Survey Data Analysis and Visualization in R

Brady T. West, Ph.D.
Survey Research Center
Institute for Social Research (ISR)
University of Michigan – Ann Arbor
bwest@umich.edu

A 2017 AAPOR Webinar

1

Webinar Objectives

- Identify R packages and procedures that should be used for analyzing and visualizing survey data
- Import survey data into R, download and install the correct packages, and then write R syntax to perform specific types of analyses and generate specific types of plots
- Understand how to communicate the results of these analyses, and interpret both the estimates and the plots generated

2

Webinar Format

- In this webinar, we will consider several examples using the R computing environment
- R syntax can be entered interactively from a text editor (e.g., WinEdt); **R Studio** is another popular tool for using R (www.rstudio.com)
- It is good practice to keep a history of any R syntax that works correctly in a text editor, so that you can re-run it at a later date!
- Also, make sure to **Save Workspaces!**

3

The R Software

- R is both a computing language and an environment for statistical computing and statistical graphics
- R is **FREE**, open source software
- The origins of R were in development of the S computing language
- S-PLUS is a similar commercial software package; R can do many if not all of the same things, with some time and effort!

4

The R Software, cont'd

- The R Development Project Web Page:
  http://www.r-project.org
- This web page contains a plethora of information about the R software
- The actual software is downloaded from a Comprehensive R Archive Network (CRAN) mirror (note the CRAN link on the left side of the page)

5

Downloading the R Software

- From the R web page, click the CRAN link on the left side of the page
- Select an appropriate mirror for your location (any U.S.A. location is fine)
- Click the link Download R for Windows (or for any other platform)
- Click base (for the base R software)

6

Downloading the R Software, cont'd

- This will download an executable file, and running the file will start a Wizard to guide you through the installation
- After the software has been installed, you should see an R shortcut on your desktop
- Double-click the shortcut to start R!

7

The R Environment

- When starting R, you should see:
  - The RGui (R Graphical User Interface), including menus and "quick" buttons
  - The R Console (the `red prompt >` is where R syntax can be interactively entered and processed)
- By default, all windows (including the R Console) will be nested within the RGui

8

Adding Contributed Packages to R

- The R software automatically comes with several standard **packages**
- These packages generally contain several **functions** enabling standard data analysis and data management techniques
- Scientists use the R language to write their own functions, implementing specialized statistical methods, and include these functions in **contributed packages** that can be downloaded, installed, and used!

9

Adding Contributed Packages to R, cont'd

- **Steps in Installing a Contributed Package:**
  - Identify the name of the package containing the functions that you want to use (case sensitive) from a research article or a colleague (e.g., `survey`)
  - In the RGui, select **Packages** and then **Install package(s)…**
  - Select a CRAN Mirror
  - Find your desired package (the list is QUITE LONG), and click OK
  - R does the rest!

10

Adding Contributed Packages to R, cont'd

- What if you don't have an internet connection?
  - R users can also download contributed packages from a CRAN mirror (the contrib link, instead of base), in .zip format
  - Suppose that a colleague sends you one of these .zip files…
  - From the **Packages** menu, select **Install package(s) from local zip files…**
  - Browse to your .zip file, and R does the rest!

11

Adding Contributed Packages to R, cont'd

- After a package is installed, it has to be **loaded** into R for its functions to work
- The `library()` function (or the `require()` function) is used to **load** a package that has been installed
- An example of loading the contributed `survey` package for this webinar (assuming that this package has been **installed**):

```
> library(survey)
```

12

Adding Contributed Packages to R, cont'd

- Once a package has been loaded, the functions available within the package are ready to use!
- What happens if you try to load a package that has not been installed?

```
> library(survey)
Error in library(survey) : there is no
  package called 'survey'
```

- What happens if you try to use a function from a package that has not been loaded?

```
> lmer(depvar ~ indvar)
Error: could not find function "lmer"
```

13

Finding Help in R

- The RGui **Help** Menu:
  - Quick notes on working in the Console
  - FAQs
  - Downloading PDF manuals
  - Links to R-related web pages
  - Help on specific functions

14

Finding Help in R, cont'd

- Function-specific help:
  - At the console prompt, type:
    `> help(function.name)`
    or
    `> ?function.name`
  - This will open up a detailed help window specific to the function indicated
  - To see the open-source R language underlying a function (which can be modified!), type:
    `> fix(function.name)`
- Try this for the `lm()` function, which fits linear regression models: `> fix(lm)`

15

Analysis of Survey Data

- Survey researchers are often interested in analyzing survey data sets collected from large, nationally representative, probability samples of individuals or establishments
- **Example:** the National Health and Nutrition Examination Survey (NHANES)
- Specialized analytic methods are needed to take the complex design features of these samples into account when making unbiased inferences about finite populations

16

Analysis of Survey Data, cont'd

- Many researchers believe that specialized tools for survey data analysis are only available in large general purpose statistical software packages like SAS, SPSS, Stata, and SUDAAN
- Fortunately, all of these specialized analyses have been implemented in the **`survey`** package within R!
- We now consider examples of analyses of survey data that are possible using functions within this package; we will not go into great detail about all possible options

17

Descriptive Analysis of Survey Data

- Step 1: install the survey package, and then load the package:

```
> require(survey)
```

- Step 2: identify the complex sample design features (sampling stratum codes, sampling cluster codes, and final sampling weights) of the survey data set that you are working with:

```
> require(foreign)
> nhanes <- read.dta("C:\\nhanes.dta")
# note: id -> cluster codes, strata -> stratum
# codes, weights -> weights, nest -> are
# clusters nested within strata?
> nhanes.dsgn <- svydesign(id=~ppsu,
strata=~stratum, weights=~fwgtexam, data=nhanes,
nest=TRUE)
> summary(nhanes.dsgn)
```

18

Descriptive Analysis of Survey Data, cont'd

- **Step 3:** use the function appropriate for the estimate that you wish to compute, and identify the design features to the function:

```
> svymean(~bpsyst,nhanes.dsgn,na.rm=T)
         mean     SE
bpsyst 121.68 0.5255
# note: for estimated proportions
> svymean(~factor(gender),nhanes.dsgn,na.rm=T)
                   mean     SE
factor(gender)1 0.48666 0.0039
factor(gender)2 0.51334 0.0039
> svytotal(~factor(gender),nhanes.dsgn,na.rm=T)
                    total      SE
factor(gender)1  99062020 1824237
factor(gender)2 104491706 1917239
```

19

Descriptive Analysis of Survey Data, cont'd

- **Step 4:** if you wish to focus inferences on subsets, make sure to use the `svyby()` function with the estimator specified rather than deleting cases!

```
> svyby(~bpsyst, ~race, nhanes.dsgn, na.rm=T, svymean)
  race   bpsyst        se
1    1 121.8716 0.5809082
2    2 121.1912 0.6613768
3    3 117.1255 0.8950235
> confint(svymean(~bpsyst, subset(nhanes.dsgn, race ==
   1), na.rm=T))
         2.5 %   97.5 %
bpsyst 120.7331 123.0102
> svyby(~factor(gender), ~race, nhanes.dsgn, na.rm=T,
   svymean) # use vartype = "ci" for confidence intervals
  race factor(gender)1 factor(gender)2 se.factor(gender)1 se.factor(gender)2
1    1       0.4873625       0.5126375        0.004108989        0.004108989
2    2       0.4683368       0.5316632        0.012730301        0.012730301
3    3       0.5464037       0.4535963        0.029494571        0.029494571
```

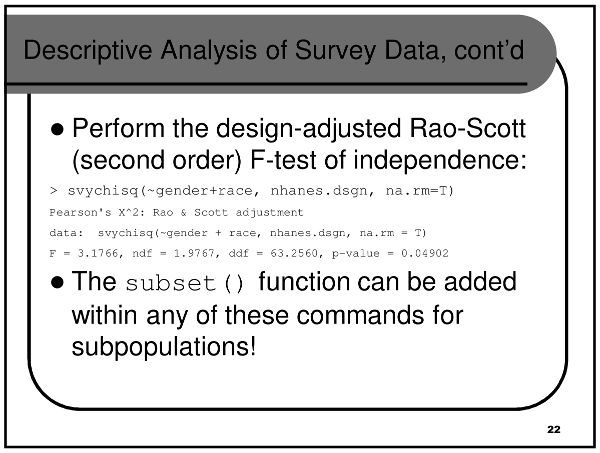
20

Descriptive Analysis of Survey Data, cont'd

- Contingency table analyses of associations between categorical variables:

```
> svytable(~gender+race, nhanes.dsgn, Ntotal=1.0)
         race
gender          1          2          3
     1 0.41762385 0.05488138 0.01415754
     2 0.43928217 0.06230220 0.01175286
# for row percentages
> svyby(~factor(race),~gender,nhanes.dsgn,na.rm=T,vartype="se",svymean)
 gender factor(race)1 factor(race)2 factor(race)3 se.factor(race)1
1      1     0.8581381     0.1127709    0.02909107       0.01822180
2      2     0.8557380     0.1213670    0.02289501       0.01554331
         se.factor(race)2 se.factor(race)3
   1       0.01347439      0.012176382
   2       0.01321531      0.007385834
```

21



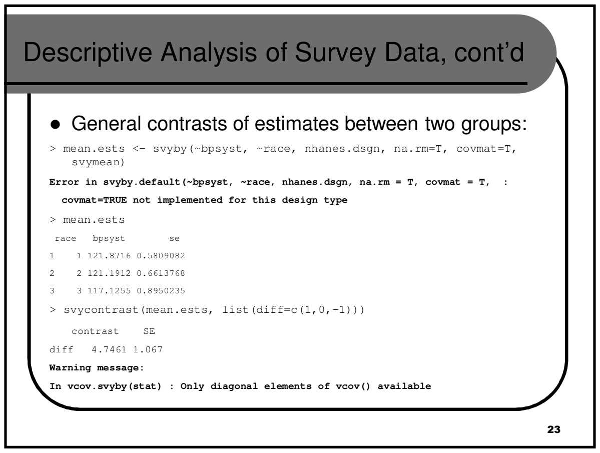
Descriptive Analysis of Survey Data, cont’d

- Perform the design-adjusted Rao-Scott (second order) F-test of independence:

```
> svychisq(~gender+race, nhanes.dsgn, na.rm=T)
Pearson's X^2: Rao & Scott adjustment
data:  svychisq(~gender + race, nhanes.dsgn, na.rm = T)
F = 3.1766, ndf = 1.9767, ddf = 63.2560, p-value = 0.04902
```

- The `subset()` function can be added within any of these commands for subpopulations!

22

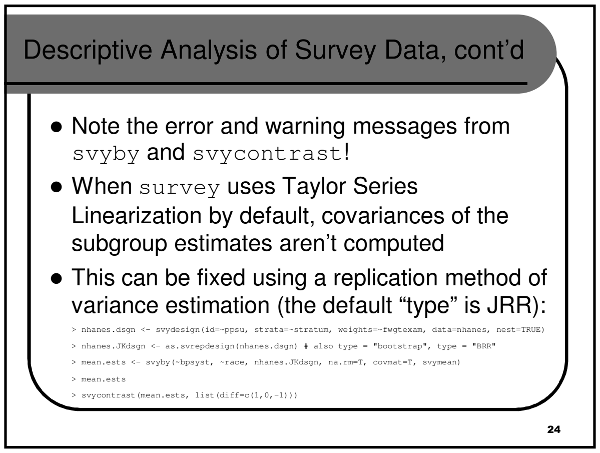


Descriptive Analysis of Survey Data, cont’d

- General contrasts of estimates between two groups:

```
> mean.ests <- svyby(~bpsyst, ~race, nhanes.dsgn, na.rm=T, covmat=T,
   svymean)
Error in svyby.default(~bpsyst, ~race, nhanes.dsgn, na.rm = T, covmat = T,  :
  covmat=TRUE not implemented for this design type
> mean.ests
 race   bpsyst        se
1    1 121.8716 0.5809082
2    2 121.1912 0.6613768
3    3 117.1255 0.8950235
> svycontrast(mean.ests, list(diff=c(1,0,-1)))
   contrast    SE
diff   4.7461 1.067
Warning message:
In vcov.svyby(stat) : Only diagonal elements of vcov() available
```

23



Descriptive Analysis of Survey Data, cont’d

- Note the error and warning messages from `svyby` and `svycontrast`!
- When `survey` uses Taylor Series Linearization by default, covariances of the subgroup estimates aren’t computed
- This can be fixed using a replication method of variance estimation (the default “type” is JRR):

```
> nhanes.dsgn <- svydesign(id=~ppsu, strata=~stratum, weights=~fwgtexam, data=nhanes, nest=TRUE)
> nhanes.JKdsgn <- as.svrepdesign(nhanes.dsgn) # also type = "bootstrap", type = "BRR"
> mean.ests <- svyby(~bpsyst, ~race, nhanes.JKdsgn, na.rm=T, covmat=T, svymean)
> mean.ests
> svycontrast(mean.ests, list(diff=c(1,0,-1)))
```

24

Descriptive Analysis of Survey Data, cont'd

- Here is the new output (no warnings):

```
> mean.ests <- svyby(~bpsyst, ~race, nhanes.JKdsgn, na.rm=T,
   covmat=T, svymean)
> mean.ests # note how similar the JRR and TSL SEs are…
  race   bpsyst        se
1    1 121.8716 0.5809259
2    2 121.1912 0.6620447
3    3 117.1255 0.9229559
> svycontrast(mean.ests, list(diff=c(1,0,-1)))
     contrast     SE
diff   4.7461 1.1798
```

25

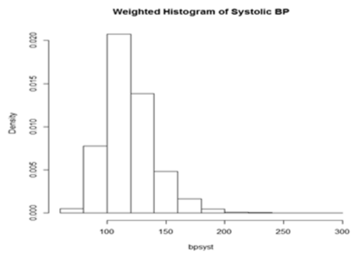
Visualizing Survey Data

- Plots of survey data need to recognize the possibility that sampled observations have different survey weights
- We wish to visualize what the **population** looks like by employing the weights!
- We will now consider examples of functions for visualizing survey data that incorporate the survey weights defined for a design object
- First, consider a weighted histogram for a continuous variable:

```
> svyhist(~bpsyst, nhanes.dsgn, main="Weighted
Histogram of Systolic BP")
```

26

Visualizing Survey Data



27

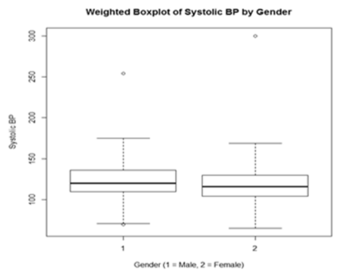
Visualizing Survey Data

- Next, consider a weighted side-by-side box plot for comparing distributions on a continuous variable between two groups:

```
> svyboxplot(bpsyst ~ factor(gender), nhanes.dsgn,
main="Weighted Boxplot of Systolic BP by Gender",
ylab="Systolic BP", xlab="Gender (1 = Male, 2 =
Female)")
```

28

Visualizing Survey Data



29

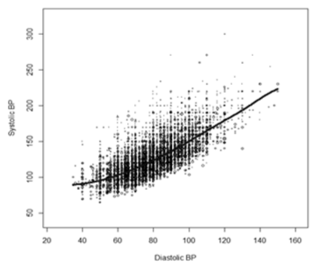
Visualizing Survey Data

- Next, consider a scatter plot with the individual data points weighted by their respective survey weights
- We combine this with a weighted smoothing function that visualizes the general (non-parametric) functional relationship between these two variables

```
> svyplot(bpsyst ~ bpdiast, design=nhanes.dsgn,
  xlab="Diastolic BP", ylab="Systolic BP")
> smoother <- svysmooth(bpsyst ~ bpdiast,
  design=nhanes.dsgn, bandwidth=10)
> lines(smoother, col="blue", lwd=4)
```

30

Visualizing Survey Data



31

Fitting Regression Models to Survey Data

- The survey package includes functions for fitting regression models as well!
- Available functions for fitting regression models to complex sample survey data:
  - `svyglm()`: generalized linear models, including linear, logistic, Poisson, etc.
  - `svycoxph()`: Cox proportional hazards regression models

32

Fitting Models to Survey Data, cont'd

- The `svyglm()` function operates in a very similar manner to the `lm()` and `glm()` functions introduced earlier
- Linear Regression Model Example:

```
> fit.surv <- svyglm(bpsyst ~ ager +
factor(race) + factor(gender), nhanes.dsgn)
> summary(fit.surv)
```

33

Fitting Models to Survey Data, cont'd

```
Call:
svyglm(formula = bpsyst ~ ager + factor(race) + factor(gender),
    nhanes.dsgn)

Survey design:
svydesign(id = ~ppsu, strata = ~stratum, weights = ~fwgtexam,
    data = nhanes, nest = TRUE)

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    101.9310     0.7705 132.291  < 2e-16 ***
ager             0.6581     0.0163  40.382  < 2e-16 ***
factor(race)2    1.9284     0.5479   3.520  0.00150 **
factor(race)3   -3.7128     1.1655  -3.186  0.00353 **
factor(gender)2 -5.3403     0.3817 -13.991 3.67e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 347.2067)
```

34

Fitting Models to Survey Data, cont'd

- Logistic Regression Model Example:

```
> nhanes$highsbp <- 0
> nhanes$highsbp[nhanes$bpsyst>120] <- 1
> nhanes.dsgn <- svydesign(id=~ppsu,
strata=~stratum, weights=~fwgtexam, data=nhanes,
nest=TRUE)
> fit.logit <- svyglm(highsbp ~ ager +
factor(race) + factor(gender), nhanes.dsgn,
family = binomial(link = "logit"))
> summary(fit.logit)
```

35

Fitting Models to Survey Data, cont'd

```
Call:
svyglm(formula = highsbp ~ ager + factor(race) + factor(gender),
    nhanes.dsgn, family = binomial(link = "logit"))

Survey design:
svydesign(id = ~ppsu, strata = ~stratum, weights = ~fwgtexam,
    data = nhanes, nest = TRUE)

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -2.309620   0.092798 -24.889  < 2e-16 ***
ager             0.066085   0.001357  48.714  < 2e-16 ***
factor(race)2    0.094307   0.071470   1.320  0.19768
factor(race)3   -0.490113   0.135906  -3.606  0.00119 **
factor(gender)2 -0.752593   0.043982 -17.111 2.32e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Dispersion parameter for binomial family taken to be 0.9640284)

Number of Fisher Scoring iterations: 4
```

36

Fitting Models to Survey Data, cont'd

- Cox Proportional Hazards Model Example:

```
> require(survival)
> data(pbc, package="survival")
> pbc.n <- pbc[pbc$status != 1,]
> pbc.n$status[pbc.n$status == 2] <- 1
> dpbc <- svydesign(id=~1,
strata=~edema, data=subset(pbc.n,trt>0))
> fit.cox <- svycoxph(Surv(time,status)
~ protime + albumin, design=dpbc)
> summary(fit.cox)
```

37

Fitting Models to Survey Data, cont'd

```
Stratified Independent Sampling design (with replacement)
svydesign(id = ~1, strata = ~edema, data = subset(pbc.n, trt >  0))
Call:
svycoxph(formula = Surv(time, status) ~ protime + albumin, design = dpbc)

  n= 293, number of events= 125

           coef exp(coef) se(coef)      z Pr(>|z|)
protime  0.3995    1.4910   0.1013  3.943 8.03e-05 ***
albumin -1.6649    0.1892   0.2257 -7.377 1.61e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

        exp(coef) exp(-coef) lower .95 upper .95
protime    1.4910     0.6707    1.2225    1.8185
albumin    0.1892     5.2852    0.1216    0.2945
```

38

A Tremendous Online Resource!

- This is, quite simply, one of the best online resources out there for survey researchers interested in using R:

http://cran.r-project.org/web/views/OfficialStatistics.html

- Packages of particular interest to analysts of complex sample survey data:
  - `lavaan.survey` (fitting SEM models)
  - `rpms` (fitting classification and regression trees)

39

References

Heeringa, S.G., West, B.T., and Berglund, P.A. (2017). *Applied Survey Data Analysis, Second Edition.* Chapman Hall / CRC Press: Boca Raton, FL.

Lumley, T. (2010). *Complex Surveys: A Guide to Analysis using R.* Wiley.

Murrell, P. (2011). *R Graphics, Second Edition.* Chapman & Hall / CRC.

40

Questions (and Answers?)

- Hopefully we will have time for a short question and answer session!
- Some example questions:

1. How do we change variance estimation procedures "on the fly"?
2. What types of complex sample designs can R accommodate?

41

Thank You!

- I hope that you have enjoyed this AAPOR webinar, and I wish you luck with using R for your survey analysis and visualization purposes!
- Please feel free to email me at **bwest@umich.edu** if you have questions.

42

R syntax:

```
require(survey)
require(foreign)

nhanes <- read.dta("X:\\Brady\\Presentations\\AAPOR\\2017 Webinars\\nhanes.dta")

nhanes.dsgn <- svydesign(id=~ppsu, strata=~stratum, weights=~fwgtexam, data=nhanes, nest=TRUE)
summary(nhanes.dsgn)

svymean(~bpsyst,nhanes.dsgn,na.rm=T)
svymean(~factor(gender),nhanes.dsgn,na.rm=T)
svytotal(~factor(gender),nhanes.dsgn,na.rm=T)

svyby(~bpsyst, ~race, nhanes.dsgn, na.rm=T, svymean)
svymean(~bpsyst, subset(nhanes.dsgn,race == 1), na.rm=T)
svyby(~factor(gender), ~race, nhanes.dsgn, na.rm=T, vartype = "ci", svymean)

confint(svymean(~bpsyst, subset(nhanes.dsgn,race == 1), na.rm=T))

svytable(~gender+race, nhanes.dsgn, Ntotal=1.0)
svyby(~factor(race), ~gender, nhanes.dsgn, na.rm=T, vartype = "se", svymean)
svychisq(~gender+race, nhanes.dsgn, na.rm=T)

mean.ests <- svyby(~bpsyst, ~race, nhanes.dsgn, na.rm=T, covmat=T, svymean)
mean.ests
svycontrast(mean.ests, list(diff=c(1,0,-1)))

nhanes.dsgn <- svydesign(id=~ppsu, strata=~stratum, weights=~fwgtexam, data=nhanes, nest=TRUE)
nhanes.JKdsgn <- as.svrepdesign(nhanes.dsgn)

mean.ests <- svyby(~bpsyst, ~race, nhanes.dsgn, na.rm=T, covmat=T, svymean)
mean.ests
svycontrast(mean.ests, list(diff=c(1,0,-1)))

svyhist(~bpsyst,nhanes.dsgn,main="Weighted Histogram of Systolic BP")

svyboxplot(bpsyst ~ factor(gender), nhanes.dsgn, main="Weighted Boxplot of Systolic BP by Gender",
ylab="Systolic BP", xlab="Gender (1 = Male, 2 = Female)")

svyplot(bpsyst ~ bpdiast, design=nhanes.dsgn, xlab="Diastolic BP", ylab="Systolic BP")
smoother <- svysmooth(bpsyst ~ bpdiast, bandwidth=10)
lines(smoother, col="blue")

fit.surv <- svyglm(bpsyst ~ ager + factor(race) + factor(gender), nhanes.dsgn)
summary(fit.surv)

nhanes$highsbp <- 0
nhanes$highsbp[nhanes$bpsyst>120] <- 1
nhanes.dsgn <- svydesign(id=~ppsu, strata=~stratum, weights=~fwgtexam, data=nhanes, nest=TRUE)

fit.logit <- svyglm(highsbp ~ ager + factor(race) + factor(gender), nhanes.dsgn, family = binomial(link = "logit"))
summary(fit.logit)

require(survival)
data(pbc, package="survival")
pbc.n <- pbc[pbc$status != 1,]
pbc.n$status[pbc.n$status == 2] <- 1
dpbc <- svydesign(id=~1, strata=~edema, data=subset(pbc.n,trt>0))
fit.cox <- svycoxph(Surv(time,status) ~ protime + albumin, design=dpbc)
summary(fit.cox)
```

Web Sites:

http://www.rstudio.com
http://www.r-project.org
http://cran.r-project.org/web/views/OfficialStatistics.html