Description and Manual for spd

Software for Pseudonymization of Data

Author: Bernhard Lehnert

Date/Version: 2023-08-04

Contents

Name

`spd` is an abbreviation of „share pseudonymized data“ or „safely protect data“ or “secure personal data”.

Purpose

`spd` is a command line-based program aiming to provide researchers with an easy yet versatile way to produce high-quality pseudonyms for tabulated data e. g. in medical research. Often medical or otherwise sensible data is registered and there is no need to store the persons' real names in order to investigate the data or the data is to be shared with cooperating scientists without revealing true identities or a different set of pseudonyms is needed in order share different parts of the data for different projects.

We suppose the data is given in a tabular format such as in rows in an Excel sheet, a CSV file or some proprietary tabular format such as in Stata or SPSS.

Command Line Interface

As of now, `spd` does not provide a graphical user interface (GUI) but is used from the command line. This has a number of advantages, one of them being that each call of `spd` can easily be documented and thus reproduced (albeit with a random and thus different result). Scientists using the same setting repeatedly can store the call in a short shell script or in a R script or Stata do file or similar (cf table 1).

Table 1: Running command line tools like `spd` can be done from a variety of commonly used statistics software as well as obviously from any command line interface or from PowerShell.

Calling from	Function or command
Julia	run
Python	subprocess.run, os.system, …
R	base::system
SAS	CALL SYSTEM
Stata	winexec, shell, …

To call `spd` from the command line it needs to be either in a folder in the system PATH or in the current working directory it is called from. To call `spd` from the PowerShell it needs to either in the system PATH or it needs to be called as "`.\spd`". Placing commands in the system PATH is beyond the scope of this document (but well documented on many sources in the WWW).

Introductory example – a simple scratch list

Assuming you have put spd in the system PATH or navigated the command line interface to the folder containing spd you could simply type

```
spd
```

`spd` will then run, prompt the version number and write a file called `pseudonyms.txt` which will roughly look like this:

```
id_XiEG_A
id_WTNR_A
id_MWK6_A
id_3ENW_A
id_P76R_A
```

It could be opened in an editor and copy&pasted into an Excel column or it could be opened in a statistics software using CSV reading functions or simply be printed on paper to make a scratchlist of pseudonyms. Each time `spd` is run it tries to delete pseudonym.txt to avoid old data fram being mistaken as new data.

The user has a number of options to change that output, which will be described in this document. The means to change `spd` behavior is via so called flags. Running `spd` as

```
spd -v
```

Will print the list of pseudonyms on the command line stdout. `-v` is a command line flag to direct the output to the command line.

Prespecified number of pseudonyms

The number of pseudonyms to be created is set via the `-n` flag. To produce 50 pseudonyms type

```
spd -n 50
```

Or alternatively

```
spd -n=50
```

This can be combined with other flags as in

```
spd -n=50 -v
```

The construction of pseudonyms

Each pseudonym consists of three parts:

1) A prefix, in the above example “id_”. This is so that if taken out of context it is easy to determine that a code like `id_XiEG_A` is meant to be used as an identifier. It can be changed via the `-pre` flag.
2) A well-defined number of random letters taken from an alphabet. This is the random part which will be different each time spd is run. Its length is defined via the `-l` flag, the alphabet via the `-a` flag.
3) A postfix, in the above sample “_A”. When spd is called it will produce unique pseudonyms with no repetition. If called a second, maybe because more people were included in the sample, there is no way to guarantee that there are no two identical pseudonyms in the first and the second list. Changing the postfix from “_A” to “_B” or something else will guarantee unique pseudonyms.

Changing the prefix and the postfix

The prefix is determined via the -pre, the postfix via the -post flag. To produce pseudonyms with no pre- nor postfix enter

```
spd -v -pre="" -post=""
```

which could yield pseudonyms like 6XK3. Another example is

```
spd -v -pre="number." -post="" -n=10 -a=1
```

Changing the length of pseudonyms

The `-l` flag is used to determine the length of the random part of a pseudonym.

```
spd -v -n 3 -l 8
```

Will give you n = 3 pseudonyms like `id_X4Li988T_A`. Smaller l will make pseudonyms easier to read for humans, larger l will increase the chance of `spd` finding enough unique combinations for large samples. If `spd` does not find enough unique combinations, increase `-l` and/or choose a larger alphabet. In fact, if that happens `spd` will try using l+1 on its own.

Choosing an alphabet

There are five alphabets in spd that differ in the number of contained characters:

1) Alphabet 1 (`-a=1`) consists of digits 0 – 9 only, producing pseudonyms like id_428_A
2) Alphabet 2 (`-a=2`) consists of some letters and some digits chosen to prevent mistakes when written by hand. Neither the letter "O" not the digit zero are contained, neither "S" nor "5", neither "V" nor "U" so there is as little risk of human mistake when written by hand as possible whilst allowing for shorter pseudonyms then alphabet 1.
3) Alphabet 3 (`-a=3`) consists of small and large letters and digits thus featuring a lot more letters to choose from. It should be safe on any system using ASCII-codes.
4) Alphabet 4 (`-a=4`) consists of a really large number of runes, not necessarily meant for humans to read no safe when leaving utf-8 encoding. Choose for large samples or when humans are not meant to memorize pseudonyms.
5) Alphabet 5 is a spelling alphabet for evaluation. Hi likelihood of being changed in later versions.

Examples:

```
spd -v -a 1 -l 8   →   id_71268702_A
spd -v -a 2 -l 8   →   id_PRFR7M9P_A
spd -v -a 3 -l 8   →   id_df4MB2cE_A
spd -v -a 4 -l 8   →   id_Ψ⑨.↖TX>❻_A
```

For a full list of all items in all alphabets use the `-alphabets` flag:

```
spd -alphabets
```

Saving results as a single column and column header

`spd` writes the results to `pseudonyms.txt` or a filename specified with the `-out` flag. If the file already exists it is overwritten. Therefore, it is good practice to either specify a sensible filename or copy and process `pseudonyms.txt` immediately after building it.

```
spd -n 200 -out my_beloved_study.csv -post ""
```

CSV files come either with or without a header row. If the `spd` generated file is to be column binded to a CSV file with a header, it is sensible to give it a header as well. This is achieved via the `-head` flag

(the `-h` is reserved for help). If `-head=""` then no header is saved or printed out, otherwise `-h` will name the header:

```
> spd -v -out pseudos.csv -head new_id
new_id
id_MMWA_A
id_iHiB_A
id_CE88_A
id_7C86_A
id_F3P6_A
```

Glueing to existing CSV and long data formats

If a csv file is given with the `-in` flag, then that csv file is loaded and a pseudonym added to each line with the separator given via the `-sep` flag (usually comma or semicolon):

```
spd -in test.csv
```

```
spd -in test.csv -out result.csv
```

```
spd -in test.csv -sep=";"
```

In case of tab delimited data, a tab as delimiter can be written as `"tab"` or `"\t"`

```
spd -in test.csv -sep="tab"
```

Unique identifiers for CSV files in long format

If a CSV file is in long format, i.e., the same name or id appears in more then one line, that file can still be read by `spd`. Use the `-col` flag to specify the number of the column containing the names. If the names are in the first column:

```
spd -in test2.csv -col 1
```

Given that information `spd` will read only the specified column and produce an unambiguous pseudonym for each name. The result can then be joined/merged to the original data in your statistics program. This feature is rather new and needs further testing.

Complete list of all implemented flags

To get a help message including a complete list of implemented flags just type

```
spd -h
```

License, Legal Stuff, Error reports

spd is free software under the GNU GPL Version 3 license https://www.gnu.org/licenses/gpl.txt. The license can be displayed by calling spd with the `-legal` flag. Under the GPL you are encouraged to pass the software (together with its source code and the license) on to anyone wanting to try it.

This document, however, is copyrighted and all rights are reserved. You can distribute it but must not alter this document without the author's permission.

Translations of this document into other languages are very much encouraged by the author, as long as this document is passed on together with the translation and the name of the translator as well as

any changes to the text are marked in an obvious manner. Please inform the author of any translations you distribute via email to bernhard.lehnert@med.uni-greifswald.de

The same email address may be used for error reports.

Appendix 1: Example random scratch list made with spd

```
> spd -n 120 -a 2 -l 6 -pre "" -post ""
```

X9iGRG	ENPTC6	BBLMM6	Li8P3E
MR3R4K	i3NE3R	LEX6BR	F7XLKX
XGR3TW	H3NM3K	PFW4A4	3TiKT7
MWKRH7	9EEBB9	978LNT	GFMGKM
N394TT	KXWFTR	6LN9B6	3NRLPM
CLE3M3	PMR9CX	AF46W9	XEL8T4
CPWR4E	PGGLH7	3TNHXE	BL8RHN
3E3i4E	NRCE9W	TWTAPR	H74WKR
XHA6AE	X49PF7	PF6F64	TiLXNK
C7HAAW	3HMPT9	KBKFB4	MKL4Mi
XPGi7B	XXBCFG	4FCP99	XTP8TP
LA9ERH	3CWFTK	G47T3H	TGPTE4
P76NA7	T3EiMH	9K66Ni	CGPP3L
K8FB9L	C6FAC9	6T46EW	8A48G8
M7ET4i	PAiA8C	ECLRP6	P8BEiE
3W63FB	636TE7	E7BGL9	4HiB6B
iA6i3M	EAECR7	P7RCME	6iAR78
44N9GK	6GB4BC	EH8MRP	iFEG3N
i6RW66	THG8MG	i3WTNH	XN9NGN
GRA7W3	6TXTMF	WRMWP4	H8W6AG
F9GNWF	6TMNGW	3N6HEF	EFMH9P
744F9W	PMHKMi	RK3E9P	GCH39L
8F9PTP	9NHGNT	LL97G8	C68T49
MC76HC	LT9iC4	M7i3CR	i4FKX6
FEiEEK	XiXAPF	KMNLiN	B8NREi
WBTTPB	4FWP4R	AEP4CC	RGM9PT
KRRLR6	RFM6AG	388HLL	7XKCPC
G4RLP9	AM6PPP	E3BWCB	M4iRMi
46BF9E	iMi7MK	B6LLR7	Bi6KT4
GN9A9B	4XCE46	3K78B4	8FPXPH

Appendix 2: Alphabetic list of flags

-a int

Alphabet to be used, possible values: 1, 2, 3, 4 and 5 (default 2)

-alphabets

print all alphabets to console

-col int

define id column in "in" file (CSV) to guarantee only one pseudonym per id, zero if off

-head string

header string, no header if ""

-in string

filename of CSV to read in

-l int

length of pseudonyms minus prefix and postfix (default 4)

-legal

print license and author contact data

-n int

how many pseudonyms to produce (default 5)

-out string

filename to write pseudonyms to (default "pseudonyms.txt")

-post string

postfix for pseudonyms, e.g. to avoid abiguity (default "_A")

-pre string

prefix for pseudonyms (default "id_")

-sep string

separator in CSV files (default ",")

-v print pseudonyms to console (stdout)

-version

print spd version