Transitioning to R: Replicating SAS, Stata, and SUDAAN Analysis Techniques in Health Policy Data

by Anthony Damico

Abstract: Statistical, data manipulation, and presentation tools make R an ideal integrated package for research in the fields of health policy and healthcare management and evaluation. However, the technical documentation accompanying most data sets used by researchers in these fields does not include syntax examples for analysts to make the transition from another statistical package to R. This paper describes the steps required to import health policy data into R, to prepare that data for analysis using the two most common complex survey variance calculation techniques, and to produce the principal set of statistical estimates sought by health policy researchers. Using data from the Medical Expenditure Panel Survey Household Component (MEPS-HC), this paper outlines complex survey data analysis techniques in R, with side-by-side comparisons to the SAS, Stata, and SUDAAN statistical software packages.

Introduction

Health-care researchers use survey data to evaluate the state of the market and to inform the creation and implementation of new policies. America's continuing discussion about the health-care system highlights the unique role of population-generalizable data as a gauge for estimating how proposed legislation will affect specific demographic groups and the $2.2 trillion health-care industry.[1] Despite powerful statistical analysis, data manipulation, and presentation capabilities, R has not become widely adopted as a statistical package in the fields of health policy or health-care management and evaluation. User guidelines and other technical documents for government-funded and publicly available data sets rarely provide appropriate code examples to R users. The objective of this paper is to describe the steps required to import health policy data into R, to prepare that data for analysis using the two most common complex survey variance calculation techniques, and to produce the principal set of statistical estimates sought by health policy researchers.

This article compares means and standard errors produced by technical documentation-recommended code examples for selected proprietary statistical packages with those of the **survey** package available in R, using data from the Medical Expenditure Panel Survey–Household Component (MEPS-HC).[2]

Medical Expenditure Panel Survey–Household Component

- Nationally representative (non-institutionalized) sample of health services utilization in the United States
- Administered by the Agency for Healthcare Research and Quality (AHRQ)
- Panel survey design allows tracking of all individuals and families for two calendar years
- Person-level and event-level annual data available since 1996

MEPS-HC data provides a useful basis for cross-package comparisons because variances can be estimated using both Taylor-series linearization and replicate weighting methods, unlike most complex sample survey data sets which recommend only one or the other. Presenting a step-by-step methodology with the MEPS-HC data set allows more convenient generalization to other data sets.

Data preparation

AHRQ provides all downloadable MEPS files in both ASCII (.txt) and SAS transport (.xpt and .ssp) formats, along with programming statements for SAS and SPSS. The **foreign** package imports SAS transport files with minimal effort:

```
> library(foreign)
> h105 <- read.xport("h105.ssp")
```

Reading larger data sets with the `read.xport` function may overburden system memory.[3] If a memory error occurs, SAS transport files can be converted to tab-delimited (.txt) files using the free SAS System Viewer® and then alternatively imported with the `read.table` function:

```
> h105 <- read.table("h105.dat", header=TRUE )
```

[1] http://www.cms.hhs.gov/NationalHealthExpendData/downloads/highlights.pdf
[2] http://www.meps.ahrq.gov/mepsweb/data_stats/download_data_files.jsp
[3] http://www.biostat.jhsph.edu/~rpeng/docs/R-large-tables.html

Should memory errors continue to occur due to a large number of variables in the data set, database-backed survey design objects allow R to access a table stored within a relational database using either DBI or ODBC.[4] This alternative does not conserve memory in data sets that are large due to record count. The table is accessed directly by the `svydesign` function (discussed later), through the addition of a `dbname` argument to specify the location and a `dbtype` argument to specify the database driver.

After successful import, unnecessary variables should be eliminated from the data frame to increase processing speed. The `select` argument to the `subset` command acts as the SAS *KEEP* option in a DATA step or the Stata *keep* command, by dropping all unspecified fields. Distinct subpopulation data frames could also be created with the `subset` command; however, observations should not be excluded until the design object has been created, in order to preserve the entire complex sample design.[5]

```
> #limit to specified fields
> meps06 <- subset(h105,
+   select = c(DUPERSID, PANEL, PERWT06F, VARSTR,
+     VARPSU, AGE06X, TOTEXP06, REGION06, RTHLTH53))

> #clear memory of full data frame
> rm(h105)
```

All other recoding and re-categorization should occur at this stage. The example below creates a categorical variable based on the age of the individual using the `cut` function to delineate the appropriate intervals.

```
> #recode linear to categorical
> # below 25 ; 25-34 ; 35-44 ; 45-54 ;
> # 55-64 ; 65-74 ; 75-84 ; 85+
> meps06[, "AGECAT"] <- cut(meps06[, "AGE06X"],
+   br = c(-1, 24 + 10*0:6, Inf), labels = 1:8)
```

Note that the `labels` argument can directly produce character strings rather than numeric categories. This example creates the same categories as above, with character labels.

```
> #alternate character labels
> meps06[, "cAGECAT"] <- cut(meps06[, "AGE06X"],
+   br = c(-1, 24+10*0:6, Inf),
+   labels = c('Below 25', '25-34', '35-44',
+     '45-54', '55-64', '65-74', '75-84', '85+'))
```

Once all needed variables have been created, inapplicable or invalid values should be converted to missing. This example combines the 'Excellent' and 'Very good' self-reported health status responses, then assigns incomplete records as missing values.[6]

```
> #combine ex (1) & vg health (2)
> # into the same response option
> meps06[, "RTHLTH53"] <- meps06[ ,"RTHLTH53"] - 1
> meps06[meps06[ ,"RTHLTH53"] == 0, "RTHLTH53"] <-
+   1

> #recode negatives to missing (NA)
> meps06[meps06[ , "RTHLTH53"] < 0, "RTHLTH53"] <-
+   NA
```

The addition of missing values to any variable within a data frame requires that subsequent analyses of the variable include the `na.rm = TRUE` argument in order to appropriately exclude those observations.

At the conclusion of all data transformations, all variables that one wishes to display in weighted column frequencies should be converted to a character string in a new variable. The rationale behind this conversion is discussed in the *Analysis* section.

```
> #char vars for region & health stat.
> meps06 <- transform(meps06,
+   cRTHLTH53 = as.character(RTHLTH53))
> meps06 <- transform(meps06,
+   cREGION06 = as.character(REGION06))
```

Survey design for Taylor-series linearization

The estimation weight (PERWT06F), stratum (VARSTR), and primary sampling unit (VARPSU) fields make up the MEPS survey design and allow for the Taylor-series linearization method of standard error computation. The **survey** package can then be used to create a complex sample survey design object for each data set. After creation, the complex survey design object replaces the R data frame as the target of all analysis functions.

```
> library(survey)

> #create main tsl survey design obj.
> meps.tsl.dsgn <- svydesign(id = ~ VARPSU,
+   strata = ~ VARSTR, weights = ~ PERWT06F,
+   data = meps06, nest = TRUE)
```

Survey design objects store (rather than point to) data sets at the time of design object creation; any modifications of the `meps06` data frame in subsequent steps require the above re-assignment of the design objects in order to see those changes in subsequent analysis commands.

[4] http://faculty.washington.edu/tlumley/survey/svy-dbi.html

[5] http://faculty.washington.edu/tlumley/survey/example-domain.html

[6] Original Self-Reported Health Status Values in MEPS: (1) Excellent; (2) Very good; (3) Good; (4) Fair; (5) Poor; (-9) Not Ascertained; (-8) Don't Know; (-7) Refused; (-1) Inapplicable.

Taylor-series linearization cross-package comparisons

Many health policy data sets with complex sample designs use the Taylor-series linearization method for standard error computation. After the appropriate survey design object has been created, point estimates can be obtained.

```
> #total adult expendure mean & se
> svymean(~ TOTEXP06,
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #total nonelderly adult expenditure
> svymean(~ TOTEXP06 ,
+   subset(meps.tsl.dsgn, AGE06X > 17
+     & AGE06X < 65))
```

As seen in Table 1, R accurately replicates the estimates derived by each of the other statistical packages, SAS, Stata, and SUDAAN. The programming code for each of these other three packages matches the AHRQ recommendations found in their data documentation. [7]

Since policy research and evaluations often dictate that adults and children be studied separately, this paper will chiefly analyze the data set after restricting the sample to individuals aged 18 or above. Although used throughout this paper, note that the `subset` design object embedded in the functions above could be re-created as a pointer.

```
> x <- subset(meps.tsl.dsgn, AGE06X > 17)
```

Data preparation and survey design for replicate weighting

Note: Analysts solely interested in Taylor-series linearization can skip this section. Standard errors in MEPS-HC can be calculated without the Balanced Repeated Replication (BRR) technique; however, these survey design steps can be generalized to other health policy data sets that recommend replicate weighting, including the Consumer Expenditure Survey (CE) and the Survey of Income and Program Participation (SIPP). Replicate weighting designs provide the dual advantages of allowing quantile point estimate standard errors to be calculated with relative ease and of maintaining accurate variances when a subpopulation under examination has been so restricted as to contain a stratum with only one PSU.[8] The code below imports the MEPS-HC Replicates file and joins that file with the adult file and the non-elderly adult file.

```
> #load the half-sample data
> library(foreign)
> brr <- read.xport("h36b06.ssp")

> #merge half-sample with main file
> meps.brr <- merge(meps06, brr,
+   by.x = c("DUPERSID", "PANEL"),
+   by.y = c("DUPERSID", "PANEL"),
+   all=FALSE)
```

Note that if additional variables need to be removed from a data frame, the select argument captures all fields in between two fields separated by a colon - a shortcut particularly valuable for maintaining 64 replicate weight variables.

```
> meps.brr.sub <- subset(meps.brr,
+   select = c(DUPERSID, PANEL, PERWT06F,
+     VARSTR, VARPSU, AGE06X, AGECAT,
+     TOTEXP06, REGION06, cREGION06,
+     RTHLTH53, cRTHLTH5, BRR1:BRR64))
```

The survey replicate weight function `svrepdesign` allows the creation of a replicate weight-based survey design object. The `repweights` argument requires the identification of the 64 replicate weight fields in the data frame, obtained in the code below by isolating all fields containing the string `"BRR"` with the `grep` function. Half-sample replication techniques generally rely on the replicate weights equation BRR**X**wt = BRR**X** * 2 * PERWT06F, where **X** is 1:64 and each BRR**X** field is a binary variable containing an equal number of zeroes and ones that are randomly distributed across the records in the data set. If the series of replicate weights has already been computed to BRR**X**wt, the `combined.weights` argument should be set to `TRUE`. The downloadable MEPS BRR file contains only binary BRR**X** variables, indicating that R must take the extra step of computing the BRR**X**wt weights.[9]

```
> #create meps brr survey design obj.
> meps.brr.dsgn <-svrepdesign(data = meps.brr,
+   repweights =
+     meps.brr[ ,grep("^BRR", colnames(meps.brr))],
+   type = "BRR", combined.weights = FALSE,
+   weights = meps.brr$PERWT06F)
```

Replicate weighting cross-package comparisons

Note: Analysts solely interested in Taylor-series linearization can skip this section. Data sets used for health policy research often require the calculation of standard errors using replicate weights. Using the same `svymean` function, the point estimate can be obtained again by simply changing the survey design object.

```
> #brr method: total $ mean & se
> svymean(~ TOTEXP06,
+   subset(meps.brr.dsgn, AGE06X > 17))
```

[7] http://www.meps.ahrq.gov/mepsweb/survey_comp/standard_errors.jsp
[8] http://faculty.washington.edu/tlumley/survey/exmample-lonely.html
[9] http://www.meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-036BRR

Table 1: Comparison of Taylor-Series Linearization Standard Error Calculation Methods

		R	SAS	Stata	SUDAAN
All Adults	Code	`svymean(~TOTEXP06, subset( meps.tsl.dsgn, AGE06X > 17))`	`proc surveymeans data=meps.adult; stratum varstr; cluster varpsu; weight perwt06f; var totexp06;`	`svyset [pweight=PERWT06F], strata(VARSTR) psu(VARPSU) svy: mean TOTEXP06`	`proc descript data="adult" filetype=sasxport design=wr notsorted; nest varstr varpsu; weight perwt06f; var totexp06;`
	Mean	4009.1	4009.1	4009.1	4009.1
	SE	82.4	82.4	82.4	82.4
Non-Elderly Adults	Code	`svymean(~TOTEXP06, subset( meps.tsl.dsgn, AGE06X > 17 & AGE06X < 65))`	`proc surveymeans data=meps.adult; where age06x < 65; stratum varstr; cluster varpsu; weight perwt06f; var totexp06;`	`svyset [pweight=PERWT06F], strata(VARSTR) psu(VARPSU) svy: mean TOTEXP06 if AGE06X < 65`	`proc descript data = "adult" filetype=sasxport design=wr notsorted; nest varstr varpsu; weight perwt06f; var totexp06; subpopn age06x < 65;`
	Mean	3155.4	3155.4	3155.4	3155.4
	SE	76.2	76.2	76.2	76.2

As seen in Table 2, the survey design object created by R matches the Stata variance estimation precisely where the mean-squared error (MSE) option has *not* been selected; this estimate is derived from the sum of squares of the replicate estimates centered at the mean of those estimates. Stata's variance estimation using the MSE option and SUDAAN's built-in replicate weighting algorithm both center on the actual point estimate rather than the mean of the replicate estimates, resulting in a slightly larger confidence interval. The choice between these two techniques does not have any theoretical considerations, and the results are almost always similar. Though marginally different, standard errors computed with R are equally valid as those computed with SUDAAN or with Stata's MSE method. Two statistical tests conducted using the standard errors from the different packages will almost always produce a numerically different but substantively compatible result.

Comparison code for SAS was not included in Table 2 because the base package requires the calculation of replicate weighting estimates with user-written code.

Analysis

The core functions included in R's **survey** package allow the rapid estimation and cross-tabulation of means, distributions, frequencies, and quantiles. Each of the discussed operations also produces standard errors, except for the quantile calculations using Taylor-series linearization design objects.

Unweighted frequencies

Before conducting analysis on the complex survey design object, simple unweighted frequencies can be produced using the `xtabs` function. Unweighted frequencies only assure that sufficient cell sizes exist, meaning that the complex design does not need to be accounted for. Therefore, this function refers to the source data set, rather than the design object.

```
> #calculate unweighted cell size
> xtabs( ~ RTHLTH53 + REGION06,
+   subset(meps06, AGE06X > 7))
```

Distributions

As previously shown, the `svymean` function calculates mean values for any numeric variable. When given character variables, however, `svymean` computes percentages of the total. Notice the presence of the `na.rm = TRUE` argument to eliminate missing values from the results.

```
> #percent distribution w/ svymean
> #determine % adults in each state
> # (ex. poor health 3.26%)
> a <- svymean(~ cRTHLTH53,
+   subset(meps.tsl.dsgn, AGE06X > 17),
+     na.rm = TRUE)
> a
```

Although this paper focuses on data manipulation and analysis, R includes powerful formatting and presentation tools. Individuals interested in converting simple tables — like the one produced by the code below — into more advanced graphics should research the R packages **graphics**, **lattice**, **grid**, **grDevices**, as well as LaTeX converters like **Hmisc** and **xtable**.

```
> #label & format health status
> ftable(a,
+   list(c("Excellent / Very good",
+     "Good", "Fair", "Poor")))
```

Variables not converted to character before the creation of the design object can be hastily converted using the `as.character` function within the `svymean` command. Also notice the absence of the `na.rm` argument, since missing values were not assigned for any cases of the *REGION06* variable.

```
# % adults living in census regions
# (ex. northeastern 18.64%)
> svymean(~ as.character(REGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17))
```

Weighted frequencies and totals

The `svytotal` and `svytable` functions can be used to calculate both weighted aggregate values and population estimates.

```
> #total 2006 health expenditures
> #all non-institutionalized adults
> # ($892 billion)
> svytotal(~ TOTEXP06,
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #num adults in census regions
> # (ex. northeastern 41.47 million)
> svytable(~ REGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #alternate method
> svytotal(~ as.character(REGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17))
```

Medians and quantiles

The `svyquantile` function calculates each of the percentiles specified in the `quantile` argument. For example, a value of 0.5 generates the median. However, any number of quantiles can be produced with by specifying a vector of values.

```
> #median 2006 health expenditure
> svyquantile(~ TOTEXP06,
```

Table 2: Comparison of Balanced Repeated Replication Standard Error Calculation Methods

		R	Stata	Stata (MSE)	SUDAAN
All Adults	Code	`svymean(~TOTEXP06, subset( meps.brr.dsgn, AGE06X > 17))`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) svy: mean TOTEXP06`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) mse svy: mean TOTEXP06`	`proc descript data = "adult" filetype=sasxport design=brr notsorted; repwgt BRR1 - BRR64 ; weight perwt06f; var totexp06;`
	Mean	4009.1	4009.1	4009.1	4009.1
	SE	83.0	83.0	83.6	83.6
Non-Elderly Adults	Code	`svymean(~TOTEXP06, subset( meps.brr.dsgn, AGE06X > 17 & AGE06X < 65))`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) svy: mean TOTEXP06 if AGE06X < 65`	`svyset [pweight=PERWT06F], brrweight(BRR*x) vce(brr) mse svy: mean TOTEXP06 if AGE06X < 65`	`proc descript data = "adult" filetype=sasxport design=brr notsorted; repwgt BRR1 - BRR64 ; weight perwt06f; var totexp06; subpopn age06x < 65;`
	Mean	3155.4	3155.4	3155.4	3155.4
	SE	77.0	77.0	77.5	77.5

```
+   subset(meps.tsl.dsgn, AGE06X > 17), 0.5)

> #calculate expenditure percentiles
> svyquantile(~ TOTEXP06,
+   subset(meps.tsl.dsgn, AGE06X > 17),
+   c(0.1, 0.25, 0.5, 0.75, 0.9))
```

Although there are a number of other methods for constructing an interval estimate, the Taylor-series linearization design object will not produce standard errors in `svyquantile` since the CDF is not differentiable.[10] Median standard errors can be obtained normally if a replicate weighting design object exists for the data.

```
> #brr method: median health $
> svyquantile(~ TOTEXP06,
+   subset(meps.brr.dsgn, AGE06X > 17), 0.5)
```

If replicate weights do not exist for the data set, a jackknife replication design can be created by converting the Taylor-series linearization design object.

```
> #convert tsl to jackknife design
> meps.jackknife.dsgn <-
+   as.svrepdesign(meps.tsl.dsgn, "auto")

> #jackknife method: median health $
> svyquantile(~ TOTEXP06,
+   subset(meps.jackknife.dsgn, AGE06X > 17), 0.5)
```

Although this jackknife technique might produce a ballpark standard error statistic, recommended quantile standard error calculations can vary depending on the data set and analysis method. Some textbooks simply estimate median standard error as 25% larger than the mean standard error.[11] Therefore, technical documentation and/or data support staff should be consulted before relying on this survey design conversion.

Crosstabulation of totals, means, and quantiles

The `svyby` command repeats a specified function, iterating across the set of distinct values in another factor variable also within the data frame. This function is comparable to `lapply`, which runs across individual elements contained in an external vector. Rather than running the indicated function on the entire data set, `svyby` analyzes discrete subpopulations.

```
> #health spending by census region
> # (ex. northeast $173 billion)
> svyby(~ TOTEXP06, ~ REGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17), svytotal)

> #regional mean spending on health
> # (ex. northeast $4,171)
> svyby(~ TOTEXP06, ~ REGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean)

> #stratified by health status
> # (ex. northeast poor hlth $18,038)
> svyby(~ TOTEXP06, ~ REGION06 + RTHLTH53,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean)

> #top quartile by health status
> # (ex. poor health $19,130)
> svyby(~ TOTEXP06, ~ RTHLTH53,
+   design = subset(meps.tsl.dsgn, AGE06X > 17),
+   FUN = svyquantile, quantiles = 0.75, ci = TRUE)
```

Crosstabulation of distributions

In addition to the straightforward distributions presented using the `svymean` function, the `svyby` function allows row and column percents to be calculated using any desired factor variable(s).

```
> #dist of hlth stat by census region
> svyby(~ cREGION06, ~cRTHLTH53,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean)

> #dist of census region by hlth stat
> svyby(~ cRTHLTH53, ~cREGION06,
+   subset(meps.tsl.dsgn, AGE06X > 17), svymean,
+   na.rm = TRUE)
```

Unlike row and column percents, table percents do not require the `svyby` command and can be calculated using the `interaction` function. This approach is principally useful when one wishes to examine a sub-subpopulation in relation to the entire population, rather than in relation to its parent subpopulation.

```
> #total % in each hlth stat & region
> svymean(~ interaction(cRTHLTH53, cREGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17),
+   na.rm = TRUE)

> #repeat above with weighted counts
> svytotal(~ interaction(cRTHLTH53, cREGION06),
+   subset(meps.tsl.dsgn, AGE06X > 17),
+   na.rm = TRUE)
```

Statistical tests

Though most individual estimates are more easily compared using a general difference formula, the `svyttest` function can be used to conduct quick two-sample t-tests on any binary value that can be generated in the data set. For example, to assess statistical differences in health spending between adults in a particular census region of the United States and adults living in all other regions, one can rapidly create a binary variable to test for spending differences.

[10] http://faculty.washington.edu/tlumley/survey/html/svyquantile.html
[11] http://davidmlane.com/hyperstat/A106993.html

```
> #spending in northeast vs. others
> svyttest(TOTEXP06 ~ (cREGION06 == 1),
+   subset(meps.tsl.dsgn, AGE06X > 17 ))

> #spending in midwest vs. others
> svyttest(TOTEXP06 ~ (cREGION06 == 2),
+   subset(meps.tsl.dsgn, AGE06X > 17 ))
```

Using this method, any Boolean expression can be tested for differences between groups. The examples below show that adults aged 45 to 54 do not have significantly higher or lower spending than the set of all other adults, while pre-retirees (aged 55 to 64r) do spend significantly more than all other adults.

```
> #test 45-54 vs. all others
> svyttest(TOTEXP06 ~ (AGECAT == 4),
+   subset(meps.tsl.dsgn, AGE06X > 17))

> #test 55-64 vs. all others
> svyttest(TOTEXP06 ~ (cAGECAT == '55-64'),
+   subset(meps.tsl.dsgn, AGE06X > 17 ))
```

Conclusion

This article has outlined the capacity to prepare and accurately analyze health policy data with complex survey designs using R. By highlighting the necessary steps to transition from the proprietary statistical package code provided in the technical documentation of the MEPS-HC data, this article equips R users with the conversion knowledge required to conduct important research on health policy data sets. The data manipulation and analysis techniques proven to work on MEPS can be generalized to the majority of available health policy data sets.

As a secondary goal behind contributing to the health policy analyst's toolkit, this article has laid the groundwork for health-care survey administration organizations to follow the lead of the National Center for Health Statistics by including instructions and example code for R in future versions of their technical documentation.[12]

Version notes

The calculations in this paper were performed with R version 2.9.0, using the R survey package version 3.16. Statistical package comparisons were calculated using SAS version 9.1.3 Service Pack 2, Stata/MP version 9.2, and SUDAAN release 10.0.0.

Acknowledgements

I gratefully acknowledge Dr. Thomas Lumley, Professor of Biostatistics at the University of Washington, Seattle and author of the R survey package for assistance with the replication design methodology, and Dr. Roger Peng, Professor of Biostatistics at Johns Hopkins School of Public Health for code editing.

Anthony Damico
Statistical Analyst
Kaiser Family Foundation
adamico@kff.org

[12] ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/NHIS/2008/srvydesc.pdf