Raghunathan (2015), page 168).

```
/* Synthesize Example Using PCA and Omega 3 Fatty Acids Data */
<sas name="SYNTHESIZE Example">
/* Set libnames */
libname d1 'P:\IVEware_and_MI_Applications_Book\DataSets\PCA and Omega 3 Fatty Acids Data' ;
libname dout 'P:\IVEware 0.3 documentation 2016\SAS XML Examples';

<synthesize name="Synthesize All Variables Using PCA and Omega3 Data">
datain d1.test;
continuous AGE NUMCIG YRSSMOKE FATINDEX DHA_EPA REDTOT WGTKG TOTLKCAL HGTCM ;
categorical CASECNT GENDER RACE3 HYPER DIAB SMOKE FAMMI EDUSUBJ3 CHOLESTH ;
mixed CAFFTOT ALCOHOL3 ;
transfer STUDYID ;
synthesize CASECNT AGE GENDER RACE3 HYPER DIAB SMOKE NUMCIG YRSSMOKE FATINDEX
 FAMMI EDUSUBJ3 DHA_EPA REDTOT CHOLESTH CAFFTOT WGTKG TOTLKCAL ALCOHOL3 HGTCM ;
restrict NUMCIG(smoke=2,3) YRSSMOKE(smoke=2,3) ;
bounds NUMCIG(>0) YRSSMOKE(>0, <age-12) DHA_EPA(>0)  REDTOT(>0)
 CAFFTOT(>0)  TOTLKCAL(>0)  ALCOHOL3(>0);
ITERATIONS 2;
MULTIPLES 5;
SEED 2001;
IMPLICATES 5;
DATAOUT dout.synthesize all ;
run;
```

```
</synthesize>

/* Examine Contents of Output Data Set*/
proc contents data=dout.synthesize ;
run ;

data synthesized ;
   set dout.synthesize ;
   * use implicates 1-5 only ;
      if _impl_ >=1 then _imputation_= cat(_mult_, _impl_) ;
run ;

proc freq ;
 tables _mult_*_impl_  _imputation_ ;
 run ;

proc sort data=synthesized ;
 by _imputation_ ;
run ;

 /* Obtain mean kcalories per day by _imputation_ and then combine using correct rules for synthesized data*/
proc means data=synthesized (where=(_imputation_ ne ' ' )) mean stderr ;
 var totlkcal ;
 by _imputation_ ;
 ods output summary=outstat ;
run ;
proc print data=outstat ;
run ;


proc sql ;
 create table outstat1
  as select *, mean(totlkcal_mean) as qbar, mean(totlkcal_stderr*totlkcal_stderr) as ubar
       from outstat ;
proc sql ;
 create table outstat2
  as select *, sum((totlkcal_mean - qbar)**2)/24 as btwvar
   from outstat1

/* Combining rules for fully imputed and synthesized data*/
data final ;
 set outstat2 ;
 if _n_=1 ;
 syn_estimate=qbar ;
 syn_variance=ubar ;
 syn_se=sqrt(syn_variance) ;
 btw=btwvar  ;
 total_syn_var=((1+1/25)*btwvar) - ubar ;

proc print data=final ;
 var syn_estimate syn_se syn_variance btwvar total_syn_var;
run ;
```

Results from Fully Synthesized Data Set

Parameter	Mean	SE	Total Variance
Total Kilocalories	1822.24	40.44	106.10

9.7.2 Partially Synthesized Data Set

The second SYNTHESIZE example demonstrates how to synthesize just selected "sensitive" variables, again using the PCA data set. As a reminder, multiple imputation is done before synthesis, therefore each of 3 imputation multiples contains 5 synthesized data sets. Two

key variables are created during this process: _IMPL_ with values of 1,2,3,4,5 denoting implicates and _MULT_ with values of 1,2,3 denoting imputations. The syntax below includes imputation and synthesis followed by an example of application of correct combining rules for partially synthesized data (Raghunathan, (2015) page 168).

```
/* Partial Synthesize Example Using PCA and Omega 3 Fatty Acids Data */
<sas name="SYNTHESIZE Partial Example">
/* Set libnames */
libname d1 'P:\IVEware_and_MI_Applications_Book\DataSets\PCA and Omega 3 Fatty Acids Data' ;
libname dout 'P:\IVEware 0.3 documentation 2016\SAS XML Examples';

<synthesize name="Synthesize Selected Variables Using PCA and Omega3 Data">
datain d1.test;
continuous AGE NUMCIG YRSSMOKE FATINDEX DHA_EPA REDTOT WGTKG TOTLKCAL HGTCM ;
categorical CASECNT GENDER RACE3 HYPER DIAB SMOKE FAMMI EDUSUBJ3 CHOLESTH ;
mixed CAFFTOT ALCOHOL3 ;
transfer STUDYID ;

/* synthesize select health and personal information variables */
synthesize CASECNT HYPER DIAB FATINDEX FAMMI DHA_EPA REDTOT WGTKG HGTCM ;
restrict NUMCIG(smoke=2,3) YRSSMOKE(smoke=2,3) ;
bounds NUMCIG(>0) YRSSMOKE(>0, <age-12) DHA_EPA(>0)  REDTOT(>0)  CAFFTOT(>0)  TOTLKCAL(>0)  ALCOHOL3(>0);
ITERATIONS 2;
MULTIPLES 3;
SEED 2001;
IMPLICATES 5;
DATAOUT dout.synthesize_partial all ;
run;
</synthesize>

proc sort data=dout.synthesize_partial ;
 by _mult_ _impl_ ;
run ;
data synthesized ;
   set dout.synthesize_partial ;
   * use implicates 1-5 only ;
      if _impl_ >=1 then _imputation_= cat(_mult_, _impl_) ;
run ;

proc freq ;
 tables _mult_*_impl_  _imputation_ ;
 run ;

proc sort data=synthesized ;
 by _imputation_ ;
run ;

 /* Obtain mean fatindex by _imputation_ and then combine using correct rules for synthesized data*/
proc means data=synthesized (where=(_imputation_ ne ' ' )) mean stderr ;
 var fatindex ;
 by _imputation_ ;
ods output summary=outstat ;
run ;
proc print data=outstat ;
run ;

proc sql ;
 create table outstat1
  as select *, mean(fatindex_mean) as qbar, mean(fatindex_stderr*fatindex_stderr) as ubar
       from outstat ;
proc sql ;
 create table outstat2
  as select *, sum((fatindex_mean - qbar)**2) as sumdiffs, calculated sumdiffs/14 as btwvar
   from outstat1 ;
```

```
/* Combining rules for fully imputed and partially synthesized data are different from fully synthesized data*/
data final ;
 set outstat2 ;
 if _n_=1 ;
 syn_estimate=qbar ;
 syn_variance=ubar ;
 syn_se=sqrt(syn_variance) ;

 /* total variance is ubar + btwvar/m */
 total_partialsyn_var=ubar + btwvar/15 ;

proc print data=final ;
 var syn_estimate syn_se syn_variance btwvar total_partialsyn_var;
run ;
</sas>
```

Results from Partially Synthesized and Imputed Data Sets

Parameter	Mean	SE	Total Variance
FatIndex	21.56	0.14	0.02

9.8 COMBINE Example

The COMBINE example joins four waves of Health and Retirement Survey data from 2006, 2008, 2010 and 2012. Though the data sets have the same variable names and values, each has manufactured missing data on some variables. For example, the HRS 2006 data has no observed data on three variables: DIABETES, ARTHRITIS, and SELFRHEALTH, HRS 2008 data is completely missing on DIABETES and ARTHRITIS but has observed data on SELFRHEALTH, and so on. In addition, other variables such as marital status (MARCAT) have small amounts of missing data in some or all waves of data. This example demonstrates a three step process: 1. "stacking" the data using COMBINE, 2. imputing the missing data in the combined data set using IMPUTE, and 3. use of DESCRIBE to perform MI and design-based descriptive analysis of arthritis by year.

```
<sas name="COMBINE Example using 4 Waves of HRS Data">
libname d "P:\IVEware_and_MI_Applications_Book\Chapter2DataSources";

/* COMBINE Example Using HRS 2006 2008 2010 and 2012 Data */
<combine name="COMBINE Example">
datain d.hrs2006_27jul2016 d.hrs2008_27jul2016 d.hrs2010_27jul2016 d.hrs2012_27jul2016 ;
dataout d.combined_hrs_2006_2012 ;
run;
</combine>

/* Use SAS to Examine Contents of Combined Data */
proc contents data=d.combined_hrs_2006_2012 ;
run ;

/* Examine Means of Combined Data */
proc means n nmiss mean min max data=d.combined_hrs_2006_2012 ;
class yr ;
run ;

/* Use IMPUTE to impute missing data */
<impute name="Impute_Post_COMBINE">
datain d.combined_hrs_2006_2012 ;
dataout d.impute_mult1;
```

```
default categorical;
continuous stratum age wgt;
transfer hhid pn ;
iterations 5;
multiples 5;
seed 2016 ;
run;
</impute>

/* Extract remaining 4 data sets */
<putdata name="Impute_Post_combine" mult="2" dataout="d.impute_mult2" />
<putdata name="Impute_Post_combine" mult="3" dataout="d.impute_mult3" />
<putdata name="Impute_Post_combine" mult="4" dataout="d.impute_mult4" />
<putdata name="Impute_Post_combine" mult="5" dataout="d.impute_mult5" />

/* Use Imputed Data Sets for Descriptive Analysis*/
<describe name="Descriptive Analysis of Combined and Imputed Arthritis by Year">
datain d.impute_mult1 d.impute_mult2 d.impute_mult3 d.impute_mult4 d.impute_mult5 ;
stratum stratum ;
cluster secu;
weight wgt ;
table arthritis ;
by yr ;
run;
</describe>

</sas>
```

Proportion of Arthritis by Year, Based on Combined, Imputed HRS Data

Outcome	Year	Mean	SE
Diagnosis of Arthritis	2006	0.548	0.007
”	2008	0.567	0.008
”	2010	0.532	0.006
”	2012	0.550	0.008

Chapter 10

IVEware and Stata, SPSS, and R

10.1 Introduction

Chapter 10 presents syntax to replicate the examples in Chapter 9, again using *IVEware* with the SRCware Shell editor with Stata, SPSS and R. In this chapter, just the *IVEware* command syntax is presented while subsequent coding and analyses are left to the analyst to code in a software of choice. Though the software with *IVEware* differs, the *IVEware* syntax and resultant output should match the Chapter 9 examples.

10.2 IMPUTE Example, *IVEware* and Stata

The first example demonstrates use of *IVEware* and Stata with IMPUTE, using NCS-R data. After imputation, the 5 multiples are stored in data sets called "impute_mult1-impute_mult5". By default, these are saved in Stata .dta format and can be used in additional analyses.

```
<stata name="Impute Example with Stata">

use "P:\IVEware_and_MI_Applications_Book\Chapter3\Examples\Stata\ncsr_ex1.dta"

/* Multiple Imputation*/
<impute name="Impute">
title Multiple Imputation Using IMPUTE ;
datain ncsr_ex1 ;
dataout impute_mult1;
default categorical;
continuous bmi intwage ncsrwtsh sestrat;
transfer caseid;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

/*Extract remaining 4 data sets*/
<putdata name="Impute" mult="2" dataout="impute_mult2"/>
<putdata name="Impute" mult="3" dataout="impute_mult3"/>
<putdata name="Impute" mult="4" dataout="impute_mult4"/>
<putdata name="Impute" mult="5" dataout="impute_mult5"/>

</stata>
```

10.3 BBDESIGN Example, *IVEware* and SPSS

Section 10.3 demonstrates use of *IVEware* and SPSS with the BBDESIGN command to produce an output population data set called "bbdesign_samples.sav" (SPSS format). As usual, the output data set can be used in additional analyses within SPSS or transferred to another statistical software package.

```
<spss name="BBDESIGN Examples">
/* example uses 2011 - 2012 NHANES data, subset for age 18+ for adults*/
/* SPSS .sav file is stored in working folder */

<bbdesign name="BBdesign">
title Use of BBdesign;
datain nhanes1112_adult;
dataout bbdesign_samples ;
stratum sdmvstra ;
cluster sdmvpsu ;
weight wtmec2yr ;
csamples 5 ;
wsamples 5 ;
seed 2001;
run;
</bbdesign>
</spss>
```

10.4 DESCRIBE Example, *IVEware* and R

The DESCRIBE command uses *IVEware* and R to perform a descriptive analysis of age of interview and Body Mass Index with a gender contrast. This examples uses NCS-R data with a design-based/multiple imputation approach for variance estimation. The five previously imputed NCS-R data sets are first imported into R and then used in the DESCRIBE command.

```
<R name="DESCRIBE Example">
# iveware examples - R version

# import the input datasets
impute_mult1 <- read.delim("impute_mult1.txt")
save(impute_mult1, file="impute_mult1.rda")

impute_mult2 <- read.delim("impute_mult2.txt")
save(impute_mult2, file="impute_mult2.rda")

impute_mult3 <- read.delim("impute_mult3.txt")
save(impute_mult3, file="impute_mult3.rda")

impute_mult4 <- read.delim("impute_mult4.txt")
save(impute_mult4, file="impute_mult4.rda")

impute_mult5 <- read.delim("impute_mult5.txt")
save(impute_mult5, file="impute_mult5.rda")

# Descriptive Analysis of Age at Interview and BMI
<describe name="DESCRIBE">
title MI Design-based Description;
datain impute_mult1 impute_mult2 impute_mult3 impute_mult4 impute_mult5;
stratum sestrat;
cluster seclustr;
weight ncsrwtsh ;
model mult;
mean intwage bmi;
```

```
contrast sexf;
run;
</describe>

</R>
```

10.5 REGRESS Example, *IVEware* and Stata

The REGRESS example inputs five previously imputed NCS-R data sets and performs linear regression with *IVEware* and Stata. Note that the imputed input data sets were previously stored as Stata data sets in .dta format.

```
<stata name="REGRESS Example with Stata">
/* Analyze 5 imputed data sets with Linear Regression */
<regress name="REGRESS for Linear Regression with Imputed Data Sets">
title Example of REGRESS ;
datain impute_mult1 impute_mult2 impute_mult3 impute_mult4 impute_mult5 ;
estout impute_regress;
stratum sestrat;
cluster seclustr;
weight ncsrwtsh;
dependent bmi;
predictor mde sexf intwage ;
run;
</regress>
</stata>
```

10.6 SYNTHESIZE Example, *IVEware* and Stata

This example uses *IVEware* and Stata to synthesize all variables in the Primary Cardiac Arrest data set. An output data set called "synthesize.dta" contains the synthesized and multiply imputed data.

```
* Synthesize Example Using PCA and Omega 3 Fatty Acids Data */
<stata name="SYNTHESIZE Example">

<synthesize name="Synthesize All Variables Using PCA and Omega3 Data">
datain test;
continuous AGE NUMCIG YRSSMOKE FATINDEX DHA_EPA REDTOT WGTKG TOTLKCAL HGTCM ;
categorical CASECNT GENDER RACE3 HYPER DIAB SMOKE FAMMI EDUSUBJ3 CHOLESTH ;
mixed CAFFTOT ALCOHOL3 ;
transfer STUDYID ;
synthesize CASECNT AGE GENDER RACE3 HYPER DIAB SMOKE NUMCIG YRSSMOKE FATINDEX
 FAMMI EDUSUBJ3 DHA_EPA REDTOT CHOLESTH CAFFTOT WGTKG TOTLKCAL ALCOHOL3 HGTCM ;
restrict NUMCIG(smoke=2,3) YRSSMOKE(smoke=2,3) ;
bounds NUMCIG(>0) YRSSMOKE(>0, <age-12) DHA_EPA(>0) REDTOT(>0) CAFFTOT(>0) TOTLKCAL(>0) ALCOHOL3(>0);
ITERATIONS 2;
MULTIPLES 5;
SEED 2001;
IMPLICATES 5;
DATAOUT synthesize all ;
run;
</synthesize>
</stata>
```

10.7 COMBINE Example, *IVEware* and R

The COMBINE example pairs *IVEware* with R to demonstrate how to combine multiple data sets. The four HRS data sets (2006, 2008, 2010, 2012) are first imported into R

using the Foreign package and then concatenated by COMBINE. An output data set called "combined_hrs_2006_2012" in R format is created for subsequent imputation and analysis.

```
<R name="COMBINE Example">
# iveware examples - R version

# load foreign package and read in SAS data sets
library(foreign)
hrs2006_27jul2016r <- read.xport("hrs2006_27jul2016.xpt")
hrs2008_27jul2016r <- read.xport("hrs2008_27jul2016.xpt")
hrs2010_27jul2016r <- read.xport("hrs2010_27jul2016.xpt")
hrs2012_27jul2016r <- read.xport("hrs2012_27jul2016.xpt")


<combine name="COMBINE_Example_R">
datain hrs2006_27jul2016r hrs2008_27jul2016r hrs2010_27jul2016r hrs2012_27jul2016r ;
dataout combined_hrs_2006_2012;
run;
</combine>

summary(combined_hrs_2006_2012)

</R>
```

Chapter 11

SRCWare

11.1 Introduction

Chapter 11 demonstrates use of SRCWare, the stand-alone version of *IVEware*. This chapter includes examples of the IMPUTE, BBDESIGN, DESCRIBE, REGRESS, SYNTHESIZE, and COMBINE commands. Input data sets are read into SRCWare using the GETDATA command while data sets from SRCWare can be output using the PUTDATA command. Output data sets can be used in subsequent analyses using a software of choice. For examples of the GETDATA and PUTDATA commands, see Section 1.6 and provided examples on the *IVEware* website. Because the *IVEware* commands have been detailed in previous chapters, just the code is presented in this chapter.

11.2 IMPUTE Example

```
<srcware name="SRCWARE_IMPUTE">

/* import the input datasets */
<getdata name="ncsr_ex1">
table ncsr_ex1.txt;
run;
</getdata>

/* Multiple Imputation using IMPUTE*/
<impute name="MI">
title Multiple Imputation Using IMPUTE ;
datain ncsr_ex1 ;
dataout impute_mult1;
default categorical;
continuous bmi intwage ncsrwtsh sestrat ;
transfer caseid ;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

/* Extract remaining data sets */
<putdata name="MI" mult="2" dataout="impute_mult2" />
<putdata name="MI" mult="3" dataout="impute_mult3" />
<putdata name="MI" mult="4" dataout="impute_mult4" />
<putdata name="MI" mult="5" dataout="impute_mult5" />
```

```
</srcware>
```

11.3 BBDESIGN Example

```
<srcware name="SRCWARE_BBDESIGN">

/* import the input dataset*/
<getdata name="nhanes1112_adult">
table nhanes1112_adult.txt;
run;
</getdata>

<bbdesign name="BBdesign">
title Use of BBdesign;
datain nhanes1112_adult;
dataout bbdesign_samples ;
stratum sdmvstra ;
cluster sdmvpsu ;
weight wtmec2yr ;
csamples 5 ;
wsamples 5 ;
seed 2001;
run;
</bbdesign>

</srcware>
```

11.4 DESCRIBE Example

```
<srcware name="SRCWARE_IMPUTE">

/* import the input dataset */
<getdata name="ncsr_ex1">
table ncsr_ex1.txt;
run;
</getdata>

/* Review of Multiple Imputation using IMPUTE*/
<impute name="MI">
title Multiple Imputation Using IMPUTE ;
datain ncsr_ex1 ;
dataout impute_mult1;
default categorical;
continuous bmi intwage ncsrwtsh sestrat ;
transfer caseid ;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

/* Extract remaining data sets */

<putdata name="MI" mult="2" dataout="impute_mult2" />
<putdata name="MI" mult="3" dataout="impute_mult3" />
<putdata name="MI" mult="4" dataout="impute_mult4" />
<putdata name="MI" mult="5" dataout="impute_mult5" />

/* Descriptive Analysis of Age at Interview and BMI, Missing Data Imputed by IVEware */
<describe name="DESCRIBE">
title MI Design-based Description;
datain impute_mult1 impute_mult2 impute_mult3 impute_mult4 impute_mult5 ;
stratum sestrat;
cluster seclustr;
```

```
weight ncsrwtsh ;
model mult;
mean intwage bmi;
contrast sexf;
run;
</describe>
</scrware>
```

11.5 REGRESS Example

```
<srcware name="SRCWARE_IMPUTE">

/* import the input datasets */
<getdata name="ncsr_ex1">
table ncsr_ex1.txt;
run;
</getdata>

/* Review of Multiple Imputation using IMPUTE*/
<impute name="MI">
title Multiple Imputation Using IMPUTE ;
datain ncsr_ex1 ;
dataout impute_mult1;
default categorical;
continuous bmi intwage ncsrwtsh sestrat ;
transfer caseid ;
iterations 5;
multiples 5;
seed 2001;
run;
</impute>

/* Extract remaining data sets */

<putdata name="MI" mult="2" dataout="impute_mult2" />
<putdata name="MI" mult="3" dataout="impute_mult3" />
<putdata name="MI" mult="4" dataout="impute_mult4" />
<putdata name="MI" mult="5" dataout="impute_mult5" />

/* Analyze Previously Imputed Data Sets using Linear Regression with REGRESS */
/* Example uses Complex Sample Design Variables and Weight Plus Plots */

<regress name="REGRESS">
title Linear Regression using REGRESS with Imputed Data Sets;
datain impute_mult1 impute_mult2 impute_mult3 impute_mult4 impute_mult5 ;
estout impute_regress;
stratum sestrat;
cluster seclustr;
weight ncsrwtsh;
dependent bmi;
predictor mde sexf intwage ;
link linear ;
plots outplots ;
run;
</regress>
 </srcware>
```

11.6 SYNTHESIZE Example

```
<srcware name="SRCWARE_SYNTHESIZE">
/* import the input datasets */
<getdata name="test">
table test.txt;
run;
</getdata>
```

```
/* Use SYNTHESIZE command to prepare fully synthesized data set*/
<synthesize name="Synthesize">
datain test;
continuous AGE NUMCIG YRSSMOKE FATINDEX DHA_EPA REDTOT WGTKG TOTLKCAL HGTCM ;
categorical CASECNT GENDER RACE3 HYPER DIAB SMOKE FAMMI EDUSUBJ3 CHOLESTH ;
mixed CAFFTOT ALCOHOL3 ;
transfer STUDYID ;
synthesize CASECNT AGE GENDER RACE3 HYPER DIAB SMOKE NUMCIG
YRSSMOKE FATINDEX FAMMI EDUSUBJ3 DHA_EPA REDTOT CHOLESTH CAFFTOT WGTKG TOTLKCAL
ALCOHOL3 HGTCM ;
restrict NUMCIG(smoke=2,3) YRSSMOKE(smoke=2,3) ;
bounds NUMCIG(>0) YRSSMOKE(>0, <age-12) DHA_EPA(>0)  REDTOT(>0)  CAFFTOT(>0)  TOTLKCAL(>0)  ALCOHOL3(>0);
ITERATIONS 2;
MULTIPLES 5;
SEED 2001;
IMPLICATES 5;
DATAOUT synthesize all ;
run;
</synthesize>
</srcware>
```

11.7 COMBINE Example

```
<srcware name="SRCWARE_COMBINE">

/* import the input datasets */
<getdata name="hrs2006_27jul2016">
table hrs2006_27jul2016.csv;
run;
</getdata>

<getdata name="hrs2008_27jul2016">
table hrs2008_27jul2016.csv;
run;
</getdata>

<getdata name="hrs2010_27jul2016">
table hrs2010_27jul2016.csv;
run;
</getdata>

<getdata name="hrs2012_27jul2016">
table hrs2012_27jul2016.csv;
run;
</getdata>

/* COMBINE Example Using HRS 2006 2008 2010 and 2012 Data*/
<combine name="COMBINE">
datain hrs2006_27jul2016 hrs2008_27jul2016 hrs2010_27jul2016 hrs2012_27jul2016 ;
dataout combined_hrs_2006_2012 all ;
run;
</combine>
</srcware>
```

Bibliography

[1] Atkinson, A. C. (1985). Plots, transformations and regression: An introduction to graphical methods of diagnostic regression analysis. Oxford: Clarendon Press.

[2] Bondarenko, I. & Raghunathan, T. E. (2010). Multiple imputation for causal inference. Arbor, 1001(48109)

[3] Bondarenko, I. & Raghunathan, T. E. (2016). Graphical and numerical diagnostic tools to assess suitability of multiple imputations and imputation models. Statistics in Medicine, 35, 3007-3020.

[4] Dong, Q., Elliott, M. R. & Raghunathan, T. E. (2014a). A nonparametric method to generate synthetic populations to adjust for complex sampling design features. Survey Methodology, 40(1), 29-46.

[5] Dong, Q., Elliott, M. R. & Raghunathan, T. E. (2014b). Combining information from multiple complex surveys. Survey Methodology, 40, 347-354.

[6] Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. (1995). Bayesian data analysis. London: Chapman and Hall.

[7] Gelman, A. & Hill, J. (2006). Data analysis using regression and Multilevel/Hierarchical models. New York: Cambridge University Press.

[8] He, Y. & Raghunathan, T. E. (2006). Tukey's gh distribution for multiple imputation. The American Statistician, 60, 251-256: Response.

[9] Heeringa, S. G., Little, R. J. A., & Raghunathan, T. E. (1997). Imputation of multivariate data on household net worth. University of Michigan, Ann Arbor, Michigan,

[10] Kish, L. & Frankel, M. (1974). Inference from complex systems. Journal of the Royal Statistical Society.Series B (Methodological), 36(1), 1-37.

[11] Li, K. H., Raghunathan, T. E. & Rubin, D. B. (1991). Large-sample significance levels from multiply imputed data using moment-based statistics and an F-reference distribution. Journal of the American Statistical Association, 86(416), 1065-1073.

[12] Little, R. J. A., Liu, F. Raghunathan, T. E. (2004). Statistical Disclosure Techniques Based on Multiple Imputation. P.p. 141-152 in Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives: An Essential Journey with Donald Rubins Statistical Family (A. Gelman X.-L. Meng, eds.).

[13] Raghunathan, T. E. (1994). Monte carlo methods for exploring sensitivity to distributional assumptions in a bayesian analysis of a series of 2 2 tables. Statistics in Medicine, 13(15), 1525-1538.

[14] Raghunathan, T. E., Lepkowski, J. M., Hoewyk, J. V. & Solenberger, P. (2001). A multivariate technique for multiply imputing missing values using a sequence of regression models. Survey Methodology, 27(1), 85-95.

[15] Raghunathan, T. E., Reiter, J. P. & Rubin, D. B. (2003). Multiple imputation for statistical disclosure limitation. Journal of Official Statistics-Stockholm-, 19(1), 1-16.

[16] Raghunathan, T. E. & Rubin, D. B. (1998). Roles for Bayesian techniques in survey sampling. Proceedings of the Silver Jubilee Meeting of the Statistical Society of Canada, , 51-55.

[17] Raghunathan, T. E. (2015). In Chapman & Hall/CRC Interdisciplinary Statistics Series (Ed.), Missing data analysis in practice. Boca Raton: CRC Press.

[18] Raghunathan, T. E, Berglund, P., and Solenberger, P. W. (2017). Multiple Imputation in Practice: With Examples Using *IVEware*. Boca Raton: CRC Press.

[19] Reiter, J. (2002). Satisfying disclosure restrictions with synthetic data sets. Journal of Official Statistics, 18(4), 531-543.

[20] Rubin, D. B. (1976). Inference and missing data. Biometrika, 63(3), 581-592.

[21] Rubin, D. B. (1987b). Multiple imputation for nonresponse in surveys (99th ed.) Wiley.

[22] Schenker, N., Raghunathan, T. E. & Bondarenko, I. (2010). Improving on analyses of self-reported data in a large-scale health survey by using information from an examination-based survey. Statistics in Medicine, 29(5), 533-545.

[23] van Buuren, S. (2012). Flexible imputation of missing data. Chapman and Hall/CRC.

[24] van Buuren, S. & Oudshoorn, K. (1999). Flexible multivariate imputation by MICE. Technical Report, Leiden: TNO Preventie En Gezondheid, TNO/VGZ/PG 99.054.

[25] Vittinghoff, E., Glidden, D. V., Shiboski, S. C. & McCulloch, C. E. (2005). Regression methods in biostatistics: Linear, logistic, survival, and repeated measures models. New York, NY, US: Springer Publishing Co.

[26] Weisberg, S. (2013). Applied linear regression (4th ed.) Wiley.

[27] Zhou, H., Elliott, M. R. & Raghunathan, T. E. (2016). Synthetic Multiple-Imputation Procedure for Multistage Complex Samples, Journal of Official Statistics, 32, 231-256.

[28] Zhou, H., Elliott, M. R.& Raghunathan, T. E. (2016). Multiple imputation in two-stage cluster samples using the weighted finite population bayesian bootstrap. Journal of Survey Statistics and Methodology, 4, 139-170.

[29] Zhou, H., Elliott, M. R. & Raghunathan, T. E. (2015). A two-step semiparametric method to accommodate sampling weights in multiple imputation. Biometrics, 72, 242-252.