Stat 854 Project: Mirror-match Bootstrap

. . .

April 14 2024

Introduction

Method

Experiments

Related Works

Conclusion

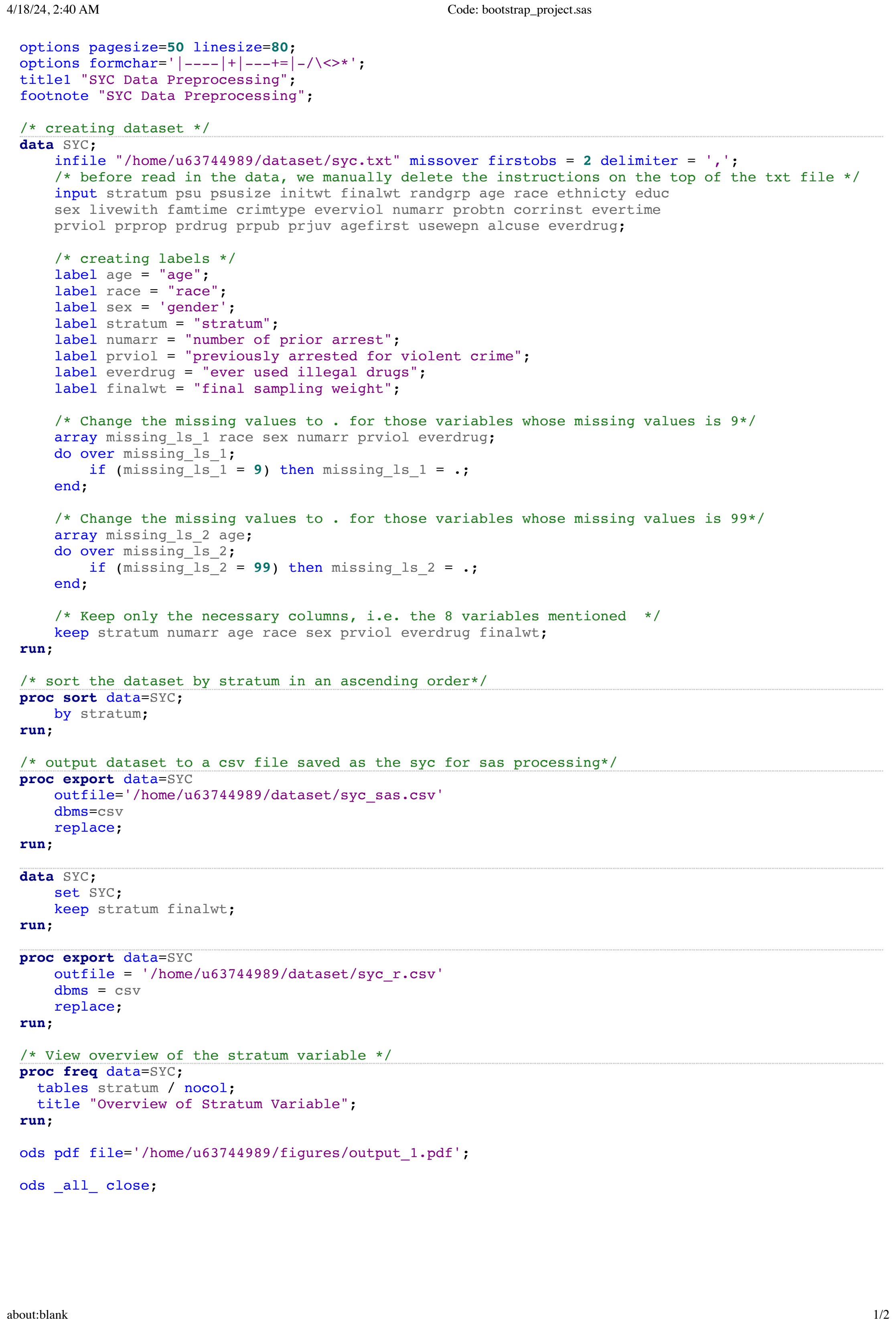
Appendix

1. We pre-process the syc.txt in SAS, where

- we deal with missing values in the 8 variables mentioned in A2Q4 by setting missing values as '.'
- we keep only the necessary columns, i.e. the 8 variables mentioned in A2Q4
- sort the dataset by stratum in an ascending order
- export a syc dataset called syc_sas.csv with 8 variables for calculation in SAS
- export another syc dataset call syc_r.csv with 2 variables (stratum, finalwt) for computing bootstrap weight in R

In addition, PROC FREQ is used to generate frequency counts for the stratum variable.

SAS code

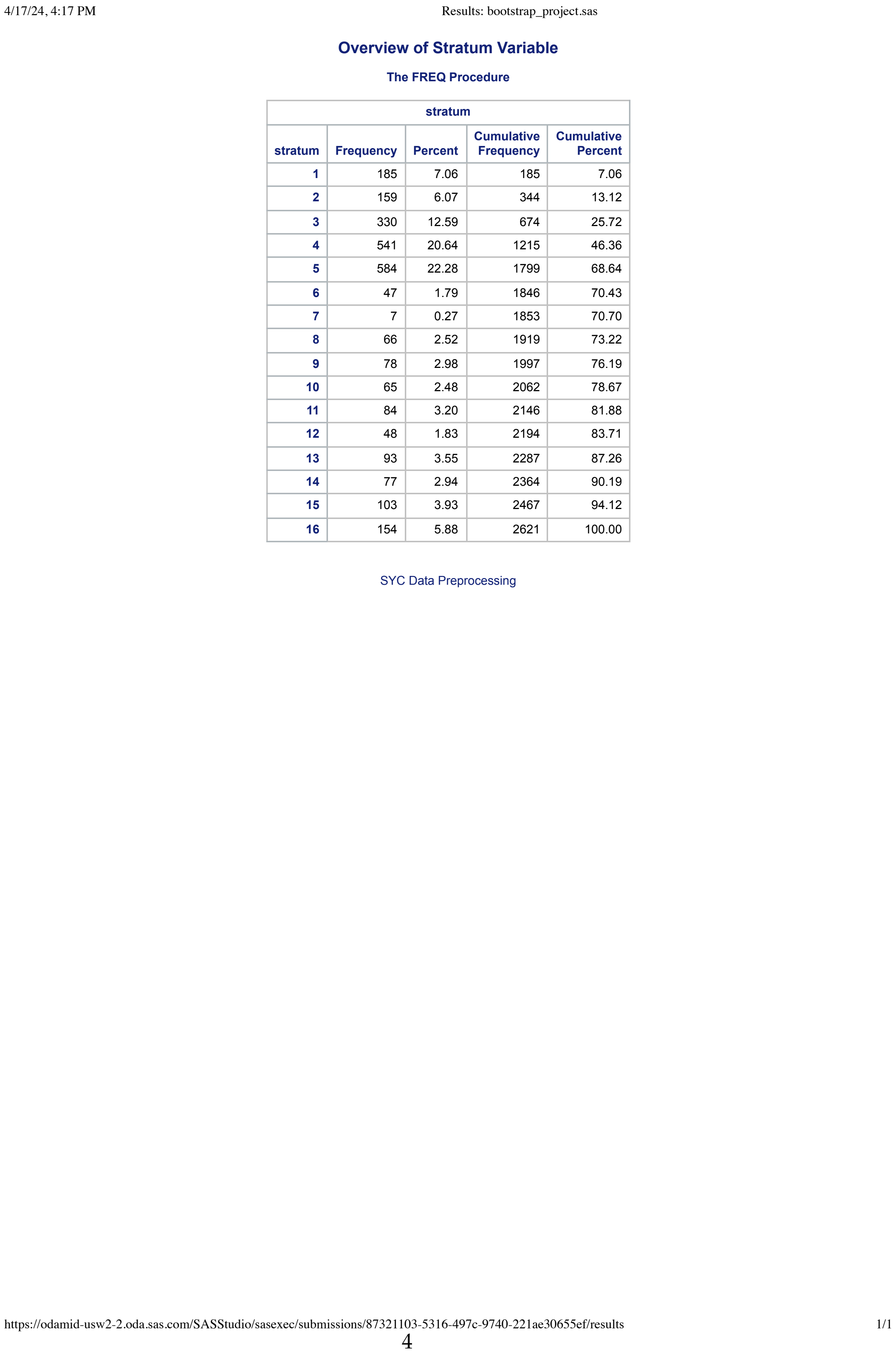


4/18/24, 2:40 AM Code: bootstrap_project.sas

```
options pagesize=50 linesize=80;
options formchar='|----|+|---+=|-/\<>*';
title1 "SYC Data Preprocessing";
footnote "SYC Data Preprocessing";

/* creating dataset */
data SYC;
    infile "/home/u63744989/dataset/syc.txt" missover firstobs = 2 delimiter = ',';
    /* before read in the data, we manually delete the instructions on the top of the txt file */
    input stratum psu psusize initwt finalwt randgrp age race ethnicty educ
    sex livewith famtime crimtype everviol numarr probtn corrinst evertime
    prviol prprop prdrug prpub prjuv agefirst usewepn alcuse everdrug;

    /* creating labels */
    label age = "age";
    label race = "race";
    label sex = 'gender';
    label stratum = "stratum";
    label numarr = "number of prior arrest";
    label prviol = "previously arrested for violent crime";
    label everdrug = "ever used illegal drugs";
    label finalwt = "final sampling weight";

    /* Change the missing values to . for those variables whose missing values is 9*/
    array missing_ls_1 race sex numarr prviol everdrug;
    do over missing_ls_1;
        if (missing_ls_1 = 9) then missing_ls_1 = .;
    end;

    /* Change the missing values to . for those variables whose missing values is 99*/
    array missing_ls_2 age;
    do over missing_ls_2;
        if (missing_ls_2 = 99) then missing_ls_2 = .;
    end;

    /* Keep only the necessary columns, i.e. the 8 variables mentioned  */
    keep stratum numarr age race sex prviol everdrug finalwt;
run;

/* sort the dataset by stratum in an ascending order*/
proc sort data=SYC;
    by stratum;
run;

/* output dataset to a csv file saved as the syc for sas processing*/
proc export data=SYC
    outfile='/home/u63744989/dataset/syc_sas.csv'
    dbms=csv
    replace;
run;

data SYC;
    set SYC;
    keep stratum finalwt;
run;

proc export data=SYC
    outfile = '/home/u63744989/dataset/syc_r.csv'
    dbms = csv
    replace;
run;

/* View overview of the stratum variable */
proc freq data=SYC;
  tables stratum / nocol;
  title "Overview of Stratum Variable";
run;

ods pdf file='/home/u63744989/figures/output_1.pdf';

ods _all_ close;
```

about:blank 1/2

SAS output



4/17/24, 4:17 PM Results: bootstrap_project.sas

Overview of Stratum Variable

The FREQ Procedure

stratum				
stratum	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	185	7.06	185	7.06
2	159	6.07	344	13.12
3	330	12.59	674	25.72
4	541	20.64	1215	46.36
5	584	22.28	1799	68.64
6	47	1.79	1846	70.43
7	7	0.27	1853	70.70
8	66	2.52	1919	73.22
9	78	2.98	1997	76.19
10	65	2.48	2062	78.67
11	84	3.20	2146	81.88
12	48	1.83	2194	83.71
13	93	3.55	2287	87.26
14	77	2.94	2364	90.19
15	103	3.93	2467	94.12
16	154	5.88	2621	100.00

SYC Data Preprocessing

https://odamid-usw2-2.oda.sas.com/SASStudio/sasexec/submissions/87321103-5316-497c-9740-221ae30655ef/results 1/1

2. We read in the syc in R and log the N_h information provided in A2Q4

```r
# Read in the Survey of Youth in Custody
syc <- readr::read_csv(
  file = "data/syc_r.csv", # Tell it where the file is
  col_types = "nn", # Tell it that there are two columns, and they are "numeric" (n)
)

# glimpse the read data set
dplyr::glimpse(syc)
```

```
## Rows: 2,621
## Columns: 2
## $ stratum <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ finalwt <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,~
```

```r
# Read the stratum size provided in Assignment 2 Question 4, shape (16, 1)
N_h_list = c(2724, 3192, 4107, 2705, 3504, 376, 56,
             528, 624, 520, 672, 384, 744, 847, 824, 1848)
```

3. We write the function for performing one mirror-match bootstrap for stratum h

For n'_h, we choose $n'_h = f_n \times n_h$ advised by author in the paper (Sitter 1992b), so that the first two moments of the distribution of the bootstrap estimate of $\bar{y}$ match the usual unbiased estimates of the first two moments of $\bar{y}$.

Randomization of n_h

For stratum h in each bootstrap, the n'_h is a random variable with discrete distribution across $\{n'_{h,floor}, n'_{h,ceil}\}$ whose pmf is given below:

$$Pr(n'_h = n'_{h,floor}) = n'_{h,ceil} - n'_h = p,$$

and $Pr(n'_h = n'_{h,ceil})$ is:

$$Pr(n'_h = n'_{h,ceil}) = 1 - p,$$

where $n'_{h,floor}$ is equal to `floor(n_h_prime)` in R; $n'_{h,ceil}$ is equal to `ceiling(n_h_prime)` in R. The randomization is done at the beginning of a bootstrap for each stratum h independently and repeated at each bootstrap as described in the paper (Sitter 1992a).

Randomization of k_h

For stratum h in each bootstrap, given the value of n'_h obtained in the randomization above, the k_h is a random variable with discrete distribution across $\{k_{h,floor}, k_{h,ceil}\}$ whose pmf is described below:

$$Pr(k_h = k_{h,floor}) = \frac{\left(\frac{1}{k_h} - \frac{1}{k_{h,ceil}}\right)}{\left(\frac{1}{k_{h,floor}} - \frac{1}{k_{h,ceil}}\right)} = p_h,$$

and $Pr(k_h = k_{h,ceil})$ is:

$$Pr(k_h = k_{h,ceil}) = 1 - p_h,$$

where $k_{h,floor}$ is equal to `floor(k_h)` in R; $k_{h,ceil}$ is equal to `ceiling(k_h)` in R.
The randomization is done independently for each stratum h and repeated at each bootstrap as described in the paper (Sitter 1992a).

```r
mirror_match_bootstrap_h = function(dataset, N_h_list, h)
  # Perform one mirror-match bootstrap for stratum h
  # parameters
  # ----------
  #
  # dataset : data.frame
  #   dataset of shape (n, 2)
  # N_h_list : vector
  #   vector of shape (16, 1)
  # h : int
  #   stratum number
  #
  # return
  # ------
  #
  # bootstrap_weights : vector
  #   bootstrap weights for the sample units in the stratum (from 1 to n_h)
  #   shape (n_h, 1)

{
  # get the stra_weights df, shape is (n_h, 2)
  stra_weights_df = get_stratum(dataset, h)
  # get the stra_weights vector, shape is (n_h, 1)
  stra_weights = stra_weights_df$finalwt
  # get n_h from the dimension of stra_weights_df
  n_h = dim(stra_weights_df)[1]
  # get N_h from the N_h_list
  N_h = N_h_list[h]
  # compute the n_h_prime using the formula n_{h}^{\prime} = f_n \times n_h
  n_h_prime = n_h * (n_h / N_h)
  # get random integer n_h_prime by applying random_n_h_prime if it is not integer
  if (!n_h_prime %% 1 == 0)
  {
    n_h_prime = random_n_h_prime(n_h_prime, n_h)
  }
  # compute k_h
  k_h = get_k_h(n_h, n_h_prime, N_h)
  # create a vector of indices for the stra_weights, shape is (n_h, 1)
  indices = seq(1, n_h)
  # create a counter to count the number of times a index is selected, shape is (n_h, 1)
  counter = rep(0, n_h)
  # print information about the stratum h
  # print(paste("The stratum is", h, "the n_h is", n_h, "the n_h_prime is", n_h_prime, "the k

  # carry out sample without replacement from indices (k_h times), sample size is n_h_prime
  for (s in 1:k_h)
  {
    # sample without replacement
    bootstrapsample = sample(indices, n_h_prime, replace = FALSE)
    for (index in bootstrapsample)
    {
      # log the number of appearances of index of elements in stratum h in the bootstrapsampl
      counter[index] = counter[index] + 1
    }
  }
  # calculate the bootstrap weights using stra_weights (n_h, 1) and counter (n_h, 1)
  # apply element-wise multiplication of the two vectors
  bootstrap_weights = counter * stra_weights

  # make sure the length of bootstrap_weights is n_h
  if (length(bootstrap_weights) == n_h)
  {
    return(bootstrap_weights)
  }
  else
  {
    stop("bootstrap_weights does not have length n_h")
  }

}

get_stratum = function(dataset, h)  # get the weights of stratum h from the dataset
{
  # make sure stratum number is in the range of 1 to 16
  if (h %in% seq(1, 16))
  {
    # get the weights of stratum h from the dataset
    stra_weights_df = dataset[dataset$stratum == h, ]
    return(stra_weights_df)
  }
  else
  {
    stop("Stratum number does not exist.")
  }
}

get_k_h = function(n_h, n_h_prime, N_h)  # compute the k_h, the number of resamplings with re
{
  f_h = n_h / N_h  # compute f_h
  f_h_star = n_h_prime / n_h  # compute f_h_star
  k_h = (n_h * (1 - f_h_star)) / (n_h_prime * (1 - f_h))  # compute k_h using the formula
  # apply the random_k_h function if k_h is not integer

  if (!k_h%%1 == 0)
  {
    k_h = random_k_h(k_h)
  }
  return(k_h)
}

random_n_h_prime = function(n_h_prime, n_h)  # randomization for k_h
{
  # make sure n_h_prime is in the proper range
  if (n_h_prime >= 1 && n_h_prime < n_h)
  {
    # get k_h_floor and k_h_ceil
    n_h_prime_floor = floor(n_h_prime)
    n_h_prime_ceil = ceiling(n_h_prime)

    # get prob for n_h_prime_floor
    # apply the formula in the paper
    p_floor = n_h_prime_ceil - n_h_prime
    # get prob for k_h_ceil
    p_ceil = 1 - p_floor

    # get the random integer n_h_prime based on the p_floor and p_ceil
    n_h_prime = sample(c(n_h_prime_floor, n_h_prime_ceil), 1, prob = c(p_floor, p_ceil))
    return(n_h_prime)
  }
  else
  {
    # if n_h_prime is less than 1, assign it as 1
    if (n_h_prime < 1)
    {
      n_h_prime = 1
      return(n_h_prime)
    }
    else
    {
      stop(paste("n_h_prime must be larger than or equal to 1 and less than n_h, however got"
    }
  }
}

random_k_h = function(k_h)  # randomization for k_h
{
  # make sure k_h is larger than 1
  if (k_h >= 1)
  {
    # get k_h_floor and k_h_ceil
    k_h_floor = floor(k_h)
    k_h_ceil = ceiling(k_h)

    # get prob for k_h_floor
    # apply the formula in the paper
    p_floor = ((1 / k_h) - (1 / k_h_ceil)) / ((1 / k_h_floor) - (1 / k_h_ceil))
    # get prob for k_h_ceil
    p_ceil = 1 - p_floor

    # get the random integer k_h based on the p_floor and p_ceil
    k_h = sample(c(k_h_floor, k_h_ceil), 1, prob = c(p_floor, p_ceil))
    return(k_h)
  }
  else
  {
    stop("k_h must be larger than or equal to 1")
  }
}

weights = mirror_match_bootstrap_h(syc, N_h_list, 3)
weights
```

```
##   [1] 11 22 11 11 11  0 11 11 11 22 11  0  0 11  0  0 11  0 11 11  0 22 11  0  0
##  [26] 22 11  0 11 11  0  0  0 22 22  0  0 33 11 22 22 11 11 22 11  0  0  0 22 11
##  [51]  0 11 11 11  0 11  0 24  0 12 24 24  0  0  0  0 12 12  0  0 36 36 24 12 24
##  [76] 12  0 12 12 24 12  0 12 12 12 24  0 24 12  0 12 12 12 24  0  0 12  0 36  0
## [101]  0  0 12 36  0 24 12 12 24 36  0 36 24  0 12 36 12  0 24 12 48 12 24 12 24
## [126] 12 12 12  0 12 24 12  0 12 24 24  0 24 12  0 12  0 12 12 12  0  0 12  0 24
## [151]  0 12 15 15 21 15  0  0 21 42 63  0 42 15 15  0 42 30 15 15  0 30  0 45 30
## [176] 30 45  0  0 15  0 15  0  0 15 15 21 21 15  0 21  0 15  0 42 21  0 42  0 30
## [201]  0  0  0 42 15 21 21 15 15  0 42 15  0 42 45 21  0  0  0 30 30  0  0 42 15
## [226] 30 15 42  0 15  0  0  0 15 15 21 42 21 42 15 63 21 15  0 60 63 42  0 42 15
## [251] 22 22 22  0 22  0 22 11 33  0  0  0 22 11 11  0 33 11 11  0 11  0 22 11 11
## [276] 22  0 11 11 33 33  0 11 11  0  0 33 22 11 22  0 11 22 11 44 22  0 11 11 22
## [301] 22  0  0 11  0 22  0 11 22 22 11  0  0 11 11 11 22 11  0  0  0  0 11  0  0
## [326] 11 33  0 11  0
```

4. We perform one complete bootstrap for the sample of size n. Specifically, we perform bootstrap for all the 16 strata in the sample.

```
mirror_match_bootstrap_full = function(dataset, N_h_list, stratum_list = seq(1, 16))
{
  # Perform one full mirror-match bootstrap for 16 strata
  # parameters
  # ---------
  #
  # dataset : data.frame
  #   dataset of shape (n, 2)
  # N_h_list : vector
  #   vector of shape (16, 1)
  # stratum_list : vector
  #   list contatining all the stratum numbers
  #   default is seq(1, 16)
  #
  # return
  # ------
  #
  # bootstrap_weights : vector
  #   bootstrap weights for the full sample of size n consisting of 16 strata
  #   shape (n, 1)

  # compute the sample size n
  n = dim(dataset)[1]
  # create an empty vector for storing bootstrap weights
  bootstrap_weights_full = c()
  for (h in stratum_list)
  {
    # obtain the bootstrap weights for stratum h by applying the function mirror_match_bootst
    bootstrap_weights_h = mirror_match_bootstrap_h(dataset, N_h_list, h)
    # concatenate the new bootstrap weights to the bootstrap_weights_full
    bootstrap_weights_full = c(bootstrap_weights_full, bootstrap_weights_h)
  }

  # make sure the length of bootstrap_weights_full be n
  if (length(bootstrap_weights_full) == n)
  {
    return(bootstrap_weights_full)
  }
  else
  {
    stop("the length of bootstrap_weights_full must be n")
  }
}

a = mirror_match_bootstrap_full(syc, N_h_list)
dplyr::glimpse(a)
```

```
##  num [1:2621] 0 7 0 7 0 0 0 0 7 14 ...
```

5. Finally, we obtain the bootstrap weight matrix by repeating the full bootstrap B times. Bootstrap weight matrix has shape (n, B).

```
mirror_match_bootstrap_B = function(dataset, N_h_list, B = 100, stratum_list = seq(1, 16))
{
  stratum_vector = dataset$stratum
  bootstrap_weights_matrix = stratum_vector
  for (exp in 1:B)
  {
    bootstrap_weights_vector = mirror_match_bootstrap_full(dataset, N_h_list)
    bootstrap_weights_matrix = cbind(bootstrap_weights_matrix, bootstrap_weights_vector)
  }
   bootstrap_weights_matrix = as.data.frame(bootstrap_weights_matrix)
   names(bootstrap_weights_matrix) = c("stratum", paste0("w", 1:100))
  return(bootstrap_weights_matrix)
}

m <- mirror_match_bootstrap_B(syc, N_h_list)
```

```r
dplyr::glimpse(m)
```

```
## Rows: 2,621
## Columns: 101
## $ stratum <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ w1      <dbl> 7, 14, 0, 0, 7, 7, 7, 0, 21, 0, 7, 0, 28, 0, 7, 0, 7, 0, 0, 0,~
## $ w2      <dbl> 14, 7, 0, 7, 7, 7, 7, 0, 7, 7, 7, 7, 0, 0, 7, 7, 7, 7, 0, 0, 0~
## $ w3      <dbl> 7, 7, 14, 21, 7, 7, 7, 21, 14, 7, 21, 21, 14, 21, 7, 14, 7, 0,~
## $ w4      <dbl> 0, 0, 14, 0, 7, 0, 14, 7, 7, 0, 0, 21, 0, 0, 7, 14, 0, 7, 7, 7~
## $ w5      <dbl> 28, 14, 7, 7, 7, 0, 7, 0, 0, 7, 14, 14, 0, 0, 14, 0, 7, 14, 14~
## $ w6      <dbl> 28, 0, 7, 14, 7, 7, 0, 0, 14, 0, 7, 7, 0, 0, 21, 7, 0, 0, 7, 3~
## $ w7      <dbl> 14, 0, 14, 0, 7, 14, 14, 0, 0, 14, 14, 7, 7, 7, 7, 7, 0, 28, 0~
## $ w8      <dbl> 7, 7, 0, 14, 7, 0, 0, 21, 0, 14, 0, 14, 7, 14, 0, 14, 0, 0, 7,~
## $ w9      <dbl> 14, 7, 14, 0, 14, 14, 7, 0, 0, 0, 7, 7, 0, 14, 7, 7, 14, 14, 7~
## $ w10     <dbl> 7, 14, 7, 7, 14, 7, 0, 0, 0, 0, 14, 21, 0, 14, 7, 14, 14, 0, 7~
## $ w11     <dbl> 0, 0, 0, 7, 14, 7, 21, 21, 7, 7, 0, 0, 7, 0, 7, 0, 7, 0, 0, 7,~
## $ w12     <dbl> 7, 7, 0, 14, 7, 0, 0, 7, 7, 7, 0, 0, 7, 14, 0, 14, 14, 0, 0, 7~
## $ w13     <dbl> 21, 14, 7, 7, 0, 0, 0, 0, 0, 0, 14, 7, 7, 0, 7, 14, 0, 14, 0, ~
## $ w14     <dbl> 7, 14, 14, 7, 0, 7, 14, 14, 7, 0, 7, 7, 0, 0, 7, 7, 0, 21, 0, ~
## $ w15     <dbl> 0, 0, 0, 7, 14, 7, 7, 14, 14, 7, 0, 7, 7, 0, 0, 14, 7, 0, 21, ~
## $ w16     <dbl> 0, 0, 0, 21, 14, 7, 0, 0, 0, 14, 14, 0, 7, 7, 0, 14, 14, 0, 0,~
## $ w17     <dbl> 7, 7, 0, 7, 0, 14, 14, 21, 14, 0, 7, 21, 7, 7, 21, 7, 28, 7, 7~
## $ w18     <dbl> 14, 7, 7, 0, 7, 7, 0, 7, 7, 0, 7, 7, 0, 14, 14, 7, 7, 7, 0, 7,~
## $ w19     <dbl> 14, 0, 0, 14, 28, 14, 0, 14, 7, 0, 14, 14, 0, 7, 0, 0, 14, 7, ~
## $ w20     <dbl> 7, 14, 7, 14, 0, 14, 7, 14, 14, 0, 0, 0, 0, 7, 7, 7, 7, 14, 7,~
## $ w21     <dbl> 0, 0, 0, 7, 7, 14, 14, 35, 14, 0, 0, 0, 7, 0, 0, 7, 7, 7, 7, 7~
## $ w22     <dbl> 0, 0, 14, 7, 14, 7, 7, 0, 14, 0, 7, 7, 7, 0, 0, 0, 14, 7, 7, 0~
## $ w23     <dbl> 21, 7, 14, 0, 7, 0, 14, 28, 0, 7, 7, 7, 7, 14, 0, 21, 7, 7, 14~
## $ w24     <dbl> 7, 0, 7, 0, 0, 0, 7, 14, 0, 0, 14, 14, 21, 7, 14, 7, 0, 0, 0, ~
## $ w25     <dbl> 0, 14, 14, 7, 7, 0, 0, 7, 21, 0, 7, 7, 0, 14, 0, 14, 7, 7, 21,~
## $ w26     <dbl> 7, 7, 0, 7, 7, 0, 7, 7, 0, 7, 7, 0, 7, 7, 7, 7, 7, 0, 0, 0, 0,~
## $ w27     <dbl> 14, 14, 0, 7, 0, 7, 14, 7, 0, 7, 14, 14, 14, 21, 14, 14, 14, 7~
## $ w28     <dbl> 0, 0, 0, 0, 14, 7, 14, 7, 0, 14, 7, 14, 7, 14, 7, 0, 7, 7, 7, ~
## $ w29     <dbl> 7, 21, 14, 7, 7, 21, 7, 7, 21, 21, 14, 7, 7, 0, 0, 7, 0, 0, 0,~
## $ w30     <dbl> 0, 0, 14, 14, 0, 7, 7, 14, 14, 21, 7, 0, 7, 7, 7, 0, 7, 7, 0, ~
## $ w31     <dbl> 0, 0, 0, 7, 7, 14, 14, 7, 0, 0, 14, 7, 0, 0, 7, 7, 14, 0, 7, 7~
## $ w32     <dbl> 0, 0, 0, 7, 7, 0, 7, 14, 0, 7, 14, 7, 7, 14, 7, 7, 0, 0, 21, 7~
## $ w33     <dbl> 14, 0, 0, 21, 0, 0, 7, 7, 7, 14, 14, 14, 0, 0, 14, 7, 7, 7, 7,~
## $ w34     <dbl> 7, 14, 7, 0, 7, 0, 0, 7, 14, 0, 7, 0, 28, 7, 0, 0, 0, 0, 14, 2~
## $ w35     <dbl> 0, 7, 0, 7, 0, 0, 0, 14, 7, 7, 0, 7, 7, 14, 7, 28, 7, 0, 21, 0~
## $ w36     <dbl> 0, 7, 7, 0, 7, 7, 7, 0, 7, 7, 0, 0, 0, 7, 7, 21, 7, 21, 0, 14,~
## $ w37     <dbl> 7, 7, 7, 14, 0, 0, 0, 7, 0, 7, 0, 0, 7, 14, 7, 0, 0, 7, 7, 0, ~
## $ w38     <dbl> 21, 0, 0, 0, 14, 14, 28, 7, 7, 7, 7, 7, 7, 7, 7, 14, 7, 14, 0,~
## $ w39     <dbl> 0, 7, 0, 21, 0, 0, 0, 0, 7, 14, 0, 7, 7, 0, 14, 0, 7, 0, 0, 28~
## $ w40     <dbl> 0, 0, 7, 0, 21, 0, 0, 0, 0, 0, 21, 0, 14, 0, 14, 7, 7, 0, 7, 1~
## $ w41     <dbl> 7, 14, 7, 7, 0, 7, 7, 14, 7, 0, 0, 14, 21, 14, 0, 0, 0, 14, 14~
## $ w42     <dbl> 0, 21, 14, 7, 14, 0, 14, 14, 14, 0, 7, 0, 7, 0, 0, 0, 7, 0, 0,~
## $ w43     <dbl> 7, 0, 0, 14, 21, 0, 7, 7, 14, 7, 0, 7, 7, 0, 7, 7, 7, 21, 0, 1~
## $ w44     <dbl> 0, 7, 0, 7, 0, 7, 0, 0, 7, 14, 7, 7, 7, 7, 14, 21, 14, 0, 14, ~
## $ w45     <dbl> 14, 14, 0, 14, 21, 7, 0, 7, 0, 7, 7, 0, 0, 7, 14, 7, 21, 0, 0,~
## $ w46     <dbl> 7, 21, 7, 7, 7, 0, 0, 7, 0, 0, 21, 7, 0, 0, 0, 14, 14, 0, 7, 2~
## $ w47     <dbl> 0, 0, 14, 14, 28, 7, 0, 14, 0, 21, 7, 7, 0, 7, 0, 14, 7, 7, 0,~
## $ w48     <dbl> 0, 7, 7, 0, 7, 0, 0, 7, 0, 0, 14, 14, 0, 0, 7, 0, 21, 0, 0, 7,~
## $ w49     <dbl> 7, 14, 14, 0, 14, 0, 0, 28, 7, 7, 0, 14, 21, 7, 0, 0, 0, 21, 7~
## $ w50     <dbl> 0, 7, 0, 0, 7, 7, 7, 28, 7, 7, 0, 14, 7, 0, 0, 0, 0, 7, 14, 7,~
## $ w51     <dbl> 14, 0, 7, 0, 0, 7, 0, 0, 0, 7, 28, 7, 7, 14, 7, 0, 21, 14, 0, ~
## $ w52     <dbl> 0, 7, 7, 14, 7, 7, 0, 21, 0, 14, 0, 0, 0, 7, 7, 7, 28, 7, 7, 7~
## $ w53     <dbl> 7, 14, 7, 0, 0, 14, 14, 14, 0, 7, 0, 7, 0, 0, 0, 7, 0, 7, 7, 7~
## $ w54     <dbl> 0, 7, 0, 7, 0, 7, 7, 14, 7, 0, 0, 14, 0, 7, 7, 0, 0, 14, 7, 14~
## $ w55     <dbl> 7, 7, 0, 7, 7, 7, 0, 0, 7, 0, 0, 7, 0, 7, 0, 0, 0, 0, 7, 0, 7,~
## $ w56     <dbl> 7, 14, 0, 7, 0, 0, 21, 0, 0, 7, 7, 0, 0, 7, 7, 7, 28, 7, 7, 7,~
## $ w57     <dbl> 7, 7, 0, 0, 14, 14, 7, 0, 0, 7, 7, 0, 14, 0, 14, 0, 7, 7, 0, 0~
## $ w58     <dbl> 0, 7, 0, 0, 0, 14, 0, 7, 0, 0, 14, 0, 0, 0, 0, 0, 14, 14, 7, 2~
## $ w59     <dbl> 14, 7, 21, 14, 7, 7, 0, 14, 0, 7, 7, 7, 7, 0, 7, 0, 7, 7, 7, 0~
## $ w60     <dbl> 21, 7, 0, 7, 0, 7, 7, 14, 0, 0, 7, 7, 0, 7, 14, 0, 7, 7, 7, 7,~
## $ w61     <dbl> 7, 7, 7, 14, 7, 21, 7, 7, 14, 0, 7, 0, 14, 7, 7, 7, 7, 7, 7, 7~
## $ w62     <dbl> 42, 7, 7, 14, 0, 21, 0, 0, 0, 7, 14, 14, 21, 0, 21, 0, 7, 14, ~
## $ w63     <dbl> 0, 0, 7, 7, 7, 0, 7, 0, 7, 21, 7, 7, 7, 0, 0, 7, 7, 0, 7, 0, 2~
## $ w64     <dbl> 0, 7, 21, 0, 0, 7, 21, 0, 14, 7, 0, 7, 7, 7, 21, 7, 14, 14, 7,~
## $ w65     <dbl> 7, 14, 7, 0, 14, 7, 0, 7, 7, 0, 21, 14, 7, 0, 7, 7, 14, 0, 7, ~
## $ w66     <dbl> 7, 7, 7, 7, 14, 7, 7, 0, 0, 7, 7, 7, 7, 0, 0, 0, 21, 7, 14, 0,~
## $ w67     <dbl> 7, 7, 14, 0, 7, 7, 7, 7, 21, 7, 14, 7, 14, 7, 0, 14, 7, 0, 0, ~
## $ w68     <dbl> 21, 14, 0, 7, 0, 28, 7, 0, 7, 7, 0, 14, 14, 14, 0, 0, 7, 0, 0,~
## $ w69     <dbl> 7, 7, 0, 0, 7, 0, 14, 0, 7, 14, 0, 14, 14, 7, 0, 14, 0, 14, 14~
## $ w70     <dbl> 7, 7, 7, 0, 0, 0, 7, 7, 7, 0, 0, 7, 7, 0, 7, 7, 7, 7, 7, 7, 0,~
## $ w71     <dbl> 0, 7, 0, 21, 7, 7, 7, 7, 14, 0, 7, 14, 7, 14, 7, 7, 14, 0, 14,~
## $ w72     <dbl> 7, 0, 14, 14, 14, 7, 21, 7, 7, 14, 0, 21, 7, 21, 0, 7, 14, 0, ~
## $ w73     <dbl> 7, 28, 7, 0, 7, 7, 7, 0, 0, 7, 0, 7, 14, 21, 0, 14, 21, 7, 0, ~
## $ w74     <dbl> 0, 14, 0, 7, 0, 0, 7, 0, 7, 14, 14, 7, 0, 0, 7, 7, 14, 0, 0, 7~
## $ w75     <dbl> 0, 14, 7, 28, 7, 14, 21, 7, 7, 7, 7, 7, 14, 7, 0, 0, 0, 14, 0,~
## $ w76     <dbl> 0, 7, 21, 7, 7, 0, 21, 7, 0, 0, 14, 7, 7, 28, 0, 0, 0, 7, 7, 1~
## $ w77     <dbl> 0, 7, 7, 14, 7, 0, 14, 0, 0, 0, 7, 28, 7, 7, 0, 7, 7, 7, 7, 7,~
## $ w78     <dbl> 7, 0, 7, 7, 7, 7, 0, 7, 7, 7, 7, 0, 14, 21, 0, 7, 0, 0, 14, 14~
## $ w79     <dbl> 7, 0, 7, 7, 0, 7, 0, 0, 0, 0, 7, 0, 21, 7, 0, 14, 0, 14, 0, 7,~
## $ w80     <dbl> 7, 14, 7, 7, 21, 14, 7, 7, 0, 14, 14, 14, 7, 7, 14, 28, 14, 7,~
## $ w81     <dbl> 14, 14, 7, 7, 0, 14, 0, 7, 7, 7, 0, 0, 0, 21, 0, 0, 0, 28, 7, ~
## $ w82     <dbl> 14, 7, 14, 21, 14, 0, 0, 0, 7, 0, 0, 7, 0, 7, 0, 14, 7, 0, 7, ~
## $ w83     <dbl> 14, 14, 14, 7, 7, 0, 21, 14, 7, 7, 7, 7, 7, 14, 7, 14, 21, 21,~
## $ w84     <dbl> 21, 0, 35, 7, 7, 7, 7, 7, 0, 7, 14, 7, 0, 0, 0, 7, 7, 0, 0, 0,~
## $ w85     <dbl> 7, 7, 0, 7, 7, 7, 7, 7, 7, 14, 0, 7, 21, 21, 0, 14, 14, 14, 7,~
## $ w86     <dbl> 7, 14, 14, 0, 7, 14, 7, 7, 7, 14, 0, 7, 0, 14, 7, 0, 0, 14, 0,~
## $ w87     <dbl> 21, 14, 0, 0, 0, 0, 7, 7, 14, 14, 14, 7, 0, 0, 14, 7, 0, 7, 7,~
## $ w88     <dbl> 0, 7, 7, 7, 0, 0, 0, 0, 0, 0, 7, 14, 28, 0, 7, 14, 7, 14, 0, 1~
## $ w89     <dbl> 0, 21, 7, 0, 7, 0, 14, 0, 14, 14, 14, 7, 14, 7, 21, 14, 7, 0, ~
## $ w90     <dbl> 0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 7, 21, 0, 7, 21, 7, 14, 7, 0, 0,~
## $ w91     <dbl> 14, 14, 7, 14, 21, 7, 0, 35, 7, 0, 21, 7, 0, 0, 21, 7, 21, 7, ~
## $ w92     <dbl> 0, 0, 7, 0, 7, 0, 0, 7, 14, 7, 0, 21, 0, 14, 0, 7, 0, 0, 7, 21~
## $ w93     <dbl> 0, 14, 7, 7, 7, 7, 0, 14, 14, 7, 14, 7, 0, 0, 14, 0, 7, 14, 0,~
## $ w94     <dbl> 0, 0, 14, 7, 14, 7, 0, 7, 0, 7, 0, 14, 0, 7, 7, 14, 14, 0, 7, ~
## $ w95     <dbl> 7, 0, 14, 7, 14, 7, 14, 7, 0, 0, 7, 0, 7, 0, 14, 7, 0, 14, 14,~
## $ w96     <dbl> 14, 0, 14, 0, 0, 7, 7, 7, 7, 14, 7, 0, 14, 0, 7, 0, 7, 7, 28, ~
## $ w97     <dbl> 7, 7, 7, 7, 0, 14, 7, 0, 21, 28, 0, 7, 0, 0, 0, 7, 7, 0, 0, 7,~
## $ w98      <dbl> 14, 14, 14, 7, 7, 21, 7, 0, 21, 7, 7, 0, 0, 0, 21, 7, 7, 7, 7,~
## $ w99      <dbl> 0, 7, 14, 7, 7, 0, 0, 7, 7, 0, 0, 7, 0, 7, 21, 0, 28, 0, 14, 2~
## $ w100     <dbl> 14, 21, 0, 7, 7, 7, 7, 14, 7, 28, 0, 7, 14, 21, 7, 7, 0, 7, 7,~
```

```
write.csv(m, "data/bootstrap_weights_matrix.csv", row.names = FALSE)
```

References

Sitter, Randy Rudolf. 1992a. "A Resampling Procedure for Complex Survey Data." *Journal of the American Statistical Association* 87 (419): 755–65.

———. 1992b. "Comparing Three Bootstrap Methods for Survey Data." *Canadian Journal of Statistics* 20 (2): 135–54.