Package 'factorH'

September 11, 2025

Type Package

Title Multifactor Nonparametric Rank-Based ANOVA with Post Hoc Tests

Version 0.4.0

Description Multifactor nonparametric analysis of variance based on ranks. Builds on the Kruskal-Wallis H test and its 2x2 Scheirer-Ray-Hare extension to handle any factorial designs. Provides effect sizes, Dunn-Bonferroni pairwise-comparison matrices, and simple-effects analyses. Tailored for psychology and the social sciences, with beginner-friendly R syntax and outputs that can be dropped into journal reports. Includes helpers to export tab-separated results and compact tables of descriptive statistics (to APA-style reports).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1)

Imports rcompanion, FSA, car, dplyr, stats, utils, rlang

Suggests testthat (>= 3.0.0), knitr, rmarkdown, haven

Config/testthat/edition 3

VignetteBuilder knitr

Contact tomasz.rak@upjp2.edu.pl

LazyData true

NeedsCompilation no

Author Tomasz Rak [aut, cre],
Szymon Wrzesniowski [aut]

Maintainer Tomasz Rak <tomasz.rak@upjp2.edu.pl>

Repository CRAN

Date/Publication 2025-09-11 06:50:02 UTC

Contents

`factorH` *factorH: Multifactor rank-based ANOVA utilities*

Description

Multifactor nonparametric analysis of variance based on ranks. Builds on the Kruskal-Wallis H test and its 2x2 Scheirer-Ray-Hare extension to handle any factorial designs. Provides effect sizes, Dunn-Bonferroni pairwise-comparison matrices, and simple-effects analyses. Tailored for psychology and the social sciences, with beginner-friendly R syntax and outputs that can be dropped into journal reports. Includes helpers to export tab-separated results and compact tables of descriptive statistics (to APA-style reports).

Details

What this package does (and why):

factorH provides a **simple, single-call workflow** for **multifactor nonparametric, rank-based ANOVA** and publication-ready outputs:

- ANOVA-like table based on `ranks` (rooted in **Kruskal-Wallis H** and the 2x2 **Scheirer-Ray-Hare** extension),
- **effect sizes** computed directly from H
- **Dunn–Bonferroni** post hoc **comparison matrices**
- `simple-effects` post hocs (pairwise comparisons **within levels** of conditioning factors),
- compact **descriptive tables** and a **TSV writer** for quick formatting in Excel or a manuscript.

Why? Popular GUI stats tools do not offer a ready-made, **user-friendly multifactor rank-based** pipeline that mirrors standard H / SRH analyses in a way that is easy for beginners. factorH aims to fill that gap with **clear, R-like formula syntax** and a `one-command` report function.

The package is intentionally small: most users will only ever need:

- srh.kway.full(...) to compute everything
- write.srh.kway.full.tsv(...) to export the results into a single tab-separated file.

Formula syntax at a glance:

All high-level functions use standard R **model formulas**:

```
response ~ factorA + factorB + factorC
```

lists **main effects** - Interactions are handled internally. You do not need to write A:B or A*B. The response (left of ~) must be numeric (e.g., a Likert score coded as 1..5 stored as numeric).

Examples below use the included dataset mimicry.

```
library(factorH)
data(mimicry, package = "factorH")
str(mimicry)
```

Predictors should be factors. If not, functions will coerce them.

What is allowed?

```
# One factor (KW-style):
  liking ~ condition

# Two factors (SRH-style):
  liking ~ gender + condition

# Three or more factors (k-way):
  liking ~ gender + condition + age_cat
```

You do not need to write gender:condition or gender*condition. The package will build all needed interactions internally when relevant.

Numeric response (Likert note):

The response must be numeric. For Likert-type items (e.g., 1 = strongly disagree ... 5 = strongly agree), keep them numeric; rank-based tests are robust for such ordinal-like data.

If your Likert is accidentally a factor or character, coerce safely:

```
# if stored as character "1","2",...:
mimicry$liking <- as.numeric(mimicry$liking)
# if stored as factor with labels "1","2",...:
mimicry$liking <- as.numeric(as.character(mimicry$liking))
```

The one-call pipeline:

The main function srh.kway.full() runs:

1. ANOVA-like table on ranks
2. descriptive summary
3. post hoc matrices (Dunn–Bonferroni; P.adj)
4. simple-effects post hocs (within-family Bonferroni).

For 2 factors:

```
res2 <- srh.kway.full(liking ~ gender + condition, data = mimicry)
names(res2)
res2$anova
head(res2$summary)
names(res2$posthoc_cells)
names(res2$posthoc_simple)[1:4]
```

For 3 factors:

```
res3 <- srh.kway.full(liking ~ gender + condition + age_cat, data =
mimicry)
res3$anova
```

Export to a tab-separated file

```
f <- tempfile(fileext = ".tsv")
write.srh.kway.full.tsv(res3, file = f, dec = ".") # decimal dot
file.exists(f)
```

If you need comma as decimal mark:

```
f2 <- tempfile(fileext = ".tsv")
write.srh.kway.full.tsv(res3, file = f2, dec = ",") # decimal comma
file.exists(f2)
```

The TSV contains clearly separated sections: ## SRH: EFFECTS TABLE, ## SUMMARY STATS, ## POSTHOC CELLS, ## SIMPLE EFFECTS, ## META.

What is in the example dataset?:
mimicry is a real study on the **chameleon effect (Trzmielewska, Duras, Juchacz & Rak, 2025)**: how `mimicry` vs other **movement conditions** affect `liking` of an interlocutor. Potential moderators include `gender` and `age` (with dichotomized age_cat, and a 3-level age_cat2). This makes it a natural playground for `multifactor` rank-based analyses.

```
table(mimicry\$condition)
table(mimicry\$gender)
table(mimicry\$age_cat)
```

What the functions compute (high level):

- **srh.kway()**: rank-based k-way ANOVA table using Type II SS on ranks; p-values are tie-corrected; H is reported with and without the correction factor; effect sizes from unadjusted H.
- **srh.effsize()**: 2-way SRH table with effect sizes (eta2H, eps2H) computed from H.
- **nonpar.datatable()**: compact descriptive tables with **global ranks** (means of ranks per cell), medians, quartiles, IQR, etc., for all main effects and interactions.
- **srh.posthocs()**: Dunn–Bonferroni **pairwise matrices** (P.adj) for **all effects** (main and interactions).
- **srh.simple.posthoc() / srh.simple.posthocs()**: `simple-effects` pairwise comparisons **within levels** of conditioning factors (SPSS-like "within" scope by default).
- **srh.kway.full()**: orchestrates all of the above.
- **write.srh.kway.full.tsv()**: exports everything into one TSV (with dot or comma decimal mark).

That is it. For most users, the intro ends here: **use srh.kway.full()** and export with **write.srh.kway.full.tsv()**.

Author(s)

Maintainer: Tomasz Rak <tomasz.rak@upjp2.edu.pl>

Authors:

- Szymon Wrzesniowski <szymon.wrzesniowski@upjp2.edu.pl>

---

factorH_dataset *Datasets in factorH*

Description

Datasets in factorH

Details

What is in the example dataset?:

mimicry is a real study on the **chameleon effect** by Trzmielewska et al. (2025) doi:10.18290/rpsych2024.0019 about how `mimicry` vs other movement conditions affect `liking` of an interlocutor. Potential moderators include `gender` and `age` (with dichotomized age_cat, and a 3-level age_cat2). This makes it a natural playground for `multifactor` rank-based analyses.

```
table(mimicry$condition)
table(mimicry$gender)
table(mimicry$age_cat)
```

---

factorH_reference *factorH functions reference*

Description

factorH functions reference

Details

Function reference:

This document collects **call patterns** and **options** for each public function. All formulas follow response ~ A + B (+ C ...) with **numeric** response and **factor** predictors.

srh.kway.full()

Purpose: one-call pipeline: ANOVA on ranks + descriptives + post hocs + simple effects. **Syntax:** srh.kway.full(y ~ A + B (+ C ...), data, max_levels = 30)

- Automatically chooses the ANOVA engine:
  - 1 factor: srh.kway()
  - 2 factors: srh.effsize()
  - 3+ factors: srh.kway()

- Returns a list: anova, summary, posthoc_cells, posthoc_simple, meta.
- Placeholders:
  - *not applicable* when a component does not apply (e.g., simple effects with 1 factor),
  - *failed...* when a sub-step errors out (keeps the pipeline alive).

Example:

```
res <- srh.kway.full(liking ~ gender + condition + age_cat, data = mimicry)
names(res)
res$anova[1:3]
head(res$summary)
names(res$posthoc_cells)
names(res$posthoc_simple)[1:3]
res$meta
```

Notes:

- Predictors are coerced to factor internally; levels must be 2..max_levels.
- Missing values are removed **pairwise** on the variables in the formula.

write.srh.kway.full.tsv()

Purpose: export the srh.kway.full() result into a single TSV file for fast formatting. **Syntax:** write.srh.kway.full.tsv(obj, file = "srh_kway_full.tsv", sep = ", na ="", dec =".")

- dec = "." or "," controls the decimal mark.
- Numeric fields are written without scientific notation.
- Pretty-printed character tables (e.g., from post hocs) are normalized so that dec="," also affects numbers embedded in strings.

Example:

```
f <- tempfile(fileext = ".tsv")
write.srh.kway.full.tsv(res, file = f, dec = ",")
file.exists(f)
```

srh.kway()

Purpose: general k-way SRH-style ANOVA on ranks (Type II SS), tie-corrected p-values. **Syntax:** srh.kway(y ~ A + B (+ C ...), data, clamp0 = TRUE, force_factors = TRUE, ...)

- Reports: Effect, Df, Sum Sq, H, Hadj (tie correction), p.chisq, k, n, eta2H, eps2H.
- eta2H and eps2H are computed from **unadjusted H** (classical SRH practice).
- force_factors = TRUE coerces predictors to factor (recommended).

Example:

```
k3 <- srh.kway(liking ~ gender + condition + age_cat, data = mimicry)
k3
```

One-factor check (KW-like):

```
k1 <- srh.kway(liking ~ condition, data = mimicry)
k1
```

srh.effsize()

Purpose: 2-way SRH table with effect sizes from H. **Syntax:** srh.effsize(y ~ A + B, data, clamp0 = TRUE, ...)

- Same columns as above but tailored to 2-way SRH.
- clamp0 = TRUE clamps small negatives to 0 for effect sizes.

Example:

```
e2 <- srh.effsize(liking ~ gender + condition, data = mimicry)
e2
```

nonpar.datatable()

Purpose: compact descriptive tables (APA-style), with **global rank means**, medians, quartiles, IQR. **Syntax:** nonpar.datatable(y ~ A + B (+ C ...), data, force_factors = TRUE)

- Returns rows for all **main effects** and all **interaction cells** (constructed internally).
- Rank means are computed on **global ranks** (all observations ranked together), which matches how rank-based ANOVA effects are formed.

Example:

```
dt <- nonpar.datatable(liking ~ gender + condition, data = mimicry)
head(dt)
```

srh.posthoc()

Purpose: Dunn–Bonferroni **pairwise comparison matrix** for a specified effect. **Syntax:** srh.posthoc(y ~ A (+ B + ...), data, method = "bonferroni", digits = 3, triangular = c("lower","upper","full"), numeric = FALSE, force_factors = TRUE, sep = ".")

- Builds a single grouping variable (cells) from the RHS factors and runs FSA::dunnTest.
- Returns a list of three matrices (as data.frames): Z, P.unadj, P.adj.
- triangular = "lower" (default) shows only the lower triangle; diagonal and upper triangle are blank.
- numeric = FALSE returns pretty-printed character tables; set TRUE to get numeric.

Example:

```
ph <- srh.posthoc(liking ~ condition, data = mimicry)
```

srh.posthocs()

Purpose: Dunn–Bonferroni **pairwise matrices for all effects** (main and interactions). **Syntax:** srh.posthocs(y ~ A + B (+ C ...), data, ...)

- Iterates srh.posthoc over: A, B, C, A:B, A:C, B:C, A:B:C, ...
- Returns a named list: names are "A", "B", "A:B", etc.; each value is a P.adj matrix.

Example:

```
phs <- srh.posthocs(liking ~ gender + condition + age_cat, data = mimicry)
names(phs)
phs[["gender:condition"]][1:5, 1:5]
```

srh.simple.posthoc()

Purpose: Simple-effects post hocs (pairwise comparisons **within** levels of conditioning factors). **Syntax:** srh.simple.posthoc(y ~ A + B (+ C ...), data, compare = NULL, scope = c("within","global"), digits = 3)

- compare selects the target factor for pairwise comparisons (default: first RHS factor).

- **Scope**:
  - "within" (default): Bonferroni **within each by-table** (SPSS-like).
  - "global": one Bonferroni across **all** tests from all by-tables combined.
- Returns a data.frame with conditioning columns (BY), Comparison, Z, P.unadj, P.adj, m.tests, adj.note. An "adjustment" attribute describes the correction.

Example:

```
simp <- srh.simple.posthoc(liking ~ gender + condition + age_cat, data = mimicry, compare = "gender", s
head(simp)
```

srh.simple.posthocs()

Purpose: enumerate **all simple-effect configurations** for a given design. **Syntax:** srh.simple.posthocs(y ~ A + B (+ C . . .), data)

- For each target factor and each non-empty combination of the remaining factors as BY, runs srh.simple.posthoc(. . . , scope = "within").
- Returns a named list, names like COMPARE(gender) | BY(condition x age_cat).

Example:

```
sps <- srh.simple.posthocs(liking ~ gender + condition + age_cat, data = mimicry)
head(names(sps), 6)
```

Formula tips and pitfalls

- Do **not** write A:B or A*B. Use A + B (+ C . . .); the package computes all necessary interaction structures internally.
- Response must be **numeric**. For Likert data, keep it numeric 1..k.
- Predictors should be **factors**. If they are not, they will be coerced.
- Coerce predictors to factor explicitly if needed

Example:

```
#coercing
mimicry$gender <- factor(mimicry$gender)
mimicry$condition <- factor(mimicry$condition)
```

Performance and reproducibility

- Functions use ranks and Type II sums of squares (via car::Anova under the hood) and Dunn tests (FSA::dunnTest).
- P-values apply a standard tie correction factor for ranks; effect sizes are derived from unadjusted H (classical SRH practice).
- All outputs are plain data.frames and lists, easy to save and post-process.

factorH_syntax *Syntax and formula patterns*

Description

Syntax and formula patterns

Details

Formula syntax at a glance:

All high-level functions use standard R **model formulas**: response ~ factorA + factorB + factorC

- + lists **main effects** - Interactions are handled internally. You do not need to write A:B or A*B.
- The response (left of ~) must be numeric (e.g., a Likert score coded as 1..5 stored as numeric).

Examples below use the included dataset mimicry.

```
library(factorH)
data(mimicry, package = "factorH")
str(mimicry)
```

Predictors should be factors. If not, functions will coerce them.

What is allowed?

```
# One factor (KW-style):
  liking ~ condition

# Two factors (SRH-style):
  liking ~ gender + condition

# Three or more factors (k-way):
  liking ~ gender + condition + age_cat
```

You do not need to write gender:condition or gender*condition. The package will build all needed interactions internally when relevant.

Numeric response (Likert note):

The response must be numeric. For Likert-type items (e.g., 1 = strongly disagree ... 5 = strongly agree), keep them numeric; rank-based tests are robust for such ordinal-like data.

If your Likert is accidentally a factor or character, coerce safely:

```
# if stored as character "1","2",...:
mimicry$liking <- as.numeric(mimicry$liking)
# if stored as factor with labels "1","2",...:
mimicry$liking <- as.numeric(as.character(mimicry$liking))
```

`mimicry` *Mimicry dataset*

Description

A dataset used to demonstrate rank-based (nonparametric) multifactor ANOVA.

Usage

```
data(mimicry)
```

Format

A data frame with 533 rows and 7 variables:

condition factor; 5 levels

gender factor; 2 levels

age numeric

age_cat factor; 2 levels

age_cat2 factor; 3 levels

field factor; 2 levels

liking numeric; dependent variable

Details

Factor encodings follow the original SPSS labels converted to R factors.

Source

Converted from an SPSS file as part of the factorH package examples.

References

Trzmielewska, W., Duras, J., Juchacz, A., & Rak, T. (2025). Examining the impact of control condition design in mimicry–liking link research: how motor behavior may impact liking. *Annals of Psychology*, 4, 351–378. doi:10.18290/rpsych2024.0019

nonpar.datatable *Compact descriptive tables (APA-style) with global rank means*

Description

Produces descriptive statistics for all main effects and interaction cells implied by the RHS of `formula`. Ranks are computed *globally* (across all observations) and cell-wise mean ranks are reported (**recommended** for interpreting rank-based factorial effects).

Usage

```
nonpar.datatable(formula, data, force_factors = TRUE)
```

Arguments

`formula`	A formula of the form `y ~ A (+ B + ...)`.
`data`	A `data.frame` containing `y` and the grouping factors.
`force_factors`	Logical; coerce grouping variables to `factor` (default `TRUE`).

Details

The function first subsets to complete cases on `y` and all RHS factors, then computes global ranks of `y` (`ties.method = "average"`). For each effect (every non-empty combination of factors up to full order), it returns a row per cell with: `count`, `mean`, `sd`, `median`, quartiles (`q1`, `q3`), `IQR`, and `mean_rank`. The column `Effect` identifies the effect (e.g., `"A"`, `"B"`, `"A:B"`). Missing factor columns for a given effect are added with `NA` values but retain the proper factor levels for easy binding.

Value

A base `data.frame` with columns:

- `Effect` (character),
- factor columns for all RHS factors (factors, possibly `NA` in some rows),
- `count`, `mean`, `sd`, `median`, `q1`, `q3`, `IQR`, `mean_rank`.

The original call is attached as attribute `"call"`.

Examples

```
data(mimicry, package = "factorH")

# One factor
nonpar.datatable(liking ~ condition, data = mimicry)

# Two factors: rows for gender, for condition, and for gender:condition
nonpar.datatable(liking ~ gender + condition, data = mimicry)
```

```
# Three factors: all mains + 2-way and 3-way cells
nonpar.datatable(liking ~ gender + condition + age_cat, data = mimicry)
```

`srh.effsize` *SRH with effect sizes for two-factor designs*

Description

Extends `rcompanion::scheirerRayHare()` by adding popular rank-based effect sizes for each SRH term: eta^2_H and epsilon^2_H, and stores the original function call.

Usage

```
srh.effsize(formula, data, clamp0 = TRUE, ...)
```

Arguments

`formula`	A formula of the form `y ~ A + B`.
`data`	A `data.frame` containing all variables in `formula`.
`clamp0`	Logical; if `TRUE` (default), negative eta^2_H is truncated to 0 and epsilon^2_H truncated to the interval $[0, 1]$.
`...`	Passed to `rcompanion::scheirerRayHare()`.

Details

Let H be the SRH H-statistic for a given term, n the sample size used by SRH (complete cases on y and factors), and k the number of groups compared by that term (for interactions, the number of observed combinations).

Effect sizes computed:

- **Eta^2_H**: $(H - k + 1)/(n - k)$.
- **Epsilon^2_H** (KW-like): $H * (n + 1)/(n^2 - 1)$.

The original call is stored as an attribute and can be retrieved with `getCall()`.

Value

A `data.frame` (classed as `c("srh_with_call","anova","data.frame")`) with the SRH table extended by columns: `k`, `n`, `eta2H`, `eps2H`.

Examples

```
data(mimicry, package = "factorH")
res <- srh.effsize(liking ~ gender + condition, data = mimicry)
res
getCall(res)
```

`srh.kway` *K-way SRH on ranks with tie-corrected p-values and rank-based effect sizes*

Description

Generalizes the Scheirer–Ray–Hare (SRH) approach to k-factor designs by using Type II sums of squares from a linear model on ranks, with a standard tie correction D applied to p-values. The function returns H, tie-corrected H (`Hadj`), p-values and rank-based effect sizes (`eta2H`, `eps2H`) for each main effect and interaction up to the full order (i.e., `(A + B + ...)^k`).

Usage

```
srh.kway(formula, data, clamp0 = TRUE, force_factors = TRUE, ...)
```

Arguments

`formula`	A formula of the form `y ~ A + B (+ C ...)`.
`data`	A `data.frame` with the variables in `formula`.
`clamp0`	Logical; if `TRUE` (default), negative `eta2H` is truncated to 0 and `eps2H` truncated to the interval $[0, 1]$.
`force_factors`	Logical; coerce grouping variables to `factor` (default `TRUE`).
`...`	Passed to `stats::lm()` if applicable.

Details

Ranks are computed globally on `y` (`ties.method = "average"`). Type II sums of squares are obtained from `car::Anova(fit, type = 2)` on the rank model `R ~ (A + B + ...)^k`. The tie correction is

$$D = 1 - \frac{\sum(t^3 - t)}{n^3 - n},$$

where t are tie block sizes and n is the number of complete cases. We report `Hadj = H / D` and $p = P(\chi^2_{df} \geq Hadj)$.

Rank-based effect sizes are computed from the *uncorrected* `H` (classical SRH convention):

- `eta2H = (H - k + 1) / (n - k)`, where `k` is the number of groups compared by the term (for interactions, the number of observed combinations),
- `eps2H = H * (n + 1) / (n^2 - 1)` (KW-like epsilon squared).

Value

A `data.frame` with class `c("srh_kway","anova","data.frame")` containing columns: `Effect`, `Df`, `Sum Sq`, `H`, `Hadj`, `p.chisq`, `k`, `n`, `eta2H`, `eps2H`. The original call is attached as an attribute and can be retrieved with `getCall()`.

Examples

```
data(mimicry, package = "factorH")
# One factor (KW-style check)
srh.kway(liking ~ condition, data = mimicry)

# Two factors
srh.kway(liking ~ gender + condition, data = mimicry)

# Three factors
srh.kway(liking ~ gender + condition + age_cat, data = mimicry)
```

`srh.kway.full` *Full pipeline: rank-based k-way ANOVA + descriptives + post hocs*

Description

Runs a complete nonparametric, rank-based workflow for factorial designs: (1) SRH-style ANOVA table, (2) compact descriptive stats with global ranks, (3) Dunn-Bonferroni post hoc matrices for all effects, and (4) simple-effects post hocs (Bonferroni within each by-table).

Usage

```
srh.kway.full(formula, data, max_levels = 30)
```

Arguments

`formula`	A formula `y ~ A (+ B + ...)`.
`data`	A `data.frame` with variables present in `formula`.
`max_levels`	Safety cap for number of levels per factor (default 30).

Details

Choice of the ANOVA engine:

- 1 factor: `srh.kway()` (KW-like),
- 2 factors: `srh.effsize()` (SRH 2-way + effect sizes),
- 3+ factors: `srh.kway()` (general k-way on ranks).

Value

A list with elements:

- `anova` – ANOVA-like table,
- `summary` – descriptive stats data.frame,
- `posthoc_cells` – list of p.adj matrices for all effects (from `srh.posthocs`), or a string when failed,

- posthoc_simple – list of simple-effect tables (from srh.simple.posthocs); for 1 factor: "[not applicable]",
- meta – list with call, n, factor levels, and empty-cell info (if 2+ factors).

Components that cannot be computed for the given design are returned as the string "[not applicable]"; failures are reported as "[failed] <message>".

Examples

```
data(mimicry, package = "factorH")
# 1 factor
f1 <- srh.kway.full(liking ~ condition, data = mimicry)
# 2 factors
f2 <- srh.kway.full(liking ~ gender + condition, data = mimicry)
# 3 factors
f3 <- srh.kway.full(liking ~ gender + condition + age_cat, data = mimicry)
```

---

srh.posthoc *Dunn post hoc in a symmetric matrix form (one specified effect)*

Description

Computes Dunn's rank-based pairwise comparisons for the effect implied by formula and returns symmetric matrices for Z, unadjusted p-values, and adjusted p-values. Cells on one triangle (or both) can be blanked for compact reporting. For multi-factor RHS, factors are combined into a single grouping via interaction() (e.g., "A:B" cells).

Usage

```
srh.posthoc(
  formula,
  data,
  method = "bonferroni",
  digits = 3,
  triangular = c("lower", "upper", "full"),
  numeric = FALSE,
  force_factors = TRUE,
  sep = "."
)
```

Arguments

formula	A formula of the form y ~ factor or y ~ A + B (the latter is treated as *one* combined grouping via interaction).
data	A data.frame containing variables in formula.
method	P-value adjustment method passed to FSA::dunnTest(). Default "bonferroni". See p.adjust.methods for options.

digits	Number of digits for rounding in the returned matrices when numeric = FALSE. Default 3.
triangular	Which triangle to show ("lower", "upper", or "full"). Default "lower".
numeric	Logical; if TRUE, return numeric matrices/data frames with NA on the masked triangle/diagonal. If FALSE (default), return character data frames with masked cells as empty strings.
force_factors	Logical; coerce grouping variables to factor (default TRUE).
sep	Separator used in interaction() when combining factors. Default ".".

Details

The function subsets to complete cases on y and RHS factors, optionally coerces factors, builds a single grouping variable (._grp) and calls FSA::dunnTest(y ~ ._grp, data = ..., method = ...). The pairwise results are placed into symmetric matrices Z, P.unadj, and P.adj. By default only the lower triangle (excluding diagonal) is shown for compactness.

Value

A list with three data.frames:

- Z – Z statistics,
- P.unadj – unadjusted p-values,
- P.adj – adjusted p-values (per method).

The original call is attached as attribute "call".

Examples

```
data(mimicry, package = "factorH")

# One factor
ph1 <- srh.posthoc(liking ~ condition, data = mimicry)
ph1$`P.adj`    # gotowa macierz p po korekcji

# Two factors combined (all A:B cells vs all A:B cells)
ph2 <- srh.posthoc(liking ~ gender + condition, data = mimicry)
ph2$`P.adj`

# Upper triangle, numeric frames
ph3 <- srh.posthoc(liking ~ condition, data = mimicry,
                   triangular = "upper", numeric = TRUE)
ph3$Z
```

`srh.posthocs` *Dunn post hoc tables (p.adj only) for all effects in a factorial design*

Description

For a given `y ~ A (+ B + ...)` formula, runs `srh.posthoc` for every main effect and interaction implied by the RHS (all non-empty combinations of factors) and returns a named list of adjusted p-value matrices (`P.adj`) for each effect.

Usage

```
srh.posthocs(
  formula,
  data,
  method = "bonferroni",
  digits = 3,
  triangular = c("lower", "upper", "full"),
  numeric = FALSE,
  force_factors = TRUE,
  sep = "."
)
```

Arguments

`formula`	A formula of the form `y ~ A (+ B + ...)`.
`data`	A `data.frame` containing variables in `formula`.
`method`	P-value adjustment method passed to `FSA::dunnTest()` via `srh.posthoc`. Default `"bonferroni"`.
`digits`	Rounding used inside `srh.posthoc` when `numeric = FALSE`. Default `3`.
`triangular`	Which triangle to show in each matrix (`"lower"`, `"upper"`, `"full"`). Default `"lower"`.
`numeric`	Logical; if `TRUE`, return numeric data frames with `NA`s on the masked triangle/diagonal; if `FALSE` (default), return character data frames with masked cells as empty strings.
`force_factors`	Logical; coerce grouping variables to `factor` before analysis (default `TRUE`).
`sep`	Separator for combined factor labels when needed (passed through to `srh.posthoc`). Default `"."`.

Details

The function enumerates all non-empty subsets of RHS factors (mains, 2-way, ..., k-way) and calls `srh.posthoc` on each corresponding sub-formula. If a subset has fewer than 2 observed levels (e.g., due to missing data after subsetting to complete cases), that effect is skipped.

Value

A named `list` where each element is a `data.frame` of adjusted p-values (`P.adj`) for an effect. Names use `"A"`, `"B"`, `"A:B"`, ..., matching the effect structure. The original call is attached as attribute `"call"`.

Examples

```
data(mimicry, package = "factorH")

# Two-factor design: p.adj for 'gender', 'condition', and 'gender:condition'
L2 <- srh.posthocs(liking ~ gender + condition, data = mimicry)
names(L2)
L2$gender
L2$condition
L2$`gender:condition`

# Three-factor design: includes mains, all 2-ways, and the 3-way effect
L3 <- srh.posthocs(liking ~ gender + condition + age_cat, data = mimicry)
names(L3)
```

`srh.simple.posthoc` *Simple-effects post hoc (Dunn) with Bonferroni adjustment*

Description

Computes Dunn's pairwise comparisons for **simple effects** of one target factor (`compare`) within levels of the remaining conditioning factors (`by`). Adjustment can be done **within** each conditioning table (SPSS-like) or **globally** across all tests.

Usage

```
srh.simple.posthoc(
  formula,
  data,
  compare = NULL,
  scope = c("within", "global"),
  digits = 3
)
```

Arguments

`formula`	A formula of the form `y ~ A + B (+ C ...)`; requires at least two RHS factors to define a simple effect.
`data`	A `data.frame` containing variables in `formula`.
`compare`	Character; the factor to compare pairwise. By default, the first factor on the RHS of `formula`.

scope	"within" (default) applies Bonferroni adjustment **within each by-table**; "global" applies one Bonferroni across **all** pairwise tests produced for all by-tables combined.
digits	Number of digits for rounding numeric columns (Z, P.unadj, P.adj). Default 3.

Details

The data are subset to complete cases on y and all RHS factors. All RHS variables are coerced to factor. The table is split by all factors except compare and Dunn's test (FSA::dunnTest) is run per split. With scope = "within", the Bonferroni correction is applied separately in each split (with m.tests = choose(k,2) for that split). With scope = "global", P.adj is re-computed once with stats::p.adjust(..., method = "bonferroni") across **all** pairwise tests from all splits (and m.tests is set to the total number of tests).

Value

A data.frame with columns:

- conditioning factor columns (one value repeated per split),
- Comparison, Z, P.unadj, P.adj,
- m.tests (number of tests used for Bonferroni),
- adj.note (human-readable note).

Attributes: "adjustment" (one-line description) and "call".

Examples

```
data(mimicry, package = "factorH")

# Two factors: pairwise comparisons for 'gender' within levels of 'condition'.
# By default, compare = first RHS factor ('gender' here).
# p.adj uses Bonferroni within each by-table (scope = "within").
tab1 <- srh.simple.posthoc(liking ~ gender + condition, data = mimicry)
head(tab1); attr(tab1, "adjustment")

# One global family of tests (global Bonferroni across all subgroup tests):
tab2 <- srh.simple.posthoc(liking ~ gender + condition, data = mimicry,
                           scope = "global")
head(tab2); attr(tab2, "adjustment")

# Three factors: compare 'gender' within each condition × age_cat cell.
tab3 <- srh.simple.posthoc(liking ~ gender + condition + age_cat, data = mimicry)
head(tab3)

# Choose a different target factor to compare: here 'condition'
# (within each gender × age_cat cell).
tabA <- srh.simple.posthoc(liking ~ gender + condition + age_cat, data = mimicry,
                           compare = "condition")
head(tabA)
```

```
# Global Bonferroni variants (less common, but sometimes requested):
tabG  <- srh.simple.posthoc(liking ~ gender + condition + age_cat, data = mimicry,
                            scope = "global")
tabG2 <- srh.simple.posthoc(liking ~ condition + gender, data = mimicry)
tabG3 <- srh.simple.posthoc(liking ~ condition + gender, data = mimicry,
                            scope = "global")
head(tabG); head(tabG2); head(tabG3)
```

`srh.simple.posthocs` *Simple-effects post hoc tables for all possible effects (within-scope)*

Description

For a formula `y ~ A + B (+ C ...)`, enumerates **all simple-effect setups** of the form `COMPARE(target) | BY(other factors)` and runs `srh.simple.posthoc` with `scope = "within"` for each. Returns a named list of data frames (one per simple-effect configuration).

Usage

```
srh.simple.posthocs(formula, data)
```

Arguments

`formula`	A formula `y ~ A + B (+ C ...)` with at least two RHS factors.
`data`	A `data.frame` containing the variables in `formula`.

Details

For each choice of the comparison factor `target` from the RHS, all non-empty combinations of the remaining factors are treated as conditioning sets `BY`. For each pair `(target, BY)` we call `srh.simple.posthoc()` with `compare = target` and `scope = "within"`. Effects where the conditioning subset has < 2 levels of `target` are skipped; messages are collected in attribute `"skipped"`.

Labels use ASCII: `"COMPARE(A) | BY(B x C)"` (plain `" x "`).

Value

A named `list` of `data.frames`. Each element contains the columns produced by `srh.simple.posthoc` (e.g., `Comparison`, `Z`, `P.unadj`, `P.adj`, `m.tests`, `adj.note`). Attributes: `"call"` and (optionally) `"skipped"` with messages.

Examples

```
data(mimicry, package = "factorH")

# All simple-effect tables for a 2-factor design
tabs2 <- srh.simple.posthocs(liking ~ gender + condition, data = mimicry)
names(tabs2)
# e.g., tabs2[["COMPARE(gender) | BY(condition)"]]
```

```
# Three factors: all COMPARE(target) | BY(conditioning) combinations
tabs3 <- srh.simple.posthocs(liking ~ gender + condition + age_cat, data = mimicry)
names(tabs3)
attr(tabs3, "skipped")  # any skipped combos with reasons
```

`write.srh.kway.full.tsv`

Write full SRH pipeline result to a TSV file

Description

Exports the result of `srh.kway.full` into a single, tab-separated text file, in the order: *ANOVA* > *SUMMARY* > *POSTHOC CELLS* > *SIMPLE EFFECTS* > *META*. Supports choosing the decimal mark for numeric values.

Usage

```
write.srh.kway.full.tsv(
  obj,
  file = "srh_kway_full.tsv",
  sep = "\t",
  na = "",
  dec = "."
)
```

Arguments

`obj`	A list produced by `srh.kway.full`.
`file`	Path to the output TSV file. Default `"srh_kway_full.tsv"`.
`sep`	Field separator (default tab `"\t"`).
`na`	String to use for missing values (default empty string).
`dec`	Decimal mark for numbers: dot `"."` (default) or comma `","`.

Details

Each section is preceded by a header line (e.g., `## SRH: EFFECTS TABLE`). For post hoc sections, each effect/table is prefixed with a subheader (e.g., `### posthoc_cells: gender:condition`). For simple-effect tables, the attribute `"adjustment"` (if present) is written as a comment line beginning with `"# "`.

Components that are not applicable (e.g., simple effects in 1-factor designs) or failed computations are written as literal one-line messages.

Value

(Invisibly) the normalized path to `file`.

Examples

```
data(mimicry, package = "factorH")
res <- srh.kway.full(liking ~ gender + condition, data = mimicry)

# Write to a temporary file (CRAN-safe)
f <- tempfile(fileext = ".tsv")
write.srh.kway.full.tsv(res, file = f, dec = ".")
file.exists(f)
```

Index