Package: cld (via r-universe)

May 13, 2026

Title Create Compact Letter Display (CLD) for Statistical Comparisons

Version 0.0.1

Description Creates compact letter displays (CLDs) for pairwise comparisons from statistical post-hoc tests. Groups sharing the same letter are not significantly different from each other. Supports multiple input formats including results from 'stats' pairwise tests, 'DescTools', 'PMCMRplus', 'rstatix', symmetric matrices of p-values, and data frames. Provides a consistent interface for visualizing statistical groupings across different testing frameworks.

License GPL (>= 3)

URL https://gegznav.github.io/cld/

BugReports https://github.com/GegznaV/cld/issues

Imports checkmate, multcompView, rlang

Suggests DescTools, dplyr, ggplot2, knitr, PMCMR, PMCMRplus, rcompanion, rmarkdown, rstatix, testthat (>= 3.0.0), tibble

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 2.10)

VignetteBuilder knitr

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

Date/Publication 2025-11-10 19:57:34 UTC

RemoteUrl https://github.com/gegznav/cld

RemoteRef HEAD

RemoteSha 9a7fd17b96c8ca2665e080f8bb3d11ae5ac1dbb0

Contents

`as_cld` *Convert objects to cld_object*

Description

Generic function to convert various objects to `cld_object` format. This is useful when you have CLD results from other packages or custom formats and want to use them with cld's methods.

Usage

```
as_cld(x, ...)

## S3 method for class 'data.frame'
as_cld(x, ...)

## S3 method for class 'character'
as_cld(x, ...)

## S3 method for class 'cld_object'
as_cld(x, ...)
```

Arguments

`x`	Object to convert to `cld_object`
`...`	Additional arguments passed to methods

Details

The `as_cld()` function provides a way to convert various formats to the standard `cld_object` structure. This is particularly useful when:

- Working with CLD results from other packages
- Converting custom formats to use cld's print and coercion methods
- Creating CLD objects programmatically

Value

A `cld_object` (data frame with class `c("cld_object", "data.frame")`) containing three columns:

- `group` - Character. The names of the groups
- `cld` - Character. The compact letter display for each group
- `spaced_cld` - Character. A monospaced version of the CLD where spaces are replaced with underscores, useful for maintaining alignment in output

See Also

- `make_cld()` for creating compact letter displays
- `pval_matrix_to_df()` for converting p-value matrices
- cld_methods for other cld_object methods

Other helper functions: `pval_matrix_to_df()`

Other data conversion functions: `pval_matrix_to_df()`

Examples

```
# Convert from data frame
df <- data.frame(
  group = c("A", "B", "C"),
  cld = c("a", "b", "b")
)
as_cld(df)

# Convert from named character vector
letters_vec <- c(A = "a", B = "b", C = "b")
as_cld(letters_vec)
```

---

cld_methods *Print and coercion methods for cld_object*

Description

Print and coercion methods for cld_object

Usage

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

## S3 method for class 'cld_object'
as.data.frame(x, ...)

## S3 method for class 'cld_object'
as.character(x, ...)
```

```
## S3 method for class 'cld_object'
as_tibble(x, ...)
```

Arguments

x	A cld_object to print or convert
...	Additional arguments passed to print methods

Value

- print.cld_object(): Invisibly returns the input object
- as.data.frame.cld_object(): Returns a plain data frame
- as.character.cld_object(): Returns a named character vector of letters
- as_tibble.cld_object(): Returns a tibble

See Also

- make_cld() for creating compact letter displays
- as_cld() for converting objects to cld_object

Examples

```
obj <- pairwise.wilcox.test(chickwts$weight, chickwts$feed, exact = FALSE)
result <- make_cld(obj)

# Print method
print(result)

# Convert to plain data frame
as.data.frame(result)

# Convert to named character vector
as.character(result)

# Convert to tibble
if (requireNamespace("tibble", quietly = TRUE)) {
  tibble::as_tibble(result)
}
```

---

make_cld *Make a compact letter display (CLD) for pair-wise comparisons*

Description

Creates a compact letter display for results of pair-wise comparisons (e.g., ANOVA post-hoc tests, Kruskal-Wallis post-hoc tests, and others). Groups that share the same letter are not significantly different from each other. This provides a visual summary of which groups differ significantly, commonly used in agricultural, biological, and statistical reporting.

Usage

```
make_cld(obj, ..., alpha = 0.05)

## S3 method for class 'pairwise.htest'
make_cld(obj, ..., alpha = 0.05)

## S3 method for class 'PMCMR'
make_cld(obj, ..., alpha = 0.05)

## S3 method for class 'PostHocTest'
make_cld(obj, ..., alpha = 0.05)

## S3 method for class 'DunnTest'
make_cld(obj, ..., alpha = 0.05)

## S3 method for class 'formula'
make_cld(obj, ..., data = NULL, alpha = 0.05)

## S3 method for class 'matrix'
make_cld(obj, ..., alpha = 0.05)

## S3 method for class 'data.frame'
make_cld(
  obj,
  ...,
  alpha = 0.05,
  gr1_col = "group1",
  gr2_col = "group2",
  p_val_col = "p.adj",
  remove_space = FALSE
)

## S3 method for class 'pairwise_pval_df'
make_cld(obj, ..., alpha = 0.05)
```

Arguments

`obj`	Object with pair-wise comparisons (e.g., post-hoc test results). Currently supported object classes: • `PMCMR` - from packages **PMCMR** and **PMCMRplus** • `pairwise.htest` - from base R (e.g., `pairwise.t.test`, `pairwise.wilcox.test`) • `data.frame` - with comparison results from packages like **rstatix** (e.g., `games_howell_test`, `tukey_hsd`). Requires `gr1_col`, `gr2_col`, `p_val_col` specification. • `PostHocTest` - from package **DescTools** (e.g., `ConoverTest`, `DunnettTest`) • `DunnTest` - from package **DescTools** • `matrix` - symmetric matrices of p-values

- `formula` - interface for data frames
- `pairwise_pval_df` - output from `pval_matrix_to_df()`

`...` Further arguments passed to internal methods. These may include:

- `reversed` - Logical. If `TRUE`, reverses the letter ordering (default: `FALSE`)
- `swap_compared_names` - Logical. If `TRUE`, swaps group order in comparisons (default: `FALSE`)
- `print_comp` - Logical. If `TRUE`, prints comparison names (default: `FALSE`)
- `sep` - Character. Custom separator for comparison strings (default: `"-"`). Use this when your group names contain hyphens. For example, use `sep = ":"` or `sep = ";"`. Only applies to single-variable formula method. Not needed for two-variable formula. Additional cleaning options (default: `TRUE` for most methods):
- `remove_space` - Removes spaces from comparison strings
- `remove_equal` - Removes equal signs from comparison strings
- `swap_colon` - Replaces colons with hyphens (use `FALSE` if using `:` as separator)
- `swap_vs` - Replaces "vs" with hyphens (default: `FALSE`)

`alpha` Numeric value between 0 and 1. The significance level (alpha) for determining which comparisons are significantly different. Comparisons with p-values below this threshold are considered significant. Default is 0.05.

`data` A data frame with p-values and names of comparisons. This argument is required when `obj` is a formula. The data frame should contain at least two columns: one for p-values and one for comparison labels. See examples for details.

`gr1_col` Character string. Name of the column in the data frame containing the first group names in each pairwise comparison. Default is `"group1"`. Only used for the data.frame method. The function will construct comparisons in the format `gr2-gr1`.

`gr2_col` Character string. Name of the column in the data frame containing the second group names in each pairwise comparison. Default is `"group2"`. Only used for the data.frame method.

`p_val_col` Character string. Name of the column in the data frame containing the p-values for each comparison. Default is `"p.adj"` (adjusted p-values). Only used for the data.frame method. Can also be `"p_value"` or any other column name containing numeric p-values.

`remove_space` Logical. If `TRUE`, removes spaces from comparison strings. Default is `FALSE` for the data.frame method to preserve original formatting. Set to `TRUE` if your group names contain spaces and you want compact comparisons.

`formula` An R model `stats::formula()` with two possible formats:

- **Two-variable formula** (recommended): `p_value ~ group1 + group2` where `group1` and `group2` are separate columns containing group names. This format automatically handles hyphens in group names.
- **Single-variable formula**: `p_value ~ Comparison` where `Comparison` is a column with pre-formatted comparison strings (e.g., "A-B", "A-C"). This format has limitations with hyphenated group names. Usually used in combination with `data`.

Value

A data frame of class `c("cld_object", "data.frame")` with three columns:

- `group` - Character. The names of the groups being compared
- `cld` - Character. The compact letter display for each group. Groups sharing at least one letter are not significantly different from each other
- `spaced_cld` - Character. A monospaced version of the CLD where spaces are replaced with underscores, useful for maintaining alignment in output

The rows are ordered according to the group names. Groups with the same letter(s) do not differ significantly at the specified alpha level.

Handling Group Names with Hyphens

The underlying multcompView package uses hyphens (`-`) as the default separator between group names in comparison strings (e.g., "GroupA-GroupB"). This creates a conflict when group names themselves contain hyphens (e.g., "Plant-based", "Treatment-1").

Automatic Handling: Most methods (matrix, data.frame, pairwise.htest, pairwise_pval_df, and others) automatically detect and handle hyphens in group names by:

1. Temporarily replacing hyphens with alternative characters (underscore, en-dash, etc.)
2. Processing the comparisons
3. Restoring the original hyphens in the output

An informational message is shown when this occurs. To suppress it, use `quiet_hyphen_warning = TRUE`.

Formula Method Limitation: The formula method (e.g., `make_cld(p_value ~ comparison, data = df)`) has limited support for group names with hyphens because it receives pre-formatted comparison strings where the separator hyphens cannot be reliably distinguished from hyphens within group names.

Best Practice for Hyphenated Group Names: Use the data.frame method with `gr1_col` and `gr2_col` parameters. This method handles hyphens automatically and seamlessly. For example:

```
# Instead of: make_cld(p_value ~ comparison, data = df)
# Use: make_cld(df, gr1_col = "group1", gr2_col = "group2", p_val_col = "p_value")
```

Alternative Workarounds for Formula Method:

- Convert your data to matrix format (also handles hyphens automatically)
- Replace hyphens in group names with underscores before creating comparisons
- Use a different separator in comparison strings (e.g., " vs " with `swap_vs = TRUE`)

References

Piepho HP (2004). An algorithm for a letter-based representation of all-pairwise comparisons. Journal of Computational and Graphical Statistics, 13(2), 456-466. doi:10.1198/1061860043515, available at https://www.tandfonline.com/doi/abs/10.1198/1061860043515

See Also

Helper Functions:

- pval_matrix_to_df() for converting p-value matrices to data frames
- as_cld() for converting other formats to cld_object

Output Methods:

- print.cld_object() for displaying CLD results
- cld_methods for coercion methods (as.data.frame, as.character, as_tibble)

Underlying Algorithm:

- multcompView::multcompLetters() for the letter generation algorithm

Examples

```
# Example 1: Using pairwise.htest (Wilcoxon test)
obj1 <- stats::pairwise.wilcox.test(chickwts$weight, chickwts$feed, exact = FALSE)
cld::make_cld(obj1)

# Example 2: Using pairwise.htest (t-test)
obj2 <- with(OrchardSprays, stats::pairwise.t.test(decrease, treatment))
cld::make_cld(obj2)

# Example 3: Using pairwise.htest (proportion test)

smokers <- c(83, 90, 129, 70)
patients <- c(86, 93, 136, 82)
obj3 <- stats::pairwise.prop.test(smokers, patients)
cld::make_cld(obj3)


# Example 4: Using PMCMR objects

obj4 <- PMCMRplus::kwAllPairsConoverTest(count ~ spray, data = InsectSprays)
cld::make_cld(obj4)


# Example 5: Using DescTools objects

obj5 <- DescTools::ConoverTest(weight ~ group, data = PlantGrowth)
cld::make_cld(obj5)


# Example 6: Using rstatix data frames (via data.frame method)

obj6 <- rstatix::games_howell_test(PlantGrowth, weight ~ group)
cld::make_cld(obj6, gr1_col = "group1", gr2_col = "group2", p_val_col = "p.adj")


# Example 7: Using formula interface
```

```
my_dataframe <- utils::read.table(
  text = c(
    'Comparison     p_value p.adjust
    "EE - GB = 0"        1 1.000000
    "EE - CY = 0" 0.001093 0.003279
    "GB - CY = 0" 0.005477 0.008216'
  ),
  header = TRUE
)
cld::make_cld(p.adjust ~ Comparison, data = my_dataframe)

# Example 8: Using a symmetric matrix of p-values
# Create matrix
m <- c(
  1.00, 0.22, 0.05, 0.00,
  0.22, 1.00, 0.17, 0.01,
  0.05, 0.17, 1.00, 0.22,
  0.00, 0.01, 0.22, 1.00
)
obj8 <- matrix(m, nrow = 4)
rownames(obj8) <- colnames(obj8) <- c("P", "O", "I", "U")
obj8

# Make CLD
cld::make_cld(obj8)

# Example 9: Using data.frame method with custom column names
my_data <- data.frame(
  group1 = c("A", "A", "B"),
  group2 = c("B", "C", "C"),
  p.adj = c(0.001, 0.045, 0.892)
)
cld::make_cld(my_data, gr1_col = "group1", gr2_col = "group2", p_val_col = "p.adj")
```

`pval_matrix_to_df` *Convert matrix with p-values to data frame*

Description

This function converts a matrix of p-values into a data frame with three columns: `gr1` (column names), `gr2` (row names), and `p_values` (the p-values). This is particularly useful when you have a symmetric matrix of p-values from pairwise comparisons and need to convert it to a long-format data frame for further analysis or for use with `make_cld()`.

Usage

```
pval_matrix_to_df(x)
```

Arguments

x	A numeric matrix containing p-values. If the matrix has no column or row names, they will be assigned sequential numbers. The matrix can be symmetric (with diagonal and upper/lower triangle) or triangular. NA values are removed from the output.

Value

A data frame of class `c("pairwise_pval_df", "data.frame")` with columns:

- `gr1` - names from the columns of the matrix
- `gr2` - names from the rows of the matrix
- `p_values` - the corresponding p-values from the matrix (excluding NA values)

See Also

- `make_cld()` for creating compact letter displays from the resulting data frame
- `as_cld()` for converting other formats to cld_object

Other helper functions: `as_cld()`

Other data conversion functions: `as_cld()`

Examples

```
# Create a matrix of p-values
m <- matrix(c(NA, 0.05, 0.01, 0.05, NA, 0.03, 0.01, 0.03, NA), nrow = 3)
colnames(m) <- c("A", "B", "C")
rownames(m) <- c("A", "B", "C")

# Convert to data frame
cld::pval_matrix_to_df(m)
```

Index