Package 'DesignLibrary'

November 16, 2026

Title A Library of Declared Designs

Version 2.0.0

Description A lightweight library of research designs declared with 'DeclareDesign'. Designs live as self-contained R files; editable parameters are read from the design itself. Optional YAML metadata adds labels, categories, tips, and book aliases. Includes maintainer tools to index, audit, and bake small diagnosis previews, plus a bundled Shiny browser (`designs_shiny_app()`).

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0), DeclareDesign (>= 2.0.0)

Imports fabricatr (>= 2.0.0), randomizr (>= 2.0.0), estimatr (>= 2.0.0), rlang, yaml, utils, stats

Suggests shiny, generics, bslib, htmltools, markdown, DT, ggplot2, remotes, pkgdown, testthat (>= 3.0.0), knitr, rmarkdown, withr, margins, broom

Config/testthat/edition 3

Config/Needs/website pkgdown

URL https://github.com/DeclareDesign/DesignLibrary, https://declaredesign.org/r/designlibrary/

BugReports https://github.com/DeclareDesign/DesignLibrary/issues

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Macartan Humphreys [aut],
Graeme Blair [aut],
Jasper Cooper [aut],
Alexander Coppock [aut, cre],
Clara Bicalho [aut],
Neal Fultz [aut],
Lily Medina [aut]

Maintainer Alexander Coppock <acoppock@gmail.com>

Repository CRAN

Date/Publication 2026-11-16 07:10:02 UTC

Contents

audit_designs . . . 3
bake_previews . . . 4
binary_iv_designer . . . 4
block_cluster_two_arm_designer . . . 6
build_docs . . . 7
cluster_sampling_designer . . . 8
contributor_checklist . . . 9
copy_library_shiny . . . 9
designs_shiny_app . . . 10
design_info . . . 10
design_profile . . . 11
excluded_diagnosands . . . 12
factorial_designer . . . 12
get_args . . . 14
get_code . . . 15
get_design_code . . . 16
get_preview . . . 17
has_preview . . . 18
install_library_dependencies . . . 19
list_designs . . . 20
make_design . . . 20
make_index . . . 22
mediation_analysis_designer . . . 23
multi_arm_designer . . . 24
param_coverage_gaps . . . 25
param_coverage_report . . . 26
preferred_diagnosands . . . 26
pretest_posttest_designer . . . 27
print.design_library_args . . . 28
process_tracing_designer . . . 28
randomized_response_designer . . . 30
refresh_library . . . 31
regression_discontinuity_designer . . . 31
simple_two_arm_designer . . . 33
spillover_designer . . . 34
two_arm_attrition_designer . . . 35
two_arm_covariate_designer . . . 36
two_arm_designer . . . 37
two_by_two_designer . . . 38
write_audit_report . . . 39

Index **41**

`audit_designs` *Audit one or more designs*

Description

Checks that each file loads, exposes a design object, and that any YAML `params:` names are a subset of the design's modifiable parameters. Also records pre-design objects that are used by the design but missing from the redesignable parameter list (see `param_coverage_report()`).

Usage

```
audit_designs(
  designs = NULL,
  sims = 2,
  write_report = FALSE,
  report_dir = NULL
)
```

Arguments

`designs`	Character vector of ids/aliases, or `NULL` for all.
`sims`	Number of simulations for a short `diagnose_design()` (default 2). If `NULL`, skip the run check (load and params only).
`write_report`	If `TRUE`, write CSV + markdown under `tools/` in the source tree (default `FALSE`).
`report_dir`	Directory for reports; default `tools/` under package root.

Details

By default each design is diagnosed with a short `diagnose_design()` run (`sims = 2`). A design that loads but does not run is a failure. Pass `sims = NULL` only for a load-and-params scan while editing a single file; `refresh_library()` and the test suite always run the design.

Failures are collected per design and (by default) written to `tools/audit_report.csv`, `tools/audit_report.md`, and `tools/audit_report.txt` (FAIL/SKIP first, then OK) under the package root via `write_audit_report()`. Soft notes (undocumented params, coverage gaps) do not mark a design as failed.

Value

An object of class `design_library_audit`.

bake_previews *Bake compact diagnosis previews into inst/previews*

Description

Each design is baked independently. Failures are collected and returned; they do not abort the rest of the bake.

Usage

```
bake_previews(designs = NULL, sims = 100, seed = 343, dir = NULL)
```

Arguments

designs	Ids/aliases, or NULL for shiny-included designs.
sims	Number of simulations (package default is 100).
seed	Seed set before each design is built and diagnosed, so a preview does not depend on which other designs were baked with it. The caller's random number stream is restored on exit. NULL leaves the stream alone.
dir	Directory to write previews into. NULL (default) means inst/previews in the source tree; an installed package is refused.

Value

Invisibly, a list with paths (character) and failures (data frame with id and error).

binary_iv_designer *Create a binary instrumental variables design*

Description

Routes to make_design() with id "binary_iv": make_design("binary_iv", N = N, type_probs = type_probs, ...).

Usage

```
binary_iv_designer(
  N = 100,
  type_probs = c(1/3, 1/3, 1/3, 0),
  assignment_probs = c(0.5, 0.5, 0.5, 0.5),
  a_Y = 1,
  b_Y = 0,
  d_Y = 0,
  outcome_sd = 1,
  a = c(1, 0, 0, 0) * a_Y,
```

```
  b = rep(b_Y, 4),
  d = rep(d_Y, 4),
  args_to_fix = NULL
)
```

Arguments

N	Sample size.
type_probs	Shares of always-takers, never-takers, compliers, and defiers.
assignment_probs	Probability that Z = 1 for each type.
a_Y	Outcome intercept for always-takers, used in the default a.
b_Y	Effect of X on the outcome for every type, used in the default b.
d_Y	Direct effect of Z on the outcome for every type, used in the default d.
outcome_sd	Standard deviation of the outcome shock.
a, b, d	Length-4 outcome intercepts, effects of X, and direct effects of Z, by type.
args_to_fix	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

Units are always-takers, never-takers, compliers, or defiers, in that order in every length-4 argument. a_Y, b_Y, and d_Y fill the defaults of a, b, and d and are not passed on. Argument names match DesignLibrary 0.1's binary_iv_designer. See also make_design("encouragement") and make_design("instrumental_variables").

Value

A design object.

See Also

make_design()

Examples

```
binary_iv_designer(N = 200, b_Y = 0.5)
binary_iv_designer(type_probs = c(0.2, 0.2, 0.5, 0.1), b = c(0, 0, 1, -1))
```

`block_cluster_two_arm_designer` *Create a blocked and clustered two-arm design*

Description

Routes to `make_design()` with id `"block_cluster_two_arm"`: `make_design("block_cluster_two_arm", N_blocks = N_blocks, ...)`.

Usage

```
block_cluster_two_arm_designer(
  N = NULL,
  N_blocks = 20,
  N_clusters_in_block = 4,
  N_i_in_cluster = 10,
  sd = NULL,
  sd_block = 0.577,
  sd_cluster = 0.577,
  sd_i = 0.577,
  sd_i_0 = NULL,
  ate = 0.2,
  assignment_prob = 0.5,
  assignment_probs = NULL,
  args_to_fix = NULL,
  ...
)
```

Arguments

`N`, `sd`	Removed DesignLibrary 0.1 arguments. Supplying either is an error that names the arguments to set instead.
`N_blocks`	Number of blocks.
`N_clusters_in_block`	Number of clusters in each block.
`N_i_in_cluster`	Number of units in each cluster.
`sd_block`	Standard deviation of the block-level shock.
`sd_cluster`	Standard deviation of the cluster-level shock.
`sd_i`, `sd_i_0`	Individual-level shock. `sd_i_0` is an alias.
`ate`	Average treatment effect.
`assignment_prob`, `assignment_probs`	Assignment probability. A vector of per-block probabilities is not supported.
`args_to_fix`	Ignored.
`...`	Removed DesignLibrary 0.1 arguments, which error; any other argument is warned and dropped.

Details

A simplified signature relative to DesignLibrary 0.1: level sizes and three shock standard deviations, a scalar assignment probability, and `ate`. The DesignLibrary 0.1 arguments `N`, `sd`, `sd_i_1`, `rho`, `control_mean`, and `treatment_mean` are errors that name the argument to use instead. Other unused arguments (such as 0.1's `verbose`) are ignored with a warning, and per-block `assignment_probs` are an error.

Value

A design object.

See Also

`make_design()`

Examples

```
block_cluster_two_arm_designer(N_blocks = 4, N_clusters_in_block = 2, N_i_in_cluster = 5)
```

`build_docs` *Build the pkgdown site (Dropbox-safe)*

Description

On Windows Dropbox folders, `pkgdown::build_site()` / `build_article()` often fail in `xml2::write_html()` with "Invalid argument" / "Error closing file" when writing directly into `docs/`. This helper builds into a local temp directory, then copies the result into `docs/`.

Usage

```
build_docs(pkg = NULL, ...)
```

Arguments

`pkg`	Package root. Default: `find_package_root()` via `options(DesignLibrary.root=...)` or the current working directory.
`...`	Passed to `pkgdown::build_site()` (for example `devel = TRUE`).

Value

Invisibly, the path to `docs/`.

Examples

```
## Not run:
options(DesignLibrary.root = "C:/path/to/DesignLibrary")
build_docs()

## End(Not run)
```

`cluster_sampling_designer` *Create a two-stage cluster sampling design*

Description

Routes to `make_design()` with id `"cluster_sampling"`: `make_design("cluster_sampling", n_clusters_in_block = n_clusters_in_block, ...)`.

Usage

```
cluster_sampling_designer(
  N_blocks = 1,
  N_clusters_in_block = 1000,
  N_i_in_cluster = 50,
  n_clusters_in_block = 100,
  n_i_in_cluster = 10,
  icc = 0.2,
  args_to_fix = NULL
)
```

Arguments

`N_blocks`	Number of blocks.
`N_clusters_in_block`	Number of clusters in each block.
`N_i_in_cluster`	Number of individuals in each cluster.
`n_clusters_in_block`	Number of clusters sampled in each block.
`n_i_in_cluster`	Number of individuals sampled in each sampled cluster.
`icc`	Intra-cluster correlation of the latent outcome, in [0, 1].
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

Clusters are sampled within blocks, then individuals within sampled clusters, from a population drawn once when the design is built. Argument names match DesignLibrary 0.1's `cluster_sampling_designer`. For the book's budget-constrained version, see `make_design("cluster_random_sampling")`.

Value

A design object.

See Also

`make_design()`

Examples

```
cluster_sampling_designer(N_clusters_in_block = 200, n_clusters_in_block = 20)
```

---

contributor_checklist *Contributor checklist for a design*

Description

Used by `audit_designs()` and documented for authors.

Usage

```
contributor_checklist()
```

Value

Character vector of checklist items.

---

copy_library_shiny *Copy the bundled Shiny app to a standalone folder*

Description

Materializes `app.R` (and `www/` if present) so a Shiny Server / Posit Connect host can point at a folder. The folder relies on the installed `DesignLibrary` package for designs and helpers.

Usage

```
copy_library_shiny(dest = getwd(), overwrite = TRUE, package = "DesignLibrary")
```

Arguments

`dest`	Directory to write (created if needed). Default: working directory.
`overwrite`	If `TRUE`, replace `app.R` / `www` files (not `local.R`).
`package`	Package that ships the app; default `"DesignLibrary"`.

Details

Typical server workflow:

```
install.packages("DesignLibrary")
DesignLibrary::install_library_dependencies()
DesignLibrary::copy_library_shiny("/srv/shiny-server/designlibrary")
```

Existing `local.R` in `dest` is never overwritten.

Value

Invisibly, the normalized destination path.

`designs_shiny_app` *Launch the DesignLibrary Shiny browser*

Description

Launch the DesignLibrary Shiny browser

Usage

```
designs_shiny_app(...)
```

Arguments

`...`	Passed to `shiny::runApp()`.

Value

The value of `shiny::runApp()` (called for side effects).

Examples

```
if (interactive()) {
  designs_shiny_app()
}
```

`design_info` *Design metadata (YAML + defaults)*

Description

Printing uses short English prose. The underlying list is unchanged for programmatic use (`info$id`, `info$params`, `info$diagnosands`, `info$coupled`, etc.).

Usage

```
design_info(
  design = c("two_arm_simple", "two_arm_flexible", "three_arm", "two_by_two",
    "two_arm_block_cluster", "two_arm_with_blocks", "two_arm_attrition",
   "factorial_2x2x2", "multiarm_trial", "pretest_posttest", "randomized_response",
    "mediation_analysis", "binary_iv", "cluster_sampling", "factorial",
   "process_tracing_bayes", "regression_discontinuity_polynomial", "spillover",
   "two_arm_covariate", "audit_experiment", "audit_intervention", "baseline_over_N",
    "blocked_and_clustered", "bootstrapped", "cluster_random_sampling",
```

```
    "conditional_expectation", "conjoint", "covariate_adjustment",
    "declaration_using_declare", "design_10_2", "design_10_4", "diff_in_diff",
   "encouragement", "example_declaration", "factorial_2x2", "instrumental_variables",
    "italian_village", "italian_village_bayes", "italian_village_continued",
   "latent_variables", "lin_estimator", "linear_regression", "list_experiment",
   "list_or_direct_questions", "logit_probit_ols", "matching", "multi_site_studies",
    "multilevel", "multilevel_answer_strategies",
     "network_experiment",
   "pate_with_sampling", "population_estimands", "process_tracing", "random_forests",
   "randomization_inference", "randomized_saturation", "regression_discontinuity",
   "regression_discontinuity_fuzzy", "simple_random_sampling", "single_period_two_arm",
   "stepped_wedge", "structural_estimation", "subgroup_effects", "survey_nonresponse",
    "trust_game", "two_arm_bare_bones", "two_arm_block_rdss", "two_arm_rdss_1",
    "two_arm_rdss_2", "two_arm_trial_rdss", "two_arm_with_blocks_rdss",

   "two_outcome_model_a", "two_outcome_model_b", "uncertainty_over_effect_size",
   "village_campaign", "17.1", "17.2", "11.1", "18.5", "9.5", "15.3", "11.4", "17.5",
   "18.2", "10.1", "10.2", "10.4", "16.3", "18.8", "5.1", "18.7", "16.4", "9.1", "9.3",
   "9.2", "15.6", "18.3", "13.1", "17.3", "17.4", "11.5", "16.2", "19.4", "15.4",
   "15.5", "18.13", "4.1", "7.1", "16.1b", "19.1", "9.7", "18.12", "16.5", "16.6",
   "15.1", "18.11", "18.10", "19.2", "18.6", "15.2", "17.6", "11.3", "18.4", "2.1",
    "13.2", "18.1",
     "2.2", "10.3a", "10.3b", "11.2", "12.1a")
)
```

Arguments

design	Design id or book alias.

Value

A named list of metadata with class design_library_info.

Examples

```
design_info("two_arm_simple")
```

---

design_profile *Character profile for a design (same prose as printing design_info())*

Description

Character profile for a design (same prose as printing design_info())

Usage

```
design_profile(design)
```

Arguments

design	Design id or book alias.

Value

Character scalar.

---

excluded_diagnosands *Diagnosands excluded from display by YAML*

Description

Tokens like `-bias` in `diagnosands:` remove that name from the Shiny diagnosand list for the design.

Usage

```
excluded_diagnosands(design)
```

Arguments

design	Design id or book alias.

Value

Character vector (possibly empty).

---

factorial_designer *Create a 2^k factorial design*

Description

Routes to `make_design()` with id `"factorial"`: `make_design("factorial", N = N, k = k, ...)`.

Usage

```
factorial_designer(
  N = 256,
  k = 3,
  outcome_means = rep(0, 2^k),
  sd = 1,
  outcome_sds = rep(sd, 2^k),
  assignment_probs = rep(0.5, k),
  outcome_name = "Y",
  treatment_names = NULL,
  args_to_fix = NULL
)
```

Arguments

N	Sample size.
k	Number of factors, an integer of at least 2.
outcome_means	Mean outcome in each of the 2^k treatment combinations.
sd	Standard deviation of every potential outcome, used when outcome_sds is not supplied.
outcome_sds	Standard deviation of the outcome in each of the 2^k treatment combinations.
assignment_probs	Probability of assignment to each factor (length k).
outcome_name	Name of the outcome variable.
treatment_names	Names of the k factors. NULL gives T1, ..., Tk.
args_to_fix	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

Each of k factors is assigned with its own probability, and each of the 2^k treatment combinations has its own normally distributed potential outcome. The inquiries are the overall average, the average effect of each factor, and every interaction, each averaged with equal weight over the conditions of the other factors. The estimator regresses the outcome on the full interaction of the treatments centered at 0.5, weighted by the inverse probability of the assigned combination, so the intercept estimates the overall average and each coefficient an average effect. Argument names match DesignLibrary 0.1's `factorial_designer`.

`outcome_means` and `outcome_sds` follow the row order of `expand.grid(rep(list(c(0, 1)), k))`, in which the first factor varies fastest.

Value

A design object.

See Also

`make_design()`

Examples

```
factorial_designer(k = 3, assignment_probs = c(1/2, 1/4, 1/8),
                   outcome_means = c(0, 0, 0, 0, 0, 0, 0, 4))
```

`get_args` *Editable parameters for a design*

Description

Reads parameters from the design object. Optional YAML `params:` entries only add tips (and never invent new parameter names).

Usage

```
get_args(
  design = c("two_arm_simple", "two_arm_flexible", "three_arm", "two_by_two",
    "two_arm_block_cluster", "two_arm_with_blocks", "two_arm_attrition",
   "factorial_2x2x2", "multiarm_trial", "pretest_posttest", "randomized_response",
    "mediation_analysis", "binary_iv", "cluster_sampling", "factorial",
    "process_tracing_bayes", "regression_discontinuity_polynomial", "spillover",
   "two_arm_covariate", "audit_experiment", "audit_intervention", "baseline_over_N",
    "blocked_and_clustered", "bootstrapped", "cluster_random_sampling",

    "conditional_expectation", "conjoint", "covariate_adjustment",
    "declaration_using_declare", "design_10_2", "design_10_4", "diff_in_diff",
   "encouragement", "example_declaration", "factorial_2x2", "instrumental_variables",
    "italian_village", "italian_village_bayes", "italian_village_continued",
    "latent_variables", "lin_estimator", "linear_regression", "list_experiment",
   "list_or_direct_questions", "logit_probit_ols", "matching", "multi_site_studies",
    "multilevel", "multilevel_answer_strategies",
     "network_experiment",
   "pate_with_sampling", "population_estimands", "process_tracing", "random_forests",
   "randomization_inference", "randomized_saturation", "regression_discontinuity",
   "regression_discontinuity_fuzzy", "simple_random_sampling", "single_period_two_arm",
   "stepped_wedge", "structural_estimation", "subgroup_effects", "survey_nonresponse",
    "trust_game", "two_arm_bare_bones", "two_arm_block_rdss", "two_arm_rdss_1",
    "two_arm_rdss_2", "two_arm_trial_rdss", "two_arm_with_blocks_rdss",

   "two_outcome_model_a", "two_outcome_model_b", "uncertainty_over_effect_size",
   "village_campaign", "17.1", "17.2", "11.1", "18.5", "9.5", "15.3", "11.4", "17.5",
   "18.2", "10.1", "10.2", "10.4", "16.3", "18.8", "5.1", "18.7", "16.4", "9.1", "9.3",
    "9.2", "15.6", "18.3", "13.1", "17.3", "17.4", "11.5", "16.2", "19.4", "15.4",
   "15.5", "18.13", "4.1", "7.1", "16.1b", "19.1", "9.7", "18.12", "16.5", "16.6",
   "15.1", "18.11", "18.10", "19.2", "18.6", "15.2", "17.6", "11.3", "18.4", "2.1",
    "13.2", "18.1",
     "2.2", "10.3a", "10.3b", "11.2", "12.1a")
)
```

Arguments

design	Design id or book alias.

Details

The kind column is "scalar", "vector", "list", "data", or "function". shiny is TRUE for scalar and short-vector parameters that the browser can edit as text. Data frames, matrices, long vectors, lists, and functions stay redesignable in R (make_design(..., dataset = ...), make_design(..., Y = ...)) but are not Shiny controls.

The default column holds the values themselves, so it is a list column and may contain functions and lists; the printed table shows value_str in its place.

Value

A design_library_args data frame with name, default, value_str, tip, kind, and shiny.

Examples

```
get_args("two_arm_simple")
```

---

get_code *Code for a design: simple* make_design() *call and/or full source*

Description

Returns a list with $simple (a one-line make_design() call) and $full (the design file source, prefixed with library() calls for YAML packages: when any are listed). Printing uses cat() on $full so the console output is copy-paste ready. Use .style = "simple" to print the one-liner instead, or .style = "both" for the one-liner followed by the source. Programmatic access is unchanged: get_code(id)$simple, get_code(id)$full.

Usage

```
get_code(
  .design = c("two_arm_simple", "two_arm_flexible", "three_arm", "two_by_two",
    "two_arm_block_cluster", "two_arm_with_blocks", "two_arm_attrition",
   "factorial_2x2x2", "multiarm_trial", "pretest_posttest", "randomized_response",
    "mediation_analysis", "binary_iv", "cluster_sampling", "factorial",
   "process_tracing_bayes", "regression_discontinuity_polynomial", "spillover",
   "two_arm_covariate", "audit_experiment", "audit_intervention", "baseline_over_N",
    "blocked_and_clustered", "bootstrapped", "cluster_random_sampling",

    "conditional_expectation", "conjoint", "covariate_adjustment",
    "declaration_using_declare", "design_10_2", "design_10_4", "diff_in_diff",
   "encouragement", "example_declaration", "factorial_2x2", "instrumental_variables",
    "italian_village", "italian_village_bayes", "italian_village_continued",
   "latent_variables", "lin_estimator", "linear_regression", "list_experiment",
```

```
    "list_or_direct_questions", "logit_probit_ols", "matching", "multi_site_studies",
     "multilevel", "multilevel_answer_strategies",
      "network_experiment",
    "pate_with_sampling", "population_estimands", "process_tracing", "random_forests",
    "randomization_inference", "randomized_saturation", "regression_discontinuity",
    "regression_discontinuity_fuzzy", "simple_random_sampling", "single_period_two_arm",
    "stepped_wedge", "structural_estimation", "subgroup_effects", "survey_nonresponse",
     "trust_game", "two_arm_bare_bones", "two_arm_block_rdss", "two_arm_rdss_1",
     "two_arm_rdss_2", "two_arm_trial_rdss", "two_arm_with_blocks_rdss",

    "two_outcome_model_a", "two_outcome_model_b", "uncertainty_over_effect_size",
    "village_campaign", "17.1", "17.2", "11.1", "18.5", "9.5", "15.3", "11.4", "17.5",
    "18.2", "10.1", "10.2", "10.4", "16.3", "18.8", "5.1", "18.7", "16.4", "9.1", "9.3",
     "9.2", "15.6", "18.3", "13.1", "17.3", "17.4", "11.5", "16.2", "19.4", "15.4",
    "15.5", "18.13", "4.1", "7.1", "16.1b", "19.1", "9.7", "18.12", "16.5", "16.6",
    "15.1", "18.11", "18.10", "19.2", "18.6", "15.2", "17.6", "11.3", "18.4", "2.1",
     "13.2", "18.1",
      "2.2", "10.3a", "10.3b", "11.2", "12.1a"),
  .style = c("full", "simple", "both"),
  ...
)
```

Arguments

`.design`	Design id or book alias.
`.style`	`"full"`, `"simple"`, or `"both"`. Controls which snippet `print()` and `as.character()` show. `$simple` and `$full` are always both present.
`...`	Optional parameter values included in the simple snippet.

Value

A list with `simple` and `full`, class `design_library_code`.

Examples

```
get_code("two_arm_simple")
get_code("two_arm_simple")$simple
```

`get_design_code` *Defunct helpers from DesignLibrary 0.1*

Description

`get_design_code()` read the code attribute that DesignLibrary 0.1 designers attached to each design. Designs no longer carry that attribute: each designer builds a library design, and `get_code()` returns that design's source from its id. `match.call.defaults()` was a helper the 0.1 designers used to write that code; use `match.call()` and `formals()`. Both error with a message saying so.

Usage

```
get_design_code(...)

match.call.defaults(...)
```

Arguments

...	(optional) Ignored.

Value

Never returns; always raises an error.

---

get_preview *Load a baked diagnosis preview*

Description

Reads `inst/previews/<id>.rds` written by `bake_previews()` / `refresh_library()`.

Usage

```
get_preview(
  design = c("two_arm_simple", "two_arm_flexible", "three_arm", "two_by_two",
    "two_arm_block_cluster", "two_arm_with_blocks", "two_arm_attrition",
   "factorial_2x2x2", "multiarm_trial", "pretest_posttest", "randomized_response",
    "mediation_analysis", "binary_iv", "cluster_sampling", "factorial",
   "process_tracing_bayes", "regression_discontinuity_polynomial", "spillover",
   "two_arm_covariate", "audit_experiment", "audit_intervention", "baseline_over_N",
    "blocked_and_clustered", "bootstrapped", "cluster_random_sampling",

    "conditional_expectation", "conjoint", "covariate_adjustment",
    "declaration_using_declare", "design_10_2", "design_10_4", "diff_in_diff",
   "encouragement", "example_declaration", "factorial_2x2", "instrumental_variables",
    "italian_village", "italian_village_bayes", "italian_village_continued",
   "latent_variables", "lin_estimator", "linear_regression", "list_experiment",
   "list_or_direct_questions", "logit_probit_ols", "matching", "multi_site_studies",
    "multilevel", "multilevel_answer_strategies",
     "network_experiment",
   "pate_with_sampling", "population_estimands", "process_tracing", "random_forests",
   "randomization_inference", "randomized_saturation", "regression_discontinuity",
   "regression_discontinuity_fuzzy", "simple_random_sampling", "single_period_two_arm",
   "stepped_wedge", "structural_estimation", "subgroup_effects", "survey_nonresponse",
    "trust_game", "two_arm_bare_bones", "two_arm_block_rdss", "two_arm_rdss_1",
    "two_arm_rdss_2", "two_arm_trial_rdss", "two_arm_with_blocks_rdss",

   "two_outcome_model_a", "two_outcome_model_b", "uncertainty_over_effect_size",
```

```
    "village_campaign", "17.1", "17.2", "11.1", "18.5", "9.5", "15.3", "11.4", "17.5",
    "18.2", "10.1", "10.2", "10.4", "16.3", "18.8", "5.1", "18.7", "16.4", "9.1", "9.3",
     "9.2", "15.6", "18.3", "13.1", "17.3", "17.4", "11.5", "16.2", "19.4", "15.4",
    "15.5", "18.13", "4.1", "7.1", "16.1b", "19.1", "9.7", "18.12", "16.5", "16.6",
    "15.1", "18.11", "18.10", "19.2", "18.6", "15.2", "17.6", "11.3", "18.4", "2.1",
      "13.2", "18.1",
       "2.2", "10.3a", "10.3b", "11.2", "12.1a")
)
```

Arguments

design	Design id or book alias.

Value

A list with at least id, sims, and summary, or NULL if missing.

---

has_preview *Whether a baked preview exists for a design*

Description

Whether a baked preview exists for a design

Usage

```
has_preview(design)
```

Arguments

design	Design id or book alias.

Value

Logical.

install_library_dependencies *Install DesignLibrary system dependencies*

Description

Installs package Depends and Imports (and, by default, Suggests needed for the Shiny browser), plus any extra packages declared in design YAML `packages:` fields. Everything comes from CRAN except `interference` (used by `network_experiment`), which comes from GitHub via `remotes::install_github()`. An installed `DeclareDesign`, `fabricatr`, `estimatr`, or `randomizr` older than 2.0.0 is upgraded from CRAN.

Usage

```
install_library_dependencies(
  include_shiny = TRUE,
  include_suggests = FALSE,
  ask = FALSE,
  upgrade = "never",
  verbose = TRUE,
  dry_run = FALSE
)
```

Arguments

include_shiny	If `TRUE` (default), also install Shiny Suggests (`shiny`, `bslib`, `htmltools`, `markdown`).
include_suggests	If `TRUE`, install all DESCRIPTION Suggests. Ignored when `include_shiny` already covers the Shiny set unless you want test/doc packages too.
ask	Passed to `remotes::install_github()`.
upgrade	Passed to `remotes::install_github()`.
verbose	Print progress messages.
dry_run	If `TRUE`, install nothing and return the plan: `to_install` names the packages that would be installed or upgraded.

Details

Typical server workflow:

```
install.packages("DesignLibrary")
DesignLibrary::install_library_dependencies()
DesignLibrary::copy_library_shiny("/path/to/shiny-app")
```

Value

Invisibly, a list with `installed`, `already_ok`, `failed`, and `to_install`. If a package that was already loaded is upgraded, restart R before using it; a message says so.

list_designs *List designs in the library*

Description

Returns a data frame of design metadata. Printing is a grouped list of `id (label)` lines, starting with a short getting-started sequence, then other templates, RDSS, and remaining designs. Default listing is metadata-only (baked library index plus a live-file overlay). The `params` column is a comma-separated name string from that index (YAML `params:` keys plus pre-design assignment names); designs are not evaluated. Use `design_info()`, `get_args()`, or `as.data.frame(list_designs(discover_params = TRUE))` for redesignable parameters from a loaded design. Row order matches print order.

Usage

```
list_designs(shiny_only = FALSE, discover_params = FALSE, list_all = FALSE)
```

Arguments

`shiny_only`	If `TRUE`, only designs with `include_in_shiny: true` (the default when the field is omitted).
`discover_params`	If `TRUE`, load each design once to list redesignable parameters (overrides index names). Default `FALSE` uses baked index names and does not evaluate designs.
`list_all`	If `FALSE` (default), printing shows the full getting-started sequence and up to 10 designs in each remaining group. If `TRUE`, printing lists every design.

Value

A data frame with class `design_library_list`.

Examples

```
list_designs()
list_designs(shiny_only = TRUE)
```

make_design *Build a design, optionally with redesigned parameters*

Description

Loads the declared design at its defaults. Named ... values are applied with DeclareDesign::redesign(). The design object is the source of truth for which names can be changed; see get_args(). Vector parameters are wrapped so redesign() replaces the whole vector instead of sweeping. A list-valued parameter (kind = "list") follows the same rule read structurally: a bare list replaces the parameter whole, and a list whose every element is itself a list sweeps over those lists, one design each. So make_design("conjoint", levels_list = list(a = "x", b = "y")) builds one design, and make_design("conjoint", levels_list = list(l1, l2)) builds two. YAML coupled: drivers (for example m_arms) emit a message() when dependents do not match in length; redesign() still runs.

Usage

```
make_design(
  .design = c("two_arm_simple", "two_arm_flexible", "three_arm", "two_by_two",
    "two_arm_block_cluster", "two_arm_with_blocks", "two_arm_attrition",
   "factorial_2x2x2", "multiarm_trial", "pretest_posttest", "randomized_response",
    "mediation_analysis", "binary_iv", "cluster_sampling", "factorial",
   "process_tracing_bayes", "regression_discontinuity_polynomial", "spillover",
   "two_arm_covariate", "audit_experiment", "audit_intervention", "baseline_over_N",
    "blocked_and_clustered", "bootstrapped", "cluster_random_sampling",

    "conditional_expectation", "conjoint", "covariate_adjustment",
    "declaration_using_declare", "design_10_2", "design_10_4", "diff_in_diff",
   "encouragement", "example_declaration", "factorial_2x2", "instrumental_variables",
    "italian_village", "italian_village_bayes", "italian_village_continued",
   "latent_variables", "lin_estimator", "linear_regression", "list_experiment",
   "list_or_direct_questions", "logit_probit_ols", "matching", "multi_site_studies",
    "multilevel", "multilevel_answer_strategies",
     "network_experiment",
   "pate_with_sampling", "population_estimands", "process_tracing", "random_forests",
   "randomization_inference", "randomized_saturation", "regression_discontinuity",
   "regression_discontinuity_fuzzy", "simple_random_sampling", "single_period_two_arm",
   "stepped_wedge", "structural_estimation", "subgroup_effects", "survey_nonresponse",
    "trust_game", "two_arm_bare_bones", "two_arm_block_rdss", "two_arm_rdss_1",
    "two_arm_rdss_2", "two_arm_trial_rdss", "two_arm_with_blocks_rdss",

   "two_outcome_model_a", "two_outcome_model_b", "uncertainty_over_effect_size",
   "village_campaign", "17.1", "17.2", "11.1", "18.5", "9.5", "15.3", "11.4", "17.5",
   "18.2", "10.1", "10.2", "10.4", "16.3", "18.8", "5.1", "18.7", "16.4", "9.1", "9.3",
   "9.2", "15.6", "18.3", "13.1", "17.3", "17.4", "11.5", "16.2", "19.4", "15.4",
   "15.5", "18.13", "4.1", "7.1", "16.1b", "19.1", "9.7", "18.12", "16.5", "16.6",
   "15.1", "18.11", "18.10", "19.2", "18.6", "15.2", "17.6", "11.3", "18.4", "2.1",
    "13.2", "18.1",
     "2.2", "10.3a", "10.3b", "11.2", "12.1a"),
  ...
)
```

Arguments

`.design`	Design id or book alias. Defaults to `"two_arm_simple"` (or the first installed design). Tab-completion offers the full library list. Dotted so that a design parameter such as `d` cannot partially match it.
`...`	Named parameter values passed to `redesign()`, and optionally `.expand`, which `redesign()` uses to choose between crossing swept parameters (`TRUE`, the default) and zipping them (`FALSE`).

Details

In RStudio, Positron, and other tools that complete from formals, typing `make_design("` and pressing Tab lists installed design ids (and aliases). See `list_designs()` for the same catalogue in the console.

Value

A design object (or a list of designs if a parameter is a vector and `redesign()` expands).

Examples

```
make_design()
make_design("two_arm_simple", b = 0.5)
make_design("2.1", b = 0.5)  # book alias
```

`make_index` *Build an in-memory index of all designs*

Description

Shared by `list_designs()` (metadata-only default) and maintainer tools. When `use_cache = TRUE`, reads the baked `inst/library_index` artifact if live `.R` filenames match, and YAML-parses only extra, missing, or newer files so a local `inst/designs/my_design.R` appears without refresh.

Usage

```
make_index(use_cache = TRUE)
```

Arguments

`use_cache`	If `TRUE` (default), use the baked index with a live-file overlay. If `FALSE`, parse YAML from every design file (used by `refresh_library()`).

Value

A data frame (same columns as `list_designs()`, plus description). `params` is a comma-separated name string from the baked index, or from YAML `params:` keys plus pre-design assignment names for overlay files.

`mediation_analysis_designer` *Create a mediation analysis design*

Description

Routes to `make_design()` with id `"mediation_analysis"`: `make_design("mediation_analysis", N = N, a = a, ...)`.

Usage

```
mediation_analysis_designer(
  N = 200,
  a = 1,
  b = 0.4,
  c = 0,
  d = 0.5,
  rho = 0,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`a`	Effect of treatment on the latent index for the mediator.
`b`	Effect of the mediator on the outcome when Z = 0.
`c`	Interaction of the mediator and treatment in the outcome.
`d`	Direct effect of treatment on the outcome when M = 0.
`rho`	Correlation between mediator and outcome shocks (-1 to 1).
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Value

A design object.

See Also

`make_design()`

Examples

```
mediation_analysis_designer(N = 40, a = 0.5, d = 0.2)
```

`multi_arm_designer` *Create a multi-arm design*

Description

Routes to `make_design()` with id `"multiarm_trial"`: `make_design("multiarm_trial", N = N, m_arms = m_arms, ...)`. Defaults for `outcome_means`, `outcome_sds`, and `conditions` are evaluated after `m_arms`, so `multi_arm_designer(m_arms = 4)` expands those vectors before calling `make_design()`. `make_design("multiarm_trial", m_arms = 4)` does the same inside the design: constant vectors are repeated and conditions `1..k` become `1..m_arms`. Any other wrong-length vector errors. Inquiries are the differences in arm means for every pair of arms, `ate_Y_j_k`, and each is estimated by a difference in means between its two arms (estimator `"DIM (Z_j - Z_k)"`). The contrasts between two treatment arms are usually the least powered. Library-file knobs match these formals.

Usage

```
multi_arm_designer(
  N = 90,
  m_arms = 3,
  outcome_means = rep(0, m_arms),
  sd_i = 1,
  outcome_sds = rep(0, m_arms),
  conditions = seq_len(m_arms),
  Y = NULL,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`m_arms`	Number of arms. May vary. Must match `length(outcome_means)` unless `outcome_means` is a scalar (then recycled).
`outcome_means`	Average outcome in each arm. Default `rep(0, m_arms)`.
`sd_i`	Standard deviation of the individual-level shock.
`outcome_sds`	Extra standard deviation in each arm. Default `rep(0, m_arms)`.
`conditions`	Assignment conditions, length `m_arms`. Default `seq_len(m_arms)`.
`Y`	Optional outcome function of `Z`, `u`, and `outcome_sds`. The library default is used when `Y` is `NULL`.
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Value

A design object.

See Also

make_design()

Examples

```
make_design("multiarm_trial", N = 90)
multi_arm_designer(m_arms = 4)
multi_arm_designer(m_arms = 4, outcome_means = c(0, 0.5, 1, 2))
```

---

param_coverage_gaps *Objects declared before design, used by it, but missing from design params*

Description

Runs the design, reads `discover_design_params()`, and compares to top-level assignments before `design <-`. Flags names that are used in the design body (or seen by DeclareDesign's object finder) but not in the redesignable parameter list.

Usage

```
param_coverage_gaps(design, include_steps = FALSE)
```

Arguments

`design`	Design id/alias, or a parsed design list from `parse_design_file()`.
`include_steps`	If `TRUE`, also report design pieces. Default `FALSE` (audit-friendly).

Details

Design steps (MIDA pieces built with `declare_*`, etc.) are not parameters and are omitted unless `include_steps = TRUE`. Functions assigned before `design <-` are R-only parameters.

Value

Data frame of gaps (possibly empty) with columns `id`, `name`, `type`, `atomic`, `used_in_code`, `in_finder`, `in_params`.

param_coverage_report *Report declared-before-design objects missing from design parameters*

Description

For each library design: evaluate the file, read redesignable parameters from the design object, and list top-level objects that are used by the design but do not appear in that parameter list.

Usage

```
param_coverage_report(
  designs = NULL,
  atomic_only = FALSE,
  include_steps = FALSE
)
```

Arguments

designs	Ids/aliases, or NULL for all.
atomic_only	If TRUE, only report atomic gaps (likely should be redesignable). If FALSE, also report other non-step gaps (e.g. data frames).
include_steps	If TRUE, also report design pieces.

Value

A data frame (class design_library_param_coverage) of gaps.

preferred_diagnosands *Preferred diagnosands declared in a design's YAML*

Description

Reads optional diagnosands: from the design frontmatter (for example diagnosands: [rmse, bias] or diagnosands: rmse, -bias, power). Positive names are preferred defaults for diagnosis and redesign display. Names prefixed with - (e.g. -bias) are exclusions and are omitted here; see excluded_diagnosands().

Usage

```
preferred_diagnosands(design)
```

Arguments

design	Design id or book alias.

Value

Character vector (possibly empty).

Examples

```
preferred_diagnosands("two_arm_simple")
```

`pretest_posttest_designer` *Create a pretest-posttest design*

Description

Routes to `make_design()` with id `"pretest_posttest"`: `make_design("pretest_posttest", N = N, ate = ate, ...)`.

Usage

```
pretest_posttest_designer(
  N = 100,
  ate = 0.25,
  sd_1 = 1,
  sd_2 = 1,
  rho = 0.5,
  attrition_rate = 0.1,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`ate`	Average treatment effect.
`sd_1`	Standard deviation of the pretest shock.
`sd_2`	Standard deviation of the posttest shock.
`rho`	Correlation between pretest and posttest shocks (-1 to 1).
`attrition_rate`	Share of units not observed at the posttest.
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Value

A design object.

See Also

`make_design()`

Examples

```
pretest_posttest_designer(N = 40, attrition_rate = 0.2)
```

`print.design_library_args` *Print the parameter table from get_args()*

Description

The `default` column holds the values themselves, and a function or a list has no one-line form, so the printed table shows `value_str` in its place.

Usage

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

Arguments

`x`	A `design_library_args` table.
`...`	Unused.

Value

`x`, invisibly.

`process_tracing_designer` *Create a process tracing design with two clues*

Description

Routes to `make_design()` with id `"process_tracing_bayes"`: `make_design("process_tracing_bayes", prior_H = prior_H, ...)`.

Usage

```
process_tracing_designer(
  N = 100,
  prob_X = 0.5,
  process_proportions = c(0.25, 0.25, 0.25, 0.25),
  prior_H = 0.5,
  p_E1_H = 0.8,
  p_E1_not_H = 0.2,
  p_E2_H = 0.3,
```

```
  p_E2_not_H = 0,
  cor_E1E2_H = 0,
  cor_E1E2_not_H = 0,
  label_E1 = "Straw in the Wind",
  label_E2 = "Smoking Gun",
  args_to_fix = NULL
)
```

Arguments

`N`	Number of cases in the population.
`prob_X`	Probability that X = 1.
`process_proportions`	Shares of the four causal processes: X causes Y, Y regardless, X causes not Y, and not Y regardless.
`prior_H`	Prior probability that X caused Y.
`p_E1_H`, `p_E1_not_H`	Probability of clue 1 if X did and did not cause Y.
`p_E2_H`, `p_E2_not_H`	Probability of clue 2 if X did and did not cause Y.
`cor_E1E2_H`, `cor_E1E2_not_H`	Correlation of the two clues if X did and did not cause Y.
`label_E1`, `label_E2`	Ignored unless left at their defaults.
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

One case with X = 1 and Y = 1 is selected, two clues are observed, and the posterior that X caused Y is updated by Bayes' rule. Argument names match DesignLibrary 0.1's `process_tracing_designer`. The estimator labels are fixed at "Straw in the Wind" and "Smoking Gun", so `label_E1` and `label_E2` other than those warn and are ignored. For the book's CausalQueries version, see `make_design("process_tracing")`.

Value

A design object.

See Also

`make_design()`

Examples

```
process_tracing_designer(prior_H = 0.3, p_E2_H = 0.5)
```

`randomized_response_designer` *Create a randomized response design*

Description

Routes to `make_design()` with id `"randomized_response"`: `make_design("randomized_response", N = N, ...)`.

Usage

```
randomized_response_designer(
  N = 1000,
  prob_forced_yes = 0.6,
  prevalence_rate = 0.1,
  withholding_rate = 0.5,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`prob_forced_yes`	Probability a respondent is forced to answer yes.
`prevalence_rate`	Share of units holding the sensitive trait.
`withholding_rate`	Share of trait holders who deny it under direct questioning.
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Value

A design object.

See Also

`make_design()`

Examples

```
randomized_response_designer(N = 40, prevalence_rate = 0.2)
```

refresh_library *Refresh the library (maintainer one-stop)*

Description

Runs the contributor-facing checks and refreshes baked artifacts: index, audits (including a short diagnosis of every design), and diagnosis previews (`sims = 100` by default). Audit and preview failures are reported at the end; they do not abort the refresh.

Usage

```
refresh_library(sims = 100, designs = NULL, seed = 343)
```

Arguments

`sims`	Preview simulations (default 100).
`designs`	Optional subset; default all for audit, shiny-on for previews.
`seed`	Passed to `bake_previews()`.

Details

A refresh of a subset leaves the rest of the library as it was: the audit and refresh reports under `tools/` are rewritten only by a full refresh, and index rows outside the subset keep the `params` column already written.

Value

A list with `index`, `audit`, `previews`, `ok_ids`, `preview_failures`, and `report`.

regression_discontinuity_designer *Create a regression discontinuity design with a polynomial fit*

Description

Routes to `make_design()` with id `"regression_discontinuity_polynomial"`: `make_design("regression_discontinu N = N, tau = tau, ...)`.

Usage

```
regression_discontinuity_designer(
  N = 1000,
  tau = 0.15,
  outcome_sd = 0.1,
  cutoff = 0.5,
  bandwidth = 0.5,
  control_coefs = c(0.5, 0.5),
  treatment_coefs = c(-5, 1),
  poly_reg_order = 4,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`tau`	Jump in the outcome at the cutoff.
`outcome_sd`	Standard deviation of the outcome shock.
`cutoff`	Cutoff on the running variable, in (0, 1).
`bandwidth`	Largest distance from the cutoff for a unit to be kept.
`control_coefs`	Polynomial coefficients of the control outcome in the running variable.
`treatment_coefs`	Polynomial coefficients of the treated outcome in the running variable.
`poly_reg_order`	Order of the polynomial in the estimator.
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

Treatment is assigned when a uniform running variable passes `cutoff`, and each potential outcome is a polynomial in the running variable. Argument names match DesignLibrary 0.1's `regression_discontinuity_designer`. For the book's local-linear version, see `make_design("regression_discontinuity")`.

Value

A design object.

See Also

`make_design()`

Examples

```
regression_discontinuity_designer(N = 500, poly_reg_order = 2)
```

simple_two_arm_designer *Defunct designer aliases from DesignLibrary 0.1*

Description

DesignLibrary 0.1 deprecated these aliases in favor of the designers they called. DesignLibrary 2.0.0 removes them. Each errors with a message naming its replacement, which takes the same arguments.

Usage

```
simple_two_arm_designer(...)

simple_factorial_designer(...)

simple_iv_designer(...)

simple_spillover_designer(...)
```

Arguments

...	(optional) Ignored.

Details

Removed	Use instead
simple_two_arm_designer()	two_arm_designer()
simple_factorial_designer()	two_by_two_designer()
simple_iv_designer()	binary_iv_designer()
simple_spillover_designer()	spillover_designer()

Value

Never returns; always raises an error.

`spillover_designer` *Create a two-arm design with spillovers within groups*

Description

Routes to `make_design()` with id `"spillover"`: `make_design("spillover", N_groups = N_groups, gamma = gamma, ...)`.

Usage

```
spillover_designer(
  N_groups = 80,
  N_i_group = 3,
  sd_i = 0.2,
  gamma = 2,
  args_to_fix = NULL
)
```

Arguments

`N_groups`	Number of groups.
`N_i_group`	Number of units in each group.
`sd_i`	Standard deviation of the individual-level shock.
`gamma`	Curvature of the outcome in the group's treated share.
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

A unit's outcome depends on the share of its group that is treated, raised to the power `gamma`. Argument names match DesignLibrary 0.1's `spillover_designer`. For a design that randomizes saturation, see `make_design("randomized_saturation")`.

Value

A design object.

See Also

`make_design()`

Examples

```
spillover_designer(N_groups = 40, gamma = 1)
```

`two_arm_attrition_designer` *Create a two-arm design with attrition*

Description

Routes to `make_design()` with id `"two_arm_attrition"`: `make_design("two_arm_attrition", N = N, a_R = a_R, ...)`.

Usage

```
two_arm_attrition_designer(
  N = 100,
  a_R = 0,
  b_R = 1,
  a_Y = 0,
  b_Y = 1,
  rho = 0,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`a_R`	Constant in the equation relating treatment to reporting.
`b_R`	Slope relating treatment to reporting.
`a_Y`	Constant in the equation relating treatment to the outcome.
`b_Y`	Slope relating treatment to the outcome.
`rho`	Correlation between reporting and outcome shocks (0 to 1).
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Value

A design object.

See Also

`make_design()`

Examples

```
two_arm_attrition_designer(N = 40, b_R = 0.5)
```

`two_arm_covariate_designer` *Create a two-arm design with a possibly prognostic covariate*

Description

Routes to `make_design()` with id `"two_arm_covariate"`: `make_design("two_arm_covariate", N = N, rho_WY = rho_WY, ...)`.

Usage

```
two_arm_covariate_designer(
  N = 100,
  prob = 0.5,
  control_mean = 0,
  sd = 1,
  ate = 1,
  h = 0,
  treatment_mean = NULL,
  rho_WY = 0,
  rho_WZ = 0,
  args_to_fix = NULL
)
```

Arguments

`N`	Sample size.
`prob`	Probability of assignment to treatment.
`control_mean`	Average outcome in control.
`sd`	Standard deviation of the outcome shock.
`ate`	Average treatment effect.
`h`	Heterogeneity of the treatment effect by `W`.
`treatment_mean`	Average outcome in treatment. If supplied, overrides `ate` (`ate` becomes `treatment_mean - control_mean`).
`rho_WY`	Correlation between `W` and the outcome shock (-1 to 1).
`rho_WZ`	Correlation between `W` and the latent assignment variable (-1 to 1).
`args_to_fix`	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

The covariate `W` can predict the outcome (`rho_WY`), moderate the effect (`h`), and predict assignment (`rho_WZ`). With `rho_WZ = 0` assignment is random and adjustment buys precision; otherwise `W` confounds. Argument names match DesignLibrary 0.1's `two_arm_covariate_designer`.

Value

A design object.

See Also

make_design()

Examples

```
prognostic <- two_arm_covariate_designer(N = 40, ate = 0.2, rho_WY = 0.9, h = 0.5)
confounded <- two_arm_covariate_designer(N = 40, ate = 0, rho_WZ = 0.9, rho_WY = 0.9)
```

---

two_arm_designer *Create a one-level two-arm design*

Description

Routes to make_design() with id "two_arm_flexible": make_design("two_arm_flexible", N = N, assignment_prob = assignment_prob, ...).

Usage

```
two_arm_designer(
  N = 100,
  assignment_prob = 0.5,
  control_mean = 0,
  control_sd = 1,
  ate = 1,
  treatment_mean = NULL,
  treatment_sd = NULL,
  rho = 1,
  args_to_fix = NULL
)
```

Arguments

N	Sample size.
assignment_prob	Probability of assignment to treatment.
control_mean	Average outcome in control.
control_sd	Standard deviation in control.
ate	Average treatment effect.
treatment_mean	Average outcome in treatment. If supplied, overrides ate (ate becomes treatment_mean - control_mean).
treatment_sd	Standard deviation in treatment. Defaults to control_sd.
rho	Correlation between treatment and control potential outcomes.
args_to_fix	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

Builds a design with one treatment and one control arm. Treatment effects can be specified by `ate` or by `treatment_mean` (which overrides `ate`). Argument names match DesignLibrary 0.1's `two_arm_designer`.

Value

A design object.

See Also

`make_design()`

Examples

```
make_design("two_arm_flexible", N = 40, ate = 0.2)
two_arm_designer(N = 40, ate = 0.2)
```

---

two_by_two_designer *Create a two-by-two factorial design*

Description

Routes to `make_design()` with id `"two_by_two"`: `make_design("two_by_two", N = N, outcome_means = outcome_means, ...)`.

Usage

```
two_by_two_designer(
  N = 100,
  prob_A = 0.5,
  prob_B = 0.5,
  weight_A = 0.5,
  weight_B = 0.5,
  outcome_means = c(0, 0, 0, 0),
  mean_A0B0 = NULL,
  mean_A0B1 = NULL,
  mean_A1B0 = NULL,
  mean_A1B1 = NULL,
  sd_i = 1,
  outcome_sds = c(0, 0, 0, 0),
  args_to_fix = NULL
)
```

Arguments

N	Sample size.
prob_A	Probability of assignment to A = 1.
prob_B	Probability of assignment to B = 1.
weight_A	Weight on A = 1 when defining the average effect of B.
weight_B	Weight on B = 1 when defining the average effect of A.
outcome_means	Average outcome in each cell, length 4.
mean_A0B0, mean_A0B1, mean_A1B0, mean_A1B1	Optional cell-mean overrides.
sd_i	Standard deviation of the individual-level shock.
outcome_sds	Extra standard deviation in each cell, length 4.
args_to_fix	Ignored. Present for compatibility with DesignLibrary 0.1.

Details

Cell means are outcome_means in order AB = 00, 01, 10, 11. Scalar mean_A0B0 / mean_A0B1 / mean_A1B0 / mean_A1B1 override those entries when supplied.

Value

A design object.

See Also

make_design()

Examples

```
two_by_two_designer(N = 40, outcome_means = c(0, 0.5, 0.5, 1))
```

---

write_audit_report *Write audit results to CSV, markdown, and plain text under* tools/

Description

Write audit results to CSV, markdown, and plain text under tools/

Usage

```
write_audit_report(x, dir = NULL)
```

Arguments

x	A design_library_audit object from audit_designs().
dir	Output directory. Default: tools/ under the package root.

Value

Character vector of paths written (invisibly).

Index

as.character(), *16*
audit_designs, 3
audit_designs(), *9*, *39*

bake_previews, 4
bake_previews(), *17*, *31*
binary_iv_designer, 4
binary_iv_designer(), *33*
block_cluster_two_arm_designer, 6
build_docs, 7

cat(), *15*
cluster_sampling_designer, 8
contributor_checklist, 9
copy_library_shiny, 9

DeclareDesign::redesign(), *21*
design_info, 10
design_info(), *11*, *20*
design_profile, 11
designs_shiny_app, 10

excluded_diagnosands, 12
excluded_diagnosands(), *26*

factorial_designer, 12
formals(), *16*

get_args, 14
get_args(), *20*, *21*
get_code, 15
get_code(), *16*
get_design_code, 16
get_preview, 17

has_preview, 18

install_library_dependencies, 19

list_designs, 20
list_designs(), *22*

make_design, 20
make_design(), *4–8*, *12*, *13*, *23–25*, *27–32*, *34–39*
make_index, 22
match.call(), *16*
match.call.defaults *(*get_design_code*)*, 16
mediation_analysis_designer, 23
multi_arm_designer, 24

param_coverage_gaps, 25
param_coverage_report, 26
param_coverage_report(), *3*
pkgdown::build_site(), *7*
preferred_diagnosands, 26
pretest_posttest_designer, 27
print(), *16*
print.design_library_args, 28
process_tracing_designer, 28

randomized_response_designer, 30
refresh_library, 31
refresh_library(), *17*, *22*
regression_discontinuity_designer, 31

shiny::runApp(), *10*
simple_factorial_designer *(*simple_two_arm_designer*)*, 33
simple_iv_designer *(*simple_two_arm_designer*)*, 33
simple_spillover_designer *(*simple_two_arm_designer*)*, 33
simple_two_arm_designer, 33
spillover_designer, 34
spillover_designer(), *33*

two_arm_attrition_designer, 35
two_arm_covariate_designer, 36
two_arm_designer, 37
two_arm_designer(), *33*

two_by_two_designer, 38
two_by_two_designer(), *33*

write_audit_report, 39
write_audit_report(), *3*