Package: dormancy (via r-universe)

May 16, 2026

Type Package

Title Detection and Analysis of Dormant Patterns in Data

Version 0.1.0

Description A novel framework for detecting, quantifying, and analyzing dormant patterns in multivariate data. Dormant patterns are statistical relationships that exist in data but remain inactive until specific trigger conditions emerge. This concept, inspired by biological dormancy (seeds, pathogens) and geological phenomena (dormant faults), provides tools to identify latent risks, hidden correlations, and potential phase transitions in complex systems. The package introduces methods for quantifying dormancy depth, trigger sensitivity, and awakening risk - enabling analysts to discover patterns that conventional methods miss because they focus only on currently active relationships.

License file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports stats, utils, grDevices, graphics, Rcpp (>= 1.0.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, covr

LinkingTo Rcpp

VignetteBuilder knitr

URL https://github.com/danymukesha/dormancy, https://danymukesha.github.io/dormancy

BugReports https://github.com/danymukesha/dormancy/issues

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

Date/Publication 2026-03-04 21:38:34 UTC

RemoteUrl https://github.com/danymukesha/dormancy

RemoteRef HEAD

RemoteSha f3ecc1e248ec4ed2fdee3a6e622fc07c84429c6a

Contents

`awaken` *Simulate Awakening of Dormant Patterns*

Description

Simulates what would happen if a dormant pattern were to "awaken" - i.e., become active. This function allows exploration of potential future states and scenario analysis without waiting for actual pattern activation.

Usage

```
awaken(
  dormancy_result,
  pattern_id = 1,
  intensity = 1,
  n_sim = 100,
  return_data = FALSE,
  verbose = FALSE
)
```

Arguments

`dormancy_result`	An object of class "dormancy" from `dormancy_detect`.
`pattern_id`	Integer or "all". Which pattern(s) to awaken. Default is 1.
`intensity`	Numeric. Intensity of awakening, from 0 (dormant) to 1 (fully active). Default is 1.
`n_sim`	Integer. Number of simulation runs. Default is 100.
`return_data`	Logical. Whether to return simulated data. Default is FALSE.
`verbose`	Logical. Whether to print progress messages. Default is FALSE.

Details

Awakening simulation is valuable for:

- Scenario planning and stress testing
- Understanding potential system behaviors
- Preparing for pattern activation events
- Testing the robustness of current strategies

The simulation works by:

1. Identifying the dormant pattern's trigger conditions
2. Simulating data where those conditions are met
3. Applying the pattern's relationship to the simulated data
4. Measuring the resulting effects on the system

Value

A list containing:

- `awakening_effects` - Data frame describing the effects of awakening
- `simulated_stats` - Summary statistics from simulations
- `cascade_effects` - Effects on other patterns (if any)
- `simulated_data` - If `return_data = TRUE`, simulated datasets

Examples

```
set.seed(42)
n <- 500
x <- rnorm(n)
z <- sample(c(0, 1), n, replace = TRUE)
y <- ifelse(z == 1, 0.8 * x + rnorm(sum(z), 0, 0.3), rnorm(n))
data <- data.frame(x = x, y = y, z = factor(z))

result <- dormancy_detect(data, method = "conditional")
awakening <- awaken(result, pattern_id = 1, n_sim = 50)
print(awakening)
```

`dormancy_depth` *Measure the Depth of Dormancy in Patterns*

Description

Quantifies how "deeply asleep" a dormant pattern is - measuring the energy required to activate it and the stability of its dormant state. Deeper dormancy implies greater resistance to activation but potentially larger effects when awakened.

Usage

```
dormancy_depth(
  dormancy_result,
  method = "combined",
  normalize = TRUE,
  verbose = FALSE
)
```

Arguments

`dormancy_result`	An object of class "dormancy" from `dormancy_detect`.
`method`	Character. The depth measurement method: • "energy" - Measures the statistical "energy barrier" to activation • "stability" - Measures how stable the dormant state is • "entropy" - Measures information-theoretic dormancy depth • "combined" - Weighted combination of all methods Default is "combined".
`normalize`	Logical. Whether to normalize depth scores to [0, 1]. Default is TRUE.
`verbose`	Logical. Whether to print progress messages. Default is FALSE.

Details

Dormancy depth is a novel concept in statistical analysis, inspired by:

- **Physics**: Potential energy barriers in phase transitions
- **Biology**: Depth of seed dormancy (stratification requirements)
- **Geology**: Locked fault segments and earthquake potential

A deeply dormant pattern:

- Requires significant change in conditions to activate
- Is stable against minor perturbations
- May have a larger effect when finally awakened
- Represents accumulated "potential energy" in the system

The depth measurement helps prioritize which patterns to monitor and what magnitude of change would be required to awaken them.

Value

A list containing:

- depths - Data frame with depth measurements for each pattern
- depth_distribution - Summary statistics of depth distribution
- awakening_effort - Estimated effort required to activate each pattern
- stability_index - Stability measure for each pattern's dormant state

Examples

```
set.seed(42)
n <- 500
x <- rnorm(n)
# Create a deeply dormant pattern (only active in extreme conditions)
z <- ifelse(abs(x) > 2, 1, 0)
y <- ifelse(z == 1, 0.9 * x + rnorm(sum(z), 0, 0.1), rnorm(n))
data <- data.frame(x = x, y = y)

result <- dormancy_detect(data, method = "threshold")
depths <- dormancy_depth(result)
print(depths)
```

---

dormancy_detect *Detect Dormant Patterns in Multivariate Data*

Description

Identifies dormant patterns in multivariate data - statistical relationships that exist but are currently inactive. Dormant patterns only manifest when specific trigger conditions emerge in the data.

Usage

```
dormancy_detect(
  data,
  threshold = 0.3,
  method = "conditional",
  n_bins = 10,
  min_cluster = 0.05,
  parallel = FALSE,
  verbose = FALSE
)
```

Arguments

`data`	A numeric matrix or data frame with observations in rows and variables in columns.
`threshold`	Numeric. The minimum dormancy score for a pattern to be considered significant. Default is 0.3.
`method`	Character. The detection method to use. Options are: • "conditional" - Detects patterns that are conditionally suppressed (active only under specific conditions) • "threshold" - Detects patterns that emerge when variables cross certain thresholds • "phase" - Detects patterns that exist in specific phase regions of the data space • "cascade" - Detects cascade-ready patterns that could trigger chain reactions Default is "conditional".
`n_bins`	Integer. Number of bins for discretizing continuous variables when searching for conditional patterns. Default is 10.
`min_cluster`	Numeric. Minimum proportion of observations that must be in a region for it to be considered. Default is 0.05 (5%).
`parallel`	Logical. Whether to use parallel processing. Default is FALSE.
`verbose`	Logical. Whether to print progress messages. Default is FALSE.

Details

Dormant patterns are fundamentally different from weak or spurious correlations. A dormant pattern is a genuine relationship that is currently suppressed by prevailing conditions in the data. The key insight is that:

1. The pattern exists (verified through subset analysis)
2. The pattern is currently inactive (not visible in aggregate statistics)
3. The pattern can "awaken" when conditions change

The detection algorithm works by:

1. Segmenting the data space into regions
2. Computing pairwise relationships in each region
3. Identifying relationships that vary significantly across regions
4. Flagging relationships that are strong in some regions but weak overall
5. Estimating the conditions under which dormant patterns would activate

Value

A list of class "dormancy" containing:

- `patterns` - A data frame of detected dormant patterns with columns: variables, dormancy_score, trigger_variable, trigger_region, activation_risk
- `data` - The input data
- `method` - The detection method used
- `threshold` - The threshold used
- `conditional_regions` - List of regions where patterns are dormant
- `metadata` - Additional information about the detection process

See Also

`dormancy_trigger` for identifying activation triggers, `dormancy_depth` for measuring dormancy depth, `awaken` for simulating pattern activation

Examples

```
# Create data with a dormant pattern
set.seed(42)
n <- 1000
x <- rnorm(n)
z <- sample(c(0, 1), n, replace = TRUE)
# Relationship between x and y only exists when z == 1
y <- ifelse(z == 1, 0.8 * x + rnorm(n, 0, 0.3), rnorm(n))
data <- data.frame(x = x, y = y, z = factor(z))

# Detect dormant patterns
result <- dormancy_detect(data, method = "conditional")
print(result)
```

`dormancy_risk` *Assess Risk of Dormant Pattern Activation*

Description

Quantifies the risk associated with dormant patterns, including the probability of activation, potential impact, and uncertainty in risk estimates. This function provides actionable risk metrics for decision-making and monitoring priorities.

Usage

```
dormancy_risk(
  dormancy_result,
  depth_result = NULL,
  impact_weights = NULL,
  time_horizon = 1,
  risk_tolerance = 0.3,
  verbose = FALSE
)
```

Arguments

dormancy_result	An object of class "dormancy" from dormancy_detect.
depth_result	Optional. An object of class "dormancy_depth" from dormancy_depth. If provided, uses depth information for more accurate risk assessment.
impact_weights	Optional named vector of weights for different impact types. Default considers symmetric positive/negative impacts.
time_horizon	Numeric. The time horizon for risk assessment (in abstract units). Longer horizons increase activation probability. Default is 1.
risk_tolerance	Numeric. Risk tolerance threshold for flagging. Default is 0.3.
verbose	Logical. Whether to print progress messages. Default is FALSE.

Details

Risk assessment for dormant patterns considers multiple dimensions:

- **Activation Probability**: Likelihood that trigger conditions will be met in the given time horizon
- **Impact Magnitude**: Expected effect size if the pattern activates
- **Impact Direction**: Whether activation would be beneficial, harmful, or neutral
- **Cascade Potential**: Risk of triggering other patterns
- **Uncertainty**: Confidence in risk estimates

The risk score combines these dimensions into an actionable metric:

$$Risk = P(activation) \times Impact \times CascadeFactor \times (1 + Uncertainty)$$

Value

A list containing:

- risk_scores - Data frame with risk metrics for each pattern
- risk_matrix - Matrix of activation probability x impact
- priorities - Ordered list of patterns by risk priority
- recommendations - Risk management recommendations
- summary - Overall risk summary statistics

Examples

```
set.seed(42)
n <- 500
x <- rnorm(n)
z <- sample(c(0, 1), n, replace = TRUE)
y <- ifelse(z == 1, 0.8 * x + rnorm(sum(z), 0, 0.3), rnorm(n))
data <- data.frame(x = x, y = y, z = factor(z))

result <- dormancy_detect(data, method = "conditional")
risk <- dormancy_risk(result, time_horizon = 2)
print(risk)
```

`dormancy_scout` *Scout for Dormant Pattern Regions in Data*

Description

Systematically scans the data space to identify regions where dormant patterns might emerge. Unlike `dormancy_detect` which identifies specific patterns, `dormancy_scout` maps the "terrain" of dormancy potential.

Usage

```
dormancy_scout(
  data,
  grid_resolution = 20,
  scout_method = "density",
  return_map = TRUE,
  verbose = FALSE
)
```

Arguments

`data`	A numeric matrix or data frame.
`grid_resolution`	Integer. Resolution of the scanning grid. Higher values give finer resolution but slower computation. Default is 20.
`scout_method`	Character. Scanning method: • "density" - Identifies low-density regions where patterns might hide • "variance" - Identifies high-variance regions with pattern potential • "correlation" - Maps local correlation landscapes • "entropy" - Identifies high-entropy regions with dormancy potential Default is "density".
`return_map`	Logical. Whether to return the full dormancy map. Default is TRUE.
`verbose`	Logical. Whether to print progress messages. Default is FALSE.

Details

Scout analysis is useful for:

- Identifying regions to monitor for future pattern emergence
- Understanding the "geography" of your data's pattern space
- Finding data regions that are underexplored or anomalous
- Planning targeted data collection in high-potential regions

The scout creates a map of "dormancy potential" - not actual patterns, but locations where patterns are more likely to exist or emerge.

Value

A list containing:

- `scout_results` - Data frame with coordinates and dormancy potential
- `hotspots` - Regions with highest dormancy potential
- `dormancy_map` - If `return_map = TRUE`, a matrix representing the dormancy landscape
- `summary` - Summary statistics of the scan

Examples

```
set.seed(42)
n <- 500
x <- rnorm(n)
y <- rnorm(n)
# Create a region with hidden pattern
z <- ifelse(x > 1 & y > 1, 0.9 * x + rnorm(sum(x > 1 & y > 1), 0, 0.1), y)
data <- data.frame(x = x, y = z)

scout <- dormancy_scout(data, grid_resolution = 15)
print(scout)
```

---

dormancy_trigger *Identify Trigger Conditions for Dormant Patterns*

Description

Analyzes detected dormant patterns to identify the specific conditions under which they would activate. This function goes beyond detection to characterize the precise trigger mechanisms and their sensitivity.

Usage

```
dormancy_trigger(
  dormancy_result,
  sensitivity = 0.5,
  n_bootstrap = 100,
  verbose = FALSE
)
```

Arguments

dormancy_result	An object of class "dormancy" from dormancy_detect.
sensitivity	Numeric. The sensitivity level for trigger detection, ranging from 0 (low sensitivity, only major triggers) to 1 (high sensitivity, minor triggers included). Default is 0.5.
n_bootstrap	Integer. Number of bootstrap samples for confidence intervals. Default is 100.
verbose	Logical. Whether to print progress messages. Default is FALSE.

Details

Trigger identification is crucial for risk management and early warning systems. A dormant pattern might be triggered by:

- **Threshold triggers**: When a variable crosses a specific value
- **Region triggers**: When observations fall within specific data regions
- **Categorical triggers**: When a categorical condition is met
- **Compound triggers**: When multiple conditions align
- **Temporal triggers**: When time-dependent patterns emerge

The function uses bootstrap resampling to provide confidence intervals around trigger estimates, ensuring robust identification even with limited data.

Value

A list containing:

- `triggers` - A data frame with trigger details: pattern_id, trigger_variable, trigger_type, threshold_value, sensitivity_score, confidence_lower, confidence_upper
- `trigger_map` - A matrix showing trigger relationships
- `recommendations` - Character vector of actionable insights

See Also

dormancy_detect, dormancy_risk

Examples

```
set.seed(42)
n <- 500
x <- rnorm(n)
z <- sample(c(0, 1), n, replace = TRUE)
y <- ifelse(z == 1, 0.8 * x + rnorm(n, 0, 0.3), rnorm(n))
data <- data.frame(x = x, y = y, z = factor(z))

result <- dormancy_detect(data, method = "conditional")
triggers <- dormancy_trigger(result)
print(triggers)
```

`hibernate` *Identify Patterns That Have Become Dormant Over Time*

Description

Analyzes time series or sequential data to identify patterns that were once active but have become dormant. This is the inverse problem from `dormancy_detect` - finding patterns that "went to sleep" rather than patterns that are currently dormant.

Usage

```
hibernate(
  data,
  time_var = NULL,
  window_size = 0.2,
  threshold = 0.3,
  min_observations = 30,
  verbose = FALSE
)
```

Arguments

`data`	A data frame with a time column or sequential index.
`time_var`	Character. Name of the time/sequence variable. If NULL, uses row order as sequence. Default is NULL.
`window_size`	Integer or numeric. Size of the rolling window for detecting changes. If integer, uses number of observations. If numeric < 1, uses proportion of data. Default is 0.2 (20% of data).
`threshold`	Numeric. Minimum change in pattern strength to be considered hibernation. Default is 0.3.
`min_observations`	Integer. Minimum observations required for analysis. Default is 30.
`verbose`	Logical. Whether to print progress messages. Default is FALSE.

Details

Hibernation detection is important for:

- Understanding system evolution and regime changes
- Identifying lost relationships that might return
- Detecting structural breaks in relationships
- Monitoring degradation of system components

A pattern is considered to have "hibernated" if:

1. It was strong in an earlier time window
2. It has weakened significantly in recent windows
3. The weakening is not due to noise or reduced sample size

Value

A list containing:

- `hibernated_patterns` - Patterns that have become dormant
- `timeline` - When patterns transitioned to dormancy
- `hibernation_depth` - How deeply patterns have hibernated
- `revival_potential` - Likelihood patterns could reawaken

Examples

```
set.seed(42)
n <- 500
time <- 1:n
x <- rnorm(n)
# Relationship that fades over time
effect_strength <- exp(-time / 200)
y <- effect_strength * 0.8 * x + (1 - effect_strength) * rnorm(n)
data <- data.frame(time = time, x = x, y = y)

hib <- hibernate(data, time_var = "time", window_size = 0.15)
print(hib)
```

`plot.dormancy` *Plot Methods for Dormancy Objects*

Description

Visualization methods for dormancy analysis results. Creates informative plots showing dormant patterns, trigger regions, and risk landscapes.

Usage

```
## S3 method for class 'dormancy'
plot(x, type = "overview", ...)

## S3 method for class 'dormancy_map'
plot(x, type = "overview", ...)
```

Arguments

`x`	An object of class "dormancy", "dormancy_map", "dormancy_depth", "dormancy_risk", or "awakening".
`type`	Character. Type of plot to generate: • "overview" - General overview of results • "patterns" - Focus on detected patterns • "risk" - Risk-focused visualization • "timeline" - Time-based visualization (if applicable)
`...`	Additional arguments passed to plot functions.

Value

Invisibly returns the plot data.

Examples

```
set.seed(42)
n <- 500
x <- rnorm(n)
z <- sample(c(0, 1), n, replace = TRUE)
y <- ifelse(z == 1, 0.8 * x + rnorm(sum(z), 0, 0.3), rnorm(n))
data <- data.frame(x = x, y = y, z = factor(z))

result <- dormancy_detect(data, method = "conditional")
plot(result)
```

Index