

Methodological advancements on the use of administrative data in Official Statistics - User Manual

Natalie Shlomo, Sook Kim, University of Manchester

Dec 2022

Table of Contents

Project aims and objectives.....	2
Quality indicators.....	2
R-indicators.....	3
Using sample-based frame information as benchmark.....	3
Distance metrics with Standardization	3
Download and inspect the contents.....	4
1 Download.....	4
2 Downloaded contents explained	5
Example datasets.....	5
Folders and R scripts.....	5
Launch RStudio and get ready	6
1 Open the entire master folder in RStudio	6
2 Set custom path	6
3 Automatically create output folders	7
4 Install packages	7
RUNNING 2_Prep_Wtsample_Freq_Table.R.....	7
PREP PART 1: Generate a weighted sample data	8
PREP PART 2: Auxiliary file for R-indicators.....	10
RUNNING 3A_Distance_Metrics.R.....	11
RUNNING 3B_R-indicator.R	15
Q & A.....	19
How do I know where to customise the code to suit my needs?	19
How to use Starting path in multiple machines?	19
What are the commonly used commands?.....	19

How to free up memory space and speed up RStudio?	19
I get error messages when a pre-defined function is used.	19
How do I modify pre-defined functions?.....	20
Technical notes and programming strategies.....	21
Can I ignore Warning messages?.....	21
Troubleshooting	21
Unused argument error	21
I get errors when computing.....	21
I am experiencing slowness in computation.....	21
Error: cannot allocate vector of size xxxx.x Gb	22
What version of R is used?.....	22
References	22
Citation.....	22

This is the manual accompanying R code publicly available on GitHub for the project, *Methodological advancements on the use of administrative data in Official Statistics*, which is led by Professor Natalie Shlomo. As part of the research team, Sook Kim documented the manual.

Project aims and objectives

The Office for National Statistics (ONS) have strategic priorities on embedding and advancing the use of administrative data into their official statistics processes. Their immediate priority is the use of administrative data in the quality assurance of the 2021 census and the production of administrative-based population estimates (ABPEs). A more long-term priority is the Population Statistics Transformation Programme which will feed into a recommendation to Government due in 2023 on the future of census and population statistics. In particular, the objective is to create population characteristic estimates from administrative and integrated data sources.

This manual concerns one of the sub-projects related to the quality framework for a single administrative data source. Here, we restrict the scope to distance metrics and R-indicators.

Quality indicators

Sources of error for representation of administrative data are frame errors, selection errors and missing redundancy. We focus here on errors arising from coverage and representativity of administrative data, particularly when the data is streamed over time such as tax data for the

business register or migration statistics for population estimates. It is vital that statistical agencies have good quality indicators to ensure the fit of administrative data to the population and to identify those sub-groups that are missing or over-covered, especially when the administrative data is used to quality assure other data sources such as surveys or a census.

We develop new methodology for calculating a quality indicator to measure representativeness and coverage.

R-indicators

One such indicator is the R-indicator and its related partial R-indicators that were originally designed to assess the representativeness of responses from a survey and are particularly useful as an objective function in adaptive survey designs where data are collected over time (Schouten, et al. 2009, Schouten and Shlomo, 2009). The R-indicators measure the contrast between those who are missing and not missing in the data and identify those groups that are not represented in the data. We will investigate the usefulness of this framework to assess the representativeness and coverage of administrative data compared to a target population.

Using sample-based frame information as benchmark

Recent research by Bianchi, et al. 2019 adapts the R-indicator to the case where only population auxiliary information are available instead of sample-based frame information. We draw upon the approach by Bianchi, et al. 2019, and utilise sample-based auxiliary information.

Distance metrics with Standardization

We look at other quality indicators. These are essentially distance metrics, such as the indicator of dissimilarity (Duncan and Duncan, 1955; Agresti, 2013). We also draw upon Hellinger's distance (HL) and Kull-back-Leibler divergence (KL).

First, we define categories of categorical variable (or cross-classified) categorical variables by k , $k=1, 2, \dots, K$.

Let p_k be the proportion of individuals in k in the census (weighted survey count).

Let q_k be the proportion of individuals in k in the administrative data.

The Entropy is

$$-\sum_k p_k \log(p_k)$$

The formulae for three distance metrics are given below.

Distance metrics	Formula	Standardisation
Duncan	$\frac{1}{2} \sum_k p_k - q_k $	1 - Duncan
Hellinger's distance (HL)	$\frac{1}{\sqrt{2}} \sqrt{\sum_k (\sqrt{p_k} - \sqrt{q_k})^2}$	1 - HK
Kullback-Leibler divergence (KL)	$\sum_k p_k \log\left(\frac{p_k}{q_k}\right)$	1 - KL

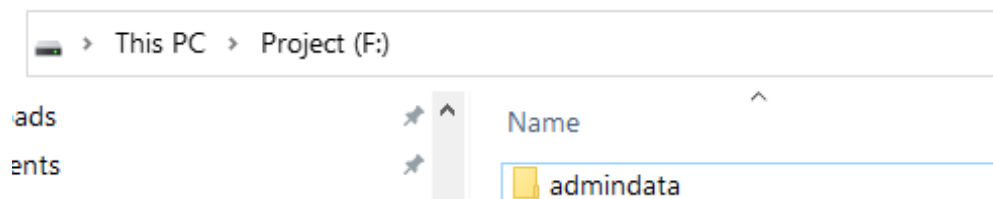
Download and inspect the contents

1 Download

Please visit the github site here: [qualadmin link](#). Click on Code at the top-right corner. Then, click on Download ZIP to download to your local machine.

Now, the downloaded folder needs to be placed in the meaningful location. We recommend users decide the appropriate Drive (C, D, E, F, etc) to house the downloaded contents. Then, **create a new folder** called admindata in File Explorer of your PC. Users can customise the new folder name as appropriate. This is your **starting path**.

The screenshot showing **starting path**:



Starting_path

Under this Starting path, F:/admindata, place the downloaded folder from GitHub. Extract the zip folder as necessary.

As such, F:/admindata/qualadmin becomes the MASTER project folder. We'll set it as working directory in RStudio later.

Notice that the terms, folder, directory, and path are used interchangeably in the user manual.

2 Downloaded contents explained












Example datasets

We provide two example data sources.

Data type	File name
Administrative	public_release_admin.csv
Census	pop_u_short_before_sim_5vars.Rdata

Folders and R scripts

Under F:\admindata\qualadmin folder, you'll be presented with the following contents.

-  Functions
-  User_manual
-  0_Custom_Path.R
-  1_Create_Folders.R
-  1_Install_Packages.R
-  2_Prep_Wtsample_Freq_Table.R
-  3_A_Distance_Metrics.R
-  3_B_R-indicator.R
-  3_MASTER_Run_AB.R
-  pop_u_short_before_sim_5vars.RData
-  public_release_admin.csv

The “**User_manual**” folder contains instructions on using the provided R code files.

The users do not need to do anything with the folder titled “**Functions**”. These pre-defined functions are used to either enclose complex procedures or perform repetitive tasks including cleaning and computing quality indicators. There are two files containing pre-defined functions. There is no need to run function files independently.

The functions will be automatically called in when the three main R script files are run:
2_Prep_Wtsample_Freq_Table.R 3_A_Distance_Metrics.R 3_B_R-indicator.R

The 2_Prep_Wtsample_Freq_Table.R file creates necessary data needed to compute distance metrics and R-indicators. The master file, 3_MASTER_Run_AB.R runs the above *two* main R script files, (3_A_Distance_Metrics.R 3_B_R-indicator.R) automatically in sequence.

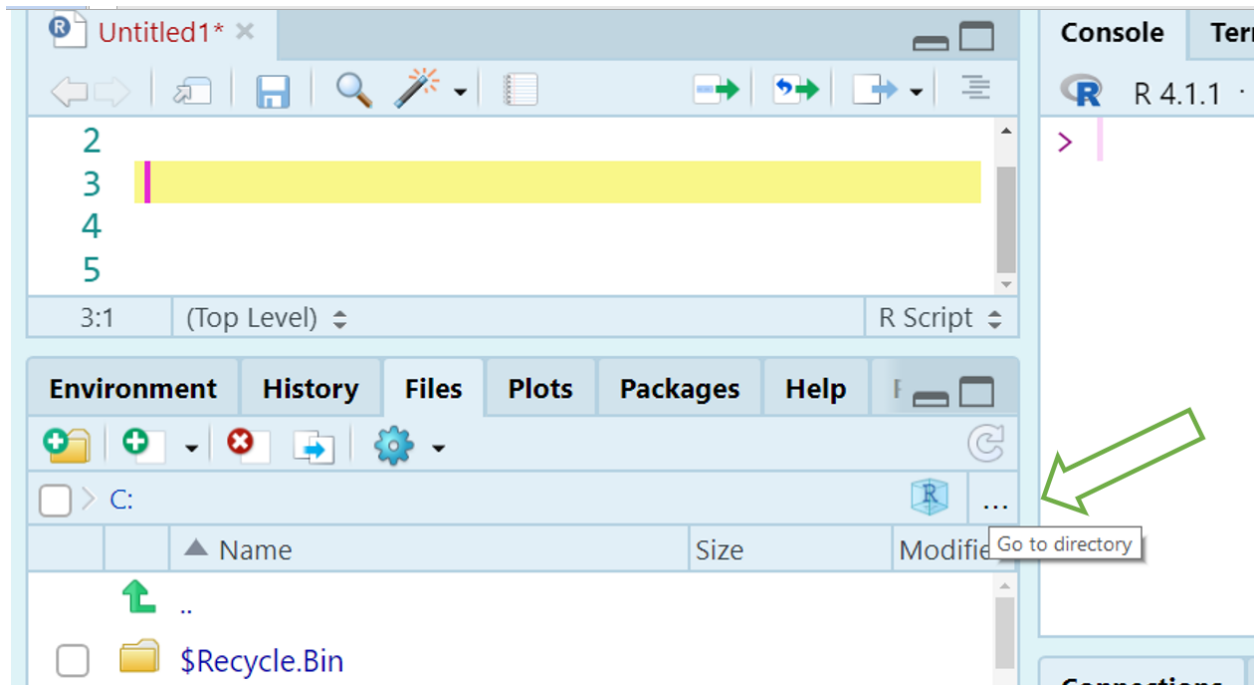
The first three files, 0_Custom_Path.R, 1_Create_Folders.R and 1_Install_Packages.R can be run to get ready to run the above main analysis files, as discussed in the following section.

Launch RStudio and get ready

1 Open the entire master folder in RStudio

First, launch RStudio. Then, we need to **open the entire folder** F:/admindata/qualadmin where downloaded materials are located.

Unfortunately, RStudio has no feature in the menu, but you could do so by accessing **Files** tab. Click on ... as shown below.



Then, locate the master folder. In our example, it is F:/admindata/qualadmin.

2 Set custom path

Click open the R script file, 0_Custom_Path.R. Customise the starting path as needed, and set the path to indicate the master folder. The example code is:

```
# Starting path (CUSTOMISE PLEASE)
setwd("F:/admindata")

# Master project folder (USE AS IT IS)
setwd("./qualadmin")

# Check your current directory
getwd()
```

Please ensure to use a single forward slash / as above. R will print an error when backward slash \ is used in path. For instance,

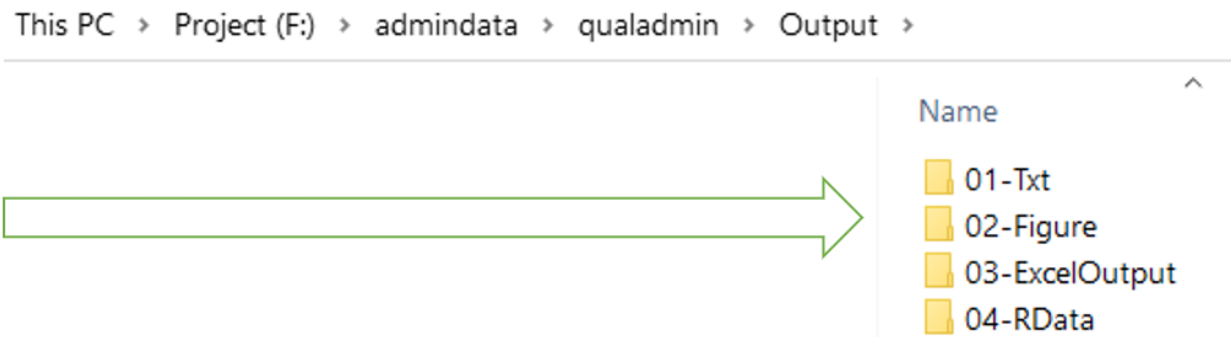
```
setwd("F:\\admindata`)
Error: '\a' is an unrecognized escape in character string starting
""F:\\a"
```

Please ensure your working directory is set at the master project path through the analytical steps.

3 Automatically create output folders

The three main R script files 2_Prep_Wtsample_Freq_Table.R, 3_A_Distance_Metrics.R, 3_B_R-indicator.R produce outputs. The outputs may be text, figure or in spreadsheet form. For the existing programmes to work, users need to create dedicated output folders.

To do so, please click on the 1_Create_Folders.R file to open. Then run line by line. The resulting folder structure is provided here:



Outputs_folder

4 Install packages

The final preparation step is installing packages. Open 1_Install_Packages.R file, and run line by line.

```
#-----
# Install packages (Run once)
#-----

install.packages("ggplot2")
install.packages("tidyverse")

install.packages("car")
```

Now, you're all set to proceed with quality measures indicators!

RUNNING 2_Prep_Wtsample_Freq_Table.R

This code file consists of two parts: generating a weighted sample data, and preparing an auxiliary file for R-indicators.

PREP PART 1: Generate a weighted sample data

- Open the 2_Prep_Wtsample_Freq_Table.R file.
- Step 1: Load a small percentage of Census data. It is called **pop_u_short_before_sim_5vars** in the provided example code.

```
#H-----  
##> 1. Load Census data  
#H-----  
  
load("pop_u_short_before_sim_5vars.RData")  
df <- pop_u_short_before_sim_5vars  
dim(df) # obs = 1163659  
names(df)
```

In our example Census data, we have 1,163,659 observations with five categorical variables including geography, sex, age groups, ethnic groups, and economic activity status. One can declare which variable to tabulate. Here, we declare all five variables using var object¹.

```
var <- c("geog1", "sex", "agecode1",  
        "eth_code5", "econg")
```

The description of categories, and distribution is shown below.

Variable	Category	Description	N	(%)
Total			1163659	
geog1	1	LA codes	116128	(10.0)
	2	LA codes	150139	(12.9)
	3	LA codes	137520	(11.8)
	4	LA codes	170624	(14.7)
	5	LA codes	90873	(7.8)
	6	LA codes	498375	(42.8)
sex	1	male	564905	(48.5)
	2	female	598754	(51.5)
agecode1	1	16-20	82426	(7.1)
	2	21-25	94643	(8.1)
	3	26-30	110296	(9.5)
	4	31-35	120398	(10.3)
	5	36-40	119393	(10.3)

¹ As the var object is treated as global macro, the programme runs automatically using the information stored in global macro, and produces the results.

Variable	Category	Description	N	(%)
eth_code5	6	14-45	101711	(8.7)
	7	46-50	94209	(8.1)
	8	51-55	100159	(8.6)
	9	56-60	77799	(6.7)
	10	61-65	65833	(5.7)
	11	66-70	57305	(4.9)
	12	71-75	51263	(4.4)
	13	76-80	43678	(3.8)
	14	81+	44546	(3.8)
	1	White	1081812	(93.0)
	2	Mixed/Multiple ethnic groups	10487	(0.9)
	3	Asian/Asian British	46446	(4.0)
	3	Black/African/Caribbean/Black British	16268	(1.4)
	4	Other ethnic group	8646	(0.7)
econg	1	In employment(FT, PT)	689140	(59.2)
	2	Unemployed	27744	(2.4)
	3	Out of workforce	446775	(38.4)

Using this prior information on the population distribution (based on Census), we can mimic the distribution in a random sample. See the next step.

- Step 2: Then, we draw a random sample 1:50.
- Step 3: From the randomly selected sample ($1163659/50 = 23273$), we then obtain frequency table of categorical variables (count of categories). Users can run the pre-defined function, `fn_maxvar5_freq_table()` to perform the task. The function² automatically obtains counts and structure the output in long form, organised by each variable, and by its discrete category.

```
fn_maxvar5_freq_table()
```

This procedure is to *assess distribution* of categories in a random sample ($N = 23273$). Based on the counts of the randomly selected sample, we multiply the counts by 50. One may wonder why we multiply. As we *reduced* the census sample by drawing a random sample by the 1:50 ratio, we need to *convert* the shrank sample back to the original size (with the priori distribution). That's why we multiply by 50 (Weighted Sample $N = 23273 * 50 = 1163650$). This completes the process of generating weighted sample survey data.

² We also provide `fn_maxvar4_freq_table()` for users who declare four categorical variables

- Step 4: Carry out checks to see if the calculated frequency tables are accurate.

```
freq_table[1:8, 1:9]

##   seq twdigits raw_n      n      p oneway v   by1 by2
## 1  1      101  2265 113250 0.09732308    1 1 geog1 01
## 2  2      102  2968 148400 0.12752976    1 1 geog1 02
## 3  3      103  2749 137450 0.11811971    1 1 geog1 03
## 4  4      104  3502 175100 0.15047480    1 1 geog1 04
## 5  5      105  1846  92300 0.07931938    1 1 geog1 05
## 6  6      106  9943 497150 0.42723327    1 1 geog1 06
## 7  7      201 11307 565350 0.48584196    1 2   sex 01
## 8  8      202 11966 598300 0.51415804    1 2   sex 02
```

When we printed the first 8 lines and 10 variables, we can see the count, n, and the corresponding proportion, p by each variable. The following code obtains the total observation size and confirms that the total proportion adds up to 1, for geog1 variable. Here, the total observation size can be viewed as the population size.

```
# Check whether the total adds up to 1
sum(freq_table[1:6, "n"])

## [1] 1163650

sum(freq_table[1:6, "p"])

## [1] 1
```

- Step 5: Rename and save.
- Step 6: Export the output frequency table in Excel with the file name, **Weightedsample_freq_table.xlsx**.
- Step 7: Save the R objects as RData. Done.

PREP PART 2: Auxiliary file for R-indicators

Thus far, we have created a weighted sample data based on the random sampling procedure, and obtained both one-way and two-way frequency tables. We also identified the population size of 1,116,350 (popsiz = 1163650).

For R-indicator calculations, we need to compute meanpop by variables which are geography, sex, age groups, ethnic groups, and economic activity status.

```
# Compute meanpop
auxiliary <- freq_table %>%
  filter(oneway == 1) %>%
  group_by(by1) %>%
  mutate(meanpop = n / popsiz) %>%
  ungroup() %>%
  dplyr::select(seq, count = n, by1,
    v, by2, meanpop, raw_n)
```

To do so, we first remove two-way and keep the one-way frequency table only. Then, by variable-level, which is indicated by `by1`, we compute `meanpop`. As seen before, the count of each category is stored in `n`. The `meanpop` is obtained by dividing `n` by `popsiz`. For example, the value of **meanpop** for first category of `geog1` is calculated as $113250/1163650 = 0.0973$, and the second category of `geog1` is $148400/1163650 = 0.1275$ and so on.

Finally, we add the `popsiz` in the first row, to complete the Auxiliary file.

Let's look at the Auxiliary file:

```
print(wtsample_auxiliary_econg)
```

##	seq	count	by1	v	by2	meanpop	raw_n	type
## 1	1	1163650	total	0	00	1.000000000	0	wtsample
## 2	2	113250	geog1	1	01	0.097323078	2265	wtsample
## 3	3	148400	geog1	1	02	0.127529756	2968	wtsample
## 4	4	137450	geog1	1	03	0.118119710	2749	wtsample
## 5	5	175100	geog1	1	04	0.150474799	3502	wtsample
## 6	6	92300	geog1	1	05	0.079319383	1846	wtsample
## 7	7	497150	geog1	1	06	0.427233275	9943	wtsample
## 8	8	565350	sex	2	01	0.485841963	11307	wtsample
## 9	9	598300	sex	2	02	0.514158037	11966	wtsample
## 10	10	84600	agecode1	3	01	0.072702273	1692	wtsample
## 11	11	93300	agecode1	3	02	0.080178748	1866	wtsample
## 12	12	111200	agecode1	3	03	0.095561380	2224	wtsample
## 13	13	124250	agecode1	3	04	0.106776092	2485	wtsample
## 14	14	118200	agecode1	3	05	0.101576935	2364	wtsample
## 15	15	99950	agecode1	3	06	0.085893525	1999	wtsample
## 16	16	95800	agecode1	3	07	0.082327160	1916	wtsample
## 17	17	95600	agecode1	3	08	0.082155287	1912	wtsample
## 18	18	78450	agecode1	3	09	0.067417179	1569	wtsample
## 19	19	67600	agecode1	3	10	0.058093069	1352	wtsample
## 20	20	57950	agecode1	3	11	0.049800198	1159	wtsample
## 21	21	50650	agecode1	3	12	0.043526834	1013	wtsample
## 22	22	42250	agecode1	3	13	0.036308168	845	wtsample
## 23	23	43850	agecode1	3	14	0.037683152	877	wtsample
## 24	24	1083250	eth_code5	4	01	0.930907060	21665	wtsample
## 25	25	10450	eth_code5	4	02	0.008980364	209	wtsample
## 26	26	45600	eth_code5	4	03	0.039187041	912	wtsample
## 27	27	16300	eth_code5	4	04	0.014007648	326	wtsample
## 28	28	8050	eth_code5	4	05	0.006917888	161	wtsample
## 29	29	691300	econg	5	01	0.594078976	13826	wtsample
## 30	30	28250	econg	5	02	0.024277059	565	wtsample
## 31	31	444100	econg	5	03	0.381643965	8882	wtsample

RUNNING 3A_Distance_Metrics.R

To calculate distance metrics, we first obtain one-way, and two-way frequency tables of categorical variables, and calculates proportions of each sub-category by each data source.

Next, we combine master (benchmark) and admin freq tables. Then, we create domains for quality indicators, and finally compute distance metrics as part of quality indicators. We offer three different types of distance metrics. To allow comparison across the metrics, we standardise the calculations.

Users can also produce a summary table of three types of distance metrics, and visualise the results.

The preliminary step is to ensure we have benchmark data. Simply *source* the previous file as below to update it.

```
source("2_Prep_Wtsample_Freq_Table.R")
```

- Step 1: read admin data

```
df <- read_csv("public_release_admin.csv")

## Rows: 1033664 Columns: 6
## -- Column specification -----
## Delimiter: ","
## dbl (6): person_id, geog1a, sex, agecode1, eth_code5, econg
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this
message.

tail(df)

## # A tibble: 6 x 6
##   person_id geog1a   sex agecode1 eth_code5 econg
##   <dbl>    <dbl> <dbl>    <dbl>    <dbl> <dbl>
## 1  1033659      1     1         6         1     1
## 2  1033660      6     1         6         1     1
## 3  1033661      6     2         5         1     1
## 4  1033662      5     1        12         1     3
## 5  1033663      1     2         9         1     1
## 6  1033664      6     2         9         1     1
```

The last six observations of the example admin data is shown above. As the person id goes up to 1033664, we can see there are 1033664 observations in admin data. We also notice that *geog1a* is used instead of *geog1*. As such, we declare all five variables for tabulations.

```
var <- c("geog1a", "sex", "agecode1",
        "eth_code5", "econg")
```

- Step 2: Obtain admin freq tables.

As mentioned in the previous section, we obtain frequency table of categorical variables (count of categories). Users can run the pre-defined function, `fn_maxvar5_freq_table()` to perform

the task. The function³ automatically obtains counts and structure the output in long form, organised by each variable, and by its discrete category.

Once the frequency tables are obtained, we rename the object as `Admin_f_table_one`. Let's inspect `Admin_f_table_one`.

```
head(Admin_f_table_one)

##      seq twodigits admin_n admin_perc oneway v      by1 by2      by3 by4 by5
## 1    1         101  137993  0.1334989      1 1 geog1a 01 oneway  0  0
## 2    2         102  124051  0.1200110      1 1 geog1a 02 oneway  0  0
## 3    3         103  131176  0.1269039      1 1 geog1a 03 oneway  0  0
## 4    4         104  139867  0.1353119      1 1 geog1a 04 oneway  0  0
## 5    5         105  142304  0.1376695      1 1 geog1a 05 oneway  0  0
## 6    6         106  358273  0.3466049      1 1 geog1a 06 oneway  0  0

tail(Admin_f_table_one)

##      seq twodigits admin_n  admin_perc oneway v      by1 by2      by3 by4 by5
## 340 340    404501    8557 0.0082783187      2 4 eth_code5 04 econg  5 01
## 341 341    404502     791 0.0007652390      2 4 eth_code5 04 econg  5 02
## 342 342    404503    5007 0.0048439338      2 4 eth_code5 04 econg  5 03
## 343 343    405501    3867 0.0037410609      2 4 eth_code5 05 econg  5 01
## 344 344    405502     255 0.0002466953      2 4 eth_code5 05 econg  5 02
## 345 345    405503    3420 0.0033086187      2 4 eth_code5 05 econg  5 03
```

- Step 3: Merge admin + Weighted sample freq tables

```
fn_merge_one_admin_wtsample_f_table_temp()
```

The code above executes merging two data sources.

- Step 4: Create domains of quality indicators

```
fn_create_domain_temp()
```

To check the domains, we can use *Janitor* package's `tabyl` function⁴. The function creates 15 domains, including five single variables' domain, and ten bivariate domains.

```
display_domain %>% tabyl(fct_domain)

##      fct_domain  n    percent
##      geog1     6 0.017391304
##      sex       2 0.005797101
##      agecode1  14 0.040579710
```

³ We also provide `fn_maxvar4_freq_table()` for users who declare four categorical variables

⁴ This is essentially almost identical to `table(display_domain$fct_domain)`, but the approach `tabyl` produces percent by default

```
##          eth_code5  5 0.014492754
##          econg     3 0.008695652
##          geog1:sex 12 0.034782609
##          geog1:agecode1 84 0.243478261
##          geog1:eth_code5 30 0.086956522
##          geog1:econg 18 0.052173913
##          sex:agecode1 28 0.081159420
##          sex:eth_code5 10 0.028985507
##          sex:econg  6 0.017391304
##          agecode1:eth_code5 70 0.202898551
##          agecode1:econg 42 0.121739130
##          eth_code5:econg 15 0.043478261
```

- Step 5: Compute distance metrics

Run the functions to compute three types of distance metrics.

```
fn_unstd_distance_metrics_full()
fn_unstd_distance_metrics_tidy()
```

- Step 6: Standardise distance metrics
- Step 7: Reshape, then tidy

From wide form, the outputs have been reshaped to long form. Then, we keep standardised solutions. The results are as follows:

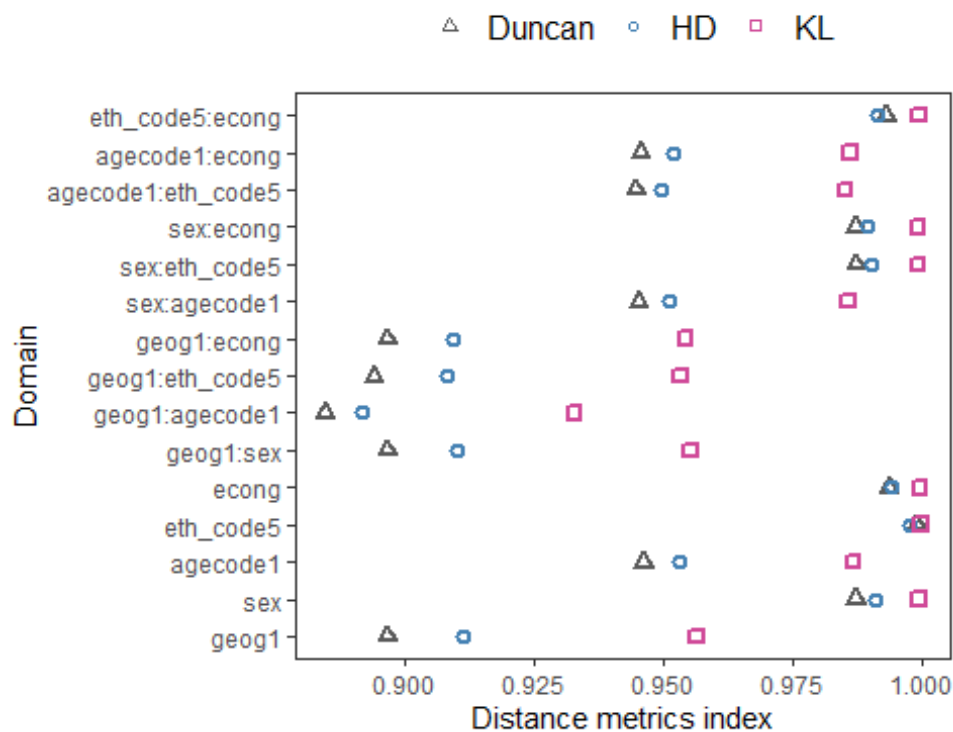
```
df <- distance_metrics_long
# std_test(1-Duncan, 1-HD, 1-KL) only
df <- df %>% filter(std_test_use == 1)

df[1:9, c(1:2, 4:5, 9)]

## # A tibble: 9 x 5
##   domain_id domain  indicator  index std_test_use
##   <int> <chr>    <chr>    <dbl>    <dbl>
## 1      1 geog1    Std_Duncan 0.897      1
## 2      1 geog1    Std_t_HD   0.911      1
## 3      1 geog1    Std_t_KL   0.957      1
## 4      2 sex      Std_Duncan 0.987      1
## 5      2 sex      Std_t_HD   0.991      1
## 6      2 sex      Std_t_KL   1.00      1
## 7      3 agecode1 Std_Duncan 0.946      1
## 8      3 agecode1 Std_t_HD   0.953      1
## 9      3 agecode1 Std_t_KL   0.987      1
```

- Step 8: Plot the distance metrics

```
plot(p)
```



RUNNING 3B_R-indicator.R

The file computes the overall R-indicator. Users can also proceed with computing **partial** R-indicators by category level, and variable level. The procedure can be computationally extensive. This is noted in the relevant section, so that users can allow some time to execute the code.

- Step 1: Benchmark mean population ready

Load auxiliary file and remove the last category From each categorical variable (`group_by(by1)`), we identify the last category (`max(by2)`). We then remove the last category(`dplyr::select(-c(lastcat_, lastcat))`). Here, we explicitly instruct R to use `dplyr` package to access `select` function⁵.

We only need `meanpop`. It is shown in a single column (column vector). Let's check.

```
# Print meanpop
print(col_auxiliary[1:nrow(col_auxiliary), "meanpop"],
      n = nrow(col_auxiliary))
```

⁵ This is to avoid warnings messages from R when R searches for a particular function from two different packages.

```
## # A tibble: 26 x 1
##   meanpop
##   <dbl>
## 1 1
## 2 0.0973
## 3 0.128
## 4 0.118
## 5 0.150
## 6 0.0793
## 7 0.486
## 8 0.0727
## 9 0.0802
## 10 0.0956
## 11 0.107
## 12 0.102
## 13 0.0859
## 14 0.0823
## 15 0.0822
## 16 0.0674
## 17 0.0581
## 18 0.0498
## 19 0.0435
## 20 0.0363
## 21 0.931
## 22 0.00898
## 23 0.0392
## 24 0.0140
## 25 0.594
## 26 0.0243
```

- Step 2: Keep wtsample distributions as row vectors

We then transpose meanpop to a single row format (row vector). Now, our benchmark data is ready. The next step is open the corresponding administrative data, and carry out computing R-indicators.

- Step 3: Compute R-indicators

Users can use pre-defined functions. Please allow a minute to execute.

```
aa <- read_csv("public_release_admin.csv")

nrow(aa) # 1033664

df <- NULL
between <- NULL
partial <- NULL
partialtemp <- NULL
fn_overall_r_indicator_1()
fn_overall_r_indicator_2()
```



```
fn_overall_r_indicator_3()
fn_overall_r_indicator_4()
```

Users can consult Functions/2_Functions_R-indicators.R file for more details and operationalisation.

- Step 4: Save in Excel and inspect

At this stage, users can inspect accordingly. Let's have a look. Here, we can see the overall R-indicator is estimated as 0.496 based on the administrative data (N=1033664). Looking at the variable-level R-indicator (see rows 4-8), geog1a was seen to have the greatest R-indicator (0.04) compared to econg (0.0002).

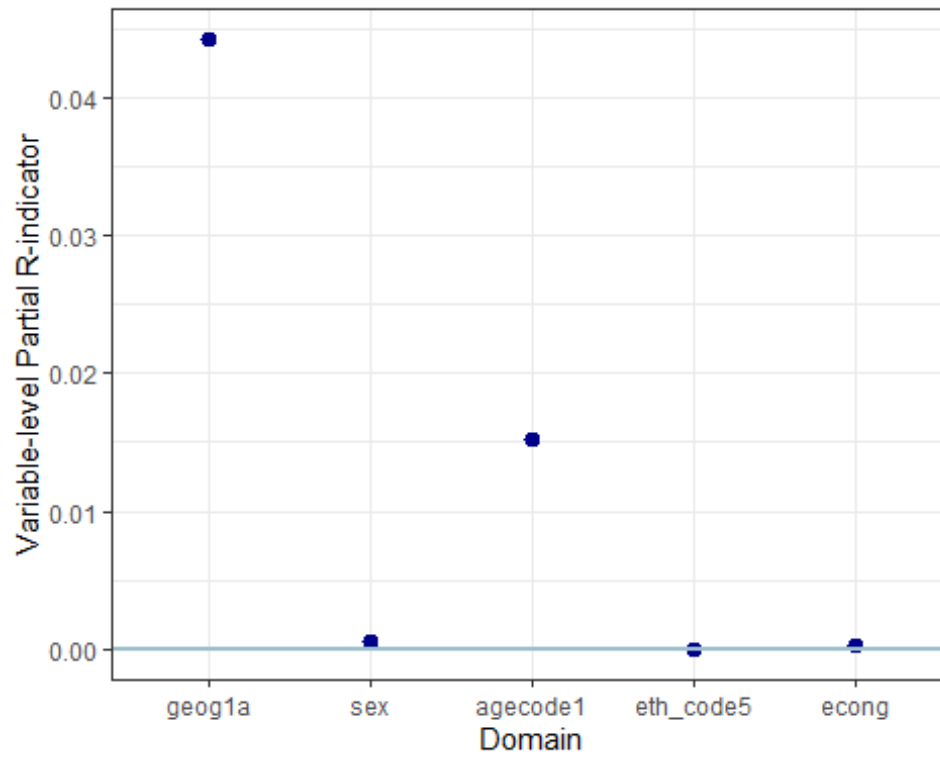
```
partial[1:17, c(1:2, 4, 8:10)]
```

##	seq	domain	R_indicator	count	n_cat	domain_n
## 1	1	Overall	0.49602631564	NA	<NA>	NA
## 2	2	mrphata11	0.95977686066	NA	<NA>	NA
## 3	3	resppop	1033664.00000000000	NA	<NA>	NA
## 4	4	geog1a	0.04424182738	NA	<NA>	NA
## 5	5	sex	0.00049830140	NA	<NA>	NA
## 6	6	agecode1	0.01528411959	NA	<NA>	NA
## 7	7	eth_code5	0.00002968059	NA	<NA>	NA
## 8	8	econg	0.00021170900	NA	<NA>	NA
## 9	9	des1	NA	0	1	0
## 10	10	geog1a_1	0.08795013527	137993	1	1
## 11	11	geog1a_2	-0.01927948285	124051	2	1
## 12	12	geog1a_3	0.02190390676	131176	3	1
## 13	13	geog1a_4	-0.03661612887	139867	4	1
## 14	14	geog1a_5	0.13969467338	142304	5	1
## 15	15	geog1a_6	-0.12165434160	358273	6	1
## 16	16	sex_1	-0.01620055057	489228	1	2
## 17	17	sex_2	0.01535719891	544436	2	2

- Step 5: Scatterplot

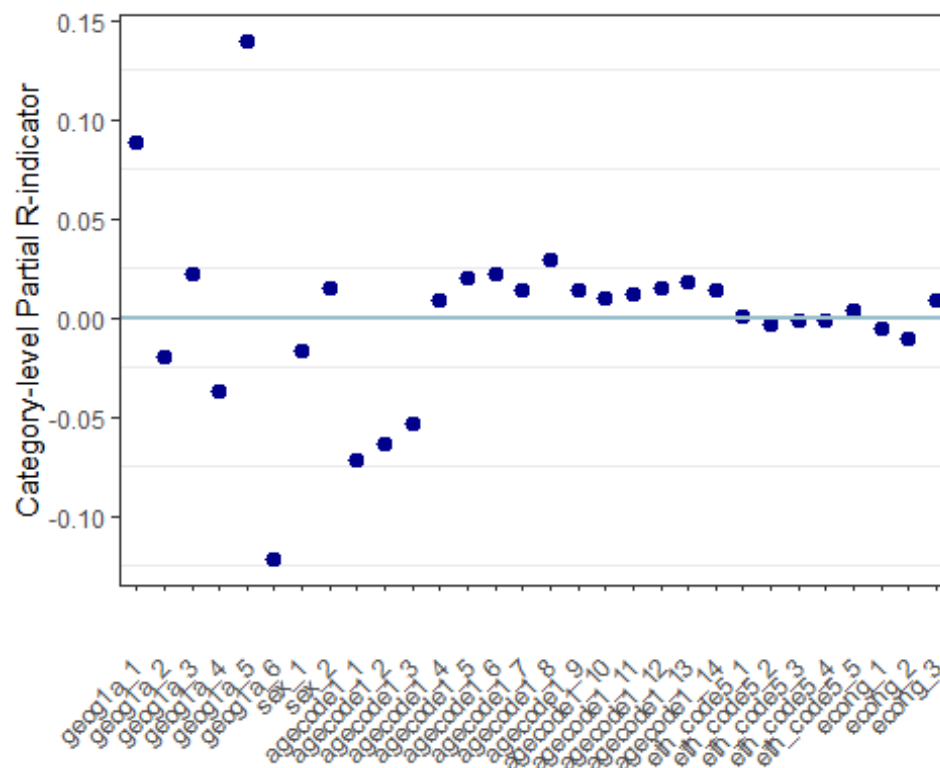
R-indicator by the variable level.

```
plot(p1)
```



R-indicator by the category-level.

```
plot(p2)
```



This concludes the manual. Thank you for taking the time reading the material. Please get in touch with any query or errata at fanfurcada@gmail.com.

If you need technical support, please consult Q & A.

Q & A

How do I know where to customise the code to suit my needs?

Unless indicated as “Customise as needed”, users can run the code as it is. Please consult each code file.

How to use Starting path in multiple machines?

If users plan to use different machines, simply by changing the “starting path”, users can carry out the analysis with minimal disruption. To achieve this, please ensure to use the consistent master project folder name.

What are the commonly used commands?

Most commonly used commands in the tidyverse package are:

```
arrange : sort variables.
bind_rows: append multiple dataframes.
mutate   : manipulate variables, and
           create new variables based on old variables.
select   : order, and keep(drop) variables of interest.
shell.exec: launch a software and opens the target file (Windows PC only)
```

How to free up memory space and speed up RStudio?

You can remove objects that you no longer need.

```
# To remove objects except for certain objects
ls()
keepobjectlist <- c("a", "b", "c")
rm(list = ls()[!ls() %in% keepobjectlist])
ls()
```

I get error messages when a pre-defined function is used.

Users can inspect the codes used in the function, and identify the issues. It is recommended NOT edit the function file directly, as the functions are used repeatedly, and the interlinked sections may not run as expected. Where preferable, users may copy the codes in the function, and use locally with minor tweaks.

How do I modify pre-defined functions?

Users can modify 1_Functions and 2_Functions_R-indicators.R under *Functions* folder.

```
# 1_Functions.R
fn_output_folder_path <- function() {

  currentdate <- Sys.Date()
  txtpath <- "Output/01-Txt/"
  figpath <- "Output/02-Figure/"
  xlsxpath <- "./Output/03-ExcelOutput/"
  Rdatapath <- "Output/04-RData/"
}
```

We can check how the output folder names are set as path to save the results during the analytical process.

```
fn_output_folder_path()
```

Let's run the function. We can see that xlsxpath is set as `"./Output/03-ExcelOutput/"`.

```
xlsxpath
## [1] "./Output/03-ExcelOutput/"
```

Let's customise the xlsxpath, by renaming the folder name. If we customise 1_Functions.R file, we can edit the information enclosed in the brackets. Notice that we use `<-` with functions so that the object created by a function will exist in the global R environment. This is very important.

Alternatively, We could ignore the pre-defined function and just write relevant lines of code and keep it in the main R script file. For instance, we could put `output_folder_path` at the top of the 2_Prep_Wtsample_Freq_Table.R. Here, we edited the xlsxpath. Notice that `fn_output_folder_path <- function() { }` is removed.

```
xlsxpath_2 <- "./Output/03-Excel/"

xlsxpath_2
## [1] "./Output/03-Excel/"

#H-----
## > Step 1. Load Census data
#H-----
# Load("pop_u_short_before_sim_5vars.RData")
```

Notice that we use `<-`. Using `<<-` is not necessary here. Users can remember the usage of `<-` and can modify the functions as appropriate, should the function incurs errors.

Technical notes and programming strategies

When loop is used, base R functions were used (table, tapply, etc). For data manipulation, tidyverse package was used extensively. This strategy is partly to improve readability of the code.

To enhance users' workflow, output files are programmed to launch using the pre-defined functions.

Can I ignore Warning messages?

Some packages alert users with compatibility issues arising from old version. These can be ignored. For example,

```
library("fastDummies")
Warning message:
package 'fastDummies' was built under
R version 4.1.2

library(rlist)
Warning message:
package 'rlist' was built under R version 4.1.2
```

Troubleshooting

Unused argument error

For example, `sim %>% select(geog1a)` the select command can cause an error:

```
Error in select(., geog1a) :
  unused arguments (geog1a)
```

This maybe due to the conflict in packages.

The error can be fixed by adding the name of the package used, dplyr, explicitly. `sim %>% dplyr::select(geog1a)`

I get errors when computing...

Please inspect zero cells, and ensure 0 (numeric value) is entered for n and perc, as well as admin_n and admin_perc. Errors may occur with NA coding and data attributes(character, factor, numeric).

I am experiencing slowness in computation.

R can be not responsive if memory is full. Please identify bottlenecks and remove them. It may be due to certain commands. For example, `View(object)` command could take a while if the object is huge in size. Unless one should inspect the data, suppress the View command to expedite the computation where possible.

It can also be the case that for loop functions can be slow as well. In some instances, removing objects may help as this procedure can free up memory space. See above commonly used commands for more information.

Error: cannot allocate vector of size xxxx.x Gb

If matrix symbols have entered mistakenly, R shows an error message like this. Please double check whether there are any mistakes. For instance, one may have typed `a*b` instead of `a%*%b`. Users can type `memory.limit()` to check the current memory limit and increase as necessary.

What version of R is used?

Tested with Windows PC. R version used: 4.1.1 RStudio version: RStudio 2022.07.2 Build 576

References

Agresti, Alan. Categorical Data Analysis. 2nd ed. Vol. 482. John Wiley & Sons, (2013).

Bianchi, Annamaria, Natalie Shlomo, Barry Schouten, Damião N. Da Silva, and Chris Skinner. 'Estimation of Response Propensities and Indicators of Representative Response Using Population-Level Information'. *Survey Methodology* 45, no. 2 (2019): 217–47.

Duncan, Otis Dudley, and Beverly Duncan. 'A Methodological Analysis of Segregation Indexes'. *American Sociological Review* 20, no. 2 (1955): 210–17. <https://doi.org/10.2307/2088328>.

R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

Shlomo, Natalie, Barry Schouten, and Vincent de Heij. 'Adaptive Survey Designs Using R-Indicators', 2009, 17.

Citation

Please cite this work as:

Shlomo, Natalie & Kim, Sook (2022). "Methodological advancements on the use of administrative data in Official Statistics - User Manual", available at <https://github.com/sook-tusk/qualadmin>