Package: grmtree (via r-universe)

July 1, 2026

Title Recursive Partitioning for Graded Response Models

Version 0.2.0

Date 2026-07-01

Maintainer Olayinka I. Arimoro <olayinka.arimoro@ucalgary.ca>

Description Provides methods for recursive partitioning based on the 'Graded Response Model' ('GRM'), extending the 'MOB' algorithm from the 'partykit' package. The package allows for fitting 'GRM' trees that partition the population into homogeneous subgroups based on item response patterns and covariates. Includes specialized plotting functions for visualizing 'GRM' trees with different terminal node displays (threshold regions, parameter profiles, and factor score distributions). The package also implements the Longitudinal GRMTree for detecting response shift in PROMs measured at two time points, embedding a constrained two-factor longitudinal GRM within recursive partitioning, with post-hoc characterization of recalibration and reprioritization. For more details on the methods, see Samejima (1969) <doi:10.1002/J.2333-8504.1968.TB00153.X>, Komboz et al. (2018) <doi:10.1177/0013164416664394> and Arimoro et al. (2025) <doi:10.1007/s11136-025-04018-6>.

License GPL-3

Depends R (>= 4.1.0), partykit (>= 1.2-9), mirt (>= 1.36.1)

Imports stats, graphics, grid, ggplot2, rlang, strucchange

Suggests hlt, dplyr, magrittr, testthat (>= 3.0.0), knitr, rmarkdown, psychotools, psychotree, psych

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/Predicare1/grmtree

BugReports https://github.com/Predicare1/grmtree/issues

VignetteBuilder knitr

LazyData true

Config/pak/sysreqs cmake make libuv1-dev libx11-dev

Repository https://predicare1.r-universe.dev

Date/Publication 2026-07-01 07:53:45 UTC

RemoteUrl https://github.com/predicare1/grmtree

RemoteRef HEAD

RemoteSha ecc763fb8d041fbbe7874dad661da8a0c56e4b33

Contents

discrpar_grmtree . . . 3
discrpar_longitudinal_grmtree . . . 4
fscores_grmtree . . . 5
fscores_longitudinal_grmtree . . . 6
generate_node_scores_dataset . . . 8
grmforest . . . 9
grmforest.control . . . 11
grmtree . . . 12
grmtree.control . . . 16
grmtree_data . . . 17
grmtree_long_data . . . 19
itempar_grmtree . . . 20
itempar_longitudinal_grmtree . . . 21
latentpar_longitudinal_grmtree . . . 23
longitudinal_grmtree . . . 24
plot.grmtree . . . 29
plot.longitudinal_grmtree . . . 31
plot.varimp . . . 33
plot_rs_heatmap . . . 34
plot_rs_tree . . . 35
prepare_longitudinal_data . . . 37
print.grmforest . . . 39
print.grmtree . . . 39
print.rs_characterization . . . 40
rs_characterize . . . 41
threshpar_grmtree . . . 46
threshpar_longitudinal_grmtree . . . 48
varimp . . . 49

Index **51**

discrpar_grmtree *Extract Discrimination Parameters from GRM Tree*

Description

Extracts discrimination parameters (slope parameters) for each item from all terminal nodes of a graded response model tree. The discrimination parameter indicates how well an item distinguishes between respondents with different levels of the latent trait.

Usage

```
discrpar_grmtree(object, node = NULL, ...)
```

Arguments

object	A grmtree object.
node	Optional vector of node IDs to extract from. If NULL (default), extracts from all terminal nodes.
...	Additional arguments (currently unused).

Value

A data.frame with discrimination parameters for each item in each node, with columns:

Node	Node ID
Item	Item name
Discrimination	Discrimination parameter (a1)

See Also

grmtree fits a Graded Response Model Tree, grmforest for GRM Forests, fscores_grmtree for computing factor scores, threshpar_grmtree for extracting threshold parameters, itempar_grmtree for extracting item parameters

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit GRM tree with gender and group as partitioning variables
tree <- grmtree(resp ~ gender + group,
        data = asti,
        control = grmtree.control(minbucket = 30))

# Get all discrimination parameters
discr <- discrpar_grmtree(tree)
```

```
print(discr)
```

`discrpar_longitudinal_grmtree` *Extract Discrimination Parameters from Longitudinal GRM Tree*

Description

Extracts discrimination (slope) parameters for each unique item from all terminal nodes of a longitudinal GRM tree. The longitudinal model uses columns "a1" (T1 loading) and "a2" (T2 loading); this function extracts the T1 discrimination since a1 = a2 under the equality constraint.

Usage

```
discrpar_longitudinal_grmtree(object, node = NULL, clean_names = TRUE, ...)
```

Arguments

`object`	A `longitudinal_grmtree` object.
`node`	Optional vector of node IDs. If NULL (default), extracts from all terminal nodes.
`clean_names`	Logical. If TRUE (default), clean item names.
`...`	Additional arguments (currently unused).

Value

A data.frame with columns:

Node Terminal node ID

Item Item name

Discrimination Discrimination parameter (a1 from T1 items)

See Also

`longitudinal_grmtree` for Phase 1 (tree fitting), `threshpar_longitudinal_grmtree` for extracting threshold parameters for longitudinal GRMTree, `itempar_longitudinal_grmtree` for extracting item parameters for longitudinal GRMTree

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )

  # Phase 1: fit the longitudinal GRM tree
  ltree <- longitudinal_grmtree(
    resp_wide ~ sex + age + residency + job +
      education + comorbidity_count + ever_smoker,
    data = ld, n_items = 8,
    control = grmtree.control(minbucket = 200)
  )

  # Print the discrimination parameters
  discr <- discrpar_longitudinal_grmtree(ltree)
print(discr)
```

`fscores_grmtree` *Compute Latent Factor Scores for Each Terminal Node in a GRM Tree*

Description

This function calculates latent factor scores for each terminal node in a GRM tree object using specified scoring method (EAP, MAP, ML, or WLE).

Usage

```
fscores_grmtree(grmtree_obj, method = "EAP")
```

Arguments

`grmtree_obj`	A GRM tree object (from `grmtree()` function) containing fitted models in its terminal nodes.
`method`	Scoring method to use: "EAP" (default), "MAP", "ML", or "WLE". See `mirt::fscores()` for details.

Value

A named list where each element contains the factor scores for a terminal node. Names correspond to node IDs. Returns NULL for nodes where computation fails. If no scores can be computed for any node, returns NULL with a warning.

See Also

fscores for factor scoring methods, grmtree fits a Graded Response Model Tree, grmforest for GRM Forests, threshpar_grmtree for extracting threshold parameters, discrpar_grmtree for extracting discrimination parameters, itempar_grmtree for extracting item parameters, generate_node_scores_dataset generates combined dataset with node assignments and factor scores

Examples

```
  library(grmtree)
  library(hlt)
  data("asti", package = "hlt")
  asti$resp <- data.matrix(asti[, 1:4])

  # Fit GRM tree with gender and group as partitioning variables
  tree <- grmtree(resp ~ gender + group,
          data = asti,
          control = grmtree.control(minbucket = 30))

# Compute EAP scores for all terminal nodes
node_scores <- fscores_grmtree(tree)

# Compute MAP scores instead
node_scores_map <- fscores_grmtree(tree, method = "MAP")
```

---

fscores_longitudinal_grmtree

Compute Latent Factor Scores for Longitudinal GRM Tree

Description

Computes latent factor scores (theta) for both T1 and T2 within each terminal node of a longitudinal GRM tree. Unlike the cross-sectional fscores_grmtree which returns a single theta per person, this function returns two scores per person: θ_{T1} and θ_{T2}.

Usage

```
fscores_longitudinal_grmtree(object, method = "EAP", ...)
```

Arguments

object	A longitudinal_grmtree object.
method	Scoring method: "EAP" (default), "MAP", "ML", or "WLE".
...	Additional arguments (currently unused).

Value

A named list (one element per terminal node). Each element is a data.frame with columns:

Theta_T1 Estimated latent trait at Time 1

Theta_T2 Estimated latent trait at Time 2

See Also

longitudinal_grmtree for fitting the tree, generate_node_scores_dataset to generate node assignment and factor scores

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )

  # Phase 1: fit the longitudinal GRM tree
  ltree <- longitudinal_grmtree(
    resp_wide ~ sex + age + residency + job +
      education + comorbidity_count + ever_smoker,
    data = ld, n_items = 8,
    control = grmtree.control(minbucket = 200)
  )

  # Print the factor scores
  scores <- fscores_longitudinal_grmtree(ltree)
  # Scores for node 2
  head(scores[["2"]])
```

`generate_node_scores_dataset` *Generate Dataset with Node Assignments and Factor Scores*

Description

Creates a dataset by augmenting the original data with node assignments and computed factor scores. Unlike the previous version which only returned model frame variables, this version merges node and score information back to the full original data frame.

Usage

```
generate_node_scores_dataset(object, data = NULL, method = "EAP")
```

Arguments

`object`	A `grmtree` or `longitudinal_grmtree` object.
`data`	The original data frame used to fit the tree. If provided, the output contains all columns from this data frame plus node and factor score columns. If NULL (default), returns only model frame variables (backward-compatible behavior).
`method`	Scoring method: "EAP" (default), "MAP", "ML", or "WLE".

Details

The function works by:

1. Predicting node membership for each observation using `predict(object, type = "node")`
2. Computing factor scores within each terminal node using the node-specific model
3. Merging the results back to the original data by row position

When `data` is provided, the function ensures that the output contains all columns from the original data frame, not just the variables used in the model formula. This is important when the original data contains clinical variables, identifiers, or other columns not used as partitioning variables.

Value

A data.frame containing:

All original columns From `data` if provided

node Factor indicating terminal node membership

factor_score For cross-sectional grmtree: single latent score. For longitudinal: Theta_T1.

Theta_T1 (Longitudinal only) Latent trait at T1

Theta_T2 (Longitudinal only) Latent trait at T2

See Also

grmtree fits a Graded Response Model Tree, grmforest for GRM Forests, fscores_grmtree for computing factor scores, threshpar_grmtree for extracting threshold parameters, discrpar_grmtree for extracting discrimination parameters, itempar_grmtree for extracting item parameters, longitudinal_grmtree for longitudinal GRMTree, fscores_longitudinal_grmtree, for computing factor scores for longitudinal GRMTree

Examples

```
# Cross-sectional GRMTree
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

  # Fit GRM tree with gender and group as partitioning variables
  tree <- grmtree(resp ~ gender + group,
          data = asti,
          control = grmtree.control(minbucket = 30))

# Generate combined dataset
scored_data <- generate_node_scores_dataset(tree, data = asti)

# Plot scores by node
boxplot(factor_score ~ node, data = scored_data)
```

---

grmforest *Fit a Forest of Graded Response Model Trees for Ensemble-Based DIF Detection*

Description

This function implements a forest of graded response model trees (GRM Forest) using bootstrap aggregation (bagging) or random subsampling to enhance the detection and analysis of differential item functioning (DIF) in polytomous items. The GRM Forest approach combines the strengths of multiple GRMTrees to provide more robust and stable DIF detection, particularly for complex datasets with high-dimensional covariates or subtle DIF patterns.

Usage

```
grmforest(formula, data, control = grmforest.control(), ...)
```

Arguments

formula	A formula specifying the model structure with the response matrix on the left and partitioning variables on the right (e.g., response_matrix ~ age + gender + education + clinical_variables).

`data`	A data frame containing the response matrix and partitioning variables. The response matrix should contain polytomous items coded as ordered factors.
`control`	A control object created by `grmforest.control()`.
`...`	Additional arguments passed to underlying `grmtree()` function.

Details

The algorithm works by fitting multiple GRMTrees, each on a random sample of the original data (either through bootstrap sampling or subsampling). For each tree, approximately one-third of the observations are left out as out-of-bag (OOB) samples, which are used for internal validation and variable importance calculation. The ensemble approach reduces variance, minimizes overfitting, and provides more reliable identification of covariates associated with DIF.

Key advantages of the GRM Forest approach include:

- Enhanced stability in DIF detection across different sampling variations
- Robust variable importance measures that quantify the relative contribution of each covariate to DIF patterns
- Reduced false positive rates through consensus-based detection
- Ability to handle high-dimensional covariate spaces effectively
- Internal validation through out-of-bag error estimation

The forest implementation supports both bootstrap aggregation (where samples are drawn with replacement) and subsampling (without replacement), allowing flexibility for different data characteristics and research objectives.

Value

An object of class `grmforest` containing:

`trees`	List of fitted GRM trees
`oob_samples`	List of out-of-bag samples for each tree
`formula`	The model formula
`data`	The original dataset
`call`	The function call

See Also

`grmtree` fits a Graded Response Model Tree, `grmtree.control` creates a control object for `grmtree`, `grmforest.control` creates a control object for `grmforest`, `varimp` calculates the variable importance for GRM Forest, `plot.varimp` creates a bar plot of variable importance scores

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
```

```
asti$resp <- data.matrix(asti[, 1:4])

# Fit forest with default parameters
forest <- grmforest(resp ~ gender + group, data = asti)

# Fit with custom control
ctrl <- grmforest.control(n_tree = 20, sampling = "subsample")
forest <- grmforest(resp ~ gender + group, data = asti, control = ctrl)
```

`grmforest.control` *Control Parameters for GRM Forest*

Description

Creates a control object for `grmforest` containing parameters that control the forest growing process including sampling, tree growing, and error handling.

Usage

```
grmforest.control(
  n_tree = 100,
  sampling = "bootstrap",
  sample_fraction = 0.632,
  mtry = NULL,
  remove_dead_trees = TRUE,
  control = grmtree.control(),
  alpha = 0.05,
  minbucket = 20,
  seed = NULL
)
```

Arguments

`n_tree`	Number of trees in the forest (default: 100).
`sampling`	Sampling method: "bootstrap" (with replacement) or "subsample" (without replacement) (default: "bootstrap").
`sample_fraction`	Fraction of data to sample for each tree (default: 0.632).
`mtry`	Number of variables randomly sampled as candidates at each split. If NULL, all variables are considered (default: NULL).
`remove_dead_trees`	Logical indicating whether to remove trees that encounter errors during fitting (default: TRUE).
`control`	Control parameters for individual trees created by `grmtree.control()`.

alpha	Significance level for splitting (default: 0.05).
minbucket	Minimum number of observations in terminal nodes (default: 20).
seed	Random seed for reproducibility (default: NULL).

Value

A list of class grmforest_control containing:

n_tree	Number of trees
sampling	Sampling method
sample_fraction	Sample fraction
mtry	Number of variables to try at each split
remove_dead_trees	Whether to remove failed trees
control	Tree control parameters
seed	Random seed

See Also

grmtree.control creates a control object for grmtree, plot.grmtree creates plot for the grmtree object, grmforest for GRM Forests,

Examples

```
library(grmtree)
# Control with 50 trees using subsampling
ctrl <- grmforest.control(n_tree = 50, sampling = "subsample")

# Control with specific tree parameters
ctrl <- grmforest.control(
  control = grmtree.control(minbucket = 30, alpha = 0.01)
)
```

---

grmtree *Fit a Graded Response Model Tree for Differential Item Functioning Detection*

Description

This function implements a tree-based graded response model (GRM) using model-based recursive partitioning to detect and account for differential item functioning (DIF) in polytomous items. The GRMTree combines the statistical framework of item response theory with recursive partitioning to identify heterogeneous subgroups in the population where item parameters (discrimination and thresholds) vary systematically across covariates.

Usage

```
grmtree(
  formula,
  data,
  na.action = na.omit,
  control = grmtree.control(),
  mtry = NULL,
  ...
)
```

Arguments

formula	A formula specifying the model structure with the response matrix on the left and partitioning variables on the right (e.g., response_matrix ~ age + gender).
data	A data frame containing the variables in the model.
na.action	How to handle missing values (default: na.omit).
control	A list of control parameters created by grmtree.control().
mtry	Number of variables randomly sampled as candidates at each split. If NULL, all variables are considered.
...	Additional arguments passed to the fitting function.

Details

The algorithm works by first estimating a global GRM for the entire sample, then recursively testing for parameter instability with respect to available covariates. When significant DIF is detected, the sample is partitioned into homogeneous subgroups, each with their own set of item parameters. This approach allows for the identification of complex interaction effects and provides interpretable tree structures that visualize how item functioning varies across different patient subgroups.

GRMTree is particularly useful in health outcomes research where patient-reported outcome measures may function differently across diverse demographic, clinical, or socioeconomic subgroups. The resulting tree diagrams facilitate the development of personalized assessment strategies and can inform targeted interventions by identifying specific patient characteristics associated with differential item interpretation.

Conventional Graded Response Model (GRM):

Let Y_{im} denote the response of the i^{th} $(i = 1, \ldots, N)$ individual to the m^{th} $(m = 1, 2, \ldots, M)$ item. The graded response model is described as:

$$P(Y_{im} \geq j|\tau_{mj}, \lambda_m, \theta_i) = \frac{\exp(-(\tau_{mj} - \lambda_m\theta_i))}{1 + \exp(-(\tau_{mj} - \lambda_m\theta_i))}$$

where:

- $P(Y_{im} \geq j|\tau_{mj}, \lambda_m, \theta_i)$ is the probability that individual i's response is in category j or higher on item m,
- τ_{mj} is the threshold parameter between categories $j - 1$ and j for item m,
- λ_m is the discrimination parameter for item m,
- $\theta_i \sim N(0, 1)$ is the latent trait score for individual i.

This parametrization is equivalent to the conventional IRT formulation where item discrimination is $a_m = \lambda_m$ and item difficulty is $b_{mj} = \tau_{mj}/\lambda_m$.

Graded Response Model Tree (GRMTree) Implementation:

The GRMTree is a hybrid model that integrates the GRM with model-based recursive partitioning to detect and account for differential item functioning (DIF) across subgroups defined by covariates. The algorithm proceeds through the following steps:

Step 1: Global Model Estimation

Estimate the GRM item parameters $(\hat{\tau}_{mj}, \hat{\lambda}_m)$ jointly for all individuals in the study cohort at the root node via maximum likelihood estimation:

$$\hat{\beta}_{\text{global}} = \arg\max_{\beta} \sum_{i=1}^{N} \log L(\beta; \mathbf{y}_i)$$

where $\beta = (a_1, \ldots, a_J, b_{11}, \ldots, b_{J,m-1})$ contains all item parameters (discrimination and difficulty), providing a baseline model assuming parameter invariance.

Step 2: Parameter Stability Testing

For each available covariate X_p ($p = 1, \ldots, P$), assess the stability of the item parameters by conducting score-based structural change tests. This involves: 1. Calculating the score function contributions $s(\hat{\beta}; y_i, x_i)$ for each individual, 2. Ordering these scores with respect to each covariate X_p, 3. Testing the null hypothesis $H_0 : \mathbb{E}[s(\hat{\beta}; y_i, x_i)] = 0$ for all i against the alternative that scores fluctuate systematically with X_p, indicating parameter instability (DIF).

Step 3: Recursive Partitioning

If significant instability is detected ($p < \alpha_{\text{adj}}$):

- **Covariate Selection:** Identify the covariate X_p^* with the most significant instability (smallest adjusted p-value),
- **Split Point Determination:** Find the optimal cut-point c^* that maximizes the partitioned log-likelihood:

$$\ell_{\text{left}}(\beta) + \ell_{\text{right}}(\beta) = \sum_{i: X_{pi} \leq c^*} \log L(\beta; y_i) + \sum_{i: X_{pi} > c^*} \log L(\beta; y_i)$$

  over all possible cut-points on X_p^*,
- **Sample Splitting:** Partition the sample into two child nodes based on the rule $X_p^* \leq c^*$.

Step 4: Recursive Application & Stopping Criteria

Repeat Steps 1-3 recursively within each resulting child node until one of the following stopping criteria is met:

1. **No Significant Instability:** No covariate shows significant parameter instability after multiple testing correction ($\alpha_{\text{adj}} = \alpha/m$, where multiple adjustment methods can be applied, including Bonferroni, Holm, Benjamini-Hochberg, etc.).
2. **Minimum Node Size:** The subsample size falls below a prespecified minimum (e.g., $n <$ $10\times$ the number of item parameters).

Formal GRMTree Structure:

The final GRMTree provides a piecewise GRM where each terminal node represents a subgroup with homogeneous item parameters, explicitly modeling the detected DIF structure within the data. The resulting GRMTree model can be expressed as a mixture of subgroup-specific GRMs:

$$P(Y_{ij} = k|\theta_i, \mathbf{x}_i) = \sum_{b=1}^{B} I(\mathbf{x}_i \in \mathcal{X}_b) \cdot P_b(Y_{ij} = k|\theta_i)$$

where:

- B is the number of terminal nodes,
- $\mathcal{X}_b$ is the covariate subspace defining terminal node b,
- P_b is the node-specific GRM with parameters β_b,
- $I(\cdot)$ is the indicator function.

Each terminal node b contains a complete GRM with:

- **Node-specific item parameters:** $\beta_b = (a_{1b}, \ldots, a_{Jb}, b_{11b}, \ldots, b_{J,m-1,b})$
- **Local ability distribution:** $\theta_i|\mathbf{x}_i \in \mathcal{X}_b \sim N(0, 1)$

This approach allows **differential item functioning (DIF)** to be detected and modeled explicitly through the tree structure, where item parameters can vary across subgroups defined by covariates, while maintaining the conditional distribution of the latent trait within each subgroup.

Post-Hoc Multiple Comparison Adjustments:

For holm, BH, BY, hochberg, and hommel methods, the algorithm employs a two-stage approach: (1) build a tree using `initial_alpha` as the splitting threshold, (2) apply global p-value adjustment across all splits, and (3) prune splits that do not meet the adjusted threshold `alpha`. This differs from Bonferroni correction, which is applied locally during tree construction via `partykit::mob`. The choice of `initial_alpha` represents a statistical trade-off. The default (`min(3 * alpha, 0.20)`) provides a balance between power and computational efficiency. Note that global post-hoc adjustments in hierarchical tree structures may be conservative compared to per-node adjustments, as they account for all tests performed during tree exploration. This global adjustment approach controls the tree-wide error rate and may be conservative (Type I error often <3%). This conservativeness ensures strong control of family-wise error rate or false discovery rate across all splits in the tree. Users requiring less conservative control may prefer the Bonferroni method, which applies per-node adjustment during tree construction.

Value

An object of class `grmtree` inheriting from `modelparty` containing the fitted tree structure.

Author(s)

Olayinka Imisioluwa Arimoro <olayinka.arimoro@ucalgary.ca>, Lisa M. Lix, Tolulope T. Sajobi

References

Methodological Foundations:

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. Psychometrika Monograph Supplement, 34, 100-114.

Strobl, C., Kopf, J., & Zeileis, A. (2015). Rasch trees: A new method for detecting differential item functioning in the Rasch model. Psychometrika, 80(2), 289-316.

Komboz, B., Strobl, C., & Zeileis, A. (2018). Tree-based global model tests for polytomous Rasch models. Educational and psychological measurement, 78(1), 128-166. https://doi.org/10.1177/0013164416664394

Arimoro, O. I., Lix, L. M., Patten, S. B., Sawatzky, R., Sebille, V., Liu, J., Wiebe, S., Josephson, C. B., & Sajobi, T. T. (2025). Tree-based latent variable model for assessing differential item functioning in patient-reported outcome measures: a simulation study. Quality of Life Research. https://doi.org/10.1007/s11136-025-04018-6

Applied Examples:

Arimoro, O. I., Josephson, C. B., James, M. T., Patten, S. B., Wiebe, S., Lix, L. M., & Sajobi, T. T. (2024). Screening for depression in patients with epilepsy: same questions but different meaning to different patients. Quality of Life Research, 33(12), 3409-3419. https://doi.org/10.1007/s11136-024-03782-1

See Also

`print.grmtree` prints the detailed summary results of the `grmtree` object, `grmtree.control` creates a control object for `grmtree`, `plot.grmtree` creates plot for the `grmtree` object, `grmforest` for GRM Forests, `varimp` calculates the variable importance for GRM Forest, `fscores_grmtree` for computing factor scores, `threshpar_grmtree` for extracting threshold parameters, `discrpar_grmtree` for extracting discrimination parameters, `itempar_grmtree` for extracting item parameters

Examples

```
library(grmtree)
library(hlt)

# Prepare the asti data (from the hlt package)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit GRM tree with gender and group as partitioning variables
tree <- grmtree(resp ~ gender + group,
        data = asti,
        control = grmtree.control(minbucket = 30))

## Print and plot the tree
print(tree)
plot(tree)

# Extract item parameters for specific subgroups
discr_params <- discrpar_grmtree(tree)
threshold_params <- threshpar_grmtree(tree)
```

`grmtree.control` *Control Parameters for GRM Trees*

Description

Creates a control object for `grmtree` containing various parameters that control the tree growing process.

Usage

```
grmtree.control(
  minbucket = 20,
  p_adjust = "none",
  alpha = 0.05,
  initial_alpha = NULL,
  ...
)
```

Arguments

minbucket	Minimum number of observations in a terminal node (default: 20).
p_adjust	Method for p-value adjustment. One of: "none", "bonferroni", "holm", "BH", "BY", "hochberg", or "hommel" (default: "none").
alpha	Significance level for splitting (default: 0.05).
initial_alpha	For post-hoc adjustment methods (holm, BH, BY, hochberg, hommel), the significance threshold for initial tree construction before pruning. Must satisfy alpha < initial_alpha < 1. Default is min(3 * alpha, 0.20). Lower values produce more conservative results but run faster; higher values provide more power but require more computation and may increase Type I error. Ignored for "none" and "bonferroni" methods.
...	Additional arguments passed to partykit::mob_control().

Value

A list of control parameters with class grmtree_control.

See Also

grmtree fits a Graded Response Model Tree

Examples

```
# Use Bonferroni correction with alpha = 0.01
ctrl <- grmtree.control(p_adjust = "bonferroni", alpha = 0.01)
```

---

grmtree_data *Medical Outcomes Study Social Support Survey (MOS-SS) Test Data*

Description

A dataset containing sample responses to the MOS-SS emotional domain items and demographic variables. This data is provided for testing and demonstration purposes within the grmtree package. The items are numbered 1-5, representing None of the time, A little of the time, Some of the time, Most of the time, All of the time, respectively.

Usage

```
grmtree_data
```

Format

A tibble with 3,500 rows and 17 variables:

MOS_Listen Someone you can count on to listen to you when you need to talk (1-5 Likert scale)

MOS_Info Someone to give you information to help you understand a situation (1-5 Likert scale)

MOS_Advice_Crisis Someone to give good advice about a crisis (1-5 Likert scale)

MOS_Confide Someone to confide in or talk to about yourself or your problems (1-5 Likert scale)

MOS_Advice_Want Someone whose advice you really want (1-5 Likert scale)

MOS_Fears Someone to share private worries or fears (1-5 Likert scale)

MOS_Personal Someone to turn to for suggestions about how to deal with a personal problem (1-5 Likert scale)

MOS_Understand Someone who understands your problems (1-5 Likert scale)

sex Gender (Male, Female)

age Age in years (numeric)

residency Residence location (rural, urban)

depressed Depression status (No, Yes)

bmi Body Mass Index (numeric)

Education Education level (Primary/High school, College/University)

job Employment status (Employed, Unemployed)

smoker Smoking status (No, Yes)

multimorbidity Number of chronic conditions (0, 1, 2+)

Source

Simulated data generated for package testing and demonstration

Examples

```
library(dplyr)

# Load and take a glimpse at the data
data(grmtree_data, package = "grmtree")
glimpse(grmtree_data)
```

grmtree_long_data *Synthetic Longitudinal MOS-SS Social Support Survey Data*

Description

A synthetic dataset containing responses to the eight MOS-SS emotional domain items measured at two time points (baseline and one-year follow-up), together with baseline demographic and clinical covariates. The data are provided for testing and demonstration of the longitudinal GRMTree functions (`longitudinal_grmtree`, `rs_characterize`) within the grmtree package.

Usage

```
grmtree_long_data
```

Format

A tibble with 1,500 rows and 23 variables:

MOS_Listen Baseline: someone you can count on to listen when you need to talk (1–5)

MOS_Info Baseline: someone to give you information to help you understand a situation (1–5)

MOS_Advice_Crisis Baseline: someone to give good advice in a crisis (1–5)

MOS_Confide Baseline: someone to confide in or talk to about yourself or your problems (1–5)

MOS_Advice_Want Baseline: someone whose advice you really want (1–5)

MOS_Fears Baseline: someone to share your private worries or fears (1–5)

MOS_Personal Baseline: someone to turn to for suggestions about a personal problem (1–5)

MOS_Understand Baseline: someone who understands your problems (1–5)

MOS_Listen_year1 One-year follow-up: listen item (1–5)

MOS_Info_year1 One-year follow-up: information item (1–5)

MOS_Advice_Crisis_year1 One-year follow-up: crisis advice item (1–5)

MOS_Confide_year1 One-year follow-up: confide item (1–5)

MOS_Advice_Want_year1 One-year follow-up: advice wanted item (1–5)

MOS_Fears_year1 One-year follow-up: fears item (1–5)

MOS_Personal_year1 One-year follow-up: personal suggestions item (1–5)

MOS_Understand_year1 One-year follow-up: understand item (1–5)

sex Sex (Male, Female)

age Age in years (numeric)

residency Residence location (Urban, Rural)

job Employment status (Employed, Unemployed)

education Education level (Primary/High school, College/University)

ever_smoker Ever-smoker status (Yes, No)

comorbidity_count Number of chronic conditions (0–5)

bmi Body mass index (numeric)

Details

The data were simulated from a two-factor longitudinal graded response model. Response shift was intentionally built into the older subgroup (age > 61): the `MOS_Info` item exhibits reprioritization (a discrimination change over time) and the `MOS_Fears` item exhibits recalibration (a threshold change over time). Younger patients show no response shift. As a result, `longitudinal_grmtree` produces an age split, and `rs_characterize` detects response shift within the older subgroup. Items are coded 1–5, representing None of the time, A little of the time, Some of the time, Most of the time, and All of the time, respectively.

Source

Synthetic data generated from a two-factor longitudinal graded response model for package testing and demonstration. Contains no real patient data.

Examples

```
library(grmtree)
library(dplyr)

  # Load the synthetic longitudinal data and take a glimpse
  data("grmtree_long_data", package = "grmtree")
  glimpse(grmtree_long_data)
```

`itempar_grmtree` *Extract Item Parameters from GRM Tree*

Description

Extracts both discrimination parameters and average threshold parameters for each item from all terminal nodes of a graded response model tree. This provides a comprehensive view of item characteristics across different nodes of the tree.

Usage

```
itempar_grmtree(object, node = NULL, ...)
```

Arguments

`object`	A `grmtree` object.
`node`	Optional vector of node IDs to extract from. If NULL (default), extracts from all terminal nodes.
`...`	Additional arguments (currently unused).

Value

A data.frame with item parameters for each item in each node, with columns:

`Node`	Node ID
`Item`	Item name
`Discrimination`	Discrimination parameter (a1)
`AvgThreshold`	Average of threshold parameters
`Thresholds`	All threshold parameters as a list column

See Also

`grmtree` fits a Graded Response Model Tree, `grmforest` for GRM Forests, `fscores_grmtree` for computing factor scores, `threshpar_grmtree` for extracting threshold parameters, `discrpar_grmtree` for extracting discrimination parameters

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit GRM tree with gender and group as partitioning variables
tree <- grmtree(resp ~ gender + group,
        data = asti,
        control = grmtree.control(minbucket = 30))

# Get all item parameters
items <- itempar_grmtree(tree)
print(items)
```

---

itempar_longitudinal_grmtree

Extract Item Parameters from Longitudinal GRM Tree

Description

Extracts both discrimination and threshold parameters for each unique item from all terminal nodes, combining them into a single data frame with an average threshold column. Only T1 items are returned.

Usage

```
itempar_longitudinal_grmtree(object, node = NULL, clean_names = TRUE, ...)
```

Arguments

object	A longitudinal_grmtree object.
node	Optional vector of node IDs. If NULL, all terminal nodes.
clean_names	Logical. If TRUE (default), clean item names.
...	Additional arguments (currently unused).

Value

A data.frame with columns:

Node Terminal node ID

Item Item name

Discrimination Discrimination parameter

AvgThreshold Mean of all threshold parameters for the item

b1, b2, ..., bK Individual threshold parameters

See Also

longitudinal_grmtree for Phase 1 (tree fitting), discrpar_longitudinal_grmtree for extracting discrimination parameters for longitudinal GRMTree, threshpar_longitudinal_grmtree for extracting threshold parameters for longitudinal GRMTree

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )

  # Phase 1: fit the longitudinal GRM tree
  ltree <- longitudinal_grmtree(
    resp_wide ~ sex + age + residency + job +
      education + comorbidity_count + ever_smoker,
    data = ld, n_items = 8,
    control = grmtree.control(minbucket = 200)
  )

  # Print the item parameters
```

```
items <- itempar_longitudinal_grmtree(ltree)
print(items)
```

`latentpar_longitudinal_grmtree` *Extract Latent Trait Parameters from Longitudinal GRM Tree*

Description

Extracts the latent trait distribution parameters from each terminal node of a longitudinal GRM tree: the T2 latent mean shift (μ_{T2}), the T2 latent variance (σ^2_{T2}), and the T1-T2 correlation.

Usage

```
latentpar_longitudinal_grmtree(object, node = NULL, ...)
```

Arguments

`object`	A `longitudinal_grmtree` object.
`node`	Optional vector of node IDs. If NULL, all terminal nodes.
`...`	Additional arguments (currently unused).

Value

A data.frame with columns:

Node Terminal node ID

n Sample size in the node

mu_T2 Latent mean at T2 (positive = improvement)

sigma2_T2 Latent variance at T2

cor_T1_T2 Correlation between T1 and T2 latent traits

See Also

`longitudinal_grmtree` for Phase 1 (tree fitting), `fscores_longitudinal_grmtree`, for computing factor scores for longitudinal GRMTree

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
```

```
              "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
ld <- prepare_longitudinal_data(
  data = grmtree_long_data,
  items_t1 = items_t1,
  items_t2 = paste0(items_t1, "_year1"),
  covariates = c("sex", "age", "residency", "job",
                 "education", "comorbidity_count", "ever_smoker")
)

# Phase 1: fit the longitudinal GRM tree
ltree <- longitudinal_grmtree(
  resp_wide ~ sex + age + residency + job +
    education + comorbidity_count + ever_smoker,
  data = ld, n_items = 8,
  control = grmtree.control(minbucket = 200)
)

# Print latent trait summary
latent <- latentpar_longitudinal_grmtree(ltree)
print(latent)
```

`longitudinal_grmtree` *Fit a Longitudinal Graded Response Model Tree for Response Shift Detection*

Description

This function implements a tree-based longitudinal graded response model using model-based recursive partitioning (MOB) to detect measurement heterogeneity in patient-reported outcome measures (PROMs) measured at two time points. The Longitudinal GRMTree extends the cross-sectional `grmtree` to the longitudinal setting by embedding a constrained two-factor GRM within the MOB framework. The resulting tree identifies patient subgroups where the longitudinal measurement model differs, providing the foundation for subgroup-specific response shift characterization via `rs_characterize`.

Usage

```
longitudinal_grmtree(
  formula,
  data,
  n_items = NULL,
  na.action = na.omit,
  control = grmtree.control(),
  mtry = NULL,
  ...
)
```

Arguments

formula	A formula specifying the model structure with a wide-format response matrix on the left-hand side and partitioning variables on the right-hand side (e.g., resp_wide ~ age + sex + comorbidity_count). The response matrix must have 2 * n_items columns: the first n_items columns are Time 1 (T1) responses and the next n_items columns are Time 2 (T2) responses for the same items.
data	A data frame containing the variables in the formula. Must include the response matrix as a matrix-valued column (created via data$resp_wide <- as.matrix(...)) and all partitioning variables as separate columns.
n_items	Integer specifying the number of items per time point. If NULL (default), automatically detected as half the number of columns in the response matrix. Must satisfy ncol(response) == 2 * n_items.
na.action	How to handle missing values. Default is na.omit, which removes rows with any missing values in the response matrix or partitioning variables.
control	A list of control parameters created by grmtree.control. Key parameters include:
	minbucket Minimum number of observations in a terminal node. Should be at least 10 times the number of free parameters per node to ensure stable estimation. For 8 items with 5 categories each, the constrained longitudinal GRM has approximately 40 item parameters plus 3 latent parameters, suggesting minbucket >= 200.
	alpha Significance level for the parameter instability tests. Default is 0.05.
	p_adjust Method for adjusting p-values across covariates at each split. Options include "bonferroni" (default, applied locally during tree construction), "BH" (Benjamini-Hochberg, applied post-hoc with pruning), and others. See grmtree.control for details.
mtry	Number of variables randomly sampled as candidates at each split. If NULL (default), all variables are considered. Can be used for random forest extensions.
...	Additional arguments passed to the internal fitting function longitudinal_grmfit and ultimately to mirt.

Details

Overview:

The Longitudinal GRMTree is a unified framework for response shift detection that operates in two phases:

Phase 1 (this function): Identifies patient subgroups where the constrained longitudinal measurement model differs. The constrained model represents the null hypothesis of no response shift (item parameters equal across time). MOB tests whether this null model's parameters are stable across patient covariates and recursively partitions the sample where instability is detected.

Phase 2 (rs_characterize): Within each terminal node, tests whether item parameters actually change from T1 to T2 by comparing the constrained model to an unconstrained model using likelihood ratio tests. This phase characterizes response shift at the item level, classifying changes as recalibration, reprioritization, or both.

Constrained Two-Factor Longitudinal GRM:

Let Y_{im} denote the response of individual i to item m at time t. The longitudinal GRM specifies two correlated latent factors:

$$\theta_{i,T1} \sim N(0, 1)$$
$$\theta_{i,T2} \sim N(\mu_{T2}, \sigma^2_{T2})$$
$$Cov(\theta_{i,T1}, \theta_{i,T2}) = \sigma_{12}$$

For item m at time t, the graded response model is:

$$P(Y_{im,t} \geq j|\theta_{i,t}) = \frac{\exp(a_{m,t}(\theta_{i,t} - b_{mj,t}))}{1 + \exp(a_{m,t}(\theta_{i,t} - b_{mj,t}))}$$

where $a_{m,t}$ is the discrimination parameter and $b_{mj,t}$ are threshold parameters for item m at time t.

Under the no-response-shift constraint:

$$a_{m,T1} = a_{m,T2} \quad \text{and} \quad b_{mj,T1} = b_{mj,T2} \quad \forall m, j$$

This means item parameters are identical across time, so any observed changes in responses are attributed to true latent change (μ_{T2}) rather than changes in measurement properties.

The Longitudinal GRMTree Algorithm:

Step 1: Global Model Estimation. Fit the constrained longitudinal GRM to all individuals at the root node, estimating item parameters $\hat{\beta}$ and latent parameters $(\mu_{T2}, \sigma^2_{T2}, \sigma_{12})$ via maximum likelihood.

Step 2: Parameter Stability Testing. For each covariate X_p, compute individual score function contributions $s(\hat{\beta}; \mathbf{y}_i)$ and test whether these scores fluctuate systematically with X_p using structural change tests. The null hypothesis is that model parameters are stable across all values of X_p.

Step 3: Recursive Partitioning. If significant instability is detected (after p-value adjustment):

- Select the covariate X^*_p with strongest instability
- Find the optimal split point c^* maximizing the partitioned log-likelihood
- Split the sample: $X^*_p \leq c^*$ vs. $X^*_p > c^*$

Step 4: Recursion. Repeat Steps 1–3 within each child node until no significant instability remains or the minimum node size is reached.

Step 5 (Post-hoc): Apply `rs_characterize` to test for response shift within each terminal node.

Formal Model Structure:

The fitted Longitudinal GRMTree provides a piecewise constrained longitudinal GRM:

$$P(Y_{im,t} = k|\theta_{i,t}, \mathbf{x}_i) = \sum_{b=1}^{B} I(\mathbf{x}_i \in \mathcal{X}_b) \cdot P_b(Y_{im,t} = k|\theta_{i,t})$$

where B is the number of terminal nodes, $\mathcal{X}_b$ is the covariate subspace defining terminal node b, and P_b is the node-specific constrained longitudinal GRM. Each terminal node contains:

- Node-specific constrained item parameters (equal across T1 and T2)
- Node-specific latent trait parameters: $\mu_{T2,b}$, $\sigma^2_{T2,b}$, $r_{T1,T2,b}$

Interpretation:

The tree structure identifies patient subgroups where the constrained longitudinal measurement model differs. This can reflect:

- Different item discrimination patterns across subgroups
- Different threshold locations across subgroups
- Different magnitudes of true latent change (μ_{T2})
- Different test-retest correlations

Crucially, the tree does **not** directly detect response shift (temporal change in item parameters within a subgroup). Response shift is tested in Phase 2 using `rs_characterize`, which relaxes the equality constraints within each terminal node. A tree with only one terminal node (no split) indicates that no covariate moderates the measurement model; it does **not** imply the absence of response shift, since `rs_characterize` can still detect uniform RS in the unsplit root node.

Relationship to Existing Methods:

The Longitudinal GRMTree extends several existing approaches:

- **GRMTree** (Arimoro et al., 2026): Cross-sectional DIF detection using tree-based GRM. The longitudinal extension embeds a two-factor model instead of a single-factor model.
- **LIRTree** (Ames & Leventhal, 2021): Longitudinal tree-based IRT using Rasch/2PL models. The GRM extension allows item-specific discrimination parameters, which are important for PROMs where items vary in discriminating ability.
- **Oort SEM** (Oort, 2005): Response shift detection via structural equation modeling. The tree-based approach removes the requirement for pre-specified subgroups.

Practical Recommendations:

- **Minimum node size:** Use at least 10–25 times the number of free parameters per node. For M items with K response categories, the constrained model has approximately $M(K-1)+M+3$ parameters (M discriminations, $M(K-1)$ thresholds, and 3 latent parameters). For 8 items with 5 categories: $8\times4+8+3=43$ parameters, suggesting `minbucket = 200--400`.
- **Number of covariates:** Bonferroni correction becomes increasingly conservative with more covariates. Consider using `p_adjust = "BH"` for exploratory analyses with many covariates.
- **Sample size:** With typical PROM instruments (5–15 items), a total sample of at least 500–1000 is recommended for adequate power to detect meaningful splits.
- **Response matrix preparation:** Use `prepare_longitudinal_data` to construct the wide-format response matrix from separate T1 and T2 item columns.

Value

An object of class `c("longitudinal_grmtree", "grmtree", "modelparty", "party")` containing the fitted tree structure. The object inherits from `modelparty` and includes:

Tree structure Accessible via standard `partykit` methods such as `nodeids`, `data_party`, and indexing with `[[`.

Node models Each terminal node contains a fitted constrained longitudinal GRM with item parameters (discrimination and thresholds), latent trait means (μ_{T2}), and latent trait covariance matrix (σ^2_{T2}, $r_{T1,T2}$).

`info$n_items` Number of items per time point.

`info$model_type` `"longitudinal_grm"`.

`info$p_adjust` The p-value adjustment method used.

`info$call` The original function call.

Author(s)

Olayinka Imisioluwa Arimoro <olayinka.arimoro@ucalgary.ca>, Lisa M. Lix, Tolulope T. Sajobi

References

Arimoro, O. I., Lix, L. M., Patten, S. B., Sawatzky, R., Sebille, V., Liu, J., Wiebe, S., Josephson, C. B., & Sajobi, T. T. (2025). Tree-based latent variable model for assessing differential item functioning in patient-reported outcome measures: a simulation study. *Quality of Life Research*. doi:10.1007/s11136025040186

Ames, A. J., & Leventhal, B. C. (2021). Application of a longitudinal IRTree model: response style changes over time. *Educational and Psychological Measurement*, 81(3), 561–582.

Oort, F. J. (2005). Using structural equation modeling to detect response shifts and true change. *Quality of Life Research*, 14(3), 587–598.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monograph Supplement*, 34, 100–114.

Sprangers, M. A., & Schwartz, C. E. (1999). Integrating response shift into health-related quality of life research. *Social Science & Medicine*, 48(11), 1507–1515.

Zeileis, A., Hothorn, T., & Hornik, K. (2008). Model-based recursive partitioning. *Journal of Computational and Graphical Statistics*, 17(2), 492–514.

See Also

`rs_characterize` for Phase 2 response shift characterization, `prepare_longitudinal_data` for data preparation, `grmtree` for cross-sectional DIF detection, `grmtree.control` for control parameters, `mob` for the underlying MOB framework

Examples

```
library(grmtree)

  # Load the synthetic longitudinal data
  data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
```

```
  items_t1 = items_t1,
  items_t2 = paste0(items_t1, "_year1"),
  covariates = c("sex", "age", "residency", "job",
                 "education", "comorbidity_count", "ever_smoker")
)

# Phase 1: fit the longitudinal GRM tree
ltree <- longitudinal_grmtree(
  resp_wide ~ sex + age + residency + job +
    education + comorbidity_count + ever_smoker,
  data = ld, n_items = 8,
  control = grmtree.control(minbucket = 200)
)

# Print tree structure
print(ltree)

# Plot threshold regions
plot(ltree, type = "regions", tnex = 2L)

# Phase 2: characterize response shift within each subgroup
rs <- rs_characterize(ltree, p_adjust = "fdr",
  global_p_adjust = "bonferroni")
print(rs)
```

`plot.grmtree` *Plot Method for GRM Tree Objects*

Description

Visualizes a GRM (Graded Response Model) tree with different types of terminal node plots. This function extends `plot.modelparty` from the partykit package with specialized visualizations for GRM trees.

Usage

```
## S3 method for class 'grmtree'
plot(
  x,
  type = c("regions", "profile", "histogram"),
  what = c("item", "threshold", "discrimination"),
  tnex = 2L,
  drop_terminal = TRUE,
  spacing = 0.1,
  ...
)
```

Arguments

x	A GRM tree object of class 'grmtree'.
type	Type of terminal node plot to display: **"regions"** Threshold regions plot (default) **"profile"** Item parameter profile plot **"histogram"** Histogram of factor scores with normal curve
what	Type of parameters to plot when type = "profile": **"item"** Both discrimination and threshold parameters (default) **"threshold"** Only threshold parameters **"discrimination"** Only discrimination parameters
tnex	Numeric scaling factor for terminal node extension (default: 2).
drop_terminal	Logical indicating whether to drop terminal node IDs (default: TRUE).
spacing	Numeric value controlling spacing between elements (default: 0.1).
...	Additional arguments passed to the terminal panel functions.

Details

The function provides three visualization types:

- **Regions plot**: Shows threshold parameters as colored regions, useful for visualizing the difficulty parameters across items and nodes.
- **Profile plot**: Displays either item parameters (discrimination and average thresholds), just thresholds, or just discrimination parameters as line plots across items.
- **Histogram**: Shows the distribution of factor scores in each node with an overlaid normal curve.

Value

Invisibly returns the GRM tree object. Primarily called for its side effect of producing a plot.

See Also

plot.modelparty for the underlying plotting infrastructure, grmtree for creating GRM tree objects, plot.varimp creates a bar plot of variable importance scores

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit GRM tree with gender and group as partitioning variables
tree <- grmtree(resp ~ gender + group,
          data = asti,
```

```
          control = grmtree.control(minbucket = 30))

# Default regions plot
plot(tree)

# Profile plot showing item parameters
plot(tree, type = "profile")

# Profile plot showing only thresholds
plot(tree, type = "profile", what = "threshold")

# Histograms of factor scores
plot(tree, type = "histogram")
```

`plot.longitudinal_grmtree` *Plot Method for Longitudinal GRM Tree Objects*

Description

Visualizes a longitudinal GRM tree with threshold region plots in terminal nodes. Unlike the cross-sectional `plot.grmtree`, this method displays only the unique item parameters (T1 items), since the constrained longitudinal model enforces equal parameters across T1 and T2 within each node.

Usage

```
## S3 method for class 'longitudinal_grmtree'
plot(
  x,
  type = "regions",
  tnex = 2L,
  drop_terminal = TRUE,
  names = FALSE,
  abbreviate = TRUE,
  ...
)
```

Arguments

`x`	A `longitudinal_grmtree` object.
`type`	Type of terminal node plot. Currently only `"regions"` is supported for the longitudinal model.
`tnex`	Numeric scaling factor for terminal node extension (default: 2).
`drop_terminal`	Logical indicating whether to drop terminal node IDs (default: TRUE).

names	Logical or character vector. If TRUE, use item names from the response matrix. If a character vector, use as custom labels. If FALSE (default), use numeric indices 1 through n_items.
abbreviate	Logical or numeric. If TRUE, abbreviate item names. If numeric, abbreviate to that many characters.
...	Additional arguments passed to the terminal panel function.

Details

The region plot displays threshold parameters as colored horizontal bands for each item within each terminal node. Darker shading represents lower response categories and lighter shading represents higher categories. The height of each band corresponds to the range of the latent trait over which that response category is most likely.

Because the constrained longitudinal GRM enforces $a_{m,T1} = a_{m,T2}$ and $b_{k,m,T1} = b_{k,m,T2}$, the T1 and T2 item parameters are identical. The plot therefore shows only the `n_items` unique items rather than all `2 * n_items` columns in the response matrix.

Value

Invisibly returns the tree object. Called for its side effect of producing a plot.

See Also

longitudinal_grmtree for fitting the tree, plot.grmtree for cross-sectional tree plots

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )

  # Phase 1: fit the longitudinal GRM tree
  ltree <- longitudinal_grmtree(
    resp_wide ~ sex + age + residency + job +
      education + comorbidity_count + ever_smoker,
    data = ld, n_items = 8,
    control = grmtree.control(minbucket = 200)
  )
```

```
# Region plot with numeric labels
plot(ltree)

# Region plot with item names
plot(ltree, names = TRUE)

# Custom labels
plot(ltree, names = c("Listen", "Info", "Crisis",
  "Confide", "Advice", "Fears", "Personal", "Understand"))
```

`plot.varimp` *Plot Variable Importance*

Description

Creates a bar plot of variable importance scores with options for both ggplot2 and base R graphics.

Usage

```
## S3 method for class 'varimp'
plot(x, top_n = NULL, use_ggplot = TRUE, ...)
```

Arguments

`x`	A `varimp` object from `varimp()`.
`top_n`	Number of top variables to display (NULL for all).
`use_ggplot`	Logical indicating whether to use ggplot2 (if available).
`...`	Additional arguments passed to plotting functions.

Value

Invisibly returns the input object.

See Also

`varimp` calculates the variable importance for GRM Forest, `grmforest` for GRM Forests, `grmforest.control` creates a control object for `grmforest`, `plot.grmtree` creates plot for the `grmtree` object

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit forest with default parameters
```

```
forest <- grmforest(resp ~ gender + group, data = asti)
imp <- varimp(forest)
plot(imp)
plot(imp, top_n = 1) ## select top 1 importance variable
plot(imp, use_ggplot = FALSE) # Use base R graphics
```

`plot_rs_heatmap` *Plot Item-Level Response Shift Heatmap*

Description

Creates a standalone heatmap showing RS type for each item across all terminal nodes. Significant items are filled with RS-type colors; non-significant items are white/light gray.

Usage

```
plot_rs_heatmap(
  rs,
  item_labels = NULL,
  node_labels = NULL,
  show_chi2 = FALSE,
  title = NULL,
  ...
)
```

Arguments

`rs`	An `rs_characterization` object from `rs_characterize`.
`item_labels`	Optional character vector of item labels. If NULL, uses "Item 1", "Item 2", etc.
`node_labels`	Optional character vector of node labels. If NULL, uses "Node X (n=Y)" from the global results.
`show_chi2`	Logical. If TRUE, display chi-squared values inside cells. Default is FALSE.
`title`	Optional plot title. If NULL, uses a default title.
`...`	Additional arguments (unused).

Details

The heatmap uses the following color coding:

Blue Recalibration (threshold change)

Red/Orange Reprioritization (discrimination change)

Purple Both recalibration and reprioritization

Light orange Significant but small effect

White Not significant or not tested

Cells with significant RS (after p-value adjustment) are marked with an asterisk (*). The omnibus RS result for each node is displayed at the top of each column.

Value

Invisibly returns the rs object. Called for its side effect of producing a plot.

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )

  # Phase 1: fit the longitudinal GRM tree
  ltree <- longitudinal_grmtree(
    resp_wide ~ sex + age + residency + job +
      education + comorbidity_count + ever_smoker,
    data = ld, n_items = 8,
    control = grmtree.control(minbucket = 200)
  )

  # Phase 2: characterize response shift within each subgroup
  rs <- rs_characterize(ltree, p_adjust = "fdr",
    global_p_adjust = "bonferroni")

  # Basic heatmap
  plot_rs_heatmap(rs)

  # With custom labels
  plot_rs_heatmap(rs,
    item_labels = c("Listen", "Info", "Crisis", "Confide",
      "Advice", "Fears", "Personal", "Understand"))

  # Show chi-squared values
  plot_rs_heatmap(rs, show_chi2 = TRUE)
```

`plot_rs_tree` *Plot Response Shift Summary Tree*

Description

Displays the Longitudinal GRMTree structure with RS characterization results annotated in each terminal node panel. Each panel shows the omnibus test result, latent trait parameters, and an item-level RS heatmap color-coded by RS type.

Usage

```
plot_rs_tree(
  tree,
  rs,
  item_labels = NULL,
  tnex = 2.5,
  drop_terminal = TRUE,
  ...
)
```

Arguments

tree	A longitudinal_grmtree object.
rs	An rs_characterization object from rs_characterize.
item_labels	Optional character vector of short item labels. If NULL, uses "Item 1", "Item 2", etc.
tnex	Numeric scaling factor for terminal node panels (default: 2.5).
drop_terminal	Logical (default: TRUE).
...	Additional arguments passed to plot.modelparty.

Details

Each terminal node panel contains:

- Omnibus RS result: chi-squared, adjusted p-value, and detection status
- Latent parameters: mu_T2 and cor(T1,T2)
- Item-level RS bar: colored cells for each item indicating RS type (blue = recalibration, red = reprioritization, purple = both, orange = significant small effect, gray = none/not tested)

Value

Invisibly returns the tree object.

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
```

```
              "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
ld <- prepare_longitudinal_data(
  data = grmtree_long_data,
  items_t1 = items_t1,
  items_t2 = paste0(items_t1, "_year1"),
  covariates = c("sex", "age", "residency", "job",
                 "education", "comorbidity_count", "ever_smoker")
)

# Phase 1: fit the longitudinal GRM tree
ltree <- longitudinal_grmtree(
  resp_wide ~ sex + age + residency + job +
    education + comorbidity_count + ever_smoker,
  data = ld, n_items = 8,
  control = grmtree.control(minbucket = 200)
)

# Phase 2: characterize response shift within each subgroup
rs <- rs_characterize(ltree, p_adjust = "fdr",
  global_p_adjust = "bonferroni")

plot_rs_tree(ltree, rs,
  item_labels = c("Listen", "Info", "Crisis", "Confide",
    "Advice", "Fears", "Personal", "Understand"))
```

`prepare_longitudinal_data` *Prepare Wide-Format Longitudinal PROM Data*

Description

Constructs a wide-format data frame suitable for `longitudinal_grmtree` from separate Time 1 and Time 2 item response columns. The function creates a matrix-valued column containing the concatenated T1 and T2 responses, along with any specified covariates.

Usage

```
prepare_longitudinal_data(
  data,
  items_t1,
  items_t2,
  covariates = NULL,
  id = NULL
)
```

Arguments

`data`	A data frame containing item response columns for both time points and any covariates.

items_t1	Character vector of column names for Time 1 item responses, in the order they should appear in the response matrix.
items_t2	Character vector of column names for Time 2 item responses, in the same order as items_t1. Must have the same length.
covariates	Optional character vector of column names for covariates to include in the output data frame (e.g., c("age", "sex")).
id	Optional character string specifying the subject ID column name. If provided, the ID column is included in the output.

Value

A data frame with:

Covariates All columns specified in covariates

resp_wide A matrix-valued column with 2 * length(items_t1) columns. The first half contains T1 responses (named Item1_T1, ..., ItemM_T1) and the second half contains T2 responses (named Item1_T2, ..., ItemM_T2).

Rows with any missing values in the response matrix are removed, with a message indicating how many rows were dropped.

See Also

longitudinal_grmtree for fitting the tree

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix from separate T1 and T2 columns
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )

  # Check structure
  str(ld$resp_wide)  # 1500 x 16 matrix
```

print.grmforest *Print Method for GRM Forest*

Description

Print Method for GRM Forest

Usage

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

Arguments

x	A grmforest object
...	Additional arguments (currently unused)

Value

Invisibly returns the input object

print.grmtree *Print Method for GRM Tree Objects*

Description

Displays a formatted summary of a GRM (Graded Response Model) tree object. This function extends print.modelparty from the partykit package with specialized formatting for GRM trees.

Usage

```
## S3 method for class 'grmtree'
print(
  x,
  title = "Graded Response Model Tree",
  objfun = "negative log-likelihood",
  ...
)
```

Arguments

x	A GRM tree object of class 'grmtree'.
title	Character string specifying the title for the print output (default: "Graded Response Model Tree").
objfun	Character string labeling the objective function (default: "negative log-likelihood").
...	Additional arguments passed to print.modelparty.

Details

The print method provides a comprehensive summary of the GRM tree, including:

- Model formula used for fitting
- Tree structure with node information
- Item parameter estimates for each terminal node
- Confidence intervals for parameters
- Group parameters (mean and covariance)
- Summary statistics (number of nodes, objective function value)

Value

Invisibly returns the GRM tree object. Primarily called for its side effect of printing a formatted summary.

See Also

`print.modelparty` for the underlying printing infrastructure, `grmtree` for creating GRM tree objects

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit GRM tree
tree <- grmtree(resp ~ gender + group,
                data = asti,
                control = grmtree.control(minbucket = 30))

# Print the tree summary
print(tree)

# Alternative syntax (automatically calls print.grmtree)
tree
```

---

print.rs_characterization

Print Method for Response Shift Characterization Results

Description

Displays a formatted summary of response shift characterization results from `rs_characterize`, including omnibus RS test results per terminal node and item-level RS testing results with significance markers.

Usage

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

Arguments

`x`	An `rs_characterization` object returned by `rs_characterize`.
`...`	Additional arguments (currently unused).

Details

The output is organized in two sections:

Omnibus RS Test: For each terminal node, displays the LRT statistic, degrees of freedom, raw and adjusted p-values (if applicable), RS detection status, effect sizes (pseudo R-squared and AIC difference), latent mean shift, T1-T2 correlation, and convergence warnings.

Item-Level RS Testing: For nodes where omnibus RS was detected, displays per-item LRT statistics, raw and adjusted p-values, RS type classification, and significance markers (*** for items significant after adjustment). Also reports the p-value adjustment method and RS classification threshold used.

Value

Invisibly returns `x`. Called for its side effect of printing formatted output to the console.

See Also

`rs_characterize` for computing the results

`rs_characterize` *Characterize Response Shift Within Terminal Nodes of a Longitudinal GRM Tree*

Description

After fitting a `longitudinal_grmtree` (Phase 1), this function performs Phase 2: post-hoc response shift (RS) characterization within each terminal node. For each subgroup identified by the tree, it compares the constrained model (no RS: item parameters equal across time) to an unconstrained model (item parameters free across time) using likelihood ratio tests (LRTs), then identifies which specific items exhibit response shift and classifies the type of shift.

Usage

```
rs_characterize(
  object,
  node = NULL,
  item_level = TRUE,
  alpha = 0.05,
  p_adjust = "bonferroni",
  global_p_adjust = "bonferroni",
  rs_threshold = 0.3,
  verbose = TRUE,
  ...
)
```

Arguments

object	A longitudinal_grmtree object fitted by longitudinal_grmtree.
node	Optional integer vector of terminal node IDs to analyze. If NULL (default), all terminal nodes are analyzed.
item_level	Logical. If TRUE (default), perform item-level RS testing by sequentially freeing one item at a time from the constrained model. Only performed for nodes where the omnibus test is significant.
alpha	Numeric significance level for RS tests. Default is 0.05. Used for both omnibus and item-level significance decisions (after any p-value adjustments).
p_adjust	Character string specifying the method for adjusting item-level p-values **within each node**. Controls the error rate when testing multiple items for RS within a single subgroup. Options: "bonferroni" (default), "holm", "BH" (Benjamini-Hochberg), "BY", "fdr", "hochberg", "hommel", or "none". See p.adjust.
global_p_adjust	Character string specifying the method for adjusting omnibus RS p-values **across terminal nodes**. Controls the family-wise error rate when testing RS in multiple subgroups. Options are the same as p_adjust. Default is "bonferroni". The RS_detected column in the global results is updated based on adjusted p-values. Only applied when there are 2 or more nodes.
rs_threshold	Numeric threshold for classifying the magnitude of parameter change in RS type determination, on the IRT parameter scale. Default is 0.3. A discrimination difference exceeding this threshold indicates reprioritization; a maximum threshold difference exceeding this indicates recalibration. Larger values produce more conservative classifications (fewer items classified as showing meaningful RS). Items that are statistically significant but below this threshold are classified as "Significant (small effect)".
verbose	Logical. If TRUE (default), print progress messages including log-likelihoods, LRT statistics, and RS classifications for each node during processing.
...	Additional arguments passed to mirt during model fitting.

Details

The Two-Phase Framework:

The Longitudinal GRMTree operates as an integrated two-phase method:

Phase 1 (`longitudinal_grmtree`): Embeds a constrained longitudinal GRM (item parameters equal across T1 and T2) within model-based recursive partitioning. The tree identifies patient subgroups requiring different constrained measurement models. This ensures that subsequent RS testing is conducted within measurement-homogeneous groups.

Phase 2 (this function): Within each terminal node, relaxes the equality constraint and tests whether item parameters actually change over time. This separates between-group measurement heterogeneity (captured by the tree) from within-group temporal change (response shift).

Omnibus RS Test:

For each terminal node, the function fits two models to the node's data:

1. **Constrained model:** Item parameters are equal across T1 and T2 (no response shift). This model has the same structure as the model fitted during Phase 1.
2. **Unconstrained model:** Item parameters are free to differ between T1 and T2 (response shift allowed). All equality constraints on discrimination and threshold parameters are removed.

The likelihood ratio test statistic is:

$$\chi^2 = -2(LL_{constrained} - LL_{unconstrained})$$

with degrees of freedom equal to the number of freed constraints (typically $M \times K$ for M items with $K - 1$ thresholds each, plus M discrimination parameters).

Item-Level RS Testing:

When the omnibus test is significant, each item is tested individually by freeing only that item's constraints while keeping all other items constrained. The LRT compares the fully constrained model to the partially constrained model with one item freed. This identifies which specific items contribute to the overall response shift.

RS Type Classification:

For items showing statistically significant RS (after p-value adjustment), the type is classified by examining the freed item's T1 vs T2 parameters from the partially constrained model:

Recalibration Maximum absolute threshold difference exceeds `rs_threshold`. Interpretation: the patient's internal standards for endorsing response categories changed over time (Sprangers & Schwartz, 1999).

Reprioritization Absolute discrimination difference exceeds `rs_threshold`. Interpretation: the item's relationship to the underlying latent construct changed over time — the item became more or less central to the construct.

Both Both discrimination and threshold changes exceed `rs_threshold`.

Significant (small effect) The LRT is significant but neither parameter change exceeds `rs_threshold`. The RS is statistically detectable but small in magnitude.

Hierarchical P-Value Correction:

The function implements a two-level correction strategy:

1. **Across nodes (omnibus):** `global_p_adjust` controls the family-wise error rate for the question "in which subgroups does RS occur?" Default is Bonferroni.

2. **Within nodes (item-level):** `p_adjust` controls the error rate for the question "which items exhibit RS within a given subgroup?" Default is Bonferroni; FDR (Benjamini-Hochberg) is recommended for exploratory analyses.

Effect Size Measures:

McFadden's pseudo R-squared $1 - LL_{uncon}/LL_{con}$. Inherently small (~0.02) in IRT models because item parameters account for only a fraction of the total likelihood. Values should be compared across nodes rather than interpreted in absolute terms.

AIC difference $AIC_{con} - AIC_{uncon}$. Positive values indicate the unconstrained model fits better even after penalizing for additional parameters. Negative values suggest the constrained model is preferred (response shift is not substantial enough to justify the additional parameters).

Value

A list of class `"rs_characterization"` containing:

`global` A `data.frame` with one row per terminal node and columns:

- `Node` Terminal node ID
- `n` Sample size in the node
- `LL_constrained` Log-likelihood of the constrained (no-RS) model
- `LL_unconstrained` Log-likelihood of the unconstrained (RS-allowed) model
- `LRT_chi2` Likelihood ratio test statistic: $-2(LL_{con} - LL_{uncon})$
- `LRT_df` Degrees of freedom (number of freed constraints)
- `LRT_p` Raw p-value from chi-squared distribution
- `LRT_p_adj` Adjusted p-value (present only if `global_p_adjust != "none"` and there are 2+ nodes)
- `RS_detected` Logical: whether RS is detected at the specified `alpha` level (uses adjusted p if available)
- `pseudo_R2` McFadden's pseudo R-squared: $1 - LL_{uncon}/LL_{con}$
- `AIC_diff` AIC difference: $AIC_{con} - AIC_{uncon}$. Positive values indicate the unconstrained model fits better after parsimony penalty.
- `mu_T2` Latent mean shift at T2 from the constrained model. Positive = improvement, negative = decline.
- `sigma2_T2` Latent variance at T2
- `cor_T1_T2` Correlation between latent traits at T1 and T2, reflecting stability of individual differences
- `converged_constrained` Logical: whether the constrained model converged
- `converged_unconstrained` Logical: whether the unconstrained model converged

`item_level` A `data.frame` with item-level RS test results (one row per item per node where omnibus RS was detected), with columns:

- `Node` Terminal node ID
- `Item` Item number (1 to `n_items`)
- `LRT_chi2` LRT statistic for freeing this item
- `LRT_df` Degrees of freedom (1 discrimination + thresholds for this item)

LRT_p Raw p-value

LRT_p_adj Adjusted p-value (if p_adjust != "none")

RS_type Classification: "Recalibration", "Reprioritization", "Both", "Significant (small effect)", or "None"

NULL if no nodes showed significant omnibus RS or item_level = FALSE.

parameters A named list (one element per node) of lists containing constrained and unconstrained item parameter matrices (IRT parameterization with discrimination a1/a2 and thresholds b1, ..., bK).

n_items Number of items per time point.

alpha Significance level used.

p_adjust Item-level p-value adjustment method.

global_p_adjust Omnibus p-value adjustment method.

rs_threshold RS classification threshold.

Author(s)

Olayinka Imisioluwa Arimoro <olayinka.arimoro@ucalgary.ca>, Lisa M. Lix, Tolulope T. Sajobi

References

Sprangers, M. A., & Schwartz, C. E. (1999). Integrating response shift into health-related quality of life research. *Social Science & Medicine*, 48(11), 1507–1515.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1), 289–300.

See Also

longitudinal_grmtree for Phase 1 (tree fitting), print.rs_characterization for the print method, prepare_longitudinal_data for data preparation, p.adjust for p-value adjustment methods

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
  items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
                "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
  ld <- prepare_longitudinal_data(
    data = grmtree_long_data,
    items_t1 = items_t1,
    items_t2 = paste0(items_t1, "_year1"),
    covariates = c("sex", "age", "residency", "job",
                   "education", "comorbidity_count", "ever_smoker")
  )
```

```
# After fitting the tree (Phase 1)
ltree <- longitudinal_grmtree(
  resp_wide ~ sex + age + residency + job +
    education + comorbidity_count + ever_smoker,
  data = ld, n_items = 8,
  control = grmtree.control(minbucket = 200)
)

# Phase 2: characterize response shift within each subgroup
rs <- rs_characterize(ltree, p_adjust = "fdr",
  global_p_adjust = "bonferroni")
print(rs)

# Phase 2: Characterize RS
# Default: Bonferroni at both levels, threshold = 0.3
rs <- rs_characterize(ltree)

# Recommended: FDR for item-level, Bonferroni for omnibus
rs <- rs_characterize(ltree,
  p_adjust = "fdr",
  global_p_adjust = "bonferroni",
  rs_threshold = 0.3
)

# No omnibus correction (each node tested independently)
rs <- rs_characterize(ltree, global_p_adjust = "none")

# More conservative RS classification
rs <- rs_characterize(ltree, rs_threshold = 0.5)

# View results
print(rs)

# Access components
rs$global         # Omnibus results per node
rs$item_level     # Item-level results
rs$parameters     # Raw parameter matrices
```

`threshpar_grmtree` *Extract Threshold Parameters from GRM Tree*

Description

Extracts threshold parameters for each item from all terminal nodes of a graded response model tree. The thresholds represent the points on the latent trait continuum where the probability of scoring in adjacent response categories is equal.

Usage

```
threshpar_grmtree(object, node = NULL, ...)
```

Arguments

object	A grmtree object.
node	Optional vector of node IDs to extract from. If NULL (default), extracts from all terminal nodes.
...	Additional arguments (currently unused).

Value

A data.frame with threshold parameters for each item in each node, with columns:

Node	Node ID
Item	Item name
d1, d2, ...	Threshold parameters for each category

See Also

grmtree fits a Graded Response Model Tree, grmforest for GRM Forests, fscores_grmtree for computing factor scores, discrpar_grmtree for extracting discrimination parameters, itempar_grmtree for extracting item parameters

Examples

```
library(grmtree)
library(hlt)

data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

# Fit GRM tree with gender and group as partitioning variables
tree <- grmtree(resp ~ gender + group,
        data = asti,
        control = grmtree.control(minbucket = 30))

# Get all thresholds
thresholds <- threshpar_grmtree(tree)
print(thresholds)
```

`threshpar_longitudinal_grmtree` *Extract Threshold Parameters from Longitudinal GRM Tree*

Description

Extracts threshold (difficulty) parameters for each unique item from all terminal nodes of a longitudinal GRM tree. Only T1 item parameters are returned since the constrained model enforces equal thresholds across T1 and T2 within each node.

Usage

```
threshpar_longitudinal_grmtree(object, node = NULL, clean_names = TRUE, ...)
```

Arguments

`object`	A `longitudinal_grmtree` object.
`node`	Optional vector of node IDs. If NULL (default), extracts from all terminal nodes.
`clean_names`	Logical. If TRUE (default), clean item names by removing common prefixes (e.g., "resp_wideMOS_") and suffixes (e.g., "_BL", "_T1").
`...`	Additional arguments (currently unused).

Value

A data.frame with columns:

Node Terminal node ID

Item Item name (cleaned if `clean_names = TRUE`)

b1, b2, ..., bK Threshold parameters for each category boundary

Contains `n_items` rows per node (not `2 * n_items`).

See Also

`longitudinal_grmtree` for Phase 1 (tree fitting), `discrpar_longitudinal_grmtree` for extracting discrimination parameters for longitudinal GRMTree, `itempar_longitudinal_grmtree` for extracting item parameters for longitudinal GRMTree

Examples

```
library(grmtree)

# Load the synthetic longitudinal data
data("grmtree_long_data", package = "grmtree")

  # Prepare the wide-format response matrix
```

```
items_t1 <- c("MOS_Listen", "MOS_Info", "MOS_Advice_Crisis", "MOS_Confide",
              "MOS_Advice_Want", "MOS_Fears", "MOS_Personal", "MOS_Understand")
ld <- prepare_longitudinal_data(
  data = grmtree_long_data,
  items_t1 = items_t1,
  items_t2 = paste0(items_t1, "_year1"),
  covariates = c("sex", "age", "residency", "job",
                 "education", "comorbidity_count", "ever_smoker")
)

# Phase 1: fit the longitudinal GRM tree
ltree <- longitudinal_grmtree(
  resp_wide ~ sex + age + residency + job +
    education + comorbidity_count + ever_smoker,
  data = ld, n_items = 8,
  control = grmtree.control(minbucket = 200)
)

# Print the threshold parameters
thresholds <- threshpar_longitudinal_grmtree(ltree)
print(thresholds)

# Using the raw names
threshpar_longitudinal_grmtree(ltree, clean_names = FALSE)  # raw names
```

`varimp` *Calculate Variable Importance for GRM Forest*

Description

Computes permutation importance scores for variables in a GRM forest using out-of-bag samples. Importance is measured by the decrease in log-likelihood when a variable's values are permuted.

Usage

```
varimp(forest, method = "permutation", verbose = FALSE, seed = NULL)
```

Arguments

`forest`	A `grmforest` object created by `grmforest()`.
`method`	Importance calculation method (currently only "permutation").
`verbose`	Logical indicating whether to show progress messages.
`seed`	Random seed for reproducibility.

Value

A named numeric vector of importance scores with class `varimp`. Higher values indicate more important variables.

See Also

grmtree fits a Graded Response Model Tree, grmforest for GRM Forests, grmforest.control creates a control object for grmforest, plot.varimp creates a bar plot of variable importance scores

Examples

```
library(grmtree)
library(hlt)
data("asti", package = "hlt")
asti$resp <- data.matrix(asti[, 1:4])

## Fit the GRM Forest
forest <- grmforest(resp ~ gender + group, data = asti,
control = grmforest.control(n_tree = 5))
  importance <- varimp(forest)

## Print and plot the variable importance scores
print(importance)
plot(importance)
```

Index

∗ **datasets**
    grmtree_data, 17
    grmtree_long_data, 19

data_party, *27*
discrpar_grmtree, 3, *6*, *9*, *16*, *21*, *47*
discrpar_longitudinal_grmtree, 4, *22*, *48*

fscores, *6*
fscores_grmtree, *3*, 5, *6*, *9*, *16*, *21*, *47*
fscores_longitudinal_grmtree, 6, *9*, *23*

generate_node_scores_dataset, *6*, *7*, 8
grmforest, *3*, *6*, 9, *9*, *12*, *16*, *21*, *33*, *47*, *50*
grmforest.control, *10*, 11, *33*, *50*
grmtree, *3*, *6*, *9*, *10*, 12, *17*, *21*, *24*, *28*, *30*, *40*, *47*, *50*
grmtree.control, *10*, *12*, 16, *16*, *25*, *28*
grmtree_data, 17
grmtree_long_data, 19

itempar_grmtree, *3*, *6*, *9*, *16*, 20, *47*
itempar_longitudinal_grmtree, *4*, 21, *48*

latentpar_longitudinal_grmtree, 23
longitudinal_grmfit, *25*
longitudinal_grmtree, *4*, *7*, *9*, *19*, *20*, *22*, *23*, 24, *32*, *37*, *38*, *41–43*, *45*, *48*

mirt, *25*, *42*
mob, *28*
modelparty, *27*

nodeids, *27*

p.adjust, *42*, *45*
plot.grmtree, *12*, *16*, 29, *31–33*
plot.longitudinal_grmtree, 31
plot.modelparty, *30*
plot.varimp, *10*, *30*, 33, *50*
plot_rs_heatmap, 34
plot_rs_tree, 35
prepare_longitudinal_data, *27*, *28*, 37, *45*
print.grmforest, 39
print.grmtree, *16*, 39
print.modelparty, *40*
print.rs_characterization, 40, *45*

rs_characterize, *19*, *20*, *24–28*, *34*, *36*, 41, *41*

threshpar_grmtree, *3*, *6*, *9*, *16*, *21*, 46
threshpar_longitudinal_grmtree, *4*, *22*, 48

varimp, *10*, *16*, *33*, 49