Package: resLIK (via r-universe)

November 9, 2026

Type Package

Title Representation-Level Control Surfaces for Reliability Sensing

Version 0.1.2

Description Implements the Representation-Level Control Surfaces (RLCS) paradigm for ensuring the reliability of autonomous systems and AI models. It provides three deterministic sensors: Residual Likelihood (ResLik) for population-level anomaly detection, Temporal Consistency Sensor (TCS) for drift and shock detection, and Agreement Sensor for multi-modal redundancy checks. These sensors feed into a standardized control surface that issues 'PROCEED', 'DEFER', or 'ABSTAIN' signals based on strict safety invariants, allowing systems to detect and react to out-of-distribution states, sensor failures, and environmental shifts before they propagate to decision-making layers.

License MIT + file LICENSE

Encoding UTF-8

Imports utils

Suggests testthat, stats, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author MD. Arshad [aut, cre]

Maintainer MD. Arshad <arshad10867c@gmail.com>

Repository https://sulkysubject37.r-universe.dev

Date/Publication 2026-02-06 14:10:02 UTC

RemoteUrl https://github.com/cran/resLIK

RemoteRef HEAD

RemoteSha 9195d712cb2db14734ae05e596d989dba07d3752

Contents

agreement 2
reslik 3
rlcs_control 4
tcs 5

Index **7**

`agreement` *Agreement Sensor*

Description

The Agreement Sensor measures the alignment between two latent representations, typically from different modalities or redundant sensors. It uses cosine similarity to quantify agreement.

Usage

```
agreement(z1, z2, eps = 1e-08)
```

Arguments

`z1`	Numeric vector or matrix. The first representation.
`z2`	Numeric vector or matrix. The second representation. Must have the same shape as `z1`.
`eps`	Numeric. Small constant for numerical stability. Defaults to 1e-8.

Details

Agreement Sensor

The sensor computes the cosine similarity between `z1` and `z2`:

$$A = \frac{z_1 \cdot z_2}{||z_1|| ||z_2|| + \epsilon}$$

Values close to 1 indicate strong agreement, 0 indicates orthogonality (no agreement), and -1 indicates opposition. In the context of RLCS, high positive agreement is generally required for a `PROCEED` signal.

Value

A list containing:

`agreement`	The cosine similarity score (-1 to 1).

Examples

```
# Example 1: Perfect Agreement
z1 <- c(1, 2, 3)
z2 <- c(2, 4, 6)
agreement(z1, z2)

# Example 2: Disagreement (Orthogonal)
z3 <- c(1, 0, 0)
z4 <- c(0, 1, 0)
agreement(z3, z4)
```

`reslik` *Residual Likelihood Sensor*

Description

The Residual Likelihood (ResLik) sensor measures the conformity of a latent representation against a population reference distribution. It acts as a soft gate, suppressing "out-of-distribution" (OOD) signals while preserving "in-distribution" (ID) fidelity.

Usage

```
reslik(z, ref_mean = 0, ref_sd = 1, lambda = 1, tau = 0.05)
```

Arguments

`z`	Numeric vector or matrix. The latent representation to evaluate.
`ref_mean`	Numeric or vector. The reference mean of the population. Defaults to 0.
`ref_sd`	Numeric or vector. The reference standard deviation of the population. Defaults to 1.
`lambda`	Numeric. The sensitivity of the gate. Higher values suppress OOD samples more aggressively. Defaults to 1.0.
`tau`	Numeric. The dead-zone threshold. Discrepancies below this value are ignored. Defaults to 0.05.

Details

Population-Level Sensor (ResLik)

The sensor operates in four steps:

1. **Normalization**: The input `z` is Z-score normalized using `ref_mean` and `ref_sd`.
2. **Discrepancy**: The Mean Absolute Deviation (MAD) is computed for each sample.
3. **Gating**: A gating factor is computed as `exp(-lambda * max(0, discrepancy - tau))`. This creates a "dead-zone" `tau` where minor deviations are ignored.
4. **Output**: The original `z` is scaled by the gating factor.

This implementation is fully deterministic and stateless.

Value

A list containing:

gated	The gated representation (same shape as z).
diagnostics	A list of diagnostic metrics:

- discrepancy: The raw discrepancy scores.
- max_discrepancy: The maximum discrepancy in the batch.
- mean_discrepancy: The mean discrepancy in the batch.

Examples

```
# Example 1: In-Distribution Sample
z_id <- c(0.1, -0.2, 0.05)
out_id <- reslik(z_id)
print(out_id$gated) # Should be close to z_id

# Example 2: Out-of-Distribution Sample
z_ood <- c(5.0, 5.0, 5.0)
out_ood <- reslik(z_ood)
print(out_ood$gated) # Should be strongly suppressed
```

---

rlcs_control *RLCS Control Surface*

Description

The Control Surface integrates inputs from multiple reliability sensors (ResLik, TCS, Agreement) to issue a standardized control signal (PROCEED, DEFER, ABSTAIN). It uses a deterministic, rule-based logic to ensure safety and predictability.

Usage

```
rlcs_control(reslik, tcs = NULL, agreement = NULL, thresholds = list())
```

Arguments

reslik	List. The output from the reslik() function.
tcs	List (Optional). The output from the tcs() function.
agreement	List (Optional). The output from the agreement() function.
thresholds	List. Custom thresholds to override defaults:

- reslik_max_disc (default 3.0)
- tcs_consistency (default 0.2)
- agreement (default 0.3)

Details

Deterministic Control Surface

The logic implements a "Conservative OR" strategy:

- **ABSTAIN**: Triggered if `ResLik` discrepancy exceeds `reslik_max_disc`. This indicates the input is fundamentally invalid (e.g., sensor failure).
- **DEFER**: Triggered if `TCS` consistency is below `tcs_consistency` OR `Agreement` is below `agreement`. This indicates valid but unstable or conflicting data.
- **PROCEED**: Default state if no negative flags are raised.

Value

A character vector of signals (same length as input batch).

Examples

```
# Mock Inputs
res_pass <- list(diagnostics = list(discrepancy = c(0.1), max_discrepancy = 0.1))
res_fail <- list(diagnostics = list(discrepancy = c(5.0), max_discrepancy = 5.0))
tcs_pass <- list(consistency = c(0.9))
tcs_fail <- list(consistency = c(0.1))

# Scenario 1: All Good
rlcs_control(res_pass, tcs_pass)

# Scenario 2: ResLik Fail -> ABSTAIN
rlcs_control(res_fail, tcs_pass)

# Scenario 3: TCS Fail -> DEFER
rlcs_control(res_pass, tcs_fail)
```

---

tcs *Temporal Consistency Sensor*

Description

The Temporal Consistency Sensor (TCS) monitors the evolution of a latent representation over time. It quantifies "drift" as the relative rate of change and converts it into a stability score.

Usage

```
tcs(z_t, z_prev, eps = 1e-06)
```

Arguments

z_t	Numeric vector or matrix. The current latent representation.
z_prev	Numeric vector or matrix. The previous latent representation. Must have the same shape as z_t.
eps	Numeric. Small constant to avoid division by zero. Defaults to 1e-6.

Details

Temporal Consistency Sensor

The sensor computes the L2 distance between the current state z_t and the previous state z_prev. This distance is normalized by the magnitude of the previous state to produce a relative drift score. Consistency is defined as exp(-drift).

This metric is essential for detecting "shock" events where the representation changes too rapidly for the downstream system to adapt safely.

Value

A list containing:

drift	The relative drift score (non-negative).
consistency	The consistency score (0 to 1).

Examples

```
# Example 1: Stable Evolution
z_t0 <- c(1.0, 0.0)
z_t1 <- c(1.01, 0.01)
tcs(z_t1, z_t0)

# Example 2: Sudden Shock
z_shock <- c(5.0, 0.0)
tcs(z_shock, z_t0)
```

Index

agreement, 2

reslik, 3
rlcs_control, 4

tcs, 5