# Package: CKNNRLD (via r-universe)

May 29, 2026

**Title** Clustering-Based K-Nearest Neighbor Regression for Longitudinal Data

**Version** 0.1.2

**Description** Implements the 'CKNNRLD' algorithm (Clustering-Based K-Nearest Neighbor Regression for Longitudinal Data) for improving K-Nearest Neighbor ('KNN') regression on longitudinal data through cluster-based partitioning and localized prediction. Offers enhanced computational efficiency and accuracy for high-volume longitudinal datasets. The clustering is performed using the 'latrend' package, which provides a unified interface for various longitudinal clustering methods including 'KML' (K-Means for Longitudinal data). The acronym 'KNN' stands for K-Nearest Neighbor. The acronym 'KML' stands for K-Means for Longitudinal data. References: Loeloe MS, Tabatabaei SM, Sefidkar R, Mehrparvar AH, Jambarsang S (2025). ``Boosting K-nearest neighbor regression performance for longitudinal data through a novel learning approach." BMC Bioinformatics, 26, 232. <doi:10.1186/s12859-025-06205-1>; Genolini C, Falissard B (2010). ``KmL: k-means for longitudinal data." Computational Statistics, 25(2), 317-328. <doi:10.1007/s00180-009-0178-4>.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** Directional, graphics, Rfast, latrend

**Depends** R (>= 3.5.0)

**NeedsCompilation** no

**Language** en-US

**Author** Mohammad Sadegh Loeloe [aut, cre], Seyyed Mohammad Tabatabaei [aut], Reyhane Sefidkar [aut], Amir Houshang Mehrparvar [aut], Sara Jambarsang [aut, ths]

**Maintainer** Mohammad Sadegh Loeloe <mslbiostat@gmail.com>

**Config/pak/sysreqs** libabsl-dev cmake libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglu1-mesa-dev make texlive libpng-dev libuv1-dev libgl1-mesa-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

**Repository** https://mslbiostat.r-universe.dev

**Date/Publication** 2026-05-28 18:40:02 UTC

**RemoteUrl** https://github.com/cran/CKNNRLD

**RemoteRef** HEAD

**RemoteSha** 858e0778980076789089b8dbdda1900c50cd1501

# Contents



---

`BestC` *Find Optimal Number of Clusters for Longitudinal Data*

---

## Description

This function determines the best number of clusters (C) for longitudinal data clustering based on internal validation indices using the latrend package.

## Usage

```
BestC(Y, range_clusters = 2:6, method = "kml")
```

## Arguments

| | |
|---|---|
| `Y` | A matrix or data frame of longitudinal outcomes (subjects x timepoints). |
| `range_clusters` | A numeric vector of cluster numbers to evaluate (e.g., 2:6). |
| `method` | Clustering method to use. Currently supports "kml" (default). |

## Value

A list with:

| | |
|---|---|
| `best_c` | Optimal number of clusters |
| `criteria` | Data frame of criteria values for each cluster number |
| `criteria_best` | Criteria values for the best cluster number |

## Examples

```
set.seed(123)
n <- 30
T <- 3
y <- matrix(rnorm(n * T), nrow = n)
best_c_info <- BestC(Y = y, range_clusters = 2:4)
print(best_c_info$best_c)
```

---

CKNNRLD *Cluster-based KNN Regression for Longitudinal Data (CKNNRLD)*

---

## Description

This function implements a clustering-based KNN regression method designed for longitudinal datasets. It first clusters the training data using longitudinal clustering (via latrend) and then performs KNN regression within each selected cluster.

## Usage

```
CKNNRLD(xnew, y, x, k = 5, c = 4, cluster_method = "kml")
```

## Arguments

| | |
|---|---|
| xnew | A matrix of predictor values for test data (new observations). |
| y | A matrix or data frame of longitudinal responses (subjects x timepoints). |
| x | A matrix or data frame of predictors for training data. |
| k | Number of nearest neighbors to use. Can be a scalar (same k for all clusters) or a vector (different k per cluster). |
| c | Number of clusters for longitudinal clustering. |
| cluster_method | Clustering method to use. Currently supports "kml" (default). |

## Value

A data frame with predicted values and cluster assignment for each observation in xnew. Columns: cluster, Y1, Y2, ..., Yd (where d = number of timepoints).

## Examples

```
set.seed(123)
n <- 30
T <- 3
d <- 2
x <- matrix(runif(n * d), nrow = n)
y <- matrix(rnorm(n * T), nrow = n)
train_idx <- sample(1:n, 20)
```

```
test_idx <- setdiff(1:n, train_idx)
result <- CKNNRLD(
  x = x[train_idx, ],
  y = y[train_idx, ],
  xnew = x[test_idx, ],
  k = 3,
  c = 2
)
head(result)
```

---

CKNNRLD.tune *Tune CKNNRLD Model with Automatic Cluster Selection*

---

## Description

Automatically selects the best number of clusters (C) using internal validation criteria, then tunes the CKNNRLD model within each cluster via cross-validation.

## Usage

```
CKNNRLD.tune(
  y,
  x,
  nfolds = 10,
  folds = NULL,
  seed = NULL,
  A = 10,
  C_range = 2:6,
  cluster_method = "kml"
)
```

## Arguments

| | |
|---|---|
| y | Matrix of longitudinal outcomes (subjects x timepoints). |
| x | Matrix of predictor variables (subjects x features). |
| nfolds | Number of folds for cross-validation (default = 10). |
| folds | Optional list of pre-specified fold indices. |
| seed | Random seed for reproducibility. |
| A | Maximum number of neighbors to evaluate (searches from 2 to A, default = 10). |
| C_range | Range of cluster numbers to evaluate (default = 2:6). |
| cluster_method | Clustering method to use. Currently supports "kml" (default). |

## Value

A list containing:

| | |
|---|---|
| best_c | Optimal number of clusters |
| cluster_results | Tuning results for each cluster (from KNNRLD.tune) |
| cluster_sizes | Size of each cluster |
| cluster_assignments | Cluster labels for each subject |
| criteria | Data frame of quality criteria used for selecting best C |

## Examples

```
set.seed(123)
n <- 30
T <- 3
d <- 2
x <- matrix(runif(n * d), nrow = n)
y <- matrix(rnorm(n * T), nrow = n)
tune_result <- CKNNRLD.tune(
  y = y,
  x = x,
  nfolds = 3,
  A = 4,
  C_range = 2:3
)
print(tune_result$best_c)
```

---

# KNNRLD *Standard K-Nearest Neighbor Regression for Longitudinal Data*

---

## Description

This function performs KNN regression for longitudinal data without clustering. It predicts longitudinal outcomes for new observations based on the weighted average of their k nearest neighbors in the predictor space.

## Usage

```
KNNRLD(xnew, y, x, k = 5)
```

## Arguments

| | |
|---|---|
| xnew | A matrix of predictor values for prediction (test set). |
| y | A matrix or data frame of longitudinal responses (training set). |
| x | A matrix or data frame of training predictor values. |
| k | Number of nearest neighbors to use. Can be a scalar or a vector. |

## Value

A list of matrices with predicted values for each value of k. Each matrix has dimensions nrow(xnew) x ncol(y).

## Examples

```
set.seed(123)
n <- 30
T <- 3
d <- 2
x <- matrix(runif(n * d), nrow = n)
y <- matrix(rnorm(n * T), nrow = n)
train_idx <- sample(1:n, 20)
test_idx <- setdiff(1:n, train_idx)
pred <- KNNRLD(
  xnew = x[test_idx, ],
  y = y[train_idx, ],
  x = x[train_idx, ],
  k = 3
)
head(pred[[1]])
```

---

# KNNRLD.tune *Tune k in KNNRLD using Cross-Validation*

---

## Description

Finds the optimal number of neighbors for KNN regression in longitudinal data using k-fold cross-validation. Evaluates k values from 2 to A.

## Usage

```
KNNRLD.tune(
  y,
  x,
  nfolds = 10,
  folds = NULL,
  seed = NULL,
  A = 10,
  graph = FALSE
)
```

## Arguments

| | |
|---|---|
| y | Matrix of longitudinal outcomes (subjects x timepoints). |
| x | Matrix of predictor variables (subjects x features). |

| | |
|---|---|
| nfolds | Number of cross-validation folds (default = 10). |
| folds | Optional list of pre-specified fold indices. If provided, nfolds is ignored. |
| seed | Optional random seed for reproducibility. |
| A | Maximum number of neighbors to evaluate (searches from 2 to A, default = 10). |
| graph | Logical; if TRUE, plots MSPE vs. k. |

## Value

A list containing:

| | |
|---|---|
| crit | Mean squared prediction error (MSPE) for each k value |
| best_k | Optimal number of neighbors (minimizes MSPE) |
| performance | Minimum MSPE value |
| runtime | Elapsed computation time |

## Examples

```
set.seed(123)
n <- 30
T <- 3
d <- 2
x <- matrix(runif(n * d), nrow = n)
y <- matrix(rnorm(n * T), nrow = n)
tune_result <- KNNRLD.tune(
  y = y,
  x = x,
  nfolds = 3,
  A = 4
)
str(tune_result)
```

# Index