Package 'ExperimentalDesignGeneratoranddRandomiser'

September 2, 2026

Type Package

Version 0.1.0

Date 2026-08-22

Title 'EDGAR': Experimental Design Generator and Randomiser

Description Native R implementation of 'EDGAR', the Experimental Design Generator and Randomiser. 'EDGAR' was originally developed as a suite of 'Excel' <https://www.microsoft.com/microsoft-365/excel> workbooks by the Biometrics team at Rothamsted Research <http://www.edgarweb.org.uk/>. The algorithms were subsequently re-implemented in the open-source 'Python' <https://www.python.org/> project 'rotsl/edgar' <https://rotsl.github.io/edgar/>, distributed as the 'edgar-design' package on 'PyPI' <https://pypi.org/project/edgar-design/>. This R package is a native R port of that 'Python' implementation: it does not require 'Python', 'reticulate' <https://CRAN.R-project.org/package=reticulate>, or any external service at runtime, and provides deterministic, reproducible randomisation for nine experimental designs including alpha designs (Patterson and Williams, 1976) <doi:10.1093/biomet/63.1.83>. Cross-language reproducibility with the 'Python' implementation is achieved by porting the Mersenne Twister seeding implementation from 'CPython' <https://github.com/python/cpython> and the Fisher-Yates shuffle to native R.

Maintainer BiologyAutomation <phonics-tiffs1i@icloud.com>

URL https://github.com/biologyautomation/edgar-r, https://rotsl.github.io/edgar/, http://www.edgarweb.org.uk/

BugReports https://github.com/biologyautomation/edgar-r/issues

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.6)

Suggests jsonlite, openxlsx, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Rohan R [aut, cph] (ORCID: <https://orcid.org/0009-0005-9225-1775>,
Author of the upstream Python implementation rotsl/edgar and this
native R port),
BiologyAutomation [cre] (Repository owner)

Repository CRAN

Date/Publication 2026-09-02 12:30:13 UTC

Contents

`ExperimentalDesignGeneratorandRandomiser-package`
Package-level documentation.

Description

EDGAR - Experimental Design Generator and Randomiser.

Details

A native R implementation of EDGAR. EDGAR originated as Excel workbooks developed by the Biometrics team at Rothamsted Research (http://www.edgarweb.org.uk/). The algorithms were subsequently re-implemented in the open-source Python project `rotsl/edgar` (`rotsl/edgar`; https://rotsl.github.io/edgar/), distributed as the `edgar-design` package on PyPI (https://pypi.org/project/edgar-design/). This R package is a native R port of that Python implementation: it does not require Python, reticulate, or any external service at runtime.

The package provides deterministic, reproducible randomisation for nine experimental designs: `cr_eq`, `cr_uneq`, `rcb`, `rcb_uneq`, `two_factor_rcb`, `latin`, `split_plot`, `variable_blocks`, and `alpha`.

The Mersenne Twister seeding and Fisher-Yates shuffle are ported from CPython so the same integer seed produces the same design in R as in the upstream Python implementation.

Alpha designs follow the methodology described by Patterson and Williams (1976).

Author(s)

Maintainer: BiologyAutomation <phonics-tiffs1i@icloud.com> (Repository owner)

Authors:

- Rohan R <phonics-tiffs1i@icloud.com> (ORCID) (Author of the upstream Python implementation rotsl/edgar and this native R port) [copyright holder]

References

Patterson, H.D. & Williams, E.R. (1976). A new class of resolvable incomplete block designs. Biometrika, 63(1), 83-92. doi:10.1093/biomet/63.1.83

Biometrics team at Rothamsted Research, EDGAR Excel workbooks, http://www.edgarweb.org.uk/

rotsl/edgar Python implementation, https://rotsl.github.io/edgar/

edgar-design on PyPI, https://pypi.org/project/edgar-design/

See Also

Useful links:

- https://github.com/biologyautomation/edgar-r
- https://rotsl.github.io/edgar/
- http://www.edgarweb.org.uk/
- Report bugs at https://github.com/biologyautomation/edgar-r/issues

---

`as.data.frame.edgar_design` `as.data.frame` *method: returns the underlying* `rows` *data frame.*

Description

`as.data.frame` method: returns the underlying `rows` data frame.

Usage

```
## S3 method for class 'edgar_design'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	An `edgar_design` object.
`row.names`	See base::as.data.frame. Unused.
`optional`	See base::as.data.frame. Unused.
`...`	Passed on to other methods.

Value

The `rows` data frame.

`as_layout_frames` *Access the layout view as a list of data frames (one per section), with section names from* `layout_section_labels`*. Returns NULL if the design has no layout.*

Description

Access the layout view as a list of data frames (one per section), with section names from `layout_section_labels`. Returns NULL if the design has no layout.

Usage

```
as_layout_frames(x)
```

Arguments

x	An edgar_design object (or any object with an as_layout_frames method).

Value

A list of data frames, or NULL.

---

design_alpha *Generate an* alpha *design.*

Description

Generate an alpha design.

Usage

```
design_alpha(
  experiment_name = "",
  treatment_factor = "Variety",
  repeated_controls = 0L,
  treatment_count = 20L,
  replicate_factor = "Rep",
  reps = 4L,
  block_factor = "Block",
  blocks_per_replicate = 5L,
  unit_label = "Plot",
  treatment_names = NULL,
  control_names = NULL,
  seed = 0L
)
```

Arguments

experiment_name	Optional experiment name.
treatment_factor	Treatment column header (default "Variety").
repeated_controls	Number of repeated controls per block (0..6).
treatment_count	Number of treatments (20..100).
replicate_factor	Replicate header (default "Rep").

reps	Number of replicates (2..4).
block_factor	Block header (default "Block").
blocks_per_replicate	Number of blocks per replicate (5..15).
unit_label	Unit column header (default "Plot").
treatment_names	Optional character vector of treatment names.
control_names	Optional character vector of control names. If repeated_controls > 0 and this is NULL, defaults to paste0("C", 1:repeated_controls).
seed	Integer seed.

Value

An edgar_design S3 object.

References

Patterson, H.D. & Williams, E.R. (1976). A new class of resolvable incomplete block designs. Biometrika, 63(1), 83-92. doi:10.1093/biomet/63.1.83

Examples

```
res <- design_alpha(treatment_count = 24, reps = 2,
                    blocks_per_replicate = 6, seed = 100)
print(res)
```

---

design_cr *Generate a* cr_eq *design.*

Description

Generate a cr_eq design.

Usage

```
design_cr(
  experiment_name = "",
  treatment_factor = "Variety",
  treatment_count = 2L,
  reps_per_treatment = 2L,
  unit_label = "Plot",
  treatment_names = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name (string).
`treatment_factor`	Treatment column header (default "Variety").
`treatment_count`	Number of treatments (2..500).
`reps_per_treatment`	Replicates per treatment (1..100).
`unit_label`	Unit column header (default "Plot").
`treatment_names`	Optional character vector of treatment names. Defaults to `as.character(1:treatment_count)`; shorter vectors are padded with sequential numbers, matching the upstream behaviour.
`seed`	Integer seed. Default 0L.

Value

An `edgar_design` S3 object.

Examples

```
res <- design_cr(treatment_count = 4, reps_per_treatment = 2, seed = 42)
print(res)
```

`design_cr_unequal` *Generate a* `cr_uneq` *design.*

Description

Generate a `cr_uneq` design.

Usage

```
design_cr_unequal(
  experiment_name = "",
  treatment_factor = "Variety",
  treatment_count = 2L,
  per_treatment_reps = NULL,
  unit_label = "Plot",
  treatment_names = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name.
`treatment_factor`	Treatment column header (default "Variety").
`treatment_count`	Number of treatments (2..500).
`per_treatment_reps`	Integer vector of length `treatment_count` with the replication count per treatment. Defaults to `rep(2, treatment_count)` (NOT a hardcoded `[2, 2]`).
`unit_label`	Unit column header (default "Plot").
`treatment_names`	Optional character vector of treatment names.
`seed`	Integer seed. Default 0L.

Value

An `edgar_design` S3 object.

`design_info` *Convenience wrapper to fetch a design's metadata (for programmatic introspection by tests or downstream packages).*

Description

Convenience wrapper to fetch a design's metadata (for programmatic introspection by tests or downstream packages).

Usage

```
design_info(key)
```

Arguments

`key`	Design key.

Value

A list with `key`, `name`, `description`, `has_layout`, `has_blocks`, `default_params`, `param_specs`.

`design_latin` *Generate a* `latin` *design.*

Description

Generate a `latin` design.

Usage

```
design_latin(
  experiment_name = "",
  treatment_factor = "Variety",
  treatment_count = 3L,
  row_factor = "Row",
  row_count = 3L,
  column_factor = "Column",
  column_count = 3L,
  replicate_factor = "Square",
  treatment_names = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name.
`treatment_factor`	Treatment column header (default "Variety").
`treatment_count`	Number of treatments (>= 2).
`row_factor`	Row header (default "Row").
`row_count`	Number of rows. Must be a multiple of `treatment_count`.
`column_factor`	Column header (default "Column").
`column_count`	Number of columns. Must be a multiple of `treatment_count`.
`replicate_factor`	Square header (default "Square").
`treatment_names`	Optional character vector of treatment names.
`seed`	Integer seed.

Value

An `edgar_design` S3 object.

`design_rcb` *Generate an* `rcb` *design.*

Description

Generate an `rcb` design.

Usage

```
design_rcb(
  experiment_name = "",
  treatment_factor = "Variety",
  treatment_count = 2L,
  block_factor = "Block",
  block_count = 2L,
  unit_label = "Plot",
  treatment_names = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name (string).
`treatment_factor`	Treatment column header (default "Variety").
`treatment_count`	Number of treatments (2..500).
`block_factor`	Block column header (default "Block").
`block_count`	Number of blocks (1..100).
`unit_label`	Unit column header (default "Plot").
`treatment_names`	Optional character vector of treatment names. Defaults to `as.character(1:treatment_count)`; shorter vectors are padded with sequential numbers, matching the upstream behaviour.
`seed`	Integer seed. Default 0L.

Value

An `edgar_design` S3 object.

Examples

```
res <- design_rcb(treatment_count = 4, block_count = 3, seed = 42)
print(res)
```

`design_rcb_unequal` *Generate an* `rcb_uneq` *design.*

Description

Generate an `rcb_uneq` design.

Usage

```
design_rcb_unequal(
  experiment_name = "",
  treatment_factor = "Variety",
  treatment_count = 2L,
  block_factor = "Block",
  block_count = 2L,
  unit_label = "Plot",
  reps_per_treatment_per_block = NULL,
  treatment_names = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name (string).
`treatment_factor`	Treatment column header (default "Variety").
`treatment_count`	Number of treatments (2..500).
`block_factor`	Block column header (default "Block").
`block_count`	Number of blocks (1..100).
`unit_label`	Unit column header (default "Plot").
`reps_per_treatment_per_block`	Integer vector of length `treatment_count`. Each entry is the number of times the corresponding treatment appears within each block. Defaults to `rep(2, treatment_count)`.
`treatment_names`	Optional character vector of treatment names. Defaults to `as.character(1:treatment_count)`; shorter vectors are padded with sequential numbers, matching the upstream behaviour.
`seed`	Integer seed. Default 0L.

Value

An `edgar_design` S3 object.

`design_split_plot` *Generate a* `split_plot` *design.*

Description

Generate a `split_plot` design.

Usage

```
design_split_plot(
  experiment_name = "",
  block_factor = "Block",
  block_count = 2L,
  main_unit_label = "Main plot",
  main_treatment_factor = "MainTreat",
  main_treatment_count = 2L,
  sub_unit_label = "Sub-plot",
  sub_treatment_factor = "SubTreat",
  sub_treatment_count = 2L,
  main_treatment_names = NULL,
  sub_treatment_names = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name.
`block_factor`	Block column header (default "Block").
`block_count`	Number of blocks (1..100).
`main_unit_label`	Whole-plot column header (default "Main plot").
`main_treatment_factor`	Main treatment header (default "MainTreat").
`main_treatment_count`	Number of main treatments (2..100).
`sub_unit_label`	Sub-plot column header (default "Sub-plot").
`sub_treatment_factor`	Sub-treatment header (default "SubTreat").
`sub_treatment_count`	Number of sub-treatments (2..100).
`main_treatment_names`	Optional character vector of main treatment names.
`sub_treatment_names`	Optional character vector of sub-treatment names.
`seed`	Integer seed.

Value

An edgar_design S3 object.

`design_two_factor_rcb` *Generate a* `two_factor_rcb` *design.*

Description

Generate a two_factor_rcb design.

Usage

```
design_two_factor_rcb(
  experiment_name = "",
  factor_a = "Treatment1",
  factor_a_count = 2L,
  factor_b = "Treatment2",
  factor_b_count = 2L,
  block_factor = "Block",
  block_count = 2L,
  unit_label = "Plot",
  factor_a_names = NULL,
  factor_b_names = NULL,
  seed = 0L
)
```

Arguments

experiment_name	Optional experiment name.
factor_a	Header for factor A (default "Treatment1").
factor_a_count	Number of levels of factor A (2..100).
factor_b	Header for factor B (default "Treatment2").
factor_b_count	Number of levels of factor B (2..100).
block_factor	Block column header (default "Block").
block_count	Number of blocks (1..100).
unit_label	Unit column header (default "Plot").
factor_a_names	Optional character vector of factor A level names.
factor_b_names	Optional character vector of factor B level names.
seed	Integer seed.

Value

An edgar_design S3 object.

`design_variable_blocks` *Generate a* `variable_blocks` *design.*

Description

Generate a `variable_blocks` design.

Usage

```
design_variable_blocks(
  experiment_name = "",
  treatment_factor = "Variety",
  treatment_count = 2L,
  replicates = 2L,
  block_factor = "Block",
  block_count = 2L,
  unit_label = "Plot",
  treatment_names = NULL,
  control_flags = NULL,
  block_sizes = NULL,
  seed = 0L
)
```

Arguments

`experiment_name`	Optional experiment name.
`treatment_factor`	Treatment column header (default "Variety").
`treatment_count`	Number of treatments (2..500).
`replicates`	Number of replicates per treatment (1..100).
`block_factor`	Block column header (default "Block").
`block_count`	Number of blocks (1..100).
`unit_label`	Unit column header (default "Plot").
`treatment_names`	Optional character vector of treatment names.
`control_flags`	Optional logical vector of length `treatment_count`. `TRUE` marks a control treatment, which is placed in every block.
`block_sizes`	Optional integer vector of block sizes. Defaults to evenly distributing `treatment_count * replicates` across `block_count`, with any remainder distributed to the first blocks.
`seed`	Integer seed.

Value

An edgar_design S3 object.

---

edgar_py_random *Constructor: returns an environment that owns the MT state. Pass by reference lets* genrand_uint32 *advance the state without copying.*

Description

Constructor: returns an environment that owns the MT state. Pass by reference lets genrand_uint32 advance the state without copying.

Usage

```
edgar_py_random(seed)
```

Arguments

seed	Integer (or numeric coercible to one). Negative seeds are silently treated as their absolute value, matching CPython's behaviour where random.Random(-1) and random.Random(1) produce identical streams.

Value

An environment with $state (numeric vector of 624 uint32s) and $index (1-based pointer into $state).

---

generate_design *Public dispatcher: generate a design by key.*

Description

Public dispatcher: generate a design by key.

Usage

```
generate_design(type, ..., seed = 0L)
```

Arguments

type	Design key (cr_eq, cr_uneq, rcb, rcb_uneq, two_factor_rcb, latin, split_plot, variable_blocks, alpha).
...	Design-specific parameters. Forwarded to the design function.
seed	Integer seed. Default 0L. Same seed produces the same design, byte-identical to the upstream Python implementation.

Value

An edgar_design S3 object.

Examples

```
res <- generate_design("rcb", treatment_count = 4, block_count = 3, seed = 42)
print(res)
```

---

has_layout *Whether the design carries a layout view.*

Description

Whether the design carries a layout view.

Usage

```
has_layout(x)
```

Arguments

x	An edgar_design object (or any object with a has_layout method).

Value

Logical.

---

list_designs *List all registered designs with their metadata.*

Description

List all registered designs with their metadata.

Usage

```
list_designs()
```

Value

A data frame with columns key, name, has_layout, has_blocks. The full info lists are stored in the attribute "info"‘.

Examples

```
list_designs()
```

make_rng *Create a fresh seeded RNG.*

Description

Create a fresh seeded RNG.

Usage

```
make_rng(seed = 0L)
```

Arguments

seed	Integer seed. Negative seeds behave as in CPython (the absolute value is used internally).

Value

An environment usable with `seeded_shuffle()`, `seeded_sample()`, `seeded_randint()`.

new_edgar_design *Constructor: build an* edgar_design *S3 object.*

Description

Constructor: build an `edgar_design` S3 object.

Usage

```
new_edgar_design(
  design_name,
  parameters,
  seed,
  rows,
  layout = NULL,
  layout_headers = NULL,
  layout_section_labels = NULL,
  warnings = character(),
  generated_at = Sys.time()
)
```

Arguments

design_name	Character design name (e.g. "Randomised complete block design").
parameters	Named list of input parameters.
seed	Integer seed used.
rows	A data frame, or a list of named lists coercible by edgar_rows_to_dataframe().
layout	Nested layout view (sections x rows x cells), or NULL.
layout_headers	Per-section column headers, or NULL.
layout_section_labels	Section labels, or NULL.
warnings	Character vector of soft warnings.
generated_at	POSIXct timestamp; defaults to Sys.time().

Value

An edgar_design S3 object.

---

print.edgar_design *print method for* edgar_design. *Mirrors the upstream CLI summary.*

Description

print method for edgar_design. Mirrors the upstream CLI summary.

Usage

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

Arguments

x	An edgar_design object.
...	Unused.

Value

Invisible x.

`propose_alpha_structures` *Propose viable alpha design structures for a given treatment count. Mirrors the upstream Python* `choose_design(treatment_count)` *helper.*

Description

Returns a data frame with columns `s`, `k`, `blocks_per_replicate`, `plots_per_block`, `min_treatments`, `max_treatments` for every feasible (s, k) combination, where:

- `s` ranges from 5 to 15 (number of blocks per replicate)
- `k` = `ceil(v / s)` (actual plots per block)
- `min_treatments` = `4 * s`
- `max_treatments` = `s * MAX_K[s]` from the rotation table

Usage

```
propose_alpha_structures(treatment_count)

choose_design(treatment_count)
```

Arguments

`treatment_count` Number of treatments (must be >= 20 for alpha).

Value

A data frame. Empty if no feasible structures exist.

References

Patterson, H.D. & Williams, E.R. (1976). Biometrika, 63(1), 83-92. doi:10.1093/biomet/63.1.83

`register_design` *Register a design.*

Description

Register a design.

Usage

```
register_design(
  key,
  name,
  description,
  has_layout,
  has_blocks,
  default_params,
  param_specs,
  func,
  validate
)
```

Arguments

key	Design key (e.g. "cr_eq").
name	Human-readable design name.
description	One-paragraph description.
has_layout	Logical: does the design carry a layout view?
has_blocks	Logical: does the design have a block factor?
default_params	Named list of default parameters.
param_specs	List of parameter specifications (for documentation).
func	The design-generating function.
validate	The validation function for this design.

Value

Invisible TRUE; the function is registered for side effect.

---

seeded_randint *Return a random integer between a and b inclusive.*

Description

Return a random integer between a and b inclusive.

Usage

```
seeded_randint(rng, a, b)
```

Arguments

rng	An RNG returned by make_rng().
a	Lower bound (integer).
b	Upper bound (integer).

Value

An integer.

`seeded_sample` *Return k random items from* `items`*, without replacement.*

Description

Return k random items from `items`, without replacement.

Usage

```
seeded_sample(rng, items, k)
```

Arguments

`rng`	An RNG returned by `make_rng()`.
`items`	An atomic vector.
`k`	Number of items to draw.

Value

A new vector of length k.

`seeded_shuffle` *Return a new vector with* `items` *shuffled by* `rng`.

Description

The input is not modified. Mirrors CPython's `random.shuffle()` so the same seed produces the same permutation as the upstream Python implementation.

Usage

```
seeded_shuffle(rng, items)
```

Arguments

`rng`	An RNG returned by `make_rng()`.
`items`	An atomic vector (character, integer, numeric, logical).

Value

A new vector of the same type and length, shuffled.

total_units *Total number of experimental units.*

Description

Total number of experimental units.

Usage

```
total_units(x)
```

Arguments

x	An edgar_design object (or any object with a total_units method).

Value

An integer.

validate_design *Public dispatcher: validate design parameters by key.*

Description

Runs the design's validator against the supplied parameters. Returns a list with valid (logical), warnings (character vector). Hard validation failures raise an edgar_validation_error condition.

Usage

```
validate_design(type, ...)
```

Arguments

type	Design key.
...	Design-specific parameters.

Value

A list with elements valid (TRUE on success) and warnings (character vector, possibly empty).

`validate_treatment_count` *Validate treatment counts. Mirrors* `validate_treatment_count`.

Description

Validate treatment counts. Mirrors `validate_treatment_count`.

Usage

```
validate_treatment_count(value, min_val = 2L, max_val = 500L, context = "")
```

Arguments

`value`	Integer. Number of treatments.
`min_val`	Minimum (default 2).
`max_val`	Maximum (default 500).
`context`	Optional context string for the error message.

Value

Invisible TRUE; otherwise raises an `edgar_validation_error`.

`with_edgar_seed` *Run a block of code with an isolated seeded RNG, restoring the caller's* `.Random.seed` *afterwards. This is the safety net for any internal code that calls base R* `sample()` *or* `runif()` *directly; the* `make_rng()` / `seeded_shuffle()` *path does not touch the global state, so this wrapper is provided for completeness and for users who want to call base R sampling under a known seed.*

Description

Run a block of code with an isolated seeded RNG, restoring the caller's `.Random.seed` afterwards. This is the safety net for any internal code that calls base R `sample()` or `runif()` directly; the `make_rng()` / `seeded_shuffle()` path does not touch the global state, so this wrapper is provided for completeness and for users who want to call base R sampling under a known seed.

Usage

```
with_edgar_seed(seed, expr)
```

Arguments

`seed`	Integer seed.
`expr`	R expression to evaluate.

Value

The value of `expr`.

`write_edgar_csv` *Write a design to a CSV file.*

Description

The CSV format matches the upstream Python exporter: a metadata header block (`Edgar II`, `Experiment:`, `Designed:`, `Seed:`), a blank row, the column header row, and the data rows. Uses `\r\n` line terminators and `QUOTE_MINIMAL` quoting to match.

Usage

```
write_edgar_csv(result, file = "", include_header = TRUE)
```

Arguments

`result`	An `edgar_design` object.
`file`	Path to write to. Use `""` or `NULL` to return the CSV as a character scalar.
`include_header`	Whether to include the metadata header block.

Value

The CSV string (invisibly if `file` is given).

`write_edgar_json` *Write a design to a JSON file (or return as a string).*

Description

The JSON structure mirrors the upstream Python exporter: top-level fields `design_name`, `parameters`, `seed`, `generated_at` (ISO 8601), `total_units`, `rows`, `warnings`, and (if the design has a layout) `layout`, `layout_headers`, `layout_section_labels`.

Usage

```
write_edgar_json(result, file = "", pretty = TRUE)
```

Arguments

`result`	An `edgar_design` object.
`file`	Path to write to. Use `""` or `NULL` to return the JSON string.
`pretty`	Pretty-print with 2-space indent.

Details

Requires the `jsonlite` package (in Suggests).

Value

The JSON string (invisibly if `file` is given).

`write_edgar_xlsx` *Write a design to an XLSX file.*

Description

Requires the `openxlsx` package (in Suggests). The workbook contains a `Design, list` sheet with metadata and the row view, optionally a `Design, layout` sheet with the layout sections, and a `Parameters` sheet with the design parameters.

Usage

```
write_edgar_xlsx(result, file)
```

Arguments

`result`	An `edgar_design` object.
`file`	Path to write to.

Value

Invisibly `file`.

Index