Package 'ptable'

May 9, 2026

Type Package

Title Generation of Perturbation Tables for the Cell-Key Method

Version 1.0.0

Maintainer Tobias Enderle <tobias.enderle@destatis.de>

Description Tabular data from statistical institutes and agencies are mostly confidential and must be protected prior to publications. The cell-key method is a post-tabular Statistical Disclosure Control perturbation technique that adds random noise to tabular data. The statistical properties of the perturbations are defined by some noise probability distributions - also referred to as perturbation tables.
This tool can be used to create the perturbation tables based on a maximum entropy approach as described for example in Giessing (2016) <doi:10.1007/978-3-319-45381-1_18>. The perturbation tables created can finally be used to apply a cell-key method to frequency count or magnitude tables.

License EUPL

URL https://github.com/sdcTools/ptable

BugReports https://github.com/sdcTools/ptable/issues

Depends R(>= 3.6)

Imports data.table, flexdashboard, ggplot2, methods, nloptr, RColorBrewer, rlang, rmarkdown

Suggests knitr, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

NeedsCompilation no

RoxygenNote 7.2.3

Author Tobias Enderle [aut, cre]

Repository CRAN

Date/Publication 2023-03-01 20:10:09 UTC

Contents

create_ptable 2
modify_cnt_ptable 5
plot 6
ptable 7
ptable-class 8
ptable_params-class 9
pt_check 9
pt_export 10
pt_ex_cnts 11
pt_ex_nums 11
pt_optim_entropy 12
pt_vignette 13

Index **14**

`create_ptable` *Noise Probability Generator for the Cell-Key Method (CKM)*

Description

ptable makes it easy to create perturbation tables that can be used for applying noise to statistical tables with any cell-key method approach - among others either the cellKey()-package or the standalone tool TauArgus.

The package provides four main functions to create the perturbation tables:

- `create_ptable()`: generic function that creates a ptable, either for frequency count or magnitude tables with a various set of options.
- `create_cnt_ptable()`: creates a ptable suitable for frequency count tables.
- `create_num_ptable()`: creates a ptable suitable for magnitude tables (i.e. with numerical variables).
- `modify_cnt_ptable()`: modifies the ptable for a higher level of protection

Usage

```
create_ptable(
  D,
  V,
  js = 0,
  pstay = NULL,
  optim = 1,
  mono = TRUE,
  step = 1,
  icat = NULL,
  table = "cnts",
  type = "all",
```

```
  label = paste0("D", D, "V", V * 100),
  monitoring = FALSE,
  debugging = FALSE,
  create = TRUE,
  params = NULL
)

create_cnt_ptable(
  D,
  V,
  js = 0,
  pstay = NULL,
  optim = 1,
  mono = TRUE,
  label = paste0("D", D, "V", V * 100),
  monitoring = FALSE,
  create = TRUE
)

create_num_ptable(
  D,
  V,
  pstay = NULL,
  optim = 1,
  mono = TRUE,
  step = 2,
  icat = NULL,
  type = "all",
  label = paste0("D", D, "V", V * 100),
  monitoring = FALSE,
  create = TRUE
)
```

Arguments

D	perturbation parameter for maximum noise (scalar integer)
V	perturbation parameter for variance (scalar double)
js	threshold value for blocking of small frequencies (i.e. the perturbation will not produce positive cell values that are equal to or smaller than the threshold value). (scalar integer)
pstay	optional parameter to set the probability ($0 < p < 1$) of an original frequency to remain unperturbed: NA (default) no preset probability (i.e. produces the maximum entropy solution)
optim	optimization parameter: 1 standard approach (default) with regular constraints, 4 alternative approach with simplified constraints (may work if constraints using the standard approach are violated)
mono	(logical) vector specifying optimization parameter for monotony condition

`step` (integer) number of steps for the noise (between two integer values). Whereas the cell-key approach for frequency count tables only allows to have noise values that are integers (`step = 1`)

$$-D, 1-D, 2-D, ..., -1, 0, 1, ..., D-2, D-1, D$$

the noise distribution for magnitude values does not have to be integer valued:

$$-D, (1/step)-D, (2/step)-D, ..., 0, ..., D-(2/step), D-(1/step), D$$

The reciprocal of step (='step width') is computed and used internally for the perturbation table.

`icat` (integer) categorized original frequencies i

`table` (character) type of the table: frequency count (`cnts`) or magnitude table (`nums`)

`type` (character) type indicator for the extra column 'type' used for magnitude tables: `'even'`, `'odd'` or `'all'` (default)

`label` (character) label of the Output

`monitoring` (logical) output monitoring on/off

`debugging` (logical) debug monitoring on/off

`create` (logical) scalar specifying to create just the input parameters of class ptable_params (`FALSE`) or also to create the perturbation table object of class ptable (default: `TRUE`)

`params` object of class ptable_params can be used as input instead of the remaining parameters

Details

The perturbation probabilities are constructed given the following constraints:

- Maximum noise
- Zero mean (unbiased noise)
- Fixed noise variance
- Transition probabilities are between zero and one and the sum up to 1
- Perturbations will not produce negative cell values or positive cell values equal to or less than a specific threshold value

Value

Returns ptable object including the created perturbation table by default. If the argument `create = FALSE`, a ptable_params object is returned.

See Also

- `plot()` to analyze the created perturbation table visually
- `pt_export()` to export the perturbation table for external sdcTools like TauArgus or SAS.

Examples

```
# create ptable for frequency count tables
create_cnt_ptable(D = 3, V = 1.08, js = 1, label = "ptable_frequency_tab")

# create ptable for magnitude tables
create_num_ptable(D = 5, V = 2, step = 4, icat = c(1, 3, 5))

# create ptable for frequency or magnitude tables
create_ptable(D = 3, V = 1.08, js = 1, table="cnts")
create_ptable(D = 5, V = 2, step = 4, icat = c(1, 4, 5), table="nums")
```

`modify_cnt_ptable` *Modify a ptable suitable for frequency count variables*

Description

`modify_cnt_ptable()` is a function to modify the standard ptable for count variables that is generated by `create_cnt_ptable()` or within the 'cellKey'-package. The noise intervals in the standard ptable are ordered from -D to D. A modified ptable still has the same properties as the standard ptable but can ensure a higher protection of perturbed frequency tables since the noise probabilities are split and the intervals are rearranged.

Usage

```
modify_cnt_ptable(input, threshold = 0.2, seed = NULL)
```

Arguments

`input`	The ptable-object of class 'ptable', 'ck_params' or data.table
`threshold`	The maximum width of the intervals after modification
`seed`	A seed for the rearrangement of the split intervals

Details

In a first step, the noise probabilities larger than a threshold value will be split. Then, the split noise probabilities are randomly rearranged using a seed (the modifications is replicable). Finally, the intervals of the ptable will be adjusted.

Value

Returns an object of class ptable or a data.table.

Author(s)

Tobias Enderle, <tobias.enderle@destatis.de>

See Also

create_ptable()

Examples

```
# Original ptable
ptab <- create_cnt_ptable(3, 1)

# modified ptable
ptab_mod <- modify_cnt_ptable(ptab, 0.3, seed = 5467)
ptab_mod@pTable
```

---

plot *Plot the results of the perturbation table generator*

Description

plot() makes it easy to visualize the results of the created ptable-object that has been created by create_cnt_ptable(), create_cnt_ptable() or modify_cnt_ptable().

Usage

```
plot(obj, type = "d", file = NULL, ...)
```

Arguments

obj	an object of class ptable
type	(character) type of graph: distribution "d" (standard), perturbation panel ("p"), transition matrix "t"
file	if not NULL, a path to a file (with file extension, e.g. '.pdf' or '.png') where the graph is saved to
...	additional parameters passed to methods

Value

The selected graph is displayed, but there is no direct return value. The output could also be assigned to an object to receive an object of class ggplot.

Author(s)

Tobias Enderle

Examples

```
# Create a ptable for frequency count tables and modify the intervals
ptab <- create_cnt_ptable(D = 3, V = 1.05, js = 1, label = "Example")
ptab_mod <- modify_cnt_ptable(ptab, threshold = 0.3, seed = 5432)

# Distribution Plot of the Noise
plot(ptab_mod, type = "d")

# Perturbations Panel of the Noise
plot(ptab_mod, type = "p")


## Plot and Save the Transition Matrix
plot(ptab_mod, type ="t",
     file = tempfile("example_tMatrix", fileext = ".pdf"))
```

`ptable` *Perturbation Table Dashboard for Frequency Count Tables*

Description

In the `ptable`-package there is a shiny app for first time users and visual-style learners. `ptable()` makes it easy to experiment with different parameter settings while getting direct feedback by means of graphical plots and summaries. The different result output tabs are:

- `Perturbation Table` shows the output used for applying CKM methods.
- `Constraints Check` can be used to check the main constraints (e.g., zero mean, fixed variance)
- `Input Code` could be used for replication of the results (i.e. copy&paste the code for your R script).
- `Input Object` shows the input object derived from the parameters a user set.
- `Legend` gives an overview of used parameters.

Users can also visually learn how input parameters effect the perturbation table:

- `Transition Matrix`
- `Distribution` Plot
- `Perturbation Panel` Plot

Usage

```
ptable()
```

Value

No return value, the dashboard is opened in the default browser.

Note

After usage (e.g. closing the browser tab), interrupt R to stop the application (usually by pressing Ctrl+C or Esc in the console or by using the stop button in RStudio).

Author(s)

Tobias Enderle, <tobias.enderle@destatis.de>

See Also

See `create_cnt_ptable()` to get more help or `pt_vignette()` for an introduction

Examples

```
# Run the dashboard in your default browser
ptable()
```

---

ptable-class *An S4 class to represent perturbation table*

Description

An S4 class to represent perturbation table

Slots

`tMatrix` (matrix) transition matrix with perturbation probabilities

`pClasses` (numeric) numeric classes

`pTable` (data.table) perturbation table with probabilities

`empResults` (data.table) ...

`pParams` a `ptable_params` object

`tStamp` (character) ...

`type` (character) type indicator for magnitude tables

`table` (character) type of table: frequency counts (cnts) or magnitude (nums)

ptable_params-class *An S4 class to represent perturbation parameters*

Description

An S4 class to represent perturbation parameters

Slots

`D` (integer) parameter for maximum perturbation / noise

`V` (numeric) parameter for perturbation variance

`js` (integer) parameter for original counts not to be perturbed

`ncat` (integer) number of perturbation classes

`pstay` numeric vector specifying parameter for non-perturbation

`optim` (integer) specifying optimization parameter for optimization function

`mono` (logical) vector specifying optimization parameter for monotony condition

`label` (character) label for output

`icat` (integer) categorized original frequencies i

`table` (character) type of table: frequency counts (`cnts`) or magnitude (`nums`)

`step` (integer) step

`type` (character) indicator for the extra column 'type' used for magnitude tables: `'even'`, `'odd'` or `'all'`

pt_check *Check the constraint of the ptable*

Description

`pt_check()` checks the constraints of the ptable

Usage

```
pt_check(ptab)
```

Arguments

`ptab`	a `data.table` or an an object of ptable generated with `create_cnt_ptable()`.

Value

a data.table object

Author(s)

Tobias Enderle, <tobias.enderle@destatis.de>

Examples

```
# create ptable
ptab1 <- create_cnt_ptable(D = 5, V = 3, js = 2, label = "test2")

# check ptable
pt_check(ptab1)
```

---

pt_export *Export ptables as a txt-file*

Description

Function to export perturbation table to Tau-Argus, SAS or any other CKM tool (as txt-file).

Usage

```
pt_export(..., file, SDCtool = "TauArgus")
```

Arguments

...	1 or 2 input object of class ptable
file	(character) filename (only 'txt' is possible as file extension)
SDCtool	(character) either "TauArgus" or "SAS"

Value

Returns 'NULL' and the ptable is saved in the specified format.

Author(s)

Tobias Enderle

Examples

```
ptab <- create_cnt_ptable(D = 5, V = 3, js = 2, label = "test")
pt_export(ptab, file = tempfile("ptable_example"), SDCtool = "TauArgus")
```

pt_ex_cnts *A quick ptable that can be used in various examples*

Description

`pt_ex_cnts()` returns a perturbation table object from `create_cnt_ptable()` with some default parameters. This is useful for quickly creating ptables to demonstrate usage in other tools.

Usage

```
pt_ex_cnts()
```

Value

Returns a ptable object.

Examples

```
ptab <- pt_ex_cnts()
plot(ptab, type = "t")
```

pt_ex_nums *Quick ptables for numeric variables*

Description

`pt_ex_nums()` returns a perturbation table objects from `create_num_ptable()` with some default parameters. This is useful for quickly creating ptables to demonstrate usage in other tools.

Usage

```
pt_ex_nums(parity = TRUE, separation = FALSE)
```

Arguments

`parity`	a scalar logical; if `TRUE`, a single ptable will be generated. If `FALSE`, two ptables for even and odd numbers are created
`separation`	a scalar logical; if `TRUE`, an additional ptable with variance `1` will be returned that is designed to perturb small cell values

Value

Returns a ptable object if both `parity` and `separation` are `FALSE`, else a named list.

Examples

```
# extra ptable for small cells
names(pt_ex_nums(parity = FALSE, separation = TRUE))

# different ptables for even/odd cells
names(pt_ex_nums(parity = TRUE, separation = TRUE))
```

`pt_optim_entropy` *Maximum Entropy Approach*

Description

Function to solve the non-linear optimization problem used within `ptable()`.

Usage

```
pt_optim_entropy(
  optim = optim,
  mono = mono,
  v = v,
  variance = variance,
  lb = p_lb,
  ub = p_ub,
  ndigits
)
```

Arguments

`optim`	optimization parameter (1=default, 2-4=further test implementations)
`mono`	(logical) monotony parameter
`v`	(integer) vector with perturbation values (i.e. deviations to the original frequency)
`variance`	(numeric) variance parameter
`lb`	(integer) vector with lower bounds of the controls
`ub`	(integer) vector with upper bounds of the controls
`ndigits`	(integer) number of digits

Details

The main parameter is ‘optim‘: In ‘optim=1 to 3‘ the variance is stated as inequality constraint and in ‘optim=4‘ the variance condition is stated as equality constraint.

Value

The return value contains a list with two elements:

`"result"` optimal value of the controls

`"iter"` number of iterations that were executed

Author(s)

Tobias Enderle, Sarah Giessing, Jonas Peter

See Also

Giessing, S. (2016), 'Computational Issues in the Design of Transition Probabilities and Disclosure Risk Estimation for Additive Noise'. In: Domingo-Ferrer, J. and Pejic-Bach, M. (Eds.), Privacy in Statistical Databases, pp. 237-251, Springer International Publishing, LNCS, vol. 9867.

Fraser, B. and Wooton, J.: A proposed method for confidentialising tabular output to protect against differencing. In: Monographs of Official Statistics. Work session on Statistical Data Confidentiality, Eurostat-Office for Official Publications of the European Communities, Luxembourg, 2006, pp. 299-302

`pt_vignette` *Vignette*

Description

Starts the package vignette that gets you started with the package

Usage

```
pt_vignette()
```

Value

a browser windows/tab with showing the vignette

Examples

```
pt_vignette()
```

Index

∗ **check**
    pt_check, 9
∗ **dashboard**
    ptable, 7
∗ **export**
    pt_export, 10
∗ **flexdashboard**
    ptable, 7
∗ **optimization**
    pt_optim_entropy, 12
∗ **perturbation**
    ptable, 7
∗ **plot**
    plot, 6
∗ **ptable**
    pt_check, 9
∗ **shiny**
    ptable, 7
∗ **table**
    ptable, 7
∗ **visualisation**
    ptable, 7

create_cnt_ptable *(*create_ptable*)*, 2
create_cnt_ptable(), *5, 6*, *8, 9*, *11*
create_cnts_ptable *(*create_ptable*)*, 2
create_num_ptable *(*create_ptable*)*, 2
create_num_ptable(), *11*
create_nums_ptable *(*create_ptable*)*, 2
create_ptable, 2
create_ptable(), *6*

modify_cnt_ptable, 5
modify_cnt_ptable(), *2*, *5, 6*
modify_cnts_ptable *(*modify_cnt_ptable*)*, *5*

plot, 6
plot(), *4*, *6*
pt_check, 9
pt_check(), *9*
pt_ex_cnts, 11
pt_ex_cnts(), *11*
pt_ex_nums, 11
pt_ex_nums(), *11*
pt_export, 10
pt_export(), *4*
pt_optim_entropy, 12
pt_vignette, 13
pt_vignette(), *8*
ptable, *4–6*, 7, *9–12*
ptable(), *7*
ptable-class, 8
ptable_params, *4*, *8*
ptable_params-class, 9