…recently did some pro bono work for Gun Control NZ reviewing the analysis by a market research firm of the survey that led to this media release: "Most New Zealanders back stronger gun laws". The analysis all checked out ok. The task at that time was to make sure that any claims about different perceptions of different groups in New Zealand society were backed by adequately severe (for the context) tests, and in the end I just performed a bunch of pragmatic Chi-square tests with the aggregate results provided by the original market research company.

However, my first thought when I saw that I had access only to the marginal aggregate totals, not the original microdata, was that it might be easier to analyse if I could re-create the original microdata. Amongst other things, this would have meant I could cycle through various bits of analysis with less code. Creating an adequate set of microdata was a harder job than I had hoped, so I didn't use that method for the original task. But since then I have been pecking away at it to see what it would take, partly for potential future use in this situation, partly as a general self-learning exercise in further understanding issues relating to statistical disclosure control (confidentialisation by cell suppression, random rounding, etc), which is more of a relevant technique than ever today.

See this story for an example claim that an extended version of the sort of methods I describe here, in combination with repeated queries of marginal totals that have been pertured for statistical disclosure control, could be used to recover original sensitive census data. Me, I'm sceptical of the practicality of the alleged "exploit" in that story, which is more theoretical than actual, but it's true that there is a potential vulnerability here that is worth dabbling my toes in to understand a little. It's certainly true that repeated queries of a tool that delivers marginal totals (ie crosstabs) can get around the more basic forms of statistical disclosure control perturbation, which is why national stats offices are investing heavily in methods of control such as fixed random rounding to mitigate that particular vulnerability.

But to be clear on this up front – I don't think a data snooper re-creating sensitive data is a realistic fear with regard to the sort of survey, with limited number of variables, such as that used for a demo in today's post. And my analysis below bears this out.

Example – Survey of New Zealand views on gun control

When tidied up, the available data from the survey I am working with looks like this, a typical set of high level results from a market research poll:

This chart shows one of two substantive questions in the survey, which is all I will be focusing on today. The visually prominent differences in response, such as by ethnicity, region, gender and rural or not location are statistically significant. The published data show cross tabulated counts by one question and one variable at a time (which I will be calling "marginal totals" in this post) but no interactions.

The code for making these nice diverging stacked bar charts for Likert-like data is shown a bit later down this post.

My task today is, it it possible to recreate a set of microdata that adds up to the aggregates in the available data? This is a special case of creating synthetic data, a common statistical disclosure control technique used to make microdata available for analysis that has the same properties as original microdata but without revealing sensitive details. Sometimes the data is analysed directly, sometimes it is used as a training set of data with which statisticians can develop their models safely before sending them off to a secure environment to be run on the real, gold standard data.

Of course, this isn't a new problem. A nice overview of some model-based methods is provided by Alan Lee in this discussion of creating SURFs (Simulated Unit Record Files) with New Zealand data. The `synthpop` R package was developed as part of the Synthetic Data Estimation for UK Longitudinal Studies project and seems to be the most advanced software in this area. However, the motivation of these efforts is to create a SURF that has plausibly been generated from the same underlying joint distribution of the population as the actual sample; I have given myself the slightly extended task of generating a SURF that has exactly the same joint distribution as the actual sample (not the population), at least to the degree of detail revealed by the published marginal totals.

Is this possible? There must be at least one solution, which is the actual sample (unless the aggregate totals have been deliberately fuzzed eg by random rounding or other perturbation for disclosure control). Can I find that solution? Is there more than one solution; if not, could this method be something a snooper could do to recover sensitive information? If there are multiple solutions, does finding one allow us to get extra insight from the data, for example by fitting a model that looks for effects from one variable while controlling for others, something that isn't possible with just the univariate aggregates? These are interesting questions.

It turns out that the answers are:

- It is possible to find a solution that very closely matches the published marginal totals, but very difficult to get an exact match
- The close solutions aren't unique
- The existence of multiple solutions means that a snooper is unlikely to recover sensitive information by this method
- Unless you combine considerably more variables than I have done in this instance, the process of re-synthesising microdata from the marginal totals does not yield any additional insights.

Data management

Import

First step as always is data management. This first chunk of code below downloads the data, tidies it and produces the first chart.

Post continues after R code

```
library(tidyverse)
library(survey)
library(tidyverse)
library(scales)
library(readxl)
library(janitor)
library(rmarkdown)
library(kableExtra)
library(knitr)
library(RColorBrewer)

download.file("https://github.com/ellisp/blog-source/raw/master/data/NZ%20gunlaw%20survey%202019%20Sep%
20supplementary%20tables.xlsx",
              destfile = "gun-law-survey-tables.xlsx", mode = "wb")

q1 <- "How strongly do you support or oppose strengthening New Zealand's existing gun laws?"


orig1 <- read_excel("gun-law-survey-tables.xlsx",
                    sheet = "Strengthening of exisiting laws",
                    skip = 1)

names(orig1)[1:2] <-  c("variable", "value")

clean <- function(x){
  x <- gsub("/", " ", x, fixed = TRUE)
  x <- gsub(" ", "_", str_squish(x), fixed = TRUE)
  x <- gsub("M.ori", "Maori", x)
  return(x)
}

freq1c <- orig1 %>%
  select(-`Total support`, -`Total oppose`) %>%
  fill(variable) %>%
  filter(!is.na(value)) %>%
  gather(answer, prop, -variable, -value, -n) %>%
  group_by(variable, value) %>%
  mutate(Freq = pmax(0.01, n * prop / 100)) %>%
  ungroup() %>%
  select(variable, value, answer, Freq)

ans_levs <- c("Strongly oppose", "Somewhat oppose", "Neither support or oppose",
              "Unsure", "Strongly support", "Somewhat support")

# Graphic
freq1c %>%
  filter(variable != "All") %>%
  mutate(answer = ordered(answer, levels = ans_levs)) %>%
  group_by(variable, value) %>%
  mutate(prop = Freq / sum(Freq)) %>%
  filter(!answer %in% c("Unsure", "Neither support or oppose")) %>%
```

```
  mutate(prop = ifelse(grepl("oppose", answer), -prop, prop)) %>%
  ungroup() %>%
  mutate(variable = gsub("_", " ", variable),
         value = fct_relevel(str_wrap(value, 30),
                             c("Other ethnicity", "$50k-$100k", "Under $50k",
                               "Canterbury", "Wellington/ Wairarapa", "Lower North Is.", "Upper
North Is.", "Auckland"),
                             after = Inf)) %>%
  ggplot(aes(weight = prop, x = value, fill = answer)) +
  facet_wrap(~variable, scales = "free_y") +
  geom_bar(position = "stack") +
  coord_flip() +
  scale_y_continuous(label = percent_format(accuracy = 1)) +
  scale_fill_manual(breaks = c("Strongly oppose", "Somewhat oppose",
                               "Somewhat support", "Strongly support"),
                    values = brewer.pal(4, "Spectral")[c(1,2,4,3)]) +
  labs(y = "'Unsure' and 'Neither' responses omitted from chart, but not from calculation of
percentages",
         x = "",
         fill ="",
         subtitle = q1,
         title = "Support and opposition to gun control in New Zealand")
```

Approach

We will be performing this resurrection of the microdata in three steps:

1. create a table of all possible combinations of each variable (income, age, ethnicity, possible answers to the question on gun control)
2. create a set of weights for each combination that add up to the observed marginal distributions (eg combinations of age with particular answers to the question), from which we can select samples that represent the same population we have inferred the original sample was drawn from
3. draw a sample from that population,
4. evaluate our sample's marginal totals against those of the original sample, and "improve" our simulated sample by dropping excess individuals and replacing them with new 'respondents' that give the sample propoertise that more closely resemble the original sample

The usual process of generating a SURF performs just steps 1 to 3, which turn out to be fairly straightforward with our relatively small number of variables. It is task four that looks to be the difficult one; a good thing too, or it would be too easy for snoopers to re-create microdata from aggregates.

BTW if people are wondering about my use of the term "snooper", this isn't me being whimsical; this is the term generally used in the statistical disclosure control literature for the adversary that we build our confidentialisation methods to guard against.

Dealing with some variables' idiosyncracies

When we look at counts of responses by each of the reported variables, straight away we have two interesting problems.

- The "ethnicity" variable, while reported in aggregate the same as variables such as region and income, is different because in the standard New Zealand classifications it is possible for a respondent to report two ethnicities. So we have more than 1,000 total responses by ethnicity despite the reported sample size being 1,000
- Other variables have less than 1,000 responses, almost certainly due to non-response; and the "unknown" categories for those variables are not provide.

variable	sum(Freq)
Age	1000.0000
All	1000.0000
Dependent children	1000.0000
Employment	1000.0000
Ethnicity	1098.8538
Gender	995.7287
Household income	857.4557
Living Situation	928.8421
Region	1000.0100
Rural	956.4756

… which was generated with this:

```
freq1c %>%
  group_by(variable) %>%
  summarise(sum(Freq))
```

It looks like age, dependent children, employment and region were all fully responded to (perhaps mandatory) whereas other variables had varying degrees of partial response, with income being the most non-answered question (unsurprising because of its sensitivity).

For income (and similar variables) we can recover the marginal totals of those for whom income is unknown by comparing the total responses for each level of the substantive question for "All" to those for whom we do have income information.

For ethnicity we need five yes/no variables for each of the possible ethnicities. When we generate our full "all combinations" population, we will eliminate those with more than two ethnicities, which helps keep size down to reasonable levels.

A bit of mucking around lets us deduce the values of those unknowns, and turn ethnicity into multiple variables.

```
#--------------Ethnicity variables and unknowns (income etc)--------------------
freq1c %>%
  group_by(variable) %>%
  summarise(sum(Freq)) %>%
  kable() %>%
  kable_styling() %>%
  write_clip()

freq1a <- freq1c %>%
  mutate(variable = ifelse(variable == "Ethnicity", value, variable),
         variable = clean(variable),
         value = gsub("$100,000k", "$100k", value, fixed = TRUE))

ethnicity_vars <- freq1a %>%
  filter(variable == clean(value)) %>%
  distinct(variable) %>%
  pull(variable)

totals <- freq1a %>%
  filter(variable == "All") %>%
  select(answer, answer_total = Freq)


unknowns <- freq1a %>%
  filter(variable != "All") %>%
  group_by(variable, answer) %>%
  summarise(variable_total = sum(Freq)) %>%
  left_join(totals, by = "answer") %>%
  mutate(Freq = round(answer_total - variable_total, 1),
```

```
        value = ifelse(variable %in% ethnicity_vars,
                       paste("Not", variable),
                       paste("Unknown", variable))) %>%
  ungroup() %>%
  select(variable, value, answer, Freq) %>%
  filter(Freq > 0)

freq1b <- freq1a %>%
  filter(variable != "All") %>%
  rbind(unknowns) %>%
  mutate(Freq = round(Freq)) %>%
  filter(Freq > 0)
```

Step 1 – creating all possible combinations

The first few rows of the object `freq1b` above look like this:

variable	value	answer	Freq
Region	Upper North Is.	Strongly support	99
Region	Auckland	Strongly support	155
Region	Wellington/ Wairarapa	Strongly support	69
Region	Lower North Is.	Strongly support	52
Region	Canterbury	Strongly support	72
Region	Other South Is.	Strongly support	53
Rural	Yes	Strongly support	50
Rural	No	Strongly support	436
Gender	Male	Strongly support	216
Gender	Female	Strongly support	283

Here is some code that takes that `freq1b` object and turns it into a big data frame of all the possible combinations of demographic variables and answers to question 1, hence representing the full population of New Zealand in scope for the survey (although not yet in the actual proportions of that population).

```
#--------------------Create population dataset of all possible combinations of
variables---------
poss_answers1 <- freq1b %>%
  distinct(answer) %>%
  rename(value = answer) %>%
  mutate(variable = "q1")

all_vars <- freq1b %>%
  distinct(variable, value)  %>%
  rbind(poss_answers1) %>%
  group_by(variable) %>%
  mutate(sequence = 1:n()) %>%
  ungroup()

all_vars_l <- with(all_vars, split(value, variable))

all_combos <- do.call(expand.grid, all_vars_l) %>%
  as_tibble() %>%
  mutate_all(as.character) %>%
  mutate(wt = 1) %>%
  filter(!(q1 == "Unsure" & Living_Situation == "Renting from Housing New Zealand or other
social housing organisation")) %>%
  filter(!(q1 == "Unsure" & Region == "Wellington/ Wairarapa")) %>%
  filter(!(Gender == "Unknown Gender" & q1 %in% c("Neither support or oppose", "Somewhat
oppose", "Unsure"))) %>%
  # remove people with more than 2 ethnicities, to save 1+ million impossible combinations:
  mutate(number_ethnicities = (Asian == "Asian") +
           (NZ_European_Other_European == "NZ European / Other European") +
           (NZ_Maori == "NZ Māori") +
           (Other_ethnicity == "Other ethnicity") +
           (Pasifika == "Pasifika")) %>%
  filter(number_ethnicities %in% 1:2) %>%
  select(-number_ethnicities)
```

Note that I have eliminated by hand a few combinations of variables that are implicitly non-existent in terms of our original sample, and also that the last steps in the chunk above are to cut down our combinations of variables to those that have only one or two ethnicities.

Step 2 – weights that resemble the marginal totals from the actual sample

So far so good. The next step is the one that gets most attention in the synthetic data literature; creating a model or set of weights that will allow our table of all possible combinations of variables to actually be representative of the in-scope population, as understood from the original sample. There are many ways to go about this, with and without explicit statistical models, and I have opted for the simplest which is to use iterative proportional fitting (or "raking") to create a set of weights for my all-combinations table that adds up to every one of the observed combinations of aggregate marginal totals. I use Thomas Lumley's `survey` package as a short cut here, noting the irony that I am creating weights for a population to match a sample, rather than (as is usually the case) for a sample to match a population. Sometimes this makes it hard to keep track of exactly what I mean by population in sample in the code below.

```
# Create a list with 13 different population tibbles, for each combination of a variable with
Q1
pops1 <- freq1b %>%
  group_split(variable) %>%
  lapply(., function(x){
    names(x)[2:3] <- c(unique(x$variable), "q1")
    x <- x[, -1]
    return(x)})

full_data <- all_combos

d1 <- svydesign(~1, weights = ~wt, data = full_data)

d2 <- rake(d1, sample = list(~Age + q1,
                             ~Asian + q1,
                             ~Dependent_children + q1,
                             ~Employment + q1,
                             ~Gender + q1,
                             ~Household_income + q1,
                             ~Living_Situation + q1,
                             ~NZ_European_Other_European + q1,
                             ~NZ_Maori + q1,
                             ~Other_ethnicity + q1,
                             ~Pasifika + q1,
                             ~Region + q1,
                             ~Rural + q1
),
```

```
population = list(pops1[[1]],
                  pops1[[2]],
                  pops1[[3]],
                  pops1[[4]],
                  pops1[[5]],
                  pops1[[6]],
                  pops1[[7]],
                  pops1[[8]],
                  pops1[[9]],
                  pops1[[10]],
                  pops1[[11]],
                  pops1[[12]],
                  pops1[[13]]))

full_data$wt <- weights(d2)
```

This only takes a few seconds to run on my laptop.

So we now have the `full_data` object, which has 2.5 million rows and 15 columns. The first 10 rows look like this:

Age	Asian	Dependent_children	Employment	Gender	Household_income	Living_Situation	NZ_European_Other_European	NZ_Maori	Other_ethnicity	Pasifika	q1	Region	Rural	wt
18-29	Not Asian	Yes	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	2e-07
30-44	Not Asian	Yes	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	3e-07
45-59	Not Asian	Yes	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	3e-07
60+	Not Asian	Yes	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	3e-07
18-29	Not Asian	No	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	6e-07
30-44	Not Asian	No	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	8e-07
45-59	Not Asian	No	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	9e-07
60+	Not Asian	No	30 hours or more	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	1e-06
18-29	Not Asian	Yes	Less than 30 hours	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	1e-07
30-44	Not Asian	Yes	Less than 30 hours	Male	Under $50k	Renting from a private landlord or property management company	Not NZ_European_Other_European	NZ Maori	Other ethnicity	Pasifika	Strongly support	Upper North Is.	Yes	1e-07

Within rounding errors, that final `wt` column adds up to the published survey sample totals. For example:

```
> full_data %>%
+   group_by(Gender, q1) %>%
+   summarise(full_data_total = sum(wt)) %>%
+   left_join(pops1[[5]]) %>%
+   rename(original_survey_total = Freq)
Joining, by = c("Gender", "q1")
# A tibble: 15 x 4
# Groups:   Gender [3]
   Gender         q1                        full_data_total original_survey_total
   <chr>          <chr>                               <dbl>                 <dbl>
 1 Female         Neither support or oppose           55                       55
 2 Female         Somewhat oppose                     38                       38
 3 Female         Somewhat support                   108.                     107
 4 Female         Strongly oppose                     21                       21
 5 Female         Strongly support                   282.                     283
 6 Female         Unsure                               9                        9
 7 Male           Neither support or oppose           63                       63
 8 Male           Somewhat oppose                     49.                      49
 9 Male           Somewhat support                    96.5                     96
10 Male           Strongly oppose                     51                       51
11 Male           Strongly support                   216.                     216
12 Male           Unsure                               8                        8
13 Unknown Gender Somewhat support                     1.00                     1
14 Unknown Gender Strongly oppose                      1                        1
15 Unknown Gender Strongly support                     2.00                     2
```

Step 3 – a sample from that population

So, a sample drawn at random from `full_data` with probability of selection proportionate to `wt` has a fighting chance of coming from the true population with the same joint distribution that the original sample came from, that is, the actual in-scope population of New Zealand. If we were just trying to create a SURF for analysts to use we would stop here, but we are interested not just in whether it is drawn from the same population, but how exactly it resembles the original sample. So in the code below I create the function `evaluate_dissim()`, which is going to get some heavy duty use later on. It compares the discrepencies in marginal totals between the new sample and the population totals it is meant to have, returning the sum of the absolute values.

```
# we can draw a sample from the set of all combinations
set.seed(877)
latest_sample <- sample_n(full_data, 1000, replace = TRUE, weight = wt)

#' Evaluate the dissimilarity of a current sample's marginal totals compared to
#' those in the various margin "population" totals
#'
#' @param latest_sample a sample to evaluate
#' @param pops a list of population data frames with three columns; first two are variables,
third is Freq
#' @details provides the sum of the absolute differences of the marginal totals from the sample
with those
#' in pops that it is trying to resemble
evaluate_dissim <- function(latest_sample, pops){
  total_discrepency <- 0
```

```
  for(j in 1:length(pops)){
    var_names <- names(pops[[j]])[1:2]
    x <- latest_sample[ , var_names] %>%
      group_by_all() %>%
      summarise(sample = n()) %>%
      left_join(pops[[j]], by = var_names) %>%
      mutate(discrepency = abs(sample - Freq)) %>%
      ungroup() %>%
      summarise(discrepency = sum(discrepency)) %>%
      pull (discrepency)

    total_discrepency <- total_discrepency + x
  }
  return(total_discrepency)

}

discrepencies <- evaluate_dissim(latest_sample, pops1)
```

Our starting sample is out from the marginal totals of the combined sample by a total of 1,546. For example, using gender again:

```
> latest_sample %>%
+   group_by(Gender, q1) %>%
+   summarise(latest_sample_total = n()) %>%
+   left_join(pops1[[5]]) %>%
+   rename(original_sample_total = Freq)
Joining, by = c("Gender", "q1")
# A tibble: 15 x 4
# Groups:   Gender [3]
   Gender         q1                        latest_sample_total original_sample_total
   <chr>          <chr>                                   <int>                <dbl>
 1 Female         Neither support or oppose                  64                   55
 2 Female         Somewhat oppose                            45                   38
 3 Female         Somewhat support                          102                  107
 4 Female         Strongly oppose                            19                   21
 5 Female         Strongly support                          265                  283
 6 Female         Unsure                                      7                    9
 7 Male           Neither support or oppose                  71                   63
 8 Male           Somewhat oppose                            44                   49
 9 Male           Somewhat support                          100                   96
10 Male           Strongly oppose                            29                   51
11 Male           Strongly support                          240                  216
12 Male           Unsure                                      8                    8
13 Unknown Gender Somewhat support                            2                    1
14 Unknown Gender Strongly oppose                             1                    1
15 Unknown Gender Strongly support                            3                    2
```

The new sample has similar results to the original – it's plausibly from the same population, which is all we've aimed for so far – but it's clearly not *exactly* the same. For example, my sample has 64 female respondents who neither support or oppose the gun control in question, whereas the original sample published a figure of 55. Close but not exact.

Step 4 – improving the simulated sample to make it resemble the original sample's marginal totals

The fourth step is the tricky stuff. I tried several ways to do the next step and am far from convinced I've got it right; in fact even for the approach I've ended up with, I have far from optimised code. But it's good enough for illustrative purposes.

My first idea was to find a row of data (representing a whole respondent) that was in "excess" with regard to at least one of its variables, remove that respondent from the sample and replace them with someone else chosen at random, with extra probability given to new respondents that would improve the marginal totals.

To do this, I defined a function called `improve_rnd()` which takes the latest sample and confronts it with a single marginal total, just like I do with gender in the table above.

- From that I identify types of respondents who are in excess (eg Female who Neither support or oppose) and types that are missing (Female who strongly support).
- I sort my sample with the rows that most represent excess values at the top (with some randomness given there will be lots of ties)
- I lop off the top row of that sorted sample (noting that this means removing a whole respondent ie also their values for income, age, region, ethnicity, etc)
- I create a table of under-represented rows from the population with regard to this particular variable, multiplying their original population weights by how badly they are under represented
- I sample one row from that under-represented set based on those new weights and add it to my new candidate sample
- I evaluate the candidate sample overall against *all* marginal totals to see if overall the change improves things or makes them worse (for example, I might have worsened the totals for income even though I improved them for gender)
- If the candidate sample is better, I return this; otherwise I repeat the last two steps, systematically going through my under-represented set of data until I find one that unambiguously improves the data

As you can probably guess, that second last point above proved to be important – before I did that, I ended up cycling around a modestly improved equilibirium where improving the marginal totals for any one variable was making it worse for the others.

This function then becomes the work horse for an iterative process where I work through all my variables (gender, income, age, etc) one at a time at random, looking to drop one of the rows in my sample that makes it worst for that variable and replace it with an improvement that doesn't degrade the other variables' totals. I repeat this until the total discrepency gets down to some acceptable level that is probably from rounding error (noting I never had the exact frequencies from the original data, only proportions and sub-population values of `n`).

Here's the code that does that procedure. This is the expensive part of the computation; it took me many hours to converge to a combined discrepency of less than 400, so in the code below I stop it once it gets below 1,200 – still much higher than I was hoping to get.

Post continues after R code

```
#' Improve a sample on basis of discrepency with a single set of marginal totals, while
#' not making it worse based on the other marginal totals it is matching
#'
#' @param latest_sample A sample that is trying to look like the original survey
#' @param marg_totals A single data frame of marginal totals with three columns, of which the
first
#' two must be variables from full data and the third, Freq, is counts we are trying to match
#' @param full_data A dataset of all possible combinations that the sample is drawn from,
#' with population weights
#' @param disc
#' @param strict if TRUE, whether to insist the end result sample has the same number of rows
#' as the original latest_sample
#' @param pops A full set of all marginal totals (as opposed to marg_totals which is just one
#' of them) - needed for evaluating how well candidate samples go compared to their aspiration.
#' @param verbose Whether or not to print out which row of the extra sample data is being tried
#' to replace an excess row in the sample
#' @param max_attempts the maximum number of candidate rows to try as a replacement for an
excess
#' row in the sample
improve_rnd <- function(latest_sample, marg_totals, full_data,
                  disc = 0.1, strict = FALSE, pops, verbose = TRUE,
                  max_attempts = 100){

  # initial discrepency:
  starting_disc <- evaluate_dissim(latest_sample, pops)
```

```
  # initial number of rows (useful for checking if we lose any later):
  starting_rows <- nrow(latest_sample)

  # variable names we are checking against for just this marginal total
  var_names <- names(marg_totals)[1:2]

  # in case we have any excesses recorded from previously, make this NULL
  latest_sample$excesses <- NULL

  # for convenience, renaming, and being able to go back if necessary, copy the current sample
data:
  x <- latest_sample
  names(x)[names(x) == var_names[1]] <- "var1"
  names(x)[names(x) == var_names[2]] <- "var2"

  # identify which combinations of the variables listed in marg_data are in latest_sample in
excess:
  new_excesses <- x %>%
    group_by(var1, var2) %>%
    summarise(sample_freq = n()) %>%
    full_join(marg_totals, by = c("var1" = var_names[1], "var2" = var_names[2])) %>%
    mutate(sample_freq = replace_na(sample_freq, 0)) %>%
    mutate(excesses = sample_freq - Freq) %>%
    select(var1, var2, excesses)

  names(new_excesses)[1:2] <- var_names

  y <- latest_sample %>%
    left_join(new_excesses, by = var_names)

  under_rep_vars <- new_excesses %>%
    ungroup() %>%
    filter(excesses < -disc)

  # Create and sort the under-represented rows of original data, with those that
  # are more under-represented more likely to be at the top
  under_rep_data <- full_data %>%
    inner_join(under_rep_vars, by = var_names) %>%
    mutate(wt = wt * abs(excesses),
           id = runif(n() * wt)) %>%
    arrange(desc(id)) %>%
    select(-id) %>%
    slice(1:max_attempts)

  if(nrow(under_rep_data) > 0){

    z <- y %>%
      # knock off one of the worst rows with too many excesses of some variable:
      arrange(desc(jitter(excesses))) %>%
      slice(-1)

    # cycle through all the possible candidates to replace a row of our sample with
    # one that is under represented, and choose the first one that reduces the overall
    # discrepency between our marginal totals and the ones we are aiming for:
    for(i in 1:nrow(under_rep_data)){
      if(verbose){cat(i)}

      candidate_sample <- z  %>%
        # replace it with a row from our under-represented set of data:
        rbind(under_rep_data[i, ])

      # evaluate our candidate.
      new_disc <- evaluate_dissim(candidate_sample, pops)

      # Have we made things worse for the other variables by trying to fix it for this one?:
      if(new_disc < starting_disc){
        # If things are better, we take this sample and break out of the loop
        latest_sample <- candidate_sample
        break()
      }
      # if things aren't better, we keep cycling through under_rep_data until we find one that
is
    }
  } else {
    # if there's nothing to replace:
    new_disc <- starting_disc
  }

  if(strict){
    if(nrow(latest_sample) != starting_rows){
      print(c(nrow(latest_sample), starting_rows))
      stop("Somehow gained or lost some rows")
    }
  }

  return(list(latest_sample,
              new_disc))

}

latest_sample %>%
  group_by(Gender, q1) %>%
  summarise(latest_sample_total = n()) %>%
  left_join(pops1[[5]]) %>%
  rename(original_sample_total = Freq)

discrepencies <- evaluate_dissim(latest_sample, pops1)
# best result from random version is 344
while(min(discrepencies) > 1200){

  # cycle through each of our sets of marginal totals (in random order),
  # looking to improve the match if we can
  for(i in sample(1:length(pops1))){
    y <- latest_sample %>%
```

```
      excess(marg_totals = pops1[[i]], full_data = full_data, disc = 0.2,
             pops = pops1, max_attempts = 5, verbose = FALSE)

    latest_sample <- y[[1]]
    discrepencies <- c(discrepencies, y[[2]])



  }

  # at the end of a cycle of variables, print how well we're going at improving things:
  print(min(discrepencies))

}
```

So it turned out to be a pretty inefficient solution to knock out a whole person at a time and grab a new one in the hope they would be better. In retrospect, this didn't make much sense as a method – I clearly came to it because I was thinking in terms of "whole people" as though my simulated sample really represented people and I had no choice but to accept and reject. I was thinking in these terms because I liked the idea of continually sampling from my hypothetical population, and felt this would somehow better mimic the known joint distribution. But I don't think that makes much sense now that I've tried it.

An alternative, perhaps an obvious one, is to look at my current sample, find the particular combinations of some variable that is over-represented (eg nine extra females who neither support or oppose), pick nine of them at random and make them male. Clearly not possible in the real world, but that's the whole point of synthetic data. I'd resisted this originally because I'd been thinking of all the other characteristics that are jointly distributed with being female that I was now changing, but on a bit of reflection this is actually little different from chucking those nine women out of the sample and keeping grabbing more people from the bag until I found some who were right.

So to implement this I built a new function, `improve_fix` which does just that. This was much quicker than my first, resampling-based method; but is still prone to being stuck. So in the implementation below I actually combine both methods: using the "swap people's characteristics" method as the main way of "improving" my sample, but if after 20 efforts of doing this no improvements are happening, trying my original random resampling method to get stuck out of bad equilibrium.

Post continues after R code.

```
#-----------------Alternative approach - brute force, one variable at a time----

#' @details This method changes the sample by seeking a way to flick over var1 to a new value
#' that will improve the marginal combinations of var1 and var2 while leaving all other
variables
#' unchanged. Unlike improve_rnd, which swaps out whole rows at a time.
improve_fix <- function(latest_sample, marg_totals, full_data, pops){

  # initial discrepency:
  starting_disc <- evaluate_dissim(latest_sample, pops)

  # variable names we are checking against for just this marginal total
  var_names <- names(marg_totals)[1:2]

  # in case we have any excesses recorded from previously, make this NULL
  latest_sample$excesses <- NULL
  latest_sample$id <- 1:nrow(latest_sample)

  # for convenience, renaming, and being able to go back if necessary, copy the current sample
data:
  x <- latest_sample
  names(x)[names(x) == var_names[1]] <- "var1"
  names(x)[names(x) == var_names[2]] <- "var2"

  # identify which combinations of the variables listed in marg_data are in latest_sample in
excess:
  changes_possible <- x %>%
    group_by(var1, var2) %>%
    summarise(sample_freq = n()) %>%
    full_join(marg_totals, by = c("var1" = var_names[1], "var2" = var_names[2])) %>%
    mutate(sample_freq = replace_na(sample_freq, 0)) %>%
    mutate(excesses = jitter(sample_freq - Freq)) %>%
    filter(round(abs(excesses)) > 0) %>%
    select(var1, var2, excesses) %>%
    ungroup()

  candidate_var2 <- changes_possible %>%
    group_by(var2) %>%
    filter(min(excesses) < 0 & max(excesses) > 0) %>%
    select(-excesses)

  changes_needed <- changes_possible %>%
    inner_join(candidate_var2, by = c("var1", "var2")) %>%
    arrange(desc(rnorm(n()))) %>%
    filter(var2 == var2[1]) %>%
    filter(excesses %in% range(excesses)) %>%
    mutate(excesses = round(excesses))

  names(changes_needed)[1:2] <- var_names

  number_changes <- min(abs(pull(changes_needed, excesses)))
  change_from <- filter(changes_needed, excesses > 0) %>% select(-excesses)
  change_to <- filter(changes_needed, excesses < 0)

  if(nrow(change_from) == 1 & nrow(change_to) == 1){
    knock_out <- latest_sample %>%
      inner_join(change_from, by = var_names) %>%
      sample_n(number_changes)

    replace_with <- knock_out
    replace_with[, var_names[1]] <- change_to[, var_names[1]]
    replace_with[, var_names[2]] <- change_to[, var_names[2]]

    latest_sample <- latest_sample %>%
      filter(!id %in% knock_out$id) %>%
      rbind(replace_with)

  }
   latest_sample$id <- NULL

  # new  discrepency:
  new_disc <- evaluate_dissim(latest_sample, pops)

  # Note that if change_to has zero rows then
  stopifnot(new_disc <= starting_disc)
```

```
    return(list(latest_sample,
                new_disc))

  }


  while(min(discrepencies) > 10){

    # cycle through each of our sets of marginal totals (in random order),
    # looking to improve the match if we can
    for(i in sample(1:length(pops1))){
      y <-  improve_fix(latest_sample,
                        marg_totals = pops1[[i]],
                        full_data = full_data,
                        pops = pops1)

      if(is.na(y[[2]])){stop("found an NA")}
      latest_sample <- y[[1]]
      discrepencies <- c(discrepencies, y[[2]])



    }

    # at the end of a cycle of variables, print how well we're going at improving things:
    print(min(discrepencies))

    # sometimes we get prematurely stuck with the "fixed" method and it is worth substituting
    # in and out some whole rows of data to kick-start the process:
    if(length(unique(tail(discrepencies, 20))) == 1){
      for(i in sample(1:length(pops1))){
        y <- latest_sample %>%
          improve_rnd(marg_totals = pops1[[i]], full_data = full_data, disc = 0.2,
                      pops = pops1, max_attempts = 5, verbose = FALSE)

        latest_sample <- y[[1]]
        discrepencies <- c(discrepencies, y[[2]])



      }
    }

  }
```

This method delivers good results in only a few minutes of processing time. Here's a plot of the decreasing total discrepencies:

You can see the first, slowish descent in total discrepencies in the first 100 or so samples, which represents my first, replacement-based method with `improve_rnd()`. Then from a total discrepency of about 1,200 to just about 250 there is a rapid improvement from using the `improve_fix()` method. That method stalls at a discrepency level of about 250 but a return to a cycle of `improve_rnd()` gets it out of that equilibrium; this pattern recurs a few times until we minimise our total discrepencies at 22. Importantly, we can get to that sample with discrepency of 22 from multiple ways.

So here's the comparison of just one variable and question 1 of the sample's marginal totals and the target, to compare with my first synthetic sample. Note that we now have nearly an exact match – we have just one too few females who "strongly support", and one too many of unknown gender who "somewhat support". My method as implemented doesn't have a clear way to improve the sample in this case. Intuitively, we would think we could find one of those two "unknown gender – somewhat support" people and change them to "female – strongly support" but I've written my program only to change one variable at a time (in this case Gender), to minimise complications with other sets of variables. So further improvement is still possible, if anyone really wants to pursue this approach

```
> #-----------------End results--------------
> latest_sample %>%
+   group_by(Gender, q1) %>%
+   summarise(latest_sample_total = n()) %>%
+   left_join(pops1[[5]]) %>%
+   rename(original_sample_total = Freq)
Joining, by = c("Gender", "q1")
# A tibble: 15 x 4
# Groups:   Gender [3]
   Gender         q1                        latest_sample_total original_sample_total

 1 Female         Neither support or oppose                  55                    55
 2 Female         Somewhat oppose                            38                    38
 3 Female         Somewhat support                          107                   107
 4 Female         Strongly oppose                            21                    21
 5 Female         Strongly support                          282                   283
 6 Female         Unsure                                      9                     9
 7 Male           Neither support or oppose                  63                    63
 8 Male           Somewhat oppose                            49                    49
 9 Male           Somewhat support                           96                    96
10 Male           Strongly oppose                            51                    51
11 Male           Strongly support                          216                   216
12 Male           Unsure                                      8                     8
13 Unknown Gender Somewhat support                            2                     1
14 Unknown Gender Strongly oppose                             1                     1
15 Unknown Gender Strongly support                            2                     2
```

Reflections

A few conclusions:

- This was considerably harder to do than I'd realised; and would get harder again with more sets of marginal totals to match.
- We would need several orders of magnitude more sets of marginal totals before we have to worry about there being a unique solution that gave a snooper (eg someone who knows the one Female Pasifika survey respondent in region X) confidence they had obtained new, sensitive information ie their answers to the survey.
- Because I have not included any external information (eg on the ethnic characteristics of different regions), I will have only very weak interaction relations between any variables – big underestimates of the true interactions. Considerably more information of that sort would be needed for this exercise to add any value in terms of generating insights beyond what the original marginal totals did.

Overall, at the end of 1,000+ line blog post, I would say my full method set out here is unlikely to be pragmatically helpful for any realistic use case that comes to mind for me. I think the traditional generation of synthetic data (my steps 1 to 3) is potentially useful for purpose of developing models that might then be applied to the gold standard original microdata in a secure setting; but step 4 really doesn't add much value either for a bona fide researcher or for a ill-intentioned snooper.

Acknowledgement and caveat

Gun Control NZ who commissioned the original survey shared the data and gave me permission to write a blog post on the data here but they have not seen the content in advance or been involved any other way; I am not affiliated with them and they are in no sense responsible for any of the content, findings, errors or omissions in the above.