

Package ‘plotsirm’

May 9, 2026

Type Package

Title Plot Toolkit for Latent Space Item Response Models

Version 0.1.3

Description Provides publication-quality and interactive plots for exploring the posterior output of Latent Space Item Response Models, including Posterior Interaction Profiles, radar charts, 2-D latent maps, and item-similarity heat maps. The methods implemented in this package are based on work by Jeon, M., Jin, I. H., Schweinberger, M., Baugh, S. (2021) <[doi:10.1007/s11336-021-09762-5](https://doi.org/10.1007/s11336-021-09762-5)>.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4), rlang, dplyr, tidyr, patchwork, scales, grid

Suggests plotly, testthat (>= 3.0.0), spelling

RoxygenNote 7.2.3

URL <https://github.com/jevanluo/plotsirm>

BugReports <https://github.com/jevanluo/plotsirm/issues>

Author Jinwen Luo [aut, cre] (ORCID: <<https://orcid.org/0000-0002-8511-7165>>),
Minjeong Jeon [aut] (ORCID: <<https://orcid.org/0000-0002-5880-4146>>)

Maintainer Jinwen Luo <jevanluo@ucla.edu>

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2025-10-01 07:10:02 UTC

Contents

crossdist_fast	2
iccsurface	3

intermap2d	5
interprofile	11
itemheatmap	12
itemsimilarity	13
lsirmicc	15
pip_fountain	18
pip_waterfall	20
radarplot	22
rescale_to_range	24
strengthplot	25
vec_mat_dist	27

Index 29

crossdist_fast	<i>Fast pairwise (cross) Euclidean distances</i>
----------------	--

Description

Computes the Euclidean distance between every row of a "person" matrix (z , shape $N \times d$) and every row of an "item" matrix (w , shape $I \times d$). An optional `item_labels` argument lets you collapse items into groups first, replacing each group with its centroid before distances are calculated.

Usage

```
crossdist_fast(z, w, item_labels = NULL)
```

Arguments

- | | |
|--------------------------|---|
| <code>z</code> | Numeric matrix of shape $N \times d$. Each row is a point in a d -dimensional latent space representing a person. |
| <code>w</code> | Numeric matrix of shape $I \times d$. Each row is a point in a d -dimensional latent space representing an item. |
| <code>item_labels</code> | Optional character or factor vector of length I giving a group label for each item row in w . <ul style="list-style-type: none"> • If supplied, the function first replaces the items in each group with their centroid (mean position) so the output contains distances to those centroids rather than to individual items. • If NULL (default), distances are computed to every item. |

Details

The computation exploits the identity

$$\|z_j - w_i\|^2 = \|z_j\|^2 + \|w_i\|^2 - 2z_j^\top w_i.$$

allowing all pairwise squared distances to be obtained with a single matrix multiplication. Negative rounding errors are clipped at zero before taking the square root.

Value

A numeric distance matrix.

- When `item_labels` is `NULL`, the result is $N \times I$: distance from every person to every item.
- When `item_labels` is provided, the result is $N \times G$, where G is the number of distinct groups.

Examples

```
set.seed(42)
z <- matrix(rnorm(15), nrow = 5) # 5 persons in 3-D
w <- matrix(rnorm(30), nrow = 10) # 10 items in 3-D

# Person-item distances
d_full <- crossdist_fast(z, w)

# Person-group distances (items grouped into two sets)
grp <- rep(c("A", "B"), each = 5)
d_group <- crossdist_fast(z, w, item_labels = grp)
```

iccsurface

Latent-Space Item Characteristic Surface

Description

Evaluates the LSIRM probability

$$P(Y_{pi} = 1) = \text{logit}^{-1}(\alpha + \beta - \gamma d)$$

on a rectangular grid of *ability* (α) and *person-item distance* (d) values and, by default, renders the resulting surface interactively with `plotly`.

Usage

```
iccsurface(
  beta,
  alpha_lim = c(-4, 4),
  n_alpha = 60,
  dist_lim = c(0, 4),
  n_dist = 60,
  gamma = 1,
  colour_mode = c("uniform", "gradient"),
  surface_col = "steelblue",
  palette = "Viridis",
  dark_high = TRUE,
  surface_opacity = NULL,
  show_grid = TRUE,
  grid_step = 5,
  plot = TRUE
)
```

Arguments

beta	Numeric scalar β_i shifting the surface along the probability axis (item "easiness").
alpha_lim, n_alpha	Numeric. Range $c(\min, \max)$ and grid size for the ability axis. Default is $\alpha \in [-4, 4]$ with 60 points.
dist_lim, n_dist	Numeric. Range $c(\min, \max)$ and grid size for the distance axis. Default is $d \in [0, 4]$ with 60 points.
gamma	Positive scalar controlling how strongly the probability decays with distance.
colour_mode	"uniform" (default) or "gradient".
surface_col	Single colour used when <code>colour_mode = "uniform"</code> .
palette	Character name of a plotly continuous palette (e.g. "Viridis", "Hot", "Blues"); only used when <code>colour_mode = "gradient"</code> .
dark_high	Logical. If TRUE (default) reverses the palette so high probabilities map to darker shades.
surface_opacity	Numeric in (0, 1]. By default the function chooses 1 for uniform and 0.9 for gradient surfaces so the wire-frame remains visible.
show_grid	Logical. Overlay a wire-frame? Default TRUE.
grid_step	Positive integer: draw every <code>grid_step</code> -th row/column when <code>show_grid = TRUE</code> .
plot	Logical. If TRUE (default) return an interactive plotly surface; if FALSE return the raw numeric grid.

Details

Colour options:

- **Uniform** - a single-colour surface (`colour_mode = "uniform"`, default). The colour is set by `surface_col`.
- **Gradient** - a continuous plotly palette (`colour_mode = "gradient"`); the palette is chosen via `palette`, and `dark_high = TRUE` reverses the scale so higher probabilities appear darker.

Wire-frame:

Setting `show_grid = TRUE` overlays a black wire-frame every `grid_step` rows/columns to emphasise the surface curvature.

The scene's aspect is fixed to a cube, and bold zero-lines are drawn on the α - and *distance* axes so their origins align visually.

Value

- `plot = TRUE` - a plotly `htmlwidget` (prints automatically).
- `plot = FALSE` - a list with components `alpha`, `distance`, and `prob` (an $n_\alpha \times n_{\text{dist}}$ matrix of probabilities).

Dependencies

Rendering the surface requires the plotly package (`install.packages("plotly")`). No external packages are needed when `plot = FALSE`.

Examples

```
## Numeric output only
surf <- iccsurface(beta = 0, n_alpha = 21, n_dist = 21, plot = FALSE)
str(surf)
```

```
## Interactive surfaces
## 1. Uniform single-colour
iccsurface(beta = -0.5)
```

```
## 2. Gradient "Hot" palette, darker = high P
iccsurface(beta = 0.3,
            colour_mode = "gradient",
            palette      = "Hot")
```

intermap2d

2-D latent-space interaction map (persons vs. items)

Description

Persons (z) and items (w) in a 2D latent space with flexible styling. Person parameter α_p , item parameter β_i .

Usage

```
intermap2d(
  z,
  w,
  gamma = NULL,
  person_group = NULL,
  item_group = NULL,
  person_colors = NULL,
  item_colors = NULL,
  alpha = NULL,
  beta = NULL,
  z_shape_size_scale = FALSE,
  w_shape_size_scale = FALSE,
  z_shape_size_range = c(2, 6),
  w_shape_size_range = c(2, 8),
  z_shape_size = 2.5,
  w_shape_size = 3,
```

```

z_label_size_scale = NULL,
w_label_size_scale = NULL,
z_label_size_range = c(3, 7),
w_label_size_range = c(3, 7),
z_label_color = "navy",
w_label_color = "firebrick",
z_label_size = 4,
w_label_size = 4,
z_shape_opacity_scale = FALSE,
w_shape_opacity_scale = FALSE,
z_shape_opacity_range = c(0.3, 1),
w_shape_opacity_range = c(0.3, 1),
z_shape_fixed_opacity = NULL,
w_shape_fixed_opacity = NULL,
z_shape_color_gradient = FALSE,
w_shape_color_gradient = FALSE,
z_shape_color_values = NULL,
w_shape_color_values = NULL,
shape_color_gradient_low = "grey80",
shape_color_gradient_high = "navy",
z_shape_color_gradient_low = NULL,
z_shape_color_gradient_high = NULL,
w_shape_color_gradient_low = NULL,
w_shape_color_gradient_high = NULL,
show_ticks = FALSE,
xlim_range = NULL,
ylim_range = NULL,
itemlabels = NULL,
personlabels = NULL,
figuretitle = NULL,
z_shape = 16,
w_shape = 17,
z_shape_color = "navy",
w_shape_color = "firebrick",
z_border_width = 0.5,
show_z_labels = FALSE,
show_w_labels = FALSE,
show_z_shapes = TRUE,
show_w_shapes = TRUE,
legend_title = "legend",
show_size_legend = FALSE,
share_gradient_scale = FALSE,
legend_title_z = expression(alpha[p]),
legend_title_w = expression(beta[i])
)

```

Arguments

z, w Numeric matrices with 2 columns: coordinates for persons/items.

gamma	Optional scalar stretch factor applied to both z and w.
person_group, item_group	Optional factor/character for grouping colors.
person_colors, item_colors	Optional explicit color vectors (length N/I).
alpha, beta	Optional vectors used to scale shape/label sizes, opacity, and (optionally) gradients for persons/items, respectively.
z_shape_size_scale, w_shape_size_scale	Logical: scale shape sizes by alpha/beta.
z_shape_size_range, w_shape_size_range	Length-2 numeric: shape size ranges (when scaling).
z_shape_size, w_shape_size	Numeric: fixed shape sizes when size scaling is OFF.
z_label_size_scale, w_label_size_scale	Logical: scale label sizes. If NULL, defaults to the corresponding shape-size flag.
z_label_size_range, w_label_size_range	Length-2 numeric: label size ranges.
z_label_color, w_label_color	Label colors; if NULL, labels follow the layer's color (group/gradient) or fixed color/vector as appropriate.
z_label_size, w_label_size	Fixed label sizes when not scaling.
z_shape_opacity_scale, w_shape_opacity_scale	Logical: scale shape opacity by alpha/beta.
z_shape_opacity_range, w_shape_opacity_range	Length-2 numeric: opacity ranges.
z_shape_fixed_opacity, w_shape_fixed_opacity	Optional constant opacity (0..1) for shapes.
z_shape_color_gradient, w_shape_color_gradient	Logical: color shapes by a gradient (darker indicates higher). Overrides groups/colors for that layer. If both are TRUE, the gradients can be shared (share_gradient_scale = TRUE) or separated (share_gradient_scale = FALSE, default).
z_shape_color_values, w_shape_color_values	Optional numeric drivers for gradients (defaults: alpha / beta respectively).
shape_color_gradient_low, shape_color_gradient_high	Global colors for the gradient palette. Used directly when share_gradient_scale = TRUE, or as fallbacks for layer-specific palettes when share_gradient_scale = FALSE.
z_shape_color_gradient_low, z_shape_color_gradient_high	Optional colors for the persons (z) gradient when using separate scales. If NULL, fall back to shape_color_gradient_low/high.
w_shape_color_gradient_low, w_shape_color_gradient_high	Optional colors for the items (w) gradient when using separate scales. If NULL, fall back to shape_color_gradient_low/high.
show_ticks	Logical: draw axis ticks/labels.

xlim_range, ylim_range	Optional axis limits (symmetric if NULL).
itemlabels, personlabels	Optional labels (defaults: "I1..", "P1..").
figuretitle	Optional plot title.
z_shape, w_shape	ggplot2 shape codes for persons/items (see ?ggplot2::geom_point).
z_shape_color, w_shape_color	Fixed fallback shape colors when not mapping.
z_border_width	Stroke width for z shapes (when applicable).
show_z_labels, show_w_labels	Logical: draw labels for z/w.
show_z_shapes, show_w_shapes	Logical: draw shapes for z/w.
legend_title	Character or expression: the legend title (when shown).
show_size_legend	Logical: show a size legend (default FALSE).
share_gradient_scale	Logical. If TRUE, persons and items share one gradient/legend (uses shape_color_gradient_low/high and legend_title). If FALSE (default), persons and items use separate gradients/legends: persons mapped to color (title legend_title_z) and items mapped to fill (title legend_title_w).
legend_title_z, legend_title_w	Titles (character or expressions) for the separate z and w gradient legends, used only when share_gradient_scale = FALSE and both gradient mappings are enabled.

Details

Coloring & legends

- A color legend appears when colors are mapped via groups or gradients. For fixed colors/shapes on both layers, the function creates a simple two-entry legend ("Persons", "Items") using constant mappings; when shapes are off but labels are on, it builds a labels-only legend with colored swatches. The legend title is controlled by legend_title.
- When both persons and items use gradient coloring (z_shape_color_gradient = TRUE and w_shape_color_gradient = TRUE), you can either **share one gradient and legend** by setting share_gradient_scale = TRUE (uses shape_color_gradient_low/high and legend_title), or show **separate gradients/legends** for z and w by keeping share_gradient_scale = FALSE (default). In separate mode, persons use the color scale with title legend_title_z, and items use the fill scale with title legend_title_w. Layer-specific palettes can be supplied via z_shape_color_gradient_low/high and w_shape_color_gradient_low/high; if NULL, the global shape_color_gradient_low/high are used as fallbacks.
- Opacity (ggplot "alpha") and size legends are hidden by default. Set show_size_legend = TRUE to show a size legend when size mapping is used.

- When `person_colors / item_colors` are **vectors** and `z_label_color / w_label_color` are NULL, label colors follow those per-observation vectors.

Opacity

- Transparency is called *opacity* to avoid confusion with the statistic alpha. When you supply explicit fixed colors (single color or vector), default opacity is full (1) unless you enable `*_shape_opacity_scale` or set `*_shape_fixed_opacity`.

Sizes

- Shape sizes are fixed by `z_shape_size / w_shape_size` unless `z_shape_size_scale / w_shape_size_scale` are TRUE, in which case sizes are scaled by alpha / beta into `z_shape_size_range / w_shape_size_range`.
- Label text sizes are fixed by `z_label_size / w_label_size` unless `z_label_size_scale / w_label_size_scale` are TRUE, in which case they are scaled by alpha / beta into `z_label_size_range / w_label_size_range`.

Value

Invisibly returns a ggplot object; also prints the plot.

Examples

```
### example data
set.seed(1)
z <- matrix(rnorm(40), 20, 2) # persons
w <- matrix(rnorm(30), 15, 2) # items
alpha <- rnorm(nrow(z))      # person alpha
beta  <- rnorm(nrow(w))      # item beta

### 1) minimal, fixed colors & shapes
intermap2d(z, w)

### 2) minimal, fixed shapes for persons and labels for items
intermap2d(
  z, w,
  show_w_shapes = FALSE, show_w_labels = TRUE
)

### 3) Grouped colors + sized shapes, formal legend title
intermap2d(
  z, w,
  person_group = rep(c("Cohort A", "Cohort B"), length.out = nrow(z)),
  item_group   = rep(c("Domain X", "Domain Y", "Domain Z"), length.out = nrow(w)),
  alpha = alpha, beta = beta,
  z_shape_size_scale = TRUE, z_shape_size_range = c(2, 6),
  w_shape_size_scale = TRUE, w_shape_size_range = c(2, 8),
  show_z_shapes = TRUE, show_w_shapes = FALSE,
  show_w_labels = TRUE, legend_title = "Cohort / Domain"
)

### 4) Gradient for persons only (darker = higher alpha), labels scaled by alpha
intermap2d(
```

```

z, w,
alpha = alpha,
z_shape_color_gradient = TRUE,    # z by gradient
w_shape_color           = "red",   # w fixed color
z_label_size_scale     = TRUE,    # label size proportional to alpha
show_w_shapes = FALSE,
show_w_labels = TRUE,
# shape_color_gradient_low = "grey80", shape_color_gradient_high = "navy",
legend_title = expression(alpha[p])
)

### 5) Gradient for both alpha (persons) and beta (items), shared legend
intermap2d(
  z, w,
  alpha = alpha, beta = beta,
  z_shape_color_gradient = TRUE, w_shape_color_gradient = TRUE,
  shape_color_gradient_low = "grey80", shape_color_gradient_high = "navy",
  z_shape_size_scale = TRUE, w_shape_size_scale = TRUE,
  show_z_shapes = TRUE, show_w_shapes = TRUE,
  show_z_labels = FALSE, show_w_labels = FALSE,
  legend_title = "Intensity (alpha persons, beta items)"
)

### 6) Explicit per-observation colors (vectors) + label-only
z_cols <- ifelse(alpha > 0, "#1f77b4", "#AEC7E8")
w_cols <- ifelse(beta > 0, "#d62728", "#FF9896")
intermap2d(
  z, w,
  person_colors = z_cols, item_colors = w_cols,
  show_z_shapes = FALSE, show_w_shapes = FALSE,
  show_z_labels = TRUE, show_w_labels = TRUE
)

### 7) Opacity scaling + fixed shape sizes
intermap2d(
  z, w,
  alpha = alpha, beta = beta,
  z_shape_opacity_scale = TRUE, z_shape_opacity_range = c(0.2, 1.0),
  w_shape_opacity_scale = TRUE, w_shape_opacity_range = c(0.4, 1.0),
  z_shape_color = "black", w_shape_color = "orange3",
  z_shape_size = 3, w_shape_size = 3.5,          # fixed sizes (no size scaling)
  show_z_shapes = TRUE, show_w_shapes = TRUE
)

### 8) Label-only scaling; shape sizes fixed; custom legend title for groups
intermap2d(
  z, w,
  person_group = rep(c("High", "Low"), length.out = nrow(z)),
  alpha = alpha, beta = beta,
  z_label_size_scale = TRUE, w_label_size_scale = TRUE,
  z_label_size_range = c(3, 7), w_label_size_range = c(3, 7),
  show_z_labels = TRUE, show_w_labels = FALSE,
  show_z_shapes = FALSE, show_w_shapes = TRUE,

```

```

    legend_title = "Performance Group"
  )

  ### 9) Stretch coordinates + axis ticks + symmetric limits
  intermap2d(
    z, w,
    gamma = 1.5,
    show_ticks = TRUE,
    xlim_range = c(-4, 4), ylim_range = c(-4, 4),
    person_group = rep(c("Train", "Test"), length.out = nrow(z)),
    legend_title = "Set Membership"
  )

```

interprofile

Draw a Posterior Interaction Profile in either style

Description

Convenience wrapper that calls `pip_fountain()` (default) or `pip_waterfall()` depending on the `style` argument. All additional arguments are forwarded unchanged to the selected function, so you can pass `alpha`, `beta`, `distance_mat`, HDI bounds, grouping factors, and so on in exactly the same way as you would when calling the underlying plotting functions directly.

Usage

```
interprofile(style = c("fountain", "waterfall"), ...)
```

Arguments

<code>style</code>	Character string choosing the layout. Accepts "fountain" (default) or "waterfall". Matching is case-insensitive and only the first few letters are required (e.g., "wat").
<code>...</code>	Further arguments passed on to either <code>pip_fountain()</code> or <code>pip_waterfall()</code> . See those functions for a full description of valid parameters.

Value

Whatever the chosen PIP function returns: a patchwork object that combines the two ggplot2 panels, invisibly returned after being printed.

See Also

- `pip_fountain()` - "base at $-\beta$, arrow up" style
- `pip_waterfall()` - "base at β , arrow down" style

Examples

```

# Small simulated example -----
set.seed(42)
N <- 6; I <- 10
alpha <- rnorm(N)
beta <- rnorm(I, sd = 0.7)
dist <- abs(matrix(rnorm(N * I, sd = 0.8), N, I)) # fake distances

# pip_profile() defaults to the fountain view
interprofile(alpha = alpha,
             beta = beta,
             distance_mat = dist,
             focal_id = 2)

# Switch to waterfall with the same data
interprofile("waterfall",
            alpha = alpha,
            beta = beta,
            distance_mat = dist,
            item_group = rep(LETTERS[1:2], length.out = length(beta)),
            y_limits=c(-3,2))

```

itemheatmap

Heat-map of item-item similarity in latent space

Description

Draws a lower-triangle heat-map (including the main diagonal) of the similarity between item positions in a latent space. Similarity is defined as

$$\exp(-\gamma d_{ij})$$

where d_{ij} is the Euclidean distance between items i and j , and $\gamma > 0$ is a scale parameter controlling how quickly similarity decays with distance. The function can optionally reorder items via hierarchical clustering so that similar items are placed next to one another, making block-structure easier to see.

Usage

```

itemheatmap(
  w,
  gamma = 1,
  item_names = NULL,
  reorder = FALSE,
  digits = 2,
  title = NULL
)

```

Arguments

w	Numeric matrix or data frame with one row per item and two (or more) columns giving the latent coordinates of each item.
gamma	Positive numeric scalar. Controls the steepness of the similarity decay; larger values make similarity drop off more quickly. Default is 1.
item_names	Optional character vector of item labels. Must have the same length as nrow(w). Defaults to "I1", "I2",... if NULL.
reorder	Logical. If TRUE (default is FALSE) the heat-map is reordered using hierarchical clustering of the distance matrix so that similar items are grouped along the diagonal.
digits	Integer. Number of decimal places used when printing similarity values inside the cells. Default is 2.
title	Optional character string for the plot title.

Value

(Invisibly) a ggplot object containing the heat-map. The plot is also displayed as a side effect.

Examples

```
set.seed(123)
w <- matrix(rnorm(40), ncol = 2) # 20 items in 2-D latent space

# Default heat-map
itemheatmap(w)

# Stronger decay (gamma = 3) and custom item names
itemheatmap(w, gamma = 3, item_names = paste("Item", 1:nrow(w)))

# Turn off re-ordering
itemheatmap(w, reorder = FALSE, title = "Fixed item ordering")
```

itemsimilarity *Similarity profile for a focal item*

Description

Plots the similarity between one **focal item** and every other item in latent space, optionally including posterior uncertainty bands when a list of draws is supplied. Similarity is defined as

$$\exp(-\gamma d_{ij})$$

where d_{ij} is the Euclidean distance between items i and j . Bars can be color-coded by a grouping factor, reordered by decreasing similarity, displayed horizontally or vertically, and annotated with credible intervals.

Usage

```

itemsimilarity(
  w,
  focal_item,
  gamma = 1,
  item_group = NULL,
  item_names = NULL,
  ci_level = 0.95,
  reorder = FALSE,
  vertical = TRUE,
  title = NULL,
  use_gradient = TRUE,
  gradient_low = "#d9f0d3",
  gradient_high = "#1b7837",
  show_gradient_legend = TRUE,
  single_fill_color = "steelblue"
)

```

Arguments

w	Numeric matrix ($I \times d$) of item coordinates or a list of such matrices (posterior draws). When a list is given the function summarises similarity across draws and plots medians with <code>ci_level</code> credible intervals.
focal_item	Index (integer) or name (character) of the item whose similarity profile is to be displayed.
gamma	Positive numeric scalar controlling how quickly similarity decays with distance. Default is 1.
item_group	Optional character/factor vector of length I indicating group membership for each item. Used for bar colors and legend.
item_names	Optional character vector of item labels (length I). Defaults to "I1", "I2", ... if NULL.
ci_level	Numeric between 0 and 1 giving the width of the credible interval when <code>w</code> is a posterior list. Ignored for a single draw.
reorder	Logical. Reorder items on the axis by decreasing similarity to the focal item? Default FALSE.
vertical	Logical. TRUE (default) plots vertical bars; FALSE flips the axes for a horizontal layout.
title	Optional character string added as the plot title.
use_gradient	Logical. When <code>item_group</code> is NULL, color bars by a similarity gradient (low->high). Default TRUE.
gradient_low, gradient_high	Colors for the similarity gradient when <code>use_gradient = TRUE</code> . Defaults "#d9f0d3" (low) to "#1b7837" (high).
show_gradient_legend	Logical. Show legend for the similarity gradient (only when <code>item_group</code> is NULL and <code>use_gradient = TRUE</code>)? Default TRUE.

`single_fill_color`

Single fill color when `use_gradient = FALSE` and `item_group` is `NULL`. Default "steelblue".

Value

(Invisibly) a ggplot object. The plot is also drawn as a side effect.

Examples

```
set.seed(1)
w <- matrix(rnorm(40), ncol = 2) # 20 items
gp <- sample(c("Math", "Verbal"), nrow(w), replace = TRUE)

## 1) Single estimate, default gradient (ungrouped)
itemsimilarity(w, focal_item = 3, gamma = 2,
               title = "Similarity to item 3 (gradient)")

## 2) Single estimate, turn off gradient and use one color (ungrouped)
itemsimilarity(w, focal_item = 3, gamma = 2,
               use_gradient = FALSE, single_fill_color = "tomato",
               title = "Similarity to item 3 (single color)")

## 3) Grouped bars (gradient ignored because groups are used)
itemsimilarity(w, focal_item = 3, gamma = 2, item_group = gp,
               title = "Similarity to item 3 (grouped)")

## 4) Posterior list with credible intervals (ungrouped, gradient)
draws <- replicate(100, w + matrix(rnorm(length(w), sd = 0.1),
                                   nrow(w), ncol(w)), simplify = FALSE)
itemsimilarity(draws, focal_item = "I10", ci_level = 0.9,
               vertical = FALSE, show_gradient_legend = FALSE)
```

lsirmicc

Latent-Space Item Characteristic Curve (ICC)

Description

Plots the LSIRM ICC for **one item** on a user-defined grid of ability values (`alpha_grid`). The function supports two ways of supplying inputs:

Usage

```
lsirmicc(
  item_id,
  posterior = NULL,
  beta = NULL,
  gamma = NULL,
```

```

w_pos = NULL,
z_pos = NULL,
alpha_grid = seq(-4, 4, length.out = 201),
person_id = NULL,
compare = TRUE,
credibleRibbon = FALSE,
cred_level = 0.95,
reference = c("item", "person-global", "origin"),
ref_col = "grey40",
person_cols = NULL
)

```

Arguments

item_id	Scalar index of the item to plot.
posterior	Optional list of draws with components beta $M \times I$ matrix of item intercepts. gamma Length-M vector of distance weights. w Either an $M \times I \times D$ array or a length-M list of $I \times D$ matrices of item coordinates. z Optional array/list of person coordinates (same format as w). Only needed if you request person_id or reference = "person-global".
beta, gamma	Numeric point estimates used only when posterior = NULL. beta may be scalar or length-I.
w_pos, z_pos	Matrices of point estimates for item ($I \times D$) and person ($N \times D$) coordinates, used only when posterior = NULL.
alpha_grid	Numeric vector of ability values (default seq(-4, 4, length.out = 201)).
person_id	NULL (no person curves) or integer vector of respondent indices to overlay.
compare	Logical. If TRUE (default) the reference curve is drawn in addition to any person curves; if FALSE only person curves appear.
credibleRibbon	Logical. Draw the credible ribbon for draws-based inputs? Ignored (forced FALSE) when posterior = NULL. Default FALSE to keep plots uncluttered.
cred_level	Width of the credible ribbon (e.g., 0.95).
reference	One of "item", "origin", or "person-global"; see Details.
ref_col	Colour for the reference curve.
person_cols	Optional vector of colours for person curves; recycled or auto-generated as needed.

Details

- **Point-estimate inputs (default)** - leave posterior = NULL and supply deterministic beta, gamma, w_pos, and (if needed) z_pos. A single curve per requested group is drawn (no ribbon).

- **Draws-based inputs** - supply a posterior list with draws of beta, gamma, w, and optionally z. The plotted curve is the **posterior-predictive mean probability** at each θ (i.e., average over draws). Optionally add a credible ribbon via `credibleRibbon = TRUE` with width `cred_level`.

The probability model is

$$P(Y_{ij} = 1 | \theta_j, d_{ij}) = \text{logit}^{-1}(\theta_j + \beta_i - \gamma d_{ij})$$

where $d_{ij} = \|z_j - w_i\|$. Choice of the reference position (`reference = "item", "origin", or "person-global"`) determines how d_{ij} is computed for the *baseline* (grey) curve.

Value

(Invisibly) a **ggplot2** object; the plot is displayed as a side-effect.

Curve types

- **Reference curve** - distance is computed from the chosen reference position to the item for every posterior draw (or once with point-estimate inputs). Shown unless `compare = FALSE`.
- **Person curve(s)** - distance is computed from the latent position(s) of respondent(s) listed in `person_id`. Requires z (posterior draws or point estimates).

Examples

```
## ---- reproducible demonstration -----
set.seed(1)
I <- 6; N <- 40; D <- 2; M <- 300          # toy dimensions

## 1. Point-estimate inputs (default) -----
beta_hat <- 0.3
gamma_hat <- 1.2
w_hat <- matrix(rnorm(I * D), I, D)
z_hat <- matrix(rnorm(N * D), N, D)

# population curve + one person (no ribbon in point-estimate usage)
lsirmicc(item_id = 4,
         beta     = beta_hat,
         gamma    = gamma_hat,
         w_pos    = w_hat,
         z_pos    = z_hat,
         person_id = 7)

## 2. Draws-based inputs (posterior list) -----
w_base <- matrix(0, I, D); w_base[, 1] <- seq(-1.2, 1.2, length.out = I)
z_base <- matrix(0, N, D); z_base[, 1] <- rep(c(-0.6, 0.6), length.out = N)

posterior <- list(
  beta = matrix(rnorm(M * I, 0, 0.25), M, I),
  gamma = rgamma(M, shape = 300, rate = 300),
  w     = array(rep(w_base, each = M), c(M, I, D)) +
    array(rnorm(M * I * D, sd = 0.12), c(M, I, D)),
```

```

z = array(rep(z_base, each = M), c(M, N, D)) +
      array(rnorm(M * N * D, sd = 0.12), c(M, N, D))
)

# posterior-predictive mean curve with ribbon and two people
lsirmicc(item_id = 2,
         posterior = posterior,
         person_id = c(22, 31),
         credibleRibbon = TRUE,
         cred_level = 0.95,
         person_cols = c("red", "blue"))

```

pip_fountain

Posterior Interaction Profile - Fountain style

Description

Generates the **fountain** variant of a Posterior Interaction Profile (PIP) plot. The layout is identical to `pip_waterfall()` on the left (posterior density for the focal respondent's ability) but **inverts** the right-hand panel: each item dot is placed at $-\beta_i$ (the "fountain base") and an arrow rises to the personalized easiness

$$\delta_{ij} = \beta_i - d_{ij}$$

. Distance is taken from `distance_mat`. If `gamma` is supplied, distances are scaled before computing deltas, i.e. $\delta_{ij} = \beta_i - \gamma d_{ij}$. Uncertainty bounds in `distance_low/distance_up` are scaled by the same γ . Arrows that extend **above** the base indicate the item is *easier* for the respondent than average; arrows that fall short indicate it is *harder*.

Usage

```

pip_fountain(
  alpha,
  beta,
  distance_mat,
  gamma = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  distance_low = NULL,
  distance_up = NULL,
  item_group = NULL,
  focal_id = 1,
  density_adjust = 2,
  y_limits = NULL
)

```

Arguments

alpha	Numeric vector of length N . Posterior means (or draws) of person ability parameters.
beta	Numeric vector of length I . Posterior means of item easiness parameters.
distance_mat	Numeric matrix $N \times I$ containing the latent distances d_{ij} between persons and items.
gamma	Optional numeric scalar used to multiplicatively rescale all distances (and distance_low/distance_up, if provided) before computing personalized easiness. Defaults to NULL (no rescaling).
alpha_lower, alpha_upper	Optional numeric vectors (length N) providing lower/upper posterior intervals (e.g., 95% HDI) for each respondent's α_j . The focal respondent's band is shaded.
distance_low, distance_up	Optional matrices matching distance_mat that give lower/upper HDI bounds for each distance. When both are supplied, dotted vertical lines depict the uncertainty in every personalized easiness.
item_group	Optional character/factor vector of length I defining item groupings. Enables color coding and a legend.
focal_id	Integer ($1 \leq \text{focal_id} \leq N$) selecting the respondent to highlight. Defaults to the first row.
density_adjust	Positive numeric scalar passed to <code>ggplot2::geom_density(adjust = ...)</code> to control the smoothness of the left-panel density estimate. Values > 1 increase the bandwidth (smoother curve); values < 1 decrease it (more detail). Default is 2.
y_limits	Optional numeric length-2 vector <code>c(min, max)</code> that fixes the y-axis range for both panels.

Value

A patchwork object containing the combined left- and right-hand ggplot2 panels. The plot is automatically displayed; the value is returned invisibly for further tweaking.

See Also

[pip_waterfall\(\)](#) for the alternative "waterfall" framing, and [interprofile\(\)](#) for a wrapper that switches between the two.

Examples

```
# Small simulated example -----
set.seed(42)
N <- 6; I <- 10
alpha <- rnorm(N)
beta <- rnorm(I, sd = 0.7)
dist <- abs(matrix(rnorm(N * I, sd = 0.8), N, I)) # fake distances
```

```
# Plain fountain plot for respondent 2
pip_fountain(alpha, beta, gamma = 1.5, dist, focal_id = 2)

# Fountain plot with item groups and uncertainty intervals
grp <- rep(c("MCQ", "Essay"), length.out = I)
d_lo <- pmax(dist - 0.2, 0); d_up <- dist + 0.2
a_lo <- alpha - 0.3; a_up <- alpha + 0.3
pip_fountain(alpha, beta, gamma = 1, dist,
             alpha_lower = a_lo, alpha_upper = a_up,
             distance_low = d_lo, distance_up = d_up,
             item_group = grp, focal_id = 4)
```

pip_waterfall

Posterior Interaction Profile - Waterfall style

Description

Creates the **waterfall** flavour of a Posterior Interaction Profile (PIP) plot, visualizing how a single respondent's latent position (α_p) interacts with every item. The left panel shows the posterior density (and optional HDI) of the chosen respondent's ability. The right panel ("waterfall") plots each item's easiness β_i as the starting point of a vertical arrow whose tip marks the personalized easiness $\delta_{ij} = \beta_i - d_{ij}$, where d_{ij} is the latent distance taken from `distance_mat`. If `gamma` is supplied, distances are scaled before computing deltas, i.e. $\delta_{ij} = \beta_i - \gamma d_{ij}$. Uncertainty bounds in `distance_low/distance_up` are scaled by the same γ . Arrows pointing **up** indicate the item is *easier* for the focal respondent than for the average person, whereas arrows pointing **down** indicate it is *harder*.

Usage

```
pip_waterfall(
  alpha,
  beta,
  distance_mat,
  gamma = NULL,
  alpha_lower = NULL,
  alpha_upper = NULL,
  distance_low = NULL,
  distance_up = NULL,
  item_group = NULL,
  focal_id = 1,
  density_adjust = 2,
  y_limits = NULL
)
```

Arguments

`alpha` Numeric vector of length N . Posterior means (or draws) of person ability parameters.

beta	Numeric vector of length I . Posterior means of item easiness parameters.
distance_mat	Numeric matrix $N \times I$ containing the latent distances d_{pi} between persons and items.
gamma	Optional numeric scalar used to multiplicatively rescale all distances (and distance_low/distance_up, if provided) before computing personalized easiness. Defaults to NULL (no rescaling).
alpha_lower, alpha_upper	Optional numeric vectors (length N) giving lower/upper bounds (e.g., 95% HDI) for each person's α_j . If provided, the focal respondent's interval is shaded in pink.
distance_low, distance_up	Optional matrices the same size as distance_mat providing lower/upper HDI bounds for the distances. When both are supplied, dotted lines visualize the uncertainty in each personalised easiness.
item_group	Optional character/factor vector of length I assigning
focal_id	Integer index ($1 \leq \text{focal_id} \leq N$) of the respondent to highlight. Default is 1.
density_adjust	Positive numeric scalar passed to <code>ggplot2::geom_density(adjust = ...)</code> to control the smoothness of the left-panel density estimate. Values > 1 increase the bandwidth (smoother curve); values < 1 decrease it (more detail). Default is 2.
y_limits	Optional numeric length-2 vector <code>c(min, max)</code> that fixes the y-axis range for both panels.

Value

A `patchwork` object containing two `ggplot2` panels. The plot is also displayed as a side effect, so the returned object is mainly for further customization.

See Also

[pip_fountain\(\)](#) for the complementary "fountain" layout, and [interprofile\(\)](#) for a thin wrapper that chooses between the two styles.

Examples

```
set.seed(42)
N <- 6; I <- 10
alpha <- rnorm(N)
beta <- rnorm(I, sd = 0.7)
dist <- abs(matrix(rnorm(N * I, sd = 0.8), N, I)) # fake distances

# Basic waterfall plot for the first respondent
pip_waterfall(alpha, beta, gamma = 1.5, dist, focal_id = 2)

# Add grouping and uncertainty bands
groups <- rep(c("A", "B"), length.out = I)
d_low <- dist * 0.9; d_up <- dist * 1.1
a_l <- alpha - 0.25; a_u <- alpha + 0.25
```

```

pip_waterfall(alpha, beta, gamma = 1, dist,
              alpha_lower = a_l, alpha_upper = a_u,
              distance_low = d_low, distance_up = d_up,
              item_group = groups, focal_id = 3)

```

radarplot

Radar plot of branch-specific abilities (single or multiple subjects)

Description

Draws a radar / spider chart in which each axis ("branch") represents a domain-specific ability and the radial extent marks the attained score. *Single-subject mode* shades the overall-ability circle and fills the polygon formed by the branch scores. *Multi-subject mode* overlays several polygons on a common background, optionally coloring, labelling, and annotating each subject.

Usage

```

radarplot(
  data = NULL,
  labels,
  max_radius = 100,
  branch_max = 10,
  overallAbility = NA,
  ability_range = c(-3, 3),
  abilityCutoffs = c(-1.68, 1.68),
  bgColors = c("red", "yellow", "green"),
  markerInd = NULL,
  point_cex = 4,
  subjectLabels = NULL,
  sampleColors = NULL,
  showOverallAbility = FALSE,
  title = NULL,
  plot_margin = margin(t = 20, r = 20, b = 20, l = 20),
  label_angle_offset = 0
)

```

Arguments

data	Numeric vector (one subject) or matrix/data-frame ($m \times n$) of scores, where $n = \text{length}(\text{labels})$ is the number of branches. Rows correspond to subjects when data is two-dimensional.
labels	Character vector of length n giving the label for each branch / axis.
max_radius	Numeric. Maximum drawing radius in plot units after the scores in <code>ability_range</code> have been linearly rescaled. Default 100.
branch_max	Numeric. Reserved for future branch-specific scaling. Currently ignored.

overallAbility	Numeric scalar or length- m vector giving the overall ability for each subject. Used to set the radius of the shaded background circle for that subject. If NA the maximum of ability_range is used.
ability_range	Numeric length-2 vector [min, max] defining the scale of the input scores. These limits are mapped to the interval [0, max_radius].
abilityCutoffs	Numeric. Reserved for future color gradations; not used in the current version.
bgColors	Character vector of colors for the shaded background circles. Only the first element is used at present.
markerInd	Numeric vector (0 = hollow, 1 = solid) of length n indicating the point style for each branch. When data is a matrix this can be supplied as an $m \times n$ matrix so each subject has its own marker pattern.
point_cex	Numeric point size for branch markers. Default 4.
subjectLabels	Optional character vector of length m naming each subject in multi-subject mode. Row names of data are used when available; otherwise "Subject 1", "Subject 2", ... are generated.
sampleColors	Character vector of length m giving the polygon/point color for each subject. Defaults to a distinct hue palette if NULL.
showOverallAbility	Logical. If TRUE, prints each subject's overall ability beneath their label.
title	Optional plot title.
plot_margin	A ggplot2::margin() object controlling the outer whitespace around the figure. Default adds 20 pt on every side.
label_angle_offset	Numeric scalar or length- n vector specifying (in degrees) how much to rotate each branch label relative to its default tangential orientation. Useful for fine-tuning readability.

Value

A ggplot object representing the radar chart (also printed as a side effect).

Examples

```
## Single subject -----
#### the distance from a person to all items/item clusters
dist_z_w <- c(item1 = 1.6, item2 = 0.8, item3 = 1.9, item4 = 2.5, item5 = 0.4)
#### transform distance to strength
strength <- exp(-dist_z_w)
#### plot the radar
radarplot(strength, labels = names(strength),
          overallAbility = 1.8, showOverallAbility = TRUE,
          title = "Student A profile")

## Multiple subjects -----
set.seed(1)
#### strength for 3 persons on 5 items
dat <- matrix(rnorm(15, 3, 1), nrow = 3,
```

```

      dimnames = list(NULL, c("item1", "item2", "item3", "item4", "item5"))
radarplot(dat, labels = colnames(dat),
          overallAbility = c(-1.8, 0.5, 2.5),
          subjectLabels = c("Alice", "Bob", "Cara"),
          sampleColors = c("#1b9e77", "#d95f02", "#7570b3"),
          showOverallAbility = TRUE,
          title = "Class-level comparison")

```

rescale_to_range *Rescale a numeric vector to a new range*

Description

Linearly transforms the values in `x` so they fall within a specified interval. Useful, for example, when mapping latent-space coordinates to aesthetic ranges (point sizes, color scales, etc.) in a plot.

Usage

```
rescale_to_range(x, to = c(0, 1), na.rm = TRUE)
```

Arguments

<code>x</code>	Numeric vector. The data to be rescaled.
<code>to</code>	Numeric vector of length 2 giving the lower and upper limits of the target range. Defaults to <code>c(0, 1)</code> .
<code>na.rm</code>	Logical. Should missing values be ignored when computing the source range? Defaults to <code>TRUE</code> . Any NAs in <code>x</code> are returned unchanged.

Details

If all non-missing values in `x` are identical, the function returns the midpoint of the target range (`mean(to)`) for those elements to avoid division by zero.

Value

A numeric vector the same length as `x`, with values rescaled to lie within `to[1]` and `to[2]`. The function preserves the positions of NAs.

Examples

```

set.seed(123)
x <- rnorm(5)

# Default 0~1 range
rescale_to_range(x)

# Rescale to -1~1

```

```

rescale_to_range(x, to = c(-1, 1))

# Preserve NAs but ignore them when determining the range
x_with_na <- c(x, NA, 10)
rescale_to_range(x_with_na, to = c(0, 100))

```

strengthplot

Item-strength profile for a single person

Description

For a chosen respondent (`person_index`) this function plots the **strength** (likelihood of endorsement) for every item, defined as

$$\exp(-\gamma d_{ij})$$

, where d_{ij} is the Euclidean distance between the person's latent position z_j and each item position w_i . When z and w are supplied as *lists* of matrices (posterior draws), the function summarizes the distribution of strengths with medians and a `ci_level` credible interval. Bars can be colored by an item grouping factor, reordered by decreasing strength, and displayed either vertically or horizontally.

Usage

```

strengthplot(
  z,
  w,
  person_index,
  gamma = 1,
  item_group = NULL,
  item_names = NULL,
  ci_level = 0.95,
  reorder = FALSE,
  vertical = TRUE,
  title = NULL,
  use_gradient = TRUE,
  gradient_low = "#d9f0d3",
  gradient_high = "#1b7837",
  show_gradient_legend = TRUE,
  single_fill_color = "steelblue"
)

```

Arguments

`z` A numeric matrix ($N \times d$) of person coordinates **or** a *list* of such matrices representing posterior draws.

`w` A numeric matrix ($I \times d$) of item coordinates **or** a *list* of such matrices, matching the structure of `z`.

person_index	Integer giving the row of z (or each draw in z) corresponding to the focal respondent.
gamma	Positive numeric scalar controlling the decay of strength with distance; default is 1.
item_group	Optional character/factor vector of length <i>I</i> assigning each item to a group for color coding and legend.
item_names	Optional character vector of item labels. If NULL defaults to "I1", "I2",...
ci_level	Width of the credible interval (between 0 and 1) when posterior draws are given. Ignored for a single point estimate.
reorder	Logical. Reorder items on the axis by decreasing strength? Default FALSE.
vertical	Logical. TRUE (default) draws vertical bars; FALSE flips the axes for a horizontal layout.
title	Optional character string to appear as the plot title.
use_gradient	Logical. When item_group is NULL, color bars by a strength gradient (low -> high)? Default TRUE.
gradient_low, gradient_high	Colors for the gradient when use_gradient = TRUE. Defaults "#d9f0d3" (low) to "#1b7837" (high).
show_gradient_legend	Logical. Show a legend for the gradient (only when item_group is NULL and use_gradient = TRUE)? Default TRUE.
single_fill_color	Single fill color used when use_gradient = FALSE and item_group is NULL. Default "steelblue".

Details

When no `item_group` is provided, bars are color-mapped by a similarity gradient (low -> high) by default. You can disable this behavior and use a single fill color instead via `use_gradient = FALSE`.

Value

(Invisibly) a ggplot object containing the bar plot. The plot is also printed.

Examples

```
set.seed(1)
z <- matrix(rnorm(40), ncol = 2) # 20 persons
w <- matrix(rnorm(30), ncol = 2) # 15 items

## 1) Point-estimate strengths for person 5 (default gradient, ungrouped)
strengthplot(z, w, person_index = 5, gamma = 2,
             title = "Strengths for person 5 (gradient)")

## 2) Turn off gradient and use a single color
strengthplot(z, w, person_index = 5, gamma = 2,
             use_gradient = FALSE, single_fill_color = "tomato",
```

```

title = "Strengths for person 5 (single color)")

## 3) Posterior example with credible intervals and item groups
draws_z <- replicate(50, z + matrix(rnorm(length(z), sd = 0.1),
                                   nrow(z), ncol(z)), simplify = FALSE)
draws_w <- replicate(50, w + matrix(rnorm(length(w), sd = 0.1),
                                   nrow(w), ncol(w)), simplify = FALSE)
grp <- rep(c("Core", "Peripheral"), length.out = nrow(w))
strengthplot(draws_z, draws_w, person_index = 3,
             item_group = grp, ci_level = 0.9, vertical = FALSE,
             title = "Posterior strength profile for respondent 3")

```

vec_mat_dist

Euclidean distance from a single vector to each row of a matrix

Description

Calculates the Euclidean distance between a reference vector v and every row of a matrix mat . This is a thin wrapper around `rowSums()` and avoids an explicit loop, so it is fast even for large matrices.

Usage

```
vec_mat_dist(v, mat)
```

Arguments

<code>v</code>	Numeric vector of length d . The reference point in d -dimensional space.
<code>mat</code>	Numeric matrix with n rows and d columns. Each row is treated as a point whose distance from v is to be computed. The number of columns in mat must match <code>length(v)</code> .

Details

Internally the function replicates v into an $n \times d$ matrix, subtracts it from mat , squares the element-wise differences, sums across columns, and finally takes the square root, i.e.

$$d_i = \sqrt{\sum_{k=1}^d (m_{ik} - v_k)^2}$$

for each row i . Because the computation is fully vectorised it is considerably faster than a simple `apply()` or a for-loop implementation.

Value

A numeric vector of length n containing the Euclidean distance between v and each corresponding row of mat .

Examples

```
# Two-dimensional example
v <- c(0, 0)
mat <- matrix(c(1, 0,
               0, 2,
               3, 4),
              ncol = 2, byrow = TRUE)

vec_mat_dist(v, mat)
#> [1] 1 2 5
```

Index

crossdist_fast, [2](#)

iccsurface, [3](#)
intermap2d, [5](#)
interprofile, [11](#)
interprofile(), [19](#), [21](#)
itemheatmap, [12](#)
itemsimilarity, [13](#)

lsirmicc, [15](#)

pip_fountain, [18](#)
pip_fountain(), [11](#), [21](#)
pip_waterfall, [20](#)
pip_waterfall(), [11](#), [18](#), [19](#)

radarplot, [22](#)
rescale_to_range, [24](#)

strengthplot, [25](#)

vec_mat_dist, [27](#)