Package 'simpr'

February 13, 2022

Type Package

Title Flexible 'Tidyverse'-Friendly Simulations

Version 0.2.2

Description A general, 'tidyverse'-friendly framework
for simulation studies, design analysis, and power analysis.
Specify data generation, define varying parameters, generate data,
fit models, and tidy model results in a single pipeline, without
needing loops or custom functions.

License GPL-2

Encoding UTF-8

URL https://statisfactions.github.io/simpr/,
https://github.com/statisfactions/simpr/

BugReports https://github.com/statisfactions/simpr/issues/

Depends R (>= 4.0.0)

Imports broom, dplyr, furrr, generics, purrr, magrittr, rlang, tibble,
tidyr (>= 1.2.0), tidyselect

RoxygenNote 7.1.2

Suggests MASS, testthat, ggplot2, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Ethan Brown [aut, cre] (<https://orcid.org/0000-0003-3419-7465>),
Jeffrey Bye [ctb]

Maintainer Ethan Brown <ecbrown@umn.edu>

Repository CRAN

Date/Publication 2022-02-13 00:40:02 UTC

R topics documented:

`apply_fits` *Run a given function or formula expression on a simpr_mod object and tidy the output.*

Description

This function applies a given function to all fit objects and returns the result in a tidy tibble. Any function or purrr-style lambda function can be used.

Usage

```
apply_fits(obj, .f, ..., .progress = FALSE, .options = furrr_options())
```

Arguments

`obj`	a `simpr_tibble` with repetition number, metaparameters, simulated data, and fitted models, from `fit`
`.f`	A function or purrr-style lambda function (see `as_mapper`) used for computing on the fit object
`...`	Additional arguments to `.f`.
`.progress`	A logical, for whether or not to print a progress bar for multiprocess, multisession, and multicore plans .
`.options`	The `future` specific options to use with the workers when using futures. This must be the result from a call to `furrr_options()`.

Value

A `tibble` with columns `.sim_id`, `rep`, `Source` (which contains the name of the fit column), any metaparameters from `define`, and additional columns containing the results of `.f` applied to each fit object.

See Also

tidy_fits, glance_fits

Examples

```
set.seed(100)
logit_fit = specify(a = ~ sample(0:1, size = n, replace = TRUE),
        b = ~ a + rlnorm(n)) %>%
  define(n = c(40, 50)) %>%
  generate(1) %>%
  fit(logit = ~ glm(a ~ b, family = "binomial"))

logit_fit %>%
  apply_fits(broom::augment)

## Arguments to the function can be passed in ...
logit_fit %>%
  apply_fits(broom::augment, se_fit = TRUE)

## Equivalent to tidy_fits()
logit_fit %>%
  apply_fits(broom::tidy)

## Using a purrr-style lambda function
logit_fit %>%
  apply_fits(~ summary(.)$cov.scaled)
```

---

define *Define metaparameters to vary in simulation*

Description

Takes the output of specify (a simpr_spec object) and defines the metaparameters (i.e. simulation factors).

Usage

```
define(.x = NULL, ..., .suffix = "_index")
```

Arguments

.x	a simpr_spec object (the output of specify)
...	metaparameters: named arguments containing vectors or lists of objects to be used in the simulation.
.suffix	name of suffix to append onto index column for list metaparameters, "_index" by default. See *Details*.

Details

This is the second step in the simulation process, after specifying the simulated data using `specify`. The output of `define` is then passed to `generate` to actually generate the simulation.

Metaparameters are named arguments, passed to `...`, that are used in the simulation. A metaparameter is some kind of vector or list, representing something that is to be systematically varied as a part of the simulation design. Any metaparameter should also appear in the formulas of `specify`, and thus the simulation changes depending on the value of the metaparameter.

When creating the simulation, simulations for all possible combinations of metaparameters are generated, resulting in a fully crossed simulation design. If only a subset of the fully crossed design is needed, use the filtering options available in `generate`.

When one of `...` is a list, a new column is generated in the output to `generate` to serve as the index of the list. This new column will be the name of the list argument, with the `suffix` argument appended onto the end. So if `Y = list(a = 1:2, b = letters[2:3])`, and `suffix = "_index"`, the default, a column named `Y_index` would be added to the output of `generate` with values `"a"` and `"b"`.

Value

a `simpr_spec` object to pass onto `generate` for the simulation.

Examples

```
# Simple example of setting a metaparameter
simple_meta = specify(a = ~ 1 + rnorm(n)) %>%
  define(n = c(5, 10)) %>%
  generate(1)

simple_meta # $sim has a 5-row tibble and a 10-row tibble

multi_meta = specify(a = ~ mu + rnorm(n)) %>%
  define(n = c(5, 10),
       mu = seq(-1, 1, length.out = 3)) %>%
  generate(1)

multi_meta # generates simulations for all combos of n and mu


# meta can handle lists which can contain multiple matrices, etc.
meta_list_out = specify(a = ~ MASS::mvrnorm(n, rep(0, 2), Sigma = S)) %>%
  define(n = c(10, 20, 30),
       S = list(independent = diag(2), correlated = diag(2) + 2)) %>%
  generate(1)

meta_list_out # generates S_index column
```

`fit.simpr_tibble` *Fit models to the simulated data*

Description

Takes simulated data from `generate` and applies functions to it, usually model-fitting functions.

Usage

```
## S3 method for class 'simpr_tibble'
fit(
  object,
  ...,
  .quiet = TRUE,
  .warn_on_error = TRUE,
  .stop_on_error = FALSE,
  .debug = FALSE,
  .progress = FALSE,
  .options = furrr_options()
)

## S3 method for class 'simpr_spec'
fit(
  object,
  ...,
  .quiet = TRUE,
  .warn_on_error = TRUE,
  .stop_on_error = FALSE,
  .debug = FALSE,
  .progress = FALSE,
  .options = furrr_options()
)
```

Arguments

`object`	a `simpr_tibble` object–the simulated data from `generate`–or an `simpr_spec` object not yet generated.
`...`	`purrr`-style lambda functions used for computing on the simulated data. See *Details* and *Examples*.
`.quiet`	Should simulation errors be broadcast to the user as they occur?
`.warn_on_error`	Should there be a warning when simulation errors occur? See `vignette("Managing simulation errors")`.
`.stop_on_error`	Should the simulation stop immediately when simulation errors occur?
`.debug`	Run simulation in debug mode, allowing objects, etc. to be explored for each attempt to fit objects.

.progress	A logical, for whether or not to print a progress bar for multiprocess, multisession, and multicore plans .
.options	The future specific options to use with the workers when using futures. This must be the result from a call to furrr_options().

Details

This is the fourth step in the simulation process: after generating the simulation data, apply functions such as fitting a statistical model to the data. The output is often then passed to tidy_fits or glance_fits to extract relevant model estimates from the object.

Similar to specify, the model-fitting ... arguments can be arbitrary R expressions (purrr-style lambda functions, see as_mapper) to specify fitting models to the data. The functions are computed within each simulation cell, so dataset names are generally unnecessary: e.g., to compute regressions on each cell, fit(linear_model = ~ lm(c ~ a + b). If your modeling function requires a reference to the full dataset, use ., e.g. fit(linear_model = ~lm(c ~ a + b,data = .).

Value

a simpr_tibble object with additional list-columns for the output of the provided functions (e.g. model outputs). Just like the output of generate, there is one row per repetition per combination of metaparameters, and the columns are the repetition number rep, the metaparameter names, the simulated data sim, with additional columns for the function outputs specified in .... If per_sim was called previously, fit returns the object to default simpr_tibble mode.

Examples

```
## Generate data to fit models
simple_linear_data = specify(a = ~ 2 + rnorm(n),
                             b = ~ 5 + 3*a + rnorm(n, 0, sd = 0.5)) %>%
  define(n = 100:101) %>%
  generate(2)

## Fit with a single linear term
linear_fit = simple_linear_data %>%
  fit(linear = ~lm(b ~ a, data = .))

linear_fit # first fit element also prints

## Each element of $linear is a model object
linear_fit$linear

## We can fit multiple models to the same data
multi_fit = simple_linear_data %>%
  fit(linear = ~lm(b ~ a, data = .),
      quadratic = ~lm(b ~ a + I(a^2), data = .))

## Two columns, one for each model
multi_fit

## Again, each element is a model object
multi_fit$quadratic
```

```
## Can view terms more nicely with tidy_fits
multi_fit %>%
  tidy_fits

## Can view model summaries with glance_fits
multi_fit %>%
  glance_fits

## Fit functions do not actually need to be any particular kind of model, they
## can be any arbitrary function. However, not all functions will lead to useful
## output with tidy_fits and glance_fits.
add_five_data = simple_linear_data %>%
  fit(add_five = ~ . + 5)  ## adds 5 to every value in dataset

add_five_data
```

`generate.simpr_spec` *Generate simulated data from specification*

Description

Use specification from `specify` or `define` to produce simulated data.

Usage

```
## S3 method for class 'simpr_spec'
generate(
  x,
  .reps,
  ...,
  .sim_name = "sim",
  .quiet = TRUE,
  .warn_on_error = TRUE,
  .stop_on_error = FALSE,
  .debug = FALSE,
  .progress = FALSE,
  .options = furrr_options(seed = TRUE)
)
```

Arguments

`x`	a `simpr_spec` object generated by `define` or `specify`, containing the specifications of the simulation
`.reps`	number of replications to run (a whole number greater than 0)
`...`	filtering criteria for which rows to simulate, passed to `filter`. This is useful for reproducing just a few selected rows of a simulation without needing to redo the entire simulation, see `vignette("Reproducing simulations")`,

.sim_name	name of the list-column to be created, containing simulation results. Default is "sim"
.quiet	Should simulation errors be broadcast to the user as they occur?
.warn_on_error	Should there be a warning when simulation errors occur? See vignette("Managing simulation errors").
.stop_on_error	Should the simulation stop immediately when simulation errors occur?
.debug	Run simulation in debug mode, allowing objects, etc. to be explored for each generated variable specification.
.progress	A logical, for whether or not to print a progress bar for multiprocess, multisession, and multicore plans.
.options	The future specific options to use with the workers when using futures. This must be the result from a call to furrr_options(seed = TRUE).

Details

This is the third step in the simulation process: after specifying the population model and defining the metaparameters, if any, generate is the workhorse function that actually generates the simulated datasets, one for each replication and combination of metaparameters. You likely want to use the output of generate to fit model(s) with fit.

Errors you get using this function usually have to do with how you specified the simulation in specify and define.

Value

a simpr_sims object, which is a tibble with a row for each repetition (a total of rep repetitions) for each combination of metaparameters and some extra metadata used by fit. The columns are rep for the repetition number, the names of the metaparameters, and a list-column (named by the argument sim_name) containing the dataset for each repetition and metaparameter combination. simpr_sims objects can be manipulated elementwise by dplyr and tidyr verbs: the command is applied to each element of the simulation list-column.

See Also

specify and define for examples of how these functions affect the output of generate. See vignette("Optimization") and the furrr website for more information on working with futures: https://furrr.futureverse.org/

Examples

```
meta_list_out = specify(a = ~ MASS::mvrnorm(n, rep(0, 2), Sigma = S)) %>%
  define(n = c(10, 20, 30),
       S = list(independent = diag(2), correlated = diag(2) + 2)) %>%
  generate(3)

 ## View overall structure of the result and a single simulation output
 meta_list_out

 ## Changing .reps will change the number of replications and thus the number of
```

```
## rows in the output
meta_list_2 = specify(a = ~ MASS::mvrnorm(n, rep(0, 2), Sigma = S)) %>%
 define(n = c(10, 20, 30),
      S = list(independent = diag(2), correlated = diag(2) + 2)) %>%
 generate(4)

meta_list_2

## Fitting, tidying functions can be included in this step by running those functions and then
 ## generate.  This can save computation time when doing large
 ## simulations, especially with parallel processing
 meta_list_generate_after = specify(a = ~ MASS::mvrnorm(n, rep(0, 2), Sigma = S)) %>%
  define(n = c(10, 20, 30),
       S = list(independent = diag(2), correlated = diag(2) + 2)) %>%
  fit(lm = ~ lm(a_2 ~ a_1, data = .)) %>%
  tidy_fits %>%
  generate(4)

  meta_list_generate_after
```

`glance_fits` *Create tibble of model "glances" (summaries)*

Description

Turn fitted models of simulated data (from `fit`) into a tidy tibble of model summaries, each with one line (via `broom::glance`).

Usage

```
glance_fits(obj, ..., .progress = FALSE, .options = furrr_options())
```

Arguments

`obj`	tibble with repetition number, metaparameters, simulated data, and fitted models, from `fit`
`...`	Additional arguments to `broom::glance`.
`.progress`	A logical, for whether or not to print a progress bar for multiprocess, multisession, and multicore plans .
`.options`	The `future` specific options to use with the workers when using futures. This must be the result from a call to `furrr_options()`.

Details

This the fifth step of the simulation process: after fitting the model with `fit`, now tidy the model output for further analysis such as evaluating power. All model objects should be supported by `broom::glance`.

The output of this function is quite useful comparing overall model fits; see *Examples*. For looking at specific features of the model such as tests for individual parameter estimates, use `tidy_fits`.

Value

a tibble with the output of the broom::glance method for the given object.

See Also

tidy_fits to view model components (e.g. parameter estimates), apply_fits to apply an arbitrary function to the fits

Examples

```
simple_linear_data = specify(a = ~ 2 + rnorm(n),
          b = ~ 5 + 3 * x1 + rnorm(n, 0, sd = 0.5)) %>%
  define(n = 100:101) %>%
  generate(2)

## Can show tidy output for multiple competing models,
compare_degree = simple_linear_data %>%
  fit(linear = ~lm(a ~ b, data = .),
      quadratic = ~lm(a ~ b + I(b^2), data = .)) %>%
  glance_fits

compare_degree

## Models can be anything supported by broom::tidy.
cor_vs_lm = simple_linear_data %>%
  fit(linear = ~lm(a ~ b, data = .),
      cor = ~ cor.test(.$a, .$b)) %>%
  glance_fits

cor_vs_lm # has NA for non-matching terms
```

---

per_sim *Work directly with simulation results with dplyr and tidyr*

Description

Allows applying data transformations to every simulation result with syntax as if dealing with a single simulation result using dplyr and tidyr verbs

Usage

```
per_sim(obj)
```

Arguments

obj	A simpr_tibble or simpr_spec object.

Details

After producing simulation results (a `simpr_tibble` object), it is sometimes needed to do some data transformation to prepare for analysis. This can always be specified in specify through custom functions, but `per_sim` allows you to also easily specify this in your pipeline. After running per_sim, you can use the dplyr and tidyr verbs you would use on a single simulation result and it will be applied to all results.

If, after running `per_sim`, you wish to return to the default behavior to access `simpr_tibble` results as a tibble with a list_column for simulation results again, run `whole_tibble`.

Value

A `simpr_sims` object for use with dplyr and tidyr verbs.

Examples

```
## Often most convenient to specify simulations for 'wide' data
data_wide = specify(a = ~ runif(5, min = 0, max = 1),
                    b = ~ runif(5, min = 0, max = 2)) %>%
  generate(2)

data_wide

## Any dplyr or tidyr verbs can be applied after per_sim()
data_long = data_wide %>%
  per_sim() %>%
  pivot_longer(everything(), names_to = "name",
               values_to = "value")
data_long

## Now, ready for analysis
data_long %>%
  fit(lm = ~lm(value ~ name)) %>%
  tidy_fits
```

---

print.simpr_spec *Methods for simpr_spec class*

Description

Accessor & display methods for simpr_spec class

Usage

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

```
new_simpr_spec()

is.simpr_spec(x)
```

Arguments

x	a simpr_spec object
...	ignored

Details

Class simpr_spec is created by specify and/or define to specify the simulation variables, which is produced by generate. The print method provides an overview of the specification, including the conditions.

Value

print.simpr_spec has no return value and is called for its side-effects. new_simpr_spec returns an empty simpr_spec object. is.simpr_spec returns a length-1 logical vector, TRUE or FALSE, which indicates whether an object is a simpr_spec.

Examples

```
empty = new_simpr_spec()
print(empty)

## Easiest to create a simpr_spec with specify
simple_spec = specify(a = ~ rbinom(n, size, prob))
print(simple_spec)

## Adding on define adds all possible combinations
## of conditions and more info in output.
defined_spec = specify(a = ~ rbinom(n, size, prob)) %>%
  define(n = c(10, 20),
         size = c(20, 40),
         prob = c(0.2, 0.4))
print(defined_spec)
```

---

specify.formula *Specify data-generating mechanisms*

Description

Specify the data-generating mechanisms for the simulation using purrr-style lambda functions.

Usage

```
## S3 method for class 'formula'
specify(x = NULL, ..., .use_names = TRUE, .sep = "_")
```

Arguments

x	leave this argument blank (NULL); this argument is a placeholder and can be skipped.
...	named purrr-style formula functions used for generating simulation variables. x is not recommended as a name, since it is a formal argument and will be automatically assumed to be the first variable (a message will be displayed if x is used).
.use_names	Whether to use names generated by the lambda function (TRUE, the default), or to overwrite them with supplied names.
.sep	Specify the separator for auto-generating names. See *Column naming*.

Details

This is always the first command in the simulation process, to specify the actual simulated variables, which is then passed to `define` to define metaparameters and then to `generate` to generate the data.

The ... arguments use an efficient syntax to specify custom functions needed for generating a simulation, based on the purrr package. When producing one variable, one can provide an expression such as `specify(a = ~ 3 + runif(10))`; the expression is preceded by ~, the tilde operator, and can refer to previous arguments in `specify` or to metaparameters in `define`. This is called a lambda function.

Order matters: arguments are evaluated sequentially, so later argument can refer to an earlier one, e.g. `specify(a = ~ rnorm(2),b = ~ a + rnorm(2))`.

`generate` combines results together into a single tibble for each simulation, so all lambda functions should produce the same number of rows. However, a lambda function can produce multiple columns.

Value

A `simpr_specify` object which contains the functions needed to generate the simulation; to be passed to `define` for defining metaparameters or, if there are no metaparameters, directly to `generate` for generating the simulation.

Also useful is the fact that one can refer to variables in subsequent arguments. So, one could define another variable b that depends on a very simply, e.g. `specify(a = ~ 3 + runif(10),b = ~ 2 * x)`.

Finally, one can also refer to metaparameters that are to be systematically varied in the simulation study. See `define` and the examples for more details.

Column naming

Because functions can produce different numbers of columns, there are several options for naming columns. If a provided lambda function produces a single column, the name given to the argument becomes the name of the column. If the lambda function already produces column names, then the output will use these names if `.use_names = TRUE`, the default. Otherwise, simpr uses the argument name as a base and auto-numbers the columns. For instance, if the argument a generates a two-column matrix and `.sep = "_"` (the default) the columns will be named a_1and a_2.

Custom names can also be directly provided by a double-sided formula. The left-hand side must use `c` or `cbind`, e.g. `specify(c(a,b) ~ MASS::mvrnorm(5,c(0,0),Sigma = diag(2)))`.

Note

This function is an S3 method for specify from the generics package. Because x is a formal argument of specify, if you have a variable in your simulation named x it will be automatically moved to be the first variable (with a message). It is therefore safest to use any other variable name besides x.

Examples

```
## specify a variable and generate it in the simulation
single_var = specify(a = ~ 1 + rnorm(5)) %>%
  generate(1) # generate a single repetition of the simulation
single_var

two_var = specify(a = ~ 1 + rnorm(5),
                  b = ~ x + 2) %>%
  generate(1)
two_var

## Generates a_01 through a_10
autonumber_var = specify(a = ~ MASS::mvrnorm(5, rep(0, 10), Sigma = diag(10))) %>%
  generate(1)
autonumber_var

# alternatively, you could use a two-sided formula for names
multi_name = specify(cbind(a, b, c) ~ MASS::mvrnorm(5, rep(0, 3), Sigma = diag(3))) %>%
  generate(1)
multi_name

# Simple example of setting a metaparameter
simple_meta = specify(a = ~ 1 + rnorm(n)) %>%
  define(n = c(5, 10)) %>% # without this line you would get an error!
  generate(1)


simple_meta # has two rows now, one for each value of n
simple_meta$sim[[1]] # n = 5
simple_meta$sim[[2]] # n = 10
```

---

tidyverse_verbs *Simpr methods for tidyverse verbs*

Description

These are simpr-compatible methods for generic dplyr and tidyr verbs. The user is not expected to call these methods directly.

Usage

```
## S3 method for class 'simpr_sims'
add_count(
  x,
  ...,
  wt = NULL,
  sort = FALSE,
  name = NULL,
  .drop = lifecycle::deprecated()
)

## S3 method for class 'simpr_spec'
add_count(
  x,
  ...,
  wt = NULL,
  sort = FALSE,
  name = NULL,
  .drop = lifecycle::deprecated()
)

## S3 method for class 'simpr_sims'
anti_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'simpr_spec'
anti_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'simpr_sims'
arrange_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
arrange_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
arrange(.data, ..., .by_group = FALSE)

## S3 method for class 'simpr_spec'
arrange(.data, ..., .by_group = FALSE)

## S3 method for class 'simpr_sims'
as.tbl(x, ...)

## S3 method for class 'simpr_spec'
as.tbl(x, ...)

## S3 method for class 'simpr_sims'
auto_copy(x, y, copy = FALSE, ...)
```

```
## S3 method for class 'simpr_spec'
auto_copy(x, y, copy = FALSE, ...)

## S3 method for class 'simpr_sims'
collect(x, ...)

## S3 method for class 'simpr_spec'
collect(x, ...)

## S3 method for class 'simpr_sims'
compute(x, ...)

## S3 method for class 'simpr_spec'
compute(x, ...)

## S3 method for class 'simpr_sims'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'simpr_spec'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'simpr_sims'
distinct_(.data, ..., .dots, .keep_all = FALSE)

## S3 method for class 'simpr_spec'
distinct_(.data, ..., .dots, .keep_all = FALSE)

## S3 method for class 'simpr_sims'
distinct(.data, ..., .keep_all = FALSE)

## S3 method for class 'simpr_spec'
distinct(.data, ..., .keep_all = FALSE)

## S3 method for class 'simpr_sims'
do_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
do_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
do(.data, ...)

## S3 method for class 'simpr_spec'
do(.data, ...)

## S3 method for class 'simpr_sims'
dplyr_col_modify(data, cols)
```

```
## S3 method for class 'simpr_spec'
dplyr_col_modify(data, cols)

## S3 method for class 'simpr_sims'
dplyr_reconstruct(data, template)

## S3 method for class 'simpr_spec'
dplyr_reconstruct(data, template)

## S3 method for class 'simpr_sims'
dplyr_row_slice(data, i, ...)

## S3 method for class 'simpr_spec'
dplyr_row_slice(data, i, ...)

## S3 method for class 'simpr_sims'
filter_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
filter_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
filter(.data, ..., .preserve = FALSE)

## S3 method for class 'simpr_spec'
filter(.data, ..., .preserve = FALSE)

## S3 method for class 'simpr_sims'
full_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)

## S3 method for class 'simpr_spec'
full_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)
```

```
## S3 method for class 'simpr_sims'
group_by_(.data, ..., .dots = list(), add = FALSE)

## S3 method for class 'simpr_spec'
group_by_(.data, ..., .dots = list(), add = FALSE)

## S3 method for class 'simpr_sims'
group_by(.data, ..., .add = FALSE, .drop = dplyr::group_by_drop_default(.data))

## S3 method for class 'simpr_spec'
group_by(.data, ..., .add = FALSE, .drop = dplyr::group_by_drop_default(.data))

## S3 method for class 'simpr_sims'
group_data(.data)

## S3 method for class 'simpr_spec'
group_data(.data)

## S3 method for class 'simpr_sims'
group_indices_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
group_indices_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
group_indices(.data, ...)

## S3 method for class 'simpr_spec'
group_indices(.data, ...)

## S3 method for class 'simpr_sims'
group_keys(.tbl, ...)

## S3 method for class 'simpr_spec'
group_keys(.tbl, ...)

## S3 method for class 'simpr_sims'
group_map(.data, .f, ..., .keep = FALSE)

## S3 method for class 'simpr_spec'
group_map(.data, .f, ..., .keep = FALSE)

## S3 method for class 'simpr_sims'
group_modify(.data, .f, ..., .keep = FALSE)

## S3 method for class 'simpr_spec'
group_modify(.data, .f, ..., .keep = FALSE)
```

```
## S3 method for class 'simpr_sims'
group_nest(.tbl, ..., .key = "data", keep = FALSE)

## S3 method for class 'simpr_spec'
group_nest(.tbl, ..., .key = "data", keep = FALSE)

## S3 method for class 'simpr_sims'
group_size(x)

## S3 method for class 'simpr_spec'
group_size(x)

## S3 method for class 'simpr_sims'
group_split(.tbl, ..., .keep = TRUE)

## S3 method for class 'simpr_spec'
group_split(.tbl, ..., .keep = TRUE)

## S3 method for class 'simpr_sims'
group_trim(.tbl, .drop = dplyr::group_by_drop_default(.tbl))

## S3 method for class 'simpr_spec'
group_trim(.tbl, .drop = dplyr::group_by_drop_default(.tbl))

## S3 method for class 'simpr_sims'
group_vars(x)

## S3 method for class 'simpr_spec'
group_vars(x)

## S3 method for class 'simpr_sims'
groups(x)

## S3 method for class 'simpr_spec'
groups(x)

## S3 method for class 'simpr_sims'
inner_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)
```

```
## S3 method for class 'simpr_spec'
inner_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)

## S3 method for class 'simpr_sims'
intersect(x, y, ...)

## S3 method for class 'simpr_spec'
intersect(x, y, ...)

## S3 method for class 'simpr_sims'
left_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)

## S3 method for class 'simpr_spec'
left_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)

## S3 method for class 'simpr_sims'
mutate_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
mutate_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
mutate(.data, ...)
```

```
## S3 method for class 'simpr_spec'
mutate(.data, ...)

## S3 method for class 'simpr_sims'
n_groups(x)

## S3 method for class 'simpr_spec'
n_groups(x)

## S3 method for class 'simpr_sims'
nest_by(.data, ..., .key = "data", .keep = FALSE)

## S3 method for class 'simpr_spec'
nest_by(.data, ..., .key = "data", .keep = FALSE)

## S3 method for class 'simpr_sims'
nest_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, ...)

## S3 method for class 'simpr_spec'
nest_join(x, y, by = NULL, copy = FALSE, keep = FALSE, name = NULL, ...)

## S3 method for class 'simpr_sims'
pull(.data, var = -1, name = NULL, ...)

## S3 method for class 'simpr_spec'
pull(.data, var = -1, name = NULL, ...)

## S3 method for class 'simpr_sims'
relocate(.data, ..., .before = NULL, .after = NULL)

## S3 method for class 'simpr_spec'
relocate(.data, ..., .before = NULL, .after = NULL)

## S3 method for class 'simpr_sims'
rename_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
rename_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
rename_with(.data, .fn, .cols = dplyr::everything(), ...)

## S3 method for class 'simpr_spec'
rename_with(.data, .fn, .cols = dplyr::everything(), ...)

## S3 method for class 'simpr_sims'
rename(.data, ...)
```

```
## S3 method for class 'simpr_spec'
rename(.data, ...)

## S3 method for class 'simpr_sims'
right_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)

## S3 method for class 'simpr_spec'
right_join(
  x,
  y,
  by = NULL,
  copy = FALSE,
  suffix = c(".x", ".y"),
  ...,
  keep = FALSE
)

## S3 method for class 'simpr_sims'
rows_delete(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_spec'
rows_delete(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_sims'
rows_insert(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_spec'
rows_insert(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_sims'
rows_patch(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_spec'
rows_patch(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_sims'
rows_update(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_spec'
rows_update(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)
```

```
## S3 method for class 'simpr_sims'
rows_upsert(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_spec'
rows_upsert(x, y, by = NULL, ..., copy = FALSE, in_place = FALSE)

## S3 method for class 'simpr_sims'
rowwise(data, ...)

## S3 method for class 'simpr_spec'
rowwise(data, ...)

## S3 method for class 'simpr_sims'
same_src(x, y)

## S3 method for class 'simpr_spec'
same_src(x, y)

## S3 method for class 'simpr_sims'
sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = NULL, ...)

## S3 method for class 'simpr_spec'
sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = NULL, ...)

## S3 method for class 'simpr_sims'
sample_n(tbl, size, replace = FALSE, weight = NULL, .env = NULL, ...)

## S3 method for class 'simpr_spec'
sample_n(tbl, size, replace = FALSE, weight = NULL, .env = NULL, ...)

## S3 method for class 'simpr_sims'
select_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
select_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
select(.data, ...)

## S3 method for class 'simpr_spec'
select(.data, ...)

## S3 method for class 'simpr_sims'
semi_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'simpr_spec'
semi_join(x, y, by = NULL, copy = FALSE, ...)
```

```
## S3 method for class 'simpr_sims'
setdiff(x, y, ...)

## S3 method for class 'simpr_spec'
setdiff(x, y, ...)

## S3 method for class 'simpr_sims'
setequal(x, y, ...)

## S3 method for class 'simpr_spec'
setequal(x, y, ...)

## S3 method for class 'simpr_sims'
slice_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
slice_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
slice_head(.data, ..., n, prop)

## S3 method for class 'simpr_spec'
slice_head(.data, ..., n, prop)

## S3 method for class 'simpr_sims'
slice_max(.data, order_by, ..., n, prop, with_ties = TRUE)

## S3 method for class 'simpr_spec'
slice_max(.data, order_by, ..., n, prop, with_ties = TRUE)

## S3 method for class 'simpr_sims'
slice_min(.data, order_by, ..., n, prop, with_ties = TRUE)

## S3 method for class 'simpr_spec'
slice_min(.data, order_by, ..., n, prop, with_ties = TRUE)

## S3 method for class 'simpr_sims'
slice_sample(.data, ..., n, prop, weight_by = NULL, replace = FALSE)

## S3 method for class 'simpr_spec'
slice_sample(.data, ..., n, prop, weight_by = NULL, replace = FALSE)

## S3 method for class 'simpr_sims'
slice_tail(.data, ..., n, prop)

## S3 method for class 'simpr_spec'
slice_tail(.data, ..., n, prop)
```

```
## S3 method for class 'simpr_sims'
slice(.data, ..., .preserve = FALSE)

## S3 method for class 'simpr_spec'
slice(.data, ..., .preserve = FALSE)

## S3 method for class 'simpr_sims'
summarise_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
summarise_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
summarise(.data, ..., .groups = NULL)

## S3 method for class 'simpr_spec'
summarise(.data, ..., .groups = NULL)

## S3 method for class 'simpr_sims'
tally(x, wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'simpr_spec'
tally(x, wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'simpr_sims'
tbl_vars(x)

## S3 method for class 'simpr_spec'
tbl_vars(x)

## S3 method for class 'simpr_sims'
transmute_(.data, ..., .dots = list())

## S3 method for class 'simpr_spec'
transmute_(.data, ..., .dots = list())

## S3 method for class 'simpr_sims'
transmute(.data, ...)

## S3 method for class 'simpr_spec'
transmute(.data, ...)

## S3 method for class 'simpr_sims'
ungroup(x, ...)

## S3 method for class 'simpr_spec'
ungroup(x, ...)
```

```
## S3 method for class 'simpr_sims'
union_all(x, y, ...)

## S3 method for class 'simpr_spec'
union_all(x, y, ...)

## S3 method for class 'simpr_sims'
union(x, y, ...)

## S3 method for class 'simpr_spec'
union(x, y, ...)

## S3 method for class 'simpr_sims'
complete_(data, cols, fill = list(), ...)

## S3 method for class 'simpr_spec'
complete_(data, cols, fill = list(), ...)

## S3 method for class 'simpr_sims'
complete(data, ..., fill = list())

## S3 method for class 'simpr_spec'
complete(data, ..., fill = list())

## S3 method for class 'simpr_sims'
drop_na_(data, vars)

## S3 method for class 'simpr_spec'
drop_na_(data, vars)

## S3 method for class 'simpr_sims'
drop_na(data, ...)

## S3 method for class 'simpr_spec'
drop_na(data, ...)

## S3 method for class 'simpr_sims'
expand_(data, dots, ...)

## S3 method for class 'simpr_spec'
expand_(data, dots, ...)

## S3 method for class 'simpr_sims'
expand(data, ..., .name_repair = "check_unique")

## S3 method for class 'simpr_spec'
expand(data, ..., .name_repair = "check_unique")
```

```
## S3 method for class 'simpr_sims'
extract_(
  data,
  col,
  into,
  regex = "([[:alnum:]]+)",
  remove = TRUE,
  convert = FALSE,
  ...
)

## S3 method for class 'simpr_spec'
extract_(
  data,
  col,
  into,
  regex = "([[:alnum:]]+)",
  remove = TRUE,
  convert = FALSE,
  ...
)

## S3 method for class 'simpr_sims'
extract(
  data,
  col,
  into,
  regex = "([[:alnum:]]+)",
  remove = TRUE,
  convert = FALSE,
  ...
)

## S3 method for class 'simpr_spec'
extract(
  data,
  col,
  into,
  regex = "([[:alnum:]]+)",
  remove = TRUE,
  convert = FALSE,
  ...
)

## S3 method for class 'simpr_sims'
fill_(data, fill_cols, .direction = c("down", "up"))
```

```
## S3 method for class 'simpr_spec'
fill_(data, fill_cols, .direction = c("down", "up"))

## S3 method for class 'simpr_sims'
fill(data, ..., .direction = c("down", "up", "downup", "updown"))

## S3 method for class 'simpr_spec'
fill(data, ..., .direction = c("down", "up", "downup", "updown"))

## S3 method for class 'simpr_sims'
gather_(
  data,
  key_col,
  value_col,
  gather_cols,
  na.rm = FALSE,
  convert = FALSE,
  factor_key = FALSE
)

## S3 method for class 'simpr_spec'
gather_(
  data,
  key_col,
  value_col,
  gather_cols,
  na.rm = FALSE,
  convert = FALSE,
  factor_key = FALSE
)

## S3 method for class 'simpr_sims'
gather(
  data,
  key = "key",
  value = "value",
  ...,
  na.rm = FALSE,
  convert = FALSE,
  factor_key = FALSE
)

## S3 method for class 'simpr_spec'
gather(
  data,
  key = "key",
  value = "value",
  ...,
```

```
  na.rm = FALSE,
  convert = FALSE,
  factor_key = FALSE
)

## S3 method for class 'simpr_sims'
nest_legacy(data, ..., .key = "data")

## S3 method for class 'simpr_spec'
nest_legacy(data, ..., .key = "data")

## S3 method for class 'simpr_sims'
nest(.data, ..., .names_sep = NULL, .key = lifecycle::deprecated())

## S3 method for class 'simpr_spec'
nest(.data, ..., .names_sep = NULL, .key = lifecycle::deprecated())

## S3 method for class 'simpr_sims'
pivot_longer(
  data,
  cols,
  names_to = "name",
  names_prefix = NULL,
  names_sep = NULL,
  names_pattern = NULL,
  names_ptypes = list(),
  names_transform = list(),
  names_repair = "check_unique",
  values_to = "value",
  values_drop_na = FALSE,
  values_ptypes = list(),
  values_transform = list(),
  ...
)

## S3 method for class 'simpr_spec'
pivot_longer(
  data,
  cols,
  names_to = "name",
  names_prefix = NULL,
  names_sep = NULL,
  names_pattern = NULL,
  names_ptypes = list(),
  names_transform = list(),
  names_repair = "check_unique",
  values_to = "value",
  values_drop_na = FALSE,
```

```
  values_ptypes = list(),
  values_transform = list(),
  ...
)

## S3 method for class 'simpr_sims'
pivot_wider(
  data,
  id_cols = NULL,
  id_expand = FALSE,
  names_from = NULL,
  names_prefix = "",
  names_sep = "_",
  names_glue = NULL,
  names_sort = FALSE,
  names_vary = "fastest",
  names_expand = FALSE,
  names_repair = "check_unique",
  values_from = NULL,
  values_fill = NULL,
  values_fn = NULL,
  unused_fn = NULL,
  ...
)

## S3 method for class 'simpr_spec'
pivot_wider(
  data,
  id_cols = NULL,
  id_expand = FALSE,
  names_from = NULL,
  names_prefix = "",
  names_sep = "_",
  names_glue = NULL,
  names_sort = FALSE,
  names_vary = "fastest",
  names_expand = FALSE,
  names_repair = "check_unique",
  values_from = NULL,
  values_fill = NULL,
  values_fn = NULL,
  unused_fn = NULL,
  ...
)

## S3 method for class 'simpr_sims'
replace_na(data, replace, ...)
```

```
## S3 method for class 'simpr_spec'
replace_na(data, replace, ...)

## S3 method for class 'simpr_sims'
separate_(
  data,
  col,
  into,
  sep = "[^[:alnum:]]+",
  remove = TRUE,
  convert = FALSE,
  extra = "warn",
  fill = "warn",
  ...
)

## S3 method for class 'simpr_spec'
separate_(
  data,
  col,
  into,
  sep = "[^[:alnum:]]+",
  remove = TRUE,
  convert = FALSE,
  extra = "warn",
  fill = "warn",
  ...
)

## S3 method for class 'simpr_sims'
separate_rows_(data, cols, sep = "[^[:alnum:].]+", convert = FALSE)

## S3 method for class 'simpr_spec'
separate_rows_(data, cols, sep = "[^[:alnum:].]+", convert = FALSE)

## S3 method for class 'simpr_sims'
separate_rows(data, ..., sep = "[^[:alnum:].]+", convert = FALSE)

## S3 method for class 'simpr_spec'
separate_rows(data, ..., sep = "[^[:alnum:].]+", convert = FALSE)

## S3 method for class 'simpr_sims'
separate(
  data,
  col,
  into,
  sep = "[^[:alnum:]]+",
  remove = TRUE,
```

```
  convert = FALSE,
  extra = "warn",
  fill = "warn",
  ...
)

## S3 method for class 'simpr_spec'
separate(
  data,
  col,
  into,
  sep = "[^[:alnum:]]+",
  remove = TRUE,
  convert = FALSE,
  extra = "warn",
  fill = "warn",
  ...
)

## S3 method for class 'simpr_sims'
spread_(
  data,
  key_col,
  value_col,
  fill = NA,
  convert = FALSE,
  drop = TRUE,
  sep = NULL
)

## S3 method for class 'simpr_spec'
spread_(
  data,
  key_col,
  value_col,
  fill = NA,
  convert = FALSE,
  drop = TRUE,
  sep = NULL
)

## S3 method for class 'simpr_sims'
spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)

## S3 method for class 'simpr_spec'
spread(data, key, value, fill = NA, convert = FALSE, drop = TRUE, sep = NULL)

## S3 method for class 'simpr_sims'
```

```
unite_(data, col, from, sep = "_", remove = TRUE)

## S3 method for class 'simpr_spec'
unite_(data, col, from, sep = "_", remove = TRUE)

## S3 method for class 'simpr_sims'
unite(data, col, ..., sep = "_", remove = TRUE, na.rm = FALSE)

## S3 method for class 'simpr_spec'
unite(data, col, ..., sep = "_", remove = TRUE, na.rm = FALSE)

## S3 method for class 'simpr_sims'
unnest_legacy(data, ..., .drop = NA, .id = NULL, .sep = NULL, .preserve = NULL)

## S3 method for class 'simpr_spec'
unnest_legacy(data, ..., .drop = NA, .id = NULL, .sep = NULL, .preserve = NULL)

## S3 method for class 'simpr_sims'
unnest(
  data,
  cols,
  ...,
  keep_empty = FALSE,
  ptype = NULL,
  names_sep = NULL,
  names_repair = "check_unique",
  .drop = lifecycle::deprecated(),
  .id = lifecycle::deprecated(),
  .sep = lifecycle::deprecated(),
  .preserve = lifecycle::deprecated()
)

## S3 method for class 'simpr_spec'
unnest(
  data,
  cols,
  ...,
  keep_empty = FALSE,
  ptype = NULL,
  names_sep = NULL,
  names_repair = "check_unique",
  .drop = lifecycle::deprecated(),
  .id = lifecycle::deprecated(),
  .sep = lifecycle::deprecated(),
  .preserve = lifecycle::deprecated()
)
```

Arguments

x	See original function documentation
...	See original function documentation
wt	See original function documentation
sort	See original function documentation
name	See original function documentation
.drop	See original function documentation
y	See original function documentation
by	See original function documentation
copy	See original function documentation
.data	See original function documentation
.dots	See original function documentation
.by_group	See original function documentation
.keep_all	See original function documentation
data	See original function documentation
cols	See original function documentation
template	See original function documentation
i	See original function documentation
.preserve	See original function documentation
suffix	See original function documentation
keep	See original function documentation
add	See original function documentation
.add	See original function documentation
.tbl	See original function documentation
.f	See original function documentation
.keep	See original function documentation
.key	See original function documentation
var	See original function documentation
.before	See original function documentation
.after	See original function documentation
.fn	See original function documentation
.cols	See original function documentation
in_place	See original function documentation
tbl	See original function documentation
size	See original function documentation
replace	See original function documentation
weight	See original function documentation

.env	See original function documentation
n	See original function documentation
prop	See original function documentation
order_by	See original function documentation
with_ties	See original function documentation
weight_by	See original function documentation
.groups	See original function documentation
fill	See original function documentation
vars	See original function documentation
dots	See original function documentation
.name_repair	See original function documentation
col	See original function documentation
into	See original function documentation
regex	See original function documentation
remove	See original function documentation
convert	See original function documentation
fill_cols	See original function documentation
.direction	See original function documentation
key_col	See original function documentation
value_col	See original function documentation
gather_cols	See original function documentation
na.rm	See original function documentation
factor_key	See original function documentation
key	See original function documentation
value	See original function documentation
.names_sep	See original function documentation
names_to	See original function documentation
names_prefix	See original function documentation
names_sep	See original function documentation
names_pattern	See original function documentation
names_ptypes	See original function documentation
names_transform	See original function documentation
names_repair	See original function documentation
values_to	See original function documentation
values_drop_na	See original function documentation
values_ptypes	See original function documentation

values_transform
: See original function documentation

id_cols
: See original function documentation

id_expand
: See original function documentation

names_from
: See original function documentation

names_glue
: See original function documentation

names_sort
: See original function documentation

names_vary
: See original function documentation

names_expand
: See original function documentation

values_from
: See original function documentation

values_fill
: See original function documentation

values_fn
: See original function documentation

unused_fn
: See original function documentation

sep
: See original function documentation

extra
: See original function documentation

drop
: See original function documentation

from
: See original function documentation

.id
: See original function documentation

.sep
: See original function documentation

keep_empty
: See original function documentation

ptype
: See original function documentation

Details

See original function documentation for details of the functions. Two methods have been created for each `tidyr` and `dplyr` generic function: one for `simpr_spec` objects (generated by `specify` and `define`) which are simply stored for later evaluation by `generate`, and one for `simpr_sims` objects to perform the operation elementwise on each simulation.

To use these special per-simulation versions of these tidyverse verbs as a part of a `simpr` simulation, first run `per_sim` on the object. If you do NOT want to compute only on each simulation but want to return to the default behavior of operating on the entire simulation tibble, use `whole_tibble`.

Value

`simpr_sims` methods return a `simpr_sims` object with the given data transformation applied to each simulation. `simpr_spec` methods return a `simpr_spec` object that stores the given data transformation, to be executed when `generate` is called.

tidy_fits *Tidy fits into a tidy tibble*

Description

Turn models fit to simulated data (from `fit`) into a tidy tibble of model estimates (via `broom::tidy`).

Usage

```
tidy_fits(obj, ..., .progress = FALSE, .options = furrr_options())
```

Arguments

`obj`	a `simpr_tibble` with fitted models, from `fit`
`...`	Additional arguments to the `broom::tidy` method.
`.progress`	A logical, for whether or not to print a progress bar for multiprocess, multisession, and multicore plans .
`.options`	The `future` specific options to use with the workers when using futures. This must be the result from a call to `furrr_options()`.

Details

This the fifth step of the simulation process: after fitting the model with `fit`, now tidy the model output for further analysis such as evaluating power. All model objects should be supported by `broom::tidy`. See `apply_fits` for applying any arbitrary function to the data, including other tidiers.

The output of this function is quite useful for diagnosing bias, precision, and power. For looking at overall features of the model (e.g., R-squared), use `glance_fits`.

Value

a tibble with the output of the `broom::tidy` method applied to each model fit and then bound into a single tibble.

See Also

`glance_fits` to view overall model statistics (e.g. R-squared), `apply_fits` to apply an arbitrary function to the fits

Examples

```
simple_linear_data = specify(a = ~ 2 + rnorm(n),
          b = ~ 5 + 3 * x1 + rnorm(n, 0, sd = 0.5)) %>%
  define(n = 100:101) %>%
  generate(2)

## Can show tidy output for multiple competing models,
```

```
compare_degree = simple_linear_data %>%
  fit(linear = ~lm(a ~ b, data = .),
      quadratic = ~lm(a ~ b + I(b^2), data = .)) %>%
  tidy_fits

compare_degree

## Models can be anything supported by broom::tidy.
cor_vs_lm = simple_linear_data %>%
  fit(linear = ~lm(a ~ b, data = .),
      cor = ~ cor.test(.$a, .$b)) %>%
  tidy_fits

cor_vs_lm # has NA for non-matching terms
```

`whole_tibble` *Convert a simpr_sims object back to a simpr_tibble*

Description

Undoes `per_sim` to allow access to simulation results as a tibble, with simulations available as a list-column.

Usage

```
whole_tibble(x)
```

Arguments

`x`	A `simpr_sims` or `simpr_spec` object.

Details

This function is the inverse of `per_sim`. This enables tidyverse verbs to return to the default behavior of acting on the full table, as opposed to the behavior, activated by `per_sim`, of acting elementwise on the simulation results.

Value

A tibble with the metaparameters and simulation results

Index