Package: apastats2 (via r-universe)

November 7, 2026

Type Package

Title Statistical Results in APA Style

Version 1.1.0

Description Proper formatting of statistical results in accordance with the American Psychological Association (APA) Manual (sixth edition), along with convenient helpers for summaries, confidence intervals, and plotting. Development repository: <https://github.com/achetverikov/APAstats>.

License GPL-3

Encoding UTF-8

Language en-US

URL https://achetverikov.github.io/APAstats/, https://github.com/achetverikov/APAstats

BugReports https://github.com/achetverikov/APAstats/issues

Imports stringr, utils, Hmisc, plyr, labeling, scales

Suggests superb, knitr, rmarkdown, lsr, MASS, pROC, afex, lme4, ez, diptest, BayesFactor, brms, car, emmeans, lmerTest, bayestestR, rockchalk, carData, testthat (>= 3.0.0)

Depends stats, ggplot2, data.table, R (>= 4.1.0)

LazyData TRUE

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Andrey Chetverikov [aut, cre] (ORCID: <https://orcid.org/0000-0003-2767-6310>)

Maintainer Andrey Chetverikov <andrey.chetverikov@uib.no>

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

Date/Publication 2026-11-07 16:56:22 UTC

RemoteUrl https://github.com/cran/apastats2

RemoteRef HEAD

RemoteSha 102e5ecb54aedbc07c1a6d304d70fdbfd90a661b

Contents

adv_cut . . . 3
aggr2 . . . 4
apa . . . 5
apa.afex_aov . . . 6
apa.anova . . . 7
apa.anova.merMod . . . 8
apa.aov . . . 9
apa.BFBayesFactor . . . 10
apa.brmsfit . . . 11
apa.chisq.test . . . 12
apa.dip.test . . . 13
apa.ezANOVA . . . 14
apa.glm . . . 15
apa.htest . . . 18
apa.linearHypothesis . . . 19
apa.list . . . 19
apa.r . . . 20
apa.roc.test . . . 21
apa.summary.merMod . . . 22
apa.summary_emm . . . 23
apa.ttest . . . 24
apa_binom_mean_conf . . . 24
apa_ezStats . . . 25
apa_format_mean_conf . . . 26
apa_mean_and_t . . . 27
apa_mean_conf . . . 28
apa_mean_sd . . . 29
base_breaks . . . 30
binom_ci . . . 31
drop_empty_cols . . . 31
f_round . . . 32
format_results . . . 33
get_adjusted_ci . . . 33
ins_lmer . . . 36
lengthu . . . 37
lmer_fixef . . . 37
load_libs . . . 38
mean_nn . . . 38
mean_round . . . 39
memory_noise . . . 40
omit_zeroes . . . 41

paste_and 41
plot_pointrange 42
round_p 45
scale_y_exp 46
sd_nn 47
sd_round 48
seq_mean 48
sum_nn 49
table_mean_conf 49
table_mean_conf_by 50

Index **51**

`adv_cut` *Cut a numeric variable into groups (bins) with advanced options*

Description

Cut a numeric variable into groups (bins) with advanced options

Usage

```
adv_cut(
  x,
  ncuts = NULL,
  eq_groups = FALSE,
  cuts = NULL,
  num_labels = FALSE,
  labels = NULL,
  include_oob = TRUE,
  labels_at_means = TRUE,
  label_pairs_format = "[%.2f, %.2f]",
  ...
)
```

Arguments

`x`	vector of numeric values to cut into groups
`ncuts`	number of cuts (default: NULL)
`eq_groups`	should the groups be equal (default: FALSE)
`cuts`	where to put the cuts (default: NULL), not used if ncuts is used
`num_labels`	should the labels be transformed into numbers (default: FALSE)
`labels`	a vector of labels to use for the groups (default: NULL)
`include_oob`	include values outside of the boundaries provided in `cuts` (default: TRUE)
`labels_at_means`	should labels be created as means between cuts (TRUE) or as pairs of cuts (FALSE)

`label_pairs_format`
: formatting string to use when labels are generated from pairs of cuts (default: [%.2f, %.2f])

`...`
: other parameters passed to base::cut

If `ncuts` is used, then the variable is cut into N cuts either of equal group size (eq_groups = TRUE) or equally distant from each other (eq_groups = FALSE). If `labels` are not provided, they are generated as means between cuts if labels_at_means is TRUE.

Value

A factor assigning each element of `x` to a bin, or a numeric vector of bin labels when `num_labels = TRUE`. The result has the same length as `x`.

Examples

```
set.seed(1)
x <- sample(1:100, 20)
sort(x)

adv_cut(x, ncuts = 5)
adv_cut(x, ncuts = 5, eq_groups = TRUE)
adv_cut(x, ncuts = 5, eq_groups = TRUE, num_labels = TRUE)
adv_cut(x, ncuts = 5, eq_groups = TRUE, labels_at_means = FALSE)
adv_cut(x, cuts = seq(0, 100, by = 20))
adv_cut(x, cuts = seq(0, 100, by = 20), labels_at_means = FALSE)
adv_cut(x, cuts = seq(0, 100, by = 20),
           labels_at_means = FALSE, label_pairs_format = "[%i, %i]")
```

---

aggr2 *Double aggregation*

Description

Aggregates value twice providing mean of means, SD of SDs, etc.

Usage

```
aggr2(x, by, fun, ...)
```

Arguments

`x`	value to aggregate
`by`	vector to aggregate by (e.g., ID of participant)
`fun`	function to apply
`...`	additional parameters passed to fun

Value

The object returned by `fun` when applied to the group summaries. Its class depends on `fun`; scalar numeric summary functions such as base::mean return one numeric value.

Examples

```
x <- rnorm(100)
id <- rep(1:10, each = 10)

aggregate(x ~ id, FUN = mean)
aggr2(x, id, mean)
```

---

apa *Format statistical results in APA style*

Description

The `apa` function provides a convenient way to format statistical results according to American Psychological Association (APA) style guidelines. It dispatches to various S3 methods depending on the class of the input object.

Usage

```
apa(obj, ...)
```

Arguments

`obj`	An object to format in APA style
`...`	Additional arguments passed to specific methods

Details

This function supports various statistical objects directly:

- apa.ttest for stats::t.test and apa.r for stats::cor.test (both with class `htest`)
- apa.aov for stats::aov, apa.anova for car::Anova, and apa.afex_aov for afex::aov_ez, afex::aov_car, or afex::aov_4
- apa.glm for stats::glm, stats::lm, lme4::lmer, lme4::glmer, and lmerTest::lmer; apa.lmert remains available for summary objects from lmerTest::lmer
- apa.brmsfit for brms::brm and apa.BFBayesFactor for BayesFactor::anovaBF
- apa.chisq.test for stats::chisq.test
- apa.dip.test for diptest::dip.test
- apa.roc.test for pROC::roc.test

In addition, there are standalone utility functions for formatting:

- apa_mean_sd: Format means and standard deviations
- apa_mean_conf: Format means and confidence intervals
- apa_binom_mean_conf: Format binomial means and confidence intervals
- apa_mean_and_t: Compare means with t-tests and format results
- table_mean_conf: Get formatted means and CIs (for tables)
- table_mean_conf_by: Get formatted means and CIs by group (for tables)

Value

The result of the method for `obj`: usually a character vector of APA-formatted statistical results. Regression methods can return a `data.frame` of coefficient summaries, and ANOVA methods can return a `data.frame` with formatted text appended when requested. See each method's documentation for its columns and selection options.

Examples

```
# t-test example
t_res <- t.test(rnorm(30), rnorm(30, 0.5))
apa(t_res)

# Correlation example
cor_res <- cor.test(mtcars$mpg, mtcars$wt)
apa(cor_res)
```

`apa.afex_aov` *Describe afex ANOVA results*

Description

Provides formatted string like *F*(num.Df, den.Df) = ..., *p* ..., eta2 = ... based on afex::aov_ez, afex::aov_car, or afex::aov_4 results.

Usage

```
## S3 method for class 'afex_aov'
apa(
  obj,
  term,
  include_eta = TRUE,
  eta_digits = 2,
  f_digits = 2,
  df_digits = 0,
  append_to_table = FALSE,
  ...
)
```

Arguments

obj	afex ANOVA object
term	model term to describe (a string with the term name or its sequential number)
include_eta	add generalized eta^2 for the model (default: TRUE)
eta_digits	number of digits to use for eta^2 (default: 2)
f_digits	number of digits to use for F (default: 2)
df_digits	number of digits to use for df (default: 0)
append_to_table	should the results be added to the afex ANOVA table (default: FALSE)
...	other parameters passed to format_results

Value

A character vector reporting F statistics, degrees of freedom, p-values, and optional generalized eta-squared. With `append_to_table = TRUE`, a `data.frame` of selected ANOVA rows with an added `res` column containing the formatted results.

Examples

```
if (requireNamespace("afex", quietly = TRUE)) {
  data(stroop, package = "afex")

  afex_res <- afex::aov_ez(
    id = "pno",
    dv = "rt",
    within = "condition",
    data = stroop[!is.na(stroop$rt), ],
    fun_aggregate = mean
  )

  print(apa(afex_res, "condition"))
  print(apa(afex_res, "condition", eta_digits = 3))
  print(apa(afex_res, "condition", include_eta = FALSE))
}
```

---

apa.anova *Describe the results of ANOVA models or model comparisons*

Description

Describe the results of ANOVA models or model comparisons

Usage

```
## S3 method for class 'anova'
apa(obj, term = 2, f.digits = 2, ...)
```

Arguments

obj	ANOVA results from car::Anova or stats::anova
term	model term to describe (a string with the term name or its sequential number, default: 2)
f.digits	number of digits in the results (default: 2)
...	other parameters passed to format_results

Details

When using model comparison version, term can only be a number.

Value

Formatted string with F (or chi2), df, and p

Examples

```
if (requireNamespace("carData", quietly = TRUE)) {
  # Model comparison version
  mod1 <- lm(conformity ~ 1, data = carData::Moore)
  mod2 <- lm(conformity ~ fcategory, data = carData::Moore,
             contrasts = list(fcategory = contr.sum))
  mod3 <- lm(conformity ~ fcategory * partner.status, data = carData::Moore,
             contrasts = list(fcategory = contr.sum, partner.status = contr.sum))
  anova_obj <- anova(mod1, mod2, mod3)

  print(apa(anova_obj))
  print(apa(anova_obj, 3))

  # car::Anova version
  if (requireNamespace("car", quietly = TRUE)) {
    mod <- lm(conformity ~ fcategory * partner.status, data = carData::Moore,
              contrasts = list(fcategory = contr.sum, partner.status = contr.sum))
    afit <- car::Anova(mod)

    print(apa(afit, "fcategory"))
    print(apa(afit, 2, 4))
  }
}
```

---

apa.anova.merMod *Describe lmerTest anova results*

Description

Describe lmerTest anova results

Usage

```
## S3 method for class 'anova.merMod'
apa(obj, term, f.digits = 2, ...)
```

Arguments

obj	lmerTest anova results
term	model term to describe (a string with the term name or its sequential number)
f.digits	decimal digits for F value
...	other parameters passed to format_results

Value

formatted string describing the results of anova

Examples

```
if (requireNamespace("lmerTest", quietly = TRUE)) {
  # Sample data
  data(sleepstudy, package = "lme4")

  # Fit a mixed-effects model
  fit <- lmerTest::lmer(Reaction ~ Days + (1 + Days | Subject), sleepstudy)

  # ANOVA table
  afit <- anova(fit)

  # Format results in APA style
  print(apa(afit, "Days"))
  print(apa(afit, "Days", f.digits = 3))
}
```

---

apa.aov *Describe stats::aov results*

Description

`stats::aov()` models containing `Error()` return an `aovlist` object, which cannot be passed to `car::Anova()`. This method extracts the ANOVA tables from each error stratum and reports effects using the residual degrees of freedom from the same stratum.

Usage

```
## S3 method for class 'aov'
apa(obj, term, sstype = 2, ...)

## S3 method for class 'aovlist'
apa(obj, term = NULL, f.digits = 2, ...)
```

Arguments

obj	fitted stats::aov model with an Error() term
term	model term to describe. A character term name or sequential number can be supplied; if NULL, all testable effects are returned.
sstype	anova SS type (e.g., 2 or 3)
...	other parameters passed to format_results
f.digits	number of digits for the F value

Value

formatted string with F(df_numerator, df_denominator) = F_value, p =/< p_value

one or more formatted strings with F, numerator and denominator df, and p values

Examples

```
if (requireNamespace("car", quietly = TRUE)) {
  # Using the mtcars dataset
  fit <- aov(mpg ~ cyl * am, data = mtcars)
  print(apa(fit, 'cyl'))
  print(apa(fit, 'am'))
  print(apa(fit, 'cyl:am'))

  # Using a different SS type
  print(apa(fit, 'cyl', sstype = 3))
}
fit_rm <- aov(yield ~ N * P * K + Error(block), data = npk)
apa(fit_rm, "N")
apa(fit_rm)
```

---

apa.BFBayesFactor *Describe BayesFactor results*

Description

Describe BayesFactor results

Usage

```
## S3 method for class 'BFBayesFactor'
apa(obj, digits = 2, top_limit = 10000, convert_to_power = TRUE, ...)
```

Arguments

obj	an object of BayesFactor class BFBayesFactor
digits	number of digits to use
top_limit	numbers above that limit (or below the digits limit) will be converted to exponential notation (if convert_to_power is TRUE)
convert_to_power	enable or disable converting of very small or very large numbers to exponential notation
...	other parameters passed to format_results

Value

A character result reporting the Bayes factor, optionally in scientific notation for very large or small values.

Note

Code for converting to exponential notation is based on https://dankelley.github.io/r/2015/03/22/scinot.html

Examples

```
if (requireNamespace("BayesFactor", quietly = TRUE)) {
  bf <- BayesFactor::ttestBF(x = c(0.8, 1.1, 0.9, 1.5, 1.2, 0.7))
  print(apa(bf))
  print(apa(bf, top_limit = 10))
}
```

---

apa.brmsfit *Describe brms model results*

Description

Describe brms model results

Usage

```
## S3 method for class 'brmsfit'
apa(
  obj,
  term,
  trans = NULL,
  digits = 2,
  eff.size = FALSE,
  eff.size.type = "r",
  nsamples = 100,
  ci.type = "HPDI",
  ...
)
```

Arguments

obj	model object from brms::brm
term	model term to describe
trans	an optional function to transform the results to another scale (default: NULL)
digits	number of digits in the output
eff.size	string describing how to compute an effect size (currently, either 'fe_to_all', 'part_fe', or 'part_fe_re')
eff.size.type	type of the effect size ('r' or 'r2')
nsamples	number of samples to use for the effect size computations
ci.type	type of intervals to use (currently, all that is not HPDI is treated as ETI using bayestestR::eti)
...	other parameters passed to format_results

Value

A character result reporting the posterior mean coefficient, its 95% interval, and optionally an effect-size estimate.

Examples

```
if (requireNamespace("brms", quietly = TRUE) &&
    requireNamespace("bayestestR", quietly = TRUE)) {
  # Generate sample data
  set.seed(42)
  x <- seq(-1, 1, length.out = 30)
  y <- 1 + 2 * x + rnorm(30, sd = 0.5)

  # Fit a Bayesian regression model
  fit <- brms::brm(y ~ x, data = data.frame(x, y), backend = "rstan",
                   chains = 1, cores = 1, iter = 500, seed = 42, refresh = 0)
  print(apa(fit, "x"))

  # Convert x to another scale for output
  print(apa(fit, "x", trans = function(val) val + 5))
}
```

---

apa.chisq.test *Format chi-square test results in APA style*

Description

Format chi-square test results in APA style

Usage

```
## S3 method for class 'chisq.test'
apa(obj, v = TRUE, addN = TRUE, ...)
```

Arguments

obj	A result from stats::chisq.test
v	Add Cramer's V (default: TRUE)
addN	Add N (default: TRUE)
...	Other parameters passed to format_results

Value

Formatted results string

Examples

```
# Create a contingency table
M <- as.table(rbind(c(762, 327, 468), c(484, 239, 477)))
dimnames(M) <- list(
  gender = c("F", "M"),
  party = c("Democrat", "Independent", "Republican")
)
chi_result <- chisq.test(M)
apa(chi_result)
```

---

apa.dip.test *Describe Hartigans' dip test results*

Description

Describe Hartigans' dip test results

Usage

```
## S3 method for class 'dip.test'
apa(obj, ...)
```

Arguments

obj	a result from diptest::dip.test
...	other parameters passed to format_results

Value

formatted results string

Examples

```
if (requireNamespace("diptest", quietly = TRUE)) {
  # Generate bimodal data
  bimodal_data <- c(rnorm(100, -2, 1), rnorm(100, 2, 1))

  # Perform dip test
  dip_result <- diptest::dip.test(bimodal_data)

  # Format results in APA style
  apa(dip_result)
}
```

`apa.ezANOVA` *Describe ezANOVA results*

Description

Provides formatted string like *F*(DFn, DFd) = ..., *p* ..., eta2 = ... based on ezANOVA results

Usage

```
## S3 method for class 'ezANOVA'
apa(
  obj,
  term,
  include_eta = TRUE,
  spher_corr = TRUE,
  eta_digits = 2,
  f_digits = 2,
  df_digits = 0,
  append_to_table = FALSE,
  ...
)
```

Arguments

`obj`	ezANOVA object from ez::ezANOVA
`term`	model term to describe (a string with the term name or its sequential number)
`include_eta`	add eta^2 for the model (default: TRUE)
`spher_corr`	use sphericity corrections (default: TRUE)
`eta_digits`	number of digits to use for eta^2 (default: 2)
`f_digits`	number of digits to use for F (default: 2)
`df_digits`	number of digits to use for df (default: 0)
`append_to_table`	should the results be added to the original ezANOVA table (default: FALSE)
`...`	other parameters passed to format_results

Value

A character vector reporting F statistics, degrees of freedom, p-values, and optional generalized eta-squared. With `append_to_table = TRUE`, a `data.frame` of selected ANOVA rows with an added `res` column containing the formatted results.

Examples

```
if (requireNamespace("ez", quietly = TRUE) && requireNamespace("afex", quietly = TRUE)) {
  # Using the Stroop dataset from afex package
  data(stroop, package = "afex")

  # Run ezANOVA
  ez_res <- ez::ezANOVA(
    data = stroop[!is.na(stroop$rt),],
    dv = rt,
    wid = pno,
    within = condition,
    detailed = TRUE
  )

  # Format results in APA style
  print(apa(ez_res, "condition"))
  print(apa(ez_res, "condition", eta_digits = 3))
  print(apa(ez_res, "condition", include_eta = FALSE))
}
```

`apa.glm` *Describe regression model (GLM, GLMer, lm, lm.circular, ...)*

Description

Describe regression model (GLM, GLMer, lm, lm.circular, ...)

Usage

```
## S3 method for class 'glm'
apa(
  obj,
  term = NULL,
  dtype = 1,
  b.digits = 2,
  t.digits = 2,
  test.df = FALSE,
  p.as.number = FALSE,
  term.pattern = NULL,
  eff.size = FALSE,
  adj.digits = FALSE,
  ...
```

```
)

## S3 method for class 'lm'
apa(
  obj,
  term = NULL,
  dtype = 1,
  b.digits = 2,
  t.digits = 2,
  test.df = FALSE,
  p.as.number = FALSE,
  term.pattern = NULL,
  eff.size = FALSE,
  adj.digits = FALSE,
  ...
)

## S3 method for class 'glmerMod'
apa(
  obj,
  term = NULL,
  dtype = 1,
  b.digits = 2,
  t.digits = 2,
  test.df = FALSE,
  p.as.number = FALSE,
  term.pattern = NULL,
  eff.size = FALSE,
  adj.digits = FALSE,
  ...
)

## S3 method for class 'lmerModLmerTest'
apa(
  obj,
  term = NULL,
  dtype = 1,
  b.digits = 2,
  t.digits = 2,
  test.df = FALSE,
  p.as.number = FALSE,
  term.pattern = NULL,
  eff.size = FALSE,
  adj.digits = FALSE,
  ...
)
```

Arguments

obj	model object from stats::glm, stats::lm, lme4::lmer, lme4::glmer, lmerTest::lmer, etc.
term	model term to describe (a string with the term name or its sequential number); if NULL, returns a formatted summary for all coefficients
dtype	description type (1: t, p; 2: B(SE), p; 3: B, SE, t, p; or other: B (SE), t)
b.digits	how many digits to use for *B* and *SE*
t.digits	how many digits to use for *t*
test.df	should we include degrees of freedom in description?
p.as.number	should the p-values be transformed to numbers (TRUE) or shown as strings (FALSE)?
term.pattern	return only the model terms matching the regex pattern (grepl is used)
eff.size	should we include effect size (currently implemented only for simple regression)?
adj.digits	automatically adjusts digits so that B or SE would not show up as "0.00"
...	other parameters passed to format_results

Value

A character result for a selected term. Otherwise, a data.frame with columns B (coefficient estimate), SE (standard error), Stat (test statistic), p (p-value), eff (term name), and str (complete APA-formatted result). The summary columns are character vectors except that p is numeric when p.as.number = TRUE. term.pattern filters the rows of this data frame.

Examples

```
# Using the Animals dataset from MASS package
Animals <- MASS::Animals
Animals$body <- log(Animals$body)
Animals$brain <- log(Animals$brain)

# Fit a linear model
fit <- lm(brain ~ body, Animals)

# Format results in different styles
apa(fit, "body")
apa(fit, "(Intercept)")
apa(fit, "body", 2)   # B(SE), p format
apa(fit, "body", 3)   # B, SE, t, p format
apa(fit, "body", 4)   # B(SE), t format
apa(fit, "body", 3, test.df = TRUE)  # Include df in the output

# Full model summary for all coefficients
apa(fit)

# With effect size (requires rockchalk package)
if (requireNamespace("rockchalk", quietly = TRUE)) {
```

```
  print(apa(fit, "body", 4, eff.size = TRUE))
}

if (requireNamespace("lmerTest", quietly = TRUE)) {
  fm <- lmerTest::lmer(Reaction ~ Days + (Days | Subject), lme4::sleepstudy)
  print(apa(fm, "Days", test.df = TRUE))
}

if (requireNamespace("lme4", quietly = TRUE)) {
  gm <- lme4::glmer(
    cbind(incidence, size - incidence) ~ period + (1 | herd),
    data = lme4::cbpp,
    family = binomial
  )
  print(apa(gm, "period2"))
}
```

`apa.htest` *Format t-test results in APA style*

Description

Format t-test results in APA style

Usage

```
## S3 method for class 'htest'
apa(obj, show.mean = FALSE, abs = FALSE, ...)
```

Arguments

`obj`	An object from stats::t.test
`show.mean`	Include mean value in results (useful for one-sample test)
`abs`	Show the absolute value of t-statistic
`...`	Other arguments passed to format_results

Value

A string with t-test value, degrees of freedom, p-value, and optionally the mean with CI

Examples

```
# One-sample t-test
t_res <- t.test(rnorm(20, mean = -10, sd = 2))
apa(t_res)
apa(t_res, show.mean = TRUE)

# Two-sample t-test
t_two <- t.test(rnorm(20), rnorm(20, mean = 0.8))
apa(t_two)
```

apa.linearHypothesis *Describe linearHypothesis test results*

Description

Describe linearHypothesis test results

Usage

```
## S3 method for class 'linearHypothesis'
apa(obj, ...)
```

Arguments

obj	hypothesis from car::linearHypothesis
...	additional parameters passed to format_results

Value

A character result reporting the chi-squared or F statistic, degrees of freedom, and p-value for the linear-hypothesis test.

Examples

```
if (requireNamespace("car", quietly = TRUE) &&
    requireNamespace("carData", quietly = TRUE)) {
  # Create a linear model
  mod.davis <- lm(weight ~ repwt, data = carData::Davis)

  # Test hypotheses
  res <- car::linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"))
  print(apa(res))

  # Test using Chi-square
 res.chi <- car::linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"), test = "Chisq")
  print(apa(res.chi))
}
```

apa.list *Format results from list objects in APA style*

Description

This function detects the type of statistical results stored in a list object and redirects to the appropriate APA formatting method. Currently supports ezANOVA results (apa.ezANOVA).

Usage

```
## S3 method for class 'list'
apa(obj, ...)
```

Arguments

obj	A list object containing statistical results
...	Additional arguments passed to the appropriate method

Value

A character vector of formatted ANOVA results, or a data.frame containing selected ANOVA rows and a res column when append_to_table = TRUE. See apa.ezANOVA.

---

apa.r *Format correlation test results in APA style*

Description

Format correlation test results in APA style

Usage

```
## S3 method for class 'r'
apa(obj, ...)
```

Arguments

obj	An object from stats::cor.test
...	Other arguments passed to format_results

Value

A string with correlation coefficient, degrees of freedom, and p-value

Examples

```
# Pearson correlation
x <- rnorm(40)
y <- x * 0.6 + rnorm(40, 0, 0.8)
rc <- cor.test(x, y)
apa(rc)
```

`apa.roc.test` *Describe differences between ROC curves*

Description

Describe differences between ROC curves

Usage

```
## S3 method for class 'roc.test'
apa(obj, ...)

## S3 method for class 'roc.diff'
apa(obj, ...)
```

Arguments

`obj`	a difference between the ROC curves from pROC::roc.test
`...`	Additional arguments (currently unused)

Value

A character result reporting the test statistic and p-value for the comparison between ROC curves.

Examples

```
if (requireNamespace("pROC", quietly = TRUE)) {
  # Create sample data
  set.seed(42)
  n <- 100
  group <- factor(sample(c(0, 1), n, replace = TRUE))
  test1 <- rnorm(n, mean = 1 * (as.numeric(group) - 1))
  test2 <- rnorm(n, mean = 0.7 * (as.numeric(group) - 1))

  # Create ROC curves
  roc1 <- pROC::roc(group, test1)
  roc2 <- pROC::roc(group, test2)

  # Test difference between ROC curves
  roc_diff <- pROC::roc.test(roc1, roc2)

  # Format results in APA style
  apa(roc_diff)
}
```

`apa.summary.merMod` *Describe lmerTest results*

Description

Note that this function uses *summary* object from lmerTest::lmer model and not the model itself (see examples). Otherwise p-values will be computed during the call and everything will be very slow.

Usage

```
## S3 method for class 'summary.merMod'
apa(obj, factor, dtype = "t", ...)

## S3 method for class 'lmert'
apa(obj, factor, dtype = "t", ...)
```

Arguments

`obj`	*summary* object from lmerTest::lmer model
`factor`	name or number of the factor that needs to be described
`dtype`	description type ("B"/"t")
`...`	other parameters passed to format_results

Value

A character vector reporting the selected fixed-effect estimate, standard error, test statistic, degrees of freedom when available, and p-value.

Examples

```
if (requireNamespace("lmerTest", quietly = TRUE)) {
  # Fit a mixed-effects model with p-values
  fm <- lmerTest::lmer(Reaction ~ Days + (Days | Subject), lme4::sleepstudy)
  fms <- summary(fm)

  # Format results in APA style
  print(apa(fms, "Days"))
  print(apa(fms, "(Intercept)"))
  print(apa(fms, 2))
  print(apa(fms, "Days", "B"))
}
```

`apa.summary_emm` *Describe contrasts created by emmeans*

Description

Describe contrasts created by emmeans

Usage

```
## S3 method for class 'summary_emm'
apa(obj, term, dtype = "B", df = FALSE, ...)

## S3 method for class 'emmeans'
apa(obj, term, dtype = "B", df = FALSE, ...)
```

Arguments

`obj`	summary object from emmeans::contrast
`term`	contrast number(s)
`dtype`	description type, "t", "B", or any other letter
`df`	include DF in t-test description (default: False)
`...`	other parameters passed to format_results

Value

string with formatted results

Examples

```
if (requireNamespace("emmeans", quietly = TRUE)) {
  # Create a model
  warp.lm <- lm(breaks ~ wool * tension, data = warpbreaks)

  # Create estimated marginal means
  warp.emm <- emmeans::emmeans(warp.lm, ~ tension | wool)

  # Create contrasts
  sum_contr <- summary(emmeans::contrast(warp.emm, "trt.vs.ctrl"))

  # Format results in APA style
  print(apa(sum_contr, 1))
  print(apa(sum_contr, 3))
  print(apa(sum_contr, 3, dtype = "t"))
  print(apa(sum_contr, 3, dtype = "c"))
  print(apa(sum_contr, 3, dtype = "c", df = TRUE))
}
```

apa.ttest *Format t-test results in APA style*

Description

Format t-test results in APA style

Usage

```
## S3 method for class 'ttest'
apa(obj, show.mean = FALSE, abs = FALSE, ...)
```

Arguments

obj	An object from stats::t.test
show.mean	Include mean value in results (useful for one-sample test)
abs	Show the absolute value of t-statistic
...	Other arguments passed to format_results

Value

A string with t-test value, degrees of freedom, p-value, and optionally the mean with CI

Examples

```
# One-sample t-test
t_res <- t.test(rnorm(20, mean = -10, sd = 2))
apa(t_res)
apa(t_res, show.mean = TRUE)
apa(t_res, show.mean = TRUE, abs = TRUE)

# Two-sample t-test
t_two <- t.test(rnorm(20), rnorm(20, mean = 0.8))
apa(t_two)
```

apa_binom_mean_conf *Format binomial proportions and confidence intervals in APA style*

Description

Format binomial proportions and confidence intervals in APA style

Usage

```
apa_binom_mean_conf(x, digits = 2, ...)
```

Arguments

x	A vector of zeros and ones
digits	Number of digits in results
...	Additional arguments passed to format_results

Value

A string with the mean and confidence interval in square brackets

Examples

```
# Generate binomial data
set.seed(123)
x <- rbinom(500, 1, prob = 0.7)

# Format results in APA style
apa_binom_mean_conf(x)
```

---

apa_ezStats *Describe ezStats results*

Description

Returns formatted string with mean and SD from ezStats object

Usage

```
apa_ezStats(obj, term = 1, ...)
```

Arguments

obj	data.frame returned by ez::ezStats
term	model term to describe (a string with the term name or its sequential number)
...	other parameters passed to apa_mean_sd If a string is used for a term and there is more than one factor, "X:Y:Z" format is assumed (value in first column : value in the second, and so on). If no term is supplied, the first row is used. So if you need to select a term based on two or more variables, you can also just filter ezStats result beforehand (or use a row number as a term).

Value

string with formatted results

Examples

```
if (requireNamespace("ez", quietly = TRUE) && requireNamespace("afex", quietly = TRUE)) {
  # Using the Stroop dataset from afex package
  data(stroop, package = "afex")
  # Get descriptive statistics

  ezstats_res <- ez::ezStats(
    data = stroop[!is.na(stroop$rt),],
    dv = rt,
    wid = pno,
    within = .(congruency)
  )

  # Format results in APA style
  print(apa_ezStats(ezstats_res, 1))
  print(apa_ezStats(ezstats_res, "incongruent"))
  print(apa_ezStats(ezstats_res, 1, digits = 3))
}
```

---

apa_format_mean_conf *Format mean and confidence interval values into APA-style string*

Description

Format mean and confidence interval values into APA-style string

Usage

```
apa_format_mean_conf(
  mean_val,
  lower_ci,
  upper_ci,
  addCI = FALSE,
  digits = 2,
  ...
)
```

Arguments

mean_val	The mean value
lower_ci	Lower confidence interval bound
upper_ci	Upper confidence interval bound
addCI	Add "95% CI =" prefix
digits	Number of digits to use
...	Other arguments passed to format_results

Value

A formatted string with mean and confidence intervals

Examples

```
# Example usage
mean_val <- 5.67
lower_ci <- 4.56
upper_ci <- 6.78
# Format mean and CI in APA style
apa_format_mean_conf(mean_val, lower_ci, upper_ci)
# Format mean and CI with additional CI prefix
apa_format_mean_conf(mean_val, lower_ci, upper_ci, addCI = TRUE)
# Format mean and CI with custom number of digits
apa_format_mean_conf(mean_val, lower_ci, upper_ci, digits = 3)
```

---

apa_mean_and_t *Compare means with t-tests and format in APA style*

Description

Compare means with t-tests and format in APA style

Usage

```
apa_mean_and_t(
  x,
  by,
  which.mean = 1,
  digits = 2,
  paired = FALSE,
  eff.size = FALSE,
  abs = FALSE,
  aggregate_by = NULL,
  transform.means = NULL,
  ...
)
```

Arguments

x	Dependent variable (numeric vector)
by	Independent variable (factor with 2 levels)
which.mean	Which means to show (0=none, 1=first, 2=second, 3=both)
digits	Number of digits in results (default: 2)
paired	Should it be a paired test (default: FALSE)

eff.size	Should we include effect size (default: FALSE)
abs	Should we show the absolute value if the t-test (TRUE) or keep its sign (FALSE, default)
aggregate_by	Do the aggregation by the third variable(s): either NULL (default), a single vector variable, or a list of variables to aggregate by.
transform.means	A function to transform means and confidence intervals (default: NULL)
...	Additional arguments passed to format_results

Value

A formatted string with t-test results in APA style

Examples

```
# Using simulated data
rt <- rnorm(100)
gr <- factor(rep(c("A", "B"), each = 50))
apa_mean_and_t(rt, gr)
apa_mean_and_t(rt, gr, which.mean = 3)
apa_mean_and_t(rt, gr, eff.size = TRUE)
```

---

apa_mean_conf *Format means and confidence intervals in APA style*

Description

Format means and confidence intervals in APA style

Usage

```
apa_mean_conf(
  x,
  bootCI = TRUE,
  addCI = FALSE,
  digits = 2,
  transform.means = NULL,
  ...
)
```

Arguments

x	A numeric vector
bootCI	Use bootstrapped CI (TRUE) or normal approximation (FALSE)
addCI	Add "95% CI =" prefix
digits	Number of digits to use

transform.means
: An optional function to transform the means and CI to another scale

...
: Other arguments passed to format_results

Value

A string with a mean followed by confidence intervals in square brackets

Examples

```
x <- runif(100, 0, 50)
apa_mean_conf(x)
apa_mean_conf(x, bootCI = FALSE)  # Normal approximation instead of bootstrap
```

---

apa_mean_sd *Format means and standard deviations in APA style*

Description

Format means and standard deviations in APA style

Usage

```
apa_mean_sd(
  x = NULL,
  m = NULL,
  sd = NULL,
  digits = 2,
  dtype = "p",
  m_units = "",
  sd_units = "",
  ...
)
```

Arguments

x
: A numeric vector

m
: Pre-computed mean (optional)

sd
: Pre-computed standard deviation (optional)

digits
: Number of digits for rounding

dtype
: Format type: "p" for parentheses or "c" for comma

m_units
: Units for mean (e.g., "°" for degrees)

sd_units
: Units for standard deviation

...
: Additional arguments passed to format_results

Value

A formatted string with mean and standard deviation in APA style

Examples

```
x <- rnorm(1000, 50, 25)
apa_mean_sd(x)
apa_mean_sd(m = 50.2, sd = 9.8)  # With pre-computed values
apa_mean_sd(x, dtype = "c")  # Comma format instead of parentheses
```

---

base_breaks *Tufte-like breaks for axes*

Description

Tufte-like breaks for axes

Usage

```
base_breaks(x, scale = "x", addSegment = TRUE, ...)

base_breaks_x(x, addSegment = TRUE, ...)

base_breaks_y(x, addSegment = TRUE, ...)
```

Arguments

x	vector of numbers to create breaks from
scale	scale (x or y) to create breaks for
addSegment	should we add a line to the scale? (TRUE/FALSE)
...	other parameters passed to scale_(x or y)_continuous

Value

A continuous position scale (class `ScaleContinuousPosition`) with breaks spanning `x`. When `addSegment = TRUE`, a list containing an axis-segment layer and the scale, suitable for adding to a ggplot2 plot.

Functions

- `base_breaks_x()`: Tufte-like breaks for X axis
- `base_breaks_y()`: Tufte-like breaks for Y axis

Examples

```
p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  theme_minimal()

p
p + base_breaks(mtcars$wt, scale = "x") +
  base_breaks(mtcars$mpg, scale = "y")
```

---

binom_ci *Binomial confidence interval as a vector*

Description

Binomial confidence interval as a vector

Usage

```
binom_ci(x)
```

Arguments

x	a vector of 0 and 1

Value

A named numeric vector: `y` is the estimated proportion, `ymin` and `ymax` are its 95% confidence limits, and `len` is the sample size.

Examples

```
binom_ci(rbinom(500, 1, prob = 0.7))
```

---

drop_empty_cols *Drop empty columns from a data.frame*

Description

Drop empty (consisting of NA only) columns from a data.frame. Based on https://stackoverflow.com/a/2644009/1344028

Usage

```
drop_empty_cols(df)
```

Arguments

df	a data.frame

Value

data.frame without empty columns

Examples

```
df <- data.frame(x = rnorm(20), y = rep("A", 20), z = rep(NA, 20))
str(df)
df <- drop_empty_cols(df)
str(df)
```

---

f_round *Formatted rounding*

Description

Formatted rounding

Usage

```
f_round(x, digits = 2, strip.lead.zeros = FALSE)
```

Arguments

x	A number
digits	Number of decimal digits to keep
strip.lead.zeros	remove zero before decimal point (default is false)

Value

A character vector containing x rounded to the specified number of decimal places, optionally omitting zeros before decimal points.

Examples

```
f_round(5.8242)
f_round(5.8251)
f_round(5.82999, digits = 3)
f_round(5.82999, digits = 4)
```

format_results *Format results*

Description

Internal function used to convert latex-formatted results to pandoc style.

Usage

```
format_results(res_str, type = "pandoc")
```

Arguments

res_str	text
type	'pandoc', 'latex', or 'plotmath' (the latter is very poorly implemented)

Value

A character vector containing `res_str` in the requested notation: LaTeX markup, Pandoc emphasis, or text for plotmath expressions.

get_adjusted_ci *Get adjusted confidence intervals or standard errors*

Description

Computes grouped summary statistics and precision intervals for between- subject, within-subject, and mixed designs. With no within-subject factors, ordinary independent-observation intervals are computed. Supplying a subject identifier in a between-subject design first aggregates repeated rows within subject-by-group cells.

get_superb_ci() is retained for backward compatibility. New code should use `get_adjusted_ci()`. Its within-subject results are intended to remain numerically compatible with the former implementation backed by superb.

Usage

```
get_adjusted_ci(
  data,
  value_var,
  within = NULL,
  between = NULL,
  wid = NULL,
  adjustments = list(purpose = "single", decorrelation = NULL),
  errorbar = "CI",
  gamma = 0.95,
```

```
  drop_NA_subj = FALSE,
  drop_missing_levels = TRUE,
  aggr_fun = mean,
  debug = FALSE,
  ...
)

get_superb_ci(
  data,
  wid,
  within,
  value_var,
  between = NULL,
  adjustments = list(purpose = "single", decorrelation = "CM"),
  errorbar = "CI",
  drop_NA_subj = FALSE,
  drop_missing_levels = TRUE,
  aggr_fun = mean,
  debug = FALSE,
  ...
)
```

Arguments

data	dataframe to summarize
value_var	dependent variable (string)
within	within-subject variables (vector of strings; default: NULL). If supplied, wid must also be supplied.
between	between-subject/grouping variables (vector of strings; default: NULL)
wid	observational-unit/subject variable (string; default: NULL). Required for within-subject designs. For between-subject designs it is optional; if supplied, repeated rows are aggregated within subject/group cells before intervals are computed.
adjustments	adjustment settings. Supports purpose = "single" or "difference". decorrelation can be "CM", "CA", or "none" for within-subject designs and must be "none" for purely between-subject designs. If omitted, decorrelation defaults to "CM" when within is supplied and "none" otherwise. CA (correlation adjustment) multiplies interval widths by `sqrt(1 - rbar)`, where `rbar` is the mean Pearson correlation across pairs of within-subject conditions, computed separately within each between-subject group after cell aggregation and incomplete-unit removal. Centers are unchanged. CA requires at least two complete subjects and two conditions per group, with finite values and nonzero variance in every condition.
errorbar	"CI", "SE", "none", or a precision function. A precision function receives the condition vector and returns either one error-bar width or two interval limits.
gamma	confidence level for interval functions (default: 0.95)
drop_NA_subj	should subjects with missing within-subject cells be dropped? (default: FALSE)

drop_missing_levels
: should unused factor levels in within/between variables be dropped? (default: TRUE)

aggr_fun
: scalar summary function. For within-subject designs, and for between-subject designs with wid supplied, it is also used to aggregate repeated observations within observational-unit cells. Built-in precision formulas mirror superb for mean, median, var, sd, IQR, and the superb-style MAD, hmean, gmean, fisherskew, pearsonskew, and fisherkurtosis names. For other summary functions, supply errorbar as a precision function.

debug
: output additional debugging info (default: FALSE)

...
: additional parameters passed to apa_format_mean_conf

Details

The within-subject implementation and statistic-specific precision conventions are adapted from the superb framework and its R implementation (Cousineau, Goulet, & Harding, 2021). In particular, the Cousineau-Morey path follows the two-step centering and bias-correction algorithm used by superb. The code is implemented locally; apastats2 does not depend on or call superb.

Value

A `data.frame` with one row per group and the grouping variables, `center` (summary statistic), `lowerwidth` and `upperwidth` (signed interval offsets from the center), `lower_ci` and `upper_ci` (interval limits), and `descr` (formatted summary and interval text). With `errorbar = "none"`, both offsets are zero and both limits equal the center.

References

Cousineau, D., Goulet, M.-A., & Harding, B. (2021). Summary plots with adjusted error bars: The superb framework with an implementation in R. Advances in Methods and Practices in Psychological Science, 4(3). doi:10.1177/25152459211035109

Cousineau, D. (2005). Confidence intervals in within-subject designs: A simpler solution to Loftus and Masson's method. The Quantitative Methods for Psychology, 1(1), 42-45. doi:10.20982/tqmp.01.1.p042

Morey, R. D. (2008). Confidence intervals from normalized data: A correction to Cousineau (2005). The Quantitative Methods for Psychology, 4(2), 61-64. doi:10.20982/tqmp.04.2.p061

Examples

```
data(memory_noise)

# Between-subject intervals: Exp. 1A vs Exp. 1B
target_higher <- memory_noise[
  memory_noise$relative_noise == "target more noisy",
]
get_adjusted_ci(
  target_higher,
  value_var = "bias_percent",
  between = "experiment",
  wid = "participant"
```

```
)

# Within-subject Cousineau-Morey intervals for relative noise
get_adjusted_ci(
  memory_noise[memory_noise$experiment == "Exp. 1A", ],
  value_var = "bias_percent",
  within = "relative_noise",
  wid = "participant"
)

# Correlation-adjusted intervals
get_adjusted_ci(
  memory_noise[memory_noise$experiment == "Exp. 1A", ],
  value_var = "bias_percent", within = "relative_noise", wid = "participant",
  adjustments = list(decorrelation = "CA")
)

# Mixed design: experiment is between, relative noise is within
get_adjusted_ci(
  memory_noise,
  value_var = "bias_percent",
  within = "relative_noise",
  between = "experiment",
  wid = "participant"
)
```

`ins_lmer` *Describe lmer in-text*

Description

A shortcut for apa.glm(..., dtype=4)

Usage

```
ins_lmer(obj, term = NULL, digits = 2, adj.digits = TRUE)
```

Arguments

`obj`	LMER model from lme4::lmer
`term`	model term to describe (a string with the term name or its sequential number)
`digits`	number of digits for B and SD
`adj.digits`	automatically adjusts digits so that B would not show up as "0.00"

Value

A character result for a selected `term`, or a `data.frame` of coefficient summaries when `term = NULL`. The table has the columns documented in apa.glm; `str` contains estimates, standard errors, and test statistics formatted for in-text reporting.

Examples

```
if (requireNamespace("lme4", quietly = TRUE)) {
  # Fit a mixed-effects model
  fm <- lme4::lmer(Reaction ~ Days + (Days | Subject), lme4::sleepstudy)

  # Concise in-text reporting
  print(ins_lmer(fm, "Days"))
  print(ins_lmer(fm, "(Intercept)", digits = 1))
}
```

---

lengthu *Length of unique values*

Description

Counts unique values

Usage

```
lengthu(x)
```

Arguments

x	a vector

Value

A numeric scalar giving the number of distinct values in x.

Examples

```
x <- c(5, 7, 8, 9, 5, 7)
length(x)
lengthu(x)
```

---

lmer_fixef *Get nice matrix of fixed effects from lmer*

Description

Get nice matrix of fixed effects from lmer

Usage

```
lmer_fixef(fit.lmer)
```

Arguments

`fit.lmer`	fitted `lme4::lmer` model object

Value

A `data.frame` with one row per fixed-effect coefficient and columns `Estimate` (coefficient), `Std.Err` (standard error), and `t` (the ratio of the estimate to its standard error). Row names identify model terms.

---

load_libs *Quietly load libraries*

Description

Quietly load libraries

Usage

```
load_libs(libs)
```

Arguments

`libs`	a vector of libraries names

Value

Invisibly, a list of logical scalars, one per package in `libs`, indicating whether loading succeeded. Called for the side effect of loading and attaching the requested packages.

Examples

```
load_libs(c("ggplot2", "apastats2", "Hmisc"))
```

---

mean_nn *Mean with na.rm=TRUE*

Description

Mean with na.rm=TRUE

Usage

```
mean_nn(x, ...)
```

Arguments

x	a vector of numbers
...	other arguments passed to mean

Value

A numeric or complex scalar giving the mean of x, with missing values removed.

Examples

```
x <- c(NA, 10, 90)
mean(x)
mean_nn(x)
```

---

mean_round *Rounded mean*

Description

Mean rounded to the specified number of digits

Usage

```
mean_round(x, ...)
```

Arguments

x	a number
...	other arguments passed to f_round

Value

A character string containing the mean rounded to the specified number of decimal places.

Examples

```
mean_round(c(10, 99))
mean_round(c(10, 99, NA))
mean_round(c(10, 99), 2)
```

`memory_noise` *Memory bias under relative-noise manipulations*

Description

A compact participant-level dataset derived from Experiments 1A and 1B of Chetverikov and Hansmann-Roth (2026). In Exp. 1A the high-noise stimulus had SD = 20 degrees; in Exp. 1B it had SD = 45 degrees. Low-noise stimuli had SD = 5 degrees in both samples.

Usage

```
data(memory_noise)
```

Format

A data.frame with 70 observations and 6 variables.

Details

Only unequal-noise trials are included. Bias is estimated with the same weighted probability-density asymmetry approach used in the paper and averaged over target/non-target dissimilarities from 1 to 44 degrees, the range in which Experiments 1A and 1B showed the relative-noise interaction. Positive values indicate attraction toward the competing non-target and negative values indicate repulsion.

- participant. An anonymized participant identifier, unique across Experiments 1A and 1B.
- experiment. Between-subject sample: `"Exp. 1A"` or `"Exp. 1B"`.
- high_noise_sd. High stimulus-noise standard deviation used in the sample (20 or 45 degrees).
- relative_noise. Ordered within-subject factor: `"target less noisy"` or `"target more noisy"`.
- bias_percent. Density-asymmetry bias in percentage points. Positive values indicate attraction toward the non-target; negative values indicate repulsion.
- n_trials. Number of non-outlier unequal-noise trials contributing to the weighted bias curve.

Source

Derived from the public behavioral dataset at https://osf.io/kqb8t/. The package subset is based on the canonical CHR2026 representation used for model comparison.

References

Chetverikov, A., & Hansmann-Roth, S. (2026). Noise in Competing Representations Determines the Direction of Memory Biases. eLife, 15, RP111380. doi:10.7554/eLife.111380.1

omit_zeroes *Omit leading zero from number*

Description

Omit leading zero from number

Usage

```
omit_zeroes(x, digits = 2)
```

Arguments

x	A number
digits	Number of decimal digits to keep

Value

A character vector containing formatted values with the leading character removed, intended to omit the zero before a decimal point.

Examples

```
omit_zeroes(0.2312)
omit_zeroes(0.2312, digits = 3)
omit_zeroes("000.2312", digits = 1)
```

paste_and *Paste several strings, add 'and' before last*

Description

Paste several strings, add 'and' before last

Usage

```
paste_and(x, sep = ", ", suffix = "")
```

Arguments

x	vector of strings
sep	separator (is not used for only two groups)
suffix	suffix to append to each value before the separator

Value

A character string listing the values in x, with the requested separators, suffix, and final conjunction.

Examples

```
data(iris)
# get mean petal width and SD by group
res <- as.vector(by(iris$Sepal.Width, iris$Species, apa_mean_sd))
res
paste_and(res)
paste_and(res, sep = ";")

data(memory_noise)
# get bias and SD in the two experiment samples
res <- as.vector(by(
  memory_noise$bias_percent,
  memory_noise$experiment,
  apa_mean_sd
))
res
# no comma with two groups
paste_and(res)
paste_and(res, suffix = " percentage points")
```

---

plot_pointrange *Pointrange plot*

Description

Pointrange plot

Usage

```
plot_pointrange(
  data,
  mapping,
  pos = position_dodge(0.3),
  pointsize = I(3),
  linesize = I(1),
  pointfill = I("white"),
  pointshape = NULL,
  within_subj = FALSE,
  wid = "uid",
  bars = "ci",
  withinvars = NULL,
  betweenvars = NULL,
  x_as_numeric = FALSE,
```

```
  custom_geom_before = NULL,
  connecting_line = FALSE,
  pretty_breaks_y = FALSE,
  pretty_y_axis = FALSE,
  exp_y = FALSE,
  print_aggregated_data = FALSE,
  do_aggregate = FALSE,
  add_margin = FALSE,
  margin_label = "all",
  margin_x_vals = NULL,
  bars_instead_of_points = FALSE,
  geom_bar_params = list(),
  add_jitter = FALSE,
  individual_points_params = list(),
  drop_NA_subj = FALSE,
  design = "between",
  debug = FALSE
)
```

Arguments

data	the dataset to use
mapping	a list of plot aesthetics (e.g., aes(x = varA, y=varB))
pos	position adjustment function (e.g., position_dodge())
pointsize	size for the points (e.g., "I(1)")
linesize	size for the lines (e.g., "I(1)")
pointfill	fill for the points (e.g., "I('white')")
pointshape	shape for the points (e.g., "I(22)")
within_subj	should we use within-subject adjustment?
wid	within-subject ID variable
bars	should we use confidence intervals ("ci") or standard errors ("se")?
withinvars	within-subject variables
betweenvars	between-subject variables
x_as_numeric	should we treat x as numeric (otherwise it is treated as is)?
custom_geom_before	any custom geom to add before points and connecting line (geoms that should be on top of the pointrange can be added in a usual manner)
connecting_line	should we add connecting line (TRUE/FALSE)?
pretty_breaks_y	prettify y breaks
pretty_y_axis	prettify y axis
exp_y	exponentiate y (it's better to use this than scale_y_exp if pretty_breaks_y is used)

print_aggregated_data
: print aggregated data used for plotting to console

do_aggregate
: aggregate data by all conditions before plotting (FALSE)

add_margin
: add margin that would show the aggregate over all or subset of x-axis values (TRUE/FALSE)

margin_label
: margin label to use

margin_x_vals
: which levels of x-axis variable to aggregate over (NULL means all of them)

bars_instead_of_points
: use geom_bar instead of geom_point

geom_bar_params
: parameters for geom_bar if it is used

add_jitter
: add jittered individual data points

individual_points_params
: a list of parameters for the individual data points

drop_NA_subj
: drop subjects that have NAs after aggregation when computing within-subject errorbars

design
: design type to help avoid explicitly listing variables as withinvars all the time (values: between (default) - all vars not explicitly mentioned are treated as between; within - all vars not explicitly mentioned are treated as within)

debug
: outputs additional debugging info if TRUE

Details

For point and line properties (e.g., pointfill) passing NULL allows to avoid setting these values (useful when they are mapped to some variables).

Value

A ggplot object displaying group summaries and their error bars, with optional margin summaries, connecting lines, bars, or individual points.

Examples

```
data(memory_noise)

# Between-subject CI: compare Experiments 1A and 1B
target_higher <- memory_noise[
  memory_noise$relative_noise == "target more noisy",
]
plot_pointrange(
  target_higher,
  aes(x = experiment, y = bias_percent)
) + labs(
  x = "Experiment",
  y = "Bias toward non-target (%)"
)
```

```
# Mixed design: relative noise is within subjects, experiment is between
plot_pointrange(
  memory_noise,
  aes(x = relative_noise, color = experiment, y = bias_percent),
  wid = "participant",
  within_subj = TRUE,
  withinvars = "relative_noise",
  betweenvars = "experiment",
  connecting_line = TRUE
) +
  labs(
    x = "Relative noise",
    y = "Bias toward non-target (%)",
    color = "Experiment"
  ) +
  scale_x_discrete(labels = c(
    "target less noisy" = "Target less noisy",
    "target more noisy" = "Target more noisy"
  ))

# The same mixed design with standard errors
plot_pointrange(
  memory_noise,
  aes(x = relative_noise, color = experiment, y = bias_percent),
  wid = "participant",
  within_subj = TRUE,
  withinvars = "relative_noise",
  betweenvars = "experiment",
  bars = "se"
) +
  labs(
    x = "Relative noise",
    y = "Bias toward non-target (%)",
    color = "Experiment"
  ) +
  scale_x_discrete(labels = c(
    "target less noisy" = "Target less noisy",
    "target more noisy" = "Target more noisy"
  ))
```

---

round_p *Round* p-*value*

Description

If p-value is <= 0.001, returns ".001" else returns p-value rounded to the specified number of digits, optionally including relation sign ("<" or "=").

Usage

```
round_p(
  values,
  include.rel = 1,
  digits = 3,
  strip.lead.zeros = TRUE,
  replace.very.small = 0.001
)
```

Arguments

values	a vector of p-values
include.rel	include relation sign
digits	a number of decimal digits
strip.lead.zeros	remove zero before decimal point
replace.very.small	replace values lower than this criteria (NULL to keep values as is)

Value

A character vector of formatted p-values, including the requested relation signs and small-value threshold notation.

Examples

```
p_values <- c(0.025, 0.0001, 0.001, 0.568)
round_p(p_values)
round_p(p_values, digits = 2)
round_p(p_values, include.rel = FALSE)
round_p(p_values, include.rel = FALSE, strip.lead.zeros = FALSE)
round_p(p_values, include.rel = FALSE, strip.lead.zeros = FALSE, replace.very.small = 0.01)
```

---

scale_y_exp *Exponentiate scale*

Description

Helps to transform log-scale back to normal

Usage

```
scale_y_exp(digits = 0, ...)
```

Arguments

digits	number of digits to use in labels
...	other arguments passed to scale_y_continuous

Value

A continuous y-position scale (class ScaleContinuousPosition) with breaks and labels transformed from logarithmic to exponentiated values.

---

sd_nn *SD with na.rm=TRUE*

Description

SD with na.rm=TRUE

Usage

```
sd_nn(x, ...)
```

Arguments

x	a vector of numbers
...	other arguments passed to sd

Value

A numeric scalar giving the standard deviation of x, with missing values removed.

Examples

```
x <- c(NA, 10, 90)
sd(x)
sd_nn(x)
```

sd_round *Rounded SD*

Description

SD rounded to the specified number of digits

Usage

```
sd_round(x, ...)
```

Arguments

x	a number
...	other arguments passed to f_round

Value

A character string containing the standard deviation rounded to the specified number of decimal places.

Examples

```
sd_round(c(10, 99))
sd_round(c(10, 99, NA))
sd_round(c(10, 99), 2)
```

seq_mean *Get the means between the points of a sequence*

Description

Get the means between the points of a sequence

Usage

```
seq_mean(x)
```

Arguments

x	a vector of numeric values

Value

A numeric vector of midpoints between consecutive values in x.

Examples

```
seq_mean(c(1, 3, 5, 7, 10))
```

---

sum_nn *Sum with na.rm=TRUE*

Description

Sum with na.rm=TRUE

Usage

```
sum_nn(x, ...)
```

Arguments

x	a vector of numbers
...	other arguments passed to sum

Value

A numeric or complex scalar giving the sum of x, with missing values removed.

Examples

```
x <- c(NA, 10, 90)
sum(x)
sum_nn(x)
```

---

table_mean_conf *Get formatted means and confidence intervals*

Description

Get formatted means and confidence intervals

Usage

```
table_mean_conf(x, digits = 2, binom = FALSE, ...)
```

Arguments

x	numeric vector to compute the mean for
digits	number of digits in results (default: 2)
binom	compute binomial CI instead of the usual ones
...	other parameters passed to format_results

Value

A character vector of length two: the formatted mean followed by the confidence limits in square brackets.

Examples

```
# For continuous data
x <- rnorm(100, mean = 50, sd = 10)
table_mean_conf(x)
table_mean_conf(x, digits = 3)

# For binary data
y <- rbinom(100, 1, prob = 0.7)
table_mean_conf(y, binom = TRUE)
```

---

table_mean_conf_by *Get formatted means and confidence intervals by group*

Description

Get formatted means and confidence intervals by group

Usage

```
table_mean_conf_by(x, by, digits = 2, binom = FALSE)
```

Arguments

x	numeric vector to compute the mean for
by	grouping variable
digits	number of digits in results (default: 2)
binom	compute binomial CI instead of the usual ones

Value

An array of list entries, with dimensions and dimension names determined by the levels of the grouping variables in by. Each observed group contains the two-element character vector returned by table_mean_conf; unobserved combinations contain NULL.

Examples

```
# Group means for continuous data
x <- rnorm(100, mean = 50, sd = 10)
groups <- rep(c("A", "B", "C"), length.out = 100)
table_mean_conf_by(x, groups)

# Group means for binary data
y <- rbinom(100, 1, prob = 0.7)
table_mean_conf_by(y, groups, binom = TRUE)
```

Index

∗ **datasets**
memory_noise, 40

adv_cut, 3
afex, *6, 7*
afex::aov_4, *5, 6*
afex::aov_car, *5, 6*
afex::aov_ez, *5, 6*
aggr2, 4
apa, 5
apa.afex_aov, *5*, 6
apa.anova, *5*, 7
apa.anova.merMod, 8
apa.aov, *5*, 9
apa.aovlist *(*apa.aov*)*, 9
apa.BFBayesFactor, *5*, 10
apa.brmsfit, *5*, 11
apa.chisq.test, *5*, 12
apa.dip.test, *5*, 13
apa.emmeans *(*apa.summary_emm*)*, 23
apa.ezANOVA, 14, *19, 20*
apa.glm, *5*, 15, *36*
apa.glmerMod *(*apa.glm*)*, 15
apa.htest, 18
apa.linearHypothesis, 19
apa.list, 19
apa.lm *(*apa.glm*)*, 15
apa.lmerModLmerTest *(*apa.glm*)*, 15
apa.lmert, *5*
apa.lmert *(*apa.summary.merMod*)*, 22
apa.r, *5*, 20
apa.roc.diff *(*apa.roc.test*)*, 21
apa.roc.test, *5*, 21
apa.summary.merMod, 22
apa.summary_emm, 23
apa.ttest, *5*, 24
apa_binom_mean_conf, *6*, 24
apa_ezStats, 25
apa_format_mean_conf, 26, *35*
apa_mean_and_t, *6*, 27
apa_mean_conf, *6*, 28
apa_mean_sd, *6*, *25*, 29
apastats2, *35*

base::cut, *4*
base::mean, *5*
base_breaks, 30
base_breaks_x *(*base_breaks*)*, 30
base_breaks_y *(*base_breaks*)*, 30
BayesFactor, *10, 11*
BayesFactor::anovaBF, *5*
binom_ci, 31
brms, *11*
brms::brm, *5*, *12*

car::Anova, *5*, *8*
car::linearHypothesis, *19*

diptest::dip.test, *5*, *13*
drop_empty_cols, 31

emmeans, *23*
emmeans::contrast, *23*
ez::ezANOVA, *14*
ez::ezStats, *25*
ezANOVA, *19*

f_round, 32, *39*, *48*
format_results, *7–14*, *17–20*, *22–26*, *28, 29*, 33, *49*

get_adjusted_ci, 33
get_adjusted_ci(), *33*
get_superb_ci *(*get_adjusted_ci*)*, 33

ins_lmer, 36

lengthu, 37
lme4::glmer, *5*, *17*
lme4::lmer, *5*, *17*, *36*
lmer_fixef, 37

lmerTest::lmer, *5*, *17*, *22*
load_libs, 38

mean_nn, 38
mean_round, 39
memory_noise, 40

omit_zeroes, 41

paste_and, 41
plot_pointrange, 42
pROC::roc.test, *5*, *21*

round_p, 45

scale_y_exp, 46
sd_nn, 47
sd_round, 48
seq_mean, 48
stats::anova, *8*
stats::aov, *5*, *9*, *10*
stats::chisq.test, *5*, *13*
stats::cor.test, *5*, *20*
stats::glm, *5*, *17*
stats::lm, *5*, *17*
stats::t.test, *5*, *18*, *24*
sum_nn, 49

table_mean_conf, *6*, 49, *50*
table_mean_conf_by, *6*, 50