Package: TemporalForest (via r-universe)

May 22, 2026

Title Network-Guided Temporal Forests for Feature Selection in High-Dimensional Longitudinal Data

Version 0.1.4

Description Implements the Temporal Forest algorithm for feature selection in high-dimensional longitudinal data. The method combines time-aware network construction via weighted gene co-expression network analysis (WGCNA), module-based feature screening, and stability selection using tree-based models. This package provides tools for reproducible longitudinal analysis, closely following the methodology described in Shao, Moore, and Ramirez (2025) <https://github.com/SisiShao/TemporalForest>.

License MIT + file LICENSE

URL https://github.com/SisiShao/TemporalForest

BugReports https://github.com/SisiShao/TemporalForest/issues

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports WGCNA, dynamicTreeCut, flashClust, glmertree, partykit, stats, graphics, grDevices,

Suggests knitr, rmarkdown, igraph, Matrix, MASS

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev libuv1-dev

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

Date/Publication 2025-12-22 21:15:08 UTC

RemoteUrl https://github.com/sisishao/temporalforest

RemoteRef HEAD

RemoteSha 0841dc076847f9338c98ca315c66e0df1abaefdd

Contents

`calculate_fs_metrics_cv` *Calculate Feature Selection Metrics*

Description

Computes TP, FP, FN, TN, Sensitivity, Specificity, Precision, and F1-Score.

Usage

```
calculate_fs_metrics_cv(
  selected_vars,
  true_vars_global,
  total_feature_count_p_val
)
```

Arguments

`selected_vars`	A character vector of selected variable names.
`true_vars_global`	A character vector of the true variable names.
`total_feature_count_p_val`	The total number of candidate features (p).

Details

The function handles edge cases where the number of true positives, false positives, or false negatives is zero to avoid division-by-zero errors in precision, sensitivity, and F1-score calculations.

Value

A list containing the following numeric elements:

- `TP`: True Positives
- `FP`: False Positives
- `FN`: False Negatives

- TN: True Negatives
- Sens: Sensitivity (Recall)
- Spec: Specificity
- Prec: Precision
- F1: F1-Score
- N_Selected: Total number of selected variables

Note

Inputs are character vectors; no NA imputation is performed. Division by zero is guarded with 0-valued metrics as documented.

Examples

```
# --- Example for an internal function ---
# Imagine our model selected 3 variables: V1, V2, and V10
selected <- c("V1", "V2", "V10")

# And the "true" important variables were V1, V2, V3, and V4
true_set <- c("V1", "V2", "V3", "V4")

# And the total pool of variables was 50
p <- 50

# Calculate the performance metrics
metrics <- calculate_fs_metrics_cv(
  selected_vars = selected,
  true_vars_global = true_set,
  total_feature_count_p_val = p
)

print(metrics)
#> Expected output:
#> $TP
#> [1] 2
#> $FP
#> [1] 1
#> $FN
#> [1] 2
#> $TN
#> [1] 45
#> $Sens
#> [1] 0.5
#> $Spec
#> [1] 0.9782609
#> $Prec
#> [1] 0.6666667
#> $F1
#> [1] 0.5714286
#> $N_Selected
#> [1] 3
```

`calculate_pred_metrics_cv` *Calculate Prediction Metrics*

Description

Computes Root Mean Squared Error (RMSE) and R-squared.

Usage

```
calculate_pred_metrics_cv(predictions, actual)
```

Arguments

`predictions`	A numeric vector of model predictions.
`actual`	A numeric vector of the true outcome values.

Details

The function is robust to missing values (`NA`) within the input vectors, as they are removed prior to calculation (`na.rm = TRUE`). It also handles multiple edge cases, returning `NA` for both metrics if inputs are `NULL`, empty, contain only `NA`s, or are of unequal length. If the variance of `actual` values is near zero, `R_squared` is handled safely.

Value

A list containing the following numeric elements:

- `RMSE`: Root Mean Squared Error.
- `R_squared`: R-squared (Coefficient of Determination).

Note

Inputs are numeric vectors of equal length. NAs are removed via `na.rm = TRUE`; if inputs are NULL/empty/all-NA/length-mismatch, both metrics are returned as `NA_real_`.

Examples

```
# --- Example for an internal function ---
# Example predicted values from a model
predicted_values <- c(2.5, 3.8, 6.1, 7.9)

# The corresponding actual, true values
actual_values <- c(2.2, 4.1, 5.9, 8.3)

# Calculate the prediction metrics
metrics <- calculate_pred_metrics_cv(
  predictions = predicted_values,
```

```
  actual = actual_values
)

print(metrics)
#> Expected output:
#> $RMSE
#> [1] 0.3082207
#> $R_squared
#> [1] 0.981269
```

`check_temporal_consistency` *Check Consistency of Temporal Predictor Data*

Description

Verifies that the input list 'X' is properly formatted for the main `temporal_forest` function.

Usage

```
check_temporal_consistency(X)
```

Arguments

`X`	A list of numeric matrices or data frames, where each element is expected to represent a time point, with subjects as rows and predictors as columns.

Details

This helper function is called internally by `temporal_forest()` to perform critical input validation before any heavy computation begins. It checks for two main conditions:

1. That the input `X` is a list containing data for at least two time points.
2. That all data frames or matrices in the list have identical column names in the exact same order.

This prevents downstream errors during network construction and modeling, and provides a clear, informative error message to the user if their data format is incorrect.

Value

Returns `TRUE` invisibly if all consistency checks pass. If a check fails, it throws a specific error and stops execution.

Examples

```
# --- 1. A valid input that will pass ---
mat1 <- matrix(1:4, nrow = 2, dimnames = list(NULL, c("V1", "V2")))
mat2 <- matrix(5:8, nrow = 2, dimnames = list(NULL, c("V1", "V2")))
good_X <- list(mat1, mat2)

# This will run silently and return TRUE
check_temporal_consistency(good_X)

# --- 2. An invalid input that will fail ---
mat3 <- matrix(9:12, nrow = 2, dimnames = list(NULL, c("V1", "V3"))) # Mismatched colnames
bad_X <- list(mat1, mat3)

# This will throw an informative error
# We wrap it in try() to prevent the example from stopping
try(check_temporal_consistency(bad_X))
```

---

null_or *Null-safe (coalescing) operator*

Description

Returns the left-hand side (a) if it is not NULL, otherwise returns the right-hand side (b). This operator is useful for setting default values in a concise manner.

Usage

```
a %||% b
```

Arguments

a	The primary object or value.
b	The default object or value to return if a is NULL.

Details

Null-coalescing operator

This implementation mirrors the behavior of %||% used in several R ecosystems (e.g., rlang, purrr) but is defined here for convenience within the **TemporalForest** package.

Value

a if not NULL, otherwise b.

Conflicts

The `%||%` operator name is also used in packages such as **rlang** and **purrr**. If you load those packages after **TemporalForest**, they may mask this operator. To avoid ambiguity, call it explicitly using `TemporalForest::\%||%\“`.

Examples

```
a <- NULL
b <- 5
a %||% b   # Returns 5

x <- 10
y <- 20
x %||% y   # Returns 10

# Safe usage when multiple packages define %||%
TemporalForest::`%||%`(NULL, "default")
```

---

print.TemporalForest *Print Method for TemporalForest Objects*

Description

Print Method for TemporalForest Objects

Usage

```
## S3 method for class 'TemporalForest'
print(x, ...)
```

Arguments

`x`	An object of class `TemporalForest`.
`...`	Additional arguments passed to `print`.

Value

Invisibly returns the input object `x`.

`select_soft_power` *Select the best soft-thresholding power for WGCNA*

Description

Analyze scale-free topology fit across candidate powers and choose the soft-thresholding power β.

Usage

```
select_soft_power(
  data_matrix,
  r2_threshold = 0.8,
  make_plots = FALSE,
  output_dir = tempdir()
)
```

Arguments

`data_matrix`	Numeric matrix/data frame (rows = samples, cols = features).
`r2_threshold`	Target R^2 for the scale-free model (default 0.8).
`make_plots`	Logical; if `TRUE`, write diagnostic PNGs.
`output_dir`	Output directory for plots when `make_plots = TRUE`. Defaults to `tempdir()` to comply with CRAN policies.

Details

The function uses a two-stage heuristic:

1. Choose the smallest β whose scale-free fit R^2 exceeds `r2_threshold`.
2. If no power reaches the target R^2, select the smallest power at the elbow of the curve, approximated by the discrete second derivative

$$\Delta^2 f(p_i) = f(p_{i+1}) - 2f(p_i) + f(p_{i-1}),$$

and pick the p_i with the largest $|\Delta^2 f(p_i)|$.

Value

Integer scalar: the selected soft-thresholding power.

Examples

```
# --- 1. Create a small synthetic data matrix ---
set.seed(123)
example_data <- matrix(rnorm(50 * 100), nrow = 50, ncol = 100)

# --- 2. Run the function to get the selected power ---
best_power <- select_soft_power(example_data)
print(paste("Best soft power:", best_power))
```

`temporal_forest` *Temporal Forest for Longitudinal Feature Selection*

Description

The main user-facing function for the `TemporalForest` package. It performs the complete three-stage algorithm to select a top set of features from high-dimensional longitudinal data.

Usage

```
temporal_forest(
  X = NULL,
  Y,
  id,
  time,
  dissimilarity_matrix = NULL,
  n_features_to_select = 10,
  min_module_size = 4,
  n_boot_screen = 50,
  keep_fraction_screen = 0.25,
  n_boot_select = 100,
  alpha_screen = 0.2,
  alpha_select = 0.05
)
```

Arguments

`X`	A list of numeric matrices, one for each time point. The rows of each matrix should be subjects and columns should be predictors. Required unless `dissimilarity_matrix` is provided.
`Y`	A numeric vector for the longitudinal outcome.
`id`	A vector of subject identifiers.
`time`	A vector of time point indicators.
`dissimilarity_matrix`	An optional pre-computed dissimilarity matrix (e.g., `1 - TOM`). If provided, the network construction step (Stage 1) is skipped. The matrix must be square with predictor names as rownames and colnames. Defaults to `NULL`.
`n_features_to_select`	The number of top features to return in the final selection. This is passed to the `number_selected_final` argument of the internal function. Defaults to 10.
`min_module_size`	The minimum number of features in a module. Passed to the `minClusterSize` argument of the internal function. Defaults to 4.
`n_boot_screen`	The number of bootstrap repetitions for the initial screening stage within modules. Defaults to 50.

`keep_fraction_screen`
: The proportion of features to keep from each module during the screening stage. Defaults to 0.25.

`n_boot_select`
: The number of bootstrap repetitions for the final stability selection stage. Defaults to 100.

`alpha_screen`
: The significance level for splitting in the screening stage trees. Defaults to 0.2.

`alpha_select`
: The significance level for splitting in the selection stage trees. Defaults to 0.05.

Details

The function executes a three-stage process:

1. **Time-Aware Module Construction:** Builds a consensus network across time points to identify modules of stably co-correlated features.
2. **Within-Module Screening:** Uses bootstrapped mixed-effects model trees (`glmertree`) to screen for important predictors within each module.
3. **Stability Selection:** Performs a final stability selection step on the surviving features to yield a reproducible final set.

Unbalanced Panels: The algorithm is robust to unbalanced panel data (i.e., subjects with missing time points). The consensus TOM is constructed using the time points available, and the mixed-effects models naturally handle missing observations.

Outcome Family: The current version is designed for **Gaussian (continuous) outcomes**, as it relies on `glmertree::lmertree`. Support for other outcome families is not yet implemented.

Reproducibility (Determinism): For reproducible results, it is recommended to set a seed using `set.seed()` before running. The algorithm has both stochastic and deterministic components:

- **Stochastic** (depends on `set.seed()`): The bootstrap resampling of subjects in both the screening and selection stages.
- **Deterministic** (does not depend on `set.seed()`): The network construction process (correlation, adjacency, and TOM calculation).

Value

An object of class `TemporalForest` with:

- `top_features` (**character**): the K selected features in descending stability order.
- `candidate_features` (**character**): all features that entered the final (second-stage) selection.

Input contract

- **X**: list of numeric matrices, one per time point; **columns (names and order) must be identical across all time points**. The function does not reorder or reconcile columns.
- **Row order / binding rule**: when rows from `X` are stacked internally, they are assumed to already be in **subject-major × time-minor** order in the user's data. The function does **not** re-order subjects or time.
- **Y, id, time**: vectors of equal length. `id` and `time` may be integer/character/factor; `time` is coerced to a numeric sequence via `as.numeric(as.factor(time))`.

- **Missing values**: this function does **not** perform NA filtering or imputation. Users should pre-clean the data (e.g., `keep <- complete.cases(Y,id,time)`).

Unbalanced panels

Missing time points per subject are allowed **provided the user supplies** `X`**,** `Y`**,** `id`**,** `time` **that already align under the binding rule above**. Stage 1 builds a TOM at the feature level for each available time-point matrix; the **consensus TOM** is the element-wise minimum across time points. Subject-level missingness at a given time does not prevent feature-wise similarity from being computed at other times. This function does not perform any subject-level alignment across time.

Outcome family

Current version targets **Gaussian** outcomes via `glmertree::lmertree`. Other families (e.g., binomial/Poisson) are not supported in this version.

Stability selection and thresholds

Final selection is **top-K** by bootstrap frequency (`K = n_features_to_select`). A probability cutoff (e.g., `pi_thr`) is **not** used and selection probabilities are **not returned** in the current API.

Reproducibility (determinism)

- **Stochastic** (affected by `set.seed()`): bootstrap resampling and tree partitioning.
- **Deterministic**: correlation/adjacency/TOM and consensus-TOM given fixed inputs.

Internal validation

An internal helper `check_temporal_consistency` is called automatically at the start (whenever `dissimilarity_matrix` is `NULL`). It throws an error if column names across time points are not identical (names and order).

Note

The current API does not expose selection probabilities, module labels, or a parameter snapshot; these may be added in a future version.

Author(s)

Sisi Shao, Jason H. Moore, Christina M. Ramirez

References

Shao, S., Moore, J.H., Ramirez, C.M. (2025). Network-Guided Temporal Forests for Feature Selection in High-Dimensional Longitudinal Data. *Journal of Statistical Software*.

See Also

`select_soft_power`, `calculate_fs_metrics_cv`, `calculate_pred_metrics_cv`, `check_temporal_consistency`

Examples

```
# Tiny demo: selects V1, V2, V3 quickly (skips Stage 1 via precomputed A)
set.seed(11)
n_subjects <- 60; n_timepoints <- 2; p <- 20
X <- replicate(n_timepoints, matrix(rnorm(n_subjects * p), n_subjects, p), simplify = FALSE)
colnames(X[[1]]) <- colnames(X[[2]]) <- paste0("V", 1:p)
X_long <- do.call(rbind, X)
id   <- rep(seq_len(n_subjects), each = n_timepoints)
time <- rep(seq_len(n_timepoints), times = n_subjects)
u <- rnorm(n_subjects, 0, 0.7)
eps <- rnorm(length(id), 0, 0.08)
Y <- 4*X_long[,"V1"] + 3.5*X_long[,"V2"] + 3.2*X_long[,"V3"] + rep(u, each = n_timepoints) + eps
A <- 1 - abs(stats::cor(X_long)); diag(A) <- 0
dimnames(A) <- list(colnames(X[[1]]), colnames(X[[1]]))
fit <- temporal_forest(
  X, Y, id, time,
  dissimilarity_matrix = A,
  n_features_to_select = 3,
  n_boot_screen = 6, n_boot_select = 18,
  keep_fraction_screen = 1, min_module_size = 2,
  alpha_screen = 0.5, alpha_select = 0.6
)
print(fit$top_features)
```

Index