New York City
Housing and Vacancy Survey

Guide to Estimating Variances

6/13/2024

United States® Census Bureau

census.gov
@uscensusbureau

Table of Contents

1. Introduction

The New York City (NYC) Housing and Vacancy Survey (NYCHVS) collects important information including unit, building, and neighborhood conditions; housing and utility costs; and information about the people who live in the sampled address. Starting in 2017, the NYCHVS Public Use Files (PUFs) include the addition of replicate weights. The replicate weights allow data users an additional tool to calculate estimates of variance. Using the information provided in this guide and the replicate weights on the PUFs, data users will have the necessary tools to compute estimates of variance using the replicate weights.

Only Sample Design Variances Estimated

By variance, we refer to the sample design variance or simply design variance or the variance from a finite population sample. The variance measured by the replicate weights represents the variance of the estimated statistic if we repeated the sample selection many times and estimated the statistic of interest with each sample. See textbooks by Cochran (1977), Wolter (2007), and Särndal, Swensson, and Wretman (1992) for detailed discussions on sample design and sample variances.

How the Guide is Organized

Section 2 of the guide describes "how to estimate variances yourself using replicate weights." This part of the guide explains how the new variance estimation tools provided by the NYCHVS can be used to estimate variances.

Replicate weights can be used to estimate sampling variance for any complex statistic from the survey design. The bulk of the guide – Sections 3 to 6 – reviews several examples that show how to use replicate weights to estimate variances of several types of statistics. The examples are repeated in dedicated sections for the following software packages: SAS, STATA, R, and Python. We include two separate sections on SAS: one section describes the PROC SURVEY procedures and another section describes how to use base SAS to calculate the variances.

Table 1 summarizes the examples and the four main sections of the guide.

Table 1: Summary Table of Examples

Type of Statistics	Description of Specific Example	Example Number/Section				
		SAS PROC SURVEY	Base SAS	STATA	R	Python
Total	Number of 2021 occupied housing units in NYC	2.1	4.1	5.1	6.1	7.1
Difference	Difference of the number of 2021 public housing units in Manhattan and the Bronx		4.2	5.2	6.2	7.2
Mean	2021 average gross rent for renter-occupied housing units in NYC	3.1	4.3	5.3	6.3	7.3
Median	2021 Median gross rent for renter-occupied housing units in NYC		4.4	5.4	6.4	7.4
Odds Ratio	Odds of rent stabilization status (pre-1947 or post-1947) with three or more maintenance deficiencies	9.1, 9.2	9.3	9.4	9.5	9.6
Regression Coefficient	The regression coefficient representing the year the householder moved into the housing unit for renters.	3.3	4.5	5.5	6.5	7.5
Longitudinal Change[1]	The difference of the 2021 and 2017 gross vacancy rates		4.6	5.6	6.6	7.6
Percent Change[1]	Percent change in the median gross rent from 2017 to 2021		4.7	5.7	6.7	7.7

To further demonstrate the importance of using the replicate weights in variance estimation, Section 8 provides a general example of confidence interval calculations and then Section 9 provides a detailed example using the odds ratio and confidence interval calculations.

Section 10 explains how the replicate weights are calculated. We provide this section for transparency, context, and background for data users.

Scope of the Guide

The scope of this guide primarily includes the data provided in the 2021 NYCHVS. Two examples demonstrate longitudinal estimates which incorporates 2017 NYCHVS data.

The methods provided in this guide can be used for both housing unit (HU) estimates as well as person estimates. Even though examples of person estimates are not provided, the same methods can be applied with the substitution of the person replicate weights.

[1] The longitudinal change and percent change examples for the years of 2021 and 2023 for SAS, STATA, R and Python are included in Appendix B.

The results provided in this guide, including the estimations, standard errors, and confidence intervals, are approximations and are subject to errors. We do not provide a review of the sample design for the NYCHVS but refer the reader to the documentation of the survey for a more thorough discussion of the design and errors: 2021 New York City Housing and Vacancy Survey Sample Design, Weighting, and Error Estimation (U.S. Census Bureau, 2023).

While we provide code for handling all examples in four programming languages – SAS, STATA, R, and Python – we note that, as of this publication, the 2017 NYCHVS data needs to be read in via SAS or STATA first to be converted to a CSV file. There is likely another way to read in this data using just R or Python, but its text-file format makes this harder. This caveat aside though, all examples should be replicable with any of the software provided using publicly-available data sets provided by NYCHVS and the U.S. Census Bureau.

2. General Variance Estimation for New York City Housing and Vacancy Survey

The variance of any survey estimate based on a probability sample may be estimated by the method of replication. This method requires that the sample selection, the collection of data, and the estimation procedures be independently carried through (replicated) several times. Each time the sample is replicated, a different set of estimates is calculated. The dispersion of the resulting estimates then can be used to measure the variance of the sample.

However, we would not consider repeating any large survey, such as the NYCHVS, several times to obtain variance estimates. A practical alternative is to alter the sample several times by applying different weighting factors to the sample units. The alterations of the replicate weights allow the single sample to represent multiple replicate samples that can be used to estimate variances. We sometimes refer to the replicate samples as simply replicates. For the NYCHVS, we used a total of 80 replicates to calculate the NYCHVS variance estimates.

The replicate weights should only be used in estimating variances and should not be used to create independent estimates. The final weights (FW) are provided to produce all point estimates.

The user should also note that the 2021 NYCHVS replicate weights are applicable for use only on 2021 NYCHVS data. Replicate weights for 2017 are applicable for use with the 2017 NYCHVS PUF only, and so on.

Use of Replicate Estimates in Variance Calculations

Calculate variance estimates using the replication variance estimator:

$$\hat{v}(\hat{\theta}) = \frac{4}{80}\sum_{r=1}^{80}(\hat{\theta}_r - \hat{\theta}_0)^2 \qquad (2.1)$$

where $\hat{\theta}$ is the weighted estimate of the statistic of interest using the final weight for the sample and $\hat{\theta}_r$ is the replicate estimate for replicate *r* of the same statistic using the r^{th} replicate weight. The estimator $\hat{\theta}_0$ is the point estimate of θ. Technically, replicate weights allow one to use the mean of the replicate estimators to estimate $\hat{\theta}_0$. Throughout this document however, we suggest estimating $\hat{\theta}_0$ using the final weights only, since this produces a more conservative estimate of the variance.

The value of 80 in Equation (2.1) is the number of replicates used by NYCHVS. For more explanation about the factor of 4, see the end of Section 10. See also Fay and Train (1995), Ash

(2014), and Opsomer, Breidt, White, and Li (2016) for more background about Successive Difference Replication (SDR).

To ensure confidentiality of the data, some characteristics have either been bottom coded or top coded. This procedure places a lower or upper boundary on the published value for the variable in question. Therefore, some estimates calculated from the PUFs may differ from the estimates provided in the 2021 NYCHVS Selected Initial Findings (www.nyc.gov/assets/hpd/downloads/excel/2021-nychvs-selected-initial-findings-figures-data.xlsx).

Using Replication to Estimate Variances

The following example illustrates how a statistic would be estimated, replicated, and combined to form a variance estimate. We are going to estimate the variance using the 80 replicate weights provided for the NYCHVS.

Note that in 2021 NYCHVS, the replicate weights for Replicate 1 are equal to the final weights.

Example 2.1. Estimating the Variance of the Total Number of Housing Units in a Domain

The goal of this example is to estimate the total number of occupied HUs in NYC for 2021 and its corresponding estimate of variance. In 2021, we have 7,089 completed interviews that are occupied housing units in NYC. Table 2.1 displays the first four and last interview responses for occupied housing units in NYC for 2021.

Table 2.1: Example of Estimating Variances with Replication

Sample Housing Unit	Tenure	Final Weight	Replicate Weights				
			Replicate 1[2]	Replicate 2	Replicate 3		Replicate 80
1	Renter	154.440	154.440	44.909	266.893	…	263.340
2	Renter	264.709	264.709	80.340	261.718	…	457.946
3	Renter	430.840	430.840	429.362	721.599	….	127.413
4	Renter	200.362	200.362	201.630	336.421	…	194.927
…	…	…	…	…	…		…
N (7,089)	Renter	451.315	451.315	453.975	131.611	…	444.089

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey Public Use Files.

2 For NYCHVS, replicate 1 weights are the same as final weights, so therefore replicate weight 0 = replicate weight 1, and replicate estimate 0= replicate estimate 1.

In the 2021 NYCHVS, the final weight is sometimes referred to as replicate weight 0, and the estimate from the final weight is sometimes referred to as replicate estimate 0.

Step 1: Calculate the weighted survey estimate.

The statistic of interest is the total number of occupied housing units in NYC for 2021. Add the final weights of the sample cases that are occupied; the estimate of total number of occupied housing units is calculated as follows:

Occupied HUs Estimate $\widehat{N}_0$= 154.440 + 264.709 + … + 451.315 = 3,157,105.18

Step 2: Calculate the weighted survey estimate for each of the replicate samples.

The replicate estimates of occupied HUs are:

Replicate estimate 1 $\widehat{N}_{r=1}$= 154.440 + 264.709 + … + 451.315 = 3,157,105.18
Replicate estimate 2 $\widehat{N}_{r=2}$= 44.909 + 80.340 +… + 453.975 = 3,162,978.47
Replicate estimate 3 $\widehat{N}_{r=3}$= 266.893 + 261.718 + … + 131.611 = 3,153,127.53
⋮ ⋮
Replicate estimate 80 $\widehat{N}_{r=80}$= 263.340 + 457.946 + … + 444.089 = 3,168,556.06

Step 3: Use these survey estimates in Equation (2.1) to calculate the variance estimate for the total occupied housing units.

$$\begin{aligned}\hat{v}(\widehat{N}) &= \frac{4}{80}\sum_{r=1}^{80}\left(\widehat{N}_r - \widehat{N}_0\right)^2 \\ &= 0.05 \times [(3{,}157{,}105.18 - 3{,}157{,}105.18)^2 + (3{,}162{,}978.47 - 3{,}157{,}105.18)^2 \\ &\qquad + (3{,}153{,}127.53 - 3{,}157{,}105.18)^2 + \cdots \\ &\qquad + (3{,}168{,}556.06 - 3{,}157{,}105.18)^2] \\ &= 0.05 \times [0 + 34{,}495{,}535.42 + 15{,}821{,}699.52 + \cdots + 131{,}122{,}652.77] \\ &= 180{,}599{,}774.86\end{aligned}$$

The estimate of the variance of total occupied housing units is $\hat{v}(\widehat{N})$ = 180,599,775.

The survey estimate for occupied housing units in NYC is 3,157,105. This survey estimate has an estimated variance of 180,599,775, and its standard error, which is the square root of the estimated variance, is 13,439 housing units.

The three steps of Example 2.1 will be used throughout the guide to calculate variances. Sometimes steps 1 and 2 will be combined since estimating the statistic with the final weight (or for replicate 0) can be done while estimating the statistic with the 80 replicate weights.

Cautions about Domain Analysis Don't Apply

Korn and Graubard (1999; Section 5.4), Lewis (2017; Section 8), Heeringa, West, and Berglund (2010; Section 4.5), and others have provided important cautions about the analysis of domains, sometimes also referred to as subdomains. These cautions are important but do not apply to the analysis described within this Guide since our variance estimation employs replicate weights to estimate variances. When using the Taylor series to estimate variances (the default method of most software packages), the issue of domains is important: SAS will generate incorrect estimates with the `where` statements and correct estimates with the `domain` statement. Similarly with STATA, the **`subpop()`** option is needed. However, with the replicate weights, using the **`subpop()`** statement in STATA or either the `domain` or `where` statements in SAS will produce the same results.

Confidence Intervals and Significance Tests

Once the standard error is calculated, it can be combined with the estimate to calculate confidence intervals. Section 8 provides further instructions on how this can be done.

NYCHVS Public Use File Description

To access the PUF, go to the Census Bureau's NYCHVS website at: https://www.census.gov/programs-surveys/nychvs/data/datasets.html . Once there, select the "2021" tab and select the "2021 New York City Housing and Vacancy Survey Microdata". Here you will find four CSV files.

The four CSV files:

1) All Units Records, which contains all Housing unit records, with both occupied and vacant housing units,
2) Occupied Records, which contains the records for occupied housing units
3) Person Records, which contains the records for persons
4) Vacant Records, which contains the records for vacant housing units

The datasets All Units Records, Occupied Records, Person Records, and Vacant Records (CSV files #1, #2, #3, and #4 respectively) contain the sample estimate with full sample weights as well as the 80 replicate weights for all housing units, occupied housing units, persons, and vacant housing units, respectively. NYCHVS data users should use the All Units Records, Occupied Records, and Vacant Records datasets for any housing unit estimates and use the Person Records dataset for any person estimates.

Data preparation for Running SAS

When reading these files into SAS, first download and save these data files as CSV files on your local drive. Figure 2.1.1 shows how to import the three housing unit files (CSV files #1, #2, and #4) and create the SAS datasets with Proc Import. For generating SAS datasets for Person

Records (CSV file #3), follow the same steps for housing unit records. Finally, note that throughout this document, input related items are highlighted in yellow, such as dataset names and file paths.

Figure 2.1.1: SAS Code for Reading in the Public Use Files

```
*import allunits file;
proc import file="LOCATION OF CSV FILE\allunits_puf_21.csv"
  out=Allunits_puf_21
  dbms=csv;
run;

*import occupied HU file;
proc import file="LOCATION OF CSV FILE\occupied_puf_21.csv"
  out=Occupied_puf_21
  dbms=csv;
run;

*import vacant file;
proc import file="LOCATION OF CSV FILE\vacant_puf_21.csv"
  out=vacant_puf_21
  dbms=csv;
run;
```

For many housing unit estimates, users need to download all three HU data files – All Units Records (CSV file #1), Occupied Records (CSV file #2) and Vacant Records (CSV file #4) – and then combine the three files into one. Note some HU estimates can be derived using only one or two of the datafiles above. For example, the average gross rent for renter-occupied housing units (Example 2.1) can estimated using the Occupied Records file only. However, we will combine three files into one, which simplifies code for the examples below. Figure 2.1.2 shows the SAS code for combining Occupied and Vacant records, and then merging it with All Units.

Figure 2.1.2: SAS Code for Appending Occupied and Vacant Records

```
Data Occupied21;
      set occupied_puf_21;
run;

Data vacant21;
      set vacant_puf_21;
run;

* Combine Occupied and Vacant records for 2021;
Data HU_all21;
      set occupied21
        vacant21;
run;

Data ALL21;
      set Allunits_puf_21;
      keep control occ boro csr yearbuilt;
run;

proc sort data=all21; by control;
proc sort data=HU_all21; by control;
Data HU21;
      merge all21
            HU_all21;
      by control;
run;
```

For prior year NYCHVS data files, data users can go to the same website mentioned on page 7, click the relevant year's tab, and download the PUFs from there. In 2017 and prior, the datafiles were given as text files. These have accompanying import programs for both SAS and STATA software, which are also available on the website to help data users import the datafiles. Note that no All Units file was provided in 2017 and prior years. So, no additional merging step is needed to prepare those data. Figure 2.1.3 shows the partial SAS code for reading in the PUFs for 2017. We saved the 2017 Occupied HU SAS data file as occupied17, and the 2017 Vacant HU SAS data file as vacant17. The 2017 combination file for all housing units is called HU_all17. Note that some of the variable names and/or definitions have been changed in 2021. For the longitudinal examples where we are using different years data, users might notice the differences.

Figure 2.1.3: Partial SAS Code for Reading in the Public Use Files for 2017

```
* Import the Occupied data file. ;
data occupied17;
infile 'LOCATION OF TEXT FILE\uf_17_occ_web_b.txt' lrecl=1334 truncover;
input recid $1 @;
if(recid='1') then do;

***SEE THE REST OF THE SAS CODE IN SAS IMPORT PROGRAM FOUND IN THE CENSUS
BUREAU'S NYCHVS WEBSITE MENTIONED IN PAGE 7*** ;

* Import the Vacant data file. ;
data vacant17;
infile 'LOCATION OF TEXT FILE\uf_17_vac_web_b.txt' lrecl=831 truncover;
input recid $1 @;
if(recid='3') then do;

***SEE THE REST OF THE SAS CODE IN SAS IMPORT PROGRAM FOUND IN THE CENSUS
BUREAU'S NYCHVS WEBSITE MENTIONED IN PAGE 7*** ;

* Combined Occupied and Vacant records for 2017;
Data HU_all17;
      set occupied17
        vacant17;
run;
```

For the housing unit data file, the final weight is stored in variable *FW*, and the replicate weights are stored in variables *FW1-FW80*. For the persons data file, the final weight is stored in variable *PW*, and the replicate weights for persons are stored in variables *PW1-PW80*.

In 2017 and prior, these weights were stored as character variables with five implied decimal places, so they need to be converted before running any analysis. As of 2021, these weights are stored as numeric variables with the correct number of decimal places. So, conversion is no longer needed. Figure 2.1.4 shows the SAS code for converting the weights to numeric for 2017.

Figure 2.1.4: SAS Code for Converting Weights to Numeric

```
* convert weights to numeric with the correct decimal place. ;
Data HU17;
      set HU_all17;
      FW=fw/100000;
   do i=1 to 80 ;
      fwi=fwi /100000 ;
      fw_i=input(fwi, 9.);
      drop fwi;
      rename fw_i=fwi;
  end ;
run;
```

After running the code above, dataset HU17 is created. This dataset, along with the HU21 dataset (created earlier in Figure 2.1.2), will be used throughout this document.

3. Estimating Variances with SAS PROC SURVEY Procedures

Within SAS, there are currently four specialized 'survey' procedures relevant to our examples that use replicate weights directly. Table 3 reviews the SAS SURVEY procedures.

Table 3: Summary of SAS 'SURVEY' PROCs

PROC name	**Can be used to…**
SURVEYMEANS	Calculate basic statistics
SURVEYFREQ	Complete categorical data analysis
SURVEYREG	Complete regression analysis
SURVEYLOGISTIC	Complete logistic regression analysis

Statistical packages often exist in other statistical software packages, but we do not necessarily endorse their use for any given language. Relevant packages in other languages will be discussed at the end of this section. We recommend the data users carefully study documentation on other statistical packages before applying them to NYCHVS.

Because the replicate weights for NYCHVS are calculated with Fay's SDR methods, the SAS SURVEY procedures need to use the `varmethod=brr(fay)` option in the PROC statement. If the Fay option for BRR is not used, the variances will be off by a factor of four.

An unintended consequence of using our replicate weights with the `varmethod=brr(fay)` option is that all of the SAS PROC SURVEY procedures assume that BRR is being used and further that the number of replicates is the number of strata in the sample design. As a result, SAS PROC SURVEY creates confidence intervals using a critical value from a t-distribution with 80 degrees of freedom. The replicate weights that are provided for NYCHVS, however, use SDR methodology that is appropriate for the NYCHVS sample design: systematic random sample from an ordered list. We suggest using a critical value from a normal distribution instead of a t-distribution – see also "Normal Distribution versus the *t*-distribution for Confidence Intervals" in Section 8. The effect is relatively minor either way but, in any case, the choice of t vs. normal is the only difference between the canned SAS procedure and our recommendation.

Mukhopadhyay, An, Tobias, and Watts (2008) provides an excellent review of replication-based variances methods and the "survey" procedures of SAS. Taylor (2016) also provides a comprehensive review of using SAS to analyze survey data.

Each of the procedures of Table 3 can use the replicate weights in its analysis. For example, the SURVEYMEANS procedure and the replicate weight file can be used together to generate a standard error for a population total estimate, as shown in Figure 3.1.

Figure 3.1: SAS Code Using PROC SURVEYMEANS

```
proc surveymeans data=HU21 sum std clsum cvsum varmethod=brr(fay);
 var variable ;
 weight FW ;
 repweights FW1-FW80 ;
run ;
```

Example 3.1. Estimating the Variance of a Mean with SAS PROC SURVEYMEANS

In this example, we estimate the average gross rent for renter-occupied housing units. The output for PROC SURVEYMEANS of SAS will also include the estimated standard error of the average gross rent of renter-occupied HUs.

Figure 3.1.1 provides the SAS code that can be used to estimate the average gross rent of the renter-occupied HUs from the 2021 NYCHVS.

Figure 3.1.1: Example of SAS Code Using PROC SURVEYMEANS

```
Data Data1 ;
 set HU21 ;
      * only renters with valid rent records;
      if tenure=1 and GRENT not in ('-2','-1') ;
run ;

proc surveymeans data=Data1 mean std varmethod=brr(fay) ;
 domain tenure ;
 var Grent ;
 weight FW ;
 repweights fw1-fw80 ;
run ;
```

The SAS code of Figure 3.1.1 generates the output of Figure 3.1.2.

Figure 3.1.2: Partial SAS Output for PROC SURVEYMEANS

Statistics		
Variable	Mean	Std Error of Mean
rent	1726.738952	15.829266

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimate of the average gross rent in renter-occupied HUs is $1,727 with a standard error of $16.

Example 3.2. Estimating the Variance for a Contingency Table with SAS PROC SURVEYFREQ

In this example, we estimate the frequency and proportion of rent stabilization status (pre-1947 or post-1947) with three or more maintenance deficiencies. The output for PROC SURVEYFREQ will also include the standard errors of the estimated quantities. See Section 9 for this same example on calculating odds ratios.

Figure 3.2.1 provides the SAS code for PROC SURVEYFREQ that produces the two-way table of building deficiencies by rent stabilization status.

Figure 3.2.1: Example of SAS Code Using PROC SURVEYFREQ

```
Data Data2 ;
 set HU21 ;
 *subset to rent stabilized renters only;
 where CSR=32 and tenure=1;
 if MDEFCOUNT ge 3 then def='1' ;
 * Redefine HUs that did not report on deficiencies. ;
 else if MDEFCOUNT =-1 then def=' ';
 else def='0';
 if YEARBUILT lt 5 then stabilized='pre-1947' ;
 else if YEARBUILT ge 5 then stabilized='post1947';
run;

proc surveyfreq data=data2 varmethod=brr(fay) NOSUMMARY ;
 tables tenure*def*stabilized ;
 weight FW ;
 repweights fw1-fw80 ;
run ;
```

The SAS code of Figure 3.2.1 generates the output seen in Figure 3.2.2.

Figure 3.2.2: SAS Output for PROC SURVEYFREQ

The SURVEYFREQ Procedure

Variance Estimation	
Method	BRR
Replicate Weights	DATA2
Number of Replicates	80
Fay Coefficient	0.500

Table of def by stabilized

Controlling for TENURE=1

def	stabilized	Frequency	Weighted Frequency	Std Err of Wgt Freq	Percent	Std Err of Percent
0	post1947	531	231825	10166	27.6390	1.0147
	pre-1947	762	402221	13102	47.9542	1.4080
	Total	1293	634046	15546	75.5932	1.2336
1	post1947	95	41640	5467	4.9645	0.6424
	pre-1947	313	163074	10982	19.4423	1.2204
	Total	408	204714	11455	24.4068	1.2336
Total	post1947	626	273465	11902	32.6035	1.1907
	pre-1947	1075	565296	14133	67.3965	1.1907
	Total	1701	838761	16652	100.0000	

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Example 3.3. Estimating the Variance of a Regression Coefficient with SAS PROC SURVEYREG

Correctly estimating the variance of a regression coefficient can be complicated depending on which procedure is used. This example will show a regression model modeling the 2021 gross rent of a renter-occupied housing unit using the year householder moved into the housing unit. The regression model will first be shown using PROC SURVEYREG with correct parameter estimates and variances. Next, the regression model will be shown using PROC REG with the correct parameter estimates but will generate incorrect variances. Last, we demonstrate how not using any weighting produces incorrect results. Moreover, in Section 4, Example 4.5, we calculate the regression coefficient directly.

First, PROC SURVEYREG will be used to generate the correct parameter estimates and variances. Figure 3.3.1 shows how this can be done.

Figure 3.3.1: SAS SURVEYREG Code for Estimating Variance of a Regression Coefficient

```
proc surveyreg data=HU21 varmethod=brr(fay) ;
  *subset to renters with valid rent only;
 where tenure=1 and Grent not in (-2,-1) ;
 domain tenure;
 model Grent = HHFIRSTMOVEIN/ solution ;
 weight FW ;
 repweights fw1-fw80 ;
 ods output parameterEstimates = MyParmEst ;
run;

data MyParmEstfin ;
 set MyParmEst ;
 * Keep the 4th observation, which is renters. ;
 if _n_ = 4 ;
 se = stderr;
 var = se**2 ;
 drop parameter dendf tvalue probt stderr ;
run ;

proc print data=MyParmEstFin ( keep = estimate se ) noobs ;
 format estimate se 8.4 ;
run ;
```

The SAS code of Figure 3.3.1 produces the output of Figure 3.3.2.

Figure 3.3.2: SAS SURVEYREG Output for Estimating Variance of a Regression Coefficient

Estimate	se
23.6473	1.2409

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Using the `weight` statement in SAS is not a shortcut. We now show how PROC REG and the final weights estimate the correct parameter estimates but overstate the variances because it does not correctly account for the sample design of NYCHVS. Figure 3.3.3 shows how this can be done.

Figure 3.3.3: SAS REG Code for Estimating Variance of a Regression Coefficient

```
* Weighted -- Using the sample design weights.;
* CORRECT estimate but INCORRECT variances;

proc reg data=Hu21 ;
  *subset to renters with valid rent only;
 where tenure=1 and Grent not in (-2,-1) ;
 model Grent = HHFIRSTMOVEIN;
 weight FW ;
 ods output parameterEstimates = MyParmEstw ;
run ;

data MyParmEstwFin ;
 set MyParmEstw ;
 if _n_ = 2 ;
 drop model dependent variable df tvalue probt ;
run;

proc print data=MyParmEstwFin ;
format estimate stderr 8.4 ;
run ;
```

The SAS code of 3.3.3 generates the output of Figure 3.3.4.

Figure 3.3.4: SAS REG Output for Estimating Regression Coefficients

Estimate	StdErr
23.6473	1.2606

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

As expected, the estimate of the regression coefficient in Figures 3.3.4 and 3.3.2 agree, but the standard error using PROC REG and the `weight` statement overestimates the standard error by 2 percent or $\left(1 - \frac{1.2606}{1.2409}\right)$.

Next, we show the worst possible case. We show what happens when PROC REG is used without final sample weights to generate an unweighted modeled parameter estimate with its corresponding standard error. Figure 3.3.5 shows how this can be done.

Figure 3.3.5: SAS Code for Estimating Variance of a Regression Coefficient Incorrectly Using No Weights

```
* Unweighted – Not using the sample design weights. ;
* INCORRECT estimate and INCORRECT variance;

proc reg data=HU21;
  *subset to renters with valid rent only;
 where tenure=1 and Grent not in (-2,-1) ;
 model Grent = HHFIRSTMOVEIN ;
 ods output parameterEstimates = MyParmEstu ;
run ;

data MyParmEstuFin ;
 set MyParmEstu ;
 if _n_ = 2 ;
 drop model dependent variable df tvalue probt ;
run;

proc print data = MyParmEstuFin ;
  format estimate stderr 8.4 ;
run ;
```

The SAS code of Figure 3.3.5 generates the output of Figure 3.3.6.

Figure 3.3.6: SAS Output for Estimating Variance of a Regression Coefficient Incorrectly

Estimate	StdErr
18.1019	1.0979

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Estimating the regression coefficient and its variance without the weights produces an incorrect estimate of the parameter and an underestimate of the variance.

4. Examples of Calculating Variances with Base SAS

This section includes examples of calculating the variance of a total, difference, mean, median, regression coefficient, longitudinal change, and percent change within a complex survey design by using replicate weights. Unlike the previous section where we used the SAS PROC SURVEY procedures, here we calculate all values directly using programming within base SAS (the DATA step, PROC SORT, and PROC MEANS). This section is provided for data users who would rather code the variance estimation methods themselves rather than use the PROC SURVEY procedures provided by SAS. In almost all cases, the estimates calculated using SAS PROC SURVEY and the estimates from base SAS are the same and when they differ, we explain those differences.

Example 4.1. Estimating the Variance of a Total

In Example 2.1, we generally demonstrated how to estimate the variance for the occupied housing units in NYCHVS using replicate weights. We now show how to estimate the same total directly with the replicate weights.

For our example, the domain of interest is all occupied housing units in NYC for 2021. In Figure 4.1.1, we start by using the HU21 data file. This file contains all HUs, including both occupied and vacant. Thus, we need to filter the file to occupied HUs only. Then, we calculate the sample estimate and replicate estimates of the number of occupied housing units.

Figure 4.1.1: SAS Code for Estimating Variance of a Total (Step 1 &2)

```
* Steps 1 & 2: Sum the sample and the 80
 replicate weights and writes them out to a file ;

proc means data=HU21 sum noprint ;
 * only occupied HUs;
      where occ=1 ;
 * The sample and the replicates. ;
 var FW fw1-fw80 ;
 output out=Data3 sum=est rw1-rw80 ;
run ;
```

Next, the code in Figure 4.1.2 applies Equation (2.1) to the sample estimate and replicate estimates.

Figure 4.1.2: SAS Code for Estimating Variance of a Total (Step 3)

```
* Step 3: Use the sample estimate and the 80 replicate
 estimates to compute the estimated replicate variance(s)
 using the Equation 2.1 for 80 replicates. ;

data Data4 (keep = est var se) ;
 set Data3 end=eof ;
 * Fill array with the replicate sums. ;
 array repwts{80} rw1-rw80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (repwts{j} - est)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = (var)**(0.5) ;
 output ;
run ;

proc print data=Data4 noobs ;
 var est se ;
 format est se comma12.0 ;
run ;
```

The SAS code of Figures 4.1.1 - 4.1.2 generates the output of Figure 4.1.3.

Figure 4.1.4: SAS Output for Estimating Variance of a Total

est	se
3,157,105	13,439

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

In this example, the estimate of the number of occupied HUs in NYC for 2021 is 3,157,105 with an estimated standard error of 13,439. These results from Example 4.1 match exactly with the results in Example 2.1.

Example 4.2. Estimating the Variance of a Difference

This example demonstrates how to estimate the variance of a difference. The specific statistic of interest is the difference between the number of public housing units in Manhattan and the number of public housing units in the Bronx for 2021.

First, input the data file and keep those sample units in the domain of interest. For our example, the domains of interest are 1) the public housing units in Manhattan and 2) the public housing units in the Bronx. In Figure 4.2.1, we start by keeping only those sample HUs in the

domains of interest. For 2021, there are 239 completed interviews that are public housing in Manhattan, and 208 completed interviews that are public housing in the Bronx.

Figure 4.2.1: SAS Code for Flagging Domain of Interest

```
* Subset to only public housing units in Manhattan. ;
data data5 ;
 set HU21 ;
 where boro=3 and csr=5;
run ;

* Subset to only public housing units in the Bronx. ;
data Data6 ;
 set HU21 ;
 where boro=1 and csr=5;
run;
```

Next, Figure 4.2.2 shows how to calculate the sample estimate and replicate estimates of the number of public housing units in Manhattan, as well as the Bronx. Then, we merge those files to get the differences.

Figure 4.2.2: SAS Code for Estimating Variance of a Difference

```
* Manhattan;
proc means data=data5 sum noprint ;
 * The sample estimate and the replicate estimates. ;
 var FW fw1-fw80 ;
 output out=data7 sum=est1 rw1-rw80 ;
run ;

* The Bronx. ;
proc means data=data6 sum noprint ;
 var FW fw1-fw80 ;
 output out=Data8 sum=est2 rw2_1-rw2_80 ;
run ;

* Merge the two files and get their differences. ;
Data Data9 (keep = diff0 diff1-diff80) ;
 merge data7 Data8 ;
 array diff(80) diff1-diff80 ;
 array rw2_(80) rw2_1-rw2_80 ;
 array rw(80) rw1 - rw80 ;
 diff0=est1-est2 ;
 do i=1 to 80 ;
  diff(i)=rw(i)-rw2_(i) ;
  end;
run ;
```

Last, Figure 4.2.3 shows how to apply Equation (2.1) to calculate the sample estimate and the replicate estimates of the difference.

Figure 4.2.3: SAS Code for Estimating Variance of a Difference

```
data Data10 (keep=diff0 var se) ;
 set Data9 end=eof ;
 * Fill array with the replicate means ;
 array diff{80} diff1-diff80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (diff{j} - diff0)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = var**(0.5) ;
 output ;
run ;

proc print data=Data10 noobs ;
 var diff0 se ;
 format diff0 se comma15.0 ;
run ;
```

The SAS code of Figures 4.2.1 - 4.2.3 generates the output of Figure 4.2.4.

Figure 4.2.4: SAS Output for Estimating Variance of a Difference

diff0	se
7,810	604

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The difference between the number of public housing units in Manhattan and the Bronx in 2021 is 7,810, with an estimated standard error of 604.

Example 4.3. Estimating the Variance of a Mean

In this example, we will show how to estimate the variance of a mean. The variable of interest will be the average gross rent of renter-occupied HUs. Since we do not know the true total number of renter-occupied HUs in NYC, the mean in this case is a special case of the ratio estimator. Example 4.6 will explore ratio estimation in more detail.

An estimated ratio from a survey, defined as $\hat{R} = \hat{Y}/\hat{X}$, is a nonlinear statistic of two estimated totals $\hat{Y}$ and $\hat{X}$. Replicate weights are especially suited for estimating the variance of nonlinear statistics. In this example, we estimate the variance of a mean estimator $\hat{\bar{Y}} = \sum \hat{Y}/\hat{N}$.

The first step of calculating the variance of a mean with replication is to calculate the mean for each replicate. Figure 4.3.1 shows how this can be done.

Figure 4.3.1: SAS Code for Estimating Variance of a Mean (Step1 &2)

```
* Subset the dataset to only renter-occupied HUs
 with valid gross rent. ;
Data Data13;
      set HU21;
      where tenure=1 and GRENT not in (-2,-1);
      rename Grent=rent;
run;

* This empty data set is produced for the merge later. ;
data Data14 ;
 length mean0-mean80 8. ;
run;
* Estimate the replicate estimates for 80 replicates ;
%macro repss(rep) ;
proc means data=Data13 mean noprint ;
 weight fw&rep. ;
 var rent ;
 output out=datal&rep. mean=mean&rep. ;
run;

data Data14 ;
 merge Data14 datal&rep. ;
run ;
%mend repss ;

%macro doit;
 %do i=0 %to 80 ;
  %repss(&i.) ;
  %end;
%mend doit ;

%doit ;
```

Finally, we apply Equation (2.1) to calculate the replicate variance. Figure 4.3.2 shows how this can be done with SAS.

Figure 4.3.2: SAS Code for Estimating Variance of a Mean (Step 3)

```
* Apply Step 3 to the replicate estimates and
 estimate the variance. ;

data Data15 (keep=mean0 var se) ;
 set Data14 end=eof ;
 * Fill array with the replicate means ;
 array mean{80} mean1-mean80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80;
 do j = 1 to 80 ;
  sdiffsq{j} = (mean{j} - mean0)**2 ;
  end;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = var**(0.5) ;
 output ;
run ;

proc print data=Data15 noobs ;
 var mean0 se ;
 format mean0 se 8.2 ;
run ;
```

The SAS code of Figures 4.3.1 - 4.3.2 generates the output of Figure 4.3.3.

Figure 4.3.3: SAS Output for Estimating Variance of a Mean

mean0	se
1726.74	15.83

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The average gross rent for renter-occupied housing units in NYC for 2021 is $1,727 with a standard error of $16. This matches the estimate and standard error calculated using the SAS PROC SURVEYMEANS procedure in Example 3.1.

Example 4.4. Estimating the Variance of a Median

This example demonstrates how to estimate the variance of a median. The specific statistic of interest is the median gross rent for renter-occupied housing units in NYC for 2021.

Estimating the variance of a median is generally the same as a mean in that we need to calculate the median for every replicate and then apply Equation (2.1).[3] The first and second steps are to calculate the sample estimate and the replicate estimates of the median. The third step is to apply Equation (2.1) to the sample estimate and replicate estimates. Figure 4.4.1 shows how this can be done with SAS. Since the domain of interest is the same as the previous example, we can use dataset Data13 from Example 4.3.

Figure 4.4.1: SAS Code for Estimating Variance of a Median

```
* This empty data set is produced for the merge later. ;
data Data16 ;
 length med0-med80 8. ;
run;

* Steps 1 & 2: Estimate the replicate estimates for replicates 0 to 80. ;
%macro repss(rep) ;
proc means data=Data13 median noprint ;
 weight fw&rep. ;
 var rent ;
 output out=datal&rep. median=med&rep. ;
run ;

data Data16 ;
 merge Data16 datal&rep. ;
run ;

%mend repss ;
%macro doit ;
 %do i=0 %to 80 ;
  %repss(&i) ;
  %end ;
%mend doit ;
%doit ;

* Apply Step 3 to the replicate estimates and estimate the variance. ;
data Data17 (keep=med0 var se) ;
 set Data16 end=eof ;
 * Fill array with the replicate means ;
 array med{80} med1-med80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (med{j} - med0)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = var**(0.5) ;
```

3 The main difference when estimating percentiles versus moment-based statistics is that there are several methods for estimating percentiles, whereas moment-based statistics generally have a small set of accepted estimators. This guide uses the default SAS estimator of weighted medians, described here: https://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473330.htm#a000619488

Figure 4.4.1: SAS Code for Estimating Variance of a Median

```
 output ;
run ;

proc print data=Data17 noobs ;
 var med0 se ;
 format mean0 se 8.2 ;
run ;
```

The SAS code of Figure 4.4.1 generates the output of Figure 4.4.2.

Figure 4.4.2: SAS Output for Estimating Variance of a Median

med0	se
1575	14.90

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimated median gross rent for renter-occupied housing units in NYC for 2021 is $1,575 with a standard error of $14.90.

Example 4.5. Estimating the Variance of a Regression Coefficient

Instead of using PROC SURVEYREG to correctly estimate the regression coefficient estimates and their variances, this example shows how we can do it directly. We do this by calculating parameter estimates for all replicate weights with PROC REG and the sample weights. This produces the correct replicate estimates. Then we apply Step 3 and calculate the squared differences between the estimates. Figure 4.5.1 shows how this can be done. Note that we are using the Data13 dataset from a previous example, since the domain of interest is the same.

Figure 4.5.1: SAS Code for Estimating Variance of a Regression Coefficient

```
* This empty data set is produced for the merge later. ;
data Data19 ;
 length parmest0-parmest80 8. ;
run ;

* Steps 1 & 2: Estimate the replicate estimates for replicates 0 to 80;
%macro repss(rep) ;
proc reg data=Data13 noprint outest= MyParmEst&rep. tableout;
 model rent = HHFIRSTMOVEIN ;
 weight fw&rep. ;
quit;

proc transpose data=myparmest&rep.
      out=myparmest&rep.a ;
RUN;

Data myparmest&rep.a;
```

Figure 4.5.1: SAS Code for Estimating Variance of a Regression Coefficient

```
      set myparmest&rep.a ;
      drop _NAME_ _lABEL_ COL3-COL6;
      RENAME COL1=parmest&rep. COL2=se&rep. ;
      if _n_=3 then output;
run;
data Data19 ;
 merge Data19 myparmest&rep.a ;
run ;

%mend repss ;
%macro doit ;
 %do i=0 %to 80 ;
  %repss(&i.) ;
  %end ;
%mend doit ;
%doit ;

* Apply Step 3 to the replicate estimates and estimate the variance.;
data data20 (keep=parmest0 var se) ;
 set Data19 end=eof ;
 * Fill array with the replicate means ;
 array parmest{80} parmest1-parmest80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (parmest{j} - parmest0)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = var**(0.5) ;
 output ;
run;

proc print data=data20 noobs ;
 var parmest0 se;
 format parmest0 se comma8.2 ;
run ;
```

The SAS code of Figure 4.5.1 generates the output of Figure 4.5.2.

Figure 4.5.2: SAS Output for Estimating Variance of a Regression Coefficients

parmest0	se
23.65	1.24

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The standard error for the regression estimator in Figure 4.5.2 is the same as the estimate from Example 3.3.

Example 4.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 4.7 will consider an estimate of percent of change that is calculated at the HU level.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we are interested in the estimated difference in gross vacancy rates across two survey cycles. We write this as $\hat{p}_t = \hat{X}_t / \hat{N}_t * 100$, where $\hat{X}_t$ is the estimator of the total number of vacant HUs at time *t*, and $\hat{N}_t$ is the estimator of the total number of HUs at time *t.*

We'll measure the difference in the gross vacancy rate between 2021 and 2017 in this example, so the statistic of interest is:

$$\hat{\Delta}_{t=2021} = \frac{\hat{X}_{t=2021}}{\hat{N}_{t=2021}} * 100 - \frac{\hat{X}_{t=2017}}{\hat{N}_{t=2017}} * 100$$

To estimate the variance of $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces of $\hat{\Delta}_{t=2021}$ for each of the 80 replicates, making 80 replicate estimates of $\hat{\Delta}_{t=2021}$, repeat those steps for 2017, and then apply Equation (2.1). Figure 4.6.1 shows how to do this with SAS.

Figure 4.6.1: SAS Code for Estimating Variance of a Longitudinal Change

```
*Define Vacant and Occupied HUs. ;
data temp1 ;
 set HU21 ;
 if Occ=1 then type='1'; * Occupied HUs;
 else if Occ in (2,3,4)  then type='2'; * X-hat 2021* vacant HUs ;
 * Type 1 & 2 makes total number HUs, N-hat ;
run;

data temp2 ;
 set HU17 ;
 if recid="1" then type='1' ; * Occupied HUs ;
 else if recid="3" then type='2' ; * X-hat 2017* vacant HUs ;
 * Type 1 & 2 makes total number HUs, N-hat;
run;

* This empty data set is produced for the merge later. ;
data data21 ;
 length diff0-diff80 8. ;
run ;

* Estimate the replicate estimates for replicates 0-80. ;
%macro repssss(rep) ;
* Get denominator N-hat 2021. ;
proc means data= temp1 noprint ;
```

Figure 4.6.1: SAS Code for Estimating Variance of a Longitudinal Change

```
 where type in ('2','1') ;
 var fw&rep.;
 output out=den21rep&rep. sum=den21rep&rep. ;
run ;

* Get numerator X-hat 2021. ;
proc means data= temp1 noprint ;
 where type in ('2') ;
 var fw&rep.;
output out=num21rep&rep. sum=num21rep&rep. ;
run ;
* Get denominator N-hat 2017. ;
proc means data= temp2 noprint ;
 where type in ('2','1') ;
  var fw&rep.;
 output out=den17rep&rep. sum=den17rep&rep. ;
run ;

* Get numerator X-hat 2017. ;
proc means data= temp2 noprint ;
 where type in ('2') ;
 var fw&rep.;
 output out=num17rep&rep. sum=num17rep&rep.;
run ;

* Merge the replicate estimates of each year, and get difference
 in vacancy rate by replicate. ;
data datamrep&rep.;
 merge num17rep&rep. num21rep&rep. den17rep&rep. den21rep&rep. ;
 diff&rep. = (num21rep&rep./den21rep&rep.)*100 -
(num17rep&rep./den17rep&rep.)*100 ;
 keep diff&rep.;
run ;

data data21 ;
 merge data21 datamrep&rep. ;
run ;
%mend repssss ;

%macro doit ;
 %do i=0 %to 80 ;
  %repssss(&i.) ;
  %end;
%mend doit ;
%doit ;

* Apply Step 3 to the replicate estimates and estimate the variance. ;
data data22 (keep=diff0 var se) ;
 set data21 end=eof ;
 * Fill array with the replicate means. ;
 array diff{80} diff1-diff80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (diff{j} - diff0)**2 ;
```

Figure 4.6.1: SAS Code for Estimating Variance of a Longitudinal Change

```
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = (var)**(0.5) ;
 output ;
run ;

proc print data=data22 noobs ;
 var diff0 se ;
 format diff0 se 8.4 ;
run ;
```

The SAS code of Figure 4.6.1 produces the output in Figure 4.6.2.

Figure 4.6.2: SAS Output for Estimating Variance of a Longitudinal Change

diff0	se
3.0068	0.4038

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

Thus, the total gross vacancy rate in NYC for 2021 has increased 3.01 percent from 2017 with a standard error of 0.40 percent.

Example 4.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

This example considers median gross rent and how it changes over time. Both the rate of change and resulting variance are calculated.

The statistic of interest is the percent change in the median gross rent from 2017 to 2021. Let $\widehat{M}_t$ be the estimator of the median gross rent at time *t.* The statistic of interest is:

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we use the 2017 replicate weights and calculate 80 replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ and similarly use the 2021 replicate weights and calculate 80 replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2021}}$. Next, we merge the replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ and $\widehat{\mathrm{M}}_{\mathrm{t=2021}}$ by replicate and calculate 80 replicate estimates of $\%\hat{\Delta}_t$. The final step is to apply Equation (2.1) to the replicate estimates of $\%\hat{\Delta}_t$.

Figure 4.7.1 shows how this can be done with SAS.

Figure 4.7.1: SAS Code for Estimating Variance of a Rate of Change

```
* Subset to renter-occupied units with valid rent. ;
Data D2021;
 set HU21;
 where occ=1 and tenure=1 and Grent not in (-2,-1);
run;

Data D2017 ;
 set HU17 ;
 where sc116 in ('2','3') and uf26 not in ('99999') ;
 rent17=input(uf26,8.) ;
run;

* This empty data set is produced for the merge later. ;
data data23 ;
 length diff0-diff80 8. ;
run ;

* Steps 1 & 2: Estimate the replicate estimates for replicates 0-80. ;
%macro repssss(rep) ;
* Estimate theta for each year. ;
proc means data= d2021 median noprint ;
 weight fw&rep. ;
 var Grent ;
 output out=datam21rep&rep. median=median21rep&rep. ;
run ;

proc means data= d2017 median noprint ;
 weight fw&rep. ;
 var rent17 ;
 output out=datam17rep&rep. median=median17rep&rep. ;
run ;

* Merge the replicate estimates of each year by replicate. ;
data datamrep&rep. ;
 merge datam17rep&rep. datam21rep&rep. ;
 diff&rep. = (median21rep&rep.-median17rep&rep. )/median17rep&rep.;
 keep diff&rep. ;
run ;

data data23 ;
 merge data23 datamrep&rep.;
run ;

%mend repssss ;

%macro doit ;
 %do i=0 %to 80 ;
  %repssss(&i.) ;
  %end ;
%mend doit ;
%doit ;

* Apply Step 3 to the replicate estimates and estimate the variance. ;
data data24 (keep=diff0 var se) ;
```

Figure 4.7.1: SAS Code for Estimating Variance of a Rate of Change

```
 set data23 end=eof ;
 * Fill array with the replicate means. ;
 array diff{80} diff1-diff80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (diff{j} - diff0)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = (var)**(0.5) ;
 output ;
run ;

proc print data=data24 noobs ;
 var diff0 se ;
 format diff0 se 8.4 ;
run ;
```

The SAS code of Figure 4.7.1 produces the output in Figure 4.7.2.

Figure 4.7.2: SAS Output for Estimating Variance of a Rate of Change

diff0	se
0.0862	0.0126

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

So the median gross rent in NYC for 2021 has increased 8.6 percent from 2017, with a standard error of 1.2 percent.

5. Examples of Calculating Variances with STATA

Data users can also use STATA to calculate variances for NYCHVS. This section reviews the same examples as Section 4, this time using STATA. STATA produces the exact same point estimates and the same standard errors as the SAS SURVEY procedures and Base SAS methods.

Example 5.1. Estimating the Variance of a Total

When reading NYCHVS PUFs into STATA, first download and save these CSV files (mentioned in page 7) provided on the Census Bureau's NYCHVS website at: https://www.census.gov/programs-surveys/nychvs/data/datasets.html. Then, use the command lines given in Figure 5.1.1 in STATA to import these files and update the input-related items in the code (highlighted in yellow in Figure 5.1.1).

For housing unit estimates, users need to download All Units Records (CSV file #1), Occupied Records (CSV file #2) and Vacant Records (CSV file #4), and then combine the three files into one.

Figure 5.1.1 shows the code for reading in Occupied HUs and Vacant HUs data file (CSV file #2 and #4), appending them together, and then merging with All Units data file (CSV file #1).

Figure 5.1.1: Importing the Public Use Files In STATA

```
cd " LOCATION OF CSV FILES"

import delimited "vacant_puf_21.csv", clear
save "vacant_puf_21.dta", replace
import delimited "occupied_puf_21.csv", clear
append using "vacant_puf_21.dta"
drop fw*
save "occvac_puf_21.dta", replace
import delimited "allunits_puf_21.csv", clear
merge 1:1 control using "occvac_puf_21.dta"

save "hu21.dta", replace
```

For generating STATA datasets for Person Records (CSV file #3), use the import code used for the HUs records and update it for Person file.

For generating STATA datasets for prior years NYCHVS PUFs, go to the same website mentioned above, click the "2017" or different year tab. In 2017 and prior, the PUFs were given as text files, and a STATA import program was provided along with the data files. Users need to download both the data files, the STATA import program, and update the input-related items in the code (highlighted in yellow in Figure 5.1.2). Run the part of the code for Occupied Records,

and/or Person Records, and/or Vacant Records, to generate local STATA datasets. Figure 5.1.2 shows the part of the code for reading in Occupied HUs data file for 2017. Figure 5.1.3 shows the STATA code for appending the Occupied Records after saving the Vacant Records. The 2017 HU dataset is saved as HU17 and will be used in later examples.

Figure 5.1.2: STATA Code (Partial) for Reading in the Public Use Files For 2017 & Prior

```
*2017 NYCHVS STATA IMPORT PRORGRAM

cd "LOCATION OF TEXT FILES"

*OCCUPIED DATA FILE
clear
infix            ///
recid   1        ///
```

***SEE THE REST OF THE STATA CODE IN STATA IMPORT PROGRAM FOUND IN THE CENSUS BUREAU'S NYCHVS WEBSITE MENTIONED IN PREVIOUS PAGE**

Figure 5.1.3: STATA Code for Append Two Files Together for Housing Unit Estimates

```
* Append Occupied Records with Vacant Records

append using "nychvs_17_occ.dta"
```

Note in 2017 and prior, the final weights and replicate weights are stored as character variables with five implied decimal places. Users need to convert them before doing any analysis. Figure 5.1.4 shows how this be done in STATA.

Figure 5.1.4: STATA Code for Converting Weights with Implied Decimal Places

```
* Dividing weights by 100,000

replace fw=fw/100000

foreach x of var fw1-fw80 {
      replace `x' = `x' / 100000
}

save "HU17.dta", replace
```

After local STATA datasets are created, run the code in Figure 5.1.5 before running any example analysis at the household level.

Figure 5.1.5: Reading in the Data File into STATA and Setting Survey Design Parameters

```
cd "DIRECTORY PATH OF NYCHVS DATA"

* First, household level data:
use "HU21.dta", clear

* Setting survey design parameters:
svyset[pweight=fw], vce(sdr) sdrweight(fw1-fw80) fay(.5)mse
```

In this example, we estimate the total occupied housing units in NYC and use the replicate weights to calculate the variance. This is done by first identifying the domain of interest and then running the total and variance calculation by using the STATA code in Figure 5.1.6. The resulting output is provided in Figure 5.1.7.

Note that both `vce(brr)` and `vce(sdr)` options generate the same estimate and standard errors. The only difference between those two options is the first one uses a t-distribution and the second uses a z-distribution; thus, their confidence intervals are slightly different. NYCHVS uses SDR, so users should use `vce(sdr)` option. For a more detailed explanation of z-distribution and t-distribution, please refer to Section 8 on page 63.

Figure 5.1.6: STATA Code for Estimating Variance of a Total

```
* Generate occupied housing unit dummy var:
gen occ_final = (occ == 1)

* Now estimating total:
svy: total occ_final
```

The STATA code of Figure 5.1.6 generates the output of Figure 5.1.7.

Figure 5.17: STATA Output for Estimating Variance of a Total

```
Survey: Total estimation          Number of obs   =     8,394
                                  Population size =     3,644,065
                                    Replications  =        80

--------------------------------------------------------------
             |             SDR *
             |    Total   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
  occ_final  |   3157105   13438.76      3130766    3183445
--------------------------------------------------------------
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 5.1.7, the estimate of total occupied housing units in NYC is 3,157,105 with an estimated standard error of 13,439. The estimates and standard error from STATA are the same as the results produced by SAS in Example 4.1. Unlike SAS, the `[svy] sdr` command in STATA

displays the 95 percent confidence intervals by default – for more information on the confidence intervals, see Section 8.

Example 5.2. Estimating the Variance of a Difference

In this example, we are interested in calculating the difference between the number of public housing units in Manhattan and the number of public housing units in the Bronx for 2021.

First, we define the domains of interest individually and then define the difference. After we do this, we calculate the variance using the replicate weights in the STATA code defined in Figure 5.2.1.

Figure 5.2.1: STATA Code for Estimating Variance of a Difference

```
* Bronx:
gen ph_bronx = ((csr == 5) & (boro == 1))

* Manhattan:
gen ph_man = ((csr == 5) & (boro == 3))

* Creating difference between these:
gen stable_diff = ph_man - ph_bronx

* Create proper subdomain:
gen ph_boro = (((csr == 5) & boro == 3) | ((csr == 5) & (boro == 1)))

* Now estimating differences between these:
svy, subpop(ph_boro): total stable_diff
```

The STATA code of Figure 5.2.1 generates the output of Figure 5.2.2.

Figure 5.2.2: STATA Output for Estimating Variance of a Difference

```
Survey: Total estimation          Number of obs   =     8,394
                                  Population size =     3,644,065
                                  Subpop. no. obs =       447
                                    Subpop. size  =       95,730.963
                                    Replications  =       80

--------------------------------------------------------------
             |             SDR *
             |   Total   std. err.     [95% conf. interval]
-------------+------------------------------------------------
 stable_diff |  7809.567  603.7287      6626.28   8992.853
--------------------------------------------------------------
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 5.2.2, we can see the estimated difference of public housing units in Manhattan and the Bronx is 7,810 with the standard error of 604. These results are the same as produced with SAS – see Example 4.2.

Example 5.3. Estimating the Variance of a Mean

For this example, we are estimating the average gross rent for renter-occupied housing units in NYC for 2021 and its variance. This can be easily done in STATA by first identifying the domains of interest and using the STATA code to calculate the variance using the replicate weights. The STATA code is provided in Figure 5.3.1.

Figure 5.3.1: STATA Code for Estimating Variance of a Mean

```
gen gross_rent = GRENT
replace gross_rent = . if ((GRENT == -2) | (GRENT == -1))

* Now estimating mean:
Svy: mean gross_rent
```

The STATA code of Figure 5.3.1 generates the output of Figure 5.3.2.

Figure 5.3.2: STATA Output for Estimating Variance of a Mean

```
Survey: Mean estimation          Number of obs   =       4,424
                                 Population size =       1,853,923
                                   Replications  =       80


                        SDR *
               Mean     std. err.    [95% conf.  interval]

gross_rent  1726.739    15.82925    1695.714        1757.764
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 5.3.2, we can see that the average NYC gross rent calculated using STATA is $1,727 and the corresponding standard error is $16, both of which are the same as produced with SAS – see Examples 3.1 and 4.3.

Example 5.4. Estimating the Variance of a Median

In this example, we are interested to calculate the median gross rent and its standard error for renter-occupied housing units in NYC for 2021. Figure 5.4.1 shows the STATA code for calculating variance of a median.

Figure 5.4.1: STATA Code for Estimating Variance of a Median

```
quietly summarize gross_rent [w=fw], detail
scalar rent21 = r(p50)

* Matrix to store 2021 estimates:
matrix rent_ests_21 = (rent21)

* Now looping through all repweights for 2021:
foreach x of var fw1-fw80 {

      quietly summarize gross_rent [w=`x'], detail
      scalar rent21_ith = r(p50)

      * Adding to matrix that stores 2021 estimates:
      matrix rent_ests_21 = (rent_ests_21 \ rent21_ith)
}

* Now getting squared differences for variance estimation:
scalar summed_variance = 0
quietly forvalues i = 2/81 {

      scalar sq_pt_est_med = (rent_ests_21[`i',1] - rent_ests_21[1,1])^2
      scalar summed_variance = summed_variance + sq_pt_est_med
}

scalar summed_variance = (4/80) * summed_variance
scalar se_est = (summed_variance)^(0.5)
scalar point_est = rent_ests_21[1,1]

* Final Point Estimate:
display point_est

* Final Estimated Standard Error:
display se_est
```

The STATA code of Figure 5.4.1 generates the output of Figure 5.4.2.

Figure 5.4.2: STATA Output for Estimating Variance of a Median

```
* Final Point Estimate:
. display point_est
1575

. * Final Estimated Standard Error:
. display se_est
14.899664
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimate for median gross rent for renter-occupied housing units in NYC is $1,575, with standard error of $15. The results are the same as produced by SAS – see Example 4.4.

Example 5.5. Estimating the Variance of a Regression Coefficient

For this example, we are interested in modeling the 2021 gross rent of renter-occupied housing units using the year the householder moved into the housing unit.

In Figure 5.5.1, we used the `subpop` instead of the `if` option. Either would produce the same variance estimates – see also "Cautions about Domain Analysis Don't Apply" in Section 2.

Figure 5.5.1: STATA Code for Estimating Variance of a Regression Coefficient

```
gen renters = (rent_amount != -2) & (grent > 0)

* Linear regression model:
svy, subpop(renters): reg GRENT HHFIRSTMOVEIN
```

The STATA code of Figure 5.5.1 generates the output of Figure 5.5.2.

Figure 5.5.2: STATA Output for Estimating Variance of a Regression Coefficient

```
      Survey: Linear regression                 Number of obs    =     7,089
                                                Population size  = 3,157,105
                                                Subpop. no. obs  =     4,424
                                                Subpop. size     = 1,853,923
                                                Replications     =        80
                                                Wald chi2(1)     =    352.38
                                                Prob > chi2      =    0.0000
                                                 R-squared       =    0.0734

             |             SDR *
       GRENT |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+------------------------------------------------------------------
HHFIRSTMOV~N |    23.64732   1.240863    19.06   0.000     21.21527    26.07937
       _cons |   -45766.55   2486.632   -18.41   0.000    -50640.26   -40892.84
------------------------------------------------------------------------------
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 5.5.2, we can see that the calculated regression coefficient is 23.65 and the standard error is 1.24. The estimates are the same produced by SAS – see Examples 3.3 and 4.5.

Example 5.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 5.7 will consider an estimate of percent of change that is calculated at the HU level.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we are interested in the estimated gross vacancy rate or $\hat{p}_t = \hat{X}_t / \hat{N}_t * 100$, where

$\hat{X}_t$ is the estimator of the total number of vacant HUs at time t, and $\hat{N}_t$ is the estimator of the total number of HUs at time t.

Further, we are really interested in the difference in the gross vacancy rate between 2021 and 2017, so the statistic of interest is:

$$\hat{\Delta}_{t=2021} = \frac{\hat{X}_{t=2021}}{\hat{N}_{t=2021}} * 100 - \frac{\hat{X}_{t=2017}}{\hat{N}_{t=2017}} * 100$$

To estimate the variance of $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces of $\hat{\Delta}_{t=2021}$ for each of the 80 replicates, making 80 replicate estimates of $\hat{\Delta}_{t=2021}$, repeat those steps for 2017, and then apply Equation (2.1). Figure 5.6.1 shows how we do this with STATA.

Figure 5.6.1: STATA Code for Estimating Variance of a Longitudinal Change

```
* Read in 2021 data:
use "HU21.dta", clear

* Generate vacancy variable:
gen vac = (occ != 1 )

* Calculating 2021 totals:
quietly summarize fw if vac == 1
scalar vac_est21 = r(sum)

quietly summarize fw
scalar tot_est21 = r(sum)

scalar est21 = (vac_est21 / tot_est21) * 100

* Creating matrix that stores 2021 estimates:
matrix all_ests_21 = (est21)

* Now going through replicates:
foreach x of var fw1-fw80 {

      quietly summarize `x' if vac == 1
      scalar vac_est21_i = r(sum)

      quietly summarize `x'
      scalar tot_est21_i = r(sum)

      * i_th replicate estimate for 2021 portion:
      scalar est21_i = (vac_est21_i / tot_est21_i) * 100

      * Adding to matrix that stores 2021 estimates:
      matrix all_ests_21 = (all_ests_21 \ est21_i)
}

* Now reading in 2017 data:
import delimited "hu17.csv", clear
```

Figure 5.6.1: STATA Code for Estimating Variance of a Longitudinal Change

```
* Generate vacancy variable:
gen vac = (recid == 3)

* Calculating 2017 totals:
quietly summarize fw if vac == 1
scalar vac_est17 = r(sum)

quietly summarize fw
scalar tot_est17 = r(sum)

scalar est17 = (vac_est17 / tot_est17) * 100

* Creating matrix that stores 2017 estimates:
matrix all_ests_17 = (est17)

* Now going through replicates:
foreach x of var fw1-fw80 {

      quietly summarize `x' if vac == 1
      scalar vac_est17_i = r(sum)

      quietly summarize `x'
      scalar tot_est17_i = r(sum)

      * i_th replicate estimate for 2017 portion:
      scalar est17_i = (vac_est17_i / tot_est17_i) * 100

      * Adding to matrix that stores 2017 estimates:
      matrix all_ests_17 = (all_ests_17 \ est17_i)
}

* Matrix will point estimates for the sample estimate and all replicates:
matrix diffs = all_ests_21 - all_ests_17

* Now getting squared differences for variance estimation:
scalar summed_variance = 0
quietly forvalues i = 2/81 {

      scalar sq_pt_est_diffs = (diffs[`i',1] - diffs[1,1])^2
      scalar summed_variance = summed_variance + sq_pt_est_diffs
}

scalar summed_variance = (4/80) * summed_variance
scalar se_est = (summed_variance)^(0.5)
scalar point_est = diffs[1,1]

* Final Point Estimate:
display point_est

* Final Estimated Standard Error:
display se_est
```

The STATA code of Figure 5.6.1 generates the output of Figure 5.6.2.

Figure 5.6.2: STATA Output for Estimating Variance of a Longitudinal Change

```
. * Final Point Estimate:
. display diffs[1,1]
3.0067783

. * Final Estimated Standard Error:
. display se_est
.40384096
```

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

So, the change in the gross vacancy rate from 2017 to 2021 is a 3.0 percent increase, with a standard error of 0.4 percent.

Example 5.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

There are many different statistics that measure the change between two cycles of NYCHVS. Differences, percent change, and ratios can be used to measure how much a given statistic changed from one cycle of NYCHVS to another. This example considers the housing characteristic of the median gross rent and how it can change over time. Both the rate of change and resulting variance are demonstrated.

The statistic of interest is the percent change in the median gross rent from 2017 to 2021. Let $\widehat{M}_t$ be the estimator of the median gross rent at time *t.* The statistic of interest is

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we use the 2021 replicate weights and store 80 replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2021}}$ within a matrix, and similarly use the 2017 replicate weights to store 80 replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ in another matrix. Next, we take the difference of these two matrices $\widehat{\mathrm{M}}_{\mathrm{t=2021}}$ and $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ by replicate and then divide each matrix entry by $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ to get our 80 replicate estimates of $\%\hat{\Delta}_t$. The final step is to apply Equation (2.1) to the replicate estimates of $\%\hat{\Delta}_t$.

Figure 5.7.1 shows how this can be done with STATA.

Figure 5.7.1: STATA Code for Estimating Variance of a Rate of Change

```
* Read in 2021 data:
use "HU21.dta", clear

* Generating needed variables - renters and gross rent:
gen renters = (rent_amount != -2) & (grent > 0)
gen gross_rent = grent
replace gross_rent = . if grent <= 0

* Calculating 2021 median gross rent:
```

Figure 5.7.1: STATA Code for Estimating Variance of a Rate of Change

```
quietly summarize gross_rent [w=fw] if renters == 1, detail
scalar rent21 = r(p50)

* Matrix to store 2021 estimates:
matrix rent_ests_21 = (rent21)

* Now looping through all repweights for 2021:
foreach x of var fw1-fw80 {

      quietly summarize gross_rent [w=`x'] if renters == 1, detail
      scalar rent21_ith = r(p50)

      * Adding to matrix that stores 2021 estimates:
      matrix rent_ests_21 = (rent_ests_21 \ rent21_ith)
}

* Now to 2017 data:
import delimited "hu17.csv", clear

* Generating needed variables - renters and gross rent:
gen renters = (sc116 == 2 | sc116 == 3) & uf26 != 99999
gen gross_rent = uf26

* Calculating 2017 median gross rent:
quietly summarize gross_rent [w=fw] if renters == 1, detail
scalar rent17 = r(p50)

* Creating matrix that stores 2017 estimates:
matrix rent_ests_17 = (rent17)

* Now looping through all repweights for 2017:
foreach x of var fw1-fw80 {

      quietly summarize gross_rent [w=`x'] if renters == 1, detail
      scalar rent17_ith = r(p50)

      * Adding to matrix that stores 2017 estimates:
      matrix rent_ests_17 = (rent_ests_17 \ rent17_ith)
}

* Final Matrix - for now storing the numerator of the estimate for
sample and all replicates:
matrix med_rent_increase_diff = (rent_ests_21 - rent_ests_17)
matrix med_rent_increase_est = J(81,1,0)

forvalues i = 1/81 {

      * Converting entries in final matrix to hold final estiamtes for
full sample and all replicates:
      matrix med_rent_increase_est[`i',1] =
med_rent_increase_diff[`i',1] / rent_ests_17[`i',1]
}

* Now getting squared differences for variance estimation:
scalar summed_variance = 0
```

Figure 5.7.1: STATA Code for Estimating Variance of a Rate of Change

```
quietly forvalues i = 2/81 {

      scalar sq_pt_med_increase_est = (med_rent_increase_est[`i',1] -
med_rent_increase_est[1,1])^2
      scalar summed_variance = summed_variance +
sq_pt_med_increase_est
}

scalar summed_variance = (4/80) * summed_variance
scalar se_est = (summed_variance)^(0.5)
scalar point_est = med_rent_increase_est[1,1]

* Final Point Estimate:
display point_est

* Final Estimated Standard Error:
display se_est
```

The STATA code of Figure 5.7.1 generates the output of Figure 5.7.2.

Figure 5.7.2: STATA Output for Estimating Variance of a Rate of Change

```
. * Final Point Estimate:
. display point_est
.0862069

.
. * Final Estimated Standard Error:
. display se_est
.01259001
```

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

Therefore, the median gross rent for NYC in 2021 has increased 8.62 percent from 2017, with a standard error of 1.26 percent.

6. Examples of Calculating Variances with R

R is another statistical package that can calculate variances using replicate weights for the NYCHVS. This section is structured similarly to Sections 3, 4, and 5 and uses the same examples for calculating variances of a total, difference, mean, median, regression coefficient, longitudinal change, and percent change.

Unlike the 2017 version of this Guide, the examples below manually calculate variances for all examples. This results in standard error estimates that are identical to those given in SAS and STATA.

Example 6.1. Estimating the Variance of a Total

Once data users are in R, some initial code needs to run to set things up for the rest of Section 6. This includes ensuring the proper R Packages are installed. Our examples use the "`rio`" , "`tidyverse`", and "`stats`" packages. The initial steps in Figure 6.1.1 show how to install the survey packages in R.

Data users will need to download the PUFs. For 2021 NYCHVS, all the PUFs are given as CSV files, which made it easier for users to import into R. Data users can access the PUFs through the Census Bureau's NYCHVS website at: https://www.census.gov/programs-surveys/nychvs/data/datasets.html . Use the import function in R to read in all three HU files, and then combine them into one file, HU21, which will be the basis for the examples in this section. Figure 6.1.1 shows how this is done in R.

Figure 6.1.1: R Code to Importing Public Use Files

```
# Install survey package with install.packages("packagename") if not
already installed.
library(rio)
library(tidyverse)
library(stats)

# Reading in HU dataset:
Hu21 <- import("Location of the CSV file/ Allunits_PUF_21.csv",
header=TRUE)

# Reading in vacant HU dataset:
hu21vac <- import("Location of the CSV file/ vacant_puf_21.csv",
header=TRUE)

# Reading in occupied HU dataset:
hu21occ <- import("Location of the CSV file/ occupied_PUF_21.csv",
header=TRUE)

 bind_rows(hu21vac) %>% dplyr::select(-c(FW:FW80))
hu21 <- hu21 %>% inner_join(hu21occ, by = "CONTROL")
```

In 2017 and prior, the NYCHVS PUFs were ACSII text files, and there were no column/variable names on these files; therefore, data users still need to use either the SAS or STATA Import Program provided in the Census Bureau's NYCHVS website at https://www.census.gov/data/datasets/2017/demo/nychvs/microdata.html to read in the 2017 NYCHVS PUFs into SAS (please see Pages 7-10 for instructions on how to read PUFs into SAS for 2017 and prior). Once in SAS, data users need to run the code in Figures 2.1.3 and 2.1.4, which generates the dataset HU17. Then, users can export the datasets from SAS to CSV format to read the file into R. In SAS, run the code in Figure 6.1.2 to convert the SAS dataset HU17 to a CSV dataset. Then, use the code shown in Figure 6.1.1 to read in the 2017 dataset into R (see Figure 6.1.3).

Figure 6.1.2: SAS Code to Export Files into R Readable Format

```
* Exporting NYCHVS datasets to R;
proc export data = HU17 dbms = csv outfile="PATH \ TO \ NEW R
DATASET\new_dataset_name2.csv" replace;
run;
```

Figure 6.1.3: R Code to Import 2017 Data File into R

```
# Reading in 2017 HU dataset:
hu17 <- import("Location of the CSV file / new_dataset_name2.csv",
header=TRUE)
```

For our example of estimating the variance of a total, we will estimate the total occupied housing units in NYC and use the replicate weights to calculate the variance. Use the R code in Figure 6.1.4 in order to do this calculation.

Figure 6.1.4: R Code for Estimating Variance of a Total

```
reps <- rep(0,80)
point_est <- sum(hu21$FW[hu21$OCC == 1])
for (i in 1:80){
 reps[i] <- sum(hu21[paste0("FW", as.character(i))][hu21$OCC == 1,])
}
reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
 ", ", as.character(round(se_est , 4))))
```

The R code provided in Figure 6.1.4 produces the R output shown in Figure 6.1.5.

Figure 6.1.5: R Output for Estimating Variance of a Total

```
               total       SE
occ_final 3157105    13439
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From the output in Figure 6.1.5, we see that the estimated total occupied housing units in NYC is 3,157,105, with a standard error of 13,439.

Example 6.2. Estimating the Variance of a Difference

In this example, we are interested in calculating the difference between the number of public housing units in Manhattan and the number of public housing units in the Bronx for 2021.

The R code provided in Figure 6.2.1 can be used to calculate the estimated difference and its variance calculation.

Figure 6.2.1: R Code for Estimating Variance of a Difference

```
## Define domains, Difference Example:
# Renter-occupied Public housings in the Bronx:
hu21$ph_bronx = ifelse((hu21$CSR == 5 & hu21$BORO == 1), 1, 0)

# Renter-Occupied Publicn housings in Manhattan:
hu21$ph_man = ifelse((hu21$CSR == 5 & hu21$BORO == 3), 1, 0)

# Difference between Public housing units in Manhattan and the Bronx:
hu21$stable_diff = hu21$ph_man - hu21$ph_bronx

reps <- rep(0,80)
point_est <- sum(hu21$FW*hu21$stable_diff)
for (i in 1:80){
 reps[i] <- sum(hu21[paste0("FW", as.character(i))]*hu21$stable_diff)
}
reps <- (reps - point_est)^2
se = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(se, 4))))
```

The output from the code in Figure 6.2.1 is provided in Figure 6.2.1.

Figure 6.2.2: R Output for Estimating Variance of a Difference

```
               total   SE
stable_diff    7810   604
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From the R output, we can see that the estimated difference between the number of public housing units in Manhattan versus the number of public housing units in the Bronx is 7,810 and the standard error is 604.

Example 6.3. Estimating the Variance of a Mean

This example calculates the variance of a mean for gross rent. We are interested in the average gross rent for renter-occupied housing units in NYC for 2021.

The R code to make the point estimate and variance estimate for this is provided in Figure 6.3.1.

Figure 6.3.1: R Code for Estimating Variance of a Mean

```
# Define domain: Renters only with non-missing rent values
hu21$renters = ifelse((hu21$RENT_AMOUNT != -2 & hu21$GRENT > 0), 1, 0)
hu21$gross_rent <- hu21$GRENT
hu21$gross_rent[hu21$GRENT <= 0] <- NA

rent_data <- hu21 %>% filter(renters == 1)
reps <- rep(0,80)
point_est <- weighted.mean(rent_data$gross_rent,
       rent_data$FW, na.rm = TRUE)
for (i in 1:80){
 reps[i] <- weighted.mean(rent_data$gross_rent,
       rent_data[,paste0("FW", as.character(i))], na.rm = TRUE)
}
reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(se_est , 4))))
```

The R output from using the code in 6.3.1 is provided in Figure 6.3.2.

Figure 6.3.2: R Output for Estimating Variance of a Mean

```
                mean      SE
gross_rent   1726.739  15.8292
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimated average gross rent for renter-occupied housing units in NYC for 2021 is $1,727 with a standard error of $16.

Example 6.4. Estimating the Variance of a Median

We will use R to compute the median gross rent for renter-occupied housing units in NYC for 2021. We use the same domains as Example 6.3, so we don't need to re-define domains.

Figure 6.4.1: R Code for Estimating Variance of a Median

```
reps <- rep(0,80)
point_est <- wgt_per_sas(rent_data,
       "gross_rent", "FW", 0.5)
for (i in 1:80){
 reps[i] <- wgt_per_sas(rent_data,
       "gross_rent",paste0("FW", as.character(i)), 0.5)
}
reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(se_est , 4))))
```

The R output from using the code in Figure 6.4.1 is provided in Figure 6.4.2.

Figure 6.4.2: R Output for Estimating Variance of a Median

```
             median     SE
gross_rent    1575   14.8997
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimated median gross rent for NYC in 2021 is $1,575, with a standard error of $15.

Example 6.5. Estimating the Variance of Regression Coefficient

For this example, we are interested in modeling the 2021 gross rent of renter-occupied housing units using the year the householder moved into the housing unit.

In R, we use the `lm` command to create a linear regression model using the replicate weights. The R code is provided in Figure 6.5.1.

Figure 6.5.1: R Code for Estimating Variance of a Regression Coefficient

```
# Define domain:
hu21$defect = ifelse(hu21$MDEFCOUNT >= 3, 1, 0)
hu21$time_stable <- ifelse(hu21$YEARBUILT < 5, 1, 0)


reps <- rep(0,80)
lin_ex <- lm(gross_rent~HHFIRSTMOVEIN, data = rent_data, weights = FW)
point_est <- lin_ex$coefficients[[2]]
for (i in 1:80){
```

```
 temp_data <- rent_data %>% rename("wgt" = paste0("FW", as.character(i)))
 lin_ex <- lm(gross_rent~HHFIRSTMOVEIN, data = temp_data, weights = wgt)
 reps[i] <- lin_ex$coefficients[[2]]
}
reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(se_est , 4))))
```

Figure 6.5.1 generates the output in Figure 6.5.2.

Figure 6.5.2: R Output for Estimating Variance of a Regression Coefficient

```
Reg Coef      SE
23.6473   1.2409
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From the R output provided in Figure 6.5.2, we can see that the calculated regression coefficient is 23.65, with a standard error of 1.24.

Example 6.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 6.7 will consider an estimate of percent of change that is calculated at the HU level. We present these two longitudinal statistics as examples thinking these calculations are the most likely of calculations that data users would be interested in.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we'll estimate the gross vacancy rate or $\hat{p}_t = \hat{X}_t / \hat{N}_t * 100$, where $\hat{X}_t$ is the estimator of the total number of vacant HUs at time *t*, and $\hat{N}_t$ is the estimator of the total number of HUs at time *t.*

Further, we're interested in the difference in the gross vacancy rate between 2021 and 2017, so the statistic of interest is:

$$\hat{\Delta}_{t=2021} = \frac{\hat{X}_{t=2021}}{\hat{N}_{t=2021}} * 100 - \frac{\hat{X}_{t=2017}}{\hat{N}_{t=2017}} * 100$$

To estimate the variance of $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces of $\hat{\Delta}_{t=2021}$ for each of the 80 replicates, make 80 replicate estimates of $\hat{\Delta}_{t=2021}$, repeat those steps for 2017, and then apply Equation (2.1). Figure 6.6.1 shows how we do this with R.

Figure 6.6.1: R Command for Estimating Variance of a Longitudinal Change

```
## First Longitudinal Example:
# Adding vacancy dummy:
hu21$vac <- ifelse(hu21$OCC != 1, 1, 0)
hu17$vac <- ifelse(hu17$recid == 3, 1, 0)


# Calculating 2021 point estimates:
hu21_vac <- hu21 %>% filter(vac == 1)
vac_est21 <- sum(hu21_vac$FW)
total_est21 <- sum(hu21$FW)
est21 <- (vac_est21 / total_est21) * 100

# Calculating 2017 point estimates:
hu17_vac <- hu17 %>% filter(vac == 1)
vac_est17 <- sum(hu17_vac$fw)
total_est17 <- sum(hu17$fw)
est17 <- (vac_est17 / total_est17) * 100

# Final Point Estimate:
point_est <- est21 - est17

# Now looping:
reps <- rep(0,80)
for (i in 1:80) {
 vac_rep21 <- sum(hu21_vac[,paste0("FW", as.character(i))])
 tot_rep21 <- sum(hu21[,paste0("FW", as.character(i))])
 vac_rep17 <- sum(hu17_vac[,paste0("fw", as.character(i))])
 tot_rep17 <- sum(hu17[,paste0("fw", as.character(i))])
 reps[i] <- ((vac_rep21 / tot_rep21) - (vac_rep17 / tot_rep17)) * 100

}

reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(se_est , 4))))
```

The R command of Figure 6.6.1 produces the output in Figure 6.6.2.

Figure 6.6.2: R Output for Estimating Variance of a Longitudinal Change

```
> # Final Point Estimate:
> est
[1] 3.006778
> # Final Estimate of Standard Error:
> est_se
[1] 0.403841
```

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

The change in the gross vacancy rate from 2017 to 2021 is a 3.0 percent increase, with a standard error of 0.4 percent.

Example 6.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

There are many different statistics that measure the change between two cycles of NYCHVS. Differences, percent change, and ratios can be used to measure how much a given statistic changed from one cycle of NYCHVS to another. This example considers the housing characteristic of the median gross rent and how it can change over time. Both the rate of change and resulting variance are demonstrated.

The statistic of interest is the percent change in the median gross rent from 2017 to 2021. Let $\widehat{M}_t$ be the estimator of the median gross rent at time *t.* The statistic of interest is:

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we first calculate the point estimates of $\%\hat{\Delta}_t$. Then, we use both the 2021 and 2017 replicate weights to store 80 replicate estimates of $\%\hat{\Delta}_t$ within a vector and store the squared differences between these replicate estimates and the final point estimate in another vector. We apply Equation (2.1) to arrive at our final estimate of standard error for $\%\hat{\Delta}_t$.

Figure 6.7.1 shows how this can be done with R.

Figure 6.7.1: R Command for Estimating Variance of a Rate of Change

```
# Calculating 2021 point estimates:
rent21 <- wgt_per_sas(rent_data, "gross_rent", "FW", 0.5)

# Calculating 2017 point estimates:
rent_data_17 <- hu17 %>% filter(uf26 != 99999 & sc116 %in% c(2,3))
rent17 <- wgt_per_sas(rent_data_17, "uf26", "fw", 0.5)

# Final Point Estimate:
point_est <- (rent21 - rent17) / rent17

# Now looping:
reps <- rep(0,80)
for (i in 1:80) {
 rent21_est <- wgt_per_sas(rent_data, "gross_rent",
         paste0("FW", as.character(i)), 0.5)
 rent17_est <- wgt_per_sas(rent_data_17, "uf26",
         paste0("fw", as.character(i)), 0.5)
 reps[i] <- (rent21_est - rent17_est) / rent17_est

}

reps <- (reps - point_est)^2
```

Figure 6.7.1: R Command for Estimating Variance of a Rate of Change

```
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(se_est, 4))))
```

The R code of Figure 6.7.1 produces the output in Figure 6.7.2.

Figure 6.7.2: R Output for Estimating Variance of a Rate of Change

```
# Final Point Estimate:
[1] 0.0862
# Final Estimate of Standard Error:
[1] 0.0126
```

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

So, the median gross rent for NYC in 2021 has increased 8.6 percent from 2017, with a standard error of 1.3 percent.

7. Examples of Calculating Variances with Python

In addition to SAS, STATA, and R, data users can also use Python to calculate variances for NYCHVS. This section reviews the same examples using Python, which produce the same estimates and standard errors as the SAS, STATA, and R.

Example 7.1. Estimating the Variance of a Total

For 2021 NYCHVS, all the PUFs are given as CSV files, which makes it easier for data users to import into Python. Data users can access the PUFs through the Census Bureau's NYCHVS website at: https://www.census.gov/programs-surveys/nychvs/data/datasets.html. You can use the read_csv function in the Pandas Python package to import the datafiles. Before starting, users also need to import some statistical packages in Python. Figure 7.1.1 shows how to accomplish these tasks in Python.

Figure 7.1.1: Python Code to Importing Public Use Files and Set Up Domains

```
import pandas as pd
import numpy as np
import statsmodels.api as sm

hu21 = pd.read_csv('LOCATION OF CSV FILES/allunits_puf_21.csv')
hu21occ = pd.read_csv('LOCATION OF CSV FILES/occupied_puf_21.csv')
hu21vac = pd.read_csv('LOCATION OF CSV FILES/vacant_puf_21.csv')
occvac = pd.concat([hu21occ, hu21vac])
occvac = occvac.drop(occvac.loc[:, "FW":"FW80"].columns, axis=1)
hu21 = hu21.merge(occvac, on="CONTROL")
```

In 2017 and prior, the NYCHVS PUFs were ACSII text files, and there were no column/variable names on these files; therefore, similar to R, data users still need to use the SAS Import Program (provided in the Census Bureau's NYCHVS website at https://www.census.gov/data/datasets/2017/demo/nychvs/microdata.html to read in the NYCHVS PUFs into SAS (please see Pages 7 -9 for instructions on how to read PUFs into SAS for 2017 and prior). Once in SAS, data users need to run the code in Figures 2.1.3 and 2.1.4, which generates the dataset HU17. Then, users can export the datasets from SAS to CSV format to read the file into Python.

In SAS, run the code in Figure 7.1.2 to convert the SAS dataset HU17 to a CSV dataset. Then, run the code show in Figure 7.1.3 to read in the dataset, and convert weights to the right decimal places.

Figure 7.1.2: SAS Code to Output Files into Python Readable Format

```
* Exporting NYCHVS datasets from SAS to CSV file;
proc export data = HU17 dbms = csv outfile="PATH \ TO \ NEW PYTHON
DATASET\new_dataset_name2.csv" replace;
run;
```

Figure 7.1.3: Python Code for Read in 2017 Data and Convert Weights

```
hu17 = pd.read_csv('LOCATION OF CSV FILES/ new_dataset_name2.csv')
for col in hu17.filter(regex = ("^fw.*")).columns:
    hu17[col] = hu17[col] / 100000
```

Also, prior to running any of the examples in Section 7, ensure that the following code is used to set up the additional domains found in the examples. See Figure 7.1.4 for the code to use for domain definitions.

Figure 7.1.4: Python Code for Domain Definitions

```
hu21["ph_bronx"] = np.where((hu21.CSR == 5) & (hu21.BORO == 1), 1, 0)
hu21["ph_man"] = np.where((hu21.CSR == 5) & (hu21.BORO == 3), 1, 0)
hu21["renters"] = np.where((hu21.RENT_AMOUNT != -2) & (hu21.GRENT > 0),
1, 0)
hu21["gross_rent"] = np.where(hu21.GRENT > 0, hu21.GRENT, np.nan)
hu21["defect"] = np.where(hu21.MDEFCOUNT >= 3, 1, 0)
hu21["vac"] = np.where(hu21.OCC != 1, 1, 0)
hu17["vac"] = np.where(hu17.recid == 3, 1, 0)
hu21["time_stable"] = np.where(hu21.YEARBUILT < 5, 1, 0)
```

In this example, we estimate the total occupied housing units in NYC and use the replicate weights to calculate the variance. Since we have already identified the domain of interest, now we are ready to run the total and variance calculation by using the Python code in Figure 7.1.5. The resulting output is provided in Figure 7.1.6.

Figure 7.1.5: Python Code for Estimating Variance of a Total

```
reps = np.array([0.0]*80)
point_est = hu21.loc[hu21.OCC == 1].FW.sum()
for i in range(1,81):
  reps[i-1] = hu21.loc[hu21.OCC == 1]["FW"+str(i)].sum()
reps = (reps - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print(point_est.round(decimals = 4), se.round(decimals = 4))
```

Figure 7.1.5: Python Output for Estimating Variance of a Total

```
3157105.1795  13438.7416
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 7.1.5, the estimate of total occupied housing units in NYC is 3,157,105 with an estimated standard error of 13,439. The estimates and standard errors from Python are the same as the results produced by SAS in Example 4.1.

Example 7.2. Estimating the Variance of a Difference

In this example, we are interested in calculating the difference between the number of public housing units in Manhattan and the number of public housing units in the Bronx for 2021.

The Python code of Figure 7.2.1 generates the output of Figure 7.2.2.

Figure 7.2.1: Python Code for Estimating Variance of a Difference

```
reps_man = np.array([0.0]*80)
reps_bronx = np.array([0.0]*80)
point_est = hu21.loc[hu21.ph_man == 1].FW.sum() - hu21.loc[hu21.ph_bronx ==
1].FW.sum()
for i in range(1,81):
  reps_man[i-1] = hu21.loc[hu21.ph_man == 1]["FW"+str(i)].sum()
  reps_bronx[i-1] = hu21.loc[hu21.ph_bronx == 1]["FW"+str(i)].sum()
reps = (reps_man - reps_bronx - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print(point_est.round(decimals = 4), se.round(decimals = 4))
```

Figure 7.2.2: Python Output for Estimating Variance of a Difference

```
7809.5658  603.7288
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 7.2.2, we can see the estimated difference of public housing units in Manhattan and the Bronx is 7,810 and the standard error is 604. These results are the same as produced with SAS – see Example 4.2.

Example 7.3. Estimating the Variance of a Mean

For this example, we are estimating the average gross rent for renter-occupied housing units in NYC for 2021 and its variance. The Python code is provided in Figure 7.3.1.

Figure 7.3.1: Python Code for Estimating Variance of a Mean

```
# Weighted average function:
def wgt_avg(df, value, weight):
  val = df[value]
  wt = df[weight]
  return((val * wt).sum() / wt.sum())

reps = np.array([0.0]*80)
rent_data = hu21.loc[hu21.TENURE == 1 & hu21.gross_rent.notnull()]
point_est = wgt_avg(rent_data, 'gross_rent', 'FW')
for i in range(1,81):
  reps[i-1] = wgt_avg(rent_data, 'gross_rent', 'FW'+str(i))
reps = (reps - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print(point_est.round(decimals = 4), se.round(decimals = 4))
```

The Python code of Figure 7.3.1 generates the output of Figure 7.3.2.

Figure 7.3.2: Python Output for Estimating Variance of a Mean

```
1726.739  15.8292
```

Source: U.S. Census Bureau, 2021, New York City Housing and Vacancy Survey.

From Figure 7.3.2, we can see that the average NYC gross rent calculated using Python is $1,727 and the corresponding standard error is $16, both of which are the same as was produced with SAS – see Examples 3.1 and 4.3.

Example 7.4. Estimating the Variance of a Median

For this example, we are interested in calculating the median gross rent and its standard error for renter-occupied housing units in NYC for 2021. Figure 7.4.1 shows the Python code for calculating variance of a median. The domain of interest is the same as the previous example, and the code for the domain identification can be found in Example 7.1.

Figure7.4.1: Python Code for Estimating Variance of a Median

```
# Weighted percentile function:
def wgt_per_sas(df, value, weight, percentile):
  dat = df.loc[:, [value, weight]].sort_values(by=value)
  dat['cum_sum'] = dat[weight].cumsum()
  p_wgt = percentile * dat[weight].sum()
  dat['near_median'] = dat['cum_sum'] - p_wgt
```

```
  minimum = dat.loc[dat['near_median'] >= 0]['near_median'].min()
  if (minimum == 0):
    indices = dat[dat['near_median'] == 0].index.values[0]
    grab_xi_and_next = dat.loc[[indices, indices+1]]
    return((1/2)*(grab_xi_and_next[value].sum()))
  else:
    index = dat[dat['near_median'] == minimum].index.values[0]
    return(dat.loc[index][value])

reps = np.array([0.0]*80)
point_est = wgt_per_sas(rent_data, 'gross_rent', 'FW', 0.5)
for i in range(1,81):
  reps[i-1] = wgt_per_sas(rent_data, 'gross_rent', 'FW'+str(i), 0.5)
reps = (reps - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print("Ex 7.4:", point_est.round(decimals = 4), se.round(decimals = 4))
```

The Python code of Figure 7.4.2 generates the output of Figure 7.4.3.

Figure 7.4.3: Python Output for Estimating Variance of a Median

```
1575.0  14.8997
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimate for median gross rent for renter-occupied housing units in NYC is $1,575, with standard error of $15. The results are the same as produced by SAS in Example 4.4.

Example 7.5. Estimating the Variance of a Regression Coefficient

For this example, we are interested in modeling the 2021 gross rent of renter-occupied housing units using the year the householder moved into the housing unit.

The code is provided in Figure 7.5.1.

Figure 7.5.1: Python Code for Estimating Variance of a Regression Coefficient

```
reps = np.array([0.0]*80)
y = rent_data["gross_rent"]
x = rent_data["HHFIRSTMOVEIN"]
x = sm.add_constant(x)
fitted = sm.WLS(y,x, weights = rent_data["FW"]).fit()
point_est = fitted.params[1]
for i in range(1,81):
```

```
  reps[i-1] = sm.WLS(y,x, weights = rent_data["FW"+str(i)]).fit().params[1]
reps = (reps - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print( point_est.round(decimals = 4), se.round(decimals = 4))
```

The Python code of Figure 7.5.1 generates the output of Figure 7.5.2.

Figure 7.5.2: Python Output for Estimating Variance of a Regression Coefficient

```
23.6473  1.2409
```

Source: U.S. Census Bureau, 2021, New York City Housing and Vacancy Survey.

From Figure 7.5.2, we can see that the calculated regression coefficient is 23.6473 and the standard error is 1.2409. The estimates are the same produced by SAS – see Examples 3.3 and 4.5.

Example 7.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 7.7 will consider an estimate of percent of change that is calculated at the HU level.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we are interested in the estimated gross vacancy rate or $\hat{p}_t = \hat{X}_t / \widehat{N}_t * 100$, where $\hat{X}_t$ is the estimator of the total number of vacant HUs at time *t*, and $\widehat{N}_t$ is the estimator of the total number of HUs at time *t.*

Further, we are interested in the difference in the gross vacancy rate between 2021 and 2017, so the statistic of interest is:

$$\hat{\Delta}_{t=2021} = \frac{\hat{X}_{t=2021}}{\widehat{N}_{t=2021}} * 100 - \frac{\hat{X}_{t=2017}}{\widehat{N}_{t=2017}} * 100$$

To estimate the variance of $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces of $\hat{\Delta}_{t=2021}$ for each of the 80 replicates, making 80 replicate estimates of $\hat{\Delta}_{t=2021}$, repeat those steps for 2017, and then apply Equation (2.1).

Figure 7.6.1 shows how we do this with Python.

Figure 7.6.1: Python Code for Estimating Variance of a Longitudinal Change

```
reps = np.array([0.0]*80)
vac21 = (hu21.loc[hu21.vac == 1].FW.sum() / hu21.FW.sum())*100
vac17 = (hu17.loc[hu17.vac == 1].fw.sum() / hu17.fw.sum())*100
point_est = vac21 - vac17
for i in range(1,81):
  vac21 = (hu21.loc[hu21.vac == 1]["FW"+str(i)].sum() /
hu21["FW"+str(i)].sum())*100
  vac17 = (hu17.loc[hu17.vac == 1]["fw"+str(i)].sum() /
hu17["fw"+str(i)].sum())*100
  reps[i-1] = vac21 - vac17
reps = (reps - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print( point_est.round(decimals = 4), se.round(decimals = 4))
```

The Python code of Figure 7.6.1 generates the output of Figure 7.6.2.

Figure 7.6.2: Python Output for Estimating Variance of a Longitudinal Change

```
3.0068 0.4038
```

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

So, the change in the gross vacancy rate from 2017 to 2021 is a 3.0 percent increase, with a standard error of 0.4 percent.

Example 7.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

There are many different statistics that measure the change between two cycles of NYCHVS. Differences, percent change, and ratios can be used to measure how much a given statistic changed from one cycle of NYCHVS to another. This example considers the housing characteristic of the median gross rent and how it can change over time. Both the rate of change and resulting variance are demonstrated.

The statistic of interest is the percent change in the median gross rent from 2017 to 2021. Let $\widehat{M}_t$ be the estimator of the median gross rent at time *t.* The statistic of interest is:

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we use the 2021 replicate weights and store 80 replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2021}}$ within a matrix, and similarly use the 2017 replicate weights to store 80 replicate estimates of $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ in another matrix. Next, we take the difference of these two matrices $\widehat{\mathrm{M}}_{\mathrm{t=2021}}$ and $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ by replicate and then divide each matrix entry by $\widehat{\mathrm{M}}_{\mathrm{t=2017}}$ to get our 80 replicate estimates of $\%\hat{\Delta}_t$. The final step is to apply Equation (2.1) to the replicate estimates of $\%\hat{\Delta}_t$.

Figure 7.7.1 shows how this can be done with Python.

Figure 7.7.1: Python Code for Estimating Variance of a Rate of Change

```
rent_data_17 = hu17.loc[(hu17.sc116.isin([2,3])) & (hu17.uf26 != 99999)]
reps = np.array([0.0]*80)
point_est_21 = wgt_per_sas(rent_data, 'gross_rent', 'FW', 0.5)
point_est_17 = wgt_per_sas(rent_data_17, 'uf26', 'fw', 0.5)
point_est = (point_est_21 - point_est_17) / point_est_17
for i in range(1,81):
  point_est_21 = wgt_per_sas(rent_data, 'gross_rent', 'FW'+str(i), 0.5)
  point_est_17 = wgt_per_sas(rent_data_17, 'uf26', 'fw'+str(i), 0.5)
  reps[i-1] = (point_est_21 - point_est_17) / point_est_17
reps = (reps - point_est)**2
se = ((4/80)*np.sum(reps))**(1/2)
print(point_est.round(decimals = 4), se.round(decimals = 4))
```

The Python code of Figure 7.7.1 generates the output of Figure 7.7.2.

Figure 7.7.2: Python Output for Estimating Variance of a Rate of Change

```
0.0862 0.0126
```

Source: U.S. Census Bureau, 2017 and 2021 New York City Housing and Vacancy Survey.

Therefore, the median gross rent for NYC in 2021 has increased 8.6 percent from 2017, with a standard error of 1.2 percent.

8. How to Calculate Confidence Intervals

Lohr (1999; Section 9.5) has an excellent review of confidence intervals (CIs) for survey estimates, from which we borrow heavily for the introduction of this section.

Survey estimates that employ sample weights are different than unweighted estimates. However, under certain conditions, it can be shown that $(\hat{\theta} - \theta)/\sqrt{\hat{v}(\hat{\theta})}$ is asymptotically standard normal for survey estimates (Krewski and Rao, 1981). Consequently, when the assumptions are met, an approximate 95 percent CI for θ may be constructed as:

$$\hat{\theta} \pm Z_{\alpha=0.05}\sqrt{\hat{v}(\hat{\theta})} = \hat{\theta} \pm 1.96\sqrt{\hat{v}(\hat{\theta})} \qquad (8.1)$$

where $\sqrt{\hat{v}(\hat{\theta})}$ is the standard error of the estimator of $\hat{\theta}$.

Normal Distribution versus the *t*-distribution for Confidence Intervals

In this document, we suggest using a normal distribution with all CIs. We suggest this for two reasons. First, there are no good guidelines for the number of degrees of freedom that should be used with a t-distribution and variances generated from SDR. There are recommendations for degrees of freedom for clustered and stratified sample designs, for example, see Valiant and Rust (2010), Korn and Graubard (1999), and Eltinge and Jang (1996). However, the degrees of freedom for clustered and stratified sample designs do not apply to the multi-stage systematic random sample design employed by NYCHVS. Huang and Bell (2011) examined degrees of freedom for SDR, but they do not provide general guidance. The second reason for our suggestion is that the American Community Survey (ACS) has the same sample design and also suggests using *z*-values from a normal distribution (U.S. Census Bureau, 2014). Until more research is done in this area, we suggest following the lead of ACS.

We have not covered hypothesis testing, but we do point the reader to Schenker and Gentleman (2001) who discuss the pitfalls of using confidence intervals in hypothesis testing.

Table 8 is a compilation of examples from throughout the guide. For each of these examples, we calculate a 95 percent confidence interval.

Table 8: Summary Table of Examples Estimated Confidence Intervals

Example	Example Number					Estimate	Standard Error	95% Confidence Intervals
	PROC SURVEY	Base SAS	STATA	R	Python			
Total		4.1	5.1	6.1	7.1	3,157,105	13,439	(3,130,765, 3,183,445)
Difference		4.2	5.2	6.2	7.2	7,810	604	(6,626, 8,994)
Mean	3.1	4.3	5.3	6.3	7.3	$1,726.74	$15.83	($1,695.71, $1,757.77)
Median		4.4	5.4	6.4	7.4	$1,575	$14.90	($1,545.80, $1,604.20)
Regression Coefficient	3.3	4.5	5.5	6.5	7.5	23.65	1.24	(21.18, 26.12)
Longitudinal Change		4.6	5.6	6.6	7.6	3.01%	0.40%	(2.23%, 3.79%)
Percent Change		4.7	5.7	6.7	7.7	8.6%	1.26%	(6.13%, 11.07%)

Source: U.S Census Bureau, 2021 New York City Housing and Vacancy Survey.

The standard errors of Table 8 were produced with SAS, STATA, R, and Python.

The subsequent examples of this section demonstrate how to estimate CIs using SAS, STATA, R and Python with some of the earlier examples.

Example 8.1. Estimating Confidence Intervals of a Total with Base SAS

In this example, we demonstrate how users calculate the CIs of the total in Example 4.1. We will reuse the dataset Data3 that was already created and build upon the SAS code in Figure 4.1.2 to add the CI calculations. Figure 8.1.1 shows the code in SAS, and Figure 8.1.2 shows the output.

Figure 8.1.1: SAS Code for Estimating Variance of a Total with Confidence Intervals

```
data data4 (keep = est var se lowerci upperci) ;
 set data3 end=eof ;
 * Fill array with the replicate sums ;
 array repwts{80} rw1-rw80 ;
 * Fill array with the squared diffs ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (repwts{j} - est)**2 ;
  end ;
 * Sum the squared diffs ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = (var)**(0.5) ;
 *Calculate the endpoints of the confidence intervals. ;
 lowerci= est - 1.96 * se;
 upperci = est + 1.96 * se;
 output ;
run ;

proc print data=data4 noobs ;
 var est se lowerci upperci;
 format est se lowerci upperci comma12.2 ;
run ;
```

In Figure 8.1.2, we used the simple numeric value of 1.96 as the *z*-value with $\alpha = 0.05$ level because it makes the code easier to read. We could replace this with the more exacting SAS function `quantile ("Normal", 1 - alpha/2)` where `alpha = 0.05`.

Figure 8.1.2: SAS Output of Variance of Total with Confidence Intervals

est	se	lowerci	upperci
3,157,105.18	13,438.74	3,130,765.25	3,183,445.11

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

In addition to the output we saw in Example 4.1, now we get the CIs as well: the 95 percent CI for the total number of occupied housing units is (3,130,765, 3,183,445).

Example 8.2. Estimating Confidence Intervals of a Regression Coefficient with SAS PROC SURVEYREG

In this example, we demonstrate how users can use the SAS PROC SURVEYREG in Example 3.3 and add a `clparm` option in the `model` statement to get the estimates for confidence intervals.

Figure 8.2.1: SAS Code for Estimating Variance of Regression Coefficients with Confidence Intervals

```
proc surveyreg data=HU21 varmethod=brr(fay) ;
  *subset to renters with valid rent only;
 where tenure=1 and Grent not in (-2,-1) ;
 domain tenure;
 model Grent = HHFIRSTMOVEIN/ solution clparm ;
 weight FW ;
 repweights fw1-fw80 ;
 ods output parameterEstimates = MyParmEst ;
run;

data MyParmEstfin ;
 set MyParmEst ;
 if _n_ = 2 ;
 se = stderr ;
 var = se**2 ;
 drop parameter dendf tvalue probt stderr ;
run ;

proc print data=myparmestfin;
 var estimate se lowercl uppercl ;
 format estimate se lowercl uppercl 8.4 ;
run;
```

The SAS option `clparm` in the model statement of Figure 8.2.1. requests the confidence interval for the regression coefficients. The procedure generates the confidence interval as the variables `lowercl` and `uppercl`. The SAS code from Figure 8.2.1 produces the output of Figure 8.2.2.

Figure 8.2.2: SAS Output for Estimating Confidence Intervals Using t-Distribution

Estimate	se	LowerCL	UpperCL
23.6473	1.2409	21.1779	26.1167

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The point estimate and variances are the same as in Example 3.3; however, there are the two added columns at the end of the output for the confidence intervals. Please note that the confidence intervals constructed using SAS SURVEYREG are defaulted to t-distribution.

To construct confidence intervals using the normal distribution, users have to code this manually in Base SAS. Figure 8.2.3 shows the SAS code needed, in addition to Figure 8.2.1, to get the confidence intervals using normal distribution, and Figure 8.2.4 shows the results. Note that this confidence interval is slightly smaller than that using the t-distribution.

Figure 8.2.3: SAS Code for Estimating Variance of Regression Coefficients with Confidence Intervals

```
*Calculate CI using normal distribution;
data one;
      set myparmestfin;
lowerci= estimate - 1.96*se;
upperci= estimate + 1.96*se;
run;

proc print; var estimate se lowerci upperci; run;
```

Figure 8.2.4: SAS Output for Estimating Confidence Intervals Using z-Distribution

Estimate	se	lowerci	upperci
23.647	1.24086	21.2152	26.0794

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Example 8.3. Estimating Confidence Intervals of a Total with STATA Manually

Users might notice that many STATA outputs in Section 5 already include CIs. We now demonstrate how users can manually calculate CIs in STATA. Figure 8.3.2 provides output from running this code after Setting survey design parameters (Figure 5.1.5).

Figure 8.3.1.: STATA Code for Estimating Confidence Intervals Manually

```
use "puf/alloccvac_21", clear

* Calculating 2021 occupied totals:
quietly summarize fw if occ == 1
scalar point_est = r(sum)

* Creating matrix that stores 2021 estimates:
matrix occ_ests_21 = (point_est)

* Now going through replicates:
foreach x of var fw1-fw80 {

     quietly summarize `x' if occ == 1
     scalar occ_ests_21_i = r(sum)

     * Adding to matrix that stores 2017 estimates:
     matrix occ_ests_21 = (occ_ests_21 \ occ_ests_21_i)
}

* Now getting squarred differences for variance estimation:
scalar summed_variance = 0
quietly forvalues i = 2/81 {
     scalar sq_pt_occ_est = (occ_ests_21[`i',1] - occ_ests_21[1,1])^2
     scalar summed_variance = summed_variance + sq_pt_occ_est
}
scalar summed_variance = (4/80) * summed_variance
scalar se_est = (summed_variance)^(0.5)

* Final Point Estimate:
display point_est

* Final Estimated Standard Error:
display se_est

* 95% CI using standard normal (z) distribution:
scalar z_value = invnorm(.975)
display "(",point_est-z_value*se_est,",",point_est+z_value*se_est,")"
```

Figure 8.3.2: STATA Output for Estimating Confidence Intervals Manually

```
. display "(",point_est-z_value*se_est,",",point_est+z_value*se_est,")"
( 3130765.7 , 3183444.6 )
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 8.3.2, we see that there is a 95 percent chance that the interval of (3,130,765.7, 3,183,444.6) captures the true percent increase in median gross rent between the 2017 and 2021 survey cycles of NYCHVS. Comparing STATA's output with other methods, STATA produced the same confidence interval as the other methods.

Example 8.4. Estimating the Confidence Intervals of a Total with R

We now demonstrate how users can manually calculate CIs in R. Figure 8.4.1 shows code that will work after running any of the example code between Examples 6.1 and 6.7. Figure 8.4.2 provides output from running this code after running Example 6.1 only.

Figure 8.4.1: R Code for Estimating Confidence Intervals

```
# For a 95% CI, we need 2.5% on either tail, so get the 97.5% percentile
z_value <- qnorm(.975)

ci_95 <- c(point_est - z_value*se_est, point_est + z_value*se_est)
print(ci_95)
```

The R code provided in Figure 8.4.1 produces the R output of Figure 8.4.2 when applied to Example 6.7.

Figure 8.4.2: R Output for Estimating Confidence Intervals

```
print(ci_95)
[1] 3130766 3183445
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 8.4.2, we see that R produces the same exact confidence interval as the other methods.

Example 8.5. Estimating the Confidence Intervals of a Total with Python

We now demonstrate how users can manually calculate CIs in Python. Figure 8.5.1 shows code that will work after running any of the example code between Examples 7.1 and 7.7. Figure 8.5.2 provides output from running this code after running Example 7.7 only.

Figure 8.5.1: Python Code for Estimating Confidence Intervals

```
z_value = sps.norm.ppf(0.975)
ci = [point_est - z_value*se_est, point_est + z_value*se_est]
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Figure 8.5.2: Python Output for Estimating Confidence Intervals

```
print(ci)
[3130765.7300400576, 3183444.6289621126]
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From Figure 8.5.2, we see that Python produces the same confidence interval as the other methods.

9. How to Calculate Confidence Intervals for the Odds Ratio

This extended example shows how to calculate and how not to calculate a CI for the odds ratio. We show this by addressing the question "**Does rent stabilization status (pre-1947 or post-1947) affect the probability of having three or more maintenance deficiencies?**" with results from the 2021 NYCHVS. This example is different and much longer than Example 8.1 for two reasons: (1) the complexity of the estimator and the variance of the estimator and (2) we show how to calculate the CIs correctly, as well as several ways the confidence intervals should not be calculated.

As with the other statistics of this guide, using the replicate weights will best estimate the variance of the odds ratio. Table 9.1 summarizes the alternative methods that we will discuss and provides an overview of the estimates for each method.

Table 9.1: Examples of Odds Ratio

Example	Uses	Software	Odds Ratio $(\hat{\theta})$	$\hat{v}(\log \text{odds})$	$\hat{se}(\log \text{odds})$	95% Confidence Interval of $\hat{\theta}$
9.1	Replicate Wgts	SURVEYLOGISTIC	2.257	*	*	(1.628, 3.130)
9.2	Replicate Wgts	SURVEYFREQ	2.2572	*	*	(1.6356, 3.1151)
9.3	Replicate Wgts	Base SAS	2.2572	0.0270	0.1642	(1.6359, 3.1144)
9.4	Replicate Wgts	STATA	2.2572	0.0270	0.1642	(1.6359, 3.1144)
9.5	Replicate Wgts	R	2.2572	0.0270	0.1642	(1.6359, 3.1144)
9.6	Replicate Wgts	Python	2.2572	0.0270	0.1642	(1.6359, 3.1144)
9.7	Final Wgts	SURVEYFREQ	2.2572	*	*	(1.6945, 3.0068)
9.8	Normalized Wgts	FREQ	2.2572	0.0182	0.1350	(1.7325, 2.9408)
9.9	Final Wgts	FREQ	2.2572	0.0000	0.0061	(2.2305, 2.2842)
9.10	Unweighted	FREQ	2.2959	0.0169	0.1301	(1.7793, 2.9626)

* SAS SURVEYLOGISTIC and SURVEYFREQ procedures do not output variance or standard errors.
Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

We make a few observations about the results of Table 9.1.

First, the weighted and unweighted estimates of the odds ratio differ slightly: the unweighted estimate of the odds ratio is 2.2959 and the weighted estimate is 2.2572. Using the weights accounts for the fact that sample units contribute unequally to the estimates.

With the variance of the log odds and the standard errors, we note the following differences from the Table 9.1:

- Base SAS, STATA, R, and Python (Examples 9.3 to 9.6) all produce the same estimate of the odds ratio and standard error of the log odds using the replicate weights.
- Estimating the variance using normalized weights, final weights, and no weights (Example 9.8, 9.9, and 9.10, respectively), produce variances that are an underestimate.

Only using the replicate weights (Examples 9.1 to 9.6) fully accounts for the complex sample design of NYCHVS.

- Using the final weights with PROC SURVEYFREQ and the normalized weights with PROC FREQ (Examples 9.7 and 9.8, respectively) provide the same estimate, but we caution that it may not be true with all analyses.
- Example 9.9 using the final weights underestimates the variance by a large amount.

With the CI calculations, we note the following differences in Table 9.1.

- SAS PROC SURVEYLOGISTIC and SAS PROC SURVEYFREQ (Examples 9.1 and 9.2, respectively) produced different CIs because the PROC SURVEY procedures uses a critical value from a t-distribution with 80 degrees of freedom. Basically, SURVEYLOGISTIC is assuming that we have a stratified sample design with 80 strata. See Section 3 for a discussion.
- The CIs produced by SURVEYFREQ procedure (Example 9.2) are different from the results of SURVEYLOGISTIC (Examples 9.1) because SURVEYFREQ uses a different approximation in the variance calculation, which is not output by the procedure, but you see the effect in the difference in the confidence interval calculation. This is discussed more in Section 9.2.
- Base SAS, STATA, R, and Python (Examples 9.3 to 9.6) produced the same CIs using replicate weights.
- Estimating the variance using final weights (Example 9.9) underestimates variance by a large amount, which therefore results in a much narrower CI.

The remainder of this extended example is organized as follows. First, we generally review the odds ratio. This is provided to define our use of different terms and for readers not familiar with an odds ratio. Readers familiar with this material can skip this section.

Then each of the ten methods for estimating the CI is reviewed separately. Although we say that only Examples 9.1 to 9.6 estimate the CI accurately because replicate weights account for the complex sample design of NYCHVS, we provide the four other methods because some data users are familiar with them and they provide excellent comparisons to Examples 9.1 to 9.6.

We have color-coded the odds ratio, variance of the log odds ratio, and the confidence interval of the odds ratio throughout the example as green, purple, and light blue, respectively. This was done to help keep track of the different statistics of the examples.

Review of Odds Ratio and Log Odds

Suppose we have two sets of binary random numbers, X and Y, and we want to measure the association between the two variables. We assume both random variables have a Bernoulli distribution. Table 9.2 shows the frequencies observed in the sample for all possible combinations of *X* and *Y*.

Table 9.2: Layout of Two Binary Variables

	$X = 0$	$X = 1$	Total
$Y = 0$	n_{00}	n_{01}	n_{0+}
$Y = 1$	n_{10}	n_{11}	n_{1+}
Total	n_{+0}	n_{+1}	n

The *odds* of a single binary random variable Y are defined to be the probability of $Y = 1$ divided by the probability of $Y = 0$, i.e.,

$$\{\text{Odds of } Y\} = \frac{P(Y=1)}{P(Y=0)}$$

In the presence of the variable X, the odds of Y given $X = 0$ is

$$\{Odds\ of\ Y\ given\ X=0\} = \frac{P(Y=1|X=0)}{P(Y=0|X=0)}$$

and the odds of Y given $X = 1$ is

$$\{Odds\ of\ Y\ given\ X=1\} = \frac{P(Y=1|X=1)}{P(Y=0|X=1)}$$

The vertical bar in these probability statements is the traditional symbol for conditional probabilities.

The odds ratio θ compares the odds of Y for $X = 0$ and $X = 1$ and is defined as the ratio of the two odds:

$$\theta = \frac{\{\text{Odds of } Y \text{ given } X=0\}}{\{\text{Odds of } Y \text{ given } X=1\}} = \frac{\dfrac{P(Y=1|X=0)}{P(Y=0|X=0)}}{\dfrac{P(Y=1|X=1)}{P(Y=0|X=1)}} = \frac{P(Y=1|X=0)P(Y=0|X=1)}{P(Y=0|X=0)P(Y=1|X=1)}$$

An odds ratio of $\theta = 1$ suggests that the odds of Y is the same for $X = 0$ and $X = 1$.

In the presence of the variable X, the odds of Y given $X = 1$ is:

$$\{\text{Odds of } Y \text{ given } X=1\} = \frac{(n_{11}/n_{+1})}{(n_{01}/n_{+1})} = \frac{n_{11}}{n_{01}} \qquad (9.1)$$

and we can estimate the odds ratio as:

$$\hat{\theta} = \frac{(n_{11}/n_{+1})(n_{00}/n_{+0})}{(n_{01}/n_{+1})(n_{10}/n_{+0})} = \frac{n_{11} \cdot n_{00}}{n_{01} \cdot n_{10}} \tag{9.2}$$

The distribution of the odds ratio statistic has an atypical form, which unfortunately makes it hard to use for the purpose of statistical inferences. The natural log of the odds ratio, or simply log odds ratio, by contrast, is approximately normally distributed, provided the cell counts within each of the four categories above are sufficiently high. Consequently, the variance of the log odds can be estimated as:

$$\hat{v}(\log \text{ odds}) = \frac{1}{n_{00}} + \frac{1}{n_{01}} + \frac{1}{n_{10}} + \frac{1}{n_{11}} \tag{9.3}$$

We can use the variance of the log odds to calculate a (1 - α) CI of the odds ratio by transforming the CI of the log odds as:

$$\left(e^{\log \text{ odds} + z_{\alpha/2} \cdot \sqrt{\hat{v}(\log \text{ odds})}}, e^{\log \text{ odds} - z_{\alpha/2} \cdot \sqrt{\hat{v}(\log \text{ odds})}} \right) \tag{9.4}$$

The Maintenance Deficiencies Example. Returning to maintenance deficiencies example, we ask the question:

"**Does rent stabilization status affect the probability of having three or more maintenance deficiencies?**"

To answer this, we have a binary random variable *Y*, with *Y* = 1 indicating "having three or more maintenance deficiencies" and *Y* = 0 indicating "having fewer than three maintenance deficiencies":

$$Y = \begin{cases} 1, & \text{HU has three or more maintenance deficiencies} \\ 0, & \text{HU has fewer than three maintenance deficiencies} \end{cases}$$

We also define a second binary random variable *X* for rent stabilization status as:

$$X = \begin{cases} 1, & \text{HU has rent stabilized pre} - 1947 \\ 0, & \text{HU has rent stabilized post} - 1947 \end{cases}$$

If we calculate the odds ratio *Y* given *X*, a value of θ > 1 implies that rent stabilized pre-1947 HUs are more likely than the rent stabilized post-1947 units to have three or more maintenance deficiencies and, conversely, θ < 1 implies rent stabilized post-1947 units are more likely to have three or more maintenance deficiencies.

Example 9.1. Estimating a Confidence Interval of an Odds Ratio with Replicate Weights and SAS PROC SURVEYLOGISTIC

The easiest way of estimating odds ratios with replicate weights is to use SAS SURVEYLOGISTIC. This procedure produces the odds ratio estimates and CIs with replicate weights automatically. Use the `varmethod brr(fay)` option and the default Fay coefficient of 0.5. Put the effect (rent stabilization status, in our example) in the `class` statement.

Figure 9.1.1 provides the PROC SURVEYLOGISTIC code that will calculate the odds ratios using the replicate weights.

Figure 9.1.1: SAS Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
Data Def_analysis ;
 set HU21;
 * Exclude the vacant HUs and the ones did not report on deficiencies. ;
 where MDEFCOUNT ne -1 and OCC=1 and csr = 32;
 if Mdefcount ge 3 then def='1'; else def='0' ;
 * pre-1947 ;
 If YEARBUILT LE 4 then stabilized='1' ;
 * post1947 ;
 else if YEARBUILT GE 5 then stabilized='0' ;
run ;

proc surveylogistic data=def_analysis varmethod=brr(fay=0.5) ;
 class stabilized ;
 model def=stabilized ;
 weight FW ;
 repweights fw1-fw80 ;
run ;
```

Figure 9.1.2 shows some of the output that is generated by the SAS code of Figure 9.1.1.

Figure 9.1.2: Partial SAS Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

The SURVEYLOGISTIC Procedure

Model Information

Data Set	WORK.DEF_ANALYSIS
Response Variable	def
Number of Response Levels	2
Weight Variable	FW
Model	Binary Logit
Optimization Technique	Fisher's Scoring

Number of Observations Read	1701
Number of Observations Used	1701
Sum of Weights Read	838760.7
Sum of Weights Used	838760.7

Response Profile

Ordered Value	def	Total Frequency	Total Weight
1	0	1293	634046.31
2	1	408	204714.36

Probability modeled is def='0'.

Variance Estimation

Method	BRR
Replicate Weights	DEF_ANALYSIS
Number of Replicates	80
Fay Coefficient	0.5

Odds Ratio Estimates

Effect	Point Estimate	95% Confidence Limits	
stabilized 0 vs 1	2.257	1.628	3.130

NOTE: The degrees of freedom in computing the confidence limits is 80.

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The bottom of Figure 9.1.2 mentions that 80 degrees of freedom were used to calculate the confidence interval. This tells us that the critical value was 1.99 (from a t-distribution with 80 degrees of freedom) instead of 1.96 (from a normal distribution). Although we are using SDR replicate weights, SAS assumes that we are using Balanced Repeated Replication (BRR) replicate weights. For this reason, the estimates from PROC SURVEYLOGISTIC are slightly different than the other methods.

Example 9.2. Estimating a Confidence Interval of an Odds Ratio with Replicate Weights and SAS PROC SURVEYFREQ

Another similar method as SURVEYLOGISTIC to estimate the CI for the odds ratio with the replicate weights in SAS is to use the procedure PROC SURVEYFREQ.

Figure 9.2.1 provides the PROC SURVEYFREQ code that will calculate the odds ratio using the replicate weights.

Figure 9.2.1: SAS Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
proc surveyfreq data=def_analysis varmethod=brr(fay) ;
 tables def * stabilized / or ;
 weight FW ;
 repweights fw1-fw80 ;
run ;
```

The use of the `or` option in the `tables` statement requests that SAS produce the odds ratio with both PROC SURVEYFREQ and PROC FREQ. Note that the output generated by the option `or` is formatted differently with PROC SURVEYFREQ and PROC FREQ.

The SAS code of Figure 9.2.1 generates the output of Figure 9.2.2.

Figure 9.2.2: SAS Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

The SURVEYFREQ Procedure

Data Summary

Number of Observations	1701
Sum of Weights	838760.666

Variance Estimation

Method	BRR
Replicate Weights	DEF_ANALYSIS
Number of Replicates	80
Fay Coefficient	0.500

Table of def by stabilized

def	stabilized	Frequency	Weighted Frequency	Std Err of Wgt Freq	Percent	Std Err of Percent
0	0	531	231825	10166	27.6390	1.0147
	1	762	402221	13102	47.9542	1.4080
	Total	1293	634046	15546	75.5932	1.2336
1	0	95	41640	5467	4.9645	0.6424

Figure 9.2.2: SAS Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

	1	313	163074	10982	19.4423	1.2204
	Total	408	204714	11455	24.4068	1.2336
Total	0	626	273465	11902	32.6035	1.1907
	1	1075	565296	14133	67.3965	1.1907
	Total	1701	838761	16652	100.0000	

Odds Ratio and Relative Risks (Row1/Row2)

Statistic	Estimate	95% Confidence Limits	
Odds Ratio	2.2572	1.6356	3.1151
Column 1 Relative Risk	1.7975	1.3985	2.3104
Column 2 Relative Risk	0.7964	0.7379	0.8594

Sample Size = 1701

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

The estimate of $\hat{\theta}$ = 2.2572 implies that housing units rent stabilized prior to 1947 are about twice as likely to have three or more maintenance deficiencies than HUs rent stabilized post-1947.

Please note that the CIs produced by SURVEYFREQ here are slightly different from the CIs produced by the other methods with replicate weights. This is due to two reasons: (1) the PROC SURVEY procedures use a critical value in the confidence intervals from a t-distribution with 80 degrees of freedom instead of using a critical value from a normal distribution) and (2) SURVEYFREQ uses a variance approximation when calculating the variance of the log odds, which is

$$\hat{v}(\log odds) = \frac{\hat{v}(odds\ ratio)}{(odds\ ratio)^2},$$

while the other methods calculate the variance of log odds directly using Equation (9.3) (SAS 9.2 User's Guide, 2019).

Example 9.3. Estimating a Confidence Interval of an Odds Ratio with Replicate Weights and Base SAS

A point estimate of the odds ratio is straightforward to calculate with weights – it just uses the sum of all the weights in each of the four categories and follows the procedure above. The variance estimate is more complicated to calculate manually.

Table 9.3.1 provides the weighted estimates of the frequencies of each cell. The final weight and Equation (9.2) were used to produce the totals.

Table 9.3.1: Estimated Counts for Maintenance Deficiencies Example

	Fewer than 3 maintenance deficiencies	3 or more maintenance deficiencies	Total
Rent Stabilized Post 1947	n_{00} = 231,825	n_{01} = 41,640	n_{0+} = 273,465
Rent Stabilized Pre-1947	n_{10} = 402,221	n_{11} = 163,074	n_{1+} = 565,296
Total	n_{+0} = 634,046	n_{+1} = 204,714	n = 838,761

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Using the weighted totals, the odds of having three or more maintenance deficiencies given the HU is rent stabilized prior to 1947 is calculated as:

$$\{\text{Odds of } Y \text{ given Rent Stabilized Pre} - 1947\ (X = 1)\} = \frac{163{,}074}{41{,}640} = 3.92$$

and we calculate the odds ratio as:

$$\hat{\theta} = \frac{231.825 * 163{,}074}{402{,}221 * 41{,}640} = 2.2572$$

The natural log of the odds ratio is *log odds* = 0.8141. To estimate the variance of the log odds, the log odds is calculated for each replicate and Equation (2.1) is applied. This gives $\hat{v}(\log odds)$ = 0.0270, which can be used with Equation (9.3) to calculate the CI:

$$\left(e^{0.8141-1.96\sqrt{\hat{v}(\log odds)}}, e^{0.8141+1.96\sqrt{\hat{v}(\log odds)}}\right)$$

The SAS code of Figure 9.3.3 shows how to calculate the variance, standard error, and CIs of the odds ratio directly.

Figure 9.3.3: SAS Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
* Define 4 categories. ;
data def_analysis ;
      set def_analysis ;
      if def='1' and stabilized='0' then n_xx='n_10' ;
      else if def='1' and stabilized='1' then n_xx='n_11' ;
      else if def='0' and stabilized='0' then n_xx='n_00' ;
      else if def='0' and stabilized='1' then n_xx='n_01' ;
run ;

* Calculate the total sums for each of our 4 categories ;
proc means data = Def_analysis sum noprint ;
 class n_xx ;
```

Figure 9.3.3: SAS Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
 var FW fw1-fw80 ;
 output out = def_collapse sum = est rw1 - rw80 ;
run ;

* Drop the first observation of totals ;
data def_collapse ;
 set def_collapse ;
 if _TYPE_ = 0 then delete ;
run ;

* Reshape into wide format to calculate ratio estimates
 with arrays later ;
proc transpose data = def_collapse out = def_collapse_transposed ;
 by _TYPE_ ;
 id n_xx ;
 var est rw1 - rw80 ;
run ;

* Now calculating log odds ratio estimate for newly transposed data. ;
data def_collapse_transposed_odds ( keep = _NAME_ dummy_id ln_odds_ratio );
 set def_collapse_transposed;

 * Formula for odds ratio ;
 ln_odds_ratio = log( (n_00 * n_11) / (n_01 * n_10) ) ;

 * ID var needed for tranposition later. ;
 dummy_id = 1 ;
run ;

* Transpose back into wide format for array calculations. ;
proc transpose data = def_collapse_transposed_odds out =
def_fay_brr_estimates;
 by dummy_id ;
 id _NAME_ ;
 var ln_odds_ratio ;
run;

* Calculate variance estimate. ;
data fay_estimate_final (keep = est var se odds_est lowerci upperci) ;
 set def_fay_brr_estimates (drop = _NAME_);
 * Filling in array with odds ratio estimates ;
 array repwts{80} rw1 - rw80 ;

 * Fill in other array with squared differences ;
 array sqdiff{80} sqdiff1 - sqdiff80 ;

 * Looping to create differences ;
 do i = 1 to 80;
  sqdiff[i] = (repwts[i] - est) ** 2 ;
 end;

 * Sum differences. ;
 total_diff = sum(of sqdiff1-sqdiff80) ;

 * Calculate variance. ;
```

Figure 9.3.3: SAS Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
 var = (4/80)* total_diff ;

 * Calculate standard error. ;
 se = var ** (0.5) ;

 * Create CI in odds ratio form. ;
 odds_est = exp(est);
 lowerci = exp((est - (1.96 * se))) ;
 upperci = exp((est + (1.96 * se))) ;
run ;

* Print final log odds ratio estimate, the variance, SE and
 then the odds ratio estimate with it's 95% CI ;
proc print data = fay_estimate_final noobs ;
 format est se lowerci upperci odds_est 8.4;
run;
```

The SAS code of Figure 9.3.3 generates the output of Figure 9.3.4. These are slightly different than PROC SURVEYFREQ due to rounding.

Figure 9.3.4: SAS Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

est	var	se	odds_est	lowerci	upperci
0.8141	0.026977	0.1642	2.2572	1.6359	3.1144

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Example 9.4. Estimating a Confidence Interval of an Odds Ratio with Replicate Weights and STATA

This example estimates the odds ratio and variance of rent stabilization status (pre-1947 or post-1947) and the presence of three or more maintenance deficiencies. The code in Figure 9.4.1 identifies the domains of interest, as well as the odds ratio and variance calculations using replicate weights in STATA.

Figure 9.4.1: STATA Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
use "hu21", clear
svyset[pweight=fw], vce(sdr) sdrweight(fw1-fw80) fay(.5)mse

* Binary variable for three or more deficiencies:
gen defect = (mdefcount >= 3)

* Need to exclude non-reported defects:
gen occ_reported_defects = (occ == 1 & mdefcount != -1 & csr == 32)

* Need binaries for time demarcation:
gen time_stable = 1 if yearbuilt < 5
replace time_stable = 0 if yearbuilt >= 5

* Now running LOG odds ratio:
svy, subpop(occ_reported_defects): logit defect time_stable

* Now for the odds ratio:
svy, subpop(occ_reported_defects): logistic defect time_stable
```

The STATA code of Figure 8.4.1 generates the output of Figure 9.4.2.

Figure 9.4.2: STATA Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
Log Odds Ratio:
Survey: Logistic regression                 Number of obs    =     8,394
                                            Population size  = 3,644,065
                                            Subpop. no. obs  =     1,701
                                              Subpop. size   = 838,760.67
                                              Replications   =        80
                                              Wald chi2(1)   =     24.57
                                              Prob > chi2    =    0.0000

------------------------------------------------------------------------------
             |                SDR *
      defect |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
 time stable |   .8141211   .1642473     4.96   0.000     .4922024    1.13604
        cons |  -1.716918   .1328527   -12.92   0.000    -1.977305   -1.456532
------------------------------------------------------------------------------

Odds Ratio:

Survey: Logistic regression                 Number of obs    =     8,394
                                            Population size  = 3,644,065
                                            Subpop. no. obs  =     1,701
                                              Subpop. size   = 838,760.67
                                              Replications   =        80
                                              Wald chi2(1)   =     24.57
                                              Prob > chi2    =    0.0000

------------------------------------------------------------------------------
             |                   SDR *
      defect | Odds Ratio   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
 time stable |   2.257191   .3707374     4.96   0.000     1.635915    3.11441
        cons |   .1796189   .0238628   -12.92   0.000     .1384419    .2330432
------------------------------------------------------------------------------
Note: _cons estimates baseline odds.
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Figure 8.4.2 shows that the odds ratio estimate from STATA is $\hat{\theta} = 2.2572$ which implies that housing units rent stabilized prior to 1947 are about twice as likely to have three or more maintenance deficiencies than HUs rent stabilized post-1947. The standard error of the log odds is 0.1642, which is the same as with the SAS method.

Example 9.5. Estimating a Confidence Interval of an Odds Ratio with Replicate Weights and R

This example estimates the odds ratio and variance calculation of rent stabilization status (pre-1947 or post-1947) and the presence of three or more maintenance deficiencies. The code in Figure 8.5.1 identifies the domains of interest, and the odds ratio and variance estimate using replicate weights in R.

Figure 9.5.1: R Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
odds_data <- hu21 %>% filter(OCC == 1 & MDEFCOUNT != -1 & CSR == 32)
n_11 <- sum(odds_data$FW[odds_data$defect == 1 & odds_data$time_stable ==
1])
n_00 <- sum(odds_data$FW[odds_data$defect == 0 & odds_data$time_stable ==
0])
n_01 <- sum(odds_data$FW[odds_data$defect == 0 & odds_data$time_stable ==
1])
n_10 <- sum(odds_data$FW[odds_data$defect == 1 & odds_data$time_stable ==
0])
point_est <- (n_11*n_00)/(n_01*n_10)

reps <- rep(0,80)
for (i in 1:80){
 n_11 <- sum(odds_data[,paste0("FW", as.character(i))][odds_data$defect ==
1 & odds_data$time_stable == 1])
 n_00 <- sum(odds_data[,paste0("FW", as.character(i))][odds_data$defect ==
0 & odds_data$time_stable == 0])
 n_01 <- sum(odds_data[,paste0("FW", as.character(i))][odds_data$defect ==
0 & odds_data$time_stable == 1])
 n_10 <- sum(odds_data[,paste0("FW", as.character(i))][odds_data$defect ==
1 & odds_data$time_stable == 0])
 reps[i] <- log((n_11*n_00)/(n_01*n_10))
}

reps <- (reps - log(point_est))^2
var_est = ((4/80)*sum(reps))
se_est = var_est^(1/2)

# 95% Confidence Intervals for Odds Ratios:
z_value <- qnorm(.975)
ci_95 <- exp(c(log(point_est) - z_value*se_est, log(point_est) +
z_value*se_est))

print(paste0(as.character(round(point_est, 4)),
       ", ", as.character(round(var_est, 4)),
       ", ", as.character(round(se_est, 4))))
```

The R code of Figure 9.5.1 generates the output of Figure 9.5.2 with some supplemental labels added in.

Figure 9.5.2: R Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

point_est	log var	log se
2.2572	0.027	0.1642

```
print(ci_95)
```

1.635915 3.114411 **<- 95% CI for Odds Ratios**

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From the output in Figure 9.5.2, we can see that the odds ratio estimate is $\hat{\theta} = 2.2572$, the standard error of the log odds is 0.1625, and the confidence interval of the odds ratio is (1.6359, 3.1144).

Example 9.6. Estimating a Confidence Interval of an Odds Ratio with Replicate Weights and Python

This example estimates the odds ratio and variance calculation of rent stabilization status (pre-1947 or post-1947) and the presence of three or more maintenance deficiencies. The code in Figure 9.6.1 first imports statistical function `scipy.stats`, then identifies the domains of interest, and estimates the odds ratio and variance using replicate weights in Python.

Figure 9.6.1: Python Code for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
import scipy.stats as sps
odds_data = hu21.loc[(hu21.OCC == 1) & (hu21.MDEFCOUNT != -1) & (hu21.CSR ==
32)]
n_11 = odds_data.loc[(odds_data.defect == 1) & (odds_data.time_stable ==
1)].FW.sum()
n_00 = odds_data.loc[(odds_data.defect == 0) & (odds_data.time_stable ==
0)].FW.sum()
n_01 = odds_data.loc[(odds_data.defect == 0) & (odds_data.time_stable ==
1)].FW.sum()
n_10 = odds_data.loc[(odds_data.defect == 1) & (odds_data.time_stable ==
0)].FW.sum()
point_est = (n_11*n_00)/(n_01*n_10)
ln_point_est = np.log(point_est)

reps = np.array([0.0]*80)
for i in range(1,81):
  n_11 = odds_data.loc[(odds_data.defect == 1) & (odds_data.time_stable ==
1)]["FW"+str(i)].sum()
  n_00 = odds_data.loc[(odds_data.defect == 0) & (odds_data.time_stable ==
0)]["FW"+str(i)].sum()
  n_01 = odds_data.loc[(odds_data.defect == 0) & (odds_data.time_stable ==
1)]["FW"+str(i)].sum()
  n_10 = odds_data.loc[(odds_data.defect == 1) & (odds_data.time_stable ==
0)]["FW"+str(i)].sum()
  reps[i-1] = np.log((n_11*n_00)/(n_01*n_10))

reps = (reps - ln_point_est)**2
se = ((4/80)*reps.sum())**(1/2)
print(point_est.round(decimals = 4), np.exp(se).round(decimals = 4))

# 95% Confidence Intervals for Odds Ratios:
z_value = sps.norm.ppf(0.975)
ci = [np.exp(ln_point_est - z_value*se), np.exp(ln_point_est + z_value*se)]
print(ci)
```

The Python code of Figure 9.6.1 generates the output of Figure 9.6.2 with some supplemental labels added in.

Figure 9.6.2: Python Output for Calculating Confidence Interval of Odds Ratio with Replicate Weights

```
2.2572  <- Odds Ratio Point Estimate

0.027  <- Log Odds Variance

0.1642  <- Log Odds SE

[1.6359149683461613, 3.1144105055894173]  <- 95% CI for Odds Ratios
```

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

From the output in Figure 9.6.2, we can see that the odds ratio estimate is $\hat{\theta} = 2.2572$, the standard error of the log odds is 0.1642, and the confidence interval of the odds ratio is (1.6359, 3.1144).

Example 9.7. Estimating a Confidence Interval of an Odds Ratio with Final weights and SAS PROC SURVEYFREQ

If you merely use the final weights with the PROC SURVEYFREQ command, omitting the replicate weights and the `varmethod` option, then PROC SURVEYFREQ defaults to the Taylor Linearization Method for estimating the variance of the odds ratio. You will get the same point estimate and estimated counts as Example 9.1, but the CI will be slightly narrower.

Figure 9.7.1 provides the PROC SURVEYFREQ code that will calculate the odds ratio using just the final weights.

Figure 9.7.1: SAS Code for Calculating Confidence Interval of Odds Ratio with Final weights

```
proc surveyfreq data=def_analysis ;
 table def * stabilized / or ;
 weight FW ;
run ;
```

The code of Figure 9.7.1 generates the output of Figure 9.7.2.

Figure 9.7.2: SAS Output for Calculating Confidence Interval of Odds Ratio with Final weights

The SURVEYFREQ Procedure

Data Summary

Number of Observations	1701
Sum of Weights	838760.666

Figure 9.7.2: SAS Output for Calculating Confidence Interval of Odds Ratio with Final weights

Table of def by stabilized

def	stabilized	Frequency	Weighted Frequency	Std Err of Wgt Freq	Percent	Std Err of Percent
0	0	531	231825	10154	27.6390	1.1987
	1	762	402221	13263	47.9542	1.3804
	Total	1293	634046	13012	75.5932	1.1793
1	0	95	41640	4677	4.9645	0.5584
	1	313	163074	9508	19.4423	1.0993
	Total	408	204714	10212	24.4068	1.1793
Total	0	626	273465	10659	32.6035	1.2599
	1	1075	565296	13753	67.3965	1.2599
	Total	1701	838761	10995	100.0000	

Odds Ratio and Relative Risks (Row1/Row2)

Statistic	Estimate	95% Confidence Limits	
Odds Ratio	2.2572	1.6945	3.0068
Column 1 Relative Risk	1.7975	1.4424	2.2401
Column 2 Relative Risk	0.7964	0.7426	0.8540

Sample Size = 1701

Source: U.S. Census Bureau, 2021, New York City Housing and Vacancy Survey.

Example 9.8. Estimating a Confidence Interval of an Odds Ratio with Normalized Weights and SAS PROC FREQ

In this estimation of the CI, we estimate the variance with the normalized weights, which reduces the underestimation of the variance that we will see in estimating a CI with final weights in Example 9.9. Normalized weights are not provided by NYCHVS but can easily be calculated as the final weight divided by the mean of all of the final weights with a positive value.

Normalizing the final weight produces a weight that accounts for the unequal weights that reflect that sample units contribute unequally to the estimates. The normalized weights have a mean value of 1.0. So the normalized weights have the same magnitude as if you didn't use weights – all sample units have a weight of 1.0 – and the variance (Example 9.8) produces a more reasonable estimate of the variance as compared to Example 9.9.

Table 9.8.1 provides the weighted estimates of the frequencies of each cell. The final weight and Equation (9.2) were used to produce the totals.

Table 9.8.1: Estimated Counts for Maintenance Deficiencies Example

	Fewer than 3 maintenance deficiencies	3 or more maintenance deficiencies	Total
Rent Stabilized Post 1947	n_{00} = 470.139	n_{01} = 84.446	n_{0+} = 554.585
Rent Stabilized Pre-1947	n_{10} = 815.701	n_{11} = 330.713	n_{1+} = 1146.41
Total	n_{+0} = 1285.84	n_{+1} = 415.159	n = 1701.00

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Using the weighted totals, the odds of having three or more deficiencies given the HU is rent stabilized pre-1947 is calculated as

$$\{\text{Odds of Y given Rent Stabilized Pre} - 1947\ (X = 1)\} = \frac{330.713}{84.446} = 3.9163$$

and we calculate the odds ratio as

$$\hat{\theta} = \frac{330.713 * 470.139}{815.701 * 84.4458} = 2.2572$$

Figure 9.8.1 provides the PROC FREQ code that will calculate the odds ratio using the normalized weights.

Figure 9.8.1: SAS Code for Calculating Confidence Interval of Odds Ratio with Normalized Weights

```
data def_analysis ;
 set def_analysis ;
 * This is the formula for normalizing weights
  Number of observations in sample (n) divided by sum of weights
  (i.e. the mean of the weights) ;
 fw_normalized = FW * (1701/838761) ;
run ;

proc freq data = def_analysis ;
 table def * stabilized / or nocol norow ;
 weight fw_normalized ;
run ;
```

The SAS code of Figure 9.8.1 generates the output of Figure 9.8.2.

Figure 9.8.2: SAS Output for Calculating Confidence Interval of Odds Ratio with Normalized Weights

The FREQ Procedure

Table of def by stabilized

Frequency / Percent	def	stabilized 0	stabilized 1	Total
	0	470.139	815.701	1285.84
		27.64	47.95	75.59
	1	84.4458	330.713	415.159
		4.96	19.44	24.41
	Total	554.585	1146.41	1701
		32.60	67.40	100.00

Statistics for Table of def by stabilized

Odds Ratio and Relative Risks

Statistic	Value	95% Confidence Limits	
Odds Ratio	2.2572	1.7325	2.9408
Relative Risk (Column 1)	1.7975	1.4665	2.2032
Relative Risk (Column 2)	0.7964	0.7471	0.8489

Sample Size = 1700.9993226

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Example 9.9. Estimating a Confidence Interval of an Odds Ratio with Final weights and SAS PROC FREQ

If we calculate the variance with the usual estimator of the variance of a log odds, Equation (9.3), and insert the weighted estimates from Table 9.3.1, we get

$$\hat{v}(\log odds) = \frac{1}{231{,}825} + \frac{1}{402{,}221} + \frac{1}{41{,}640} + \frac{1}{163{,}074} = 0.000037$$

and a standard error of $\widehat{se}(\log odds) = 0.0061$.

This answer is about two orders of magnitude smaller than the weighted estimate using the replicate weights and is misleading because it results in much smaller CIs.

Figure 9.9.1 provides the PROC FREQ code that will calculate the odds ratio using the replicate weights.

Figure 9.9.1: SAS Code for Calculating Confidence Interval of Odds Ratio with Final weights

```
proc freq data = def_analysis ;
 table def * stabilized / or nocol norow;
 weight FW ;
run ;
```

The SAS code of Figure 9.9.1 generates the output of Figure 9.9.2.

Figure 9.9.2: SAS Output for Calculating Confidence Interval of Odds Ratio with Final weights

The FREQ Procedure

Table of def by stabilized

Frequency / Percent	def	stabilized 0	stabilized 1	Total
	0	231825	402221	634046
		27.64	47.95	75.59
	1	41640.1	163074	204714
		4.96	19.44	24.41
	Total	273465	565296	838761
		32.60	67.40	100.00

Statistics for Table of def by stabilized

Odds Ratio and Relative Risks

Statistic	Value	95% Confidence Limits	
Odds Ratio	2.2572	2.2305	2.2842
Relative Risk (Column 1)	1.7975	1.7811	1.8141
Relative Risk (Column 2)	0.7964	0.7941	0.7987

Sample Size = 838760.66597

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

Example 9.10. Estimating a Confidence Interval of an Odds Ratio with No Weights and SAS PROC FREQ

The unweighted counts for the maintenance deficiencies example are provided in Table 9.10.1.

Table 9.10.1: Unweighted Counts for Maintenance Deficiencies Example

	Fewer than 3 maintenance deficiencies	3 or more maintenance deficiencies	Total
Rent Stabilized Post-1947	n_{00} = 531	n_{01} = 95	n_{0+} = 626
Rent Stabilized Pre-1947	n_{10} = 762	n_{11} = 313	n_{1+} = 1075
Total	n_{+0} = 1293	n_{+1} = 408	n = 1,701

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

If we completely ignore the sample weights, the odds of having three or more maintenance deficiencies given the HU is rent stabilized pre-1947 is calculated as

$$\{\text{Odds of Y given Rent Stabilized Pre} - 1947\ (X = 1)\} = \frac{313}{95} = 3.2947$$

and we estimate the odds ratio as

$$\hat{\theta} = \frac{313 * 531}{762 * 95} = 2.2959$$

The log odds, or the natural log of the odds ratio $\hat{\theta}$, is log odds$= \ln(\hat{\theta}) = 0.8311$. The unweighted variance of the log odds can be calculated using Equation (8.3) and Table 9.9.1 as

$$\hat{v}(\log odds) = \frac{1}{531} + \frac{1}{762} + \frac{1}{95} + \frac{1}{313} = 0.0169$$

and the standard error is $\widehat{se}(\log odds) = 0.1301$. With the estimate of $\hat{\theta}$ and the standard error, we can use Equation (7.4) to calculate the CI of (1.7793, 2.9626)

The SAS code of Figure 9.10.1 shows how the CIs for the odds ratio can be calculated directly.

Figure 9.10.1: SAS Code for Calculating Confidence Interval of Odds Ratio without Weights

```
proc freq data=def_analysis ;
  table def * stabilized / or nocol norow ;
run ;
```

The SAS code of Figure 9.10.1 generates the output of Figure 9.10.2.

Figure 9.10.2: SAS Output for Calculating Confidence Interval of Odds Ratio without Weights

The FREQ Procedure

Table of def by stabilized

Frequency Percent	def	Stabilized		
		0	1	Total
	0	531	762	1293
		31.22	44.80	76.01
	1	95	313	408
		5.58	18.40	23.99
	Total	626	1075	1701
		36.80	63.20	100.00

Statistics for Table of def by stabilized

Odds Ratio and Relative Risks

Statistic	Value	95% Confidence Limits	
Odds Ratio	2.2959	1.7793	2.9626
Relative Risk (Column 1)	1.7637	1.4617	2.1282
Relative Risk (Column 2)	0.7682	0.7161	0.8241

Sample Size = 1701

Source: U.S. Census Bureau, 2021 New York City Housing and Vacancy Survey.

10. How the Replicate Weights Are Calculated

As with most large-scale household surveys, in an effort to control costs, the NYCHVS uses a complex sample design involving multi-stage sampling and unequal sampling rates. Weights are needed in the analysis to compensate for unequal sampling rates as well as for nonresponse. Further, most estimates from complex samples are non-linear statistics, so estimates of the standard errors are often obtained using the first-order Taylor series approximations or replication methods such as balanced repeated replication or jackknife replication. Therefore, the complex sample design needs to be considered in estimating the precision of survey estimates. Not accounting for these sample design features will lead to inaccurate point estimates and underestimated variance. This section describes methods used for generating accurate variance estimates.

Variance Estimates with Replication

Replication methods provide estimates of variance for a wide variety of designs using probability sampling, even when complex estimation procedures are used. This method requires that the sample selection, the collection of data, and the estimation procedures be carried out (replicated) several times. The dispersion of the resulting estimates can be used to measure the variance of the sample.

In the variance estimation of NYCHVS, we use SDR. This technique is embodied by the replicate factors that are produced for the NYCHVS replicate variance estimator.

Replicate Weights

The base weights are multiplied by the replicate factors to produce the replicate weights. The replicate weights are further adjusted with the same weighting adjustments as applied to the final weight including a noninterview and ratio adjustments. For more details on the sample design of NYCHVS, please refer to the 2021 NYCHVS Sample Design, Weighting, and Error Estimation document (U.S. Census Bureau, 2023). By applying the other weighting adjustments to each replicate, the final replicate weights reflect the impact of the weighting adjustments on the variance.

Replicate Factors

The theoretical basis for the successive difference method was discussed by Wolter (2007) and extended by Fay and Train (1995) to produce the SDR method. See also Ash (2014) and Opsomer, Breidt, White, and Li (2016). The following is a description of SDR:

To apply SDR to the sample, we sort the sample by borough (variable on the PUFs: *BORO*) and then within borough by the same order that was used to select the original *systematic* sample. Then each sample unit is assigned two rows of the given Hadamard matrix. For example, the

assignment for a Hadamard of order 80 would be rows (1,2) assigned to the first unit, rows (2,3) assigned to the second unit, … rows (80,1) assigned to the 80th unit. The assignment is repeated in further cycles until the entire sample is assigned two rows.

For a sample, two rows of the Hadamard matrix are assigned to each pair of units creating replicate factors, $f_{i,r}$ for r = 1,..., R as

$$f_{i,r} = 1 + 2^{-3/2} h_{i+1,r} - 2^{-3/2} h_{i+2,r}$$

where

i	the index on the units of the sample
r	the index on the set of replicates
$h_{i,r}$	number in the Hadamard matrix (+1 or −1) for the ith unit in the systematic sample
R	the number of total replicate samples or simply replicates

This formula yields replicate factors of approximately 1.7, 1.0, or 0.3.

Example 10.1. Successive Difference Replication

The following is a simple example showing the SDR method. The sample contains n = 4 units, and their weights are shown in Table 10.1.1.

Table 10.1.1: Sample Weights

Sample HU	Sample Weight
1	15.00
2	23.00
3	19.00
4	16.00

We choose to use the following 4 × 4 Hadamard matrix to define the replicate factors:

$$\mathbf{H}_4 = \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix}$$

Two consecutive rows of $\mathbf{H}_4$ are assigned to each sample unit as denoted in Table 10.1.2.

Table 10.1.2: Assignment of Rows of the Hadamard Matrix

Sample HU	Sample Weight	Row I	Row II
1	15.00	1	2
2	23.00	2	3
3	19.00	3	4
4	16.00	4	1

Plugging these values into our replicate factor formula of Equation (2.1) we get

Sample HU 1

$$f_{1,1} = 1 + 2^{-\frac{3}{2}} h_{1,1} - 2^{-\frac{3}{2}} h_{2,1} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(+1) = 1.0$$

$$f_{1,2} = 1 + 2^{-\frac{3}{2}} h_{1,2} - 2^{-\frac{3}{2}} h_{2,2} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(-1) = 1 + \frac{1}{\sqrt{2}} \cong 1.7$$

$$f_{1,3} = 1 + 2^{-\frac{3}{2}} h_{1,3} - 2^{-\frac{3}{2}} h_{2,3} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(+1) = 1.0$$

$$f_{1,4} = 1 + 2^{-\frac{3}{2}} h_{1,4} - 2^{-\frac{3}{2}} h_{2,4} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(-1) = 1 + \frac{1}{\sqrt{2}} \cong 1.7$$

Sample HU 2

$$f_{3,1} = 1 + 2^{-\frac{3}{2}} h_{2,1} - 2^{-\frac{3}{2}} h_{3,1} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(+1) = 1.0$$

$$f_{3,2} = 1 + 2^{-\frac{3}{2}} h_{2,2} - 2^{-\frac{3}{2}} h_{3,2} = 1 + 2^{-\frac{3}{2}}(-1) - 2^{-\frac{3}{2}}(+1) = 1 - \frac{1}{\sqrt{2}} \cong 0.3$$

$$f_{3,3} = 1 + 2^{-\frac{3}{2}} h_{2,3} - 2^{-\frac{3}{2}} h_{3,3} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(-1) = 1 + \frac{1}{\sqrt{2}} \cong 1.7$$

$$f_{3,4} = 1 + 2^{-\frac{3}{2}} h_{2,4} - 2^{-\frac{3}{2}} h_{3,4} = 1 + 2^{-\frac{3}{2}}(-1) - 2^{-\frac{3}{2}}(-1) = 1.0$$

Sample HU 3

$$f_{2,1} = 1 + 2^{-\frac{3}{2}} h_{3,1} - 2^{-\frac{3}{2}} h_{4,1} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(+1) = 1.0$$

$$f_{2,2} = 1 + 2^{-\frac{3}{2}} h_{3,2} - 2^{-\frac{3}{2}} h_{4,2} = 1 + 2^{-\frac{3}{2}}(+1) - 2^{-\frac{3}{2}}(-1) = 1 + \frac{1}{\sqrt{2}} \cong 1.7$$

$$f_{2,3} = 1 + 2^{-\frac{3}{2}} h_{3,3} - 2^{-\frac{3}{2}} h_{4,3} = 1 + 2^{-\frac{3}{2}}(-1) - 2^{-\frac{3}{2}}(-1) = 1.0$$

$$f_{2,4} = 1 + 2^{-\frac{3}{2}} h_{3,4} - 2^{-\frac{3}{2}} h_{4,4} = 1 + 2^{-\frac{3}{2}}(-1) - 2^{-\frac{3}{2}}(+1) = 1 - \frac{1}{\sqrt{2}} \cong 0.3$$

Repeat the process for Sample HU #4 using rows 4 and 1, respectively.

If we calculate the replicate factors for every replicate and unit in the sample, we get the values in Table 10.1.3.

Table 10.1.3: Replicate Factors

Sample HU	Replicate Factors			
	Replicate 1	Replicate 2	Replicate 3	Replicate 4
1	1.0	1.7	1.0	1.7
2	1.0	0.3	1.7	1.0
3	1.0	1.7	1.0	0.3
4	1.0	0.3	0.3	1.0

Next, multiply the sample and corresponding factors to get the replicate weights of Table 10.1.4.

Table 10.1.4: Final Replicate Weights

Sample HU	Full	Replicate Weights			
	Sample Weight	Replicate 1	Replicate 2	Replicate 3	Replicate 4
1	15.0	15.0	25.5	15.0	25.5
2	23.0	23.0	6.9	39.1	23.0
3	19.0	19.0	32.3	19	5.7
4	16.0	16.0	4.8	4.8	16.0

Other Weighting Adjustments for Replicate Weights

In Example 10.1, we end at the step of adjusting the replicate base weights for the different replicates. The next step is to calculate the rest of the weighting adjustments for each set of replicate weights. The replicate weights also account for the effect on the variance of the other weighting factors. Recalculating the noninterview and ratio adjustments for each replicate ensures that the randomness injected or mitigated by the different weighting adjustments is represented in each of the replicate estimates. See also Judkins (1990; p. 224) and Brick and Kalton (1996) for additional discussion of application of other weighting adjustments within replicate weighting.

The Factor of 4 in Equation (2.1)

The replication variance estimator of Equation (2.1) includes what looks like an odd term of 4. This 4 is required by SDR because as Ash (2014) explains, each of the four rows of the Hadamard matrix produce one estimator of the successive difference variance estimator. So with NYCHVS, we need to divide the usual replication estimator by 80/4 = 20 and not 80.

The easiest solution to adding the 4 to the PROC SURVEY procedures of SAS is to use the `varmethod=brr(fay)` option which has a default value of 0.5 for the perturbing factor. SAS will then use the 4 in the calculation of all replicates, and variance estimates will be correct.

The alternative is to use the PROC SURVEY procedures of SAS with the `varmethod=brr` option and multiply the standard error by 2 or multiply the variance by 4.

11. References

Ash, S. (2014). "Using Successive Difference Replication for Estimating Variances," Survey Methodology, 40, 1, 47-59.

Brick, J.M. and Kalton, G. (1996). "Handling Missing Data in Survey Research," Statistical Methods in Medical Research, 5, 215-238.

Cochran, W.G. (1977). *Sampling Techniques*, 3rd Edition, John Wiley & Sons, New York.

Eltinge, J.L. and Jang, D.S. (1996). "Stability Measures for Variance Commponent Estimators under a Stratified Multistage Design". Survey Methodology, 22, 157-165.

Fay, R.E. and Train, G.F. (1995). "Aspects of Survey and Model-Based Postcensal Estimation of Income and Poverty Characteristics for States and Counties," Joint Statistical Meetings, Proceedings of the Section on Government Statistics, 154-159.

Heeringa, S.G., West, B. T., and Berglund, P.A. (2010). *Applied Survey Data Analysis*, CRC Press.

Huang, E.T. and Bell, W.R., (2011). "A Simulation Study of the Distribution of Fay's Successive Difference Replication Variance Estimator", retrieved February 11, 2019 from https://www.census.gov/content/dam/Census/library/working-papers/2011/demo/huang-bell2011.pdf

Judkins, D.R. (1990). "Fay's Method for Variance Estimation," Journal of Official Statistics, 6, 3, 223-239.

Kolenikov, S. (2019). Programs and Tutorials Written by Stas Kolenikov, retrieved March 3, 2019 from http://staskolenikov.net/STATA

Korn, E.L. and Graubard, B.I. (1999). *Analysis of Health Surveys*, John Wiley & Sons.

Kreuter, F. and Valliant, R. (2007). "A survey on survey statistics: What is done and can be done in STATA." *The STATA Journal*, 7(1), 1 – 21. Retrieved July 18, 2011 from http://www.STATA-journal.com/article.html?article=st0118

Krewski, D. and Rao, J.N.K. (1981). *Inference from Stratified Samples: Properties of the Linerization, Jackknife and Balanced Repeated Replication Method*, Annals of Statistics, 9, 5, 1010-1019.

Lewis, T. H. (2017). "Complex Survey Data Analysis With SAS." CPC Press, Boca Raton, FL. Section 8.

Lohr, S. (1999). *Sampling Design and Analysis*, Brooks/Cole Publishing Company.

Mukhopadhyay, P.K., An, A.B., Tobias, R.D., and Watts, D.L. (2008). "Try, Try Again: Replication-Based Variance Estimation Methods for Survey Data Analysis in SAS® 9.2," Proceedings of the 2008 NESUG.

Opsomer, J.D., Breidt, F.J., White, M., and Li, Y. (2016). "Successive Difference Replication for Variance Estimation in Two-Phase Sampling," Journal of Survey Statistics and Methodology, 4, 43-70.

SAS 9.2 User's Guide (2019), Second Edition, SAS Institute Inc. retrieved from https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_surveyfreq_a0000000220.htm

Särndal, C.-E., Swensson, B. and Wretman, J. (1992). *Model Assisted Survey Sampling*, Springer-Verlag.

Schenker, N. and Gentleman, J.F. (2001). "On Judging the Significances of Differences by Examining the Overlap Between Confidence Intervals," The American Statistician, 55:3. August 2001.

Taylor, L.H. (2016). *Complex Survey Data Analysis with SAS*, CRC Press.

U.S. Census Bureau. (2014). American Community Survey Design and Methodology (January 2014).

U.S. Census Bureau. (2018). 2017 New York City Housing and Vacancy Survey Sample Design, Weighting, and Error Estimation.

U.S. Census Bureau. (2019). 2017 New York City Housing and Vacancy Survey The Guide to Estimating Variances.

U.S. Census Bureau. (2023). 2021 New York City Housing and Vacancy Survey Sample Design, Weighting, and Error Estimation.

Valliant, R., Rust, K. (2010). "Degree of Freedom Approximations and Rules-of-Thumb," Journal of Official Statistics, Vol. 26, No. 4, 2010, pp. 585-602.

Wolter, K.M. (2007). *Introduction to Variance Estimation*, Springer-Verlag, 2nd Edition.

GLOSSARY OF TERMS

This glossary provides the definitions of several terms used with the technical document. It includes both terms related to statistics and the sample design.

Balanced Repeated Replication (BRR) – A method of variance estimation that is often used with two-stage sample designs that select one or two PSUs per first-stage strata. The method is valuable because it can be applied to estimating the variance of linear and non-linear estimates. Also, the intermediate replicate weights can be provided to data users so that they can estimate variances themselves using simple expressions for the variance. The main ideas of replication are outlined by McCarthy (1966).

Bias – The formal definition of bias of an estimator $\hat{\theta}$ of some statistic θ is the expected value of the absolute value of the difference between the estimator and statistic and its expected value, i.e., $B(\hat{\theta}) = E|\hat{\theta} - \theta|$. Informally, the bias is a measure of how close the estimator is to the value it is estimating.

Borough – A borough is determined by borough code variable (BORO) on the NYCHVS data file(s). The code BORO has the values: 1 = Bronx; 2 = Brooklyn; 3 = Manhattan; 4 = Queens; 5 = Staten Island.

Calibration – As described by Deville and Särndal (1992), calibration is a technique that can be used to reduce the variance of an estimator. Sometimes it can also have the effect of improving the coverage of the estimator. Calibration uses a set of known totals: either an "imported totals" (Särndal and Lundström, 1999) or a set of variables that are known for all units in the universe. Calibration finds weights that are "close" to the original design weights so that the estimated known totals with the new weights are the same as the known total.

Coefficient of Variation (CV) – The square root of the variance of an estimate divided by the estimate, i.e., $\sqrt{v(\hat{\theta})}/\hat{\theta}$.

Coverage – A measure of how well a frame and sample design includes the universe of interest. Usually coverage is expressed as a proportion. For example, if a study has 75% coverage then 75% of the universe of interest was included by the frame and the sample design.

Domain of Interest or Domain – A specific subset of the universe.

Eligible / Ineligible – Refers to the whether a unit of interest is in the universe of interest or not in the universe of interest. See also AAPOR (2011).

Final Weight – The sample weight that can be used with the sample to make estimates of variables collected with the sample. The name "final weight" is used in some sections of the guide to distinguish the final weight from the replicate weights. The final weight is also the same as the weight for Replicate 0.

Frame – The list of units in the universe of interest.

Generalized Variance Function (GVF) – As explained by Wolter (2007), the GVF is a simple model that expresses the variance as a function of the expected value of the survey estimate.

Housing Unit – A housing unit is an apartment, a house, a group of rooms, or a single room occupied or intended for occupancy as separate living quarters. Housing units must meet both of the following qualifications: 1) separateness, meaning occupants live separately from any other occupants in the building and 2) direct access, meaning that the entrance to the living quarters must be directly from the outside of the building or through a common hall.

Similar to prior cycles, group quarters were not considered housing units and are therefore excluded from the NYCHVS. Examples include nursing homes, prisons, rectories and dormitories for students or workers. Any persons residing in such places are also not included in the survey.

Interviews – Interviews consists of units that are either:

- Occupied - Outcome Code (OCC) has a value of 1; or
- Vacant - Outcome Code (OCC) has a value of 2, 3, or 4.

Non-interview – Units are classified as non-interviews when we expect to get interviews but don't for one of the following reasons:

1. In-scope (Type A) - refused, no one home, temporarily absent, other.

2. Temporarily Out-of-scope (Type B) - permit granted construction not started, unfit to be demolished, under construction, converted to temporary nonresidential use, other

3. Out-of-scope (Type C) - demolished, condemned, nonresidential, merged, damaged by fire, boarded up, list procedure applied, no such address, other.

Please refer to the 2021 NYCHVS Sample Design, Weighting, and Error Estimation document for more detailed descriptions of *Type A Noninterviews* and *Type C Noninterviews.*

Occupied Units – Sample units are classified as occupied units if they have people living in them at the time of observation, regardless of their condition. Units that are occupied have a value of 1 for OCC on the NYCHVS questionnaire. They are units that were interviewed.

Owner Occupied – An occupied housing unit is owner occupied if the owner or co-owner lives in the unit, even if it's not mortgaged or not fully paid for. See *Owner Occupied Units* for values of units that are owner occupied.

Owner Occupied Units – Units having a value of 1 for OCC and a value of 2 in Tenure on the NYCHVS questionnaire.

Relative Variance, Relvariance, or Relvar – is a measure of the relative dispersion of a probability distribution and is defined as the variance divided by the square of the estimate. It is also equal to the square of the coefficient of variation, i.e., $relvar(\hat{\theta}) = v(\hat{\theta})/\hat{\theta}^2$

Renter Occupied – An occupied housing unit is renter occupied if the unit is rented for cash rent or occupied without payment of cash rent. See *Renter-Occupied Units* for values of renter-occupied units.

Renter-Occupied Units – Units having a value of 1 for OCC and a value of 1 Tenure on the NYCHVS questionnaire.

Replicate Sample or Replicate – In replication variance estimation, we make several replicate samples where each has a different set of weights. Each of the replicate samples and the corresponding replicate weights used to make a replicate estimate that is used in variance estimation.

Replicate Weight – The replicate weight that can be used with the sample to make estimates of variables collected with the sample.

Replicate 0 – Refers to the replicate that uses the final weight.

Sample Design – Everything about the selection of units into the sample that determines the probability of selection for each unit. We think of estimation as separate from sample design, in that some estimation procedures are more appropriate than others for a given sample design, but any estimator could be used with the sample derived from a given sample design.

Sampling Fraction – The fraction of the universe that is in the sample. With an equal probability sample design, the sampling fraction is the ratio of the sample size and the size of the universe, often represented as $f = n/N$.

Sampling Interval – The inverse of the sampling fraction. Sometimes referred to as the "take-every" because we take every f^{-1} units of the universe into the sample.

Standard Error – Is the square root of the variance, i.e., $se(\hat{\theta}) = \sqrt{v(\hat{\theta})}$.

Successive Difference Replication (SDR) – A replication variance estimation method that mimics the successive difference variance estimator and can be used to estimate the variance from a *sys* sample design. The main ideas of replication are outline by Fay and Train (1995).

Systematic Random Sampling or *sys* – A random sampling method that requires selecting samples based on a system of intervals in an ordered population.

Unit – The following definition is from Hájek (1981, p. 4):

"The units making up the population S may be any elements worth studying – persons, families, farms, account items, temperature readings, and so on – and their nature will be irrelevant for theoretical considerations. We shall assume that the units are identifiable by certain labels (tags, names, addresses) and that we have available a frame (list, map) showing how to reach any unit given its label."

For NYCHVS, unit of interest is the housing unit.

Universe of interest – In finite population sampling, the universe of interest, or simply the universe, is the well-defined set of units for which we would like to produce an estimate.

Vacant Housing Units – Vacant housing units include vacant for sale, vacant for rent, and vacant not for sale or rent units.

1) Vacant, Available for Sale: Vacant housing units that have a value of 3 for Outcome Code (OCC).

2) Vacant, Available for Rent or Sale: Vacant housing units that have a value of 2 for Outcome Code (OCC).

3) Vacant, not Available for Rent or Sale: Vacant housing units that have a value of 4 for Outcome Code (OCC).

Variance or Sample Variance – Is a measure of the variability of an estimate. With finite population sampling, variance refers to the measure of how the estimate differs if we were to select other samples. Formally, the variance of an estimator $\hat{\theta}$ is the expected value of the squared difference between the estimator $\hat{\theta}$ and its expected value, i.e., $v(\hat{\theta}) = E\left(\hat{\theta} - E(\hat{\theta})\right)^2$.

References

The American Association for Public Opinion Research. 2011. *Standard Definitions: Final Dispositions of Case Codes and Outcome Rates for Surveys, 7th edition,* AAPOR.

Deville, J.C. and Särndal, C.E. (1992). "Calibration Estimators in Surveys," Journal of the American Statistical Association, 87, 418.

Fay, R.E. and Train, G.F. (1995). "Aspects of Survey and Model-Based Postcensal Estimation of Income and Poverty Characteristics for States and Counties," Joint Statistical Meetings, Proceedings of the Section on Government Statistics, 154-159.

Hájek, J. (1981). *Sampling From a Finite Population*, Marcel Dekker Inc.

McCarthy, P.J. (1966). "Pseudo-replication: half-samples," Review of the International Statistical Institute, 37, 239-264.

Särndal, C. E. and Lundström, S. (1999). "Calibration as a Standard Method for Treatment of Nonresponse", Journal of Official Statistics. 305-327.

U.S. Census Bureau. (2018). 2017 New York City Housing and Vacancy Survey Sample Design, Weighting, and Error Estimation.

Wolter, K.M. (2007). *Introduction to Variance Estimation*, Springer-Verlag, 2nd Edition.

LONGITUDINAL EXAMPLES FOR SAS, STATA, R, PYTHON

To assist data users with the longitudinal examples, this appendix uses 2021 and 2023 data to present example code for estimating variance using replicate weights for SAS, STATA, R, and Python. The examples presented cover longitudinal change (Examples 4.6, 5.6, 6.6, and 7.6) and percent change (Examples 4.7, 5.7, 6.7, and 7.7).

SAS

Example 4.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 4.7 will consider an estimate of percent of change that is calculated at the HU level.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we are interested in the estimated difference in gross vacancy rates across two survey cycles. We write this as $\hat{p}_t = \hat{X}_t / \hat{N}_t * 100$, where $\hat{X}_t$ is the estimator of the total number of vacant HUs at time *t*, and $\hat{N}_t$ is the estimator of the total number of HUs at time *t*.

We'll measure the difference in the gross vacancy rate between 2023 and 2021 in this example, so the statistic of interest is:

$$\hat{\Delta}_{t=2021} = \frac{\hat{X}_{t=2023}}{\hat{N}_{t=2023}} * 100 - \frac{\hat{X}_{t=2021}}{\hat{N}_{t=2021}} * 100$$

To estimate the variance *of* $\hat{\Delta}_t$ *with replicate weights*, we first calculate all the pieces of $\hat{\Delta}_{t=2023}$ for each of the 80 replicates, making 80 replicate estimates of $\hat{\Delta}_{t=2023}$, repeat those steps for 2021, and then apply Equation (2.1). Figure 4.6.1 shows how to do this with SAS.

Figure 4.6.1: SAS Code for Estimating Variance of a Longitudinal Change

```
*Define Vacant and Occupied HUs. ;
data temp1 ;
 set HU23 ;
 if Occ=1 then type='1'; * Occupied HUs;
 else if Occ in (2,3,4)  then type='2'; * X-hat 2023* vacant HUs ;
 * Type 1 & 2 makes total number HUs, N-hat ;
run;

data temp2 ;
 set HU21 ;
 if Occ=1 then type='1'; * Occupied HUs;
 else if Occ in (2,3,4)  then type='2'; * X-hat 2021* vacant HUs ;
 * Type 1 & 2 makes total number HUs, N-hat ;
```

Figure 4.6.1: SAS Code for Estimating Variance of a Longitudinal Change

```
run;

* This empty data set is produced for the merge later. ;
data data21 ;
 length diff0-diff80 8. ;
run ;

* Estimate the replicate estimates for replicates 0-80. ;
%macro repssss(rep) ;
* Get denominator N-hat 2023. ;
proc means data= temp1 noprint ;
 where type in ('2','1') ;
 var fw&rep.;
 output out=den23rep&rep. sum=den23rep&rep. ;
run ;

* Get numerator X-hat 2023. ;
proc means data= temp1 noprint ;
 where type in ('2') ;
 var fw&rep.;
output out=num23rep&rep. sum=num23rep&rep. ;
run ;
* Get denominator N-hat 2021. ;
proc means data= temp2 noprint ;
 where type in ('2','1') ;
  var fw&rep.;
 output out=den21rep&rep. sum=den21rep&rep. ;
run ;

* Get numerator X-hat 2021. ;
proc means data= temp2 noprint ;
 where type in ('2') ;
 var fw&rep.;
 output out=num21rep&rep. sum=num21rep&rep.;
run ;

* Merge the replicate estimates of each year, and get difference
 in vacancy rate by replicate. ;
data datamrep&rep.;
 merge num21rep&rep. num23rep&rep. den21rep&rep. den23rep&rep. ;
 diff&rep. = (num23rep&rep./den23rep&rep.)*100 -
(num21rep&rep./den21rep&rep.)*100 ;
 keep diff&rep.;
run ;

data data21 ;
 merge data21 datamrep&rep. ;
run ;
%mend repssss ;

%macro doit ;
 %do i=0 %to 80 ;
  %repssss(&i.) ;
  %end;
%mend doit ;
```

Figure 4.6.1: SAS Code for Estimating Variance of a Longitudinal Change

```
%doit ;

* Apply Step 3 to the replicate estimates and estimate the
variance. ;
data data22 (keep=diff0 var se) ;
 set data21 end=eof ;
 * Fill array with the replicate means. ;
 array diff{80} diff1-diff80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (diff{j} - diff0)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = (var)**(0.5) ;
 output ;
run ;

proc print data=data22 noobs ;
 var diff0 se ;
 format diff0 se 8.4 ;
run ;
```

The SAS code of Figure 4.6.1 produces the output in Figure 4.6.2.

Figure 4.6.2: SAS Output for Estimating Variance of a Longitudinal Change

diff0	se
-6.0006	0.4328

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

Thus, the total gross vacancy rate in NYC for 2023 has decreased 6 percent from 2021 with a standard error of 0.43 percent.

Example 4.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

This example considers median gross rent and how it changes over time. Both the rate of change and resulting variance are calculated.

The statistic of interest is the percent change in the median gross rent from 2021 to 2023. Let $\widehat{M}_t$ be the estimator of the median gross rent at time t. The statistic of interest is:

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we use the 2021 replicate weights and calculate 80 replicate estimates of $\widehat{M}_{t=2021}$ and similarly use the 2023 replicate weights and calculate 80 replicate estimates of $\widehat{M}_{t=2023}$. Next, we merge the replicate estimates of $\widehat{M}_{t=2023}$ and $\widehat{M}_{t=2021}$ by replicate and calculate 80 replicate estimates of $\%\hat{\Delta}_t$. The final step is to apply Equation (2.1) to the replicate estimates of $\%\widehat{\Delta}_t$.

Figure 4.7.1 shows how this can be done with SAS.

Figure 4.7.1: SAS Code for Estimating Variance of a Rate of Change

```
* Subset to renter-occupied units with valid rent. ;
Data D2023;
 set HU23;
 where occ=1 and tenure=1 and Grent not in (-2,-1,-3);
run;

Data D2021;
 set HU21;
 where occ=1 and tenure=1 and Grent not in (-2,-1);
run;

* This empty data set is produced for the merge later. ;
data data23 ;
 length diff0-diff80 8. ;
run ;

* Steps 1 & 2: Estimate the replicate estimates for replicates 0-80. ;
%macro repssss(rep) ;
* Estimate theta for each year. ;
proc means data= d2023 median noprint ;
 weight fw&rep. ;
 var Grent ;
 output out=datam23rep&rep. median=median23rep&rep. ;
run ;

proc means data= d2021 median noprint ;
 weight fw&rep. ;
 var Grent ;
 output out=datam21rep&rep. median=median21rep&rep. ;
run ;

* Merge the replicate estimates of each year by replicate. ;
data datamrep&rep. ;
 merge datam23rep&rep. datam21rep&rep. ;
 diff&rep. = (median23rep&rep.-median21rep&rep. )/median21rep&rep.;
 keep diff&rep. ;
run ;

data data23 ;
 merge data23 datamrep&rep.;
run ;
```

Figure 4.7.1: SAS Code for Estimating Variance of a Rate of Change

```
%mend repssss ;

%macro doit ;
 %do i=0 %to 80 ;
  %repssss(&i.) ;
  %end ;
%mend doit ;
%doit ;

* Apply Step 3 to the replicate estimates and estimate the variance. ;
data data24 (keep=diff0 var se) ;
 set data23 end=eof ;
 * Fill array with the replicate means. ;
 array diff{80} diff1-diff80 ;
 * Fill array with the squared diffs. ;
 array sdiffsq{80} sdiffsq1-sdiffsq80 ;
 do j = 1 to 80 ;
  sdiffsq{j} = (diff{j} - diff0)**2 ;
  end ;
 * Sum the squared diffs. ;
 totdiff = sum(of sdiffsq1-sdiffsq80) ;
 var = (4/80) * totdiff ;
 se = (var)**(0.5) ;
 output ;
run ;

proc print data=data24 noobs ;
 var diff0 se ;
 format diff0 se 8.4 ;
run ;
```

The SAS code of Figure 4.7.1 produces the output in Figure 4.7.2.

Figure 4.7.2: SAS Output for Estimating Variance of a Rate of Change

diff0	se
0.0756	0.0149

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

So the median gross rent in NYC for 2023 has increased 7.6 percent from 2021, with a standard error of 1.5 percent.

STATA

Example 5.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the

estimates. Example 5.7 will consider an estimate of percent of change that is calculated at the HU level.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we are interested in the estimated gross vacancy rate or $\hat{p}_t = \widehat{X}_t / \widehat{N}_t * 100$, where $\widehat{X}_t$ is the estimator of the total number of vacant HUs at time t, and $\widehat{N}_t$ is the estimator of the total number of HUs at time t.

Further, we are really interested in the difference in the gross vacancy rate between 2023 and 2021, so the statistic of interest is:

$$\hat{\Delta}_{t=2021} = \frac{\hat{X}_{t=2023}}{\widehat{N}_{t=2023}} * 100 - \frac{\hat{X}_{t=2021}}{\widehat{N}_{t=2021}} * 100$$

To estimate the variance of $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces of $\hat{\Delta}_{t=2023}$ for each of the 80 replicates, making 80 replicate estimates of $\hat{\Delta}_{t=2023}$, repeat those steps for 2021, and then apply Equation (2.1). Figure 5.6.1 shows how we do this with STATA.

Figure 5.6.1: STATA Code for Estimating Variance of a Longitudinal Change

```
* Read in 2023 data:
use "HU23.dta", clear

* Generate vacancy variable:
gen vac = (occ != 1)

* Calculating 2023 totals:
quietly summarize fw if vac == 1
scalar vac_est23 = r(sum)

quietly summarize fw
scalar tot_est23 = r(sum)

scalar est23 = (vac_est23 / tot_est23) * 100

* Creating matrix that stores 2023 estimates:
matrix all_ests_23 = (est23)

* Now going through replicates:
foreach x of varlist fw1-fw80 {
    quietly summarize `x' if vac == 1
    scalar vac_est23_i = r(sum)
    quietly summarize `x'
    scalar tot_est23_i = r(sum)
    scalar est23_i = (vac_est23_i / tot_est23_i) * 100
    matrix all_ests_23 = (all_ests_23 \ est23_i)
}

* Now reading in 2021 data:
import delimited "hu21.csv", clear
```

Figure 5.6.1: STATA Code for Estimating Variance of a Longitudinal Change

```
* Generate vacancy variable:
gen vac = (occ != 1)

* Calculating 2021 totals:
quietly summarize fw0 if vac == 1
scalar vac_est21 = r(sum)

quietly summarize fw0
scalar tot_est21 = r(sum)

scalar est21 = (vac_est21 / tot_est21) * 100

* Creating matrix that stores 2021 estimates:
matrix all_ests_21 = (est21)

* Now going through replicates:
foreach x of varlist fw1-fw80 {
    quietly summarize `x' if vac == 1
    scalar vac_est21_i = r(sum)
    quietly summarize `x'
    scalar tot_est21_i = r(sum)
    scalar est21_i = (vac_est21_i / tot_est21_i) * 100
    matrix all_ests_21 = (all_ests_21 \ est21_i)
}

* Matrix will point estimates for the sample estimate and all replicates:
matrix diffs = all_ests_23 - all_ests_21

* Now getting squared differences for variance estimation:
scalar summed_variance = 0
quietly forvalues i = 2/81 {
    scalar sq_pt_est_diffs = (diffs[`i',1] - diffs[1,1])^2
    scalar summed_variance = summed_variance + sq_pt_est_diffs
}

scalar summed_variance = (4/80) * summed_variance
scalar se_est = sqrt(summed_variance)
scalar point_est = diffs[1,1]

* Final Point Estimate:
display point_est

* Final Estimated Standard Error:
display se_est
```

The STATA code of Figure 5.6.1 generates the output of Figure 5.6.2.

Figure 5.6.2: STATA Output for Estimating Variance of a Longitudinal Change

```
* Final Point Estimate:
. display point_est
-6.0005652

. * Final Estimated Standard Error:
```

```
.
. display se_est
.43279899
```

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

So, the change in the gross vacancy rate from 2021 to 2023 is a 6 percent decrease, with a standard error of 0.4 percent.

Example 5.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

There are many different statistics that measure the change between two cycles of NYCHVS. Differences, percent change, and ratios can be used to measure how much a given statistic changed from one cycle of NYCHVS to another. This example considers the housing characteristic of the median gross rent and how it can change over time. Both the rate of change and resulting variance are demonstrated.

The statistic of interest is the percent change in the median gross rent from 2021 to 2023. Let $\widehat{M}_t$ be the estimator of the median gross rent at time t. The statistic of interest is

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we use the 2023 replicate weights and store 80 replicate estimates of $\widehat{M}_{t=2023}$ *within a matrix,* and similarly use the 2021 replicate weights to store 80 replicate estimates of $\widehat{M}_{t=2021}$ in another matrix. Next, we take the difference of these two matrices $\widehat{M}_{t=2023}$ and $\widehat{M}_{t=2021}$ by replicate and then divide each matrix entry by $\widehat{M}_{t=2021}$ to get our 80 replicate estimates of $\%\hat{\Delta}_t$. The final step is to apply Equation (2.1) to the replicate estimates of $\%\hat{\Delta}_t$.

Figure 5.7.1 shows how this can be done with STATA.

Figure 5.7.1: STATA Code for Estimating Variance of a Rate of Change

```
* Read in 2023 data:
use "puf/alloccvac_23", clear


* Generating needed variables - renters and gross rent:
gen renters = (rent_amount != -2) & (grent > 0)
gen gross_rent = grent
replace gross_rent = . if grent <= 0

* Calculating 2023 median gross rent:
quietly summarize gross_rent [w=fw] if renters == 1, detail
scalar rent23 = r(p50)

* Matrix to store 2023 estimates:
```

Figure 5.7.1: STATA Code for Estimating Variance of a Rate of Change

```
matrix rent_ests_23 = (rent23)

* Now looping through all repweights for 2023:
foreach x of var fw1-fw80 {

      quietly summarize gross_rent [w=`x'] if renters == 1, detail
      scalar rent23_ith = r(p50)

      * Adding to matrix that stores 2023 estimates:
      matrix rent_ests_23 = (rent_ests_23 \ rent23_ith)
}

* Now to 2021 data:
import delimited "hu21.csv", clear

* Generating needed variables - renters and gross rent:
gen renters = (rent_amount != -2) & (grent > 0)
gen gross_rent = grent
replace gross_rent = . if grent <= 0

* Calculating 2021 median gross rent:
quietly summarize gross_rent [w=fw0] if renters == 1, detail
scalar rent21 = r(p50)

* Creating matrix that stores 2021 estimates:
matrix rent_ests_21 = (rent21)

* Now looping through all repweights for 2021:
foreach x of var fw1-fw80 {

      quietly summarize gross_rent [w=`x'] if renters == 1, detail
      scalar rent21_ith = r(p50)

      * Adding to matrix that stores 2021 estimates:
      matrix rent_ests_21 = (rent_ests_21 \ rent21_ith)
}

* Final Matrix - for now storing the numerator of the estimate for
sample and all replicates:
matrix med_rent_increase_diff = (rent_ests_23 - rent_ests_21)
matrix med_rent_increase_est = J(81,1,0)

forvalues i = 1/81 {

      * Converting entries in final matrix to hold final estiamtes for
full sample and all replicates:
      matrix med_rent_increase_est[`i',1] =
med_rent_increase_diff[`i',1] / rent_ests_21[`i',1]
}

* Now getting squared differences for variance estimation:
scalar summed_variance = 0
quietly forvalues i = 2/81 {
```

Figure 5.7.1: STATA Code for Estimating Variance of a Rate of Change

```
      scalar sq_pt_med_increase_est = (med_rent_increase_est[`i',1] -
med_rent_increase_est[1,1])^2
      scalar summed_variance = summed_variance +
sq_pt_med_increase_est
}

scalar summed_variance = (4/80) * summed_variance
scalar se_est = (summed_variance)^(0.5)
scalar point_est = med_rent_increase_est[1,1]

* Final Point Estimate:
display point_est

* Final Estimated Standard Error:
display se_est
```

The STATA code of Figure 5.7.1 generates the output of Figure 5.7.2.

Figure 5.7.2: STATA Output for Estimating Variance of a Rate of Change

```
. * Final Point Estimate:
. display point_est
.07555556

.
. * Final Estimated Standard Error:
. display se_est
.01495792
```

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

Therefore, the median gross rent for NYC in 2023 has increased 7.6 percent from 2021, with a standard error of 1.5 percent.

R

Example 6.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 6.7 will consider an estimate of percent of change that is calculated at the HU level. We present these two longitudinal statistics as examples thinking these calculations are the most likely of calculations that data users would be interested in.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we'll estimate the gross vacancy rate or $\hat{p}_t = \hat{X}_t / \hat{N}_t * 100$, where $\hat{X}_t$ is the estimator of the total number of vacant HUs at time t, and $\hat{N}_t$ is the estimator of the total number of HUs at time t.

Further, we're interested in the difference in the gross vacancy rate between 2023 and 2021, so the statistic of interest is:

$$\hat{\Delta}_{t=2023} = \frac{\hat{X}_{t=2023}}{\hat{N}_{t=2023}} * 100 - \frac{\hat{X}_{t=2021}}{\hat{N}_{t=2021}} * 100$$

To estimate the variance *of* $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces of $\hat{\Delta}_{t=2023}$ for each of the 80 replicates, make 80 replicate estimates of $\hat{\Delta}_{t=2023}$, repeat those steps for 2021, and then apply Equation (2.1). Figure 6.6.1 shows how we do this with R.

Figure 6.6.1: R Command for Estimating Variance of a Longitudinal Change

```
## First Longitudinal Example:
# Adding vacancy dummy:
hu23$vac <- ifelse(hu23$OCC != 1, 1, 0)
hu21$vac <- ifelse(hu21$OCC != 1, 1, 0)

# Calculating 2023 point estimates:
hu23_vac <- hu23 %>% filter(vac == 1)
vac_est23 <- sum(hu23_vac$FW)
total_est23 <- sum(hu23$FW)
est23 <- (vac_est23 / total_est23) * 100

# Calculating 2021 point estimates:
hu21_vac <- hu21 %>% filter(vac == 1)
vac_est21 <- sum(hu21_vac$FW0)
total_est21 <- sum(hu21$FW0)
est21 <- (vac_est21 / total_est21) * 100

# Final Point Estimate:
point_est <- est23 - est21

# Now looping:
reps <- rep(0,80)
for (i in 1:80) {
  vac_rep23 <- sum(hu23_vac[,paste0("FW", as.character(i))])
  tot_rep23 <- sum(hu23[,paste0("FW", as.character(i))])
  vac_rep21 <- sum(hu21_vac[,paste0("FW", as.character(i))])
  tot_rep21 <- sum(hu21[,paste0("FW", as.character(i))])
  reps[i] <- ((vac_rep23 / tot_rep23) - (vac_rep21 / tot_rep21)) * 100

}

reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
             ", ", as.character(round(se_est , 4))))
```

The R command of Figure 6.6.1 produces the output in Figure 6.6.2.

Figure 6.6.2: R Output for Estimating Variance of a Longitudinal Change

```
> # Final Point Estimate:
> est
[1] -6.0006
> # Final Estimate of Standard Error:
> est_se
[1] 0.4328
```

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

The change in the gross vacancy rate from 2021 to 2023 is a 6.0 percent decrease, with a standard error of 0.4 percent.

Example 6.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

There are many different statistics that measure the change between two cycles of NYCHVS. Differences, percent change, and ratios can be used to measure how much a given statistic changed from one cycle of NYCHVS to another. This example considers the housing characteristic of the median gross rent and how it can change over time. Both the rate of change and resulting variance are demonstrated.

The statistic of interest is the percent change in the median gross rent from 2017 to 2021. Let $\widehat{M}_t$ *be the estimator of t*he median gross rent at time t. The statistic of interest is:

$$\%\hat{\Delta}_t = \frac{\widehat{M}_t - \widehat{M}_{t-1}}{\widehat{M}_{t-1}}$$

To estimate the variance of $\%\hat{\Delta}_t$, we first calculate the point estimates of $\%\hat{\Delta}_t$. Then, we use both the 2021 and 2017 replicate weights to store 80 replicate estimates of $\%\hat{\Delta}_t$ *within a vector* and store the squared differences between these replicate estimates and the final point estimate in another vector. We apply Equation (2.1) to arrive at our final estimate of standard error for $\%\hat{\Delta}_t$.

Figure 6.7.1 shows how this can be done with R.

Figure 6.7.1: R Command for Estimating Variance of a Rate of Change

```
# Calculating 2023 point estimates:
rent23 <- wgt_per_sas(rent_data, "gross_rent", "FW", 0.5)

# Calculating 2021 point estimates:
rent_data_21 <- hu21 %>% filter(TENURE == 1 , GRENT != -1, GRENT != -2)
rent21 <- wgt_per_sas(rent_data_21, "GRENT", "FW0", 0.5)

# Final Point Estimate:
point_est <- (rent23 - rent21) / rent21
```

Figure 6.7.1: R Command for Estimating Variance of a Rate of Change

```
# Now looping:
reps <- rep(0,80)
for (i in 1:80) {
  rent23_est <- wgt_per_sas(rent_data, "gross_rent",
                  paste0("FW", as.character(i)), 0.5)
  rent21_est <- wgt_per_sas(rent_data_21, "GRENT",
                  paste0("FW", as.character(i)), 0.5)
  reps[i] <- (rent23_est - rent21_est) / rent21_est

}

reps <- (reps - point_est)^2
se_est = ((4/80)*sum(reps))^(1/2)
print(paste0(as.character(round(point_est, 4)),
             ", ", as.character(round(se_est, 4))))
```

The R code of Figure 6.7.1 produces the output in Figure 6.7.2.

Figure 6.7.2: R Output for Estimating Variance of a Rate of Change

```
# Final Point Estimate:
[1] 0.0756
# Final Estimate of Standard Error:
[1] 0.0149
```

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

So, the median gross rent for NYC in 2023 has increased 7.6 percent from 2021, with a standard error of 1.5 percent.

PYTHON

Example 7.6. Estimating the Variance of a Longitudinal Change

Both this example and the next consider statistics that measure longitudinal change. This example will consider a statistic that measures the change of a rate between two cycles of the estimates. Example 7.7 will consider an estimate of percent of change that is calculated at the HU level.

In this example, we show how to estimate the variance of a difference in proportion. Specifically, we are interested in the estimated gross vacancy rate or $\hat{p}_t = \widehat{X}_t / \widehat{N}_t * 100$, where $\widehat{X}_t$ is the estimator of the total number of vacant HUs at time t, and $\widehat{N}_t$ is the estimator of the total number of HUs at time t.

Further, we are interested in the difference in the gross vacancy rate between 2021 and 2017, so the statistic of interest is:

$$\hat{\Delta}_{t=2023} = \frac{\hat{X}_{t=2023}}{\hat{N}_{t=2023}} * 100 - \frac{\hat{X}_{t=2021}}{\hat{N}_{t=2021}} * 100$$

To estimate the variance of $\hat{\Delta}_t$ with replicate weights, we first calculate all the pieces *of* $\hat{\Delta}_{t=2023}$ for each of the 80 replicates, making 80 replicate estimates of $\hat{\Delta}_{t=2023}$, repeat those steps for 2021, and then apply Equation (2.1).

Figure 7.6.1 shows how we do this with PYTHON.

Figure 7.6.1: Python Code for Estimating Variance of a Longitudinal Change

```
reps = np.array([0.0]*80)
vac23 = (hu23.loc[hu23.vac == 1].FW.sum() / hu23.FW.sum())*100
vac21 = (hu21.loc[hu21.vac == 1].FW0.sum() / hu21.FW0.sum())*100
point_est = vac23 - vac21
for i in range(1,81):
    vac23 = (hu23.loc[hu23.vac == 1]["FW"+str(i)].sum() /
hu23["FW"+str(i)].sum())*100
    vac21 = (hu21.loc[hu21.vac == 1]["FW"+str(i)].sum() /
hu21["FW"+str(i)].sum())*100
    reps[i-1] = vac23 - vac21
reps = (reps - point_est)**2
se_est= ((4/80)*np.sum(reps))**(1/2)
print( point_est.round(decimals = 4), se_est.round(decimals = 4))
```

The Python code of Figure 7.6.1 generates the output of Figure 7.6.2.

Figure 7.6.2: Python Output for Estimating Variance of a Longitudinal Change

```
-6.0006 0.4328
```

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

So, the change in the gross vacancy rate from 2021 to 2023 is a 6 percent decrease, with a standard error of 0.4 percent.

Example 7.7. Estimating the Variance of a Percent Change in Two Cycles of NYCHVS

There are many different statistics that measure the change between two cycles of NYCHVS. Differences, percent change, and ratios can be used to measure how much a given statistic changed from one cycle of NYCHVS to another. This example considers the housing characteristic of the median gross rent and how it can change over time. Both the rate of change and resulting variance are demonstrated.

The statistic of interest is the percent change in the median gross rent from 2021 to 2023. Let $\widehat{\mathrm{M}}_{\mathrm{t}}$ be the estimator of the median gross rent at time t. The statistic of interest is:

$$\%\widehat{\Delta}_{\mathrm{t}} = \frac{\widehat{\mathrm{M}}_{\mathrm{t}} - \widehat{\mathrm{M}}_{\mathrm{t-1}}}{\widehat{\mathrm{M}}_{\mathrm{t-1}}}$$

To estimate the variance of $\%\widehat{\Delta}_{\mathrm{t}}$, we use the 2023 replicate weights and store 80 replicate estimates of $\widehat{M}_{t=2023}$ within a matrix, and similarly use the 2021 replicate weights to store 80 replicate estimates of $\widehat{M}_{t=2021}$ in another matrix. Next, we take the difference of these two matrices $\widehat{M}_{t=2023}$ and $\widehat{M}_{t=2021}$ by replicate and then divide each matrix entry by $\widehat{M}_{t=2021}$ to get our 80 replicate estimates of $\%\widehat{\Delta}_{\mathrm{t}}$. The final step is to apply Equation (2.1) to the replicate estimates of $\%\widehat{\Delta}_{\mathrm{t}}$.

Figure 7.7.1 shows how this can be done with Python.

Figure 7.7.1: Python Code for Estimating Variance of a Rate of Change

```
rent_data_21 = hu21.loc[hu21.TENURE == 1 & hu21.gross_rent.notnull()]
reps = np.array([0.0]*80)
point_est_23 = wgt_per_sas(rent_data, 'gross_rent', 'FW', 0.5)
point_est_21 = wgt_per_sas(rent_data_21, 'gross_rent', 'FW0', 0.5)
point_est = (point_est_23 - point_est_21) / point_est_21
for i in range(1,81):
    point_est_23 = wgt_per_sas(rent_data, 'gross_rent', 'FW'+str(i), 0.5)
    point_est_21 = wgt_per_sas(rent_data_21, 'gross_rent', 'FW'+str(i), 0.5)
    reps[i-1] = (point_est_23 - point_est_21) / point_est_21
reps = (reps - point_est)**2
se_est= ((4/80)*np.sum(reps))**(1/2)
print(point_est.round(decimals = 4), se_est.round(decimals = 4))
```

The Python code of Figure 7.7.1 generates the output of Figure 7.7.2.

Figure 7.7.2: Python Output for Estimating Variance of a Rate of Change

```
0.0756 0.0149
```

Source: U.S. Census Bureau, 2021 and 2023 New York City Housing and Vacancy Survey.

Therefore, the median gross rent for NYC in 2023 has increased 7.6 percent from 2021, with a standard error of 1.5 percent.