

Package: datablinder (via r-universe)

October 7, 2026

Title Synthetic Look-Alike Copies of Sensitive Data Files

Version 0.1.0

Description Replaces the contents of a sensitive or proprietary data file with synthetic values while keeping the columns, classes, formats and file type identical, so that analysis code written against the anonymised copy runs unchanged on the real data. Intended for sharing the structure of a dataset with a collaborator or with an AI assistant (a large language model or LLM chatbot) without sharing its contents. Reads and writes delimited text, 'Excel', 'SPSS', 'Stata', 'RDS' and 'Parquet' files, and includes a local 'Shiny' app and a command line script. Reduces disclosure risk; it offers no formal privacy guarantee.

License MIT + file LICENSE

URL <https://agallinat.github.io/datablinder/>,
<https://github.com/agallinat/datablinder>

BugReports <https://github.com/agallinat/datablinder/issues>

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports bslib, data.table, haven, readxl, shiny, stats, utils, writexl

Suggests arrow, knitr, rmarkdown, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/Needs/website pkgdown

Config/testthat/edition 3

Config/pak/sysreqs cmake make libuv1-dev libx11-dev zlib1g-dev

Repository <https://agallinat.r-universe.dev>

Date/Publication 2026-10-02 21:41:50 UTC

RemoteUrl <https://github.com/agallinat/datablinder>
RemoteRef HEAD
RemoteSha 995aacc9d0c0b5878c77bbdfd1f472fd29b2d440

Contents

blind_data	2
blind_file	4
blind_summary	7
install_cli	8
run_app	9
Index	10

blind_data	<i>Blind a data frame</i>
------------	---------------------------

Description

Returns a copy of data with the same columns, classes and formats, the same number of rows unless rows says otherwise, and fake values throughout. Columns are blinded independently: relationships between them are not kept.

Usage

```
blind_data(  
  data,  
  blind_names = FALSE,  
  keep_labels = FALSE,  
  keep_real = NULL,  
  rows = NULL,  
  seed = NULL  
)
```

Arguments

data	A data frame, tibble or data.table.
blind_names	Rename the columns to col_01, col_02... and drop the variable labels, which often describe sensitive content.
keep_labels	Keep the real factor levels and category values, for when code has to filter on them. FALSE replaces them with A, B, C...
keep_real	Names of columns to copy across unchanged, with their real values , for columns that carry no sensitive information and that code has to use as they are, such as a treatment arm or a study visit. These columns keep their real names even under blind_names, and are named in the summary and excluded from the leak check. Cannot be combined with rows. Use it sparingly: see the warning below.

rows	Number of rows to generate. NULL keeps the input's row count.
seed	A seed, for a reproducible copy. NULL picks one without disturbing the session's random number stream.

Details

This reduces the risk of disclosing the real data. It is not a formal privacy guarantee: the column names, classes, row count and the broad shape of each column stay visible.

Value

A data frame of the same class as data, with a blind_summary attribute describing what was done.

What happens to each column

Types are decided by values, not only by storage class: a double holding nothing but 0 and 1 is binary, and a double holding whole numbers with few distinct values is discrete. That is what keeps `table(x)`, `x == 1` and `factor(x)` behaving the same way on both copies.

- Continuous numeric: drawn from a smoothed inverse CDF of the real column, so the shape survives; slightly shifted minimum and maximum, the same number of decimals, the same sign, the same share of exact zeros.
- Discrete numeric, whole numbers with few distinct values, 0/1 included: the same set of values in similar proportions.
- Factor, ordered factor, character with few distinct values, haven_labelled: the same number of levels or categories in similar proportions, relabelled A, B, C... unless keep_labels. Order, unused levels and haven codes kept.
- Logical: a similar share of TRUE.
- Date, POSIXct: dates in a slightly shifted range, same timezone. Dates that were text in the file stay text, in the same format.
- Identifier, meaning nearly all values distinct or a name like patient_id: new unique values of the same shape; a repeated ID repeats a similar number of times.
- Email, phone, URL: obviously fake values of the same shape, at example.com.
- Fixed-shape text, such as postcodes and codes with leading zeros: random letters, digits and punctuation in the same positions.
- Free text: placeholder words of similar length.
- Constant or all-NA: all-NA stays; a constant number stays; constant text is replaced.

Every column keeps its share of missing values, and a text file keeps the way it wrote them. Code that depends on the content of free text, on a particular real value existing, or on a relationship between two columns will not work.

Columns kept real

keep_real names columns that are copied over untouched, so their real values end up in the shared copy. It exists because code often has to use a real value to be useful at all, as in `arm == "placebo"`, and hand-editing the blinded file back is worse than asking for it.

It is also the one way to make this package disclose real data, so the decision is yours and it is worth making slowly:

- A column that is harmless by itself can still identify someone in combination with the others. A real date of birth, postcode or site, next to a real sex and a real visit date, can be enough, even with every name blinded.
- The rows still line up. A kept column stays matched to the rest of its row, which is why rows cannot be used at the same time.
- The summary says which columns were kept, and the leak check cannot vouch for them. Read both before sharing the file.

The safe default is not to use it. If a column is only needed for its categories and not its contents, keep_labels is usually enough.

See Also

`blind_file()` to do the same to a file.

Examples

```
fake <- blind_data(mtcars, seed = 1)
str(fake)
attr(fake, "blind_summary")

# cyl copied over with its real values, everything else blinded
arms <- blind_data(mtcars, keep_real = "cyl", seed = 1)
table(arms$cyl) == table(mtcars$cyl)
```

blind_file

Blind a data file

Description

Reads a data file, blinds every column of every sheet, and writes a copy in the same format next to the input, keeping the delimiter, decimal mark, encoding and sheet names of a text file or workbook, and the variable labels and display formats of an SPSS or Stata file. Code written against the copy runs unchanged on the real file.

Usage

```
blind_file(
  path,
  output = NULL,
  blind_names = FALSE,
  keep_labels = FALSE,
  keep_real = NULL,
  rows = NULL,
  seed = NULL
)
```

Arguments

path	Path to a CSV, TSV, Excel, SPSS, Stata, RDS or Parquet file. An RDS file must hold a data frame. Parquet needs the arrow package.
output	Where to write the copy. The default adds _blinded to the input's name. An .xls input is written as .xlsx, which is the only Excel format that can be written.
blind_names	Rename the columns to col_01, col_02... and drop the variable labels, which often describe sensitive content.
keep_labels	Keep the real factor levels and category values, for when code has to filter on them. FALSE replaces them with A, B, C...
keep_real	Names of columns to copy across unchanged, with their real values , for columns that carry no sensitive information and that code has to use as they are, such as a treatment arm or a study visit. These columns keep their real names even under blind_names, and are named in the summary and excluded from the leak check. Cannot be combined with rows. Use it sparingly: see the warning below.
rows	Number of rows to generate. NULL keeps the input's row count.
seed	A seed, for a reproducible copy. NULL picks one without disturbing the session's random number stream.

Details

This reduces the risk of disclosing the real data. It is not a formal privacy guarantee: the column names, classes, row count and the broad shape of each column stay visible.

Value

A blind_summary, printed when called at the console, describing the blinded file one column at a time. Paste it into a chat to tell an AI what the data looks like.

What happens to each column

Types are decided by values, not only by storage class: a double holding nothing but 0 and 1 is binary, and a double holding whole numbers with few distinct values is discrete. That is what keeps `table(x)`, `x == 1` and `factor(x)` behaving the same way on both copies.

- Continuous numeric: drawn from a smoothed inverse CDF of the real column, so the shape survives; slightly shifted minimum and maximum, the same number of decimals, the same sign, the same share of exact zeros.
- Discrete numeric, whole numbers with few distinct values, 0/1 included: the same set of values in similar proportions.
- Factor, ordered factor, character with few distinct values, `haven_labelled`: the same number of levels or categories in similar proportions, relabelled A, B, C... unless `keep_labels`. Order, unused levels and haven codes kept.
- Logical: a similar share of TRUE.
- Date, `POSIXct`: dates in a slightly shifted range, same timezone. Dates that were text in the file stay text, in the same format.
- Identifier, meaning nearly all values distinct or a name like `patient_id`: new unique values of the same shape; a repeated ID repeats a similar number of times.
- Email, phone, URL: obviously fake values of the same shape, at `example.com`.
- Fixed-shape text, such as postcodes and codes with leading zeros: random letters, digits and punctuation in the same positions.
- Free text: placeholder words of similar length.
- Constant or all-NA: all-NA stays; a constant number stays; constant text is replaced.

Every column keeps its share of missing values, and a text file keeps the way it wrote them. Code that depends on the content of free text, on a particular real value existing, or on a relationship between two columns will not work.

Columns kept real

`keep_real` names columns that are copied over untouched, so their real values end up in the shared copy. It exists because code often has to use a real value to be useful at all, as in `arm == "placebo"`, and hand-editing the blinded file back is worse than asking for it.

It is also the one way to make this package disclose real data, so the decision is yours and it is worth making slowly:

- A column that is harmless by itself can still identify someone in combination with the others. A real date of birth, postcode or site, next to a real sex and a real visit date, can be enough, even with every name blinded.
- The rows still line up. A kept column stays matched to the rest of its row, which is why rows cannot be used at the same time.
- The summary says which columns were kept, and the leak check cannot vouch for them. Read both before sharing the file.

The safe default is not to use it. If a column is only needed for its categories and not its contents, `keep_labels` is usually enough.

See Also

[`blind\_data\(\)`](#) to do the same to a data frame.

Examples

```
csv <- file.path(tempdir(), "cars.csv")
write.csv(mtcars, csv, row.names = FALSE)
blind_file(csv, seed = 1)
```

blind_summary	<i>The summary of a blinded table</i>
---------------	---------------------------------------

Description

[blind_file\(\)](#) returns a `blind_summary` and [blind_data\(\)](#) hangs one on its result as the `blind_summary` attribute. It describes the blinded data only, never a real value, so it is safe to paste into a chat along with the blinded file, and it tells an AI what each column is before it writes a line of code.

Usage

```
## S3 method for class 'blind_summary'
format(x, ...)

## S3 method for class 'blind_summary'
print(x, ...)
```

Arguments

<code>x</code>	A <code>blind_summary</code> , from blind_file() or from the <code>blind_summary</code> attribute of a blind_data() result.
<code>...</code>	Ignored.

Details

Printed, it is the output file if there was one, then one line per column giving the name, the class and what was done, then the row count and the result of the leak check, which names any column keep_real kept. `format()` returns those same lines as a character vector, for writing them somewhere instead of printing them.

Read it before sharing the file. It is the list of decisions the package made, and a column blinded as a category when it is really an identifier, or as free text when it is really a code, is the case to catch.

Value

`format()` returns a character vector, one element per line. `print()` prints those lines and returns `x` invisibly.

See Also

[blind_file\(\)](#), [blind_data\(\)](#).

Examples

```
fake <- blind_data(mtcars[1:3], seed = 1)
info <- attr(fake, "blind_summary")
info
format(info)
```

install_cli

Install the datablinder command

Description

Copies the command line wrapper out of the installed package and into a directory the shell looks in, so that a file can be blinded with `datablinder patients.csv` instead of a line of R. On Windows a `.cmd` shim is written beside it, since `cmd.exe` cannot run the script itself.

Usage

```
install_cli(dir = NULL)
```

Arguments

`dir` Where to put the command. `NULL` picks a sensible place.

Details

Nothing has to be installed to use the package from a terminal: `Rscript -e 'datablinder::blind_file("patients.csv")'` works everywhere as soon as the package does. This is for people who blind files often enough to want a shorter command.

The default `dir` is the first of `~/.local/bin`, `~/bin` and `/usr/local/bin` that is already on the `PATH` and writable, or `~/.local/bin` if none of them is. On Windows it is a `datablinder\bin` folder under `LOCALAPPDATA`. Missing directories are created. A message says whether the shell will find the command where it landed, and what to do if it will not.

The command is a copy, but it holds no logic: it calls the installed package, so upgrading `datablinder` upgrades the command too. Run this again after upgrading R itself, which moves the `Rscript` the Windows shim points at.

This is the only function in the package that writes a file anywhere permanent, so it asks before it does: it prints the directory and waits for `y`. Any other answer writes nothing, not even the directory. Called where there is nobody to ask, in a script or `Rscript -e`, it is an error rather than a silent write to your home directory. A `dir` inside `tempdir()` is written without asking, since the session takes it away again.

Value

The path of the installed command, invisibly, or `NULL` invisibly if you decline.

See Also

[blind_file\(\)](#), which the command calls, and [run_app\(\)](#) for the same job in a browser.

Examples

```
# Into a scratch directory, to show what is written:
dir <- file.path(tempdir(), "bin")
install_cli(dir)
list.files(dir)
```

run_app

Blind a data file in a browser

Description

Opens a small app in the default browser: choose a file, set the options, press *Blind*, read the summary and download the blinded copy. It runs on 127.0.0.1, so everything stays on this computer, and the uploaded file is deleted when the app closes.

Usage

```
run_app()
```

Details

This reduces the risk of disclosing the real data. It is not a formal privacy guarantee: the column names, classes, row count and the broad shape of each column stay visible.

In RStudio the app can also be started from *Addins > Blind a data file*.

Value

Nothing. Called to run the app, which stops when the browser tab or the R session is interrupted.

See Also

[blind_file\(\)](#) for the same job at the console.

Examples

```
if (interactive()) {
  run_app()
}
```

Index

`blind_data`, [2](#)
`blind_data()`, [6](#), [7](#)
`blind_file`, [4](#)
`blind_file()`, [4](#), [7](#), [9](#)
`blind_summary`, [7](#)

`format.blind_summary (blind_summary)`, [7](#)

`install_cli`, [8](#)

`print.blind_summary (blind_summary)`, [7](#)

`run_app`, [9](#)
`run_app()`, [9](#)