

Package ‘citydistR’

October 7, 2026

Type Package

Title City-Adaptive Distance Modelling Utilities

Version 0.1.0

Author Elif Kozan [aut, cre, cph]

Maintainer Elif Kozan <elif.kozan@ege.edu.tr>

Description Tools for distribution-aware and city-adaptive analysis of learning-based road-network distance estimates. Provides detour-factor diagnostics, a Topological Predictability Index, upper-tail summaries, modular robust losses, hybrid objective components, adaptive validation weights, and multi-criteria model evaluation. The functions are designed as reusable building blocks rather than a fixed model specification.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-10-07 08:10:07 UTC

Contents

adaptive_validation_score	2
adaptive_weights	3
city_indices	4
combine_objectives	4
compare_distance_models	5
detour_factor	6
evaluate_distance_model	7
hybrid_objective	7
median_df_regularization	9

point_loss	9
robust_loss_active	10
tail95	11
tpi	11
Index	13

adaptive_validation_score
<i>City-Adaptive Validation Score</i>

Description

Combines normalized MAE, normalized P95 absolute error, and Spearman rank correlation. Lower values indicate better validation performance.

Usage

```
adaptive_validation_score(  
  true_distance,  
  pred_distance,  
  tail95_value,  
  tpi_value,  
  tail95_sat = 2,  
  lambda_max = 1,  
  gamma_max = 1,  
  eps = 1e-08  
)
```

Arguments

- true_distance Numeric vector of true distances.
- pred_distance Numeric vector of predicted distances.
- tail95_value Numeric Tail95 value computed from training data.
- tpi_value Numeric TPI value computed from training data.
- tail95_sat Tail95 saturation constant.
- lambda_max Maximum tail-risk weight.
- gamma_max Maximum ranking-consistency weight.
- eps Small positive constant.

Value

A named numeric vector containing the score and its components.

Examples

```
adaptive_validation_score(  
  c(10, 20, 30),  
  c(11, 19, 32),  
  tail95_value = 1.5,  
  tpi_value = 0.8  
)
```

adaptive_weights*City-Adaptive Validation Weights*

Description

Computes tail-risk and ranking-consistency weights from Tail95 and TPI.

Usage

```
adaptive_weights(  
  tail95_value,  
  tpi_value,  
  tail95_sat = 2,  
  lambda_max = 1,  
  gamma_max = 1  
)
```

Arguments

tail95_value	Numeric Tail95 value.
tpi_value	Numeric TPI value.
tail95_sat	Tail95 saturation constant; must exceed 1.
lambda_max	Maximum tail-risk weight.
gamma_max	Maximum ranking-consistency weight.

Value

A named numeric vector containing lambda, gamma, and normalized_tail.

Examples

```
adaptive_weights(1.6, 0.78)
```

city_indices	<i>City-Level Detour Diagnostics</i>
--------------	--------------------------------------

Description

Returns TPI, Tail95, and descriptive statistics for detour factors.

Usage

```
city_indices(
  network_distance = NULL,
  euclidean_distance = NULL,
  detour = NULL,
  eps = 1e-08,
  na.rm = TRUE
)
```

Arguments

network_distance	Optional numeric vector of network distances.
euclidean_distance	Optional numeric vector of Euclidean distances.
detour	Optional numeric vector of pre-computed detour factors.
eps	Small positive constant.
na.rm	Logical; remove missing and non-finite values.

Value

A one-row data frame containing city-level diagnostics.

Examples

```
city_indices(detour = c(1.0, 1.1, 1.2, 1.4, 2.5))
```

combine_objectives	<i>Combine Objective Components</i>
--------------------	-------------------------------------

Description

Combines scalar objective components with configurable weights.

Usage

```
combine_objectives(components, weights = NULL)
```

Arguments

components	Named or unnamed numeric vector of scalar objective components.
weights	Optional named or positional numeric vector of weights.

Value

A single weighted objective value.

Examples

```
combine_objectives(c(a = 1, b = 2), c(a = 2, b = 3))
```

`compare_distance_models`*Compare Distance-Prediction Models*

Description

Evaluates multiple named prediction vectors using a common metric set.

Usage

```
compare_distance_models(true_distance, predictions, p = 0.95)
```

Arguments

true_distance	Numeric vector of true distances.
predictions	Named list of numeric prediction vectors.
p	Tail probability for high-quantile absolute error.

Value

A data frame containing one row per model.

Examples

```
compare_distance_models(  
  c(10, 20, 30),  
  list(a = c(11, 19, 31), b = c(12, 18, 32))  
)
```

detour_factor	<i>Compute Detour Factors</i>
---------------	-------------------------------

Description

Computes the ratio between network distance and Euclidean distance.

Usage

```
detour_factor(
  network_distance,
  euclidean_distance,
  eps = 1e-08,
  min_df = 0,
  max_df = Inf,
  invalid = c("na", "error")
)
```

Arguments

network_distance	Numeric vector of network or shortest-path distances.
euclidean_distance	Numeric vector of Euclidean distances.
eps	Small positive constant used for numerical protection.
min_df	Lower clipping bound for valid detour factors.
max_df	Upper clipping bound for valid detour factors.
invalid	How to handle non-positive Euclidean distances: "na" or "error".

Value

A numeric vector of detour factors.

Examples

```
detour_factor(c(12, 25), c(10, 20))
```

 evaluate_distance_model

Evaluate Distance Predictions

Description

Computes average, tail, bias, and ranking metrics for distance predictions.

Usage

```
evaluate_distance_model(true_distance, pred_distance, p = 0.95)
```

Arguments

`true_distance` Numeric vector of true distances.
`pred_distance` Numeric vector of predicted distances.
`p` Tail probability used for the high-quantile absolute error.

Value

A one-row data frame with sample size, MAE, RMSE, P95, bias, and Spearman correlation.

Examples

```
evaluate_distance_model(c(10, 20, 30), c(11, 18, 33))
```

 hybrid_objective

Modular Hybrid Objective

Description

Combines distance loss, log-detour loss, and median-based detour regularization. The point-loss family is configurable.

Usage

```
hybrid_objective(
  true_distance,
  pred_distance,
  euclidean_distance,
  beta_logdf = 1,
  beta_median = 0.1,
  loss = c("auto", "mse", "mae", "huber", "logcosh"),
  tail95_value = NULL,
  robust_threshold = 1.35,
```

```

    delta = 1,
    standardize = TRUE,
    eps = 1e-08,
    df_min = 1e-08,
    df_max = Inf
  )

```

Arguments

<code>true_distance</code>	Numeric vector of true network distances.
<code>pred_distance</code>	Numeric vector of predicted network distances.
<code>euclidean_distance</code>	Numeric vector of Euclidean distances.
<code>beta_logdf</code>	Weight for the log-detour objective component.
<code>beta_median</code>	Weight for the median regularization component.
<code>loss</code>	Point-loss choice: "auto", "mse", "mae", "huber", or "logcosh".
<code>tail95_value</code>	Optional Tail95 value used when <code>loss = "auto"</code> .
<code>robust_threshold</code>	Tail95 threshold used for automatic Huber activation.
<code>delta</code>	Positive Huber transition parameter.
<code>standardize</code>	Logical; standardize distance and log-detour targets before point-loss calculation.
<code>eps</code>	Small positive constant.
<code>df_min</code>	Lower clipping bound for detour factors.
<code>df_max</code>	Upper clipping bound for detour factors.

Value

A list containing total loss, component losses, and the selected point-loss method.

Examples

```

hybrid_objective(
  c(12, 18, 31),
  c(11, 20, 29),
  c(10, 15, 25),
  loss = "mse"
)

```

median_df_regularization

Median-Based Detour Regularization

Description

Penalizes dispersion of predicted detour factors around their median.

Usage

```
median_df_regularization(
  pred_distance,
  euclidean_distance,
  eps = 1e-08,
  df_min = 0,
  df_max = Inf
)
```

Arguments

`pred_distance` Numeric vector of predicted network distances.

`euclidean_distance` Numeric vector of Euclidean distances.

`eps` Small positive constant.

`df_min` Lower clipping bound for predicted detour factors.

`df_max` Upper clipping bound for predicted detour factors.

Value

A single numeric regularization value.

Examples

```
median_df_regularization(c(12, 18, 26), c(10, 15, 20))
```

point_loss

Pointwise Loss Aggregator

Description

Computes a mean pointwise loss from residuals using a selectable loss family.

Usage

```
point_loss(
  residuals,
  method = c("mse", "mae", "huber", "logcosh"),
  delta = 1,
  na.rm = TRUE
)
```

Arguments

residuals	Numeric residual vector.
method	Loss family: "mse", "mae", "huber", or "logcosh".
delta	Positive Huber transition parameter.
na.rm	Logical; remove missing and non-finite residuals.

Value

A single numeric loss value.

Examples

```
point_loss(c(-2, -1, 0, 1, 2), "huber", delta = 1)
```

robust_loss_active	<i>Robust-Loss Activation Rule</i>
--------------------	------------------------------------

Description

Checks whether Tail95 reaches a configurable robust-loss activation threshold.

Usage

```
robust_loss_active(tail95_value, threshold = 1.35)
```

Arguments

tail95_value	Numeric Tail95 value.
threshold	Numeric activation threshold.

Value

A logical value.

Examples

```
robust_loss_active(1.42)
```

tail95	<i>Tail95 Detour-Heaviness Index</i>
--------	--------------------------------------

Description

Computes an upper-quantile detour-factor summary relative to the median.

Usage

```
tail95(detour, prob = 0.95, eps = 1e-08, na.rm = TRUE)
```

Arguments

detour	Numeric vector of detour factors.
prob	Quantile probability.
eps	Small positive constant for numerical protection.
na.rm	Logical; remove missing and non-finite values.

Value

A single numeric tail-heaviness value.

Examples

```
tail95(c(1.0, 1.1, 1.2, 1.4, 2.5))
```

tpi	<i>Topological Predictability Index</i>
-----	---

Description

Computes the Topological Predictability Index (TPI) from detour factors.

Usage

```
tpi(detour, eps = 1e-08, na.rm = TRUE)
```

Arguments

detour	Numeric vector of detour factors.
eps	Small positive constant.
na.rm	Logical; remove missing and non-finite values.

Value

A single numeric TPI value.

Examples

```
tpi(c(1.0, 1.1, 1.3, 1.8))
```

Index

`adaptive_validation_score`, [2](#)
`adaptive_weights`, [3](#)

`city_indices`, [4](#)
`combine_objectives`, [4](#)
`compare_distance_models`, [5](#)

`detour_factor`, [6](#)

`evaluate_distance_model`, [7](#)

`hybrid_objective`, [7](#)

`median_df_regularization`, [9](#)

`point_loss`, [9](#)

`robust_loss_active`, [10](#)

`tail95`, [11](#)
`tpi`, [11](#)