Package: effectcheck (via r-universe)

October 8, 2026

Type Package

Title Statistical Consistency Checker for Published Research Results

Version 0.7.17

Description A conservative, assumption-aware statistical consistency checker for already-extracted research-results text. Parses test statistics, effect sizes, and confidence intervals across multiple citation styles including American Psychological Association (APA), Harvard, Frontiers, PLOS ONE, Scientific Reports, Nature Human Behaviour, PeerJ, eLife, PNAS, and others. Recomputes effect sizes using all plausible variants when design is ambiguous, and validates internal consistency. Supports t-tests, F-tests/ANOVA, correlations, chi-square, z-tests, regression, and nonparametric tests. Explicitly tracks all assumptions and uncertainty in output. Detects decision errors (significance reversals) similar to 'statcheck'. From v0.4.0 file extraction is no longer part of the package — pair with an external extractor (e.g., 'docpluck' at <https://docpluck.app>) and pass the resulting text to check_text(). Note: this package is under active development and results should be independently verified. Use is at the sole responsibility of the user. Contributions and verification reports are welcome.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/giladfeldman/escicheck

BugReports https://github.com/giladfeldman/escicheck/issues

Imports stringr, stringi, dplyr, purrr, tibble, glue, logger, graphics, stats, utils

Suggests shiny, shinythemes, DT, knitr, rmarkdown, testthat (>= 3.0.0), MBESS, effectsize, jsonlite, statcheck, xml2, rvest

Depends R (>= 4.1.0)

Config/testthat/edition 3

VignetteBuilder knitr

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-10-08 12:30:33 UTC

RemoteUrl https://github.com/giladfeldman/escicheck

RemoteRef HEAD

RemoteSha a12feafb5becbae29ce6d6360eb221becba51efa

Contents

.effectcheck_version . . . 3
[.effectcheck . . . 3
check_text . . . 4
compare_to_variants . . . 8
compare_with_statcheck . . . 8
count_by . . . 9
ec_identify . . . 9
effectcheck-defunct-api . . . 10
export_csv . . . 11
export_json . . . 12
filter_by_delta . . . 12
filter_by_source . . . 13
filter_by_test_type . . . 14
filter_by_uncertainty . . . 14
format_variants . . . 15
generate_report . . . 15
get_alternatives . . . 16
get_decision_errors . . . 17
get_effect_family . . . 18
get_errors . . . 18
get_same_type_variants . . . 19
get_variant_metadata . . . 19
get_variants . . . 20
get_warnings . . . 20
is.effectcheck . . . 21
parse_text . . . 21
plot.effectcheck . . . 22
print.effectcheck . . . 22
print.effectcheck_comparison . . . 23
print.summary.effectcheck . . . 24
rbind.effectcheck . . . 24

render_report 25
summary.effectcheck 25

Index **27**

`.effectcheck_version` *EffectCheck S3 Class Definition and Methods*

Description

This file defines the effectcheck S3 class and its associated methods for printing, summarizing, and plotting results.

Usage

```
.effectcheck_version()
```

`[.effectcheck` *Subset method for effectcheck objects*

Description

Preserves effectcheck class when subsetting.

Usage

```
## S3 method for class 'effectcheck'
x[...]
```

Arguments

`x`	An effectcheck object
`...`	Subsetting arguments

Value

An effectcheck object

Examples

```
res <- check_text("t(28) = 2.21, p = .035. F(1, 50) = 4.03, p = .049")
res[1, ]
```

check_text *Check raw text for statistical consistency*

Description

Parses APA-style statistical results from text and checks for consistency between reported and computed values. Uses type-matched comparison to ensure reported effect sizes are compared against the same type of computed values.

Usage

```
check_text(
  text,
  stats = c("t", "F", "r", "chisq", "z", "U", "W", "H", "regression", "spearman",
   "kendall", "kendall_w", "dscf", "cochran_q", "RR", "rdpct", "md_hl", "binomial",
   "interaction_p", "mediation_indirect", "mcnemar_or", "bayes_factor", "hazard_ratio",
    "d_reported_only", "wts", "ats", "brunner_munzel", "yuen", "mean_diff_ci"),
  ci_level = 0.95,
  alpha = 0.05,
  one_tailed = FALSE,
  paired_r_grid = c(seq(0.1, 0.9, by = 0.1), 0.95),
  assume_equal_ns_when_missing = TRUE,
  ci_method_phi = "bonett_price",
  ci_method_V = "bonett_price",
  tol_effect = list(d = 0.02, r = 0.005, phi = 0.02, V = 0.02),
  tol_ci = 0.02,
  tol_p = 0.001,
  messages = FALSE,
  max_text_length = 10^7,
  max_stats_per_text = 10000,
  cross_type_action = "NOTE",
  ci_affects_status = TRUE,
  plausibility_filter = TRUE,
  sign_sensitive = FALSE,
  method_context_action = "NOTE",
  design_ambiguous_action = "WARN",
  unknown_groups_action = "WARN",
  min_confidence = 0L,
  table_rows = NULL,
  extraction_provenance = NULL
)
```

Arguments

text	Character vector of text to check
stats	Character vector of test types to check (default: all supported types)

`ci_level` Default confidence interval level (default 0.95)

`alpha` Significance threshold for decision error detection (default 0.05)

`one_tailed` Logical, assume one-tailed tests (default FALSE)

`paired_r_grid` Numeric vector of correlation values for paired t-test grid search

`assume_equal_ns_when_missing`
Logical, assume equal group sizes when missing (default TRUE)

`ci_method_phi` CI method for phi coefficient (default "bonett_price") – accepted and recorded in the result's settings, but currently has NO effect on any computed value or status

`ci_method_V` CI method for Cramer's V (default "bonett_price") – accepted and recorded in the result's settings, but currently has NO effect on any computed value or status

`tol_effect` List of tolerances for effect sizes by type

`tol_ci` Tolerance for CI bounds (default 0.02)

`tol_p` Tolerance for p-values (default 0.001) – accepted and recorded in the result's settings, but currently has NO effect on any computed value or status

`messages` Logical, show progress messages (default FALSE)

`max_text_length`
Maximum total text length in characters (default 10^7)

`max_stats_per_text`
Maximum number of stats to process per text (default 10000)

`cross_type_action`
Action when cross-type match found ("NOTE", "WARN", or "ERROR"; default "NOTE")

`ci_affects_status`
Whether CI mismatches affect status (default TRUE)

`plausibility_filter`
Whether to apply plausibility bounds filter (default TRUE)

`sign_sensitive` Intended to make sign differences affect status (default FALSE) – accepted and recorded in the result's settings, but currently has NO effect on any computed value or status

`method_context_action`
Action when method context detected in chunk ("NOTE", "WARN", or "SKIP"; default "NOTE")

`design_ambiguous_action`
Action when a design-ambiguous t-test (or F(1,df), or z with d/g) effect-size ERROR occurs ("WARN", "NOTE", or "ERROR"; default "WARN").

It has one exception, and it is deliberate. The downgrade is applied only where design ambiguity is a candidate EXPLANATION for the discrepancy – that is, where the reported effect lies within the range of the computed independent and paired variants plus a 50\ reported effect matching NEITHER design is not explained by not knowing which design was used, so it keeps its ERROR and this parameter is inert on that row. Same principle as the v0.6.18 omnibus-df rule: an effect matching neither candidate keeps its flag. The row states the reason ("Extreme discrepancy ... likely reflects data extraction error").

Documented in v0.7.9 after a downstream consumer probed the parameter and found 2 rows in 148,984 where a caller asking for "WARN" got "ERROR". The behaviour was correct and the SILENCE was the defect: a policy knob whose exception is not documented reads, from outside, exactly like a policy knob that does not work. Pinned by `test-v079-design-ambiguous-action-scope.R`, both directions.

`unknown_groups_action`
: Action when d/g ERROR occurs with unknown group sizes n1/n2 ("WARN", "NOTE", or "ERROR"; default "WARN")

`min_confidence`
: Minimum confidence score (0-10) for results to be included in output (default 0)

`table_rows`
: Optional list of docpluck structured table rows (`?structured=true` `flattened_rows[]`, docpluck v2.4.95+). Each element is a list with `label`, `row_label`, `row_idx`, and a `fields` list of typed statistics. Rows whose `fields` carry a recognised statistic are mapped via `flattened_rows_to_parsed()` and run through the same verification pipeline as text-parsed rows, tagged `result_context = "table"` and deduplicated against any identical prose row. Default NULL (text-only; output is byte-identical to prior versions).

`extraction_provenance`
: Optional list: the extractor's own normalization report, passed through verbatim (docpluck's `normalization` block). Read, never reinterpreted. It records whether the extractor REWROTE VALUES in this document rather than merely canonicalising notation – docpluck's sign-recovery rules can turn a printed `SE = 0.199` into `-0.199`, and before v0.7.6 that rewrite was invisible downstream. Surfaced as the document-level columns `upstream_sign_rewrites` (integer; NA when no report was supplied, 0 when one was supplied and reported none) and `upstream_normalization_version`. **A zero does not mean no rewrite happened** – corrected 2026-08-22, and stronger than the "lower bound" this said before. The earlier stated reason (docpluck assigned rather than accumulated the metric, so only the last of three rules survived) was fixed upstream in normalization 1.9.59. The remaining cause is different and worse: docpluck derives the metric from a *character-length delta* and skips it entirely when that delta is zero, so a length-neutral rewrite – a 2-for-minus or U+2212 glyph substitution, which is precisely the sign-corruption class this column exists to expose – increments nothing and the key can be absent. An absent key is read here as 0 (docpluck omits NA fields rather than sending them), so such a document publishes `upstream_sign_rewrites = 0` while values were in fact rewritten. Non-zero still reliably means values were rewritten, and the magnitude is not an occurrence count in any case. Deliberately does NOT set `extraction_suspect`: the report is document-level, that flag is per-row and gates effect-size rewriting and two ERROR-path downgrades. Default NULL.

Value

An effectcheck S3 object whose `results` tibble carries the parsed and recomputed statistics. Notable output columns:

`design_ambiguous`
: Logical. TRUE when the row's matching is design-uncertain. INTENTIONALLY BROAD – see `ambiguity_reason` for the specific category (one of two: *structural-*

design for a t / F(1,df) / z that reports d or g and produced BOTH paired and independent variant families; *cross-family* for a reported ES type that has no same-type variants in the computed-variants set, e.g. a Cohen's d reported on an F(2,df) omnibus). Equal to `ambiguity_level != "clear"`; the flag never under-reports the row's uncertainty.

`ambiguity_level` Character. `"clear"`, `"ambiguous"`, or `"highly_ambiguous"`. The category-A cases tend to land on `"ambiguous"`; the category-B cases land on `"highly_ambiguous"`.

`ambiguity_reason` Character. Human-readable explanation of the ambiguity. Since v0.5.11, a stable bracket-tagged category suffix is appended when applicable: `"[category: structural-design]"` or `"[category: cross-family]"`; since v0.7.11 also `"[category: not-computed]"` when an effect size was reported but no variant could be computed at all (e.g. an F without df), so nothing was matched. Consumers can grep for the tag to programmatically split the semantics without parsing English.

`matched_variant` The computed variant matched against the reported ES. A cross-family fallback row will name a variant from a different family than the reported ES type (e.g. `matched_variant="eta"` when `effect_reported_name="d"`).

`sign_ci_violation` Logical (since v0.6.3, R-0007). TRUE when a sign-bearing reported estimate (d, g, dz, dav, drm, r, beta, partial_r) lies OUTSIDE its reported CI but its sign-flip lies inside – the signature of a dropped-minus extraction error (e.g. `r = .74` reported with `95% CI [-0.92, -0.30]`). FLAG ONLY: the parsed value is never mutated; the violation is also surfaced in `uncertainty_reasons`. NA on error/short-circuit rows, FALSE otherwise.

`estimate_outside_ci` Logical (since v0.7.9). TRUE when the same estimate-in-CI invariant is violated with NO sign explanation: the reported estimate lies outside its reported CI *and so does its sign-flip*, so a dropped minus cannot account for it and at least one of the three published numbers is wrong. Mutually exclusive with `sign_ci_violation`, which keeps its own, more specific diagnosis. Added because the invariant was evaluated but only ever REPORTED the dropped-minus shape, so a row whose estimate lay outside its own interval was indistinguishable from one where it lay inside – while the row's message said the invariant had been checked. Real instance: Chan & Feldman (2025), *Cognition and Emotion* 39(6), p. 1238, Table 9 row 2a prints `r = .70, 95% CI [0.73, 0.76]`. FLAG ONLY: the parsed value is never mutated. NA on error/short-circuit rows, FALSE otherwise.

Plus all other columns: `location`, `raw_text`, test identification (`test_type`, `chisq_subtype`, `df1`, `df2`, `stat_value`, `N`), p-values (`p_reported`, `p_computed`, `decision_error`), effect-size family columns (`d_ind`, `dz`, `g_ind`, `eta2`, `partial_eta2`, `omega2`, ...), CI metadata (`ci_reported`, `ci_expected`, `ci_width_ratio`, `ci_level_source`, ...), and status (`status`, `check_type`, `check_scope`, `extraction_suspect`, `design_inferred`, `uncertainty_level`, `uncertainty_reasons`, ...).

Examples

```
result <- check_text("t(28) = 2.21, p = .035, d = 0.80")
print(result)
summary(result)
```

compare_to_variants *Compare reported value to all variants*

Description

Creates a comparison table showing the reported value against all computed variants.

Usage

```
compare_to_variants(x, row_index = 1)
```

Arguments

x	An effectcheck object
row_index	The row index

Value

A data frame with variant comparisons

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
compare_to_variants(res, 1)
```

compare_with_statcheck *Compare effectcheck results with statcheck*

Description

Runs both effectcheck and statcheck on the same text and returns a merged comparison tibble.

Usage

```
compare_with_statcheck(text, ...)
```

Arguments

text	Character string containing APA-formatted statistics
...	Additional arguments passed to check_text()

Value

A tibble with source column ("both", "effectcheck_only", "statcheck_only")

Examples

```
comp <- compare_with_statcheck("t(28) = 2.21, p = .035, d = 0.80")
print(comp)
```

---

count_by *Count statistics by category*

Description

Provides counts of statistics grouped by various categories.

Usage

```
count_by(x, by = c("status", "test_type", "uncertainty", "design", "source"))
```

Arguments

x	An effectcheck object
by	Character, grouping variable: "status", "test_type", "uncertainty", "design", or "source"

Value

A data frame with counts

Examples

```
results <- check_text("t(28) = 2.21, p = .035. F(1, 50) = 4.03, p = .049")
count_by(results, "status")
count_by(results, "test_type")
```

---

ec_identify *Identify and Filter EffectCheck Results*

Description

Functions for filtering and identifying problematic results in effectcheck output. Identify problematic results

Usage

```
ec_identify(
  x,
 what = c("errors", "warnings", "decision_errors", "high_uncertainty", "insufficient",
    "all_problems"),
  ...
)
```

Arguments

x	An effectcheck object
what	Character vector specifying what to identify: • "errors": Results with ERROR status • "warnings": Results with WARN status • "decision_errors": Results with significance reversal • "high_uncertainty": Results with high uncertainty level • "insufficient": Results with insufficient data • "all_problems": All of the above
...	Additional arguments (ignored)

Details

Filters effectcheck results to show only problematic cases based on specified criteria.

Value

An effectcheck object containing only the identified results

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
errors <- ec_identify(results, "errors")
```

---

effectcheck-defunct-api

EffectCheck API Functions (DEFUNCT in v0.4.0)

Description

All file-input functions in this file became `.Defunct()` in effectcheck 0.4.0. ESCImate delegates document extraction to docpluck; pass the resulting text to `check_text()` for analysis.

Details

Migration:

```
## Before (errors in 0.4.0):
results <- effectcheck::checkPDFdir("path/to/pdfs/")

## After:
library(httr2)
pdfs <- list.files("path/to/pdfs/", pattern = "\\.pdf$", full.names = TRUE)
results <- purrr::map_dfr(pdfs, function(p) {
  resp <- request("https://docpluck.app/api/extract") |>
```

```
    req_headers(Authorization = paste("Bearer", Sys.getenv("DOCPLUCK_API_KEY"))) |>
      req_url_query(normalize = "academic", quality = "true") |>
      req_body_multipart(file = curl::form_file(p)) |>
      req_perform()
    dplyr::mutate(check_text(resp_body_json(resp)$text), source = basename(p))
  })
```

`export_csv` *Export results to CSV*

Description

Exports check results to CSV format with proper handling of special characters and NA values.

Usage

```
export_csv(res, out, na = "", row.names = FALSE)
```

Arguments

`res`	tibble returned by check_text() / check_files()
`out`	output file path (csv)
`na`	string to use for NA values (default: "")
`row.names`	logical, include row names (default: FALSE)

Value

Invisible path to the generated CSV file.

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
export_csv(res, out = tempfile(fileext = ".csv"))
```

export_json *Export results to JSON*

Description

Exports check results to JSON format with structured metadata.

Usage

```
export_json(res, out, pretty = TRUE)
```

Arguments

res	tibble returned by check_text() / check_files()
out	output file path (json)
pretty	logical, pretty-print JSON (default: TRUE)

Value

Invisible path to the generated JSON file.

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
export_json(res, out = tempfile(fileext = ".json"))
```

filter_by_delta *Filter results by effect size delta*

Description

Filters effectcheck results by the magnitude of effect size discrepancy.

Usage

```
filter_by_delta(x, min_delta = 0, max_delta = Inf)
```

Arguments

x	An effectcheck object
min_delta	Minimum absolute delta to include (default 0)
max_delta	Maximum absolute delta to include (default Inf)

Value

An effectcheck object containing only results within the delta range

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
filter_by_delta(results, min_delta = 0.1)
```

`filter_by_source` *Filter results by source file*

Description

Filters effectcheck results to show only results from specific files.

Usage

```
filter_by_source(x, files, pattern = FALSE)
```

Arguments

`x`	An effectcheck object
`files`	Character vector of file names or patterns to include
`pattern`	Logical, if TRUE treat files as regex patterns (default FALSE)

Value

An effectcheck object containing only results from specified files

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
filter_by_source(results, "text_input")
```

filter_by_test_type *Filter results by test type*

Description

Filters effectcheck results to show only specific test types.

Usage

```
filter_by_test_type(x, types)
```

Arguments

x	An effectcheck object
types	Character vector of test types to include (e.g., "t", "F", "r", "chisq", "z")

Value

An effectcheck object containing only the specified test types

Examples

```
results <- check_text("t(28) = 2.21, p = .035. F(1, 50) = 4.03, p = .049")
filter_by_test_type(results, "t")
```

filter_by_uncertainty *Filter results by uncertainty level*

Description

Filters effectcheck results by uncertainty level.

Usage

```
filter_by_uncertainty(x, levels)
```

Arguments

x	An effectcheck object
levels	Character vector of uncertainty levels to include ("low", "medium", "high")

Value

An effectcheck object containing only the specified uncertainty levels

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
filter_by_uncertainty(results, "high")
```

`format_variants` *Format variants for display*

Description

Creates a formatted string representation of variants for a row.

Usage

```
format_variants(x, row_index = 1, include_alternatives = TRUE)
```

Arguments

`x`	An effectcheck object
`row_index`	The row index
`include_alternatives`	Whether to include alternative suggestions

Value

A character string with formatted variant information

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
cat(format_variants(res, 1))
```

`generate_report` *Generate a submission-ready EffectCheck report*

Description

Creates a self-contained HTML report with executive summary, color-coded results table, expandable details, reproducible R code, and footer stamp.

Usage

```
generate_report(
  res,
  out,
  format = "html",
  title = "EffectCheck Report",
  author = NULL,
  source_name = NULL,
  include_repro_code = TRUE,
  style = "beginner"
)
```

Arguments

res	tibble returned by check_text() / check_files()
out	output file path (html)
format	Output format: "html" (default) or "pdf" (requires rmarkdown)
title	Report title (default: "EffectCheck Report")
author	Author name (optional)
source_name	Source file name (optional)
include_repro_code	Logical, include reproducible R code section (default TRUE)
style	Report style: "beginner" for plain English narrative (default), "expert" for the traditional technical table format

Value

Invisible path to the generated report file

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
generate_report(res, out = tempfile(fileext = ".html"))
```

---

get_alternatives *Get alternative suggestions for a row*

Description

Get alternative suggestions for a row

Usage

```
get_alternatives(x, row_index = 1)
```

Arguments

x	An effectcheck object
row_index	The row index

Value

A list of alternative effect size suggestions

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
get_alternatives(res, 1)
```

---

get_decision_errors *Get decision errors from effectcheck results*

Description

Extracts results where the significance decision would be reversed (i.e., reported as significant when computed is not, or vice versa).

Usage

```
get_decision_errors(x)
```

Arguments

x	An effectcheck object

Value

An effectcheck object containing only decision errors

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
get_decision_errors(results)
```

get_effect_family *Get effect size family information*

Description

Returns information about an effect size family and its variants.

Usage

```
get_effect_family(effect_type)
```

Arguments

effect_type	The effect size type (e.g., "d", "eta2", "r")

Value

A list with family, variants, alternatives, and description

Examples

```
get_effect_family("d")
```

get_errors *Get errors from effectcheck results*

Description

Convenience function to extract only ERROR status results.

Usage

```
get_errors(x)
```

Arguments

x	An effectcheck object

Value

An effectcheck object containing only errors

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
get_errors(results)
```

get_same_type_variants *Get same-type variants for a row*

Description

Get same-type variants for a row

Usage

```
get_same_type_variants(x, row_index = 1)
```

Arguments

x	An effectcheck object
row_index	The row index

Value

A list of same-type variants with their values and metadata

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
get_same_type_variants(res, 1)
```

get_variant_metadata *Get variant metadata*

Description

Returns metadata for a specific effect size variant type.

Usage

```
get_variant_metadata(variant_name)
```

Arguments

variant_name	The name of the variant (e.g., "d_ind", "dz", "eta2")

Value

A list with name, assumptions, when_to_use, and formula

Examples

```
get_variant_metadata("d_ind")
```

get_variants *Get all variants for a specific row*

Description

Extracts and parses the all_variants JSON structure for a given row.

Usage

```
get_variants(x, row_index = 1)
```

Arguments

x	An effectcheck object
row_index	The row index to extract variants from

Value

A list with same_type and alternatives sublists

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
get_variants(res, 1)
```

get_warnings *Get warnings from effectcheck results*

Description

Convenience function to extract only WARN status results.

Usage

```
get_warnings(x)
```

Arguments

x	An effectcheck object

Value

An effectcheck object containing only warnings

Examples

```
results <- check_text("t(28) = 2.21, p = .035, d = 0.80")
get_warnings(results)
```

is.effectcheck *Test if object is an effectcheck object*

Description

Test if object is an effectcheck object

Usage

```
is.effectcheck(x)
```

Arguments

x	Object to test

Value

Logical

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
is.effectcheck(res)
```

parse_text *Parse APA-style stats and effects from text*

Description

Extracts test statistics, effect sizes, confidence intervals, and sample sizes from APA-style text. Includes context window extraction for design inference.

Usage

```
parse_text(text, context_window_size = 2)
```

Arguments

text	Character vector of text to parse
context_window_size	Number of sentences before/after to capture (default 2)

Value

Tibble with parsed elements including context windows

Examples

```
parsed <- parse_text("t(28) = 2.21, p = .035, d = 0.80")
parsed$test_type
parsed$stat_value
```

---

plot.effectcheck *Plot method for effectcheck objects*

Description

Creates visualizations of effectcheck results.

Usage

```
## S3 method for class 'effectcheck'
plot(x, type = c("status", "uncertainty", "test_type", "delta", "all"), ...)
```

Arguments

x	An effectcheck object
type	Type of plot: "status", "uncertainty", "test_type", "delta", or "all"
...	Additional arguments passed to plotting functions

Value

Invisibly returns x.

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
plot(res, type = "status")
```

---

print.effectcheck *Print method for effectcheck objects*

Description

Displays a formatted summary of effectcheck results.

Usage

```
## S3 method for class 'effectcheck'
print(x, short = TRUE, n = 10, ...)
```

Arguments

x	An effectcheck object
short	Logical, if TRUE show abbreviated output (default TRUE)
n	Maximum number of rows to display (default 10)
...	Additional arguments (ignored)

Value

Invisibly returns x.

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
print(res)
```

---

print.effectcheck_comparison

Print method for effectcheck comparison

Description

Print method for effectcheck comparison

Usage

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

Arguments

x	An effectcheck_comparison object
...	Additional arguments (ignored)

Value

Invisibly returns x.

Examples

```
comp <- compare_with_statcheck("t(28) = 2.21, p = .035, d = 0.80")
print(comp)
```

print.summary.effectcheck *Print method for summary.effectcheck objects*

Description

Print method for summary.effectcheck objects

Usage

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

Arguments

x	A summary.effectcheck object
...	Additional arguments (ignored)

Value

Invisibly returns x.

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
s <- summary(res)
print(s)
```

rbind.effectcheck *Combine effectcheck objects*

Description

Combine effectcheck objects

Usage

```
## S3 method for class 'effectcheck'
rbind(...)
```

Arguments

...	effectcheck objects to combine

Value

Combined effectcheck object

Examples

```
res1 <- check_text("t(28) = 2.21, p = .035")
res2 <- check_text("F(1, 50) = 4.03, p = .049")
combined <- rbind(res1, res2)
```

---

render_report *Render an enhanced HTML report*

Description

Creates an HTML report with summary statistics, expandable sections, and uncertainty visualization.

Usage

```
render_report(res, out)
```

Arguments

res	tibble returned by check_text() / check_files()
out	output file path (html)

Value

Invisible path to the generated HTML report file.

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
render_report(res, out = tempfile(fileext = ".html"))
```

---

summary.effectcheck *Summary method for effectcheck objects*

Description

Provides comprehensive summary statistics for effectcheck results.

Usage

```
## S3 method for class 'effectcheck'
summary(object, ...)
```

Arguments

`object`	An effectcheck object
`...`	Additional arguments (ignored)

Value

A list of class "summary.effectcheck" containing summary statistics

Examples

```
res <- check_text("t(28) = 2.21, p = .035, d = 0.80")
summary(res)
```

Index

.effectcheck_version, 3
[.effectcheck, 3

check_text, 4
check_text(), *10*
compare_to_variants, 8
compare_with_statcheck, 8
count_by, 9

ec_identify, 9
effectcheck-defunct-api, 10
export_csv, 11
export_json, 12

filter_by_delta, 12
filter_by_source, 13
filter_by_test_type, 14
filter_by_uncertainty, 14
format_variants, 15

generate_report, 15
get_alternatives, 16
get_decision_errors, 17
get_effect_family, 18
get_errors, 18
get_same_type_variants, 19
get_variant_metadata, 19
get_variants, 20
get_warnings, 20

is.effectcheck, 21

parse_text, 21
plot.effectcheck, 22
print.effectcheck, 22
print.effectcheck_comparison, 23
print.summary.effectcheck, 24

rbind.effectcheck, 24
render_report, 25

summary.effectcheck, 25