Generating synthetic data with the synthpop package for R

Queen's University Belfast, 20th June 2018

Timetable

Time	Lectures	Practicals
1 – 2.30 pm	Introduction to synthetic data (GR) Introduction to synthpop (BN)	1 Simple synthesis and other synthpop functions
2.30 – 2.50 pm	Coffee break	
2.50 – 5 pm	Review of Practical 1 (GR) Further methods, tips and tricks. (GR, BN)	2 Further methods

You can find copies of all our slides as well as some sample code for the practicals on the web site https://www.geos.ed.ac.uk/homes/graab

Table of contents

Preparation for the practicals

1) Make a directory to hold the data from this course, perhaps call it *syncourse*
2) Install R (https://cloud.r-project.org/) and RStudio (https://www.rstudio.com/products/rstudio/download/) if not already installed on your laptop
2) Start RStudio
3) Install the synthpop package (tools menu install packages)
4) Open a new R script (file menu)

You should type the commands you are using in the script window and then save the script in your R course directory. To run commands you just do <Ctrl><Return> when your cursor is in the line you want to run, or highlight a whole set of lines to run.

Run the following commands to get started (text after # is a comment)

```
rm(list=ls())  # to clean out (remove) any items in the workspace

library(synthpop)  # to make the synthpop package available
setwd("D://sls//synthetic//syncourse")  # to set the course directory
                                        # change this to your own
```

A test data frame `SD2011` is available as part of the synthpop package. Go to its help page using command `?SD2011`, where you can find details about it. A list of the variables in it is also at the end of these notes. The examples in the help files for the synthpop functions use this data set.

Practical 1

If you are an R beginner see page 10 for some hints if needed.

TASK 1 Your first two syntheses

Select four variables from the `SD2011` data set where you might be interested in examining relationships.

YOUR CHOICE_____

Make a smaller data frame with just these variables. Do some plots or tables to understand them. Sample code might be:-

```
test1 <- SD2011[, c(1,2,4,23)]
# or test1 <- SD2011[, c("sex", "age", "placesize", "smoke")]
summary(test1)
ls()
```

`SD2011` is in the workspace for the synthpop package, but `test1` is now in your local workspace. NOTE HERE any features that might be a problem for synthesis

_____

Synthesise your data set using the `syn()` function, with default settings and compare the univariate distributions.

```
syn1 <- syn(test1)
compare(syn1, test1)
```

Does the compare output look OK? If no, why not? _____

The synthesised object (here `syn1`) is a list of different components.

```
names(syn1) # to find their names
syn1$call   # to examine a specific one
```

Use the help file for **`syn()`** to see what they are (look under Value for the components of the saved object).

Questions:

How many synthetic data sets were produced? _____

Which method was used for each variable? _____

What visit sequence was used? _____

Which variables were used as predictors for the variable synthesised at the very end? (examine `predictor.matrix`) _____

Now change `method` and/or `visit.sequence` parameter of the `syn()` function and make a second synthetic data set with code like this (**but with your own choices**).

```
syn2 <- syn(test1, visit.sequence = 4:1,
            method = c("sample", "norm", "ctree", "ctree"))
compare(syn2, test1)
```

Does the compare output look OK? If no, why not?

_____

TASK 2 Evaluating your synthetic data

First graphically: you have already compared the univariate distributions for the real and synthetic data. Now try examining the relationships between variables in your synthetic and real data. The function to use is `multi.compare()` (check the help file). There is a single outcome variable (parameter `var`) that can be categorical or continuous and one or more categorical variables (parameter `by`) to form groups to be compared.

For some pairs of variables in your data compare bivariate plots for both of your synthetic data sets using code like this:

```
multi.compare(syn1, test1, var = "placesize", by = c("sex"))
multi.compare(syn1, test1, var = "income", by = c("sex","placesize"),
              cont.type = "boxplot")
```

Summarise your results here for at least two comparisons:

1) Var_____ by _____

syn1 results comments:-

syn2 results comments:-

2) Var_____ by _____

syn1 results comments:-

syn2 results comments:-

3) Var_____ by _____

syn1 results comments:-

syn2 results comments:-

TASK 3 Formal utility calculations: Tabular utility

The function `utility.tab()` (check help) can be used to compare tables between real and synthetic data. You need to say which variables (you can have more than two) to cross-tabulate. Numeric variables will be grouped. Select two variables and use code like this to calculate the utility statistic.

```
utility.tab(syn1, test1, c("age", "sex"))
```

Try this for several pairs of variables and all 4 variables. Enter your results below.

Synthesis	Variables	Ratio to degrees of freedom	p- values	Comments on differences between tables
syn1				
syn2				
syn1				
syn2				
syn1				
syn2				
syn1	All 4 variables			
syn2				

The p-values, which tend to be oversensitive, are less relevant than the ratio. Ratios much greater than 3 or 4 should be investigated.

TASK 4 General utility calculations for the whole data set

The function `utility.gen()` (check help) implements a method based on propensity scores described in detail in a paper by Snoke at al. It also gives results as chi-squared tests with ratios to degrees of freedom. Try the method out and fill in some results. You can also try changing the parameters of `utility.gen()` if you have time.

Synthesis	Method used	Ratio to degrees of freedom	Important variables from the z scores (print.zscores = T)
syn1	Default		
syn2	Logit maxorder=1		
syn1			
syn2			

TASK 5 Fitting models to synthetic data

Fit a linear (`lm()`) or non linear (`glm()`) model to one of your variables from the other variables in `test1`.

To investigate which model is best, you can use `glm()` on the real data (`test1`) and on the synthetic data sets (`syn1$syn`, `syn2$syn`). Here is the sample code from a model developed from an analyses of `syn1$syn`

```
synfit2 <- glm(smoke ~ sex + placesize + poly(age,2), data = syn1$syn,
               family = "binomial") # predicting smoke including
                                    # a quadratic age term
```

This is what users of synthetic data would do. You can also use the special function `glm.synds()` providing `syn1` as an `data` argument. If you also have access to the real data you can then compare the fits from real and synthetic data models using function `compare()`.

```
synfit.synds <- glm.synds(smoke ~ sex + placesize + poly(age,2), data= syn1,
                          family = "binomial")
compare(synfit.synds, test1)
```

This will produce summary statistics for differences and a plot comparing the coefficients of the model fitted to the real and synthetic data.
How did your model work out on `syn1` and `syn2`?

TASK 6 Statistical disclosure control

The aim of this task is to get familiar with `sdc()` function (sdc stands for statistical disclosure control) that can be applied to a result of the `syn()` function and to explore other options to decrease disclosure risk using parameters of the `syn()` function. Consult help pages if necessary.

Apply `sdc()` function to your synthetic data object in order to:

- apply top and bottom coding to continuous variables (remember to exclude missing data codes from recoding process, if necessary),
- add a column with "false_data" flag to make sure that nobody treats synthetic data as real ones,
- remove unique synthetic individuals that replicate unique real people, if they exist. How many individuals were removed? ___ Try to exclude a variable from a set of variables used for identifying uniques. How many individuals were removed now? __

Rerun one of the previous synthesis but try to increase minbucket value for `cart` or `ctree` method (see details section in help for `syn()`) and/or apply smoothing during synthesis. Evaluate impact on synthetic data quality using tools of your choice.

Practical 2

TASK 1 Large data sets with many variables

Now to run code to synthesise whole `SD2011` data set, but first some tips.

You are brave and have a good machine, try the whole data set but be prepared for a crash.

If you are not brave make a middle sized data set. We suggest the following variables which give you an interesting selection. Use summary and consult the back page to see what they are. Or select your own.

```
SD2011mid <-  SD2011[,c(1,2,4:6,8:11,16:21,23:27,33:35)]
```

The "ctree" implementation of CART is faster than "cart" so use that one as the default.

Use summary(SD2011) or summary(SD2011mid) to examine your variables.

Are there any variables that are functions of others? (passive method), if so work out what would be a suitable passive method for them.

Make sure that you have set the **`cont.na`** parameters correctly and chose the method(s) you want.

To change the methods used, first run a synthesis and method = "ctree", with `m = 0`, so that no synthetic data are produced but you can get the other outputs from **`syn`**`()` (e.g. `method` and `visit.sequence`). Assign the object to `bigsyn0`.

To specify new methods you can use code like:

```
mymethod <- bigsyn0$method
```

and then change any elements of `mymethods` you want and then run `syn()` with `method = mymethod`

To time your function use `system.time(your command)` to check how much computing time it will take, and assign your result to an object `synbig`, for example

```
system.time(synbig <- syn(SD2011, method = mymethod, m = 0))
```

At this stage BEFORE YOU TRY TO RUN THE CODE, be sure to save your command file in case you get bombed out when your machine runs out of memory.

Now run the code with **`m=1`**. Did things slow down at some variables? Which ones? ____

Did you succeed in synthesising the whole data set? How long did it take? _____

If not then remove some variables until your machine can handle it, so you have a synthetic data object you can call `synbig`. How long did it take to make it?_____

Use commands like `utility.tab()` and `multi.compare()` to examine relationships between variables in different positions in the visit sequence. Note your results below

EXAMPLE 1 Variables_____ Position_____

How well was relationship maintained?

EXAMPLE 2 Variables_____ Position_____

How well was relationship maintained?

TASK 2 Stratified synthesis

Select one or two variables that you could use to stratify your synthesis. Try to avoid any with missing values.

Your choice_____

Carry out stratified synthesis with the function `syn.strata()` with these variables as strata. The help for `syn.strata()` is with the `syn()` function help.

How long did it take? _____

Evaluate the relationships between your stratification factors and other variables, again using multi.compare() and utility.tab().

EXAMPLE 1

Sratification variable _____ Other variable_____

RESULTS

EXAMPLE 2

Sratification variable _____ Other variable_____

RESULTS

TASK 3 Dealing with factors with many levels

There are several tactics you can try to reduce the time and memory required to synthesise these variables.

1. Exclude some variables from predictors for variables synthesised after them (change `predictor.matrix`),
2. Group variables into fewer categories,
3. Nest detailed classifications within larger ones,

Try to see if you can implement any of these. Sample code is available for you to look at in the files *sample_code_prac2.R* and *sample_code_prac2_mid.R,* which you can download from the web site.

SPACE FOR YOUR NOTES

TASK 4 Additional tasks

Well done to have got this far! If you have time you can choose your next task from the ones listed below (4a or 4b). Feel also free to explore other options available in the synthpop package.

4a New methods `ipf` and `catall`

For this task you will need to download the development version of synthpop (1.4-4) from the course web page, save the zip file and install the package from the local zip file.

Read the help files for `syn.ipf()` and `syn.catall()` and try them out. The sample code files have some examples.

4b I-CeM data set

Explore and synthesise a larger and more complex data set from the 1901 Census of Scotland (I-CeM data) that you can download from https://www.geos.ed.ac.uk/homes/graab

You should first read the codebook for the I-CeM data available from the same website, and also note any variables that might be problematic for synthesis.

For R beginners

The main R objects you will be using are data frames (like data sets or tables in other packages). Each variable in a data frame has a data type, e.g. numeric, character, factor. An R workspace can contain many data frames and other R objects.

Here are some useful commands to use with data frames

```
summary(mydataframe)
dim(mydataframe)
names(mydataframe)
head(mydataframe)
newdf <- mydataframe[1:100, c(1,3)]      # to make a new data frame with the
                                         # first 100 rows and columns 1 and 3
```

The other R object you need to know about is a list. An R list allows you to group a whole set of R objects together, where the objects can be quite different things. The output of the `syn()` function for carrying out synthesis is a list with many components, you can see the description of the components in the help file for `syn()` (`?syn`): see the Value entry at the bottom of the help page. If you need to see the names of the components only you can use the command `names(mylist)`. Individual elements of a list are accessed from their names using the command `mylist$myelement`. For example:

```
mysyn <- syn(SD2011[1:1000, c(1,3,4)])
mysyn$call               # gives you the command you used to make mysyn
mysyndata <- mysyn$syn   # this is the data frame of synthetic data
head(mysyndata)          # shows the first 6 rows of the new data frame
```

Here is a list of a few useful R commands you should know

Command	What it does	Command	What it does
ls	lists R objects	plot	makes 2 way plots according to data type
dim	dimensions of a data frame or matrix	with	uses data frame
class	returns the class of an R object	lm	fits linear models
head	first few lines of a data frame or entries in a vector	glm	fits generalised linear models
tail	last few lines of a data frame or entries in a vector	names	names of an R object
table	makes frequency tables (note it omits missings unless you specify - see help)	summary	summarises an R object; results depend on object class
hist	histograms	NA	R missing data code

Codebook for data frame SD2011

Column	Variable name	Data type (class)	N missing	details	Number of distinct values
1	sex	factor	0		2
2	age	numeric	0	In years range 16 to 97	79
3	agegr	factor	4	Six groups	6
4	placesize	factor	0	Size of city or rural area	6
5	region	factor	0	Place of residence region	16
6	edu	factor	7	Highest level of education	4
7	eduspec	factor	20	Specialisation of education	27
8	socprof	factor	33	Employment status	9
9	unempdur	numeric	0	Length of unemployment (-8 does not apply)	30
10	income	numeric	683	Note -8 means not applicable	406
11	marital	factor	9	Marital status	6
12	mmarr	numeric	1350	Month married	12
13	ymarr	numeric	1320	Year married	74
14	msepdiv	numeric	4300	Month separated	12
15	ysepdiv	numeric	4275	Year separated	50
16	ls	factor	8	Life satisfaction	7
17	depress	numeric	89	Depression score	22
18	trust	factor	37	Trust in other people	3
19	trustfam	factor	11	Do you trust family members	3
20	trustneigh	factor	11	Do you trust family mneighbours	3
21	sport	factor	41	Do you practice any sport?	2
22	nofriend	numeric	0	Number of friends (-8 means does not apply)	44
23	smoke	factor	10	Yes/no	2
24	nociga	numeric	0	Number of cigarettes smoked	30
25	alcabuse	factor	7	Abuse of alcohol – drank too much last year	2
26	alcsol	factor	82	Use alcohol as a solution to problems	2
27	workab	factor	438	Ever work abroad	2
28	wkabdur	character	0	Years worked abroad	33
29	wkabint	factor	36	Intention to work abroad	3
30	wkabintdur	factor	4697	Intended duration of work abroad	5
31	emcc	factor	4714	Country where you intend to work	17
32	englang	factor	15	English language	3
33	height	numeric	35	Height in cms	64
34	weight	numeric	53	Weight in Kgs	90
35	bmi	numeric	59	Body-mass-index weight(kilos)/ (height in metres)^2	1387