Package: NetCoupler (via r-universe)

August 20, 2024

Title Inference of Causal Links Between a Network and an External Variable

Version 0.1.0.9000

Description The 'NetCoupler' algorithm identifies potential direct effects of correlated, high-dimensional variables formed as a network with an external variable. The external variable may act as the dependent/response variable or as an independent/predictor variable to the network.

License MIT + file LICENSE

URL https://github.com/NetCoupler/NetCoupler, https://netcoupler.github.io/NetCoupler/

BugReports https://github.com/NetCoupler/NetCoupler/issues

Depends R (>= 3.5.0)

Imports checkmate, dplyr, ids, igraph, lifecycle, magrittr, pcalg, ppcor, purrr, rlang (>= 0.4.6), stats, tibble, tidyselect, utils, tidygraph

Suggests broom, furrr, knitr, rmarkdown, spelling, testthat (>= 2.1.0)

VignetteBuilder knitr

RdMacros lifecycle

ByteCompile true

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Language en-US

Repository https://netcoupler.r-universe.dev

RemoteUrl https://github.com/netcoupler/netcoupler

RemoteRef HEAD

RemoteSha af57087fc55fef4b678a09b15a544532866dbbf8

Contents

`as_edge_tbl` *Convert network graphs to edge tables as tibbles/data.frames.*

Description

[Experimental]

Usage

```
as_edge_tbl(network_object)
```

Arguments

`network_object`	Network graph from `nc_estimate_network()`.

Value

A tibble, with at least two columns:

- `source_node`: The starting node (variable).
- `target_node`: The ending node (variable) that links with the source node.
- `adjacency_weight`: (Optional) The "weight" given to the edge, which represents the strength of the link between two nodes.

See Also

See nc_estimate_links for examples on using NetCoupler.

classify_options *Classification options for direct, ambigious, and no effect.*

Description

Classification options for direct, ambigious, and no effect.

Usage

```
classify_options(
  single_metabolite_threshold = 0.05,
  network_threshold = 0.1,
  direct_effect_adjustment = NA
)
```

Arguments

`single_metabolite_threshold`, `network_threshold`, `direct_effect_adjustment`
: See the `classify_option_list` argument in nc_estimate_links for details.

Value

List with options for the classification.

nc_estimate_links *Compute model estimates between an external (exposure or outcome) variable and a network.*

Description

[Experimental]

This is the main function that identifies potential links between external factors and the network. There are two functions to estimate and classify links:

- `nc_estimate_exposure_links()`: Computes the model estimates for the exposure side.
- `nc_estimate_outcome_links()`: Computes the model estimates for the exposure side.

Usage

```
nc_estimate_exposure_links(
  data,
  edge_tbl,
  exposure,
  adjustment_vars = NA,
  model_function,
  model_arg_list = NULL,
  exponentiate = FALSE,
  classify_option_list = classify_options()
)

nc_estimate_outcome_links(
  data,
  edge_tbl,
  outcome,
  adjustment_vars = NA,
  model_function,
  model_arg_list = NULL,
  exponentiate = FALSE,
  classify_option_list = classify_options()
)
```

Arguments

`data`	The data.frame or tibble that contains the variables of interest, including the variables used to make the network.
`edge_tbl`	Output graph object from `nc_estimate_network()`, converted to an edge table using `as_edge_tbl()`.
`exposure`, `outcome`	Character. The exposure or outcome variable of interest.
`adjustment_vars`	Optional. Variables to adjust for in the models.
`model_function`	A function for the model to use (e.g. `stats::lm()`, `stats::glm()`, survival::coxph()). Can be any model as long as the function has the arguments `formula` and `data`. Type in the model function as a bare object (without `()`, for instance as `lm`).
`model_arg_list`	Optional. A list containing the named arguments that will be passed to the model function. A simple example would be `list(family = binomial(link = "logit"))` to specify that the `glm` model is a logistic model and not a linear one. See the examples for more on the usage.
`exponentiate`	Logical. Whether to exponentiate the log estimates, as computed with e.g. logistic regression models.
`classify_option_list`	A list with classification options for direct, ambigious, or no effects. Used with the `classify_options()` function with the arguments:

- `single_metabolite_threshold`: Default of 0.05. P-values from models with only the index metabolite (no neighbour adjustment) are classified as effects if below this threshold. For larger sample sizes and networks, we recommend lowering the threshold to reduce risk of false positives.
- `network_threshold`: Default of 0.1. P-values from any models that have direct neighbour adjustments are classified as effects if below this threshold. This is assumed as a one-sided p-value threshold. Like the threshold above, a lower value should be used for larger sample sizes and networks.
- `direct_effect_adjustment`: Default is NA. After running the algorithm once, sometimes it's useful to adjust for the direct effects identified to confirm whether other links exist.

Value

Outputs a tibble that contains the model estimates from either the exposure or outcome side of the network as well as the effect classification. Each row represents the "no neighbour node adjusted" model and has the results for the outcome/exposure to index node pathway. Columns for the outcome are:

- `outcome` or `exposure`: The name of the variable used as the external variable.
- `index_node`: The name of the metabolite used as the index node from the network. In combination with the outcome/exposure variable, they represent the individual model used for the classification.
- `estimate`: The estimate from the outcome/exposure and index node model.
- `std_error`: The standard error from the outcome/exposure and index node model.
- `fdr_p_value`: The False Discovery Rate-adjusted p-value from the outcome/exposure and index node model.
- `effect`: The NetCoupler classified effect between the index node and the outcome/exposure. Effects are classified as "direct" (there is a probable link based on the given thresholds), "ambigious" (there is a potential link but not all thresholds were passed), and "none" (no potential link seen).

The tibble output also has an attribute that contains all the models generated *before* classification. Access it with `attr(output, "all_models_df")`.

See Also

`vignette("examples")` article has more details on how to use NetCoupler with different models.

Examples

```
standardized_data <- simulated_data %>%
    nc_standardize(starts_with("metabolite"))

metabolite_network <- simulated_data %>%
    nc_standardize(starts_with("metabolite"),
                   regressed_on = "age") %>%
    nc_estimate_network(starts_with("metabolite"))
edge_table <- as_edge_tbl(metabolite_network)
```

```
results <- standardized_data %>%
  nc_estimate_exposure_links(
    edge_tbl = edge_table,
    exposure = "exposure",
    model_function = lm
   )
results

# Get results of all models used prior to classification
```

`nc_estimate_network` *Create an estimate of the metabolic network as an undirected graph.*

Description

[Experimental]

The main NetCoupler network creator. Uses the input data to estimate the underlying undirected graph. The default uses the PC algorithm, implemented within NetCoupler with `pc_estimate_undirected_graph()` Defaults to using the PC algorithm to calculate possible edges. Any missing values in the input data are removed by this function, since some computations can't handle missingness.

Usage

```
nc_estimate_network(data, cols = everything(), alpha = 0.01)
```

Arguments

`data`	Data that would form the underlying network.
`cols`	<`tidy-select`> Variables to include by using `dplyr::select()` style selection.
`alpha`	The alpha level to use to test whether an edge exists or not. Default is 0.01.

Value

Outputs a `tidygraph::tbl_graph()` with the start and end nodes, as well as the edge weights.

See Also

See nc_estimate_links for examples on using NetCoupler and pc_estimate_undirected_graph for more details on the PC-algorithm network estimation method.

`nc_standardize` *Standardize the metabolic variables.*

Description

[Experimental]

Can standardize by either 1) `log()`-transforming and then applying `scale()` (mean-center and scaled by standard deviation), or 2) if `regressed_on` variables are given, then log-transforming, running a linear regression to obtain the `stats::residuals()`, and finally scaled. Use `regressed_on` to try to remove influence of potential confounding.

Usage

```
nc_standardize(data, cols = everything(), regressed_on = NULL)
```

Arguments

`data`	Data frame.
`cols`	Metabolic variables that will make up the network.
`regressed_on`	Optional. A character vector of variables to regress the metabolic variables on. Use if you want to standardize the metabolic variables on variables that are known to influence them, e.g. sex or age. Calculates the residuals from a linear regression model.

Value

Outputs a tibble object, with the original metabolic variables now standardized.

See Also

nc_estimate_links for more detailed examples or the `vignette("NetCoupler")`.

Examples

```
# Don't regress on any variable
simulated_data %>%
  nc_standardize(starts_with("metabolite_"))

# Extract residuals by regressing on a variable
simulated_data %>%
  nc_standardize(starts_with("metabolite_"), "age")

# Works with factors too
simulated_data %>%
  dplyr::mutate(Sex = as.factor(sample(rep(c("F", "M"), times = nrow(.) / 2)))) %>%
  nc_standardize(starts_with("metabolite_"), c("age", "Sex"))
```

`simulated_data` *Simulated dataset with an underlying Directed Graph structure for the metabolites.*

Description

Simulated dataset with an underlying Directed Graph structure for the metabolites.

Usage

```
simulated_data
```

Format

The simulated dataset is a tibble with the following variables:

- Two outcome variables (`outcome_continuous` and `outcome_binary`) along with survival time (`outcome_event_time`) that is used for the `outcome_binary` variable
- A generic `exposure` variable as continuous
- 12 `metabolite_*` variables
- An `age` variable used as a confounder

Index