Package 'uxr'

May 8, 2026

Type Package

Title User Experience Research

Version 0.2.0

Description Provides convenience functions for user experience research with an emphasis on quantitative user experience testing and reporting. The functions are designed to translate statistical approaches to applied user experience research.

URL https://joe-chelladurai.github.io/uxr/

BugReports https://github.com/joe-chelladurai/uxr/issues

License MIT + file LICENSE

Encoding UTF-8

Imports cli, dplyr, huxtable, magrittr, purrr, rlang, scales, stringr, tibble, tidyr

Language en

RoxygenNote 7.2.1

Depends R (>= 2.10)

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Joe Chelladurai [aut, cre] (ORCID: <https://orcid.org/0000-0001-8477-3753>)

Maintainer Joe Chelladurai <joe.chelladurai@yahoo.com>

Repository CRAN

Date/Publication 2022-12-25 07:20:02 UTC

Contents

`benchmark_event` *Compare Probability of an Event with Benchmark*

Description

Compare Probability of an Event with Benchmark

Usage

```
benchmark_event(
  data,
  column,
  benchmark,
  event,
  count,
  total,
  event_type = "",
  remove_missing = TRUE,
  notes = "minimal",
  input = "long",
  output = "console"
)
```

Arguments

data	dataset
column	name of column
benchmark	benchmark
event	specify event as given in column (example: 0, "pass", "success")
count	number of times event has occurred. Use only when using input = "values"
total	total number of all events. Use only when using input = "values"
event_type	Optional: a string describing the type of event. For example, success, failure, etc.
remove_missing	TRUE/FALSE (Default is TRUE)
notes	whether output should contain minimal or technical type of notes. Defaults to "minimal". Use "none" to turn off.
input	Default: "long" - long form of data, "values" to pass values directly. If using this option, must specify count and total.
output	Default: "console" - prints output in console and returns tibble invisibly.

Value

Dataframe of results when saved to an object. Show console output by default

Examples

```
data <- data.frame(task_1 = c("y", "y", "y", "y", "n", "n", "n", NA, NA, NA, NA, NA, NA, NA),
                   task_2 = c(0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1))
# With dataframe columns

benchmark_event(data, column = task_1, benchmark = 0.8, event = "y")
benchmark_event(data, column = task_2, benchmark = 0.3, event = 1, event_type = "success")

# Also pipeable
data |>
  benchmark_event(column = task_2, benchmark = 0.3, event = 1, event_type = "success")

# With direct values
benchmark_event(benchmark = 0.8, count = 4, total = 7, input = "values")
```

---

benchmark_score *Compare Score with a Benchmark*

Description

Compare Score with a Benchmark

Usage

```
benchmark_score(
  data,
  column,
  benchmark,
  mean,
  sd,
  n,
  tail = "one",
  remove_missing = TRUE,
  input = "long",
  output = "console"
)
```

Arguments

data	dataframe
column	a column of scores from the dataframe
benchmark	benchmark
mean	if input = "values", enter mean value
sd	if input = "values", enter standard deviation value
n	if input = "values", enter total number of scores
tail	one-tailed or two-tailed test
remove_missing	TRUE/FALSE (Default is TRUE)
input	Default: "long" - long form of data, "values" to pass values directly. If using this option, must specify mean, sd, and n.
output	Default: "console" - prints output in console and returns tibble invisibly.

Value

dataframe of results when saved to an object. show console output by default

Examples

```
scores <- 80 + 23 * scale(rnorm(172)) # 80 = mean, 23 = sd
data <- data.frame(scores = scores)
benchmark_score(data, scores, 67)
data |> benchmark_score(scores, 67)
benchmark_score(mean = 80, sd = 23, n = 172, benchmark = 67, input = "values")
```

benchmark_time *Compare Time with a Benchmark*

Description

Compare Time with a Benchmark

Usage

```
benchmark_time(
  data,
  column,
  benchmark,
  alpha,
  remove_missing = FALSE,
  input = "long",
  output = "console"
)
```

Arguments

data	dataframe
column	a column or vector of time values
benchmark	benchmark
alpha	alpha
remove_missing	TRUE/FALSE (Default is TRUE)
input	Default: "long" - long form of data, "values" to pass values directly. If using this option, must specify count and total.
output	Default: "console" - prints output in console and returns tibble invisibly.

Value

lower_ci, upper_ci, t, probability

Examples

```
data <- data.frame(time = c(60, 53, 70, 42, 62, 43, 81))
benchmark_time(data, column = time, benchmark = 60, alpha = 0.05)
```

compare_means_between_groups
Compare Means Between Groups

Description

Compare Means Between Groups

Usage

```
compare_means_between_groups(
  data,
  var1,
  var2,
  variable,
  grouping_variable,
  groups,
  test = "Welch",
  input = "wide",
  output = "console"
)
```

Arguments

data	data
var1	variable 1
var2	variable 2
variable	variable
grouping_variable	Group
groups	Specify groups from grouping variable
test	Default: "Welch", choose between "student" and "Welch"
input	Default: "wide", choose between "long" and "wide". "wide" requires data var1 var2. "long" requires data, variable, grouping_variable groups
output	Default: "console" - prints output in console and returns tibble invisibly.

Value

results

Examples

```
# Wide data - default
data_wide <- data.frame(A = c(4, 2, 5, 3, 6, 2, 5),
                        B = c(5, 2, 1, 2, 1, 3, 2))

compare_means_between_groups(data_wide, var1 = A, var2 = B)

# Long data
data_long <- data_wide |> tibble::rowid_to_column("id") |>
  tidyr::pivot_longer(cols = -id, names_to = "group", values_to = "variable")

compare_means_between_groups(data_long, variable = variable,
                         grouping_variable = group, groups = c("A", "B"), input = "long")
```

`compare_means_within_groups` *Compare Means Within Groups*

Description

Compare Means Within Groups

Usage

```
compare_means_within_groups(
  data,
  var1,
  var2,
  input = "wide",
  output = "console"
)
```

Arguments

`data`	dataframe
`var1`	variable 1
`var2`	variable 2
`input`	Default: "long" - long form of data, "values" to pass values directly. If using this option, must specify mean, sd, and n.
`output`	Default: "console" - prints output in console and returns tibble invisibly.

Value

results

Examples

```
data <- data.frame(id = c(1:7),
  task1 = c(4, 1, 2, 3, 8, 4, 4),
  task2 = c(7, 13, 9, 7, 18, 8, 10))
compare_means_within_groups(data, task1, task2)
```

`compare_rates_between_groups`
Compare Rates Between Groups

Description

Compare Rates Between Groups

Usage

```
compare_rates_between_groups(
  data,
  group,
  event,
  test,
  input = "long",
  output = "console"
)
```

Arguments

`data`	data
`group`	column in dataframe : group
`event`	column in dataframe : event
`test`	Type of test (fisher, n-1 two prop)
`input`	Defaults to "long"
`output`	"console" prints output to console; "tibble" returns tibble

Value

results

Examples

```
design = c("A","B")
complete = c(34, 24)
incomplete = c(317, 301)
data <- data.frame(design, complete, incomplete)
data <- data |> tidyr::pivot_longer(!design, names_to = "rate", values_to = "n") |>
  tidyr::uncount(n)
compare_rates_between_groups(data, group = design, event = rate)
```

compare_rates_within_groups *Compare Rates Within Groups*

Description

Compare Rates Within Groups

Usage

```
compare_rates_within_groups(
  data,
  x,
  y,
  conf_level = 0.95,
  input,
  output = "console"
)
```

Arguments

data	data
x	var 1
y	var 2
conf_level	Confidence level
input	input type currently only accepts "wide"
output	Default is "console", also accepts "tibble"

Value

results

Examples

```
A <- c(1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1)
B <- c(0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0)
data <- data.frame(A, B)
compare_rates_within_groups(data, A, B, input = "wide")
```

dist_t *T Distribution*

Description

T Distribution

Usage

```
dist_t(t, df, tail)
```

Arguments

t	t
df	degrees of freedom
tail	'one' or 'two'

Value

value

Examples

```
dist_t(1.4, 2, "one")
dist_t(1.4, 2, "two")
```

get_concordant_discordant_pairs *Get concordant and discordant pairs for two variables*

Description

Get concordant and discordant pairs for two variables

Usage

```
get_concordant_discordant_pairs(data, x, y)
```

Arguments

data	= data
x	variable 1
y	variable 2

Value

a data frame

Examples

```
mtcars$id <- seq.int(nrow(mtcars))
get_concordant_discordant_pairs(mtcars, x = vs, y = am)
```

---

get_confidence_intervals_event

Get Confidence Intervals for Event Rate

Description

Get Confidence Intervals for Event Rate

Usage

```
get_confidence_intervals_event(event, total, confidence_level)
```

Arguments

event	event
total	total
confidence_level	confidence level as z value

Value

lower_ci, upper_ci

Examples

```
get_confidence_intervals_event(event = 80,
                               total = 100,
                               confidence_level = 1.96)
```

get_confidence_intervals_within_groups *Get Confidence Intervals Within Groups*

Description

Get Confidence Intervals Within Groups

Usage

```
get_confidence_intervals_within_groups(data, x, y, conf_level = 0.95)
```

Arguments

data	data
x	var 1
y	var 2
conf_level	Confidence level

Value

results

Examples

```
A <- c(1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1)
B <- c(0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0)
data <- data.frame(A, B)
get_confidence_intervals_within_groups(data, A, B)
```

stat_mean_ci *Mean Confidence Intervals*

Description

Mean Confidence Intervals

Usage

```
stat_mean_ci(x, alpha)
```

Arguments

x	values
alpha	alpha

Value

lower_ci, upper_ci

Examples

```
stat_mean_ci(c(1, 2, 3, 4, 5, 6, 7), 1.96)
stat_mean_ci(c(2, 4, 6, 8), 1.96)
```

---

stat_mean_ci_2 *Mean Confidence Intervals (Large Samples)*

Description

Mean Confidence Intervals (Large Samples)

Usage

```
stat_mean_ci_2(x, z)
```

Arguments

x	values
z	z value

Value

lower_ci, upper_ci

Examples

```
stat_mean_ci_2(c(1, 2, 3, 4, 5, 6, 7), 1.96)
stat_mean_ci_2(c(2, 4, 6, 8), 1.96)
```

---

table_observed_expected *Observed Expected Table*

Description

Observed Expected Table

Usage

```
table_observed_expected(data, x, y)
```

Arguments

data	data
x	x
y	y

Value

results

---

test_chisq_one *Chi-squared One Sample*

Description

Chi-squared One Sample

Usage

```
test_chisq_one(data, x)
```

Arguments

data	data
x	x

Value

results

Examples

```
data <- tibble::tribble(~fruit, ~count,
                        "Apple"          , 29,
                        "Banana"         , 24,
                        "Cucumber"       , 22,
                        "Dragon Fruit"   , 19
                        )

data <- data |>
  tidyr::uncount(weights = count) |>
  tibble::rowid_to_column("id")

test_chisq_one(data, fruit)
```

test_chisq_two *Chi-squared Two Sample*

Description

Chi-squared Two Sample

Usage

```
test_chisq_two(data, x, y)
```

Arguments

data	data
x	x
y	y

Value

results

Examples

```
data <- tibble::tribble(~fruit, ~count,
                        "Apple"          , 29,
                        "Banana"         , 24,
                        "Cucumber"       , 22,
                        "Dragon Fruit"   , 19
                        )

data <- data |>
  tidyr::uncount(weights = count) |>
  tibble::rowid_to_column("id")

test_chisq_one(data, fruit)
```

test_fisher *Fisher's Test*

Description

Fisher's Test

Usage

```
test_fisher(data, x, y)
```

Arguments

data	data
x	x
y	y

Value

results

Examples

```
design = c("A","B")
complete = c(11, 5)
incomplete = c(1, 5)
data <- data.frame(design, complete, incomplete)
data <- data |> tidyr::pivot_longer(!design, names_to = "rate", values_to = "n") |>
  tidyr::uncount(n)
test_fisher(data, design, rate)
```

---

test_mcnemar *McNemar Test*

Description

McNemar Test

Usage

```
test_mcnemar(data, x, y)
```

Arguments

data	data
x	var 1
y	var 2

Value

results

Examples

```
A <- c(1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1)
B <- c(0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0)
data <- data.frame(A, B)
test_mcnemar(data, A, B)
```

test_n_1_prop *N-1 Two Proportions Test*

Description

N-1 Two Proportions Test

Usage

```
test_n_1_prop(data, x, y, conf_level = 0.95)
```

Arguments

data	data
x	x
y	y
conf_level	Confidence Level (default = 0.95)

Value

results

Examples

```
design = c("A","B")
complete = c(37, 22)
incomplete = c(418, 416)
data <- data.frame(design, complete, incomplete)
data <- data |> tidyr::pivot_longer(!design, names_to = "rate", values_to = "n") |>
  tidyr::uncount(n)
test_n_1_prop(data, design, rate, conf_level = 0.95)
```

test_t *T-test*

Description

T-test

Usage

```
test_t(x, y, ...)
```

Arguments

x	x
y	y
...	other arguments passed to t-test

Value

results

Examples

```
test_t(mtcars$mpg, mtcars$am)
```

`test_t_paired` *Paired t-test*

Description

Paired t-test

Usage

```
test_t_paired(x, y, ...)
```

Arguments

x	x
y	y
...	other arguments passed to paired t-test

Value

results

test_wald *Wald Confidence Intervals*

Description

Wald Confidence Intervals

Usage

```
test_wald(success, total, conf_level = 0.95)
```

Arguments

success	success
total	total
conf_level	conf_level (default: 0.95)

Value

lower_ci, upper_ci

Examples

```
test_wald(10, 12, 0.95)
test_wald(5, 7, 0.95)
```

test_wald_adj *Adjusted Wald Confidence Intervals*

Description

Adjusted Wald Confidence Intervals

Usage

```
test_wald_adj(success, total, conf_level = 0.95)
```

Arguments

success	success
total	total
conf_level	conf_level (default: 0.95)

Value

lower_ci, upper_ci

Examples

```
test_wald_adj(10, 12, 0.95)
test_wald_adj(5, 7, 0.95)
```

Index