Package: cellkeyperturbation (via r-universe)

October 10, 2026

Title Cell Key Perturbation

Version 3.0.0

Description Provides functions to generate frequency tables and apply
cell key perturbation to protect against statistical disclosure
in tabular outputs. The implemented methods are described in
``Cell Key Perturbation User Guide"
<https://github.com/ONSdigital/cell-key-perturbation-R/blob/main/documentation/SML_UserDoc_CKP_R.md>.
Developed at the UK Office for National Statistics.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Suggests bigrquery, DBI, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports data.table

Depends R (>= 3.5.0)

LazyData true

VignetteBuilder knitr

URL https://github.com/ONSdigital/cell-key-perturbation-R

BugReports https://github.com/ONSdigital/cell-key-perturbation-R/issues

Repository https://onsdigital.r-universe.dev

Date/Publication 2026-05-21 10:52:49 UTC

RemoteUrl https://github.com/onsdigital/cell-key-perturbation-r

RemoteRef HEAD

RemoteSha 2187cf12018e2f4fcd63d3def6d87f11601ff4bf

Contents

check_for_na . . . 2
create_perturbed_table . . . 2
create_perturbed_table_bigquery . . . 4
generate_ptable_10_5_rule . . . 6
generate_random_rkey . . . 7
generate_record_key_from_ons_id . . . 7
generate_test_data . . . 8
micro . . . 9
ptable_10_5 . . . 9
validate_inputs_bigquery . . . 10

Index **12**

`check_for_na` *Check perturbed table for missingness in tabulation variables*

Description

Check perturbed table for missingness in tabulation variables

Usage

```
check_for_na(DT, cols)
```

Arguments

`DT`	– `data.table` Perturbed frequency table
`cols`	– `character vector` Tabulation variables

Value

Warning message if any tabulation variable contain missing values

---

`create_perturbed_table` *Create a frequency table with cell key perturbation applied*

Description

`create_perturbed_table()` creates a frequency table which has had cell key perturbation applied to the counts. A p-table file needs to be supplied which determines which cells are perturbed. The data needs to contain a 'record key' variable which along with the ptable allows the process to be repeatable and consistent.

Usage

```
create_perturbed_table(
  data,
  ptable,
  geog,
  tab_vars,
  record_key,
  use_existing_ons_id = TRUE,
  threshold = 10
)
```

Arguments

data	A data.table containing the data to be tabulated and perturbed The data should contain one row per statistical unit (person, household, business or other) and one column per variable (age, sex, health status)
ptable	A data.table containing the ptable file which determines when perturbation is applied.
geog	A character vector giving the column name in data that contains the desired geography level for the frequency table. This can be an empty vector, c(), if no geography level is required.
tab_vars	A character vector giving the column names in data of the variables to be tabulated. This can be an empty vector, c(), provided a geography level is supplied.
record_key	A character containing the column name in data giving the record keys required for perturbation. If data contains "ons_id" and use_existing_ons_id = TRUE, set (record_key = NULL), as record key will be generated from "ons_id".
use_existing_ons_id	A logical on whether to create record keys from ons_id, if ons_id exists in data. It will be irrelevant if microdata does not contain ons_id. Default is TRUE.
threshold	An integer specifying the value below which counts are suppressed, with a default value of 10.

Value

Returns a data.table giving a frequency table which has had cell key perturbation applied according to the ptable supplied.

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  data.table::setDTthreads(1)
}

geog <- "var1"
tab_vars <- c("var5","var8")
record_key <- "record_key"
```

```
perturbed_table <- create_perturbed_table(micro,
                                          ptable_10_5,
                                          geog,
                                          tab_vars,
                                          record_key)

# Alternatively
perturbed_table <- create_perturbed_table(data = micro,
                                          ptable = ptable_10_5,
                                          geog = c(),
                                          tab_vars = c("var1","var5","var8"),
                                          record_key = "record_key",
                                          threshold = 10)
```

`create_perturbed_table_bigquery` *Create a perturbed frequency table in BigQuery and return it as a data frame*

Description

This function runs the perturbation method fully in BigQuery (via SQL) and only downloads the result, which allows handling large datasets efficiently.

Usage

```
create_perturbed_table_bigquery(
  con,
  data,
  ptable,
  geog,
  tab_vars,
  record_key,
  use_existing_ons_id = TRUE,
  threshold = 10,
  return_query = FALSE
)
```

Arguments

con	–DBIConnection. An active BigQuery connection created with DBI::dbConnect()
data	–character. BigQuery table name for microdata in full format: "<PROJECT>.<DATASET>.<TABLE>". One row per statistical unit (person, household, business, etc.), and one column per variable (e.g. age, sex, health status)
ptable	–character. BigQuery table name for the p-table in full format: "<PROJECT>.<DATASET>.<TABLE>".

`geog` –`character vector`. Column name containing the desired geography level for the frequency table. e.g., `c("Region")` or `c("LocalAuthority")`. Use `c()` if no geography breakdown required.

`tab_vars` –`character vector`. Column names to tabulate, e.g., `c("Age", "Health", "Occupation")`.

`record_key` –`character`. Column name with record keys required for perturbation, e.g., `"Record_Key"`. If data contains "ons_id" and use_existing_ons_id = TRUE, set (record_key = NULL), as record key will be generated from "ons_id".

`use_existing_ons_id` – `logical` Whether to create record keys from ons_id, if ons_id exists in data. It will be irrelevant if microdata does not contain ons_id. Default is TRUE.

`threshold` –`integer`. Suppression threshold; perturbed counts below this value are suppressed. Default 10.

`return_query` –`logical`. If `TRUE`, returns the generated SQL query without executing it. Default `FALSE`.

Details

Function workflow:

1. Generate BigQuery SQL query to run perturbation on BigQuery
2. If return_query = TRUE, return the query text and exit: otherwise, execute the rest
3. Validate inputs using BigQuery
4. Run perturbation using BigQuery
5. Convert perturbed table to data.table and sort

The query build by this function does the following when executed:

- Computes counts and cell keys for each unique combination of geographic and tabulation variables.
- Includes zero-count cells by generating the full cartesian product of variable combinations.
- Calculates pcv by ensuring the rows of ptable 501-750 are reused for cell values above 750.
- Applies perturbation values from a perturbation table based on cell keys and pseudo cell values (pcv).
- Suppresses cells below a specified threshold by setting their perturbed count to NULL.

Value

- When `return_query = FALSE`: a `data.table` containing the perturbed frequency table, sorted by `geog` and `tab_vars`.
- When `return_query = TRUE`: a character string containing the query.

Examples

```
# --- Return query text without executing it ---
query <- create_perturbed_table_bigquery(
  con        = NULL,
  data       = "my-gcp-project.survey.microdata",
  ptable     = "my-gcp-project.sdc.ptable",
  geog       = c("Region"),
  tab_vars   = c("AgeGroup", "HealthStatus", "Occupation"),
  record_key = "Record_Key",
  threshold  = 10,
  return_query = TRUE
)
cat(query)
```

---

generate_ptable_10_5_rule

Generate ptable (10-5 rule)

Description

`generate_ptable_10_5_rule()` generates a sample p-table based on 10-5 rule, which means a suppression threshold of 10 and rounding to the nearest 5.

Usage

```
generate_ptable_10_5_rule(max_pcv = 750, ckey_range = 255)
```

Arguments

`max_pcv`	Max value for pcv. Default is 750.
`ckey_range`	The max range for cell keys. Default is 255.

Value

A data.table assigning a pvalue to each ckey and pcv combination

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  data.table::setDTthreads(1)
}
ptable <- generate_ptable_10_5_rule()
```

generate_random_rkey *Generate and attach random record keys to microdata*

Description

generate_random_key() attaches randomly generated record keys to microdata tables for testing purposes.

Usage

```
generate_random_rkey(data, rkey_range = 255, seed = NULL)
```

Arguments

data	A data.table or data.frame containing the microdata
rkey_range	The max range for record keys. Default is 255.
seed	A seed for the random number generator

Value

A data.table with a new integer column record_key

Examples

```
library(data.table)
data <- data.table(id = 1:1000)
data <- generate_random_rkey(data, rkey_range = 255, seed = 2005)
```

generate_record_key_from_ons_id *Generate Record Key from ONS ID*

Description

This function creates a new record key column by taking the modulo 4096 of the ons_id column. It converts ons_id to numeric, preserving NA for non-numeric values, and assigns the result as an integer.

Usage

```
generate_record_key_from_ons_id(data, record_key_col)
```

Arguments

data	A data.table containing the ons_id column.
record_key_col	A character string specifying the name of the new record key column to create.

Details

- The function checks that `data` is a `data.table`.
- Non-numeric values in `ons_id` are converted to `NA`.
- The record key is computed as `ons_id %% 4096` and stored as integer.

Value

A `data.table` with the new record key column added.

`generate_test_data` *Generate sample microdata*

Description

`generate_test_data()` creates a sample microdata containing randomly generated microdata columns and record keys for testing purposes.

Note: You can set a seed for random value generator to obtain same output in different runs. However, the sample microdata included in the package will be different than this one, as it was generated from the corresponding python package for consistency in test output.

Usage

```
generate_test_data(size = 1000, rkey_range = 255, seed = NULL)
```

Arguments

`size`	Number of rows in the sample microdata. Default is 1000.
`rkey_range`	The max range for record keys. Default is 255.
`seed`	A seed for the random number generator

Value

A data.table containing randomly generated microdata and record keys

Examples

```
if (requireNamespace("data.table", quietly = TRUE)) {
  data.table::setDTthreads(1)
}
data <- generate_test_data(size = 1000)
data <- generate_test_data(size = 1000, rkey_range = 255, seed = 111)
```

micro *Example data (micro)*

Description

A data set containing randomly generated data to showcase the cell key perturbation method.

Usage

```
data(micro)
```

Format

A data.table containing 1000 observations of 11 variables

Details

- record_key. record key value (0-255)
- var1. example variable 1 (1-5)
- var2. example variable 2 (1,2)
- var3. example variable 3 (1-4)
- var4. example variable 4 (1-4)
- var5. example variable 5 (1-10)
- var6. example variable 6 (1-5)
- var7. example variable 7 (1-5)
- var8. example variable 8 (A-D)
- var9. example variable 9 (A-H)
- var10. example variable 10 (1-49)

ptable_10_5 *Perturbation table*

Description

A data set containing the rules to apply cell key perturbation with a threshold of 10, and rounding to base 5. In other words, counts less than 10 will be removed, and all others will be rounded to the nearest 5.

Usage

```
data(ptable_10_5)
```

Format

A data.table containing 192000 observations of 3 variables

Details

- pcv. perturbation cell value (1-750)
- ckey. cell key value (0-255)
- pvalue. perturbation value to be applied

`validate_inputs_bigquery` *Validate Inputs Before Perturbation using BigQuery*

Description

Validates BigQuery inputs for a perturbation process.

- Validate input arguments
  - Check that at least one variable specified for geog or tab_vars
  - Check geog and tab_vars are either character vectors or NULL
  - Check specified record_key is character vector or NULL
  - Check threshold is an integer and non-negative
- Validate microdata and ptable contain required columns
  - Check data contain the specified geog, tab_vars & record_key
  - Check ptable contains required columns
- Validate the range of record keys and cell keys
- Validate data has sufficient records with record keys to apply perturbation

Usage

```
validate_inputs_bigquery(
  con,
  data,
  ptable,
  geog,
  tab_vars,
  record_key,
  use_existing_ons_id,
  threshold
)
```

Arguments

con	–DBIConnection. An active BigQuery connection created with DBI::dbConnect()
data	–character. BigQuery table name for microdata in full format: "<PROJECT>.<DATASET>.<TABLE>". One row per statistical unit (person, household, business, etc.), and one column per variable (e.g. age, sex, health status)
ptable	–character. BigQuery table name for the p-table in full format: "<PROJECT>.<DATASET>.<TABLE>".
geog	–character vector. Column name containing the desired geography level for the frequency table. e.g., c("Region") or c("LocalAuthority"). Use c() if no geography breakdown required.
tab_vars	–character vector. Column names to tabulate, e.g., c("Age", "Health", "Occupation").
record_key	–character. Column name with record keys required for perturbation, e.g., "Record_Key". If data contains "ons_id" and use_existing_ons_id = TRUE, set (record_key = NULL), as record key will be generated from "ons_id".
use_existing_ons_id	– logical Whether to create record keys from ons_id, if ons_id exists in data. It will be irrelevant if microdata does not contain ons_id. Default is TRUE.
threshold	–integer. Suppression threshold; perturbed counts below this value are suppressed. Default 10.

Value

Invisibly returns TRUE on success. Throws stop or Warning messages if any validation fails.

Index

∗ **datasets**
micro, 9
ptable_10_5, 9

check_for_na, 2
create_perturbed_table, 2
create_perturbed_table_bigquery, 4

generate_ptable_10_5_rule, 6
generate_random_rkey, 7
generate_record_key_from_ons_id, 7
generate_test_data, 8

micro, 9

ptable_10_5, 9

validate_inputs_bigquery, 10