

Package: lepidoptera (via r-universe)

October 28, 2026

Title Domain-Agnostic Causal Simulation Engine with Mixed Variable Types

Version 0.1.2

Description A domain-agnostic simulation engine for structural causal models with heterogeneous variable types. Users define systems of continuous, categorical, and ordinal features, declare a directed acyclic graph (DAG) over them, attach typed generative mechanisms to each node, and simulate synthetic populations. A pre-built mechanism library covers all nine parent-type/child-type combinations. Vine copula mechanisms estimate full joint dependence from observed data via pair-copula constructions (Bedford and Cooke, 2002) [doi:10.1214/aos/1031689014](https://doi.org/10.1214/aos/1031689014). Individual-level counterfactual queries are supported through partial propagation, which freezes unaffected random draws and resamples only the downstream subgraph, enabling individual treatment effect (ITE) estimation alongside population-level average treatment effects (ATE). Any pre-fitted predictive model can serve as a generative mechanism through three wrapper functions. Systems serialise to JSON for cross-language portability with the companion Python package.

License MIT + file LICENSE

URL <https://github.com/danilosarti/lepidoptera>

BugReports <https://github.com/danilosarti/lepidoptera/issues>

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats

Suggests testthat (>= 3.0.0), igraph, rvinecopulib (>= 0.6.0), jsonlite, yaml, knitr, rmarkdown

Config/testthat/edition 3

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no
Author Danilo Sarti [aut, cre]
Maintainer Danilo Sarti <danilosarti@rcsi.ie>
Repository https://cran.r-universe.dev
Date/Publication 2026-10-28 11:40:02 UTC
RemoteUrl https://github.com/cran/lepidoptera
RemoteRef HEAD
RemoteSha 52088561ae98944af19cbe848e6e3786d15861ce

Contents

add_dependency	2
add_feature	3
compare	4
copula_mechanism	4
do_distribution	5
do_fix	6
export_system	6
feature_spec	7
feature_types	7
finalize_system	8
import_system	8
learned_mechanism	9
list_copula_families	9
llm_mechanism	10
mechanism_library_get	11
mechanism_override	11
model_mechanism	12
new_system	14
plot_graph	14
propagate	15
register_mechanism	15
shift	16
simulate	17
Index	18

add_dependency	<i>Add a dependency (edge set + mechanism) to a system</i>
----------------	--

Description

Add a dependency (edge set + mechanism) to a system

Usage

```
add_dependency(system, parents, child, mechanism)
```

Arguments

system	A system object.
parents	Character vector of parent feature names (use <code>character(0)</code> for a marginal/root mechanism), in the same order used when mechanism was built.
child	Character. Name of the dependent feature. Must already exist (see add_feature()).
mechanism	A mechanism object, e.g. from mechanism_library_get() or register_mechanism() .

Value

The updated system.

add_feature	<i>Add a feature to a system</i>
-------------	----------------------------------

Description

Add a feature to a system

Usage

```
add_feature(system, spec)
```

Arguments

system	A system object, as created by new_system() .
spec	A feature_spec, as created by feature_spec() .

Value

The updated system.

compare	<i>Compare baseline and intervened populations</i>
---------	--

Description

For each feature in `affected_features`: mean/sd shift for continuous features, level-frequency shift for categorical/ordinal features.

Usage

```
compare(result)
```

Arguments

`result` A simulation_result, as returned by `propagate()`.

Value

A named list, one entry per affected feature.

copula_mechanism	<i>Copula-based dependency mechanism</i>
------------------	--

Description

Fits a copula on `fit_data` and returns a mechanism that draws the child feature conditional on realised parent values. Supports both the built-in Gaussian copula (base R, no extra packages) and the full family catalogue of **rvinecopulib** (package must be installed for non-Gaussian families).

Usage

```
copula_mechanism(
  variables,
  family = "gaussian",
  marginals = "auto",
  nominal_handling = c("regression", "arbitrary_order"),
  fit_data = NULL
)
```

Arguments

variables	Character vector. <code>variables[1]</code> is the child (the feature to generate); <code>variables[-1]</code> are the parents. Match with <code>add_dependency(parents = variables[-1], child = variables[1], ...)</code> .
family	Copula family. "gaussian" (default) uses the built-in base-R path. "auto" uses rvinecopulib with AIC-based family selection if the package is installed, otherwise falls back to Gaussian. All other values require rvinecopulib : • Elliptical: "t" • Archimedean: "clayton", "gumbel", "frank", "joe" • BB families: "bb1", "bb6", "bb7", "bb8" • Rotations: append "90", "180", or "270" (e.g. "clayton90") • Non-parametric: "t11" • Meta-sets: "onepar", "twopar", "parametric", "all"
marginals	"auto" (empirical CDF, rank-based). Only "auto" is implemented in v1.
nominal_handling	"regression" (default): error if a nominal factor appears in variables. "arbitrary_order": include with a warning.
fit_data	A data.frame with all variables columns. Required.

Value

A mechanism object (not JSON-serializable; spec = NULL).

do_distribution	<i>Intervention: replace a feature's generative distribution</i>
-----------------	--

Description

v1 scope: distribution must be a *marginal* mechanism (no parents) – the feature is resampled from distribution independently of its original parents. Use `mechanism_override()` for a conditional replacement.

Usage

```
do_distribution(target, distribution)
```

Arguments

target	Character. Feature name.
distribution	A marginal mechanism, e.g. from <code>mechanism_library_get("normal")(mean = ..., sd = ...)</code> .

Value

An intervention object.

do_fix	<i>Intervention: fix a feature to a constant value</i>
--------	--

Description

Classic do-operator: disconnects target from its parents and sets it to value for every unit.

Usage

```
do_fix(target, value)
```

Arguments

target	Character. Feature name (or vector of names).
value	The fixed value (or vector of values, one per target).

Value

An intervention object.

export_system	<i>Export a system to JSON or YAML</i>
---------------	--

Description

Export a system to JSON or YAML

Usage

```
export_system(system, path, format = c("json", "yaml"))
```

Arguments

system	A system object (need not be finalized).
path	Output file path.
format	Character. "json" or "yaml".

Value

Invisibly, path.

feature_spec	<i>Declare a feature</i>
--------------	--------------------------

Description

Creates a feature_spec: name, type, domain, and optional metadata. See SPEC.md section 3.2.

Usage

```
feature_spec(name, type, domain, metadata = NULL)
```

Arguments

name	Character. Unique identifier within the system.
type	Character. One of feature_types() .
domain	List. For "continuous": list(min = ..., max = ...) (either may be -Inf/Inf). For "categorical"/"ordinal": list(levels = c(...)), where the order of levels matters for "ordinal" (it defines the scale order).
metadata	Optional named list, never interpreted by the core.

Value

A feature_spec object (S3 list with class "feature_spec").

feature_types	<i>Valid feature types</i>
---------------	----------------------------

Description

Valid feature types

Usage

```
feature_types()
```

Value

A character vector of the allowed FeatureType values: "continuous", "categorical", "ordinal".

finalize_system	<i>Finalize and validate a system</i>
-----------------	---------------------------------------

Description

Runs the invariants in SPEC.md section 3.5:

1. every feature has exactly one mechanism;
2. the dependency graph is acyclic;
3. every edge references existing features;
4. each mechanism's declared parent types match the actual types of the parents it was attached to;
5. the mechanism's output type matches the feature's declared type.

Usage

```
finalize_system(system)
```

Arguments

system	A system object.
--------	------------------

Value

The same system, marked finalized, ready for [simulate\(\)](#).

import_system	<i>Import a system from JSON or YAML</i>
---------------	--

Description

Import a system from JSON or YAML

Usage

```
import_system(path, format = c("json", "yaml"))
```

Arguments

path	Input file path.
format	Character. "json" or "yaml".

Value

A system object. Not yet finalized – call [finalize_system\(\)](#) before simulating.

learned_mechanism	<i>Fit a mechanism directly from observed data</i>
-------------------	--

Description

v1 scope: parents must all be continuous columns of `fit_data`. For a continuous child, fits OLS + Gaussian residual noise. For a categorical/ordinal child, uses a k-nearest-neighbors empirical sampler over standardized parent space.

Usage

```
learned_mechanism(parents, child, fit_data, uncertainty = "auto", k = 10L)
```

Arguments

parents	Character vector of continuous parent feature names.
child	Character. Name of the column in <code>fit_data</code> to predict.
fit_data	A <code>data.frame</code> with columns <code>parents</code> and <code>child</code> .
uncertainty	Reserved for future use; currently only "auto" is implemented (OLS + residual noise for continuous, k-NN sampling for categorical/ordinal).
k	Number of neighbours for the categorical/ordinal branch.

Value

A mechanism object (not JSON-serializable – spec is NULL, since it carries fitted parameters as a closure).

list_copula_families	<i>List available copula families</i>
----------------------	---------------------------------------

Description

Returns "gaussian" always (built-in base-R path). When **rvinecopulib** is installed the full parametric family list is prepended, together with the meta-set names "auto", "all", "parametric", "onepar", "twopar".

Usage

```
list_copula_families()
```

Value

Character vector of family identifiers accepted by `copula_mechanism()`.

llm_mechanism

A mechanism whose value is generated by an LLM call

Description

A mechanism whose value is generated by an LLM call

Usage

```
llm_mechanism(
  prompt_template,
  model,
  parents,
  parent_types,
  output_type,
  levels = NULL,
  parser = "auto"
)
```

Arguments

prompt_template	A glue/sprintf-free template for <code>base::sprintf()</code> – or, more flexibly, a function <code>function(parent_values)</code> returning the prompt string directly. When a character string is supplied it is formatted with <code>base::do.call()</code> + format, substituting named parent values.
model	Function <code>function(prompt_text)</code> returning a string (the LLM response). Supply any client you like, e.g. <code>function(p) httr2::resp_body_json(...)\$content[[1]]\$text</code> .
parents	Character vector of parent feature names.
parent_types	Character vector, one <code>feature_types()</code> entry per parent.
output_type	One of <code>feature_types()</code> .
levels	Required when <code>output_type</code> is "categorical" or "ordinal" – the response is matched (case-insensitively) against these.
parser	"auto" (numeric extraction for continuous, level matching for categorical/ordinal) or a custom function(<code>response</code>) returning the parsed value.

Value

A mechanism object (not JSON-serializable – spec is NULL, since it carries a live `model` function).

mechanism_library_get *Look up a pre-built mechanism factory from the library*

Description

Pre-built mechanisms cover the type-combination matrix in SPEC.md section 4. `mechanism_library_get(key)` returns a *factory* (a function), not a ready mechanism – pre-built mechanisms always need parameters. Call the factory with those parameters to obtain the actual mechanism object, e.g. `mechanism_library_get("linear_gaussian") = list(age = 0.1, sd = 1)`.

Usage

```
mechanism_library_get(key)
```

Arguments

key	Character. One of "normal", "categorical_marginal", "ordinal_marginal", "linear_gaussian", "multinomial_logit", "ordered_logit", "categorical_transition", "group_effect".
-----	--

Value

A factory function.

mechanism_override *Intervention: override a feature's mechanism entirely*

Description

The replacement mechanism must declare the same number and types of parents that target was originally given in `add_dependency()` (the parent set itself is not changed by this intervention) – mismatches surface as a validation error when the derived system is finalized.

Usage

```
mechanism_override(target, mechanism)
```

Arguments

target	Character. Feature name.
mechanism	A mechanism object replacing the current one.

Value

An intervention object.

model_mechanism	<i>Wrap a pre-fitted model as a generative mechanism</i>
-----------------	--

Description

Turns any model object the user has already trained (a ranger random forest, a glm, an xgboost booster, a keras network, ...) into a standard lepidoptera mechanism. The user provides a single `predict_fn` that accepts a named list of parent values and returns one predicted value.

Usage

```
model_mechanism(
  predict_fn,
  parents,
  parent_types,
  output_type,
  levels = NULL,
  name = "model_mechanism"
)
```

Arguments

<code>predict_fn</code>	A function <code>function(parent_values)</code> where <code>parent_values</code> is a named list (same structure as inside any mechanism fn). Must return a single value compatible with <code>output_type</code> . The function is called once per simulated unit; use R's RNG directly inside it for stochastic outputs.
<code>parents</code>	Character vector of parent feature names (must match the names present in <code>parent_values</code> at generation time).
<code>parent_types</code>	Character vector of feature_types() , one per parent, in the same order as <code>parents</code> .
<code>output_type</code>	One of feature_types() .
<code>levels</code>	Required when <code>output_type</code> is "categorical" or "ordinal" – used only for documentation / validation; the actual return value is whatever <code>predict_fn</code> produces.
<code>name</code>	Optional character label stored in the mechanism object. Defaults to "model_mechanism".

Details

This differs from [learned_mechanism\(\)](#) in a fundamental way: `learned_mechanism()` **fits** a model internally on `fit_data`; `model_mechanism()` **uses** a model that was already fitted outside lepidoptera. The user is responsible for the predict function, including reshaping inputs, choosing the right output slot, and optionally adding stochastic noise.

Quick patterns:

```

# ranger random forest
rf <- ranger::ranger(outcome ~ age + bp, data = df, probability = TRUE)
mech <- model_mechanism(
  predict_fn = function(pv) {
    nd <- data.frame(age = as.numeric(pv$age), bp = as.numeric(pv$bp))
    pred <- predict(rf, nd)$predictions
    sample(colnames(pred), 1, prob = pred[1,])
  },
  parents      = c("age", "bp"),
  parent_types = c("continuous", "continuous"),
  output_type  = "categorical",
  levels       = c("no_response", "response")
)

# glm (logistic regression)
fit <- glm(response ~ age + dose, data = trial, family = binomial)
mech <- model_mechanism(
  predict_fn = function(pv) {
    nd <- data.frame(age = as.numeric(pv$age), dose = as.numeric(pv$dose))
    p <- predict(fit, nd, type = "response")[[1]]
    sample(c("yes", "no"), 1, prob = c(p, 1-p))
  },
  parents      = c("age", "dose"),
  parent_types = c("continuous", "continuous"),
  output_type  = "categorical",
  levels       = c("yes", "no")
)

# lm (linear regression) with residual noise
fit_lm <- lm(sbp ~ age + bmi, data = cohort)
sigma <- summary(fit_lm)$sigma
mech <- model_mechanism(
  predict_fn = function(pv) {
    nd <- data.frame(age = as.numeric(pv$age), bmi = as.numeric(pv$bmi))
    pred <- predict(fit_lm, nd)[[1]]
    pred + rnorm(1, 0, sigma) # add residual noise manually
  },
  parents      = c("age", "bmi"),
  parent_types = c("continuous", "continuous"),
  output_type  = "continuous"
)

```

Value

A mechanism object (not JSON-serializable – spec = NULL).

new_system	Create an empty system
------------	------------------------

Description

Create an empty system

Usage

```
new_system()
```

Value

A system object with no features, edges, or mechanisms yet.

plot_graph	Plot a system's dependency graph
------------	----------------------------------

Description

Requires the optional igraph package (`install.packages("igraph")`).

Usage

```
plot_graph(system)
```

Arguments

system	A system object.
--------	------------------

Value

Invisibly, the igraph graph object plotted.

propagate

Apply interventions and propagate their effects

Description

Apply interventions and propagate their effects

Usage

```
propagate(
  system,
  interventions,
  n = NULL,
  seed = NULL,
  mode = c("full", "partial")
)
```

Arguments

system	A finalized system object.
interventions	A list of intervention objects (see do_fix() , do_distribution() , shift() , mechanism_override()).
n	Integer. Number of units to simulate for the baseline (required).
seed	Optional integer seed, shared between baseline and intervened population for paired comparison.
mode	Character. "full" resamples the whole population from the derived system. "partial" keeps unaffected features exactly equal to the baseline and recomputes only the downstream closure of the targeted features. The two modes are each internally reproducible given the same seed, but are not bit-identical to each other (they consume the per-unit random stream differently).

Value

A `simulation_result`: a list with `baseline`, `intervened`, `interventions`, `affected_features`, `propagation_order`.

register_mechanism

Register a custom mechanism

Description

Register a custom mechanism

Usage

```
register_mechanism(name, fn, parent_types, output_type)
```

Arguments

name	Character. Key under which the mechanism is registered.
fn	Function with signature <code>function(parent_values)</code> returning a value compatible with <code>output_type</code> . Use R's RNG functions (<code>rnorm()</code> , <code>sample()</code> , ...) directly inside <code>fn</code> ; the caller seeds the global RNG per unit (see <code>simulate.R</code>).
parent_types	Character vector, one <code>feature_types()</code> entry per declared parent, in the order parents will be declared in <code>add_dependency()</code> .
output_type	One of <code>feature_types()</code> .

Value

The registered mechanism object.

shift	<i>Intervention: shift the value a feature would otherwise take</i>
-------	---

Description

Preserves the dependency on parents; applies `fn` to the value that the original mechanism would have produced.

Usage

```
shift(target, fn)
```

Arguments

target	Character. Feature name.
fn	Function applied to the originally generated value.

Value

An intervention object.

simulate	<i>Simulate a population from a system</i>
----------	--

Description

Simulate a population from a system

Usage

```
simulate(system, n, seed = NULL)
```

Arguments

system	A finalized system object (see finalize_system()).
n	Integer. Number of units to generate.
seed	Optional integer seed for reproducibility.

Value

A data.frame with one row per unit and one column per feature, typed according to each feature's FeatureType (factor for categorical/ordinal columns, numeric for continuous ones).

Index

add_dependency, [2](#)
add_feature, [3](#)
add_feature(), [3](#)

base::do.call(), [10](#)
base::sprintf(), [10](#)

compare, [4](#)
copula_mechanism, [4](#)
copula_mechanism(), [9](#)

do_distribution, [5](#)
do_distribution(), [15](#)
do_fix, [6](#)
do_fix(), [15](#)

export_system, [6](#)

feature_spec, [7](#)
feature_spec(), [3](#)
feature_types, [7](#)
feature_types(), [7](#), [10](#), [12](#), [16](#)
finalize_system, [8](#)
finalize_system(), [8](#), [17](#)

import_system, [8](#)

learned_mechanism, [9](#)
learned_mechanism(), [12](#)
list_copula_families, [9](#)
llm_mechanism, [10](#)

mechanism_library_get, [11](#)
mechanism_library_get(), [3](#)
mechanism_override, [11](#)
mechanism_override(), [5](#), [15](#)
model_mechanism, [12](#)

new_system, [14](#)
new_system(), [3](#)

plot_graph, [14](#)

propagate, [15](#)
propagate(), [4](#)

register_mechanism, [15](#)
register_mechanism(), [3](#)

shift, [16](#)
shift(), [15](#)
simulate, [17](#)
simulate(), [8](#)