Stat 854 Project: Mirror-match Bootstrap

authors

April 14 2024

Experiments

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


```
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;
```

SAS output

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

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

```
# 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)
)

# 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)

# 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,~
## $ finalwt <dbl> 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,~
```

Extra step: rescaling *finalwt* variable

As can be seen from the code below, the sum of finalwt in each stratum is different from N_h. Therefore, we rescale the finalwt to make the sum of finalwt in each stratum to be equal to N_h, based on the R code provided by Professor Boudreau (here we sincerely thank him for helping us with the project and providing us with the code).

```
finalwt_h_list = c()
for (h in 1:16)
{
  sub_dataset = syc[syc$stratum == h, ]
  finalwt_h = sum(sub_dataset$finalwt)
  finalwt_h_list = c(finalwt_h_list, finalwt_h)
}

# Comparison
comparison_df <- data.frame(finalwt_h_sum = finalwt_h_list, N_h = N_h_list)
print(comparison_df)
```

```
##    finalwt_h_sum  N_h
## 1           2633 2724
## 2           3291 3192
## 3           4369 4107
## 4           3215 2705
## 5           3950 3504
## 6            329  376
## 7             78   56
## 8            528  528
## 9            624  624
## 10           520  520
## 11           672  672
```

```
## 12           540  384
## 13           744  744
## 14           847  847
## 15           824  824
## 16          1848 1848
```

```r
# we want the sum of finalwt in each stratum to be equal to N_h
# therefore, we rescale it
finalwt2 = c()
for (h in 1:16)
{
  finalwt_h = syc[syc$stratum == h, ]$finalwt
  N_h = N_h_list[h]
  finalwt_h_sum = finalwt_h_list[h]
  finalwt_h = finalwt_h * N_h / finalwt_h_sum
  finalwt2 = c(finalwt2, finalwt_h)
}

syc$finalwt = finalwt2
```

```r
# Make sure rescaling is complete
finalwt_h_list = c()
for (h in 1:16)
{
  sub_dataset = syc[syc$stratum == h, ]
  finalwt_h = sum(sub_dataset$finalwt)
  finalwt_h_list = c(finalwt_h_list, finalwt_h)
}

# Comparison
comparison_df <- data.frame(finalwt_h_sum = finalwt_h_list, N_h = N_h_list)
print(comparison_df)
```

```
##    finalwt_h_sum  N_h
## 1           2724 2724
## 2           3192 3192
## 3           4107 4107
## 4           2705 2705
## 5           3504 3504
## 6            376  376
## 7             56   56
## 8            528  528
## 9            624  624
## 10           520  520
## 11           672  672
## 12           384  384
## 13           744  744
## 14           847  847
## 15           824  824
## 16          1848 1848
```

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

For n'_h, we choose

$$n'_h = f_h \times n_h, \tag{1}$$

advised by author in the paper (Sitter 1992b), where $f_h = \frac{n_h}{N_h}$, 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 $\bar{y}$.

Selection for n'_7

However, in the actual implementation, we found that for stratum 7, the n'_7 computed using Equation (1) gave us a value that is less than 1, while the range of n'_h must be in $[1, n_h)$.

Therefore, we randomly select the n'_7 among the $\{1, 2\}$ (with equal probability) in each bootstrap independently. For the other 15 strata, the average value of n'_h/n_h computed using (1) is approx. 0.12. Hence, we choose the range $\{1, 2\}$ to approximate this proportion given n_7 is 7.

Other than stratum 7, the n'_h of all the other strata calculated using Equation (1) works well and is in the range $[1, n_h)$.

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, 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, 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 (h == 7)
  {
    n_h_prime = sample(c(1, 2), 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] 10.34035 20.68070 10.34035 10.34035  0.00000 20.68070  0.00000 20.68070
##   [9] 31.02106  0.00000 10.34035  0.00000 10.34035 20.68070  0.00000  0.00000
##  [17] 31.02106 10.34035 10.34035 31.02106 10.34035 10.34035  0.00000 20.68070
##  [25] 20.68070  0.00000 20.68070 20.68070 20.68070 31.02106 20.68070  0.00000
##  [33]  0.00000  0.00000  0.00000 10.34035 20.68070 20.68070 10.34035 10.34035
##  [41] 10.34035  0.00000 10.34035 10.34035 10.34035 20.68070 20.68070  0.00000
##  [49] 20.68070  0.00000 10.34035  0.00000 10.34035 31.02106  0.00000 10.34035
##  [57] 10.34035  0.00000  0.00000 22.56077 11.28038 11.28038 11.28038  0.00000
##  [65] 11.28038 22.56077 22.56077  0.00000 11.28038 22.56077 11.28038  0.00000
##  [73]  0.00000 11.28038  0.00000  0.00000  0.00000 11.28038  0.00000 22.56077
##  [81] 11.28038 11.28038 11.28038 11.28038 11.28038 11.28038 33.84115  0.00000
##  [89] 11.28038 11.28038  0.00000  0.00000  0.00000  0.00000 22.56077  0.00000
##  [97] 11.28038 11.28038  0.00000 22.56077  0.00000 22.56077  0.00000 11.28038
## [105] 33.84115  0.00000 22.56077  0.00000 11.28038 11.28038 11.28038 11.28038
## [113]  0.00000  0.00000  0.00000  0.00000 11.28038 11.28038 11.28038  0.00000
## [121]  0.00000 22.56077 11.28038  0.00000  0.00000 22.56077  0.00000  0.00000
## [129] 11.28038 11.28038  0.00000  0.00000 11.28038 11.28038 11.28038 11.28038
## [137] 11.28038 33.84115  0.00000  0.00000 11.28038  0.00000  0.00000 11.28038
## [145] 11.28038  0.00000 11.28038 11.28038 11.28038  0.00000  0.00000 11.28038
## [153] 14.10048 14.10048 39.48135 14.10048 39.48135 19.74067 39.48135 39.48135
## [161] 19.74067 28.20096 19.74067 14.10048  0.00000  0.00000 19.74067 28.20096
## [169] 14.10048 14.10048  0.00000 14.10048  0.00000 28.20096 14.10048 56.40192
## [177] 14.10048 14.10048  0.00000  0.00000  0.00000 14.10048 19.74067 28.20096
## [185] 28.20096  0.00000 59.22202  0.00000 14.10048 28.20096 39.48135 42.30144
## [193] 28.20096 19.74067 39.48135 19.74067 14.10048  0.00000  0.00000 14.10048
## [201] 14.10048  0.00000  0.00000 19.74067 14.10048 59.22202 19.74067 28.20096
## [209] 28.20096 14.10048  0.00000 28.20096  0.00000  0.00000 14.10048 19.74067
## [217] 14.10048  0.00000 19.74067 14.10048  0.00000 14.10048 19.74067 19.74067
## [225] 42.30144 14.10048  0.00000 39.48135 14.10048 28.20096 14.10048 19.74067
## [233]  0.00000 28.20096 14.10048 19.74067 39.48135  0.00000 59.22202 28.20096
## [241]  0.00000 19.74067  0.00000 39.48135  0.00000  0.00000 19.74067  0.00000
## [249]  0.00000 28.20096 31.02106  0.00000 10.34035  0.00000 20.68070 20.68070
## [257] 10.34035  0.00000  0.00000 10.34035 20.68070  0.00000  0.00000 20.68070
## [265]  0.00000 10.34035 20.68070 10.34035  0.00000 20.68070 20.68070  0.00000
## [273] 10.34035  0.00000  0.00000 10.34035 20.68070 20.68070  0.00000  0.00000
## [281] 20.68070 10.34035  0.00000 31.02106 10.34035 20.68070  0.00000 10.34035
## [289] 10.34035  0.00000  0.00000 10.34035 31.02106  0.00000  0.00000  0.00000
## [297] 10.34035  0.00000  0.00000  0.00000  0.00000 10.34035  0.00000  0.00000
## [305]  0.00000 20.68070 10.34035 20.68070 20.68070  0.00000  0.00000 10.34035
## [313]  0.00000  0.00000 62.04211 10.34035 10.34035  0.00000 10.34035 10.34035
## [321] 31.02106 10.34035 10.34035 10.34035 20.68070 10.34035 41.36141 20.68070
## [329] 10.34035 10.34035
```

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] 7.24 7.24 7.24 14.48 7.24 ...
```

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))
   bootstrap_weights_matrix = subset(bootstrap_weights_matrix, select = -stratum)
  return(bootstrap_weights_matrix)
}

m <- mirror_match_bootstrap_B(syc, N_h_list)
dplyr::glimpse(m)
```

```
## Rows: 2,621
## Columns: 100
## $ w1   <dbl> 0.000000, 0.000000, 14.483859, 0.000000, 28.967717, 0.000000, 0.0~
## $ w2   <dbl> 0.000000, 14.483859, 0.000000, 0.000000, 0.000000, 7.241929, 0.00~
## $ w3   <dbl> 0.000000, 7.241929, 0.000000, 14.483859, 14.483859, 21.725788, 0.~
## $ w4   <dbl> 7.241929, 0.000000, 14.483859, 14.483859, 7.241929, 14.483859, 7.~
## $ w5   <dbl> 0.000000, 14.483859, 14.483859, 7.241929, 14.483859, 0.000000, 0.~
## $ w6   <dbl> 7.241929, 0.000000, 7.241929, 0.000000, 7.241929, 7.241929, 7.241~
## $ w7   <dbl> 14.483859, 7.241929, 7.241929, 14.483859, 7.241929, 14.483859, 7.~
## $ w8   <dbl> 7.241929, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 0.000~
## $ w9   <dbl> 0.000000, 0.000000, 7.241929, 14.483859, 14.483859, 0.000000, 7.2~
## $ w10  <dbl> 7.241929, 7.241929, 7.241929, 14.483859, 14.483859, 0.000000, 0.0~
## $ w11  <dbl> 7.241929, 14.483859, 7.241929, 14.483859, 7.241929, 7.241929, 14.~
## $ w12  <dbl> 14.483859, 7.241929, 7.241929, 7.241929, 7.241929, 0.000000, 0.00~
## $ w13  <dbl> 0.000000, 0.000000, 7.241929, 7.241929, 7.241929, 0.000000, 7.241~
## $ w14  <dbl> 7.241929, 0.000000, 7.241929, 7.241929, 14.483859, 21.725788, 7.2~
## $ w15  <dbl> 14.483859, 0.000000, 7.241929, 0.000000, 7.241929, 14.483859, 7.2~
## $ w16  <dbl> 7.241929, 7.241929, 0.000000, 7.241929, 7.241929, 14.483859, 7.24~
## $ w17  <dbl> 14.483859, 7.241929, 7.241929, 7.241929, 21.725788, 7.241929, 14.~
## $ w18  <dbl> 0.000000, 0.000000, 7.241929, 14.483859, 7.241929, 0.000000, 7.24~
## $ w19  <dbl> 7.241929, 14.483859, 7.241929, 21.725788, 0.000000, 7.241929, 0.0~
## $ w20  <dbl> 0.000000, 7.241929, 14.483859, 14.483859, 0.000000, 21.725788, 21~
## $ w21  <dbl> 0.000000, 7.241929, 0.000000, 0.000000, 0.000000, 0.000000, 0.000~
## $ w22  <dbl> 0.000000, 0.000000, 7.241929, 0.000000, 14.483859, 21.725788, 0.0~
## $ w23  <dbl> 7.241929, 0.000000, 7.241929, 0.000000, 14.483859, 21.725788, 7.2~
## $ w24  <dbl> 7.241929, 14.483859, 0.000000, 7.241929, 0.000000, 14.483859, 0.0~
## $ w25  <dbl> 0.000000, 7.241929, 21.725788, 28.967717, 14.483859, 7.241929, 0.~
## $ w26  <dbl> 14.483859, 0.000000, 14.483859, 7.241929, 14.483859, 14.483859, 1~
## $ w27  <dbl> 7.241929, 7.241929, 21.725788, 0.000000, 14.483859, 14.483859, 0.~
## $ w28  <dbl> 7.241929, 0.000000, 7.241929, 7.241929, 14.483859, 7.241929, 7.24~
```

```
## $ w29  <dbl> 14.483859, 7.241929, 7.241929, 7.241929, 0.000000, 0.000000, 14.4~
## $ w30  <dbl> 0.000000, 7.241929, 7.241929, 14.483859, 7.241929, 14.483859, 0.0~
## $ w31  <dbl> 21.725788, 7.241929, 7.241929, 7.241929, 14.483859, 0.000000, 0.0~
## $ w32  <dbl> 14.483859, 7.241929, 14.483859, 14.483859, 0.000000, 0.000000, 14~
## $ w33  <dbl> 21.725788, 7.241929, 14.483859, 7.241929, 7.241929, 0.000000, 7.2~
## $ w34  <dbl> 0.000000, 7.241929, 14.483859, 7.241929, 0.000000, 0.000000, 0.00~
## $ w35  <dbl> 21.725788, 7.241929, 0.000000, 14.483859, 7.241929, 0.000000, 14.~
## $ w36  <dbl> 7.241929, 0.000000, 7.241929, 0.000000, 0.000000, 7.241929, 0.000~
## $ w37  <dbl> 21.725788, 21.725788, 0.000000, 7.241929, 7.241929, 14.483859, 28~
## $ w38  <dbl> 0.000000, 7.241929, 7.241929, 21.725788, 0.000000, 0.000000, 21.7~
## $ w39  <dbl> 0.000000, 0.000000, 0.000000, 28.967717, 7.241929, 7.241929, 7.24~
## $ w40  <dbl> 7.241929, 7.241929, 0.000000, 7.241929, 0.000000, 21.725788, 7.24~
## $ w41  <dbl> 7.241929, 0.000000, 21.725788, 0.000000, 7.241929, 14.483859, 7.2~
## $ w42  <dbl> 0.000000, 7.241929, 0.000000, 7.241929, 14.483859, 7.241929, 7.24~
## $ w43  <dbl> 14.483859, 7.241929, 14.483859, 7.241929, 14.483859, 14.483859, 0~
## $ w44  <dbl> 0.000000, 7.241929, 14.483859, 7.241929, 7.241929, 7.241929, 7.24~
## $ w45  <dbl> 0.000000, 7.241929, 0.000000, 0.000000, 14.483859, 14.483859, 7.2~
## $ w46  <dbl> 0.000000, 0.000000, 7.241929, 0.000000, 7.241929, 0.000000, 0.000~
## $ w47  <dbl> 7.241929, 0.000000, 0.000000, 0.000000, 7.241929, 14.483859, 21.7~
## $ w48  <dbl> 7.241929, 14.483859, 7.241929, 7.241929, 7.241929, 7.241929, 0.00~
## $ w49  <dbl> 0.000000, 0.000000, 14.483859, 21.725788, 7.241929, 0.000000, 7.2~
## $ w50  <dbl> 0.000000, 0.000000, 21.725788, 7.241929, 7.241929, 0.000000, 7.24~
## $ w51  <dbl> 0.000000, 14.483859, 0.000000, 0.000000, 0.000000, 7.241929, 14.4~
## $ w52  <dbl> 7.241929, 0.000000, 0.000000, 21.725788, 0.000000, 7.241929, 14.4~
## $ w53  <dbl> 0.000000, 7.241929, 7.241929, 21.725788, 7.241929, 0.000000, 0.00~
## $ w54  <dbl> 7.241929, 0.000000, 14.483859, 0.000000, 0.000000, 0.000000, 7.24~
## $ w55  <dbl> 0.000000, 7.241929, 7.241929, 7.241929, 7.241929, 0.000000, 7.241~
## $ w56  <dbl> 7.241929, 7.241929, 0.000000, 7.241929, 21.725788, 7.241929, 7.24~
## $ w57  <dbl> 0.000000, 7.241929, 7.241929, 0.000000, 7.241929, 14.483859, 21.7~
## $ w58  <dbl> 0.000000, 21.725788, 0.000000, 14.483859, 0.000000, 0.000000, 0.0~
## $ w59  <dbl> 0.000000, 0.000000, 7.241929, 0.000000, 0.000000, 7.241929, 14.48~
## $ w60  <dbl> 7.241929, 0.000000, 0.000000, 0.000000, 0.000000, 0.000000, 7.241~
## $ w61  <dbl> 7.241929, 14.483859, 0.000000, 7.241929, 7.241929, 0.000000, 0.00~
## $ w62  <dbl> 7.241929, 7.241929, 7.241929, 0.000000, 7.241929, 14.483859, 7.24~
## $ w63  <dbl> 14.483859, 7.241929, 7.241929, 7.241929, 14.483859, 14.483859, 14~
## $ w64  <dbl> 14.483859, 7.241929, 14.483859, 7.241929, 14.483859, 0.000000, 7.~
## $ w65  <dbl> 21.725788, 0.000000, 0.000000, 14.483859, 7.241929, 21.725788, 14~
## $ w66  <dbl> 0.000000, 21.725788, 0.000000, 14.483859, 0.000000, 0.000000, 0.0~
## $ w67  <dbl> 7.241929, 7.241929, 0.000000, 7.241929, 0.000000, 7.241929, 7.241~
## $ w68  <dbl> 7.241929, 0.000000, 0.000000, 0.000000, 21.725788, 7.241929, 7.24~
## $ w69  <dbl> 7.241929, 14.483859, 7.241929, 0.000000, 0.000000, 14.483859, 14.~
## $ w70  <dbl> 0.000000, 7.241929, 7.241929, 7.241929, 0.000000, 7.241929, 0.000~
## $ w71  <dbl> 21.725788, 7.241929, 0.000000, 7.241929, 7.241929, 14.483859, 14.~
## $ w72  <dbl> 0.000000, 14.483859, 0.000000, 7.241929, 14.483859, 0.000000, 14.~
## $ w73  <dbl> 7.241929, 7.241929, 7.241929, 0.000000, 7.241929, 7.241929, 14.48~
## $ w74  <dbl> 0.000000, 7.241929, 0.000000, 7.241929, 7.241929, 7.241929, 7.241~
## $ w75  <dbl> 7.241929, 7.241929, 0.000000, 0.000000, 14.483859, 0.000000, 14.4~
## $ w76  <dbl> 0.000000, 0.000000, 7.241929, 7.241929, 21.725788, 7.241929, 7.24~
## $ w77  <dbl> 0.000000, 14.483859, 0.000000, 14.483859, 0.000000, 28.967717, 7.~
## $ w78  <dbl> 14.483859, 0.000000, 14.483859, 14.483859, 0.000000, 21.725788, 0~
## $ w79  <dbl> 7.241929, 0.000000, 14.483859, 7.241929, 7.241929, 7.241929, 0.00~
```

```
## $ w80  <dbl> 21.725788, 14.483859, 7.241929, 0.000000, 0.000000, 7.241929, 7.2~
## $ w81  <dbl> 14.483859, 7.241929, 14.483859, 14.483859, 14.483859, 7.241929, 1~
## $ w82  <dbl> 21.725788, 0.000000, 7.241929, 7.241929, 0.000000, 0.000000, 7.24~
## $ w83  <dbl> 7.241929, 7.241929, 7.241929, 0.000000, 0.000000, 21.725788, 7.24~
## $ w84  <dbl> 0.000000, 0.000000, 0.000000, 21.725788, 7.241929, 14.483859, 0.0~
## $ w85  <dbl> 7.241929, 0.000000, 7.241929, 7.241929, 21.725788, 0.000000, 14.4~
## $ w86  <dbl> 0.000000, 7.241929, 21.725788, 21.725788, 14.483859, 0.000000, 0.~
## $ w87  <dbl> 0.000000, 0.000000, 14.483859, 0.000000, 7.241929, 7.241929, 0.00~
## $ w88  <dbl> 14.483859, 14.483859, 21.725788, 43.451576, 0.000000, 0.000000, 0~
## $ w89  <dbl> 7.241929, 0.000000, 7.241929, 7.241929, 0.000000, 14.483859, 0.00~
## $ w90  <dbl> 21.725788, 7.241929, 14.483859, 7.241929, 14.483859, 0.000000, 14~
## $ w91  <dbl> 21.725788, 0.000000, 7.241929, 0.000000, 7.241929, 7.241929, 14.4~
## $ w92  <dbl> 7.241929, 21.725788, 14.483859, 0.000000, 14.483859, 7.241929, 7.~
## $ w93  <dbl> 7.241929, 7.241929, 14.483859, 7.241929, 0.000000, 0.000000, 7.24~
## $ w94  <dbl> 21.725788, 7.241929, 7.241929, 7.241929, 7.241929, 28.967717, 7.2~
## $ w95  <dbl> 14.483859, 7.241929, 7.241929, 7.241929, 14.483859, 0.000000, 36.~
## $ w96  <dbl> 0.000000, 0.000000, 7.241929, 36.209647, 0.000000, 7.241929, 0.00~
## $ w97  <dbl> 7.241929, 7.241929, 0.000000, 14.483859, 7.241929, 21.725788, 7.2~
## $ w98  <dbl> 7.241929, 7.241929, 0.000000, 0.000000, 7.241929, 0.000000, 7.241~
## $ w99  <dbl> 14.483859, 7.241929, 7.241929, 0.000000, 0.000000, 0.000000, 14.4~
## $ w100 <dbl> 7.241929, 7.241929, 14.483859, 0.000000, 0.000000, 14.483859, 7.2~
```

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

Appendices

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.