Package: ThinkingGrid (via r-universe)

May 13, 2026

Title Tools for Thinking Grid Statistics

Version 0.1.0

Author Avinash Kulkarni [aut, cph], Vishal Kuvar [aut, cre], Zac Irving [ctb, fnd]

Maintainer Vishal Kuvar <kuvar001@umn.edu>

Description Provides comprehensive tools for conducting research using the Thinking Grid framework, a psychological measurement approach for understanding deliberate and automatic cognitive constraints. Includes functions for generating 'Qualtrics' surveys, processing survey responses, calculating quadrant depths, and creating various visualization types including heatmaps, animations, and statistical plots.

URL https://a-kulkarn.github.io/qualtrics-thinkgrid/

BugReports https://github.com/a-kulkarn/qualtrics-thinkgrid/issues

Imports colorspace, cowplot, ggplot2, gifski, grid, gridExtra, methods, reticulate (>= 1.40.0), scales, stats, utils

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests png, testthat (>= 3.0.0), vdiffr, lme4, knitr, rmarkdown, sjPlot

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libpng-dev python3 libclang-dev

Repository https://a-kulkarn.r-universe.dev

Date/Publication 2025-09-23 22:18:35 UTC

RemoteUrl https://github.com/a-kulkarn/qualtrics-thinkgrid

RemoteRef HEAD

RemoteSha 1348f3e9f7d831ccd4bbb341458671d0f0856fbd

RemoteSubdir ThinkingGrid

Contents

`add_depths` *Adds taxicab metric calculation columns to a dataframe.*

Description

Adds taxicab metric calculation columns to a dataframe.

Usage

```
add_depths(data, dc = "Deliberate.Constraints", ac = "Automatic.Constraints")
```

Arguments

`data`	data.frame, needed Data frame containing columns for deliberate constraints and automatic constraints.
`dc`	character, optional Name of the column containing deliberate constraints. Default is "Deliberate.Constraints".
`ac`	character, optional Name of the column containing automatic constraints. Default is "Automatic.Constraints".

Details

The function calculates the quadrant depths based on the deliberate and automatic constraints provided in the data file. The quadrant depths are calculated using the taxicab norm. Only one depth will be populated per observation, depending on the quadrant the observation falls into. The remaining three depths will be set to 0.

Value

data frame containing the quadrant depths. Columns include "sticky", "hybrid", "free", "directed", "total_depth", and "quadrant". The value of quadrant is 1-4, corresponding to top-left, top-right, bottom-left, bottom-right.

Examples

```
# Calculate quadrant depths from survey data
data_file <- system.file("extdata", "sample_data.csv", package = "ThinkingGrid")
if (file.exists(data_file)) {
  data_file <- read.csv(data_file)
  depth_results <- add_depths(data_file, dc = "dc", ac = "ac")
}
```

`check_python_available` *Checks status of Python dependency.*

Description

Checks if python is available. Installs python if appropriate flag is set.

Usage

```
check_python_available(install_if_NA = FALSE)
```

Arguments

`install_if_NA`	logical, optional If TRUE, installs python 3.13 and required packages. Default is FALSE.

Value

None

Examples

```
## Not run:
check_python_available()
check_python_available(install_if_NA = TRUE)

## End(Not run)
```

`create_custom_colorer` *Create custom color scale with enhanced visibility for small values*

Description

Create custom color scale with enhanced visibility for small values

Usage

```
create_custom_colorer(
  palette = "RdYlBu",
  zero_color = "#FFFFFF",
  n_colors = 11,
  gradient_scaling = "enhanced",
  enhanced_threshold_pct = 50,
  enhanced_expansion_factor = 1.5
)
```

Arguments

`palette`	Name of diverging palette from colorspace package (default: "RdYlBu"). Other options include "RdBu", "PiYG", "BrBG", "PuOr", "RdGy".
`zero_color`	Color for zero values in hex (default: "#FFFFFF" - white for clear distinction)
`n_colors`	Number of color steps (default: 11)
`gradient_scaling`	Type of gradient scaling: "linear" or "enhanced" (default: "enhanced"). "linear" uses standard color mapping. "enhanced" gives more color distinction to smaller values.
`enhanced_threshold_pct`	Percentage of maximum value to use as threshold for enhanced scaling (default: 50). Values below this percentage get more color distinction.
`enhanced_expansion_factor`	Factor controlling how much more color distinction small values get (default: 1.5). Higher values mean more distinction for small differences. Only used when gradient_scaling = "enhanced".

Details

The enhanced scaling works by compressing small values in the scaled space, which gives them more colors in the final gradient. For example, if your data ranges from 0-20 percent and you use enhanced_threshold_pct = 50 with enhanced_expansion_factor = 2.0, then values 0-10 percent will get twice as much color distinction compared to linear scaling.

Value

A colorer function that can be passed to plot_tg() or create_tg_animation()

Examples

```
# Create dummy data for testing
dummy_data <- data.frame(
  Deliberate.Constraints = sample(1:6, 100, replace = TRUE),
  Automatic.Constraints = sample(1:6, 100, replace = TRUE)
)

# Basic usage with linear scaling
colorer <- create_custom_colorer(gradient_scaling = "linear")

# Enhanced scaling for better small value distinction
colorer_enhanced <- create_custom_colorer(
  gradient_scaling = "enhanced",
  enhanced_threshold_pct = 30,
  enhanced_expansion_factor = 2.0
)
```

---

create_tg_animation *Create Animated Thinking Grid Visualizations*

Description

Generate animated GIF showing Thinking Grid plots across different conditions or time points.

Usage

```
create_tg_animation(
  survey_results,
  dc_column = "Deliberate.Constraints",
  ac_column = "Automatic.Constraints",
  condition_column = NULL,
  type = "depth",
  proportion_type = "overall",
  colorer = NULL,
  palette = "RdYlBu",
  zero_color = "#FFFFFF",
  gradient_scaling = "linear",
  enhanced_threshold_pct = 50,
  enhanced_expansion_factor = 1.5,
  x_label = "Directedness",
  y_label = "Stickiness",
  max_legend = NULL,
  min_legend = NULL,
  plot_title = NULL,
  legend_title = NULL,
  plot_subtitle = NULL,
  filename = "tg_animation.gif",
```

```
  duration = 1,
  width = 800,
  height = 800,
  sorted_conditions = NULL,
  subset_condition = NULL
)
```

Arguments

survey_results	Data frame containing survey results with constraint response columns.
dc_column	Name of deliberate constraints column (default: "Deliberate.Constraints").
ac_column	Name of automatic constraints column (default: "Automatic.Constraints").
condition_column	Column name containing conditions for animation frames. Each unique value becomes one frame.
type	Type of visualization (default: "depth"). See plot_tg for options.
proportion_type	Currently only "overall" is supported for animations.
colorer	Custom colorer function. If NULL, uses default based on other parameters.
palette	Color palette (default: "RdYlBu"). See plot_tg for options.
zero_color	Color for zero values (default: "#FFFFFF").
gradient_scaling	Scaling method: "linear" (default) or "enhanced".
enhanced_threshold_pct	Enhanced scaling threshold percentage (default: 50).
enhanced_expansion_factor	Enhanced scaling expansion factor (default: 1.5).
x_label	X-axis label (default: "Directedness").
y_label	Y-axis label (default: "Stickiness").
max_legend	Maximum legend value. If NULL, calculated from all conditions.
min_legend	Minimum legend value. If NULL, calculated from all conditions.
plot_title	Main title appearing on all frames.
legend_title	Legend title.
plot_subtitle	Subtitle(s). Can be single string (same subtitle for all frames) or vector (one subtitle per condition in same order as sorted_conditions if provided).
filename	Output GIF filename (default: "tg_animation.gif").
duration	Duration of each frame in seconds (default: 1).
width	GIF width in pixels (default: 800).
height	GIF height in pixels (default: 800).
sorted_conditions	Vector specifying frame order. Must contain all unique values from condition_column. If NULL, numeric conditions are sorted in ascending order and character/factor conditions are in random order (with warning).

`subset_condition`
: R expression string for subsetting data before analysis. Applied before splitting by conditions.

Details

The function creates one frame per unique value in condition_column. All frames use the same legend scale (calculated from all conditions) to ensure comparability across frames. Requires the 'gifski' package for GIF creation. Will prompt to install if missing.

Value

Invisibly returns list of ggplot objects (one per frame). The GIF file is saved to disk.

Examples

```
## Create sample data with time points for animation
set.seed(123)  ## For reproducible examples
survey_data <- data.frame(
  Deliberate.Constraints = sample(1:6, 300, replace = TRUE),
  Automatic.Constraints = sample(1:6, 300, replace = TRUE),
  time_point = rep(1:3, each = 100),
  week = rep(1:3, each = 100),
  completed_training = sample(c(TRUE, FALSE), 300, replace = TRUE)
)


## Basic animation across time points (creates temporary GIF file)
temp_gif1 <- tempfile(fileext = ".gif")
create_tg_animation(survey_data,
                    condition_column = "time_point",
                    filename = temp_gif1)
## Clean up temporary file
if (file.exists(temp_gif1)) file.remove(temp_gif1)

## Enhanced scaling for small differences
temp_gif2 <- tempfile(fileext = ".gif")
create_tg_animation(survey_data,
                    condition_column = "week",
                    gradient_scaling = "enhanced",
                    enhanced_threshold_pct = 40,
                    enhanced_expansion_factor = 2.0,
                    subset_condition = "completed_training == TRUE",
                    filename = temp_gif2)
## Clean up temporary file
if (file.exists(temp_gif2)) file.remove(temp_gif2)
```

`default_inner_theme` *Default theme for subplots in a ThinkingGrid 2x2 quadrant plot.*

Description

The default theme settings for the subplots of a 2x2 quadrant plot. In order to display correctly, the important properties of the theme are:

1. The background are set to transparant elements.
2. The main plot titles and legends are removed. (Axis labels are kept.)
3. The aspect ratio is set to 1.00.
4. Margins are adjusted to set the plot within the background squares.

Usage

```
default_inner_theme(inner_margin = 20)
```

Arguments

`inner_margin`	integer, optional Controls the padding between the inner subplots.

Value

A theme object with the settings for the subplots of a 2x2 quadrant plot.

Examples

```
plots <- ThinkingGrid:::create_test_2x2_plots()
p1 <- plots[[1]]
p2 <- plots[[2]]
p3 <- plots[[3]]
p4 <- plots[[4]]

# Here we can customize the theme.
itheme <- default_inner_theme(inner_margin = 10)

# This is the usual syntax.
thinkgrid_quadrant_plot(p1, p2, p3, p4, inner_theme = itheme)
```

generate_survey *Creates a Qualtrics importable survey file from a CSV file of questions.*

Description

Creates a Qualtrics importable survey file from a CSV file of questions.

Usage

```
generate_survey(
  survey_setup_file,
  output_file_name = "output_survey",
  question_text = TRUE
)
```

Arguments

survey_setup_file

character, required Path to a csv file containing the survey setup. This file MUST have a column called "id". Each row in this column should be unique. Indivudial thinking grids will be created for each row in this column. The other column is called "question". This column contains the question text. Quotes around question text is not required. If the question text is not provided, the function will use default text. Please note that these columns are case sensitive.

File setup for csv file without question text. For this to work the question_text parameter should be set to FALSE. A placeholder text ("Insert text here.") will be used in this case.:

id

ThinkingGrid1

ThinkingGrid2

ThinkingGrid3

File setup for csv file with question text:

id,question ThinkingGrid1,Report your thoughts on the thinking grid 1 ThinkingGrid2,Report your thoughts on the thinking grid 2 ThinkingGrid3,Report your thoughts on the thinking grid 3

The above file will create 3 thinking grids with the corresponding questions.

output_file_name

character, optional Name of the output qsf file. Default is "output_survey". The extension qsf will be added automatically. If the desired file name is "my_qsf_output.qsf, the function should be called as generate_survey("path/to/setup/survey_setup_file.csv", "path/to/output/my_qsf_output"). Default name is "output_survey.qsf". If a file with the same name exists, it will be overwritten.

question_text logical, optional If TRUE, the function will use the question text provided in the csv file. If FALSE, the function will use a default question text ("Insert text here."). Default is TRUE.

Value

None

Examples

```
# Generate survey from sample setup file
setup_file <- system.file("extdata", "sample_setup_file.csv", package = "ThinkingGrid")
if (file.exists(setup_file)) {
  generate_survey(setup_file, "_temp_output_")
  # Clean up
  file.remove("_temp_output_-0.qsf")
}
```

`install_thinkgrid` *Package setup.*

Description

Creates a virtual environment and installs the required python packages

Usage

```
install_thinkgrid(envname = "r-thinkgrid")
```

Arguments

envname	character, optional Name of the virtual environment to be created. Default is "r-thinkgrid".

Value

None

Examples

```
## Not run:
install_thinkgrid()
install_thinkgrid("new_environment_name")

## End(Not run)
```

plot_tg *Create Thinking Grid Visualizations*

Description

Generate various types of Thinking Grid plots from survey data containing deliberate and automatic constraint responses.

Usage

```
plot_tg(
  survey_results,
  proportion_type = "overall",
  type = "depth",
  colorer = NULL,
  palette = "RdYlBu",
  zero_color = "#FFFFFF",
  gradient_scaling = "linear",
  enhanced_threshold_pct = 50,
  enhanced_expansion_factor = 1.5,
  x_label = "Directedness",
  y_label = "Stickiness",
  dc_column = "Deliberate.Constraints",
  ac_column = "Automatic.Constraints",
  condition_column = NULL,
  comparison_type = "separate",
  max_legend = NULL,
  min_legend = NULL,
  plot_title = NULL,
  legend_title = NULL,
  plot_subtitle = NULL,
  subset_condition = NULL
)
```

Arguments

survey_results	Data frame containing survey results with constraint response columns.
proportion_type	Type of proportion calculation: "overall" (default) for single plot showing overall response patterns, or "condition" for separate plots for different conditions (requires condition_column).
type	Type of visualization: "depth" (default) shows distance from grid center in each quadrant, "cells" shows individual cell heatmap (6x6 grid), "quadrants" shows four-quadrant summary, "horizontal" shows horizontal bands (stickiness levels), "vertical" shows vertical bands (directedness levels), "constraints" shows diagonal constraint bands.

`colorer` Custom colorer function. If NULL, uses default based on other parameters. Create with create_custom_colorer().

`palette` Color palette name from colorspace package (default: "RdYlBu"). Options: "RdBu", "PiYG", "BrBG", "PuOr", "RdGy".

`zero_color` Color for zero values (default: "#FFFFFF").

`gradient_scaling` Color gradient scaling method: "linear" (default) for standard linear color mapping, or "enhanced" for more color distinction to smaller values.

`enhanced_threshold_pct` For enhanced scaling: percentage of max value used as threshold (default: 50). Values below this get enhanced distinction.

`enhanced_expansion_factor` For enhanced scaling: factor controlling how much more distinction small values get (default: 1.5). Higher values mean more distinction.

`x_label` X-axis label (default: "Directedness").

`y_label` Y-axis label (default: "Stickiness").

`dc_column` Name of deliberate constraints column (default: "Deliberate.Constraints").

`ac_column` Name of automatic constraints column (default: "Automatic.Constraints").

`condition_column` Column name for grouping conditions. Required when proportion_type = "condition".

`comparison_type` For condition plots: "separate" (default) for side-by-side plots for 2 conditions or separate plots for 3+, or "difference" for difference plot (condition1 - condition2, requires exactly 2 conditions).

`max_legend` Maximum legend value. If NULL, calculated from data.

`min_legend` Minimum legend value. If NULL, calculated from data.

`plot_title` Main plot title.

`legend_title` Legend title. If NULL, uses "Percentage (percent)" or "Difference (percent)".

`plot_subtitle` Plot subtitle. For condition plots with comparison_type = "separate", provide a vector with one subtitle per condition.

`subset_condition` R expression string for subsetting data before analysis. Uses standard R syntax. All referenced columns must exist in survey_results.

Details

The function expects constraint responses on a 1-6 scale where deliberate constraints (x-axis) range from 1 = low directedness to 6 = high directedness, and automatic constraints (y-axis) range from 1 = low stickiness to 6 = high stickiness.

For enhanced scaling, the algorithm compresses small values in the color space to give them more visual distinction. This is useful when most responses are in lower ranges.

Value

A list containing: plot (ggplot object or list of ggplot objects for 3+ conditions) and prop_data (calculated proportion data).

Examples

```
## Create sample data for testing
set.seed(123)  ## For reproducible examples
survey_data <- data.frame(
  Deliberate.Constraints = sample(1:6, 100, replace = TRUE),
  Automatic.Constraints = sample(1:6, 100, replace = TRUE),
  treatment_group = sample(c("A", "B"), 100, replace = TRUE),
  age = sample(20:60, 100, replace = TRUE),
  experience = sample(1:5, 100, replace = TRUE)
)

## Basic overall plot
result1 <- plot_tg(survey_data)

## Cell-level heatmap with enhanced scaling for small values
result2 <- plot_tg(survey_data,
                   type = "cells",
                   gradient_scaling = "enhanced",
                   enhanced_threshold_pct = 30,
                   enhanced_expansion_factor = 2.0)

## Compare conditions side by side
result3 <- plot_tg(survey_data,
                   proportion_type = "condition",
                   condition_column = "treatment_group",
                   comparison_type = "separate")

## Subset data before analysis
result4 <- plot_tg(survey_data,
                   subset_condition = "age > 25 & experience >= 2",
                   type = "quadrants")
```

---

read_qualtrics_data *Parses Qualtrics survey output into a dataframe.*

Description

Parses Qualtrics survey output into a dataframe.

Usage

```
read_qualtrics_data(data_file, setup_file)
```

Arguments

data_file	character, needed The path to where the Qualtrics output is located. This should be a csv file that needs to be provided to this function UNEDITED.
setup_file	character, needed setup_file used to generate the survey. This file should be the same file that was used to generate the survey. This file should have the same format as the survey_setup_file used in the generate_survey function.

Value

data frame containing the Qualtrics data. Columns include "uid", "Probe.Identifier", "Deliberate.Constraints", "Automatic.Constraints"

uid: Unique identifier for each participant. This correspons to the row number in the Qualtrics output file.

Probe.Identifier: Identifier for the probe. This is the same as the "id" column in the setup_file.

Deliberate.Constraints: The deliberate constraints provided by the participant. (X-axis on the thinking grid)

Automatic.Constraints: The automatic constraints provided by the participant. (Y-axis on the thinking grid)

Examples

```
# Read Qualtrics survey data
setup_file <- system.file("extdata", "sample_setup_file.csv", package = "ThinkingGrid")
data_file <- system.file("extdata", "sample_qualtrics_output.csv", package = "ThinkingGrid")
if (file.exists(setup_file) && file.exists(data_file)) {
  survey_data <- read_qualtrics_data(data_file, setup_file)
}
```

---

thinkgrid_quadrant_background

Creates the background for ThinkingGrid plots.

Description

Creates the background for ThinkingGrid plots.

Usage

```
thinkgrid_quadrant_background(
  arrowwidth = 1,
  xlab = "Executive Control",
  ylab = "Salience"
)
```

Arguments

arrowwidth	integer, optional The width of the arrow objects.
xlab	character, optional Label for the x-axis.
ylab	character, optional Label for the y-axis.

Value

A plot containing the 6x6 grid background image for the 2x2 thinkgrid plots which also contains the xy-axis arrows and labels.

---

thinkgrid_quadrant_plot

Creates a 2x2 plot grid overlayed on top of a ThinkingGrid background.

Description

Creates a 2x2 quadrant plot with four ggplot objects.

Usage

```
thinkgrid_quadrant_plot(
  p_sticky,
  p_salience,
  p_free,
  p_directed,
  inner_theme = NULL,
  arrowwidth = 1,
  xlab = "Executive Control",
  ylab = "Salience"
)
```

Arguments

p_sticky	(ggplot or rastergrob) A ggplot object for the "Sticky" quadrant.
p_salience	(ggplot or rastergrob) A ggplot object for the "Salience" quadrant.
p_free	(ggplot or rastergrob) A ggplot object for the "Free" quadrant.
p_directed	(ggplot or rastergrob) A ggplot object for the "Directed" quadrant.
inner_theme	(theme, optional) A theme for the inner subplots. See default_inner_theme for more details.
arrowwidth	(integer, optional) Controls the thickness of the axis arrows.
xlab	(character, optional) Label for the x-axis.
ylab	(character, optional) Label for the y-axis.

Value

A ggplot object (created via cowplot) which consists of the thinking grid background and the inlayed 2x2 subplots (or images) corresponding to the respective quadrant.

Examples

```
# This is a list of four plots generated by some regression. Their exact nature is not
# important aside from the fact that they are ggplot objects.
plots <- ThinkingGrid:::create_test_2x2_plots()
p1 <- plots[[1]]
p2 <- plots[[2]]
p3 <- plots[[3]]
p4 <- plots[[4]]

# This is the usual syntax.
thinkgrid_quadrant_plot(p1, p2, p3, p4)


# However, it is possible to use images as the subplots. IDEALLY, one uses an image
# type without backgrounds such as a PNG.
img_path <- system.file("extdata", "rabbiduck.png", package = "ThinkingGrid")
rabbi <- png::readPNG(img_path)
rabbigrob <- grid::rasterGrob(rabbi)  # Note you must raster the image!

# Create sample plots for demonstration
plots <- ThinkingGrid:::create_test_2x2_plots()
thinkgrid_quadrant_plot(p1, p2, p3, rabbigrob)
```

Index