

Package: R4VN (via r-universe)

September 30, 2026

Type Package

Title Health Data Analysis and Publication-Ready Reporting

Version 1.6

Description Provides short and consistent commands for data management, descriptive and inferential statistics, epidemiological analyses, regression models, survival and longitudinal analyses, diagnostic accuracy, scale assessment, meta-analysis, machine learning, study design, publication-ready tables, graphics, and reporting. Commands accept an explicit data frame or an active data frame selected with `usedf()`.

License MIT + file LICENSE

Encoding UTF-8

Language en

Depends R (>= 4.1.0)

Imports grDevices, graphics, splines, stats, tools, utils, survey

Suggests arrow, Boruta, bslib (>= 0.7.0), curl, DBI, dbscan, DT, e1071, flextable, forecast, foreign, geopack, ggplot2, glmnet, haven, htmltools, jsonlite, keyring, knitr, lavaan, leaflet, nortest, lme4, MASS, metafor, mgcv, nlme, nnet, officer, openxlsx, pagedown, pkgdown, pmsampsize, pROC, presize, psych, quantreg, ranger, readstata13, readxl, rmarkdown, RMariaDB, ROSE, rpart, rstudioapi, sampling, sandwich, scales, sf, shiny (>= 1.8.0), spdep, statpsych, survival, testthat (>= 3.0.0), TrialSize, tseries, WebPower, webshot2, writexl, xgboost

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.0.0

Author Thai Thanh Truc [aut, cre]

Maintainer Thai Thanh Truc <thaitanhtruc@gmail.com>

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

Date/Publication 2026-09-30 13:55:02 UTC

RemoteUrl <https://github.com/cran/R4VN>

RemoteRef HEAD

RemoteSha f19be3fb5507a4b05b9ec2e997c7bce1f101fd35

Contents

aiask	4
aisetup	6
anovai	8
appenddata	10
as.data.frame.r4vn_machine	11
as.data.frame.r4vn_meta	12
at	12
cii	13
corr	14
cox	16
dengue_its_control	17
dengue_ts	18
describe	18
design	20
dict	21
distdata	22
distlearn	24
dropvar	25
egenvar	26
epi	27
epitool	30
esizei	31
friedman	32
gbar	34
gbox	37
gdensity	39
genvar	41
gforest	42
ghist	45
gline	47
gpie	49
groc	51
gscatter	53
gsurv	55
ir	58
keepvar	59
kwallis	60
labvar	62
lincom	64
logistic	66

lrtest	69
margins	70
marginsplot	71
mcc	73
mergedata	75
nlregress	76
normtest	78
nptrend	79
opendata	80
ordervar	85
plot.r4vn_diag	86
plot.r4vn_distdata	88
plot.r4vn_graph	89
plot.r4vn_machine	89
plot.r4vn_meta	90
plot.r4vn_tabforest	94
plot.r4vn_tablong	95
plot.r4vn_tabscale	95
plot.r4vn_tabscore	96
plot.r4vn_tabts	97
poisson	98
predict	100
predict.r4vn_machine	103
predict.r4vn_tabscore	104
print.r4vn_graph	104
print.r4vn_graph_set	105
print.r4vn_machine	105
print.r4vn_meta	106
print.r4vn_quick	106
print.r4vn_stat	107
print.r4vn_tabscale	107
propi	108
qregress	111
r4fun	113
R4VN_examples	114
ranksum	115
regress	117
renvar	119
replacevar	120
savedata	121
sdtesti	123
shapevar	124
signrank	126
suml	127
summary.r4vn_survey	129
summary.r4vn_tabsurvey	130
surv_tables	130
survexport	131

surveyset	132
swilk	134
tab	135
tab1	142
tabdiag	144
tabdiagi	155
tabexport	157
tabforest	160
tabforest_demo	169
tabi	170
tablearn	172
tablong	184
tabmachine	195
tabmeta	211
tabmulti	220
tabscale	225
tabscore	231
tabsurv	243
tabsurvey	248
tabts	261
ttesti	272
usedf	274
varform	275
vars	277
ztesti	280
Index	282

aiask	<i>Ask an AI service to interpret summarized statistical results</i>
-------	--

Description

Automatically discovers compact statistical result components inside an object, sends only the most useful results to the selected AI service, and attaches the returned interpretation. Lists receive an ai element; other objects receive an "ai" attribute.

Usage

```
aiask(x, ai = TRUE, prompt = NULL)
```

Arguments

x	An R4VN result object, summarized list, result table/matrix, or character result. aiask() automatically finds useful result components.
ai	TRUE to use the default configuration or a configuration name.
prompt	Optional request specific to the current result. It is added after the permanent prompt stored by aisetup().

Details

The extractor is generic rather than command-specific. It recursively searches for result-like tables, matrices, estimates, tests, diagnostics, and short statistical context while ignoring raw data, HTML/CSS, formatting objects, plots, model internals, residuals, and other high-volume noise. The same `aiask()` can therefore be used for R4VN descriptive, multivariable, longitudinal, survival, diagnostic, scale, regression, and future result objects without a separate prepare function for every command.

A conservative input budget is applied before the API call. If the provider reports that the request is too large, `aiask()` automatically compacts the results further and retries up to two times.

Value

Invisibly returns `x` with an attached AI result. Successful results contain `status`, `name`, `model`, `comment`, `input`, optional `usage`, and `created`. API failures are attached with `status = "error"` and do not destroy `x`.

Examples

```
## Not run:
aisetup(
  name = "gpt",
  provider = "openai",
  model = "gpt-5.6",
  tokenenv = "OPENAI_API_KEY",
  default = TRUE
)

result <- list(
  analysis = "Logistic regression",
  outcome = "Hypertension",
  estimates = data.frame(
    variable = c("Age", "Smoking"),
    OR = c(1.05, 1.82),
    lower = c(1.02, 1.15),
    upper = c(1.08, 2.88),
    p = c(0.001, 0.011)
  )
)

result <- aiask(result)
result$ai$comment

result <- aiask(
  result,
  ai = "gpt",
  prompt = "Focus on modifiable risk factors."
)

## End(Not run)
```

aisetup

Configure AI services for R4VN

Description

Adds, updates, lists, selects, or removes named AI configurations. Only the configuration is saved by R4VN. A token supplied through `tokenenv` remains in an environment variable. A token supplied directly is stored in the system keyring when package keyring is available; otherwise it is kept only for the current R session.

Usage

```
aisetup(
  name = NULL,
  provider = "openai",
  model = NULL,
  url = NULL,
  token = NULL,
  tokenenv = NULL,
  api = NULL,
  language = "vi",
  prompt = NULL,
  default = FALSE,
  use = NULL,
  list = FALSE,
  clear = FALSE,
  save = TRUE,
  timeout = 120,
  temperature = NULL,
  maxtokens = 1200,
  inputtokens = 3000,
  overwrite = TRUE
)
```

Arguments

<code>name</code>	Configuration name, for example "gpt" or "local".
<code>provider</code>	"openai" or "compatible". The latter is for services implementing an OpenAI-compatible endpoint.
<code>model</code>	Model identifier required by the selected service.
<code>url</code>	Full API endpoint or API base ending in /v1. When omitted for <code>provider = "openai"</code> , the official OpenAI endpoint is used.
<code>token</code>	Optional API token. It is never written into the R4VN configuration file.
<code>tokenenv</code>	Optional environment-variable name containing the token.
<code>api</code>	"responses" or "chat". It is inferred from <code>provider</code> and <code>url</code> when omitted.

language	Default language for AI comments. Common values are "vi" and "en".
prompt	Optional permanent instruction added to every request using this configuration.
default	Logical; make this configuration the default.
use	Name of an existing configuration to make default.
list	Logical; list current configurations.
clear	FALSE, TRUE, or a configuration name. TRUE removes all configurations.
save	Logical; save the configuration for future R sessions.
timeout	Request timeout in seconds.
temperature	Optional model temperature. Leave NULL for provider defaults and maximum compatibility.
maxtokens	Maximum generated tokens.
inputtokens	Approximate target maximum tokens for statistical results sent to the AI service. <code>aiask()</code> automatically reduces this budget and retries when the provider reports that a request is too large.
overwrite	Logical; permit replacing an existing configuration.

Value

Invisibly returns the saved configuration, the configuration table, or TRUE after a management action.

Examples

```
## Not run:
# Recommended: keep the token in .Renviron
# OPENAI_API_KEY=your-token
aisetup(
  name = "gpt",
  provider = "openai",
  model = "gpt-5.6",
  tokenenv = "OPENAI_API_KEY",
  default = TRUE
)

# Direct token: saved securely when keyring is installed
aisetup(
  name = "gpt",
  provider = "openai",
  model = "gpt-5.6",
  token = "your-token",
  default = TRUE
)

# OpenAI-compatible local or third-party service
aisetup(
  name = "local",
  provider = "compatible",
  model = "local-model",
  url = "http://localhost:11434/v1",
```

```

    default = TRUE
)

aisetup(list = TRUE)
aisetup(use = "gpt")
aisetup(clear = "local")

## End(Not run)

```

anovai

One-way ANOVA with Bartlett Test

Description

anovai() reconstructs one-way ANOVA from c(n, mean, sd) summaries. anova() analyzes a numeric variable by a grouping variable. Bartlett's test of equal variances is included by default. When the first argument is a fitted model, the call is delegated to `stats::anova()`. For one-way ANOVA, posthoc can request Tukey, Games-Howell, Scheffe, Bonferroni, or other multiplicity-adjusted pairwise comparisons. Eta-squared and omega-squared remain reported by the ANOVA engine. Hierarchical by = vars(...) is supported for data-based ANOVA.

Usage

```

anovai(
  ...,
  group.names = NULL,
  bartlett = TRUE,
  level = 0.95,
  posthoc = c("none", "tukey", "games-howell", "scheffe", "bonferroni", "pairwise"),
  adjust = "holm",
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

anovai(
  ..., group.names = NULL, bartlett = TRUE, level = 0.95,
  posthoc = c("none", "tukey", "games-howell", "scheffe", "bonferroni", "pairwise"),
  adjust = "holm", digits = 3, p_digits = 3, show = TRUE, console = FALSE
)

anova(
  object, ..., by = NULL, data = NULL, bartlett = TRUE, level = 0.95,
  posthoc = c("none", "tukey", "games-howell", "scheffe", "bonferroni", "pairwise"),
  adjust = "holm", digits = 3, p_digits = 3, show = TRUE, console = FALSE
)

```

```

anova(
  object,
  ...,
  by = NULL,
  data = NULL,
  bartlett = TRUE,
  level = 0.95,
  posthoc = c("none", "tukey", "games-howell", "scheffe", "bonferroni", "pairwise"),
  adjust = "holm",
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

```

Arguments

...	For <code>anovai()</code> , group summaries <code>c(n, mean, sd)</code> . For a fitted model, additional objects or arguments passed to <code>stats::anova()</code> .
group.names	Optional names for the summarized groups.
bartlett	Include Bartlett's test of equal variances.
level	Confidence level for group means.
posthoc	Post-hoc method: "none", "tukey", "games-howell", "scheffe", "bonferroni", or "pairwise". Games-Howell is useful when equal-variance assumptions are doubtful. <code>posthoc = "bonferroni"</code> provides Bonferroni-adjusted p-values and simultaneous confidence intervals.
adjust	Multiplicity adjustment used by <code>posthoc = "pairwise"</code> ; any method accepted by <code>stats::p.adjust()</code> may be used.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
object	Numeric outcome variable or a fitted model object.
by	Grouping variable for one-way ANOVA. <code>by = vars(region, sex, group)</code> uses region and sex as nested strata and group as the ANOVA factor.
data	Data frame. If NULL, active data is used.

Value

A `r4vn_stat` object for one-way ANOVA, or the ordinary result from `stats::anova()` for fitted models.

Examples

```

anovai(c(n = 20, mean = 10, sd = 2),
       c(n = 25, mean = 15, sd = 4),
       c(n = 18, mean = 13, sd = 3), posthoc = "bonferroni")

d <- data.frame(score = c(10,12,11,18,17,20,25,24,27),
                treatment = factor(rep(c("A","B","C"), each = 3)))
anova(score, by = treatment, data = d, posthoc = "games-howell")

```

appenddata

*Append data frames by observations***Description**

Appends data frames or files below a master data frame. Active data is used as master when available.

Usage

```

appenddata(
  ...,
  data = NULL,
  source = NULL,
  force = FALSE,
  active = TRUE,
  quiet = FALSE,
  labels = c("factor", "labelled", "numeric")
)

```

Arguments

...	Data frames or existing file paths to append.
data	Optional explicit master data frame.
source	NULL, TRUE for "_source", or a source-variable name.
force	Logical; convert incompatible columns to character.
active	Logical; replace active data with the result.
quiet	Logical; suppress messages.
labels	Label handling when a file is opened.

Details

Columns are aligned by name. Missing columns are filled with NA. If no active data and no explicit data are supplied, the first object in ... becomes master.

Value

The appended data frame invisibly.

Examples

```
before <- data.frame(id = 1:2, age = c(20, 30))
after <- data.frame(id = 3:4, age = c(40, 50), sex = c("M", "F"))
usedf(before, quiet = TRUE)
appenddata(after, source = TRUE, quiet = TRUE)
combined <- usedf(quiet = TRUE)
combined2 <- appenddata(before, after, active = FALSE, quiet = TRUE)
```

as.data.frame.r4vn_machine

Convert a tabmachine result to a data frame

Description

Convert a tabmachine result to a data frame

Usage

```
## S3 method for class 'r4vn_machine'
as.data.frame(x, ...)
```

Arguments

x	A r4vn_machine object.
...	Unused.

Value

A data frame containing the final evaluation performance stored in x\$performance. Each row represents a model/metric combination and reports the evaluation data set, method, metric, estimate, confidence limits when available, and the confidence-interval method.

```
as.data.frame.r4vn_meta
```

Convert an R4VN meta-analysis to a data frame

Description

Convert an R4VN meta-analysis to a data frame

Usage

```
## S3 method for class 'r4vn_meta'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

<code>x</code>	An object created by <code>tabmeta()</code> .
<code>row.names</code>	Ignored.
<code>optional</code>	Ignored.
<code>...</code>	Additional arguments ignored.

Value

The main publication table.

```
at
```

Specify predictor values for margins

Description

Creates predictor settings used by `margins()`.

Usage

```
at(...)
```

Arguments

<code>...</code>	Named predictor values. Supply scalars or vectors; multiple vectors are crossed automatically.
------------------	--

Value

An R4VN result object, invisibly.

See Also[margins](#)**Examples**

```
at(age = 40)
at(age = seq(30, 60, 5), hypertension = c(0, 1))
```

cii*Confidence Intervals from Summaries or Variables*

Description

Calculates confidence intervals for means, proportions, and variances.

Usage

```
cii(
  n,
  mean = NULL,
  sd = NULL,
  events = NULL,
  variance = NULL,
  type = c("auto", "mean", "proportion", "variance"),
  method = c("exact", "wilson", "wald"),
  level = 0.95,
  digits = 3,
  show = TRUE,
  console = FALSE
)

ci(
  x,
  data = NULL,
  type = c("auto", "mean", "proportion", "variance"),
  event = NULL,
  method = c("exact", "wilson", "wald"),
  level = 0.95,
  digits = 3,
  show = TRUE,
  console = FALSE
)
```

Arguments

n	Sample size.
mean, sd	Mean and standard deviation.
events	Number of events for a proportion.
variance	Sample variance.
type	"auto", "mean", "proportion", or "variance".
method	Proportion confidence interval method.
level	Confidence level.
digits	Decimal places.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
x	Variable to analyze.
data	Data frame. If NULL, active data is used.
event	Event level for a binary variable.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
cii(50, mean = 10, sd = 2)
cii(100, events = 45, type = "proportion", method = "wilson")
```

corr	<i>Correlation matrix</i>
------	---------------------------

Description

Computes Pearson, Spearman, or Kendall correlations from variables in a data frame. With `data = NULL`, the active data set is used.

Usage

```
corr(
  ...,
  data = NULL,
  method = c("pearson", "spearman", "kendall"),
  missing = c("pairwise", "listwise"),
  sig = FALSE,
  obs = FALSE,
  ci = FALSE,
  star = FALSE,
```

```

    level = 0.95,
    digits = 3,
    p_digits = 3,
    show = TRUE,
    console = FALSE
  )

```

Arguments

<code>...</code>	Numeric variables. If omitted, all numeric variables are used.
<code>data</code>	Data frame or NULL for active data.
<code>method</code>	Correlation method.
<code>missing</code>	Pairwise or listwise deletion.
<code>sig</code>	Show a p-value matrix.
<code>obs</code>	Show a matrix of pairwise sample sizes.
<code>ci</code>	Show pairwise confidence intervals where available.
<code>star</code>	Add significance stars to the displayed correlation matrix.
<code>level</code>	Confidence level.
<code>digits, p_digits</code>	Decimal places.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.

Value

An object of class `r4vn_stat`, returned invisibly. Its `sections` component contains the formatted correlation matrix and any requested p-value, pairwise sample-size, or confidence-interval tables. Its `raw` component contains the numeric correlation (`correlation`), p-value (`p.value`), and pairwise sample-size (`n`) matrices plus the selected correlation method; `call` records the matched function call.

Examples

```

# Extended usage examples
d <- data.frame(age = c(20, 25, 30, 35, 40, 45),
               bmi = c(20, 22, 24, 23, 26, 28),
               score = c(60, 65, 68, 72, 75, 80))

corr(age, bmi, score, data = d, show = FALSE)
corr(age, bmi, score, data = d, method = "spearman", show = FALSE)
corr(age, bmi, score, data = d, sig = TRUE, obs = TRUE, ci = TRUE, show = FALSE)
corr(age, bmi, score, data = d, star = TRUE, show = FALSE)

```

 cox

Direct Cox Proportional Hazards Model

Description

A compact R4VN wrapper around `tabsurv()` for a final multivariable Cox model.

Usage

```
cox(
  time,
  event,
  vars,
  data = NULL,
  failure = NULL,
  id = NULL,
  start = NULL,
  strata = NULL,
  cluster = NULL,
  frailty = NULL,
  ph = FALSE,
  ci = 0.95,
  ties = c("efron", "breslow", "exact"),
  diagnosis = FALSE,
  show = TRUE,
  console = FALSE
)
```

Arguments

<code>time</code>	Follow-up or stop-time variable, supplied without quotes.
<code>event</code>	Event/status variable, supplied without quotes.
<code>vars</code>	Optional predictor specification created by <code>vars()</code> .
<code>data</code>	Optional data frame. When omitted, active R4VN data are used.
<code>failure</code>	Value of event representing the event of interest. For a binary event it defaults to the second factor level or larger numeric value.
<code>id</code>	Optional subject identifier for counting-process/recurrent data.
<code>start</code>	Optional start/entry time. When supplied, <code>time</code> is treated as stop time.
<code>strata</code>	Optional stratification variable for Cox regression.
<code>cluster</code>	Optional clustering variable for robust Cox variance.
<code>frailty</code>	Optional shared-frailty variable. Do not combine with <code>cluster</code> .
<code>ph</code>	Logical; test the proportional-hazards assumption. Default FALSE in <code>cox()</code> . Setting <code>diagnosis = TRUE</code> also requests this diagnostic.
<code>ci</code>	Confidence level, default 0.95.

<code>ties</code>	Cox tie method: "efron", "breslow", or "exact".
<code>diagnosis</code>	Logical; if TRUE, append Cox-model diagnostics including concordance, proportional-hazards testing, and residual summaries. Default FALSE.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.

Value

An object of class `r4vn_surv`.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  d <- data.frame(
    time = c(5, 8, 10, 12, 15, 18, 20, 22, 25, 30),
    event = c(1, 0, 1, 1, 0, 1, 0, 1, 1, 0),
    age = c(40, 45, 50, 55, 60, 48, 52, 63, 58, 67),
    sex = factor(rep(c("Female", "Male"), 5))
  )
  cox(time, event, vars = vars(c.age, i.sex), data = d, show = FALSE)
  cox(time, event, vars = vars(c.age, i.sex), data = d,
    diagnosis = TRUE, show = FALSE)
  cox(time, event, vars = vars(c.age), data = d, ph = TRUE, show = FALSE)
}
```

<code>dengue_its_control</code>	<i>Example controlled interrupted time-series data</i>
---------------------------------	--

Description

Synthetic long-format controlled ITS dataset with a Control and an Intervention series observed monthly over the same 96-month period.

Usage

```
data(dengue_its_control)
```

Format

A data frame with 192 rows and 7 variables:

month Monthly date.

group Control or Intervention series.

cases Synthetic monthly case count.

rainfall Synthetic monthly rainfall in mm.

temperature Synthetic mean monthly temperature in degrees C.

population Synthetic population denominator.

intervention Indicator used for teaching; intervention applies to the Intervention group from January 2023 onward.

Source

Synthetic data generated for R4VN examples.

dengue_ts	<i>Example monthly dengue time series</i>
-----------	---

Description

A reproducible synthetic monthly time-series dataset for demonstrating `tabts()`. It contains 96 monthly observations from January 2018 through December 2025. The data were simulated for teaching and software testing; they do not represent a real surveillance system.

Usage

```
data(dengue_ts)
```

Format

- A data frame with 96 rows and 6 variables:
- month** Monthly date (first day of month).
 - cases** Synthetic monthly dengue case count.
 - rainfall** Synthetic monthly rainfall in mm.
 - temperature** Synthetic mean monthly temperature in degrees C.
 - population** Synthetic population denominator.
 - intervention** 0 before January 2023 and 1 thereafter.

Source

Synthetic data generated for R4VN examples.

describe	<i>Describe variables in the active data</i>
----------	--

Description

Displays a compact description of variable names, types, missing values, distinct values, labels, and factor levels. This is intended for quickly checking the structure of active data without opening an HTML file.

Usage

```
describe(..., data = NULL, show = TRUE, console = FALSE)
```

Arguments

...	Optional variables. If omitted, all variables are described. An explicit data frame may be supplied as the first unnamed argument.
data	Optional data frame. When omitted, active data is used.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional result in the Console. Default FALSE.

Value

Invisibly returns a data frame.

Examples

```
d <- data.frame(
  id = 1:4,
  sex = factor(c("Female", "Male", "Female", "Male")),
  age = c(25, 40, NA, 52)
)
attr(d$sex, "label") <- "Sex"
usedf(d, quiet = TRUE)
describe()
describe(sex, age)
usedf(clear = TRUE, quiet = TRUE)

# Extended usage examples

patient <- data.frame(
  id = 1:4,
  sex = factor(c("Female", "Male", "Female", "Male")),
  age = c(25, 40, NA, 52),
  outcome = c(FALSE, TRUE, FALSE, TRUE)
)
attr(patient$sex, "label") <- "Sex"
usedf(patient)

# Describe every variable or selected variables
describe()
describe(sex)
describe(sex, age, outcome)

# Explicit data and invisible returned metadata
describe(patient)
info <- describe(age, sex, data = patient, show = FALSE)
info
```

Description

Launch an interactive Shiny Studio for study-design guidance, sample-size planning, probability sampling, and random allocation. The working areas are independent: users may open any area directly and may optionally reuse or save results in one R4VN design project.

Usage

```
design(
  data = NULL,
  project_name = "Untitled Study",
  launch.browser = interactive(),
  port = NULL,
  host = "127.0.0.1"
)
```

Arguments

<code>data</code>	Optional <code>data.frame</code> used as the initial sampling/participant list.
<code>project_name</code>	Initial project name shown in the Studio.
<code>launch.browser</code>	Passed to <code>shiny::runApp()</code> .
<code>port</code>	Optional port passed to <code>shiny::runApp()</code> .
<code>host</code>	Host passed to <code>shiny::runApp()</code> .

Details

`design()` contains four independent working areas:

Study Design A guided cascade that recommends a design and explains why.

Sample Size A searchable registry of formula-, precision-, power-, and model-based sample-size methods. Results include a formula or method rule, numerical substitution when available, calculation steps, adjustments, sensitivity analyses, references, and a Methods statement.

Sampling Simple, systematic, stratified, PPS, cluster, and multistage sampling from an uploaded frame or, for methods that do not require frame variables, directly from generated IDs 1...N.

Randomization Simple, complete, permuted-block, and stratified-block random allocation from a participant list or from generated participant numbers, with reproducibility metadata and printable allocation lists.

Word export uses `rmarkdown/Pandoc` so mathematical formulas are exported as formatted Word equations. PDF export prefers `pagedown` with Chrome and uses a LaTeX fallback when necessary. Excel templates/workbooks use `openxlsx`.

Advanced sample-size methods use method-specific packages when available, including `presize`, `WebPower`, `statpsych`, `TrialSize`, and `pmsampsize`. The Studio reports the method/reference used rather than silently replacing an unavailable advanced method with an unrelated approximation.

Value

Invisibly returns the Shiny app result when the app exits.

Examples

```
if (interactive()) {  
  design()  
}
```

dict

Create an HTML data dictionary

Description

Creates a dictionary from an explicit data frame or active R4VN data.

Usage

```
dict(  
  ...,  
  data = NULL,  
  describe = TRUE,  
  file = NULL,  
  title = "Data dictionary",  
  open = TRUE  
)
```

Arguments

...	Optional variable selectors. For backward compatibility, an explicit data frame may be supplied first.
data	Optional explicit data frame. When omitted, active data is used.
describe	Logical; include compact descriptive summaries.
file	Output HTML file.
title	Dictionary title.
open	Logical; open the generated file.

Value

The HTML path invisibly.

Examples

```
d <- data.frame(sex = c(1, 2), age = c(20, NA))
usedf(d, quiet = TRUE)
path <- dict(file = tempfile(fileext = ".html"), open = FALSE)
file.exists(path)

# Extended usage examples

d <- data.frame(sex = c(1, 2, 1), age = c(20, 30, NA), bmi = c(21, 24, 26))
labvar(d, sex, label = "Sex", values = c("1" = "Male", "2" = "Female"))
labvar(d, age, label = "Age in years")

# Dictionary for every variable
f1 <- dict(d, file = tempfile(fileext = ".html"), open = FALSE)

# Dictionary for selected variables
f2 <- dict(d, sex, age, file = tempfile(fileext = ".html"), open = FALSE)

# Omit descriptive summaries
f3 <- dict(d, describe = FALSE, file = tempfile(fileext = ".html"), open = FALSE)

# Active-data syntax
usedf(d, quiet = TRUE)
f4 <- dict(file = tempfile(fileext = ".html"), open = FALSE)
```

distdata

Explore a probability distribution and calculate probabilities

Description

distdata() is the command-line probability-distribution calculator for R4VN. It combines distribution properties, point probabilities/densities, tail probabilities, interval probabilities, quantiles, simulation, and a Viewer-ready plot. [distlearn\(\)](#) provides the interactive Shiny companion.

Usage

```
distdata(
  distribution,
  ...,
  x = NULL,
  lower = NULL,
  upper = NULL,
  probs = NULL,
  nsim = 0L,
  seed = NULL,
  plot = TRUE,
  digits = 6L,
```

```

    show = TRUE,
    console = FALSE
  )

```

Arguments

distribution	Distribution name. Common abbreviations are accepted, for example "normal", "binomial", "poisson", "t", "chisq", "weibull", "betabinom", and "zip".
...	Named parameters of the selected distribution. For example, n and p for binomial, mean and sd for normal, or lambda for Poisson. Defaults are teaching-friendly and are shown by distlearn().
x	Optional value(s). For discrete distributions R4VN reports $P(X=x)$, $<$, $<=$, $>$, and $>=$. For continuous distributions it reports density $f(x)$ and the corresponding tail probabilities.
lower, upper	Optional interval bounds for an interval probability.
probs	Optional cumulative probabilities for which quantiles are requested, for example <code>probs = c(.025, .5, .975)</code> .
nsim	Optional number of random observations to simulate. 0 (the default) does not simulate.
seed	Optional user-supplied random seed for simulation. The default NULL does not set a seed.
plot	Logical; retain plot data and display the probability function in the R4VN Viewer.
digits	Number of significant digits in numerical probability output.
show	Logical; open the R4VN Viewer result.
console	Logical; also print the tabular result to the console.

Details

A particularly useful teaching call is `distdata("binomial", n = 10, x = 3, p = .2)`. With only these inputs R4VN reports $P(X=3)$, $P(X<3)$, $P(X<=3)$, $P(X>3)$, and $P(X>=3)$, together with the distribution's mean, variance, standard deviation, parameters, support, and plot. Supplying lower and upper adds the probability inside and outside an interval; supplying probs adds quantiles.

The beta-binomial accepts either shape1/shape2 or the more interpretable pair p/ρ . The negative binomial accepts size with either p or μ . Gamma accepts rate or scale.

Value

Invisibly returns an object of classes `r4vn_distdata` and `r4vn_stat`. Its raw component contains the distribution specification, parameters, point/range probabilities, quantiles, simulation, and plot grid.

Examples

```

distdata("binomial", n = 10, x = 3, p = .2, show = FALSE)
distdata("binomial", n = 20, p = .35, lower = 5, upper = 10, show = FALSE)
distdata("normal", mean = 100, sd = 15, x = 130, show = FALSE)

```

```

distdata("normal", mean = 100, sd = 15, probs = c(.025, .5, .975), show = FALSE)
distdata("poisson", lambda = 2.5, x = 0:4, show = FALSE)
distdata("nbinom", size = 2, mu = 5, x = 0:4, show = FALSE)
distdata("betabinom", n = 20, p = .3, rho = .1, x = 0:5, show = FALSE)

```

distlearn

R4VN Distribution Learning Studio

Description

Launches an interactive Shiny application for learning probability distributions. The interface is inspired by the idea of a distribution explorer, but is designed around the R4VN teaching workflow: change parameters, immediately see the distribution, calculate commonly used probability statements, read historical and practical context, simulate data, and copy an equivalent `distdata()` command.

Usage

```

distlearn(
  distribution = "normal",
  launch.browser = interactive(),
  port = NULL,
  host = "127.0.0.1"
)

```

Arguments

<code>distribution</code>	Initial distribution name. Defaults to "normal".
<code>launch.browser</code>	Passed to <code>shiny::runApp()</code> .
<code>port</code>	Optional port passed to <code>shiny::runApp()</code> .
<code>host</code>	Host passed to <code>shiny::runApp()</code> .

Details

The Studio currently includes the continuous distributions Normal, Log-normal, Uniform, Exponential, Gamma, Beta, Chi-square, Student's t, F, Weibull, Logistic, Cauchy, Laplace, Gumbel, Pareto, Rayleigh, Triangular, Kumaraswamy, Log-logistic, and Half-normal; and the discrete distributions Bernoulli, Binomial, Poisson, Geometric, Negative binomial, Hypergeometric, Discrete uniform, Beta-binomial, Zero-inflated Poisson, finite Zipf, and Logarithmic series.

Each distribution has an interactive probability/density plot and CDF, numerical characteristics, parameter explanations, a historical note, typical applications, cautions, probability/quantile calculation, and simulation. Parameter values may be controlled with sliders or typed directly using numeric inputs.

Value

Invisibly returns the result of `shiny::runApp()` when the app exits.

Examples

```
if (interactive()) {
  distlearn("binomial")
}
```

dropvar

Drop variables and/or observations

Description

Drops variables or observations from an explicit data frame or active data.

Usage

```
dropvar(..., data = NULL, obs = NULL)
```

Arguments

...	Variable selectors. For backward compatibility, an explicit data frame may be the first unnamed argument.
data	Optional explicit data frame object.
obs	Optional observations to drop.

Value

The edited data frame invisibly.

Examples

```
d <- data.frame(id = 1:4, age = c(10, 20, NA, 40), note_kt = letters[1:4])
usedf(d, quiet = TRUE)
dropvar("*_kt")
dropvar(obs = age < 18)

# Extended usage examples

d <- data.frame(id = 1:5, age = c(10, 20, NA, 40, 50),
               temp_a = 1:5, temp_b = 6:10, note = letters[1:5])

# Drop one or more variables
d1 <- d; dropvar(d1, note)
d2 <- d; dropvar(d2, temp_a, temp_b)

# Drop variables with a wildcard
d3 <- d; dropvar(d3, "temp_*")

# Drop observations satisfying a condition
d4 <- d; dropvar(d4, obs = age < 18)
```

```
d5 <- d; dropvar(d5, obs = missing(age))

# Drop variables and observations together
d6 <- d; dropvar(d6, note, obs = age < 18)

# Active-data syntax
usedf(d, quiet = TRUE); dropvar("temp_*"); dropvar(obs = missing(age))
```

egenvar

*Extended variable generation***Description**

Creates row-wise, group-wise, ranking, standardization, and identifier variables using a compact syntax inspired by Stata's `egen`, while following the active-data conventions of [genvar\(\)](#).

Usage

```
egenvar(..., data = NULL, by = NULL, label = NULL, values = NULL)
```

Arguments

<code>...</code>	Named expressions in the form <code>new_variable = function(...)</code> . An explicit data frame may also be supplied first for compatibility with other R4VN data-editing commands.
<code>data</code>	Optional explicit data frame. When omitted, the active data frame selected by usedf() is used.
<code>by</code>	Optional grouping variables, for example <code>by = sex</code> or <code>by = vars(site, sex)</code> . Group summaries are repeated on every observation in the corresponding group.
<code>label</code>	Optional variable label or one label per generated variable.
<code>values</code>	Optional named value labels applied to generated variables.

Details

Row-wise functions accept individual variables, variable ranges, `vars()` selections, and wildcard selectors: `rowmin()`, `rowmax()`, `rowmean()`, `rowsum()`, `rowmedian()`, `rowstd()`, `rowmiss()`, `rownonmiss()`, `rowfirst()`, and `rowlast()`. Thus `rowmean(q1:q10)` is valid R4VN syntax.

Group/overall functions are `mean()`, `sd()`, `min()`, `max()`, `median()`, `total()`, `count()`, `n()`, `seq()`, `z()`, `pctile()`, and `rank()`. Without `by`, they operate over the complete data set. With `by`, they operate separately within groups. Missing values are ignored by summary functions; `count()` counts non-missing values.

Special identifier functions are `group()` and `tag()`. They use one or more variables supplied inside the function and do not depend on the `by` argument. `group(site, sex)` gives consecutive integer IDs for observed combinations; `tag(id)` marks the first occurrence of each distinct value.

Value

The edited data frame invisibly. If active data are linked to a visible object through `usedf()`, that object is updated as well.

Examples

```
d <- data.frame(
  id = c(1, 1, 2, 3),
  sex = c("F", "F", "M", "M"),
  q1 = c(2, 4, 3, NA), q2 = c(3, 5, 2, 4), q3 = c(4, NA, 1, 5),
  bmi = c(20, 22, 25, 27)
)
egenvar(d, min_score = rowmin(q1:q3),
        max_score = rowmax(q1:q3),
        mean_score = rowmean(q1:q3))

egenvar(d, mean_bmi = mean(bmi), z_bmi = z(bmi), by = sex)
egenvar(d, p75_bmi = pctlile(bmi, p = 75), rank_bmi = rank(bmi), by = sex)
egenvar(d, n_group = n(), sequence = seq(), by = sex)
egenvar(d, person_group = group(id, sex), first_id = tag(id))
```

Description

`epii()` analyzes typed 2 by 2 counts. `epi()` analyzes binary outcome and exposure variables. With `by`, stratum-specific estimates, a Mantel-Haenszel common odds ratio, a pooled risk ratio, homogeneity, and interaction tests are displayed. The stratified output also reports the crude-versus-Mantel-Haenszel OR difference as both $100 \times (\text{OR}_{\text{crude}} - \text{ORMH}) / \text{ORMH}$ and $100 \times (\text{OR}_{\text{crude}} - \text{ORMH}) / \text{OR}_{\text{crude}}$. `cci()/cc()` and `csi()/cs()` are familiar aliases.

Usage

```
epi(
  outcome,
  exposure,
  by = NULL,
  data = NULL,
  event = NULL,
  exposed = NULL,
  level = 0.95,
  correction = 0.5,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)
```

```
cc(  
  outcome,  
  exposure,  
  by = NULL,  
  data = NULL,  
  event = NULL,  
  exposed = NULL,  
  level = 0.95,  
  correction = 0.5,  
  digits = 3,  
  p_digits = 3,  
  show = TRUE,  
  console = FALSE  
)
```

```
cs(  
  outcome,  
  exposure,  
  by = NULL,  
  data = NULL,  
  event = NULL,  
  exposed = NULL,  
  level = 0.95,  
  correction = 0.5,  
  digits = 3,  
  p_digits = 3,  
  show = TRUE,  
  console = FALSE  
)
```

```
epii(  
  a = NULL,  
  b = NULL,  
  c = NULL,  
  d = NULL,  
  by = NULL,  
  level = 0.95,  
  correction = 0.5,  
  digits = 3,  
  p_digits = 3,  
  show = TRUE,  
  console = FALSE  
)
```

```
cci(a, b, c, d, ...)
```

```
csi(a, b, c, d, ...)
```

```

epi(
  outcome,
  exposure,
  by = NULL,
  data = NULL,
  event = NULL,
  exposed = NULL,
  level = 0.95,
  correction = 0.5,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

```

Arguments

outcome	Binary outcome variable.
exposure	Binary exposure variable.
by	For <code>epii()</code> , a list of 2 by 2 tables or a 2 by 2 by K array. For <code>epi()</code> , an optional stratification variable. Hierarchical syntax is supported: <code>by = vars(province, sex)</code> repeats the epidemiological analysis within province and uses sex as the innermost Mantel-Haenszel stratum. With three or more variables, every variable before the final one is an ordered outer stratum and the final variable is the MH stratification variable.
data	Data frame. If NULL, active data is used.
event	Outcome event level; defaults to the last observed level.
exposed	Exposure level; defaults to the last observed level.
level	Confidence level.
correction	Continuity correction used for log-scale confidence intervals when a cell is zero.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
a, b, c, d	Cell counts of a 2 by 2 table. a may also be a 2 by 2 matrix.
...	Additional arguments passed to <code>epii()</code> by immediate aliases.

Details

The 2 by 2 layout is exposure by outcome: a exposed cases, b exposed noncases, c unexposed cases, and d unexposed noncases.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
epii(40, 10, 20, 30)
epii(by = list(
  Female = c(12, 18, 8, 32),
  Male = c(28, 12, 12, 18)
))
```

epitool

R4VN EpiTool Studio

Description

Opens an interactive field epidemiology workspace for FETP-style work. EpiTool combines epidemiologic calculators, outbreak workflows, surveillance, universal file templates and validation, and optional spatial epidemiology.

Usage

```
epitool(
  data = NULL,
  mode = c("field", "teaching"),
  level = c("frontline", "advanced"),
  launch.browser = interactive(),
  host = "127.0.0.1",
  port = NULL
)
```

Arguments

<code>data</code>	Optional data frame. If <code>NULL</code> , EpiTool attempts to use the active R4VN data set selected with <code>usedf()</code> .
<code>mode</code>	Initial interface mode: "field" or "teaching".
<code>level</code>	Initial complexity level: "frontline" or "advanced".
<code>launch.browser</code>	Passed to <code>shiny::runApp()</code> .
<code>host</code>	Host passed to <code>shiny::runApp()</code> .
<code>port</code>	Optional Shiny port.

Details

File-first usability:

Every EpiTool workflow that needs external tabular data is paired with a downloadable Excel template containing DATA, DICTIONARY and INSTRUCTIONS sheets. Exact column names are not mandatory because EpiTool includes a column mapper and validation layer.

Spatial epidemiology:

Spatial functions are optional. When `sf` and `leaflet` are installed, EpiTool supports case maps, source buffers, nearest-source distance, density grids, DBSCAN point clusters, point-to-area spatial joins, population-based area rates, Local Moran's I and Getis-Ord G^* analysis.

Density, rate and hotspot outputs are deliberately separated because they answer different epidemiologic questions. A density concentration or statistical hotspot should not be interpreted automatically as a causal source or confirmed outbreak.

Privacy:

Presentation mode can jitter displayed case coordinates. Analysis should continue to use the original coordinates.

Value

Invisibly returns the Shiny app after it exits.

Examples

```
if (interactive()) {
  epitool()

  outbreak <- data.frame(
    case_id = 1:20,
    onset_date = as.Date("2026-08-01") + sample(0:7, 20, TRUE),
    latitude = 10.76 + rnorm(20, 0, .005),
    longitude = 106.66 + rnorm(20, 0, .005)
  )
  usedf(outbreak, quiet = TRUE)
  epitool(mode = "teaching")
}
```

 esizei

Standardized Mean Differences

Description

Calculates Cohen's d, Hedges' g, and Glass's delta.

Usage

```
esizei(
  n1,
  mean1,
  sd1,
  n2,
  mean2,
  sd2,
  level = 0.95,
```

```

    digits = 3,
    show = TRUE,
    console = FALSE
  )

  esize(
    x,
    by,
    data = NULL,
    level = 0.95,
    digits = 3,
    show = TRUE,
    console = FALSE
  )

```

Arguments

n1, mean1, sd1	Summary statistics for group 1.
n2, mean2, sd2	Summary statistics for group 2.
level	Confidence level.
digits	Decimal places.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
x	Numeric variable.
by	Two-level grouping variable.
data	Data frame. If NULL, active data is used.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
esizei(40, 12, 3, 35, 14, 4)
```

friedman

Friedman Test for Repeated or Matched Measurements

Description

Performs the Friedman rank-sum test for three or more repeated or matched measurements. Data may be supplied in wide form as several numeric variables, or in long form as one outcome, one occasion/treatment variable, and one subject/block identifier.

Usage

```

friedman(
  ...,
  by = NULL,
  id = NULL,
  data = NULL,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

```

Arguments

<code>...</code>	In wide form, two or more numeric repeated-measure variables. In long form, exactly one numeric outcome variable.
<code>by</code>	Long-form occasion or treatment variable.
<code>id</code>	Long-form subject or block identifier.
<code>data</code>	Data frame. If <code>NULL</code> , active R4VN data is used.
<code>digits, p_digits</code>	Decimal places for estimates and p-values.
<code>show</code>	Logical; open the formatted result in the Viewer. Default <code>TRUE</code> .
<code>console</code>	Logical; also print the traditional result in the Console. Default <code>FALSE</code> .

Details

Wide form uses only rows complete across every repeated measurement. Long form must contain one observation for every subject-by-occasion combination; incomplete blocks are rejected by the underlying Friedman test.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```

wide <- data.frame(
  baseline = c(22, 20, 19, 25, 23, 21),
  month1   = c(20, 18, 18, 22, 21, 20),
  month3   = c(18, 17, 16, 20, 19, 18)
)
friedman(baseline, month1, month3, data = wide)

long <- data.frame(
  id = rep(1:6, each = 3),
  time = factor(rep(c("Baseline", "Month 1", "Month 3"), 6),
    levels = c("Baseline", "Month 1", "Month 3")),
  score = as.vector(t(as.matrix(wide)))
)

```

```

)
friedman(score, by = time, id = id, data = long)

# Extended usage examples

wide <- data.frame(
  baseline = c(22, 20, 19, 25, 23, 21),
  month1 = c(20, 18, 18, 22, 21, 20),
  month3 = c(18, 17, 16, 20, 19, 18)
)

# Wide form: each row is a subject and each variable is an occasion
friedman(baseline, month1, month3, data = wide)

# Long form: outcome, occasion, and subject identifier
long <- data.frame(
  id = rep(1:6, each = 3),
  time = factor(rep(c("Baseline", "Month 1", "Month 3"), 6),
    levels = c("Baseline", "Month 1", "Month 3")),
  score = as.vector(t(as.matrix(wide)))
)
friedman(score, by = time, id = id, data = long)

usedf(wide)
friedman(baseline, month1, month3)

```

gbar

Bar chart

Description

Draws counts, percentages, means, or medians by a categorical variable. An optional by variable creates grouped or stacked bars. The function uses base R graphics and accepts unquoted variable names. `vars()` can request several x variables. With `by = vars(province, sex, outcome)`, province and sex are nested strata and outcome is the innermost in-graph grouping.

Usage

```

gbar(
  data = NULL, x = NULL, vars = NULL, y = NULL, by = NULL,
  stat = c("mean", "median"), percent = FALSE, position = c("dodge", "stack"),
  ci = FALSE, label = FALSE, digits = 1, xlab = NULL, ylab = NULL,
  xtitle = NULL, ytitle = NULL, ybreaks = NULL, title = NULL, subtitle = NULL,
  note = NULL, color = NULL, palette = "default", alpha = 1,
  border_color = NA, border_lwd = 1, legend = TRUE,
  legend_position = "topright", missing = FALSE, missing_label = "Missing",
  xline = NULL, yline = NULL, ref_color = "gray40", ref_lty = 2, ref_lwd = 1,
  theme = "journal", size = 11, combine = FALSE, ncol = NULL, file = NULL,

```

```

width = 7, height = 5, dpi = 300, show = TRUE, bg = "white", vline = NULL,
hline = NULL
)

```

Arguments

<code>data</code>	A data frame. It may be omitted when an active R4VN data frame exists.
<code>x</code>	Categorical variable on the horizontal axis. Supply an unquoted name or a character name.
<code>vars</code>	Optional <code>vars(...)</code> selector used to request several variables in one call.
<code>y</code>	Optional numeric variable. When omitted, bars show counts or percentages. When supplied, bars show a mean or median.
<code>by</code>	Optional grouping variable or hierarchical <code>vars(...)</code> specification.
<code>stat</code>	Summary for y: "mean" or "median".
<code>percent</code>	For count charts: FALSE for counts, TRUE or "x" for percentages within each x category, "by" for percentages within each by group, or "total" for percentages of all observations.
<code>position</code>	"dodge" for side-by-side bars or "stack" for stacked bars.
<code>ci</code>	Show a 95 percent confidence interval for mean bars. Only available when y is supplied and <code>stat = "mean"</code> .
<code>label</code>	Add values above or inside bars.
<code>digits</code>	Number of digits in value labels.
<code>xlab</code>	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
<code>ylab</code>	Optional y-axis tick labels.
<code>xtitle, ytitle</code>	Axis titles. Variable names or variable labels are used automatically when possible.
<code>ybreaks</code>	Numeric positions used with <code>ylab</code> .
<code>title, subtitle, note</code>	Main title, subtitle, and note.
<code>color</code>	A color name or vector of colors. When NULL, <code>palette</code> is used.
<code>palette</code>	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.
<code>alpha</code>	Color opacity from 0 to 1.
<code>border_color, border_lwd</code>	Bar-border colour and line width. Use NA for no border.
<code>legend</code>	Show the legend when <code>by</code> is supplied.
<code>legend_position</code>	Base-R legend position, for example "topright" or "topleft".
<code>missing</code>	Include missing x or by values as a category.
<code>missing_label</code>	Label used for missing values.

xline, yline	Optional numeric reference lines on the x and y axes. Thus xline draws vertical lines and yline draws horizontal lines.
ref_color, ref_lty, ref_lwd	Color, line type, and width for reference lines; vectors are recycled for multiple lines.
theme	Graph theme: "journal", "clean", "minimal", or "classic".
size	Base font size.
combine	When several graphs are produced, draw them as labelled panels in one figure.
ncol	Optional number of columns in a combined figure.
file	Optional output file ending in png, jpg, tiff, pdf, or svg.
width, height	Output width and height in inches.
dpi	Resolution for raster output.
show	Draw the graph in the current graphics device.
bg	Background color for exported files.
vline, hline	Deprecated aliases for xline and yline.

Value

An object of class `r4vn_graph`. Its data component contains the plotted summary.

Examples

```
d <- data.frame(
  sex = factor(c("Male", "Female", "Female", "Male", "Female")),
  hypertension = factor(c("No", "Yes", "No", "Yes", "No")),
  age = c(32, 45, 37, 51, 29)
)
gbar(d, x = sex)
gbar(d, x = sex, by = hypertension, percent = "x")
gbar(d, x = sex, y = age, ci = TRUE, ytitle = "Mean age")

# Extended usage examples

d <- data.frame(sex = factor(c("Male", "Female", "Female", "Male", "Female")),
  outcome = factor(c("No", "Yes", "No", "Yes", "No")),
  age = c(32, 45, 37, 51, 29))
gbar(d, x = sex, show = FALSE)
gbar(d, x = sex, percent = TRUE, label = TRUE, show = FALSE)
gbar(d, x = sex, by = outcome, percent = "x", position = "dodge", show = FALSE)
gbar(d, x = sex, by = outcome, percent = "total", position = "stack", show = FALSE)
gbar(d, x = sex, y = age, stat = "mean", ci = TRUE, show = FALSE)
gbar(d, x = sex, y = age, stat = "median", show = FALSE)
gbar(d, x = sex, title = "Participants by sex", palette = "journal", show = FALSE)
f <- tempfile(fileext = ".png")
gbar(d, x = sex, file = f, show = FALSE)
```

gbox

*Box plot***Description**

Draws one or several numeric distributions overall or across categories. `vars()` can request several numeric outcomes; hierarchical `by` is supported.

Usage

```
gbox(
  data = NULL, x = NULL, y = NULL, vars = NULL, by = NULL, horizontal = FALSE,
  points = FALSE, outliers = TRUE, xlab = NULL, ylab = NULL, xtitle = NULL,
  ytitle = NULL, xbreaks = NULL, ybreaks = NULL, title = NULL,
  subtitle = NULL, note = NULL, color = NULL, palette = "default",
  alpha = 0.7, border_color = "gray30", point_color = "black",
  point_alpha = 0.45, point_size = 0.55, point_pch = 16, xline = NULL,
  yline = NULL, ref_color = "gray40", ref_lty = 2, ref_lwd = 1,
  theme = "journal", size = 11, combine = FALSE, ncol = NULL, file = NULL,
  width = 7, height = 5, dpi = 300, show = TRUE, bg = "white", vline = NULL,
  hline = NULL
)
```

Arguments

<code>data</code>	A data frame. It may be omitted when an active R4VN data frame exists.
<code>x</code>	Optional categorical grouping variable.
<code>y</code>	Numeric outcome variable.
<code>vars</code>	Optional <code>vars(...)</code> selector used to request several variables in one call.
<code>by</code>	Optional grouping variable or hierarchical <code>vars(...)</code> specification.
<code>horizontal</code>	Draw horizontal boxes.
<code>points</code>	Add lightly jittered observations.
<code>outliers</code>	Show conventional box-plot outliers.
<code>xlab</code>	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
<code>ylab</code>	Optional y-axis tick labels.
<code>xtitle, ytitle</code>	Axis titles. Variable names or variable labels are used automatically when possible.
<code>xbreaks, ybreaks</code>	Numeric tick positions used with <code>xlab</code> and <code>ylab</code> .
<code>title, subtitle, note</code>	Main title, subtitle, and note.
<code>color</code>	A color name or vector of colors. When <code>NULL</code> , <code>palette</code> is used.

palette	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.
alpha	Color opacity from 0 to 1.
border_color	Box/whisker border colour.
point_color, point_alpha, point_size, point_pch	Colour, transparency, size, and point symbol for overlaid raw observations.
xline, yline	Optional numeric reference lines on the x and y axes. Thus xline draws vertical lines and yline draws horizontal lines.
ref_color, ref_lty, ref_lwd	Color, line type, and width for reference lines; vectors are recycled for multiple lines.
theme	Graph theme: "journal", "clean", "minimal", or "classic".
size	Base font size.
combine	When several graphs are produced, draw them as labelled panels in one figure.
ncol	Optional number of columns in a combined figure.
file	Optional output file ending in png, jpg, tiff, pdf, or svg.
width, height	Output width and height in inches.
dpi	Resolution for raster output.
show	Draw the graph in the current graphics device.
bg	Background color for exported files.
vline, hline	Deprecated aliases for xline and yline.

Value

An object of class `r4vn_graph`; its data component contains group sample sizes and five-number summaries.

Examples

```
d <- data.frame(sex = factor(c("Male", "Female", "Female", "Male")), bmi = c(22, 24, 27, 21))
gbox(d, y = bmi)
gbox(d, x = sex, y = bmi, points = TRUE)

# Extended usage examples

d <- data.frame(sex = factor(c("Male", "Female", "Female", "Male")),
               bmi = c(22, 24, 27, 21))
gbox(d, y = bmi, show = FALSE)
gbox(d, x = sex, y = bmi, points = TRUE, show = FALSE)
gbox(d, x = sex, y = bmi, horizontal = TRUE, outliers = FALSE, show = FALSE)
```

gdensity	<i>Density plot</i>
----------	---------------------

Description

Draws a kernel density estimate overall or by a categorical variable.

Usage

```
gdensity(
  data = NULL, x = NULL, vars = NULL, by = NULL, adjust = 1, fill = FALSE,
  line_width = 2, line_type = 1, xlab = NULL, ylab = NULL, xtitle = NULL,
  ytitle = "Density", xbreaks = NULL, ybreaks = NULL, title = NULL,
  subtitle = NULL, note = NULL, color = NULL, palette = "default",
  alpha = 0.35, legend = TRUE, legend_position = "topright", xline = NULL,
  yline = NULL, ref_color = "gray40", ref_lty = 2, ref_lwd = 1,
  theme = "journal", size = 11, combine = FALSE, ncol = NULL, file = NULL,
  width = 7, height = 5, dpi = 300, show = TRUE, bg = "white", vline = NULL,
  hline = NULL
)
```

Arguments

<code>data</code>	A data frame. It may be omitted when an active R4VN data frame exists.
<code>x</code>	Categorical variable on the horizontal axis. Supply an unquoted name or a character name.
<code>vars</code>	Optional <code>vars(...)</code> selector used to request several variables in one call.
<code>by</code>	Optional grouping variable or hierarchical <code>vars(...)</code> specification.
<code>adjust</code>	Bandwidth adjustment passed to <code>stats::density()</code> .
<code>fill</code>	Fill the area under each curve.
<code>line_width, line_type</code>	Width and line type of density curves.
<code>xlab</code>	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
<code>ylab</code>	Optional y-axis tick labels.
<code>xtitle, ytitle</code>	Axis titles. Variable names or variable labels are used automatically when possible.
<code>xbreaks, ybreaks</code>	Numeric tick positions used with <code>xlab</code> and <code>ylab</code> .
<code>title, subtitle, note</code>	Main title, subtitle, and note.
<code>color</code>	A color name or vector of colors. When <code>NULL</code> , <code>palette</code> is used.
<code>palette</code>	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.

<code>alpha</code>	Color opacity from 0 to 1.
<code>legend</code>	Show the legend when <code>by</code> is supplied.
<code>legend_position</code>	Base-R legend position, for example "topright" or "topleft".
<code>xline, yline</code>	Optional numeric reference lines on the x and y axes. Thus <code>xline</code> draws vertical lines and <code>ylines</code> draws horizontal lines.
<code>ref_color, ref_lty, ref_lwd</code>	Color, line type, and width for reference lines; vectors are recycled for multiple lines.
<code>theme</code>	Graph theme: "journal", "clean", "minimal", or "classic".
<code>size</code>	Base font size.
<code>combine</code>	When several graphs are produced, draw them as labelled panels in one figure.
<code>ncol</code>	Optional number of columns in a combined figure.
<code>file</code>	Optional output file ending in png, jpg, tiff, pdf, or svg.
<code>width, height</code>	Output width and height in inches.
<code>dpi</code>	Resolution for raster output.
<code>show</code>	Draw the graph in the current graphics device.
<code>bg</code>	Background color for exported files.
<code>vline, hline</code>	Deprecated aliases for <code>xline</code> and <code>ylines</code> .

Details

Draws densities for one or several numeric variables with optional hierarchical grouping.

Value

An object of class `r4vn_graph`; its data component contains density coordinates.

Examples

```
d <- data.frame(age = c(18, 21, 25, 27, 31, 34, 38, 45, 51, 62), sex = rep(c("M", "F"), 5))
gdensity(d, x = age)
gdensity(d, x = age, by = sex, fill = TRUE)

# Extended usage examples

d <- data.frame(age = c(18, 21, 25, 27, 31, 34, 38, 45, 51, 62),
                sex = rep(c("M", "F"), 5))
gdensity(d, x = age, show = FALSE)
gdensity(d, x = age, by = sex, show = FALSE)
gdensity(d, x = age, by = sex, fill = TRUE, adjust = 1.2, show = FALSE)
```

genvar	<i>Generate one or more variables</i>
--------	---------------------------------------

Description

Creates variables in an explicit data frame or the active R4VN data frame. Logical expressions are stored as 1 and 0. If no active data exists, `genvar()` can initialize a temporary active data frame from entered vectors. A later vector may contain more observations than the current data; R4VN expands the data frame and pads existing/shorter columns with missing values.

Usage

```
genvar(
  ..., data = NULL, label = NULL, values = NULL, recode = NULL, ref = NULL,
  ordered = FALSE, times = NULL, each = NULL, fill = NA
)
```

Arguments

<code>...</code>	Named expressions in the form <code>new_variable = expression</code> . For backward compatibility, an explicit data frame may be supplied first.
<code>data</code>	Optional explicit data frame object. When omitted, active data is used.
<code>label</code>	A character label or one label per generated variable.
<code>values</code>	A common named value-label vector.
<code>recode</code>	An optional common named recode vector.
<code>ref</code>	Optional reference category.
<code>ordered</code>	Logical; create ordered factors.
<code>times</code>	Optional repetition counts. For one variable, <code>genvar(smoking = c(1, 0), times = c(10, 10))</code> creates ten 1s followed by ten 0s without writing <code>rep()</code> manually. With several variables, use a named list.
<code>each</code>	Optional compact repetition. For example, <code>genvar(group = c(1, 2, 3), each = 5)</code> creates five observations per group.
<code>fill</code>	Value used to pad a generated variable when it is shorter than the current data. Default NA. If a new variable is longer than the data, existing columns are extended with missing values.

Value

The edited data frame invisibly.

Examples

```
d <- data.frame(sbp = c(120, 150), dbp = c(75, 95))
usedf(d, quiet = TRUE)
genvar(hypertension = sbp >= 140 | dbp >= 90,
       label = "Hypertension",
       values = c("0" = "No", "1" = "Yes"))

# Extended usage examples

d <- data.frame(id = 1:4, age = c(17, 25, 40, 70),
               sbp = c(118, 145, 132, 160),
               dbp = c(75, 92, 80, 95), bmi = c(18, 22, 25, 30))
usedf(d, quiet = TRUE)

# Arithmetic expression
genvar(age_decade = age / 10, label = "Age in decades")

# Logical expression is stored as 0/1
genvar(hypertension = sbp >= 140 | dbp >= 90,
       label = "Hypertension",
       values = c("0" = "No", "1" = "Yes"), ref = "No")

# Generate several variables in one call
genvar(adult = age >= 18, overweight = bmi >= 23,
       label = c("Adult", "Overweight or obesity"),
       values = c("0" = "No", "1" = "Yes"))

# Generate and recode a grouped variable
genvar(age_group = age,
       recode = c("min:17" = 1, "18:59" = 2, "60:max" = 3),
       label = "Age group",
       values = c("1" = "<18", "2" = "18-59", "3" = "60+"),
       ordered = TRUE)

# Explicit-data syntax remains available
genvar(d, pulse_pressure = sbp - dbp, label = "Pulse pressure")

# Quick vector entry without creating a data frame first
genvar(weight = c(29, 26, 13, 23, 23, 25, 17, 22))
ghist(x = weight)

# A later variable may be longer; existing columns are padded with NA
genvar(age = c(81, 65, 89, 70, 87, 61, 94, 98, 81, 70))

# Compact repeated values
genvar(smoking = c(1, 0), times = c(10, 10),
       values = c("0" = "No", "1" = "Yes"))
```

Description

Draws estimates and confidence intervals on a linear or logarithmic scale. It is suitable for odds ratios, risk ratios, prevalence ratios, hazard ratios, and regression coefficients.

Usage

```
gforest(  
  data = NULL,  
  estimate,  
  lower,  
  upper,  
  label,  
  reference = 1,  
  log = TRUE,  
  sort = FALSE,  
  digits = 2,  
  xtitle = NULL,  
  title = NULL,  
  subtitle = NULL,  
  note = NULL,  
  color = NULL,  
  palette = "journal",  
  point_size = 1.1,  
  line_width = 2,  
  reference_color = "gray50",  
  reference_lty = 2,  
  reference_lwd = 1,  
  theme = "journal",  
  size = 11,  
  file = NULL,  
  width = 7,  
  height = 5,  
  dpi = 300,  
  show = TRUE,  
  bg = "white"  
)
```

Arguments

data	A data frame.
estimate	Point estimate variable.
lower, upper	Lower and upper confidence-limit variables.
label	Row-label variable.
reference	Reference value, commonly 1 for ratios and 0 for coefficients.
log	Use a logarithmic horizontal axis.
sort	Sort by estimate: FALSE, TRUE, "ascending", or "descending".

digits Number of displayed digits.
 xtitle Horizontal-axis title.
 title, subtitle, note Main title, subtitle, and note.
 color Point and interval color.
 palette Color palette used when color = NULL.
 point_size Point-size multiplier.
 line_width Confidence-interval line width.
 reference_color, reference_lty, reference_lwd Color, line type, and width of the vertical reference line.
 theme, size, file, width, height, dpi, show, bg See gbar().

Value

An object of class `r4vn_graph`; its data component contains plotted rows.

Examples

```

d <- data.frame(
  term = c("Smoking", "Obesity", "Male"),
  or = c(1.8, 2.4, 1.2),
  lower = c(1.2, 1.5, 0.8),
  upper = c(2.7, 3.8, 1.8)
)
gforest(d, estimate = or, lower = lower, upper = upper, label = term)

# Extended usage examples

# Ratio measures use reference = 1 and a logarithmic axis
d <- data.frame(
  term = c("Smoking", "Obesity", "Male"),
  estimate = c(1.80, 2.40, 1.20),
  lower = c(1.20, 1.50, 0.80),
  upper = c(2.70, 3.80, 1.80)
)
gforest(d, estimate, lower, upper, term, show = FALSE)
gforest(d, estimate, lower, upper, term,
  sort = "descending", title = "Adjusted odds ratios", show = FALSE)

# Regression coefficients use reference = 0 and a linear axis
beta <- data.frame(term = c("Age", "BMI", "Male"),
  estimate = c(0.12, 0.34, -0.18),
  lower = c(0.04, 0.10, -0.45),
  upper = c(0.20, 0.58, 0.09))
gforest(beta, estimate, lower, upper, term,
  reference = 0, log = FALSE, xtitle = "Regression coefficient",
  show = FALSE)

# Export directly to a graphics file

```

```
gforest(d, estimate, lower, upper, term,
        file = tempfile(fileext = ".png"), show = FALSE)
```

ghist

Histogram

Description

Draws histograms for one or several numeric variables using base R graphics. `by` may be a single grouping variable or hierarchical `vars(...)`; histogram panels are created for observed group/stratum combinations.

Usage

```
ghist(
  data = NULL, x = NULL, vars = NULL, by = NULL, bins = "Sturges",
  density = FALSE, normal = FALSE, xlab = NULL, ylab = NULL, xtitle = NULL,
  ytitle = NULL, xbreaks = NULL, ybreaks = NULL, title = NULL,
  subtitle = NULL, note = NULL, color = NULL, palette = "default",
  alpha = 0.85, border_color = "white", normal_color = "black",
  normal_lty = 1, normal_lwd = 2, xline = NULL, yline = NULL,
  ref_color = "gray40", ref_lty = 2, ref_lwd = 1, theme = "journal",
  size = 11, combine = FALSE, ncol = NULL, file = NULL, width = 7, height = 5,
  dpi = 300, show = TRUE, bg = "white", vline = NULL, hline = NULL
)
```

Arguments

<code>data</code>	A data frame. It may be omitted when an active R4VN data frame exists.
<code>x</code>	Categorical variable on the horizontal axis. Supply an unquoted name or a character name.
<code>vars</code>	Optional <code>vars(...)</code> selector used to request several variables in one call.
<code>by</code>	Optional grouping variable or hierarchical <code>vars(...)</code> specification.
<code>bins</code>	Number of bins, a vector of break points, or a valid value for <code>graphics::hist()</code> .
<code>density</code>	If TRUE, the vertical axis shows density rather than counts.
<code>normal</code>	Add a fitted normal density curve.
<code>xlab</code>	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
<code>ylab</code>	Optional y-axis tick labels.
<code>xtitle, ytitle</code>	Axis titles. Variable names or variable labels are used automatically when possible.
<code>xbreaks</code>	Numeric x-axis tick positions used with <code>xlab</code> .
<code>ybreaks</code>	Numeric positions used with <code>ylab</code> .

title, subtitle, note	Main title, subtitle, and note.
color	A color name or vector of colors. When NULL, palette is used.
palette	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.
alpha	Color opacity from 0 to 1.
border_color	Histogram-bar border colour.
normal_color, normal_lty, normal_lwd	Colour, line type, and width of the optional normal-reference curve.
xline, yline	Optional numeric reference lines on the x and y axes. Thus xline draws vertical lines and yline draws horizontal lines.
ref_color, ref_lty, ref_lwd	Color, line type, and width for reference lines; vectors are recycled for multiple lines.
theme	Graph theme: "journal", "clean", "minimal", or "classic".
size	Base font size.
combine	When several graphs are produced, draw them as labelled panels in one figure.
ncol	Optional number of columns in a combined figure.
file	Optional output file ending in png, jpg, tiff, pdf, or svg.
width, height	Output width and height in inches.
dpi	Resolution for raster output.
show	Draw the graph in the current graphics device.
bg	Background color for exported files.
vline, hline	Deprecated aliases for xline and yline.

Value

An object of class `r4vn_graph`; its data component contains histogram breaks, counts, density, and midpoints.

Examples

```
d <- data.frame(age = c(18, 21, 25, 27, 31, 34, 38, 45, 51, 62))
ghist(d, x = age)
ghist(d, x = age, bins = 5, normal = TRUE, color = "steelblue")

d$sex <- rep(c("Female", "Male"), length.out = nrow(d))
ghist(d, x = age, by = sex, normal = TRUE, xline = 40, ref_lty = 2)
ghist(d, vars = vars(age), by = vars(sex), normal = TRUE, combine = TRUE)

# Extended usage examples

d <- data.frame(age = c(18, 21, 25, 27, 31, 34, 38, 45, 51, 62))
ghist(d, x = age, show = FALSE)
```

```
ghist(d, x = age, bins = 5, show = FALSE)
ghist(d, x = age, density = TRUE, normal = TRUE, show = FALSE)
```

gline

Line chart

Description

Draws an ordered trend for one numeric outcome, optionally with separate lines by group.

Usage

```
gline(
  data = NULL, x, y = NULL, vars = NULL, by = NULL,
  stat = c("identity", "mean", "median"), points = TRUE, line_width = 2,
  line_type = 1, sort = TRUE, pch = 16, point_size = 0.9, xlab = NULL,
  ylab = NULL, xtitle = NULL, ytitle = NULL, xbreaks = NULL, ybreaks = NULL,
  title = NULL, subtitle = NULL, note = NULL, color = NULL,
  palette = "default", alpha = 1, legend = TRUE, legend_position = "topright",
  xline = NULL, yline = NULL, ref_color = "gray40", ref_lty = 2, ref_lwd = 1,
  theme = "journal", size = 11, combine = FALSE, ncol = NULL, file = NULL,
  width = 7, height = 5, dpi = 300, show = TRUE, bg = "white", vline = NULL,
  hline = NULL
)
```

Arguments

data	A data frame. It may be omitted when an active R4VN data frame exists.
x	Categorical variable on the horizontal axis. Supply an unquoted name or a character name.
y	Optional numeric variable. When omitted, bars show counts or percentages. When supplied, bars show a mean or median.
vars	Optional vars(...) selector used to request several variables in one call.
by	Optional grouping variable or hierarchical vars(...) specification.
stat	"identity", "mean", or "median". Mean or median collapses repeated x values within each group.
points	Show points along each line.
line_width, line_type	Width and line type of connecting lines.
sort	Sort observations by x within each line.
pch	Point symbol.
point_size	Point-size multiplier.

xlab	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
ylab	Optional y-axis tick labels.
xtitle, ytitle	Axis titles. Variable names or variable labels are used automatically when possible.
xbreaks, ybreaks	Numeric tick positions used with xlab and ylab.
title, subtitle, note	Main title, subtitle, and note.
color	A color name or vector of colors. When NULL, palette is used.
palette	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.
alpha	Color opacity from 0 to 1.
legend	Show the legend when by is supplied.
legend_position	Base-R legend position, for example "topright" or "topleft".
xline, yline	Optional numeric reference lines on the x and y axes. Thus xline draws vertical lines and yline draws horizontal lines.
ref_color, ref_lty, ref_lwd	Color, line type, and width for reference lines; vectors are recycled for multiple lines.
theme	Graph theme: "journal", "clean", "minimal", or "classic".
size	Base font size.
combine	When several graphs are produced, draw them as labelled panels in one figure.
ncol	Optional number of columns in a combined figure.
file	Optional output file ending in png, jpg, tiff, pdf, or svg.
width, height	Output width and height in inches.
dpi	Resolution for raster output.
show	Draw the graph in the current graphics device.
bg	Background color for exported files.
vline, hline	Deprecated aliases for xline and yline.

Details

Draws one x variable against one or several y variables with optional hierarchical grouping.

Value

An object of class `r4vn_graph`; its data component contains the plotted or aggregated values.

Examples

```
d <- data.frame(
  year = rep(2022:2024, 2),
  rate = c(12, 15, 18, 10, 13, 17),
  sex = rep(c("Male", "Female"), each = 3)
)
gline(d, x = year, y = rate)
gline(d, x = year, y = rate, by = sex, points = TRUE)

# Extended usage examples

d <- data.frame(year = rep(2022:2024, 2),
               rate = c(12, 15, 18, 10, 13, 17),
               sex = rep(c("Male", "Female"), each = 3))
gline(d, x = year, y = rate, show = FALSE)
gline(d, x = year, y = rate, by = sex, points = TRUE, show = FALSE)

# Collapse repeated x values to means or medians
d2 <- rbind(d, transform(d, rate = rate + 2))
gline(d2, x = year, y = rate, by = sex, stat = "mean", show = FALSE)
```

gpie

Pie or donut chart

Description

Draws the distribution of a categorical variable as a pie or donut chart. Bar charts are usually preferable when there are many categories.

Usage

```
gpie(
  data = NULL, x = NULL, vars = NULL, by = NULL, donut = FALSE, label = TRUE,
  percent = TRUE, digits = 1, xlab = NULL, title = NULL, subtitle = NULL,
  note = NULL, color = NULL, palette = "default", alpha = 1,
  border_color = "white", border_lwd = 1, clockwise = TRUE, missing = FALSE,
  missing_label = "Missing", theme = "journal", size = 11, combine = FALSE,
  ncol = NULL, file = NULL, width = 7, height = 5, dpi = 300, show = TRUE,
  bg = "white"
)
```

Arguments

data	A data frame. It may be omitted when an active R4VN data frame exists.
x	Categorical variable on the horizontal axis. Supply an unquoted name or a character name.
vars	Optional vars(...) selector used to request several variables in one call.

<code>by</code>	Optional grouping variable or hierarchical <code>vars(...)</code> specification.
<code>donut</code>	Draw a donut chart instead of a conventional pie.
<code>label</code>	Show category labels.
<code>percent</code>	Add percentages to labels.
<code>digits</code>	Number of percentage decimal places.
<code>xlab</code>	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
<code>title, subtitle, note</code>	Main title, subtitle, and note.
<code>color</code>	A color name or vector of colors. When <code>NULL</code> , <code>palette</code> is used.
<code>palette</code>	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.
<code>alpha</code>	Color opacity from 0 to 1.
<code>border_color, border_lwd</code>	Slice-border colour and line width.
<code>clockwise</code>	Draw slices clockwise.
<code>missing</code>	Include missing <code>x</code> or <code>by</code> values as a category.
<code>missing_label</code>	Label used for missing values.
<code>theme</code>	Graph theme: "journal", "clean", "minimal", or "classic".
<code>size</code>	Base font size.
<code>combine</code>	When several graphs are produced, draw them as labelled panels in one figure.
<code>ncol</code>	Optional number of columns in a combined figure.
<code>file</code>	Optional output file ending in <code>png</code> , <code>jpg</code> , <code>tiff</code> , <code>pdf</code> , or <code>svg</code> .
<code>width, height</code>	Output width and height in inches.
<code>dpi</code>	Resolution for raster output.
<code>show</code>	Draw the graph in the current graphics device.
<code>bg</code>	Background color for exported files.

Details

Draws one or several categorical variables; `by` creates separate charts for group/stratum combinations.

Value

An object of class `r4vn_graph`; its data component contains counts and percentages.

Examples

```
d <- data.frame(group = c("A", "A", "B", "B", "B", "C"))
gpie(d, x = group)
gpie(d, x = group, donut = TRUE, palette = "journal")

# Extended usage examples

d <- data.frame(group = c("A", "A", "B", "B", "B", "C"))
gpie(d, x = group, show = FALSE)
gpie(d, x = group, donut = TRUE, palette = "journal", show = FALSE)
gpie(d, x = group, label = TRUE, percent = FALSE, show = FALSE)
```

groc

ROC curve

Description

Calculates and draws a receiver operating characteristic curve from a binary outcome and numeric predicted probabilities or scores. No external package is required.

Usage

```
groc(
  data = NULL, outcome, pred = NULL, vars = NULL, by = NULL, event = NULL,
  diagonal = TRUE, auc = TRUE, digits = 3, curve_lty = 1, curve_lwd = 2.5,
  diagonal_color = "gray60", diagonal_lty = 2, diagonal_lwd = 1, xlab = NULL,
  ylab = NULL, xtitle = "1 - Specificity", ytitle = "Sensitivity",
  title = NULL, subtitle = NULL, note = NULL, color = NULL,
  palette = "journal", xline = NULL, yline = NULL, ref_color = "gray40",
  ref_lty = 2, ref_lwd = 1, theme = "journal", size = 11, combine = FALSE,
  ncol = NULL, file = NULL, width = 6, height = 6, dpi = 300, show = TRUE,
  bg = "white", vline = NULL, hline = NULL
)
```

Arguments

data	A data frame.
outcome	Binary outcome variable.
pred	Numeric predicted probability or score; larger values must indicate a greater probability of the event.
vars	Optional <code>vars(...)</code> selector for several numeric predictors or scores.
by	Optional grouping variable or hierarchical <code>vars(...)</code> selector.
event	Event value. By default, the second factor level or the larger numeric value is used.
diagonal	Show the no-discrimination diagonal.

auc Show the area under the curve.
 digits Number of AUC digits.
 curve_lty, curve_lwd ROC-curve line type and width.
 diagonal_color, diagonal_lty, diagonal_lwd Colour, line type, and width of the no-discrimination diagonal.
 xlab, ylab Optional tick labels.
 xtitle, ytitle Axis titles.
 title, subtitle, note, color, palette, theme, size, combine, ncol, file, width, height, dpi, show, bg See gbar().
 xline, yline Optional numeric reference lines on the x and y axes.
 ref_color, ref_lty, ref_lwd Color, line type, and width for reference lines.
 vline, hline Deprecated aliases for xline and yline.

Details

Draws ROC curves for one or several predictor/score variables. by can create ROC analyses within hierarchical strata.

Value

An object of class `r4vn_graph`; its data component contains thresholds, sensitivity, and specificity, and its `auc` attribute contains the AUC.

Examples

```

d <- data.frame(y = c(0, 0, 1, 1, 1), p = c(.10, .35, .40, .75, .90))
groc(d, outcome = y, pred = p, event = 1)

# Extended usage examples

d <- data.frame(
  outcome = factor(c("No", "No", "No", "Yes", "Yes", "Yes", "Yes")),
  probability = c(.05, .20, .35, .40, .65, .80, .95)
)

# ROC curve with explicit event
roc1 <- groc(d, outcome, probability, event = "Yes", show = FALSE)
attr(roc1$data, "auc")

# Hide the diagonal or AUC annotation and customize labels
groc(d, outcome, probability, event = "Yes",
     diagonal = FALSE, auc = FALSE,
     xtitle = "False-positive rate", ytitle = "True-positive rate",
     show = FALSE)

# Numeric binary outcome; the larger value is the default event

```

```
d2 <- data.frame(y = c(0, 0, 1, 1, 1), score = c(.10, .35, .40, .75, .90))
groc(d2, y, score, show = FALSE)

# Export the ROC curve
groc(d, outcome, probability, event = "Yes",
     file = tempfile(fileext = ".pdf"), show = FALSE)
```

gscatter

*Scatter plot***Description**

Draws the relationship between two numeric variables, optionally colored by a group and with fitted lines.

Usage

```
gscatter(
  data = NULL, x, y = NULL, vars = NULL, by = NULL, fit = FALSE,
  fit_color = NULL, fit_lty = 1, fit_lwd = 2, cor = FALSE, pch = 16,
  point_size = 1, xlab = NULL, ylab = NULL, xtitle = NULL, ytitle = NULL,
  xbreaks = NULL, ybreaks = NULL, title = NULL, subtitle = NULL, note = NULL,
  color = NULL, palette = "default", alpha = 0.75, legend = TRUE,
  legend_position = "topright", xline = NULL, yline = NULL,
  ref_color = "gray40", ref_lty = 2, ref_lwd = 1, theme = "journal",
  size = 11, combine = FALSE, ncol = NULL, file = NULL, width = 7, height = 5,
  dpi = 300, show = TRUE, bg = "white", vline = NULL, hline = NULL
)
```

Arguments

<code>data</code>	A data frame. It may be omitted when an active R4VN data frame exists.
<code>x</code>	Categorical variable on the horizontal axis. Supply an unquoted name or a character name.
<code>y</code>	Optional numeric variable. When omitted, bars show counts or percentages. When supplied, bars show a mean or median.
<code>vars</code>	Optional <code>vars(...)</code> selector used to request several variables in one call.
<code>by</code>	Optional grouping variable or hierarchical <code>vars(...)</code> specification.
<code>fit</code>	FALSE, TRUE, "linear", or "loess".
<code>fit_color, fit_lty, fit_lwd</code>	Colour, line type, and line width for the fitted line. <code>fit_color = NULL</code> follows the point/group colour.
<code>cor</code>	Add the Pearson correlation coefficient to the graph.
<code>pch</code>	Point symbol.

<code>point_size</code>	Point-size multiplier.
<code>xlab</code>	Optional category labels. Use an unnamed vector in displayed order or a named vector such as <code>c(M = "Male", F = "Female")</code> .
<code>ylab</code>	Optional y-axis tick labels.
<code>xtitle, ytitle</code>	Axis titles. Variable names or variable labels are used automatically when possible.
<code>xbreaks, ybreaks</code>	Numeric tick positions used with <code>xlab</code> and <code>ylab</code> .
<code>title, subtitle, note</code>	Main title, subtitle, and note.
<code>color</code>	A color name or vector of colors. When <code>NULL</code> , <code>palette</code> is used.
<code>palette</code>	One of "default", "journal", "blue", "green", "warm", "gray", or "colourblind"; alternatively, a color vector.
<code>alpha</code>	Color opacity from 0 to 1.
<code>legend</code>	Show the legend when <code>by</code> is supplied.
<code>legend_position</code>	Base-R legend position, for example "topright" or "topleft".
<code>xline, yline</code>	Optional numeric reference lines on the x and y axes. Thus <code>xline</code> draws vertical lines and <code>ylines</code> draws horizontal lines.
<code>ref_color, ref_lty, ref_lwd</code>	Color, line type, and width for reference lines; vectors are recycled for multiple lines.
<code>theme</code>	Graph theme: "journal", "clean", "minimal", or "classic".
<code>size</code>	Base font size.
<code>combine</code>	When several graphs are produced, draw them as labelled panels in one figure.
<code>ncol</code>	Optional number of columns in a combined figure.
<code>file</code>	Optional output file ending in png, jpg, tiff, pdf, or svg.
<code>width, height</code>	Output width and height in inches.
<code>dpi</code>	Resolution for raster output.
<code>show</code>	Draw the graph in the current graphics device.
<code>bg</code>	Background color for exported files.
<code>vline, hline</code>	Deprecated aliases for <code>xline</code> and <code>ylines</code> .

Details

Draws one x variable against one or several y variables. Hierarchical `by` uses all but the last variable as strata and the last variable as the plotted grouping variable.

Value

An object of class `r4vn_graph`; its data component contains complete plotted observations.

Examples

```
d <- data.frame(
  age = c(22, 28, 35, 41, 55),
  bmi = c(20, 23, 25, 27, 29),
  sex = c("M", "F", "F", "M", "F")
)
gscatter(d, x = age, y = bmi)
gscatter(d, x = age, y = bmi, by = sex, fit = "linear", cor = TRUE)

# Extended usage examples

d <- data.frame(age = c(22, 28, 35, 41, 55),
               bmi = c(20, 23, 25, 27, 29),
               sex = c("M", "F", "F", "M", "F"))
gscatter(d, x = age, y = bmi, show = FALSE)
gscatter(d, x = age, y = bmi, fit = "linear", cor = TRUE, show = FALSE)
gscatter(d, x = age, y = bmi, by = sex, fit = "linear", show = FALSE)
gscatter(d, x = age, y = bmi, fit = "loess", show = FALSE)
```

gsurv

Survival and Cumulative Incidence Curves

Description

Draws Kaplan-Meier survival, cumulative risk, or competing-risk cumulative incidence curves from a `tabsurv()` result. Uses the existing R4VN base-graphics engine and can add confidence intervals, censor marks, log-rank p-values, median lines, and number-at-risk tables. Examples below are self-contained.

Usage

```
gsurv(
  x,
  type = NULL,
  xlab = NULL,
  ylab = NULL,
  xtitle = NULL,
  ytitle = NULL,
  xlim = NULL,
  ylim = NULL,
  breaks = NULL,
  percent = TRUE,
  ci = FALSE,
  censor = TRUE,
  color = NULL,
  palette = "default",
  linetype = NULL,
```

```

    line_width = 1.5,
    ci_color = NULL,
    ci_alpha = 0.45,
    ci_linetype = 3,
    ci_line_width = NULL,
    censor_color = NULL,
    censor_pch = 3,
    censor_size = 0.7,
    median_color = "gray50",
    median_lty = 2,
    median_lwd = 1,
    ref_color = "gray55",
    ref_lty = 3,
    ref_lwd = 1,
    size = 11,
    labels = NULL,
    legend = TRUE,
    legend_position = "topright",
    pvalue = FALSE,
    median = FALSE,
    risk_table = FALSE,
    risk_at = NULL,
    xline = NULL,
    yline = NULL,
    title = NULL,
    subtitle = NULL,
    note = NULL,
    theme = "journal",
    file = NULL,
    width = 7,
    height = NULL,
    dpi = 300,
    show = TRUE,
    bg = "white",
    vline = NULL,
    hline = NULL
  )

```

Arguments

<code>x</code>	An object returned by <code>tabsurv()</code> .
<code>type</code>	<code>NULL</code> , "survival", "risk", or "cif". <code>NULL</code> chooses automatically.
<code>xlab, ylab</code>	Axis titles retained for convenience.
<code>xtitle, ytitle</code>	R4VN-style aliases for the x- and y-axis titles; when supplied they override <code>xlab/ylab</code> .
<code>xlim, ylim</code>	Optional axis limits.
<code>breaks</code>	Numeric spacing between x ticks, or explicit x tick positions.

percent	Show y values as percentages.
ci	Show confidence limits.
censor	Show censor marks for ordinary Kaplan-Meier curves.
color, palette	Curve colors; compatible with other R4VN graphs.
linetype	Line types, recycled across groups.
line_width	Curve line width.
ci_color, ci_alpha, ci_linetype, ci_line_width	Confidence-limit colour, transparency, line type, and line width. ci_color = NULL follows each curve colour.
censor_color, censor_pch, censor_size	Censor-mark colour, point symbol, and size. censor_color = NULL follows each curve colour.
median_color, median_lty, median_lwd	Colour, line type, and width for median-survival reference lines.
ref_color, ref_lty, ref_lwd	Colour, line type, and width for user-specified xline/yline reference lines.
size	Base text size, consistent with other R4VN graph functions.
labels	Optional replacement labels for curve groups.
legend	TRUE/FALSE or a character legend title.
legend_position	Base graphics legend position.
pvalue	Annotate the log-rank p-value when available.
median	Draw median-survival reference lines.
risk_table	Add number-at-risk table below the plot.
risk_at	Time points for the risk table. Defaults to plot ticks or tabsurv(at=).
xline, yline	Optional reference lines at x- and y-axis values.
title, subtitle, note	Plot annotations.
theme, file, width, height, dpi, show, bg	Graph-output controls.
vline, hline	Deprecated aliases for xline and yline.

Value

An r4vn_graph object.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  d <- data.frame(
    time = c(5, 8, 10, 12, 15, 18, 20, 22),
    event = c(1, 0, 1, 1, 0, 1, 0, 1),
    group = factor(rep(c("A", "B"), 4))
  )
}
```

```

s <- tabsurv(time, event, by = group, data = d, km = TRUE, show = FALSE)
gsurv(s, show = FALSE)
gsurv(s, type = "risk", ci = TRUE, pvalue = TRUE, show = FALSE)
gsurv(s, risk_table = TRUE, risk_at = c(0, 10, 20), show = FALSE)
}

```

ir

Incidence-rate Comparison

Description

Compares incidence rates from typed cases and person-time or from variables.

Usage

```

ir(
  cases,
  exposure,
  time,
  data = NULL,
  exposed = NULL,
  digits = 4,
  p_digits = 3,
  level = 0.95,
  show = TRUE,
  console = FALSE
)

```

```

iri(
  cases.exposed,
  cases.unexposed,
  time.exposed,
  time.unexposed,
  level = 0.95,
  digits = 4,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

```

Arguments

cases	Nonnegative case-count variable.
exposure	Binary exposure variable.
time	Nonnegative person-time variable.
data	Data frame. If NULL, active data is used.

exposed	Exposure level; defaults to the last observed level.
digits, p_digits	Decimal places for estimates and p-values.
level	Confidence level.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
cases.exposed, cases.unexposed	Number of cases in exposed and unexposed groups.
time.exposed, time.unexposed	Person-time in exposed and unexposed groups.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
iri(41, 15, 28010, 19017)
```

keepvar	<i>Keep variables and/or observations</i>
---------	---

Description

Keeps variables or observations in an explicit data frame or active data.

Usage

```
keepvar(..., data = NULL, obs = NULL)
```

Arguments

...	Variable selectors. For backward compatibility, an explicit data frame may be the first unnamed argument.
data	Optional explicit data frame object.
obs	Optional observations to keep.

Value

The edited data frame invisibly.

Examples

```

d <- data.frame(
  id = 1:5,
  age = c(10, 20, NA, 40, 50),
  sex = c("M", "F", "F", "M", "F"),
  score_a = 1:5,
  score_b = 6:10
)

# Each explicit-data example uses its own copy because keepvar()
# intentionally edits the supplied object.
d1 <- d
keepvar(d1, id, age, sex)

d2 <- d
keepvar(d2, id:sex)

d3 <- d
keepvar(d3, id, "score_*")

d4 <- d
keepvar(d4, obs = age >= 18)

d5 <- d
keepvar(d5, id, age, obs = !missing(age))

d6 <- d
keepvar(d6, obs = c(1, 3, 5))

# Active-data syntax
active_d <- d
usedf(active_d, quiet = TRUE)
keepvar(id, age, obs = age >= 18)
usedf(clear = TRUE, quiet = TRUE)

```

kwallis

Kruskal-Wallis Test

Description

Performs the Kruskal-Wallis rank-sum test for a numeric outcome across two or more independent groups. Optional Dunn or pairwise Wilcoxon post-hoc tests and rank-based effect sizes can be requested. With `by = vars(region, sex, treatment)`, region and sex are nested strata and treatment is the innermost Kruskal-Wallis factor.

Usage

```

kwallis(
  x, by, data = NULL, posthoc = c("none", "dunn", "wilcoxon"),

```

```

    adjust = "holm", effect = FALSE, digits = 3, p_digits = 3, show = TRUE,
    console = FALSE
  )

```

Arguments

x	Numeric outcome.
by	Grouping variable with at least two observed groups, or hierarchical vars(...) specification.
data	Data frame. If NULL, active R4VN data is used.
posthoc	Post-hoc method: "none", "dunn", or "wilcoxon".
adjust	Multiplicity adjustment accepted by <code>stats::p.adjust()</code> .
effect	Logical; add epsilon-squared and eta-squared(H).
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional result in the Console. Default FALSE.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```

d <- data.frame(
  score = c(10, 12, 11, 18, 17, 20, 25, 24, 27),
  group = factor(rep(c("A", "B", "C"), each = 3))
)
kwallis(score, by = group, data = d, posthoc = "dunn", effect = TRUE)

# Extended usage examples

d <- data.frame(
  score = c(10, 12, 11, 18, 17, 20, 25, 24, 27),
  treatment = factor(rep(c("A", "B", "C"), each = 3))
)
kwallis(score, by = treatment, data = d)
usedf(d)
result <- kwallis(score, by = treatment, show = FALSE)
result$raw$test

```

labvar

*Label and recode existing variables***Description**

Adds variable labels, value labels, recodes values, and sets reference categories. The function can edit an explicit data frame or the active R4VN data frame.

Usage

```
labvar(
  ...,
  data = NULL,
  label = NULL,
  values = NULL,
  recode = NULL,
  ref = NULL,
  ordered = FALSE
)
```

Arguments

<code>...</code>	Variables to process. For backward compatibility, an explicit data frame may be supplied as the first unnamed argument.
<code>data</code>	Optional explicit data frame object. When omitted, active data is used.
<code>label</code>	A character label or one label per selected variable.
<code>values</code>	A named value-label vector such as <code>c("0" = "No", "1" = "Yes")</code> .
<code>recode</code>	A named recode vector such as <code>c("min:12" = 1, "13:17" = 2, "18:max" = 3)</code> .
<code>ref</code>	Optional reference category.
<code>ordered</code>	Logical; create an ordered factor.

Details

Explicit-data syntax remains valid:

```
labvar(data, sex, label = "Sex", values = c("1" = "Male", "2" = "Female"))
```

After `usedf(data)`, active-data syntax is:

```
labvar(sex, label = "Sex", values = c("1" = "Male", "2" = "Female"))
```

When active data were selected with `usedf(patient)`, edits update both the active data and the linked patient object. For an unlinked active copy, retrieve the result with `data <- usedf()`.

Value

The edited data frame invisibly.

Examples

```
d <- data.frame(sex = c(1, 2, 1), age = c(8, 15, 30))
usedf(d, quiet = TRUE)
labvar(sex, label = "Sex",
       values = c("1" = "Male", "2" = "Female"))
labvar(age,
       recode = c("min:12" = 1, "13:17" = 2, "18:max" = 3),
       label = "Age group",
       values = c("1" = "0-12", "2" = "13-17", "3" = "18+"))

# Extended usage examples

# -----
# 1. Add only a variable label; numeric values remain numeric
d1 <- data.frame(age = c(18, 25, 40))
labvar(d1, age, label = "Age in years")
attr(d1$age, "label")

# 2. Add value labels; the variable becomes a factor
d2 <- data.frame(sex = c(1, 2, 2, 1))
labvar(d2, sex, label = "Sex",
       values = c("1" = "Male", "2" = "Female"))
levels(d2$sex)

# 3. Set the reference category by stored code
d3 <- data.frame(smoke = c(0, 1, 1, 0))
labvar(d3, smoke, label = "Current smoking",
       values = c("0" = "No", "1" = "Yes"), ref = 0)
levels(d3$smoke)

# 4. Set the reference category by displayed label
d4 <- data.frame(treatment = c(1, 2, 3, 1))
labvar(d4, treatment,
       values = c("1" = "Standard", "2" = "Drug A", "3" = "Drug B"),
       ref = "Standard")

# 5. Create an ordered factor
d5 <- data.frame(severity = c(1, 3, 2, 1))
labvar(d5, severity, label = "Disease severity",
       values = c("1" = "Mild", "2" = "Moderate", "3" = "Severe"),
       ordered = TRUE)
is.ordered(d5$severity)

# 6. Recode inclusive numeric ranges and then label the new categories
d6 <- data.frame(age = c(8, 12, 13, 17, 18, 65))
labvar(d6, age,
       recode = c("min:12" = 1, "13:17" = 2, "18:max" = 3),
       label = "Age group",
       values = c("1" = "0-12", "2" = "13-17", "3" = "18+"))

# 7. Collapse several exact values into one category
d7 <- data.frame(answer = c(1, 2, 3, 2, 1))
```

```

labvar(d7, answer,
       recode = c("1" = 1, "2 3" = 0),
       values = c("0" = "No/uncertain", "1" = "Yes"))

# 8. Recode without value labels; the result remains numeric
d8 <- data.frame(score = c(2, 6, 9, 15))
labvar(d8, score,
       recode = c("min:4" = 1, "5:9" = 2, "10:max" = 3),
       label = "Score category code")
is.numeric(d8$score)

# 9. Apply common value labels to several binary variables
d9 <- data.frame(smoke = c(0, 1), alcohol = c(1, 0), exercise = c(1, 1))
labvar(d9, smoke, alcohol, exercise,
       label = c("Smoking", "Alcohol use", "Regular exercise"),
       values = c("0" = "No", "1" = "Yes"))

# 10. Supply labels as a named vector
d10 <- data.frame(sbp = c(120, 130), dbp = c(75, 85))
labvar(d10, sbp, dbp,
       label = c(sbp = "Systolic blood pressure",
                 dbp = "Diastolic blood pressure"))

# 11. Select a contiguous range of variables
d11 <- data.frame(q1 = c(0, 1), q2 = c(1, 0), q3 = c(1, 1), age = c(20, 30))
labvar(d11, q1:q3, values = c("0" = "No", "1" = "Yes"))

# 12. Select variables with a wildcard
d12 <- data.frame(symptom_a = c(0, 1), symptom_b = c(1, 1), age = c(20, 30))
labvar(d12, "symptom_*", values = c("0" = "Absent", "1" = "Present"))

# 13. Use explicit-data syntax
d13 <- data.frame(outcome = c(0, 1, 0))
labvar(d13, outcome, label = "Outcome",
       values = c("0" = "No", "1" = "Yes"), ref = "No")

# 14. Use active-data syntax
d14 <- data.frame(outcome = c(0, 1, 0))
usedf(d14, quiet = TRUE)
labvar(outcome, label = "Outcome",
       values = c("0" = "No", "1" = "Yes"), ref = "No")
d14_active <- usedf(quiet = TRUE)

# 15. Use separate calls when variables need different value-label systems
d15 <- data.frame(sex = c(1, 2), outcome = c(0, 1))
labvar(d15, sex, values = c("1" = "Male", "2" = "Female"))
labvar(d15, outcome, values = c("0" = "No", "1" = "Yes"))

```

Description

Calculates estimates, standard errors, confidence intervals and Wald tests for linear combinations of coefficients from the active or supplied model.

Usage

```
lincom(...,
  model = NULL,
  rhs = 0,
  exp = FALSE,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE)
```

Arguments

<code>...</code>	One or more linear-combination expressions using coefficient names, or character expressions.
<code>model</code>	Optional fitted model/R4VN result; active model is used when omitted.
<code>rhs</code>	Null value for the linear combination.
<code>exp</code>	Exponentiate estimate and confidence interval.
<code>level</code>	Confidence level.
<code>digits, p_digits</code>	Formatting digits.
<code>show, console</code>	R4VN display controls.

Value

An R4VN result object, invisibly.

See Also

[margins](#), [predict](#)

Examples

```
d <- data.frame(
  y = c(50, 54, 57, 61, 65, 68, 72, 76),
  age = c(20, 25, 30, 35, 40, 45, 50, 55),
  bmi = c(20, 22, 21, 24, 25, 27, 26, 29)
)
regress(y, c.age, c.bmi, data = d, show = FALSE)
lincom(age + 2 * bmi, show = FALSE)
lincom("age - bmi", show = FALSE)
```

logistic

*Binary logistic regression***Description**

Fits binary logistic regression using formula syntax or compact R4VN syntax. Compact syntax avoids the need to type ~ and +.

Usage

```
logistic(
  y,
  ...,
  vars = NULL,
  data = NULL,
  event = NULL,
  or = FALSE,
  exp = FALSE,
  noconstant = FALSE,
  vce = c("model", "robust", "cluster"),
  cluster = NULL,
  weights = NULL,
  subset = NULL,
  ref = NULL,
  gof = FALSE,
  groups = 10,
  classification = FALSE,
  cutoff = 0.5,
  vif = FALSE,
  diagnosis = FALSE,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)
```

Arguments

y	Formula or binary outcome variable.
...	Predictors or model terms when y is not a formula.
vars	Optional model terms written as vars(...). The expression is captured without evaluating the public table-oriented vars() parser, so interactions are allowed.
data	Data frame or NULL for active data.
event	Event level for a simple named outcome.

<code>or</code>	Add an odds-ratio table while retaining coefficients.
<code>exp</code>	Display odds ratios only.
<code>noconstant</code>	Fit without an intercept.
<code>vce</code>	Model-based, HC1 robust, or cluster-robust covariance.
<code>cluster</code>	Cluster variable.
<code>weights</code>	Optional non-negative weights.
<code>subset</code>	Optional logical subset.
<code>ref</code>	Optional named list of factor reference levels.
<code>gof</code>	Show a Hosmer-Lemeshow test.
<code>groups</code>	Number of groups for the Hosmer-Lemeshow test.
<code>classification</code>	Show a classification table.
<code>cutoff</code>	Classification cutoff.
<code>vif</code>	Show coefficient-level VIFs.
<code>diagnosis</code>	Logical; if TRUE, append calibration/goodness-of-fit, discrimination, residual, influence, and collinearity diagnostics appropriate for binary logistic regression. Default FALSE.
<code>level</code>	Confidence level.
<code>digits, p_digits</code>	Decimal places.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.

Details

Compact model syntax:

- `x`: use the variable as stored in the data.
- `c.x`: force `x` to be continuous.
- `i.x`: force `x` to be categorical.
- `b2.x, b3.x, ...`: categorical with the corresponding factor-level position as reference.
- `ib0.x, ib1.x, ib2.x, ...`: categorical with the requested value/level as reference. If that literal level is unavailable, a positive integer can fall back to the corresponding factor-level position.
- `i.a*i.b`: main effects for `a` and `b` plus their interaction.
- `i.a:i.b`: interaction only.
- `c.x*i.a`: continuous and categorical main effects plus interaction.

Thus `logistic(y, ib2.occupation*i.treatment, c.age)` fits occupation, treatment, occupation-by-treatment interaction, and age without requiring formula operators `~` or `+`.

The fitted `glm` object is stored in `result$raw$model`, so nested models can be compared directly with `lrtest()`.

Value

An object of class `r4vn_stat`, returned invisibly. Its `sections` component contains the formatted model summary, coefficient and/or odds- ratio tables, and any requested goodness-of-fit, classification, or VIF tables. In `raw`, `model` is the fitted binomial `glm` object, `vcov` is the covariance matrix, `coefficients` contains coefficient-level estimates and tests, `logLik` and `null.logLik` are model log likelihoods, `pseudo.r2` is McFadden-style pseudo-R-squared, `event` records the modeled outcome level, and `vce` and `model.terms` record the covariance estimator and fitted terms.

See Also

[lrtest\(\)](#), [poisson\(\)](#)

Examples

```
set.seed(2026)
d <- data.frame(
  outcome = factor(rbinom(200, 1, .35), levels = 0:1,
    labels = c("No", "Yes")),
  age = rnorm(200, 45, 12),
  occupation = factor(sample(c("Office", "Worker", "Other"), 200, TRUE)),
  treatment = factor(sample(c("No", "Yes"), 200, TRUE))
)

m1 <- logistic(
  outcome,
  c.age,
  i.occupation,
  i.treatment,
  data = d,
  event = "Yes",
  show = FALSE
)

m2 <- logistic(
  outcome,
  c.age,
  ib2.occupation*i.treatment,
  data = d,
  event = "Yes",
  show = FALSE
)

m3 <- logistic(
  outcome,
  vars = vars(c.age, ib2.occupation*i.treatment),
  data = d,
  event = "Yes",
  show = FALSE
)

lrtest(m1, m2, show = FALSE)
```

```
# Request a complete diagnostic panel
logistic(outcome, c.age, i.occupation, data = d, event = "Yes",
         diagnosis = TRUE, show = FALSE)
```

lrtest

Likelihood-ratio test for nested regression models

Description

Compares two or more nested likelihood-based regression models. Objects returned by R4VN `logistic()` and `poisson()` can be supplied directly.

Usage

```
lrtest(..., digits = 3, p_digits = 3, show = TRUE, console = FALSE)
```

Arguments

<code>...</code>	Two or more nested fitted models, ordered from the smaller model to progressively larger models. R4VN statistical results containing a fitted model in <code>raw\$model</code> are accepted directly.
<code>digits</code>	Number of decimal places for likelihood and LR statistics.
<code>p_digits</code>	Number of decimal places for p-values.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the result in the Console. Default FALSE.

Details

When more than two models are supplied, comparisons are sequential: M1 versus M2, then M2 versus M3, and so on.

The models must use the same outcome, analytic observations, weights, offsets/exposure definition, and likelihood family/link, and each larger model must contain the smaller model.

Supported fits include ordinary likelihood-based `glm` models such as logistic and Poisson regression, `MASS::glm.nb()` negative-binomial models, `survival::coxph()` Cox models, and `survival::survreg()` parametric survival models.

Quasi-likelihood models are not supported. R4VN models fitted with `vce = "robust"` or `vce = "cluster"` are also rejected because the classical likelihood-ratio chi-square test is model-likelihood inference, not robust covariance inference.

For ordinary linear regression use the nested-model F test rather than `lrtest()`.

Value

An object of class `r4vn_stat`. The unformatted comparison table is stored in `result$raw$table`; the backward-compatible `result$raw$comparison` table is also retained.

See Also

[logistic\(\)](#), [poisson\(\)](#)

Examples

```
set.seed(2026)
d <- data.frame(
  y = factor(rbinom(250, 1, .35), levels = 0:1,
             labels = c("No", "Yes")),
  age = rnorm(250, 45, 12),
  sex = factor(sample(c("Female", "Male"), 250, TRUE)),
  treatment = factor(sample(c("No", "Yes"), 250, TRUE))
)

m1 <- logistic(y, c.age, i.sex, i.treatment,
               data = d, event = "Yes", show = FALSE)
m2 <- logistic(y, c.age, i.sex*i.treatment,
               data = d, event = "Yes", show = FALSE)

lrtest(m1, m2, show = FALSE)
```

margins

Predictive margins after an R4VN model

Description

Calculates average adjusted predictions and confidence intervals from the active or supplied model.

Usage

```
margins(model = NULL,
        at = NULL,
        over = NULL,
        type = c("response",
                 "link"),
        atmeans = FALSE,
        level = 0.95,
        digits = 3,
        p_digits = 3,
        show = TRUE,
        console = FALSE)
```

Arguments

<code>model</code>	Optional fitted model or R4VN model result. The most recently fitted R4VN model is used when omitted.
<code>at</code>	Predictor settings created by <code>at()</code> ; vectors are crossed into scenarios.

over	Optional grouping variable(s), including vars(a, b), for margins within observed groups.
type	Response-scale or link-scale predictions.
atmeans	If TRUE, unspecified covariates are fixed at representative means/modes instead of averaging individual predictions.
level	Confidence level.
digits, p_digits	Formatting digits.
show, console	R4VN display controls.

Details

Unless atmeans = TRUE, variables not specified in at() retain their observed values and predictions are averaged over the estimation sample. Standard errors use the fitted covariance matrix and a delta-method gradient when the model provides the information required.

Value

An R4VN result object, invisibly.

See Also

[at](#), [marginsplot](#), [predict](#), [lincom](#)

Examples

```
d <- data.frame(
  outcome = factor(c(0, 0, 0, 1, 0, 1, 1, 1), levels = 0:1,
    labels = c("No", "Yes")),
  age = c(20, 25, 30, 35, 40, 45, 50, 55),
  sex = factor(rep(c("Female", "Male"), 4))
)
m <- logistic(outcome, c.age, i.sex, data = d, event = "Yes", show = FALSE)
margins(m, at = at(age = 40), show = FALSE)
margins(m, over = sex, show = FALSE)
```

marginsplot

Plot predictive margins

Description

Plots estimates and confidence intervals from margins().

Usage

```
marginsplot(  
  result = NULL,  
  x = NULL,  
  by = NULL,  
  ci = TRUE,  
  line = TRUE,  
  points = TRUE,  
  line_width = 2,  
  line_type = 1,  
  point_size = 1,  
  point_pch = 16,  
  ci_color = NULL,  
  ci_alpha = 1,  
  ci_lwd = 1,  
  ci_lty = 1,  
  xline = NULL,  
  yline = NULL,  
  ref_color = "gray40",  
  ref_lty = 2,  
  ref_lwd = 1,  
  xlab = NULL,  
  ylab = NULL,  
  xtitle = NULL,  
  ytitle = NULL,  
  title = NULL,  
  subtitle = NULL,  
  note = NULL,  
  color = NULL,  
  palette = "journal",  
  alpha = 1,  
  legend = TRUE,  
  legend_position = "topright",  
  theme = "journal",  
  size = 11,  
  file = NULL,  
  width = 7,  
  height = 5,  
  dpi = 300,  
  show = TRUE,  
  bg = "white",  
  vline = NULL,  
  hline = NULL  
)
```

Arguments

<code>result</code>	A <code>margins()</code> result. The most recent margins result is used when omitted.
---------------------	---

x Scenario variable for the horizontal axis; selected automatically when omitted.
by Optional grouping variable from the margins table.
ci Draw confidence intervals.
line, points Draw connecting lines and points.
xline, yline Optional reference lines at x- and y-axis values.
vline, hline Deprecated aliases for xline and yline.
ref_color, ref_lty, ref_lwd Reference-line formatting.
xlab, ylab, xtitle, ytitle, title, subtitle, note, color, palette, alpha, legend, legend_position, theme, size, file, width, height, dpi, show, bg Standard R4VN graph controls.
line_width, line_type Connecting-line width and line type.
point_size, point_pch Point size and symbol.
ci_color, ci_alpha, ci_lwd, ci_lty Confidence-interval colour, transparency, width, and line type. `ci_color = NULL` follows each group colour.

Value

An R4VN result object, invisibly.

See Also

[margins](#)

Examples

```

d <- data.frame(
  outcome = factor(c(0, 0, 0, 1, 0, 1, 1, 1), levels = 0:1,
    labels = c("No", "Yes")),
  age = c(20, 25, 30, 35, 40, 45, 50, 55)
)
m <- logistic(outcome, c.age, data = d, event = "Yes", show = FALSE)
mg <- margins(m, at = at(age = seq(30, 50, 10)), show = FALSE)
marginsplot(mg, show = FALSE)

```

Description

Calculates the matched odds ratio from the discordant pairs and an exact conditional confidence interval and p-value.

Usage

```

mcc(
  case,
  control,
  data = NULL,
  exposed = NULL,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

mcci(
  a,
  b,
  c,
  d,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

```

Arguments

case, control	Paired binary exposure variables for cases and controls.
data	Data frame. If NULL, active data is used.
exposed	Exposure level; defaults to the last observed level.
level	Confidence level.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
a, b, c, d	Matched-pair table cells. The matched odds ratio is b/c.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
mcci(20, 14, 5, 31)
```

mergedata

*Merge data frames by key variables***Description**

Merges a using data frame or file into a master data frame. Active data is used as master by default. Cardinality can be checked like Stata merges.

Usage

```
mergedata(
  using,
  by,
  data = NULL,
  type = c("1:1", "1:m", "m:1", "m:m"),
  join = c("full", "left", "right", "inner"),
  suffix = c("_master", "_using"),
  generate = "_merge",
  keep = NULL,
  update = FALSE,
  replace = FALSE,
  active = TRUE,
  quiet = FALSE,
  labels = c("factor", "labelled", "numeric")
)
```

Arguments

using	Data frame or existing file path.
by	Key variables: bare name, <code>vars(...)</code> , or character vector.
data	Optional explicit master data frame.
type	One of "1:1", "1:m", "m:1", or "m:m".
join	One of "full", "left", "right", or "inner".
suffix	Two suffixes for overlapping non-key variables.
generate	Merge-status variable name, or NULL. Codes are 1 master only, 2 using only, and 3 matched.
keep	Optional status codes to retain, such as 3 or <code>c(1, 3)</code> .
update	Logical; fill missing master values from using variables with the same names.
replace	Logical; with <code>update = TRUE</code> , replace nonmissing master values by nonmissing using values.
active	Logical; replace active data with the result.
quiet	Logical; suppress messages.
labels	Label handling when using is a file path.

Value

The merged data frame invisibly.

Examples

```
master <- data.frame(id = 1:3, age = c(20, 30, 40))
using <- data.frame(id = 2:4, sex = c("F", "M", "F"))
usedf(master, quiet = TRUE)
mergedata(using, by = id, type = "1:1", quiet = TRUE)
result <- usedf(quiet = TRUE)
result2 <- mergedata(using, data = master, by = id, type = "1:1",
  active = FALSE, quiet = TRUE)
```

nlregress

Flexible nonlinear-shape regression

Description

Fits flexible nonlinear predictor shapes without requiring users to construct spline bases manually.

Usage

```
nlregress(y,
  x,
  covariates = NULL,
  data = NULL,
  spline = c("natural",
    "bspline",
    "polynomial",
    "linear"),
  df = 4,
  degree = 3,
  knots = NULL,
  boundary_knots = NULL,
  family = c("gaussian",
    "binomial",
    "poisson"),
  event = NULL,
  robust = FALSE,
  diagnosis = FALSE,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE)
```

Arguments

y	Outcome variable.
x	Numeric predictor whose functional form is modeled flexibly.
covariates	Optional additional covariates, including <code>vars(...)</code> .
data	Data frame or active data.
spline	Natural spline, B-spline, raw polynomial, or linear form.
df	Spline degrees of freedom when knots are not supplied.
degree	B-spline/polynomial degree.
knots	Optional internal knots.
boundary_knots	Optional two boundary knots.
family	Gaussian, binomial, or Poisson model.
event	Event category for binary binomial/Poisson outcomes.
robust	Request robust covariance when available.
diagnosis	Logical; if TRUE, append diagnostics appropriate to the selected family: linear-model diagnostics for Gaussian models, calibration/influence diagnostics for binomial models, and dispersion/goodness-of-fit diagnostics for Poisson models. Default FALSE.
level, digits, p_digits, show, console	Confidence, formatting and display controls.

Details

The fitted model is stored as the active model and can be used immediately by `margins()`, `predict()` and `lincom()`.

Value

An R4VN result object, invisibly.

See Also

[qregress](#), [margins](#)

Examples

```
d <- data.frame(
  age = seq(20, 75, by = 5),
  bmi = c(20, 21, 22, 24, 23, 25, 26, 27, 29, 28, 30, 31),
  sex = factor(rep(c("Female", "Male"), 6)),
  y = c(48, 52, 55, 61, 60, 66, 69, 73, 78, 80, 85, 89),
  outcome = c(0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1)
)
nlregress(y, age, data = d, spline = "natural", df = 3, show = FALSE)
nlregress(y, age, covariates = vars(sex, bmi), data = d,
  spline = "natural", knots = c(35, 50), show = FALSE)
nlregress(outcome, age, data = d, family = "binomial", event = 1,
```

```

        show = FALSE)
nlregress(y, age, data = d, spline = "bspline", df = 4, diagnosis = TRUE, show = FALSE)
nlregress(y, age, data = d, spline = "polynomial", degree = 2, show = FALSE)

```

normtest
Normality tests for one or more variables

Description

Runs normality/distribution tests consistently for one or many variables, overall or within hierarchical groups.

Usage

```

normtest(x = NULL,
  vars = NULL,
  by = NULL,
  data = NULL,
  method = c("all",
    "shapiro",
    "ks",
    "lilliefors",
    "anderson",
    "cramer.von.mises",
    "shapiro.francia",
    "pearson",
    "jarque.bera"),
  digits = 4,
  p_digits = 3,
  show = TRUE,
  console = FALSE)

```

Arguments

x	One numeric variable. May be omitted when vars is supplied.
vars	Optional vars(...) selection of several numeric variables.
by	Optional grouping specification. With by = vars(province, sex, group) , province and sex are ordered outer strata and group is the innermost group.
data	Data frame; the active R4VN data frame is used when omitted.
method	Test or tests to run. "all" runs built-in Shapiro-Wilk, fitted-normal KS and Jarque-Bera plus available optional tests from nortest .
digits, p_digits	Decimal places for statistics and p-values.
show, console	R4VN display controls.

Details

Different normality tests have different sensitivities and sample-size behavior. The ordinary fitted-normal Kolmogorov-Smirnov p-value is approximate because mean and SD are estimated from the same sample; use the Lilliefors option when available. A nonsignificant test does not prove normality, so graphical inspection remains important.

Value

An R4VN result object, invisibly.

See Also

[swilk](#), [varform](#)

Examples

```
d <- data.frame(
  x = rnorm(80),
  y = rexp(80),
  province = rep(c("A", "B"), each = 40),
  sex = rep(rep(c("F", "M"), each = 20), 2)
)
normtest(x, data = d)
normtest(vars = vars(x, y), data = d, method = c("shapiro", "jarque.bera"))
normtest(vars = vars(x, y), by = vars(province, sex), data = d)
```

nptrend

Nonparametric trend across ordered groups

Description

Tests for monotonic trend across ordered groups, including quantitative and binary outcomes and hierarchical strata.

Usage

```
nptrend(x,
  by,
  data = NULL,
  method = c("auto",
    "cuzick",
    "cochran-armitage",
    "spearman",
    "linear"),
  event = NULL,
  scores = NULL,
  digits = 3,
```

```
p_digits = 3,  
show = TRUE,  
console = FALSE)
```

Arguments

x	Outcome variable.
by	Ordered group. With vars(...), the final variable is the ordered group and preceding variables are strata.
data	Data frame or active data.
method	Automatic, Cuzick-style rank trend, Cochran-Armitage, Spearman, or linear trend.
event	Event value for binary trend analysis.
scores	Optional numeric scores for group levels.
digits, p_digits, show, console	Formatting/display controls.

Value

An R4VN result object, invisibly.

See Also

[kwallis](#)

Examples

```
d <- data.frame(y = c(2,3,4,4,5,7,8,9,10), dose = ordered(rep(1:3, each=3)))  
nptrend(y, by = dose, data = d)
```

opendata	<i>Open data from files, research platforms, online forms, and databases</i>
----------	--

Description

Reads local/remote files and connects to supported research data sources. Existing file and REDCap behavior is retained. Additional connectors support KoboToolbox, Google Forms, Google Sheets, Microsoft Forms/Excel, and MySQL/MariaDB.

Usage

```
opendata(  
  file = NULL,  
  api = NULL,  
  url = NULL,  
  token = NULL,  
  vars = NULL,  
  obs = NULL,  
  active = FALSE,  
  labels = c("factor", "labelled", "numeric"),  
  header = TRUE,  
  sep = NULL,  
  sheet = 1,  
  range = NULL,  
  skip = 0,  
  na = c("", "NA"),  
  encoding = "UTF-8",  
  check.names = FALSE,  
  records = NULL,  
  fields = NULL,  
  forms = NULL,  
  events = NULL,  
  quiet = FALSE,  
  uid = NULL,  
  server = NULL,  
  form = NULL,  
  access_token = NULL,  
  question_names = c("id", "title"),  
  sheet_id = NULL,  
  drive_id = NULL,  
  item_id = NULL,  
  table = NULL,  
  page_size = 1000L,  
  host = "localhost",  
  port = 3306L,  
  dbname = NULL,  
  database = NULL,  
  user = NULL,  
  password = NULL,  
  query = NULL,  
  ssl_ca = NULL,  
  ssl_cert = NULL,  
  ssl_key = NULL,  
  db_timeout = 10,  
  bigint = "integer64",  
  ...  
)
```

Arguments

file	File path or web URL. A bare file name is first resolved in the current working directory and, when not found there, in R4VN's bundled inst/extdata examples. Thus <code>opendata("ivf_v3_vi.dta")</code> opens the example Stata file shipped with R4VN. For Google Forms or Microsoft Forms exports, this may be the exported CSV/XLSX file. Omit when a direct API/database connector is used.
api	Optional source name: "redcap", "kobo", "googleform", "gsheet", "msexcel", "msforms", "mysql", or "mariadb". Common aliases are accepted.
url	API/project/form/sheet URL when applicable.
token	REDCap or KoboToolbox API token. For Google/Microsoft, use <code>access_token</code> ; token is also accepted as a fallback.
vars	Optional variables to retain.
obs	Optional observations to retain. <code>missing(x)</code> is supported.
active	Logical; also place a working copy in active memory.
labels	How imported value labels are handled: "factor", "labelled", or "numeric".
header	Logical; first text/spreadsheet row contains names.
sep	Text-file delimiter. Defaults from extension.
sheet	Excel/online workbook sheet name or number.
range	Excel/online workbook cell range such as "A1:H500".
skip	Number of rows to skip for local files.
na	Strings interpreted as missing.
encoding	Text encoding.
check.names	Logical; make names syntactically valid.
records, fields, forms, events	Optional REDCap filters.
quiet	Logical; suppress summary messages.
uid	KoboToolbox asset UID. May be inferred from a Kobo project URL.
server	KoboToolbox server. Defaults to the Global server and may be inferred from <code>url</code> .
form	Google Forms form ID. May be inferred from a compatible form URL.
access_token	OAuth bearer token for Google or Microsoft APIs.
question_names	Google Forms variable naming: stable question "id" (default) or sanitized question "title".
sheet_id	Google Sheets spreadsheet ID. May be inferred from <code>url</code> .
drive_id, item_id	Microsoft Graph drive/item identifiers for the Excel workbook containing Microsoft Forms responses.
table	Microsoft Excel table name, or MySQL/MariaDB table name.
page_size	Number of records requested per API page.
host	MySQL/MariaDB server host.

port	MySQL/MariaDB TCP port, usually 3306.
dbname, database	MySQL/MariaDB database name. database is a friendly alias of dbname.
user, password	MySQL/MariaDB credentials. A read-only database account is strongly recommended.
query	Optional read-only SQL query. Use this instead of table for server-side filtering. Write/DDDL statements are rejected.
ssl_ca, ssl_cert, ssl_key	Optional SSL CA/certificate/key paths for MySQL/MariaDB.
db_timeout	Database connection timeout in seconds.
bigint	How 64-bit database integers are returned; passed to RMariaDB.
...	Additional arguments for the existing format-specific local reader.

Details

Existing local-file and REDCap behavior is unchanged. When file is a bare file name that does not exist in the current working directory, `opendata()` also looks in the package's bundled `extdata` directory. This makes the teaching dataset available simply as `opendata("ivf_v3_vi.dta")` after R4VN is installed.

KoboToolbox uses API v2 and Token authentication. Google Forms direct access uses the official Forms API and OAuth. Google Sheets public share links can be read without OAuth when the sheet is accessible to anyone with the link; private sheets can be read with an OAuth access token.

Google Forms and Microsoft Forms response files exported to CSV/XLSX can be opened directly with `opendata(file)`. `api = "googleform"` or `api = "msforms"` may also be supplied together with file as a friendly alias; in that case the normal file reader is used.

Microsoft Forms direct response access is handled through its linked Excel workbook via Microsoft Graph. MySQL/MariaDB uses DBI + RMariaDB and closes the database connection automatically before returning.

Value

A data frame.

Examples

```
# Bundled R4VN teaching dataset (Stata format).
if (requireNamespace("haven", quietly = TRUE) ||
    requireNamespace("readstata13", quietly = TRUE)) {
  ivf <- opendata("ivf_v3_vi.dta", quiet = TRUE)
  head(ivf)
}
```

```
## Not run:
# Existing REDCap behavior
redcap <- opendata(
  api = "redcap",
  url = "https://example.org/api/",
```

```

    token = "REDCAP_TOKEN",
    active = TRUE
)

# KoboToolbox
kobo <- opendata(
  api = "kobo",
  uid = "aBcDeFg123",
  token = "KOBO_TOKEN",
  active = TRUE
)

# Public Google Sheet using only a share link
gsheet <- opendata(
  api = "gsheet",
  url = "https://docs.google.com/spreadsheets/d/SPREADSHEET_ID/edit?gid=0",
  active = TRUE
)

# Named tab and range from a public Google Sheet
gsheet2 <- opendata(
  api = "gsheet",
  url = "https://docs.google.com/spreadsheets/d/SPREADSHEET_ID/edit",
  sheet = "Responses",
  range = "A1:H500"
)

# Google Forms / Microsoft Forms: easiest route after exporting responses
gform_export <- opendata("google-form-responses.xlsx")
msform_export <- opendata("microsoft-form-responses.xlsx")

# Direct Google Forms API (OAuth token required)
gform <- opendata(
  api = "googleform",
  form = "FORM_ID",
  access_token = "GOOGLE_OAUTH_ACCESS_TOKEN"
)

# Microsoft Forms linked Excel workbook via Microsoft Graph
msform <- opendata(
  api = "msforms",
  item_id = "WORKBOOK_ITEM_ID",
  access_token = "MS_GRAPH_ACCESS_TOKEN"
)

# MySQL table
mysql_data <- opendata(
  api = "mysql",
  host = "db.example.org",
  dbname = "research",
  user = "reader",
  password = "PASSWORD",
  table = "participants"
)

```

```

)

# MySQL read-only query
mysql_subset <- opendata(
  api = "mysql",
  host = "db.example.org",
  database = "research",
  user = "reader",
  password = "PASSWORD",
  query = "SELECT id, age, sex FROM participants WHERE age >= 18"
)

## End(Not run)

```

ordervar

*Reorder variables***Description**

Reorders variables in an explicit data frame or the active R4VN data frame.

Usage

```
ordervar(..., data = NULL, before = NULL, after = NULL, last = FALSE)
```

Arguments

...	Variables to move. For backward compatibility, an explicit data frame may be the first unnamed argument.
data	Optional explicit data frame object.
before	Optional anchor variable.
after	Optional anchor variable.
last	Logical; move variables to the end.

Value

The edited data frame invisibly.

Examples

```

d <- data.frame(age = 1, sex = 2, id = 3, bmi = 4)
usedf(d, quiet = TRUE)
ordervar(id, sex)
ordervar(bmi, after = age)

# Extended usage examples

d <- data.frame(age = 1, sex = 2, id = 3, bmi = 4, outcome = 5)

```

```

# Move variables to the beginning
d1 <- d; ordervar(d1, id, outcome)

# Move before or after an anchor
d2 <- d; ordervar(d2, outcome, before = age)
d3 <- d; ordervar(d3, bmi, after = age)

# Move to the end
d4 <- d; ordervar(d4, id, last = TRUE)

# Use a range or wildcard
d5 <- data.frame(id = 1, q1 = 2, q2 = 3, q3 = 4, age = 5)
ordervar(d5, q1:q3, last = TRUE)

# Active-data syntax
usedf(d, quiet = TRUE); ordervar(id, outcome)

```

plot.r4vn_diag

Plot an R4VN diagnostic analysis

Description

Draw ROC curves and/or sensitivity-specificity curves over empirical thresholds from an object returned by `tabdiag()`.

Usage

```

## S3 method for class 'r4vn_diag'
plot(
  x,
  what = c("roc", "cutoff", "both"),
  color = NULL,
  lty = 1,
  line_width = 2,
  legend = TRUE,
  legend_position = "bottomright",
  auc = TRUE,
  diagonal = TRUE,
  diagonal_lty = 2,
  grid = FALSE,
  xlim = c(0, 1),
  ylim = c(0, 1),
  xlab = "1 - Specificity",
  ylab = "Sensitivity",
  main = NULL,
  cutoff_mark = TRUE,

```

```

    cutoff_legend = TRUE,
    cutoff_xlab = "Threshold",
    cutoff_ylab = "Probability",
    cutoff_main = NULL,
    file = NULL,
    width = 7,
    height = 7,
    res = 300,
    bg = "white",
    font_family = NULL,
    cex_axis = 1,
    cex_lab = 1,
    cex_main = 1,
    bty = "l",
    ...
)

```

Arguments

<code>x</code>	An object returned by <code>tabdiag()</code> .
<code>what</code>	"roc", "cutoff", or "both".
<code>color</code>	Optional vector of line colors. Defaults to a distinct base-R qualitative palette.
<code>lty</code>	ROC line type(s).
<code>line_width</code>	ROC/cutoff line width(s).
<code>legend</code>	Logical; show the ROC legend.
<code>legend_position</code>	Base-graphics legend position, default "bottomright".
<code>auc</code>	Logical; append AUC to ROC legend labels.
<code>diagonal</code>	Logical; draw the no-discrimination diagonal.
<code>diagonal_lty</code>	Line type for the no-discrimination diagonal.
<code>grid</code>	Logical; draw a light reference grid.
<code>xlim, ylim</code>	ROC axis limits. Values are clamped to $c(0, 1)$. Defaults are exactly $c(0, 1)$ so the ROC axes never extend below 0 or above 1.
<code>xlab, ylab</code>	ROC axis labels.
<code>main</code>	Optional ROC title.
<code>cutoff_mark</code>	Logical; mark selected cutoffs with vertical reference lines on cutoff plots.
<code>cutoff_legend</code>	Logical; show sensitivity/specificity legend on cutoff plots.
<code>cutoff_xlab, cutoff_ylab</code>	Cutoff-plot axis labels.
<code>cutoff_main</code>	Optional cutoff-plot title. When multiple markers are plotted, the marker label is appended automatically.
<code>file</code>	Optional graphics filename. Supported extensions are .png, .jpg/.jpeg, .tif/.tiff, .pdf, and .svg. For <code>what = "both"</code> , save the ROC and cutoff plots separately with two calls to <code>plot()</code> .

width, height	Figure width and height in inches when file is used. Defaults are 7 and 7.
res	Raster resolution in dpi for PNG/JPEG/TIFF output. Default 300.
bg	Graphics-device background color. Default "white".
font_family	Optional base-graphics font family, for example "Arial", when available on the current system.
cex_axis, cex_lab, cex_main	Text-size controls for axes, labels, title.
bty	Box type passed to base graphics.
...	Additional arguments passed to the initial base plot() call.

Value

Invisibly returns x.

plot.r4vn_distdata	<i>Plot an R4VN probability-distribution result</i>
--------------------	---

Description

Plot an R4VN probability-distribution result

Usage

```
## S3 method for class 'r4vn_distdata'
plot(x, type = c("density", "cdf"), main = NULL, ...)
```

Arguments

x	An object returned by <code>distdata()</code> .
type	"density"/"mass" or "cdf".
main	Optional plot title.
...	Additional graphical parameters passed to base graphics where applicable.

Value

The input object, invisibly.

plot.r4vn_graph	<i>Plot an R4VN graph result</i>
-----------------	----------------------------------

Description

Plot an R4VN graph result

Usage

```
## S3 method for class 'r4vn_graph'
plot(x, ...)

## S3 method for class 'r4vn_graph_set'
plot(x, ncol = NULL, ...)
```

Arguments

x	An R4VN graph or graph collection.
...	Not used.
ncol	Number of columns for a graph collection.

Value

x, invisibly.

plot.r4vn_machine	<i>Plot a tabmachine result</i>
-------------------	---------------------------------

Description

Plot a tabmachine result

Usage

```
## S3 method for class 'r4vn_machine'
plot(x, type = NULL, title = NULL, font_family = "sans", ...)
```

Arguments

x	A r4vn_machine object.
type	Plot type: roc, pr, calibration, threshold, confusion, importance, decision, learning, observed, residual, pdp, or shap.
title	Optional figure title. A publication-ready default is supplied.
font_family	Base-graphics font family. Default "sans", which is also the safest choice for HTML/SVG Viewer rendering.
...	Additional base-graphics arguments where applicable.

Value

The input `r4vn_machine` object, invisibly. The requested figure is drawn on the current graphics device as a side effect; the fitted machine- learning result itself is not modified.

<code>plot.r4vn_meta</code>	<i>Plot an R4VN meta-analysis</i>
-----------------------------	-----------------------------------

Description

Draw forest, funnel, trim-and-fill, influence, cumulative, diagnostic, and meta-regression figures. Every figure can be drawn in the R/RStudio Plot pane or saved independently in a publication format.

Usage

```
## S3 method for class 'r4vn_meta'
plot(
  x,
  type = "forest",
  moderator = NULL,
  contour = FALSE,
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  xlab = NULL,
  ref = NULL,
  xlim = NULL,
  ticks = NULL,
  color = NULL,
  font_family = "sans",
  text_size = 0.82,
  axis_size = 0.9,
  title_size = 1.05,
  title_color = "black",
  subtitle_size = 0.9,
  subtitle_color = "gray30",
  caption_size = 0.75,
  caption_color = "gray40",
  margins = NULL,
  background = "white",
  point_color = NULL,
  point_bg = "white",
  ci_color = NULL,
  summary_color = NULL,
  summary_border = NULL,
  point_shape = NULL,
```

```

    point_size = NULL,
    line_type = 1,
    line_width = 1,
    ref_color = "gray40",
    ref_type = 2,
    ref_width = 1,
    show_weights = TRUE,
    show_prediction = FALSE,
    weight_title = "Weight",
    estimate_title = NULL,
    show_abcd = FALSE,
    abcd_titles = c("a", "b", "c", "d"),
    prediction_style = "bar",
    row_shade = "zebra",
    shade_color = "gray95",
    digits = NULL,
    study_order = NULL,
    header = NULL,
    annotate = TRUE,
    yaxis = "sei",
    ylab = NULL,
    contour_levels = c(90, 95, 99),
    contour_colors = c("#FEE2E2", "#FEF3C7", "#E5E7EB"),
    funnel_label = FALSE,
    funnel_legend = FALSE,
    bubble_ci = TRUE,
    bubble_pi = FALSE,
    bubble_shade = c("#DBEAFE", "#E5E7EB"),
    grid = FALSE,
    engine_args = list(),
    file = NULL,
    width = 1800,
    height = 1400,
    res = 180,
    ...
)

```

Arguments

<code>x</code>	An object created by <code>tabmeta()</code> .
<code>type</code>	Plot type: "forest", "funnel", "trimfill", "baujat", "influence", "leaveout", "cumulative", "radial", "labbe", or "bubble".
<code>moderator</code>	Moderator for a bubble/meta-regression plot.
<code>contour</code>	Logical; create a contour-enhanced funnel plot.
<code>title, subtitle, caption</code>	Main title, subtitle, and figure caption.
<code>xlab</code>	Plot x-axis label.

ref	Reference line on the natural effect scale.
xlim, ticks	Optional effect-scale x limits and tick positions. A study estimate or confidence interval outside xlim is indicated by an arrow; its exact, unclipped value remains in the numerical estimate column.
color	Backward-compatible overall plotting color.
font_family, text_size, axis_size, title_size, title_color	Font family, study-label size, axis size, title size, and title color.
subtitle_size, subtitle_color, caption_size, caption_color	Subtitle and caption sizes and colors.
margins	Optional four-value base-graphics margin vector.
background	Figure background color.
point_color, point_bg, ci_color, summary_color, summary_border	Colors for study points, point fill, confidence intervals, pooled diamond, and its border.
point_shape, point_size	Point symbol and optional fixed point size.
line_type, line_width	Confidence-interval line type and width.
ref_color, ref_type, ref_width	Reference-line color, type, and width.
show_weights	Show study weights in a forest plot.
show_prediction	Show the prediction interval in the forest plot. Default FALSE; the prediction interval remains available in the numerical results when requested in tabmeta(). Set TRUE to draw it for a random-effects model.
weight_title, estimate_title	Headings for the separate weight and numerical estimate columns. estimate_title=NULL uses the effect measure and confidence level. Weight is placed to the right of the numerical estimate column.
show_abcd	Show the four original binary cells in separate forest columns. This is available when tabmeta() was called with a, b, c, and d.
abcd_titles	Four headings used when show_abcd=TRUE.
prediction_style	Prediction display: "line", "polygon", "bar", "shade", or "dist".
row_shade, shade_color	Forest-row shading style and color.
digits	Number of displayed decimals.
study_order	Optional forest ordering vector or metafor order keyword.
header	Forest headings; NULL uses R4VN headings.
annotate	Show effect and confidence-interval annotations.
yaxis, ylab	Funnel-plot y-axis definition and label.
contour_levels, contour_colors	Funnel contour levels and colors.

funnel_label	Label funnel points (FALSE, TRUE, "all", "out", or a number of extreme points).
funnel_legend	Funnel legend control, including a position such as "topright".
bubble_ci, bubble_pi, bubble_shade, grid	Meta-regression confidence and prediction bands, their shading colors, and grid display.
engine_args	Named list passed to the underlying metafor plotting method. This provides access to advanced engine-specific controls.
file	Optional output file. Supported extensions are PNG, JPEG, TIFF, PDF, and SVG.
width, height	Device width and height. Raster units are pixels.
res	Raster resolution.
...	Additional named arguments merged into engine_args.

Value

The meta-analysis object invisibly.

Examples

```
if (requireNamespace("metafor", quietly = TRUE)) {
  dat <- read.csv(system.file("extdata", "meta_example.csv", package = "R4VN"))
  m <- tabmeta(
    dat, study, effect = OR, lower = LCI, upper = UCI, or = TRUE,
    show = FALSE
  )

  # Journal-style forest plot in the Plot pane.
  plot(
    m, type = "forest", font_family = "sans",
    subtitle = "Random-effects model with 95% confidence intervals",
    caption = "Square size reflects study weight; diamond is pooled effect.",
    margins = c(5.5, 4.2, 5.0, 2.0),
    point_color = "#1F4E79", ci_color = "#5B9BD5",
    summary_color = "#C00000", summary_border = "#7F0000",
    row_shade = "zebra", shade_color = "#F5F7FA",
    show_weights = TRUE, weight_title = "Weight",
    estimate_title = "OR (95% CI)", show_prediction = FALSE,
    xlim = c(0.2, 2), ticks = c(0.25, 0.5, 1, 1.5, 2)
  )

  # Contour-enhanced funnel and trim-and-fill plots.
  plot(
    m, type = "funnel", contour = TRUE,
    point_shape = 21, point_color = "#1F4E79", point_bg = "#D9EAF7",
    contour_levels = c(90, 95, 99),
    contour_colors = c("#FFF2CC", "#FCE4D6", "#E2F0D9"),
    funnel_label = "out", funnel_legend = "topright"
  )
  if (!is.null(m$models$trimfill)) {
```

```

    plot(m, type = "trimfill", contour = TRUE)
  }

  # Save a 300-dpi TIFF independently.
  forest_file <- tempfile(fileext = ".tiff")
  plot(
    m, type = "forest", file = forest_file,
    width = 2400, height = 1800, res = 300,
    font_family = "sans", point_color = "#1F4E79",
    ci_color = "#5B9BD5", summary_color = "#C00000"
  )
  unlink(forest_file)

  # Advanced metafor controls.
  plot(m, type = "forest",
        engine_args = list(efac = c(1, 1.2), plim = c(0.6, 1.8)))
}

```

plot.r4vn_tabforest *Plot an R4VN regression forest object*

Description

Re-draws a `tabforest()` object without refitting models. All graphical settings may be overridden here, which is useful for trying different journal layouts after the analysis is finalized.

Usage

```

## S3 method for class 'r4vn_tabforest'
plot(x, ..., file = NULL, width = NULL, height = NULL, dpi = NULL)

```

Arguments

`x` An `r4vn_tabforest` object.

`...` Any plotting option accepted by `tabforest()`.

`file, width, height, dpi` Optional output overrides.

Value

`x`, invisibly.

plot.r4vn_tablong	<i>Plot an R4VN Longitudinal Analysis</i>
-------------------	---

Description

Recreates the observed longitudinal profile stored by `tablong()`. This is useful when the original analysis used `plot = FALSE` or when different plot labels/sizing are wanted without refitting the statistical model.

Usage

```
## S3 method for class 'r4vn_tablong'
plot(x, ci = TRUE, ...)
```

Arguments

<code>x</code>	Object returned by <code>tablong()</code> .
<code>ci</code>	Show 95% confidence intervals. Default TRUE.
<code>...</code>	Named plot options accepted through <code>plot_args</code> , including <code>title</code> , <code>xlab</code> , <code>ylab</code> , <code>line_width</code> , <code>point_size</code> , <code>base_size</code> , and <code>legend_position</code> .

Value

An R4VN plot specification, invisibly; the plot is drawn with base R graphics.

plot.r4vn_tabscale	<i>Plot diagnostics from a scale analysis</i>
--------------------	---

Description

Displays every available tabscale graphic in the R graphics device. In RStudio, use the back/forward arrows in the Plots pane to review them.

Usage

```
## S3 method for class 'r4vn_tabscale'
plot(x, which = NULL, ...)
```

Arguments

<code>x</code>	An object created by <code>tabscale()</code> .
<code>which</code>	Optional plot names or indices. The default displays all plots.
<code>...</code>	Additional arguments currently ignored.

Value

The input object, invisibly.

plot.r4vn_tabscore	<i>Plot an R4VN scorecard</i>
--------------------	-------------------------------

Description

Draw one or all publication-oriented scorecard graphics using base R only. With `which="all"` (the default), every available graph is drawn in sequence; in RStudio the back/forward arrows in the Plots pane can be used to review the complete plot history. The same available graphics are embedded automatically in the HTML Viewer when the original `tabscore()` call used `plot=TRUE`.

Usage

```
## S3 method for class 'r4vn_tabscore'
plot(
  x,
  which = c("all", "risk", "roc", "calibration", "decision", "distribution"),
  title = NULL,
  font_family = "sans",
  ...
)
```

Arguments

<code>x</code>	A <code>r4vn_tabscore</code> object.
<code>which</code>	Plot type: <code>all</code> , <code>risk</code> , <code>roc</code> , <code>calibration</code> , <code>decision</code> , or <code>distribution</code> . <code>all</code> draws every plot that is available for the fitted model family.
<code>title</code>	Optional custom title when one plot is requested. With <code>which="all"</code> , each plot keeps its own descriptive title.
<code>font_family</code>	Base-R graphics font family. Default <code>"sans"</code> is used to keep Viewer, browser and RStudio rendering consistent without another graphics dependency.
<code>...</code>	Additional arguments are reserved.

Value

For one plot, invisibly returns its plotted data. With `which="all"`, invisibly returns a named list containing the data for every graph drawn.

Examples

```
set.seed(23)
d <- data.frame(
  age = rnorm(160, 50, 11),
  smoke = factor(rbinom(160, 1, .30), 0:1, c("No", "Yes"))
)
d$event <- rbinom(160, 1, plogis(-3.5 + .045*d$age + .7*(d$smoke == "Yes")))
z <- tabscore(event, c(age, smoke), data=d,
  validate="none", plot=TRUE, show=FALSE)
```

```

plot(z, which="risk")
plot(z, which="roc")

plot(z) # all available plots, one after another

```

plot.r4vn_tabts

Plot an R4VN tabts result

Description

Plot an R4VN tabts result

Usage

```

## S3 method for class 'r4vn_tabts'
plot(
  x,
  which = c("series", "forecast", "its", "counterfactual", "decomposition", "acf",
    "pacf", "residual"),
  type = NULL,
  file = NULL,
  width = NULL,
  height = NULL,
  dpi = NULL,
  ...
)

```

Arguments

x	An r4vn_tabts object.
which	Plot name such as "series", "forecast", "acf", "pacf", "decomposition", "residual", "its", or "counterfactual".
type	Optional alias for which.
file	Optional output path. Supported extensions are .png, .jpg, .jpeg, .tif, .tiff, .pdf, and .svg.
width, height	Figure width and height in inches when saving.
dpi	Raster resolution when saving PNG/JPEG/TIFF files.
...	Reserved for future use.

Value

Invisibly returns the plot object that is displayed or saved.

Examples

```
data(dengue_ts)
x <- tabts(cases, time = month, data = dengue_ts,
           model = "arima", order = c(1, 0, 1), forecast = 6,
           show = FALSE)
plot(x, "series")
plot(x, type = "forecast")
plot(x, "forecast", file = file.path(tempdir(), "forecast.png"),
     width = 8, height = 5, dpi = 300)
```

poisson

Poisson regression or Poisson family

Description

Fits Poisson regression using formula syntax or compact R4VN syntax. When called without a model, for example `poisson()` or `poisson(link = "identity")`, returns the ordinary `stats::poisson()` family. A binary outcome is also supported; event explicitly identifies the event category and `rr = TRUE` requests a risk-ratio display. Robust or cluster-robust VCE is generally appropriate for modified-Poisson binary models.

Usage

```
poisson(
  y, ..., vars = NULL, data = NULL, exposure = NULL, offset = NULL,
  event = NULL, irr = FALSE, rr = FALSE, exp = FALSE, link = "log",
  noconstant = FALSE, vce = c("model", "robust", "cluster"), cluster = NULL,
  weights = NULL, subset = NULL, ref = NULL, vif = FALSE, diagnosis = FALSE,
  level = 0.95, digits = 3, p_digits = 3, show = TRUE, console = FALSE
)
```

Arguments

<code>y</code>	Formula or count outcome. Omit to obtain the base R Poisson family.
<code>...</code>	Predictors or model terms when <code>y</code> is not a formula.
<code>vars</code>	Optional model terms written as <code>vars(...)</code> .
<code>data</code>	Data frame or <code>NULL</code> for active data.
<code>exposure</code>	Optional person-time variable; its logarithm is used as offset.
<code>offset</code>	Optional offset already on the linear-predictor scale.
<code>event</code>	Event value when <code>y</code> is binary. For a 0/1 variable the default event is 1; for a factor, the second level is used unless specified.
<code>irr</code>	Add an incidence-rate-ratio table for count outcomes.
<code>rr</code>	Add a risk-ratio table for binary outcomes.

exp	Display exponentiated coefficients only (IRR for counts, RR for binary outcomes).
link	Link used only in family mode. Accepts "log", "identity", or "sqrt"; the corresponding link functions are also accepted for compatibility with packages such as MASS.
noconstant	Fit without an intercept.
vce	Model-based, HC1 robust, or cluster-robust covariance.
cluster	Cluster variable.
weights	Optional non-negative weights.
subset	Optional logical subset.
ref	Optional named list of factor reference levels.
vif	Show coefficient-level VIFs.
diagnosis	Logical; if TRUE, append Poisson model diagnostics including Pearson/deviance dispersion, goodness-of-fit, residual/influence measures, influential observations, and collinearity diagnostics. Default FALSE.
level	Confidence level.
digits, p_digits	Decimal places.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional result in the Console. Default FALSE.

Details

The same compact syntax used by `logistic()` is supported: `c.x`, `i.x`, `b2.x`, `ib2.x`, `*` for main effects plus interaction, and `:` for interaction only.

Standard Poisson models fitted with model-based VCE can be compared using `lrtest()`. Quasi-Poisson models do not have an ordinary likelihood and are not supported by `lrtest()`.

Value

If `y` is omitted, a base-R family object for the Poisson distribution with the requested link is returned for compatibility with modeling functions. Otherwise an object of class `r4vn_stat` is returned invisibly. Its sections component contains the formatted model summary, coefficient and/or exponentiated-effect tables, goodness-of-fit results, and any requested VIF table. In `raw`, `model` is the fitted Poisson glm object, `vcov` is the covariance matrix, `coefficients` contains coefficient-level estimates and tests, `logLik` and `null.logLik` are model log likelihoods, `pearson` is the Pearson chi-square statistic, `offset` stores the offset used by the fitted model, and `event`, `binary`, `vce`, and `model.terms` describe binary-event handling, covariance estimation, and fitted terms.

See Also

`logistic()`, `lrtest()`

Examples

```

set.seed(2026)
d <- data.frame(
  cases = rpois(200, 2),
  time = runif(200, .5, 4),
  age = rnorm(200, 45, 12),
  sex = factor(sample(c("Female", "Male"), 200, TRUE)),
  treatment = factor(sample(c("No", "Yes"), 200, TRUE))
)

p1 <- poisson(
  cases,
  c.age,
  i.sex,
  i.treatment,
  data = d,
  exposure = time,
  show = FALSE
)

p2 <- poisson(
  cases,
  c.age,
  i.sex*i.treatment,
  data = d,
  exposure = time,
  show = FALSE
)

lrtest(p1, p2, show = FALSE)

# Modified Poisson for a binary outcome
d$event01 <- as.integer(d$cases > 1)
poisson(event01, c.age, i.sex, data = d, event = 1, rr = TRUE, vce = "robust")

# Dispersion, residual, influence, and collinearity diagnostics
poisson(cases, c.age, i.sex, data = d, exposure = time, diagnosis = TRUE, show = FALSE)

```

predict

Prediction and postestimation diagnostics

Description

predict() provides one consistent R4VN postestimation interface. With an explicit fitted model and no newvar, ordinary prediction types continue to delegate to the model's stats::predict() method. R4VN also adds common residual and influence statistics. In variable-generation mode, provide newvar and R4VN writes the selected statistic back to data or the active data frame while preserving omitted estimation rows as NA.

Usage

```
predict(
  object = NULL, ..., newvar = NULL, type = "auto", term = NULL,
  data = NULL, replace = FALSE, show = TRUE
)
```

Arguments

<code>object</code>	Optional fitted model or R4VN model result. If omitted when generating a variable, the most recent active model is used.
<code>...</code>	Additional model-specific arguments. <code>newdata</code> may be supplied here for ordinary explicit-model prediction and is normalized to the predictor types retained by the fitted model. Other examples include <code>level</code> and <code>interval</code> for linear-model prediction or arguments accepted by the underlying model's prediction method.
<code>newvar</code>	Name of a variable to create. It may be unquoted, for example <code>newvar = stdres</code> , or supplied as one character string. If omitted, the requested statistic is returned instead of being written to data.
<code>type</code>	Statistic to obtain. Common prediction aliases are "auto", "response", "fitted", "predicted", "probability"/"pr", and "link"/"xb". Residual types include "residual", "pearson", "deviance", "working", "standardized"/"stdres"/"rstandard", and "studentized"/"studres"/"rstudent". Influence statistics include "leverage"/"hat", "cooks", "dffits", "covratio", "dfbeta", and "dfbetas". "se.fit" returns prediction standard errors. For linear models, "lower" and "upper" return confidence-limit columns. Cox models additionally support "risk", "lp", "expected", "terms", "martingale", "deviance", "score", "schoenfeld", "scaledsch", and "partial" when supported by survival.
<code>term</code>	Optional coefficient/term name or column number when a statistic naturally returns several columns, notably <code>dfbeta</code> , <code>dfbetas</code> , Cox score, Schoenfeld, scaled Schoenfeld, partial residuals, or term predictions. If omitted for a multi-column result, R4VN reports the available terms.
<code>data</code>	Data frame used for prediction and/or receiving <code>newvar</code> . When omitted in generation mode, the active data frame is used.
<code>replace</code>	Logical; allow an existing <code>newvar</code> to be overwritten.
<code>show</code>	Logical; display a short generation message. Default TRUE.

Details

Standardized residuals are computed with `stats::rstandard()` and studentized residuals with `stats::rstudent()` when those methods are available. These are different from raw residuals. For linear regression, leverage is obtained with `hatvalues()`, Cook's distance with `cooks.distance()`, DFFITS with `dffits()`, and COVRATIO with `covratio()`.

Influence statistics and residuals are defined for the estimation sample. When they are written to the original active data, observations omitted from model fitting because of missing values are filled with NA.

When R4VN compact syntax declared a predictor categorical (for example `i.htn`) but the original data store it as numeric 0/1, prediction data are automatically reconstructed with the factor levels retained by the fitted model. Unknown new levels remain an error rather than being silently recoded.

Value

Without `newvar`, returns the requested prediction, residual, or diagnostic statistic. With `newvar`, invisibly returns the updated data frame after writing the generated variable.

Examples

```
# Linear regression: fitted values and regression diagnostics
d <- data.frame(
  y = c(12, 15, 17, 20, 21, 25, 28, 31, 35, 38),
  age = seq(20, 65, by = 5),
  bmi = c(19, 21, 20, 23, 25, 24, 27, 28, 30, 29)
)
usedf(d)
m1 <- regress(y, c.age, c.bmi, show = FALSE)
predict(m1, type = "response")
predict(m1, type = "standardized")
predict(m1, type = "studentized")
predict(m1, type = "leverage")
predict(m1, type = "cooksd")
predict(m1, type = "dffits")
predict(m1, type = "covratio")

# Store diagnostics in the active data frame
predict(newvar = fitted_y, type = "fitted", show = FALSE)
predict(newvar = residual_y, type = "residual", show = FALSE)
predict(newvar = stdres, type = "standardized", show = FALSE)
predict(newvar = studres, type = "studentized", show = FALSE)
predict(newvar = leverage, type = "leverage", show = FALSE)
predict(newvar = cooksd, type = "cooksd", show = FALSE)

# DFBETA/DFBETAS are coefficient-specific; select a term when generating
names(stats::coef(m1$raw$model))
predict(m1, type = "dfbetas", term = "age")
predict(m1, newvar = dfb_age, type = "dfbetas", term = "age", show = FALSE)

# Linear-model prediction standard error and confidence limits
predict(m1, type = "se.fit")
predict(m1, type = "lower", level = 0.95)
predict(m1, type = "upper", level = 0.95)

# Logistic regression: probability and residual diagnostics
g <- data.frame(
  outcome = factor(c(0,0,0,0,1,0,1,1,1,1,1), levels = 0:1,
    labels = c("No", "Yes")),
  age = seq(25, 80, by = 5),
  bmi = c(20,21,22,24,23,26,25,28,29,30,31,33)
)
```

```

usedf(g)
m2 <- logistic(outcome, c.age, c.bmi, event = "Yes", show = FALSE)
predict(m2, type = "probability")
predict(m2, type = "pearson")
predict(m2, type = "deviance")
predict(m2, type = "standardized")
predict(m2, type = "leverage")
predict(m2, type = "cooks")

```

predict.r4vn_machine *Predict from a tabmachine model*

Description

Predict from a tabmachine model

Usage

```

## S3 method for class 'r4vn_machine'
predict(
  object,
  newdata,
  type = c("response", "prob", "class"),
  threshold = NULL,
  ...
)

```

Arguments

object	A fitted r4vn_machine object.
newdata	New predictor data.
type	"response", "prob", or "class".
threshold	Optional binary threshold overriding the training-derived final threshold.
...	Unused.

Value

Predictions whose structure depends on the task and type. For a regression task, a numeric vector of predicted outcomes is returned. For a binary task, type = "response" or "prob" returns a numeric vector of probabilities for the modeled event, while type = "class" returns a factor of predicted classes. For a multiclass task, type = "response" or "prob" returns a numeric matrix of class probabilities and type = "class" returns a factor of predicted classes.

predict.r4vn_tabscore *Predict scores and risk from an R4VN scorecard*

Description

Predict scores and risk from an R4VN scorecard

Usage

```
## S3 method for class 'r4vn_tabscore'
predict(
  object,
  newdata,
  type = c("all", "score", "risk", "group", "model", "model_lp", "model_score"),
  times = NULL,
  ...
)
```

Arguments

object	A r4vn_tabscore object.
newdata	New data containing all final scorecard predictors.
type	Output type: all, score, risk, group, model, model_lp, or model_score. model returns original-model response prediction for logistic/Poisson and linear predictor for Cox. model_lp always returns the original-model linear predictor.
times	Cox prediction horizon(s). Defaults to the horizons stored in the scorecard.
...	Reserved.

Value

A numeric vector, matrix, or data frame depending on type.

print.r4vn_graph *Print an R4VN graph result*

Description

Returns an R4VN graph object invisibly without writing status text to the console.

Usage

```
## S3 method for class 'r4vn_graph'
print(x, ...)
```

Arguments

x	An object of class r4vn_graph.
...	Not used.

Value

x, invisibly.

print.r4vn_graph_set *Return an R4VN graph collection silently*

Description

Returns the collection invisibly without writing graph status or row counts to the Console. Use `plot()` to redraw the collection.

Usage

```
## S3 method for class 'r4vn_graph_set'
print(x,
      ...)
```

Arguments

x	An R4VN graph collection.
...	Not used.

Value

The graph collection, invisibly.

print.r4vn_machine *Print a tabmachine result*

Description

Print a tabmachine result

Usage

```
## S3 method for class 'r4vn_machine'
print(x, ...)
```

Arguments

x A r4vn_machine object.
... Unused.

Value

The input r4vn_machine object, invisibly, after printing a compact summary of the analysis and the final-model performance table.

print.r4vn_meta	<i>Print or reopen an R4VN meta-analysis</i>
-----------------	--

Description

Print or reopen an R4VN meta-analysis

Usage

```
## S3 method for class 'r4vn_meta'  
print(x, ...)
```

Arguments

x An object created by tabmeta().
... Additional arguments ignored.

Value

The input object invisibly.

print.r4vn_quick	<i>Print a quick R4VN console summary</i>
------------------	---

Description

Print a quick R4VN console summary

Usage

```
## S3 method for class 'r4vn_quick'  
print(x, ...)
```

Arguments

x An object returned by tab1() or sum1().
... Unused.

Value

x, invisibly.

print.r4vn_stat	<i>Print an R4VN statistical result</i>
-----------------	---

Description

Print an R4VN statistical result

Usage

```
## S3 method for class 'r4vn_stat'
print(x, ...)
```

Arguments

x	An object of class r4vn_stat.
...	Unused.

Value

The input object, invisibly.

print.r4vn_tabscale	<i>Print or reopen a scale analysis</i>
---------------------	---

Description

Print or reopen a scale analysis

Usage

```
## S3 method for class 'r4vn_tabscale'
print(x, ...)
```

Arguments

x	An object created by tabscale().
...	Additional arguments currently ignored.

Value

The input object, invisibly.

`prop`*Proportions and Tests of Proportions*

Description

Immediate and data-based commands for confidence intervals, exact binomial tests, and one- or two-sample z tests of proportions. `ptest()` and `ptest1()` support a one-sample null proportion through `p0`; `ptest()` also supports hierarchical grouping.

Usage

```
prop(
  events,
  total,
  p0 = NULL,
  method = c("all", "exact", "wilson", "wald"),
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

bitesti(
  total,
  events,
  p = 0.5,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

ptest1(
  events1,
  n1,
  events2 = NULL,
  n2 = NULL,
  p0 = NULL,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
```

```

    console = FALSE
)

propi(
  events, total, p0 = NULL, method = c("all", "exact", "wilson", "wald"),
  alternative = c("two.sided", "less", "greater"), level = 0.95, digits = 3,
  p_digits = 3, show = TRUE, console = FALSE
)

bitesti(
  total, events, p = 0.5, alternative = c("two.sided", "less", "greater"),
  level = 0.95, digits = 3, p_digits = 3, show = TRUE, console = FALSE
)

prtesti(
  events1, n1, events2 = NULL, n2 = NULL, p0 = NULL,
  alternative = c("two.sided", "less", "greater"), level = 0.95, digits = 3,
  p_digits = 3, show = TRUE, console = FALSE
)

prop(
  x, data = NULL, event = NULL, p0 = NULL,
  method = c("all", "exact", "wilson", "wald"),
  alternative = c("two.sided", "less", "greater"), level = 0.95, digits = 3,
  p_digits = 3, show = TRUE, console = FALSE
)

bitest(
  x, p = 0.5, data = NULL, event = NULL,
  alternative = c("two.sided", "less", "greater"), level = 0.95, digits = 3,
  p_digits = 3, show = TRUE, console = FALSE
)

prtest(
  x, by = NULL, data = NULL, event = NULL, p0 = NULL,
  alternative = c("two.sided", "less", "greater"), level = 0.95, digits = 3,
  p_digits = 3, show = TRUE, console = FALSE
)

prop(
  x,
  data = NULL,
  event = NULL,
  p0 = NULL,
  method = c("all", "exact", "wilson", "wald"),
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,

```

```

    p_digits = 3,
    show = TRUE,
    console = FALSE
  )

  bitest(
    x,
    p = 0.5,
    data = NULL,
    event = NULL,
    alternative = c("two.sided", "less", "greater"),
    level = 0.95,
    digits = 3,
    p_digits = 3,
    show = TRUE,
    console = FALSE
  )

  prtest(
    x,
    by = NULL,
    data = NULL,
    event = NULL,
    p0 = NULL,
    alternative = c("two.sided", "less", "greater"),
    level = 0.95,
    digits = 3,
    p_digits = 3,
    show = TRUE,
    console = FALSE
  )

```

Arguments

events, total	Number of events and total observations.
p0	Null proportion.
method	Confidence interval method: "all", "exact", "wilson", or "wald".
alternative	Alternative hypothesis.
level	Confidence level.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
p	Null probability for bitesti() and bitest().
events1, n1	Events and total in sample 1.
events2, n2	Optional events and total in sample 2.

x	Binary variable.
data	Data frame. If NULL, active data is used.
event	Event level. The last observed level is used by default.
by	Optional two-level grouping variable. For <code>prtest()</code> , <code>by = vars(region, sex, group)</code> performs the two-proportion comparison inside region and sex strata.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```

prop(32, 100, p0 = .25)
bitesti(40, 23, p = 0.5)
prtest(30, 100, p0 = .20)
prtest(30, 100, 20, 80)

d <- data.frame(case = c(rep(1,30), rep(0,70), rep(1,20), rep(0,60)),
                 group = rep(c("A","B"), c(100,80)))
prtest(case, data = d, p0 = .25, event = 1)
prtest(case, by = group, data = d, event = 1)

```

qregress	<i>Quantile regression</i>
----------	----------------------------

Description

Fits one or several conditional quantile regression models using the optional **quantreg** package and R4VN model syntax.

Usage

```

qregress(y,
  ...,
  vars = NULL,
  data = NULL,
  tau = 0.5,
  method = "br",
  se = "nid",
  weights = NULL,
  subset = NULL,
  ref = NULL,
  noconstant = FALSE,
  diagnosis = FALSE,
  level = 0.95,
  digits = 3,

```

```
p_digits = 3,
show = TRUE,
console = FALSE)
```

Arguments

<code>y</code>	Outcome variable or formula.
<code>...</code>	R4VN predictor terms.
<code>vars</code>	Optional <code>vars(...)</code> predictor specification.
<code>data</code>	Data frame or active data.
<code>tau</code>	One or more quantiles strictly between 0 and 1.
<code>method</code>	Algorithm passed to <code>quantreg::rq()</code> .
<code>se</code>	Standard-error method passed to <code>summary.rq()</code> .
<code>weights, subset, ref, noconstant</code>	Model controls consistent with other R4VN regression commands.
<code>diagnosis</code>	Logical; if TRUE, append quantile-regression diagnostics for the primary quantile, including residual median, MAD, IQR, residual balance around zero, and quantile check loss. Default FALSE.
<code>level, digits, p_digits, show, console</code>	Confidence, formatting and display controls.

Details

The model nearest $\tau = 0.5$ is stored as the primary active model; all requested quantile fits are retained in `raw$models`.

Value

An R4VN result object, invisibly.

See Also

[regress](#), [nlregress](#)

Examples

```
if (requireNamespace("quantreg", quietly = TRUE)) {
  # Use a reasonably sized, full-rank data set so the example is stable
  # across quantreg and R versions.
  d <- datasets::mtcars
  d$am <- factor(d$am, levels = c(0, 1),
                labels = c("Automatic", "Manual"))

  # Median regression with one continuous and one categorical predictor.
  qregress(mpg, c.wt, i.am, data = d, tau = .5,
           se = "iid", show = FALSE)

  # Fit several conditional quantiles in one call.
```

```

qregress(mpg, c.wt, i.am, data = d,
         tau = c(.25, .5, .75), se = "iid", show = FALSE)

# Request the R4VN quantile-regression diagnostic section.
qregress(mpg, c.wt, i.am, data = d, tau = .5,
         se = "iid", diagnosis = TRUE, show = FALSE)
}

```

r4fun

Launch R4VN Playground

Description

Starts a lightweight collection of short games, puzzles, and relaxation activities bundled with R4VN. The interface can be switched between English and Vietnamese. The function is intentionally independent of active data and statistical workflows.

Usage

```

r4fun(
  language = c("en", "vi"),
  launch.browser = TRUE,
  host = "127.0.0.1",
  port = NULL,
  ...
)

```

Arguments

language	Initial interface language. Use "en" for English or "vi" for Vietnamese.
launch.browser	Logical or a browser function passed to shiny::runApp() .
host	Host passed to shiny::runApp() . The default "127.0.0.1" keeps the app local to the user's computer.
port	Optional port. NULL lets Shiny choose an available port.
...	Additional arguments passed to shiny::runApp() .

Details

The Playground uses only the optional package shiny. Vietnamese interface strings are stored in Unicode escape form so the R source file remains ASCII-only and portable across locales. Games include a collapsible Trick section with a practical strategy or shortcut for easier play.

Value

Invisibly returns the value from [shiny::runApp\(\)](#).

Examples

```
if (interactive() && requireNamespace("shiny", quietly = TRUE)) {
  r4fun()
}
```

R4VN_examples

R4VN Example Cookbook

Description

A central index of practical examples. The detailed examples are also merged into the individual help pages, so use `?labvar`, `?tab`, `?ttest`, `?ranksum`, and the corresponding command name while working.

Data workflow

`usedf()` sets active data. `opendata()` and `savedata()` import and export. `appenddata()` stacks observations; `mergedata()` joins by keys. `keepvar()`, `dropvar()`, `ordervar()`, `renvar()`, `genvar()`, `replacevar()`, and `labvar()` edit active or explicit data.

Statistical workflow

`tab1()` and `sum1()` provide quick console checks, including nested by stratification. `describe()` shows active-data structure. `tab()` and `tabmulti()` produce publication tables. `ttest()` includes one-sample, independent, and paired t tests. `ranksum()`, `signrank()`, `kwallis()`, and `friedman()` provide the principal nonparametric tests. `swilk()` assesses normality. `corr()`, `regress()`, `logistic()`, and `poisson()` provide correlation and regression models. `tabsurv()` and `tabmeta()` provide complete one-command survival and meta-analysis reports; their help pages contain scenario-based cookbooks.

Graph workflow

`gbar()`, `ghist()`, `gbox()`, `gscatter()`, `gline()`, `gdensity()`, and `gpie()`, `gforest()`, and `groc()` use base graphics and return reusable `r4vn_graph` objects.

Examples

```
# Discover all examples attached to a command
example(labvar)
example(ttest)
example(ranksum)
example(tab1)
example(sum1)
example(describe)
example(signrank)
example(kwallis)
example(friedman)
example(swilk)
```

```

example(tabmeta)

# Open the full help pages
help("labvar", package = "R4VN")
help("tabmeta", package = "R4VN")
help("R4VN_examples", package = "R4VN")

```

ranksum

*Wilcoxon Rank-sum Test***Description**

Performs the Wilcoxon rank-sum test, also known as the Mann-Whitney U test, for two independent samples. Samples may be supplied as two numeric variables or as one numeric outcome and a two-level grouping variable.

Usage

```

ranksum(
  x,
  y = NULL,
  by = NULL,
  data = NULL,
  alternative = c("two.sided", "less", "greater"),
  exact = NULL,
  correct = TRUE,
  conf.int = TRUE,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

```

Arguments

x	Numeric outcome or first numeric sample.
y	Optional second numeric sample.
by	Optional two-level grouping variable. Use either y or by.
data	Data frame. If NULL, the active R4VN data frame is used.
alternative	Alternative hypothesis: "two.sided", "less", or "greater".
exact	Use an exact p-value when possible. NULL lets R decide.
correct	Apply continuity correction for the normal approximation.

<code>conf.int</code>	Report the Hodges-Lehmann location-shift estimate and its confidence interval when available.
<code>level</code>	Confidence level.
<code>digits, p_digits</code>	Decimal places for estimates and p-values.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.

Details

With `by`, the first observed factor level is sample 1 and the second level is sample 2. Use `factor()` or `labvar(..., ref = ...)` to control level order. Missing values are removed independently when `x` and `y` are supplied, and complete cases are used when `by` is supplied.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
d <- data.frame(
  score = c(12, 15, 11, 19, 18, 21, 14, 17),
  group = factor(rep(c("Control", "Intervention"), each = 4)),
  score2 = c(10, 13, 12, 14, 19, 20, 18, 22)
)
ranksum(score, by = group, data = d)
ranksum(score, score2, data = d)
ranksum(score, by = group, data = d, alternative = "less")

# Extended usage examples

d <- data.frame(
  score = c(10, 11, 12, 13, 18, 19, 20, 21),
  score2 = c(9, 10, 12, 11, 17, 18, 19, 22),
  group = factor(rep(c("Control", "Intervention"), each = 4))
)

# Two independent variables or one outcome by a two-level group
ranksum(score, score2, data = d)
ranksum(score, by = group, data = d)

# One-sided alternatives, approximation controls, and confidence interval
ranksum(score, by = group, data = d, alternative = "less")
ranksum(score, by = group, data = d, exact = FALSE, correct = FALSE)
ranksum(score, by = group, data = d, conf.int = FALSE)

# Active data and hidden console output
usedf(d)
result <- ranksum(score, by = group, show = FALSE)
result$raw$test
```

regress	<i>Linear regression</i>
---------	--------------------------

Description

Fits an ordinary least-squares model. R4VN compact syntax allows models such as `regress(y, c.age, i.sex, i.sex*i.treatment)` without `~` or `+`.

Usage

```
regress(
  y,
  ...,
  vars = NULL,
  data = NULL,
  noconstant = FALSE,
  vce = c("model", "robust", "cluster"),
  cluster = NULL,
  weights = NULL,
  subset = NULL,
  ref = NULL,
  standardized = FALSE,
  vif = FALSE,
  diagnosis = FALSE,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)
```

Arguments

<code>y</code>	Formula or numeric outcome variable.
<code>...</code>	Predictors or model terms when <code>y</code> is not a formula.
<code>vars</code>	Optional model terms written as <code>vars(...)</code> .
<code>data</code>	Data frame or <code>NULL</code> for active data.
<code>noconstant</code>	Fit without an intercept.
<code>vce</code>	Model-based, HC1 robust, or cluster-robust covariance.
<code>cluster</code>	Cluster variable used when <code>vce = "cluster"</code> .
<code>weights</code>	Optional non-negative analytic weights.
<code>subset</code>	Optional logical subset expression.
<code>ref</code>	Optional named list of factor reference levels.
<code>standardized</code>	Also display standardized coefficients for numeric columns.

<code>vif</code>	Also display coefficient-level variance inflation factors.
<code>diagnosis</code>	Logical; if TRUE, append model-diagnostic tables. For linear regression these include residual normality, a Breusch-Pagan heteroscedasticity test, standardized/studentized residuals, leverage, Cook's distance, DFFITS, COVRATIO, influential observations, and collinearity diagnostics. Default FALSE.
<code>level</code>	Confidence level.
<code>digits, p_digits</code>	Decimal places.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.

Details

Compact model prefixes are `c.x` for continuous, `i.x` for categorical, `b2.x` for the second factor level as reference, and `ib2.x` for value/level 2 as reference. Use `*` for main effects plus interaction and `:` for interaction only.

Ordinary linear regression models should be compared with the usual nested F test rather than [lrtest\(\)](#).

Value

An object of class `r4vn_stat`, returned invisibly. Its `sections` component contains the formatted model summary, ANOVA, coefficient table, and any requested standardized-coefficient or VIF tables. In `raw`, `model` is the fitted `lm` object, `vcov` is the covariance matrix, `coefficients` contains coefficient-level estimates and tests, `overall` contains the overall model test, and `vce` and `model.terms` record the covariance estimator and fitted terms.

Examples

```
d <- data.frame(score = c(60, 65, 68, 72, 75, 80, 77, 70),
                 age = c(20, 25, 30, 35, 40, 45, 50, 55),
                 bmi = c(20, 22, 24, 23, 26, 28, 27, 25),
                 sex = factor(rep(c("Female", "Male"), 4)))

regress(score, age, bmi, i.sex, data = d, show = FALSE)
regress(score, vars = vars(c.age, c.bmi, i.sex), data = d, show = FALSE)
# Full model diagnostics
m <- regress(score, c.age, c.bmi, i.sex, data = d, diagnosis = TRUE, show = FALSE)
m$sections$`Model diagnosis`
m$sections$`Influence diagnostics`
# Postestimation diagnostics can also be generated as variables
usedf(d)
regress(score, c.age, c.bmi, i.sex, diagnosis = FALSE, show = FALSE)
predict(newvar = stdres, type = "standardized", show = FALSE)
predict(newvar = cooks, type = "cooks", show = FALSE)
```

renvar	<i>Rename variables</i>
--------	-------------------------

Description

Renames variables in an explicit data frame or the active R4VN data frame.

Usage

```
renvar(..., data = NULL)
```

Arguments

...	In active mode: old, new. In explicit mode: data, old, new.
data	Optional explicit data frame object.

Value

The edited data frame invisibly.

Examples

```
d <- data.frame(sex = 1:2, age_kt = 3:4)
usedf(d, quiet = TRUE)
renvar(sex, gender)
renvar("*_kt", "*")

# Extended usage examples

d <- data.frame(sex = 1:2, age = 3:4, score_pre = 5:6, bmi_pre = 7:8)

# Rename one variable
d1 <- d; renvar(d1, sex, gender)

# Rename parallel groups
d2 <- d; renvar(d2, vars(sex, age), vars(gender, age_year))

# Space-separated names are also accepted
d3 <- d; renvar(d3, "sex age", "gender age_year")

# Wildcard: remove or replace a common suffix
d4 <- d; renvar(d4, "*_pre", "*")
d5 <- d; renvar(d5, "*_pre", "baseline_*)

# Active-data syntax
usedf(d, quiet = TRUE); renvar(sex, gender)
```

replacevar	<i>Replace values under one or more conditions</i>
------------	--

Description

Replaces values in an explicit data frame or the active R4VN data frame.

Usage

```
replacevar(..., data = NULL)
```

Arguments

...	In active mode: variable, value, condition. In explicit mode: data, variable, value, condition
data	Optional explicit data frame object.

Details

missing(x) may be used inside conditions. Use . as a missing replacement.

Value

The edited data frame invisibly.

Examples

```
d <- data.frame(age = c(8, 15, 200, NA_real_))
usedf(d, quiet = TRUE)
replacevar(age, ., age > 120)
replacevar(age, 99, missing(age))

# Extended usage examples

d <- data.frame(age = c(8, 15, 200, NA_real_),
                sex = factor(c("M", "F", "M", "F")))
usedf(d, quiet = TRUE)

# Replace an impossible value by missing
replacevar(age, ., age > 120)

# Replace missing values
replacevar(age, 99, missing(age))

# Replace a factor value under a condition
replacevar(sex, "Female", sex == "F")

# Explicit-data syntax
replacevar(d, age, ., age > 120)
```

savedata	<i>Save data according to file extension</i>
----------	--

Description

Saves active data or an explicitly supplied data frame. Variables and observations can be selected without changing the source data.

Usage

```
savedata(  
  data = NULL,  
  file = NULL,  
  vars = NULL,  
  obs = NULL,  
  sheet = "Data",  
  header = TRUE,  
  sep = NULL,  
  na = "",  
  encoding = "UTF-8",  
  replace = FALSE,  
  quiet = FALSE,  
  ...  
)
```

Arguments

data	Data frame, or output path when saving active data.
file	Output path. Omit when the first argument is the path.
vars	Optional variables to save.
obs	Optional observations to save. <code>missing(x)</code> is supported.
sheet	Excel sheet name.
header	Logical; write variable names for text files.
sep	Text-file delimiter. Defaults from extension.
na	Missing-value text for delimited files.
encoding	Text encoding.
replace	Logical; overwrite an existing file.
quiet	Logical; suppress summary messages.
...	Additional writer arguments.

Details

Use `savedata("file.csv")` for active data, or `savedata(data, "file.csv")` for an explicit data frame.

Value

The normalized saved path invisibly.

Examples

```
d <- datasets::iris
usedf(d, quiet = TRUE)

# Base-R formats: short examples run during R CMD check.
f_csv <- tempfile(fileext = ".csv")
f_sub <- tempfile(fileext = ".csv")
f_rds <- tempfile(fileext = ".rds")

savedata(f_csv, quiet = TRUE)
savedata(
  f_sub,
  vars = vars(Sepal.Length, Species),
  obs = Species == "setosa",
  quiet = TRUE
)
savedata(d, f_rds, quiet = TRUE)
unlink(c(f_csv, f_sub, f_rds))
usedf(clear = TRUE, quiet = TRUE)

# Optional formats are guarded because their writers are in Suggests.
if (requireNamespace("writexl", quietly = TRUE) ||
    requireNamespace("openxlsx", quietly = TRUE)) {
  f <- tempfile(fileext = ".xlsx")
  savedata(d, f, sheet = "Data", quiet = TRUE)
  unlink(f)
}

if (requireNamespace("haven", quietly = TRUE)) {
  # Stata/SPSS impose variable-name rules. Use portable names here.
  d_haven <- data.frame(
    sepal_length = d$Sepal.Length,
    sepal_width = d$Sepal.Width,
    petal_length = d$Petal.Length,
    petal_width = d$Petal.Width,
    species = as.character(d$Species),
    stringsAsFactors = FALSE
  )
  f_dta <- tempfile(fileext = ".dta")
  f_sav <- tempfile(fileext = ".sav")
  savedata(d_haven, f_dta, quiet = TRUE)
  savedata(d_haven, f_sav, quiet = TRUE)
  unlink(c(f_dta, f_sav))
}

if (requireNamespace("jsonlite", quietly = TRUE)) {
  f <- tempfile(fileext = ".json")
  savedata(d, f, quiet = TRUE)
}
```

```
    unlink(f)
  }

  if (requireNamespace("arrow", quietly = TRUE)) {
    f <- tempfile(fileext = ".parquet")
    savedata(d, f, quiet = TRUE)
    unlink(f)
  }
```

sdtesti*Tests of Standard Deviations and Variances*

Description

Performs a one-sample chi-squared variance test or a two-sample F test.

Usage

```
sdtesti(
  n1,
  sd1,
  n2 = NULL,
  sd2 = NULL,
  sd0 = NULL,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

sdtest(
  x,
  y = NULL,
  by = NULL,
  data = NULL,
  sd0 = NULL,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)
```

Arguments

n1, sd1	Sample size and standard deviation for sample 1.
n2, sd2	Optional sample size and standard deviation for sample 2.
sd0	Null standard deviation for a one-sample test.
alternative	Alternative hypothesis.
level	Confidence level.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
x, y	Numeric variables.
by	Optional two-level grouping variable.
data	Data frame. If NULL, active data is used.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
sdtesti(40, 3, sd0 = 2.5)
sdtesti(40, 3, 35, 4)
```

 shapevar

Reshape Data Between Wide and Long Formats

Description

Converts repeated-measures data from wide to long format or from long to wide format using base R only. The active R4VN data frame is used when data is omitted.

Usage

```
shapevar(
  to = NULL,
  long = FALSE,
  wide = FALSE,
  id,
  vars,
  time = time,
  value = value,
  times = NULL,
  sep = "_",
  data = NULL,
  quiet = FALSE
)
```

Arguments

<code>to</code>	Optional target shape: "long" or "wide". The shorter R4VN-style alternatives <code>long = TRUE</code> and <code>wide = TRUE</code> are also supported.
<code>long, wide</code>	Logical shortcuts. Use exactly one when <code>to</code> is omitted.
<code>id</code>	One or more subject/record identifier variables. Use a bare name or <code>vars(...)</code> .
<code>vars</code>	Variables to reshape, usually supplied by <code>vars(...)</code> .
<code>time</code>	Name of the time/index variable. In wide-to-long conversion this is the new time variable; in long-to-wide conversion it is an existing variable.
<code>value</code>	Name of the new value variable for wide-to-long conversion. Ignored for long-to-wide conversion.
<code>times</code>	Optional values assigned to the repeated wide columns. When omitted, <code>shapevar()</code> tries to infer suffixes from the selected variable names and otherwise uses 1, 2,
<code>sep</code>	Separator between value-variable names and time values when creating wide variable names.
<code>data</code>	Optional explicit data-frame object. When omitted, active data is reshaped and replaced directly.
<code>quiet</code>	Logical; suppress the reshape summary.

Details

Wide to long example:

```
shapevar(long = TRUE, id = id, vars = vars(bp1, bp2, bp3),      time = visit, value = bp)
```

Long to wide example:

```
shapevar(wide = TRUE, id = id, time = visit, vars = vars(bp))
```

For long-to-wide conversion, each `id` by `time` combination must be unique. Variables not included in `vars` are preserved when they are constant within each ID. A changing non-reshaped variable triggers an error rather than being silently discarded.

Value

The reshaped data frame invisibly.

Examples

```
wide <- data.frame(id = 1:2, sex = c("F", "M"),
                  bp1 = c(120, 130), bp2 = c(118, 128), bp3 = c(116, 125))
usedf(wide, quiet = TRUE)
shapevar(long = TRUE, id = id, vars = vars(bp1, bp2, bp3),
        time = visit, value = bp, quiet = TRUE)
long <- usedf(quiet = TRUE)
shapevar(wide = TRUE, id = id, time = visit, vars = vars(bp), quiet = TRUE)
```

signrank

*Wilcoxon Signed-rank Test***Description**

Performs a one-sample or paired Wilcoxon signed-rank test. With two variables, complete pairs are analyzed and the tested difference is $x - y$.

Usage

```
signrank(
  x,
  y = NULL,
  data = NULL,
  mu = 0,
  alternative = c("two.sided", "less", "greater"),
  exact = NULL,
  correct = TRUE,
  conf.int = TRUE,
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)
```

Arguments

<code>x</code>	Numeric variable or first paired measurement.
<code>y</code>	Optional second paired measurement.
<code>data</code>	Data frame. If NULL, active R4VN data is used.
<code>mu</code>	Null median or null median paired difference.
<code>alternative</code>	Alternative hypothesis.
<code>exact</code>	Use an exact p-value when possible. NULL lets R decide.
<code>correct</code>	Apply continuity correction for the normal approximation.
<code>conf.int</code>	Report a pseudomedian or paired location-shift estimate and confidence interval when available.
<code>level</code>	Confidence level.
<code>digits, p_digits</code>	Decimal places for estimates and p-values.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
d <- data.frame(
  before = c(18, 20, 15, 22, 17, 19, 24, 16),
  after  = c(15, 18, 14, 19, 16, 17, 20, 15)
)
signrank(before, data = d, mu = 18)
signrank(before, after, data = d)
signrank(before, after, data = d, alternative = "greater")

# Extended usage examples

d <- data.frame(
  before = c(18, 20, 15, 22, 17, 19, 24, 16),
  after  = c(15, 18, 14, 19, 16, 17, 20, 15)
)

# One-sample signed-rank test against a specified median
signrank(before, data = d, mu = 18)

# Paired signed-rank test; the analyzed difference is before - after
signrank(before, after, data = d)
signrank(before, after, data = d, alternative = "greater")

# Approximation and confidence-interval controls
signrank(before, after, data = d, exact = FALSE, correct = FALSE)
signrank(before, after, data = d, conf.int = FALSE)

usedf(d)
signrank(before, after)
```

sum1

Quick numeric descriptive statistics

Description

Displays console descriptive statistics for one or more numeric variables. With nested grouping such as `by = c(sex, agegroup)`, results are first separated by sex and then summarized for each age group within sex.

Usage

```
sum1(
  ...,
  by = NULL,
```

```

data = NULL,
detail = FALSE,
digits = 2,
missing = c("ifany", "no", "always"),
overall = TRUE,
show = TRUE,
console = FALSE
)

```

Arguments

...	One or more numeric variables. An explicit data frame may be supplied as the first unnamed argument.
by	Optional grouping variables supplied as one bare name, <code>c(sex, agegroup)</code> , or <code>vars(sex, agegroup)</code> .
data	Optional data frame. When omitted, active data is used.
detail	Logical. If FALSE, reports N, missing, mean, SD, median, minimum, and maximum. If TRUE, also reports SE, variance, quartiles, IQR, range, coefficient of variation, skewness, and kurtosis.
digits	Decimal places.
missing	Missing handling for grouping variables: "no" excludes records missing any grouping variable; "ifany" and "always" retain observed missing strata. Missing numeric values are always excluded from calculations and counted in the Missing column.
overall	Logical; display the unstratified overall summary.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional result in the Console. Default FALSE.

Value

Invisibly returns an object of class `r4vn_quick`.

Examples

```

d <- data.frame(
  sex = factor(rep(c("Female", "Male"), each = 4)),
  agegroup = factor(rep(c("<40", "40+"), 4)),
  age = c(25, 42, 31, 55, 29, 48, 36, 61),
  bmi = c(20.1, 23.5, 21.7, 25.2, 22.4, 26.1, NA, 24.8)
)
usedf(d, quiet = TRUE)

sum1(age)
sum1(age, bmi)
sum1(age, by = sex)
sum1(age, bmi, by = c(sex, agegroup))
sum1(age, detail = TRUE)

```

```

usedf(clear = TRUE, quiet = TRUE)

# Extended usage examples

d <- data.frame(
  sex = factor(rep(c("Female", "Male"), each = 6)),
  agegroup = factor(rep(c("<40", "40+"), 6)),
  province = factor(rep(c("HCMC", "Other"), each = 3, times = 2)),
  age = c(25, 42, 31, 55, 29, 48, 36, 61, 33, 47, 52, 40),
  bmi = c(20.1, 23.5, 21.7, 25.2, 22.4, 26.1, NA, 24.8, 23.0, 27.1, 22.8, 24.2),
  sbp = c(110, 128, 118, 145, 121, 138, 125, 151, 130, 142, 136, 129)
)
usedf(d)

# One or several numeric variables
sum1(age)
sum1(age, bmi, sbp)

# One and several nested grouping variables
sum1(age, by = sex)
sum1(age, bmi, by = c(sex, agegroup))
sum1(age, bmi, by = c(province, sex, agegroup))

# Detailed statistics
sum1(age, bmi, detail = TRUE)

# Missing grouping strata, decimal places, and overall display
sum1(age, by = sex, missing = "no", digits = 1)
sum1(age, by = sex, missing = "ifany", overall = FALSE)

# Retain results without printing and use explicit data
result <- sum1(age, bmi, by = sex, show = FALSE)
result$variables[[1]]$overall
sum1(d, age, bmi, by = c(sex, agegroup))
sum1(age, bmi, data = d)

```

summary.r4vn_survey *Summarize an R4VN Survey Design*

Description

Summarize an R4VN Survey Design

Usage

```

## S3 method for class 'r4vn_survey'
summary(object, ...)

```

Arguments

object An object created by `surveyset()`.
 ... Additional arguments currently ignored.

Value

A data frame describing the survey design.

`summary.r4vn_tabsurvey`

Summarize a tabsurvey Result

Description

Summarize a tabsurvey Result

Usage

```
## S3 method for class 'r4vn_tabsurvey'
summary(object, ...)
```

Arguments

object Object returned by `tabsurvey()`.
 ... Additional arguments currently ignored.

Value

A list containing the publication table, design summary, tests, effects, diagnostics, optional interpretation, and notes.

`surv_tables`

Extract Tables from an R4VN Survival Result

Description

Converts every non-empty report component of a `tabsurv()` result to a named list of data frames, including descriptive, life-table, and optional interpretation tables. For a hierarchical result, tables are retained separately for every outer stratum. This keeps survival export compatible with the existing `tabexport()` data-frame workflow without changing `tabexport()` itself. Examples are self-contained.

Usage

```
surv_tables(x)
```

Arguments

x An object returned by tabsurv().

Value

A named list of data frames.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  d <- data.frame(time = c(5, 8, 10, 12, 15, 18),
                  event = c(1, 0, 1, 1, 0, 1),
                  group = factor(rep(c("A", "B"), 3)))
  s <- tabsurv(time, event, by = group, data = d, show = FALSE)
  z <- surv_tables(s)
  names(z)
}
```

survexport

Export an R4VN Survival Analysis

Description

Passes all non-empty tables from tabsurv() to the existing tabexport() function. It is also called internally when tabsurv(export=, file=) is used, so a complete Word/Excel/HTML report can be requested in one command.

Usage

```
survexport(
  x,
  export = NULL,
  file = NULL,
  open = FALSE,
  title = NULL,
  sheet = NULL,
  overwrite = TRUE,
  quiet = FALSE
)
```

Arguments

x An object returned by tabsurv().
 export, file, open, title, sheet, overwrite, quiet
 Passed to tabexport().

Value

The result returned by `tabexport()`.

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {
  d <- data.frame(time = c(5, 8, 10, 12, 15, 18),
                 event = c(1, 0, 1, 1, 0, 1),
                 group = factor(rep(c("A", "B"), 3)))
  s <- tabsurv(time, event, by = group, data = d, show = FALSE)
  # Export when a file is wanted, for example:
  # survexport(s, export = "xlsx", file = tempfile(fileext = ".xlsx"))
}
```

surveyset

Define a Complex Survey Design for R4VN

Description

Creates a reusable complex-survey design for `tabsurvey()` and future survey-aware R4VN analyses. Designs may include sampling weights, strata, one or more clustering stages, finite-population corrections, or replicate weights. More than one named survey design can be stored in the same R session, which is useful when one data set provides different weights for interviews, examinations, laboratory subsamples, household analyses, and other analytic components.

Usage

```
surveyset(
  data = NULL,
  name = "survey",
  weight = NULL,
  strata = NULL,
  cluster = NULL,
  fpc = NULL,
  repweights = NULL,
  rep_type = NULL,
  weightscale = c("relative", "population"),
  nest = TRUE,
  pps = FALSE,
  variance = NULL,
  combined.weights = TRUE,
  rho = NULL,
  mse = getOption("survey.replicates.mse"),
  lonely = c("adjust", "fail", "average", "certainty", "remove"),
  active = TRUE
)
```

Arguments

<code>data</code>	Optional data frame. If omitted, the active R4VN data frame is used.
<code>name</code>	Name used to store the survey design. The default is "survey". Use different names when the same data set requires different survey weights.
<code>weight</code>	Sampling/design/final survey weight. Supply one unquoted variable name or a one-element character vector. If omitted, equal weights are used.
<code>strata</code>	Optional stratum variable(s). Multiple stages may be supplied with <code>vars(...)</code> or <code>c(...)</code> .
<code>cluster</code>	Optional cluster/PSU variable(s). For multistage sampling, supply variables in sampling-stage order, for example <code>cluster = vars(psu, ssu)</code> .
<code>fpc</code>	Optional finite-population correction variable(s), in the same stage order as the cluster variables when applicable.
<code>repweights</code>	Optional replicate-weight variables, supplied with <code>vars(...)</code> , a character vector, a wildcard selector, or a column range such as <code>rep1:rep80</code> .
<code>rep_type</code>	Replicate design type passed to survey , such as "BRR", "Fay", "JK1", "JKn", or "bootstrap". Required when <code>repweights</code> is supplied.
<code>weightscale</code>	Meaning of the supplied weights. "relative" (default) means the weights are suitable for weighted estimates and design-based inference but their sum must not automatically be called a population total. "population" means the weights are expansion weights whose scale supports estimated population totals.
<code>nest</code>	Logical. Treat cluster identifiers as nested within strata. The default is TRUE, which is safe when PSU identifiers are reused in different strata.
<code>pps</code>	Optional PPS specification passed to <code>survey::svydesign()</code> . The default is FALSE. Advanced users may pass a supported survey PPS object or method.
<code>variance</code>	Optional PPS variance estimator passed to <code>survey::svydesign()</code> .
<code>combined.weights</code>	Logical argument used for replicate-weight designs.
<code>rho</code>	Optional Fay coefficient for appropriate replicate designs.
<code>mse</code>	Logical argument used for replicate-weight variance estimation.
<code>lonely</code>	Handling of strata containing a single PSU. Supported values are "adjust" (default), "fail", "average", "certainty", and "remove".
<code>active</code>	Logical. The named design is always stored under <code>name</code> . If TRUE (default), it also becomes the active R4VN survey design used when <code>tabsurvey()</code> is called without <code>design=</code> .

Details

Weight meaning is explicit. R4VN deliberately does not assume that `sum(weight)` is a population size. Many public-use surveys provide normalized or relative weights. Set `weightscale = "population"` only when documentation for the survey confirms that the weight has an expansion/population interpretation.

Multiple named designs. A single survey file may contain different weights for different analytic subsamples. Define each one separately, for example "interview" and "fasting", and select it in `tabsurvey(design = "fasting")`.

Survey weight versus other weights. The weight argument is intended for sampling/design/final survey weights. Propensity-score IPTW, frequency weights, analytic weights, and arbitrary regression weights are different concepts and should not be silently treated as survey sampling weights.

After changing the data. A survey design stores the data and design information that existed when `surveyset()` was called. If rows or variables are changed afterward, recreate the survey design so the design and analytic data remain aligned.

Value

An object of class `r4vn_survey`. The design is stored internally under `name`; when `active = TRUE` it also becomes the active survey design.

See Also

[tabsurvey](#), [vars](#), [usedf](#)

Other R4VN survey: [tabsurvey\(\)](#)

Examples

```
set.seed(2026)
n <- 600
d <- data.frame(
  psu = sample(1:60, n, TRUE),
  strata = sample(1:8, n, TRUE),
  wt = runif(n, 0.5, 2.5),
  age = rnorm(n, 45, 14),
  sex = factor(sample(c("Female", "Male"), n, TRUE)),
  hypertension = factor(sample(c("No", "Yes"), n, TRUE,
                                prob = c(.72, .28)))
)

usedf(d)
surveyset(weight = wt, strata = strata, cluster = psu)

# A second named design for a hypothetical laboratory subsample
d$labwt <- d$wt * runif(n, .8, 1.2)
surveyset(d, name = "lab", weight = labwt,
          strata = strata, cluster = psu, active = FALSE)

# Inspect the active design
summary(surveyset(d, weight = wt, strata = strata, cluster = psu))
```

swilk

Shapiro-Wilk normality test

Description

Runs Shapiro-Wilk tests overall, for many variables, and/or within hierarchical groups. Each tested subgroup must have enough complete observations for the Shapiro-Wilk procedure.

Usage

```
swilk(x = NULL,
      vars = NULL,
      by = NULL,
      data = NULL,
      digits = 4,
      p_digits = 3,
      show = TRUE,
      console = FALSE)
```

Arguments

- x One numeric variable; may be omitted when vars is supplied.
- vars Optional vars(...) selection of several numeric variables.
- by Optional hierarchical grouping specification. The final variable in vars(...) is the innermost group and preceding variables are strata.
- data Data frame; active R4VN data is used when omitted.
- digits, p_digits Formatting digits.
- show, console R4VN display controls.

Value

An R4VN result object, invisibly.

See Also

[normtest](#), [varform](#)

Examples

```
d <- data.frame(x = rnorm(40), y = rnorm(40), group = rep(c("A", "B"), each = 20))
swilk(x, data = d)
swilk(vars = vars(x, y), data = d)
swilk(vars = vars(x, y), by = vars(group), data = d)
```

tab	Create Descriptive, Comparative, and Regression Tables
-----	--

Description

Creates publication-style tables for descriptive analysis, group comparisons, binary-outcome regression, and continuous-outcome linear regression.

Usage

```
tab(  
  ...,  
  data = NULL,  
  vars = NULL,  
  by = NULL,  
  superby = NULL,  
  digit = 1,  
  p_digit = 3,  
  effect_digit = 2,  
  missing = "ifany",  
  row = FALSE,  
  col = TRUE,  
  cell = FALSE,  
  overall = "first",  
  descriptive = TRUE,  
  rvrow = NULL,  
  rvcol = FALSE,  
  test = TRUE,  
  pvalue = TRUE,  
  bold_p = TRUE,  
  p_bold = 0.05,  
  test_note = TRUE,  
  interaction = TRUE,  
  or = FALSE,  
  rr = FALSE,  
  pr = FALSE,  
  event = NULL,  
  adjusted = NULL,  
  multi = NULL,  
  effect_ref = NULL,  
  template = c("journal", "clean", "minimal"),  
  append = NULL,  
  file = NULL,  
  raw = FALSE,  
  name = FALSE,  
  title = NULL,  
  show = TRUE,  
  mode = c("auto", "console", "table")  
)  
  
tab(  
  ...,  
  data = NULL,  
  vars = NULL,  
  by = NULL,  
  superby = NULL,  
  digit = 1,
```

```

p_digit = 3,
effect_digit = 2,
missing = "ifany",
row = FALSE,
col = TRUE,
cell = FALSE,
overall = "first",
descriptive = TRUE,
rvrow = NULL,
rvcol = FALSE,
test = TRUE,
pvalue = TRUE,
bold_p = TRUE,
p_bold = 0.05,
test_note = TRUE,
interaction = TRUE,
or = FALSE,
rr = FALSE,
pr = FALSE,
event = NULL,
adjusted = NULL,
multi = NULL,
effect_ref = NULL,
template = c("journal", "clean", "minimal"),
append = NULL,
file = NULL,
raw = FALSE,
name = FALSE,
title = NULL,
show = TRUE,
mode = c("auto", "console", "table")
)

```

Arguments

...	In console mode, one row variable and optionally one column variable, followed by console options such as <code>exp</code> , <code>chi</code> , and <code>fisher</code> . In publication mode, legacy positional <code>data</code> , <code>vars</code> , and <code>by</code> arguments are also accepted.
<code>data</code>	Optional data frame. When omitted or <code>NULL</code> , the active data frame set by <code>usedf()</code> or <code>opendata(..., active = TRUE)</code> is used.
<code>vars</code>	A variable specification created by <code>vars()</code> .
<code>by</code>	Optional grouping or outcome variable supplied without quotation marks. Leave it empty for an overall descriptive table. Use a regular variable name for a categorical grouping/outcome variable, <code>c.outcome</code> for a continuous outcome summarized by mean (SD), or <code>q.outcome</code> for a continuous outcome summarized by median (IQR). R4VN also accepts the unified hierarchical form <code>by = vars(province, sex, outcome)</code> : <code>province</code> and <code>sex</code> are nested superby strata, in that order, and <code>outcome</code> is the innermost grouping/outcome variable.

<code>superby</code>	Backward-compatible single stratification variable. It may be combined with hierarchical <code>by = vars(...)</code> and then becomes the outermost stratum. New code should normally prefer the unified <code>by</code> convention.
<code>digit</code>	Number of decimal places for descriptive statistics.
<code>p_digit</code>	Number of decimal places for p-values.
<code>effect_digit</code>	Number of decimal places for OR, RR, PR, or linear regression coefficients.
<code>missing</code>	Missing-value display for categorical variables: "no", "ifany", or "always".
<code>row</code>	Logical. Calculate row percentages when <code>by</code> is categorical. When <code>row = TRUE</code> , <code>col</code> and <code>cell</code> are automatically set to <code>FALSE</code> .
<code>col</code>	Logical. Calculate column percentages when <code>by</code> is categorical. This is the default percentage mode.
<code>cell</code>	Logical. Calculate percentages using the complete table total. When <code>cell = TRUE</code> and <code>row = FALSE</code> , <code>row</code> and <code>col</code> are automatically set to <code>FALSE</code> . Thus users normally need to specify only <code>row = TRUE</code> , <code>cell = TRUE</code> , or neither for the default column percentages.
<code>overall</code>	Position of the overall column: "none", "first", or "last". Logical values are accepted for backward compatibility.
<code>descriptive</code>	Logical. Display descriptive-statistics columns.
<code>rvrow</code>	Categorical variables whose displayed level order should be reversed. Accepts <code>TRUE</code> , <code>vars(...)</code> , <code>c(...)</code> , a single variable name, or a character vector. This does not change model reference categories.
<code>rvcol</code>	Logical. Reverse displayed levels of a categorical <code>by</code> variable.
<code>test</code>	Logical. Display traditional omnibus-test p-values.
<code>pvalue</code>	Logical. Display separate p-value columns for model coefficients.
<code>bold_p</code>	Logical. Bold p-values smaller than <code>p_bold</code> .
<code>p_bold</code>	Significance threshold used when <code>bold_p = TRUE</code> .
<code>test_note</code>	Logical. Add superscript letters and footnotes identifying omnibus tests.
<code>interaction</code>	Logical. When <code>superby</code> is supplied, add one final interaction p-value column. Interaction tests use predictor-by-superby terms and follow <code>multi</code> , then adjusted, then crude models.
<code>or</code>	Logical. Calculate odds ratios using logistic regression.
<code>rr</code>	Logical. Calculate risk ratios using modified Poisson regression with robust variance.
<code>pr</code>	Logical. Calculate prevalence ratios using modified Poisson regression with robust variance.
<code>event</code>	Event level of a binary outcome. The last observed level is used when omitted.
<code>adjusted</code>	Variables included as adjustment covariates in separate models for each focal predictor. Prefer <code>vars(c.age, b2.sex, q.bmi)</code> so variable types and reference levels remain explicit. Also accepts <code>c(...)</code> , a character vector, <code>TRUE</code> , or "ALL".
<code>multi</code>	Variables included together in one final multivariable model. Prefer <code>vars(c.age, b2.sex, c.bmi)</code> . <code>TRUE</code> or "ALL" includes every variable listed in <code>vars</code> .

<code>effect_ref</code>	Optional backward-compatible reference categories. The <code>b2.</code> , <code>b3.</code> , and related prefixes take precedence.
<code>template</code>	HTML style: "journal", "clean", or "minimal".
<code>append</code>	Optional previous <code>r4vn_tab</code> object or existing HTML path.
<code>file</code>	Optional output HTML path. A temporary file is created when omitted.
<code>raw</code>	Logical. Retain unformatted results in the returned object.
<code>name</code>	Logical. Display original variable names beside variable labels.
<code>title</code>	Optional table title.
<code>show</code>	Logical. Display the HTML table in the RStudio Viewer or browser.
<code>mode</code>	Dispatch mode. "auto" selects publication mode when a <code>vars()</code> specification is supplied and otherwise selects console mode. Use "console" or "table" to force a mode.

Details

Prefixes used inside `vars()` determine descriptive summaries and categorical reference levels:

- no prefix: automatic typing; numeric/integer variables use mean and standard deviation, while factor/character/logical variables are categorical with the first observed level as reference;
- `b1.`, `b2.`, `b3.`, ...: force a categorical variable with the corresponding observed level as reference;
- `c.`: mean and standard deviation;
- `q.`: median and interquartile range;
- `f.`: mean, median, and range.

For grouped categorical tables, column percentages are the default. Setting `row = TRUE` automatically turns `col` and `cell` off; setting `cell = TRUE` automatically turns `row` and `col` off. Users therefore do not need to manually disable `col = TRUE`.

With a categorical by variable, categorical predictors are tested using Pearson's chi-squared test or Fisher's exact test. Variables declared with `c.` use a t-test or one-way ANOVA; variables declared with `q.` or `f.` use the Wilcoxon rank-sum or Kruskal-Wallis test.

Binary outcomes can be analyzed with OR, RR, or PR. OR uses logistic regression. RR and PR use modified Poisson regression with robust variance.

With `by = c.outcome`, the continuous outcome is summarized by mean (SD), categorical predictors use t-tests/ANOVA, and numeric predictors use Pearson correlation tests. With `by = q.outcome`, the outcome is summarized by median (IQR), categorical predictors use Wilcoxon/Kruskal-Wallis tests, and numeric predictors use Spearman tests. Both modes report unstandardized beta coefficients from linear regression.

`adjusted` and `multi` have different roles. `adjusted` fits a separate adjusted model for each focal predictor. `multi` fits one final model containing all specified variables.

When `superby` is supplied, `tab()` first calculates the complete dataset and then repeats the same analysis independently within every level of `superby`. The resulting blocks are combined side by side. When possible, one final interaction p-value column tests whether each predictor effect differs across the levels of `superby`.

Value

Invisibly returns an object of class `r4vn_tab`. Important components include `data`, `file`, `html`, `table_html`, `rows`, `multi_model`, and `multi_diagnostics`.

Common call patterns

```
tab(data, vars = vars(...))
tab(data, vars = vars(...), by = group)
tab(data, vars = vars(...), by = outcome, or = TRUE)
tab(data, vars = vars(...), by = c.outcome)
tab(data, vars = vars(...), by = q.outcome)
tab(data, vars = vars(...), by = outcome, superby = subgroup, or = TRUE)
```

See Also

[vars](#), [tabmulti](#), and [tabexport](#).

Other R4VN tables: [tabexport\(\)](#), [tabforest\(\)](#), [tablong\(\)](#), [tabmeta\(\)](#), [tabmulti\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [tabsurvey\(\)](#), [vars\(\)](#)

Examples

```
set.seed(2026)
n <- 180
dat <- data.frame(
  age = round(rnorm(n, 45, 12)),
  sex = factor(sample(c("Female", "Male"), n, TRUE)),
  bmi = round(rnorm(n, 23, 3), 1),
  smoking = factor(sample(c("No", "Yes"), n, TRUE,
    prob = c(0.70, 0.30))),
  education = factor(sample(c("Primary", "Secondary", "College"),
    n, TRUE))
)
dat$sbp <- round(80 + 0.75 * dat$age + 1.1 * dat$bmi +
  5 * (dat$sex == "Male") +
  4 * (dat$smoking == "Yes") + rnorm(n, 0, 12), 1)
lp <- -3.2 + 0.045 * dat$age + 0.10 * (dat$bmi - 23) +
  0.45 * (dat$sex == "Male") + 0.65 * (dat$smoking == "Yes")
dat$hypertension <- factor(
  rbinom(n, 1, plogis(lp)),
  levels = c(0, 1), labels = c("No", "Yes")
)

tb0 <- tab(dat, vars = vars(c.age, b2.sex, q.bmi, b2.smoking, education),
  show = FALSE)
head(tb0$data)

tb1 <- tab(dat, vars = vars(c.age, b2.sex, c.bmi, b2.smoking, b2.education),
  by = hypertension, or = TRUE, event = "Yes",
  multi = vars(c.age, b2.sex, c.bmi, b2.smoking), show = FALSE)

tb2 <- tab(dat, vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
```

```

      by = c.sbp, multi = vars(c.age, b2.sex, c.bmi, b2.smoking),
      show = FALSE)

tb3 <- tab(dat, vars = vars(c.age, c.bmi, b2.smoking, b2.education),
      by = hypertension, superby = sex, overall = "none",
      or = TRUE, event = "Yes",
      multi = vars(c.age, c.bmi, b2.smoking), show = FALSE)

# Extended usage examples

set.seed(2026)
n <- 300
d <- data.frame(
  sex = factor(sample(c("Female", "Male"), n, TRUE)),
  age = rnorm(n, 45, 12),
  bmi = rnorm(n, 23, 3),
  smoking = factor(sample(c("No", "Yes"), n, TRUE)),
  region = factor(sample(c("Urban", "Rural"), n, TRUE)),
  outcome = factor(rbinom(n, 1, .3), levels = 0:1, labels = c("No", "Yes")),
  sbp = rnorm(n, 125, 18)
)

# Overall descriptive table. Numeric variables without a prefix are
# automatically summarized with mean (SD); factors remain categorical.
t1_auto <- tab(d, vars = vars(age, sex, bmi, smoking), show = FALSE)

# Explicit q. remains available when median (IQR) is preferred.
t1 <- tab(d, vars = vars(sex, age, q.bmi, smoking), show = FALSE)

# Compare groups, show overall first, tests, and missing values when present
t2 <- tab(d, vars = vars(sex, c.age, q.bmi, smoking), by = outcome,
  overall = "first", test = TRUE, missing = "ifany", show = FALSE)

# Row, column, or cell percentages for categorical variables
tab(d, vars = vars(sex, smoking), by = outcome, row = TRUE, show = FALSE)
tab(d, vars = vars(sex, smoking), by = outcome, show = FALSE)
tab(d, vars = vars(sex, smoking), by = outcome, cell = TRUE, show = FALSE)

# Reverse selected row levels or the by-variable columns
tab(d, vars = vars(sex, smoking), by = outcome,
  rvrow = vars(smoking), rvcol = TRUE, show = FALSE)

# Crude odds ratios for a binary outcome
tab(d, vars = vars(sex, c.age, smoking), by = outcome,
  or = TRUE, event = "Yes", show = FALSE)

# Risk ratios or prevalence ratios using modified Poisson models
tab(d, vars = vars(sex, c.age, smoking), by = outcome,
  rr = TRUE, event = "Yes", show = FALSE)
tab(d, vars = vars(sex, c.age, smoking), by = outcome,
  pr = TRUE, event = "Yes", show = FALSE)

# Separate adjusted models for every focal predictor

```

```

tab(d, vars = vars(sex, c.age, smoking), by = outcome,
    or = TRUE, adjusted = vars(age, sex), event = "Yes", show = FALSE)

# One final multivariable model; effects are placed beside their variables
tab(d, vars = vars(sex, c.age, smoking, q.bmi), by = outcome,
    or = TRUE, multi = vars(sex, age, smoking), event = "Yes", show = FALSE)

# Hide descriptive columns and show only model results
tab(d, vars = vars(sex, c.age, smoking), by = outcome,
    descriptive = FALSE, or = TRUE, multi = TRUE,
    event = "Yes", show = FALSE)

# Continuous outcome: c. gives parametric methods and beta coefficients
tab(d, vars = vars(sex, c.age, smoking, q.bmi), by = c.sbp,
    adjusted = vars(age, sex), multi = vars(age, sex, bmi), show = FALSE)

# Continuous outcome: q. gives rank-based descriptive comparisons
tab(d, vars = vars(sex, c.age, smoking, q.bmi), by = q.sbp,
    test = TRUE, show = FALSE)

# Supergroup columns plus interaction
tab(d, vars = vars(sex, c.age, smoking), by = outcome, superby = region,
    interaction = TRUE, overall = "first", show = FALSE)

# Templates, titles, raw numerical output, and named variables
tab(d, vars = vars(sex, c.age, smoking), by = outcome,
    template = "minimal", title = "Participant characteristics",
    raw = TRUE, name = TRUE, show = FALSE)

```

tab1

Quick one-way frequency tables

Description

Displays formatted frequency tables for one or more categorical variables in the Viewer, with optional Console output. Data may be supplied explicitly, but when `data = NULL` the active data set selected by `usedf()` is used.

Usage

```

tab1(
  ...,
  by = NULL,
  data = NULL,
  missing = c("ifany", "no", "always"),
  percent = "column",
  digits = 1,
  overall = TRUE,

```

```

    drop = TRUE,
    show = TRUE,
    console = FALSE
  )

```

Arguments

...	One or more variables. An explicit data frame may be supplied as the first unnamed argument for backward-compatible R4VN syntax.
by	Optional grouping variables supplied as one bare name, <code>c(sex, agegroup)</code> , or <code>vars(sex, agegroup)</code> .
data	Optional data frame. When omitted, active data is used.
missing	Missing-value display: "no", "ifany", or "always". For grouping variables, "ifany" and "always" retain observed missing strata.
percent	Percentage denominator: "column" or "within" for the terminal stratum, "row" for each response level across strata, "cell" or "overall" for all eligible observations, or "none".
digits	Decimal places for percentages.
overall	Logical; display the unstratified overall distribution.
drop	Logical; omit unused factor levels.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional result in the Console. Default FALSE.

Details

`by` accepts one or more nested stratification variables. For example, `by = c(sex, agegroup)` first separates results by sex and then displays age-group-specific results inside each sex group. Three or more nested stratification variables are also supported.

Value

Invisibly returns an object of class `r4vn_quick`.

Examples

```

d <- data.frame(
  sex = factor(c("Female", "Male", "Female", "Male", "Female", "Male")),
  agegroup = factor(c("<40", "<40", "40+", "40+", "<40", "40+")),
  smoking = factor(c("No", "Yes", "No", "Yes", NA, "No")),
  vaccinated = factor(c("Yes", "No", "Yes", "Yes", "No", "No"))
)
usedf(d, quiet = TRUE)

tab1(smoking)
tab1(smoking, vaccinated)
tab1(smoking, by = sex)
tab1(smoking, vaccinated, by = c(sex, agegroup))
tab1(smoking, by = c(sex, agegroup), missing = "always")

```

```

usedf(clear = TRUE, quiet = TRUE)

# Extended usage examples

d <- data.frame(
  sex = factor(c("Female", "Male", "Female", "Male", "Female", "Male")),
  agegroup = factor(c("<40", "<40", "40+", "40+", "<40", "40+")),
  province = factor(c("HCMC", "HCMC", "HCMC", "Other", "Other", "Other")),
  smoking = factor(c("No", "Yes", "No", "Yes", NA, "No")),
  vaccinated = factor(c("Yes", "No", "Yes", "Yes", "No", "No"))
)
usedf(d)

# One or several categorical variables
tab1(smoking)
tab1(smoking, vaccinated)

# One level of stratification
tab1(smoking, by = sex)

# Nested stratification: age groups are shown within each sex
tab1(smoking, by = c(sex, agegroup))

# Three nested levels, evaluated in the supplied order
tab1(smoking, vaccinated, by = c(province, sex, agegroup))

# Percentage denominators
tab1(smoking, by = c(sex, agegroup), percent = "column")
tab1(smoking, by = c(sex, agegroup), percent = "row")
tab1(smoking, by = c(sex, agegroup), percent = "cell")
tab1(smoking, by = c(sex, agegroup), percent = "none")

# Missing values and unused factor levels
tab1(smoking, missing = "no")
tab1(smoking, missing = "ifany")
tab1(smoking, missing = "always", drop = FALSE)

# Hide the overall section or retain the result without printing
tab1(smoking, by = sex, overall = FALSE)
result <- tab1(smoking, by = sex, show = FALSE)
result$variables[[1]]$strata

# Explicit data remains supported
tab1(d, smoking, vaccinated, by = c(sex, agegroup))
tab1(smoking, data = d)

```

Description

tabdiag() is the comprehensive diagnostic-accuracy command in R4VN. It is designed for binary diagnostic tests, continuous or ordinal biomarkers, prediction scores, and comparisons of paired ROC curves. The first variable is the binary reference standard (gold standard); diagnostic tests can be supplied directly in ... and/or through vars().

The default output follows the R4VN philosophy: a compact publication-ready Viewer table is shown, while the returned object retains the complete set of diagnostic measures and ROC information for later inspection or export. Interpretation is opt-in (interpretation = FALSE by default).

Usage

```
tabdiag(  
  outcome,  
  ...,  
  data = NULL,  
  vars = NULL,  
  event = NULL,  
  positive = NULL,  
  direction = c("auto", "<", ">"),  
  best = "youden",  
  cuts = NULL,  
  target = 0.95,  
  ci = TRUE,  
  ci_level = 0.95,  
  ci_method = c("auto", "wilson", "exact"),  
  cut_ci = FALSE,  
  boot = 2000,  
  prevalence = NULL,  
  roc = TRUE,  
  partial_auc = NULL,  
  partial_focus = c("specificity", "sensitivity"),  
  partial_correct = FALSE,  
  compare = FALSE,  
  compare_method = c("delong", "bootstrap"),  
  adjust = "none",  
  all_cuts = FALSE,  
  missing = FALSE,  
  show = TRUE,  
  measures = "core",  
  hide = NULL,  
  plot = FALSE,  
  plot_args = list(),  
  interpretation = FALSE,  
  digit = 2,  
  p_digit = 3,  
  zero_correction = 0.5,  
  title = NULL,  
  console = FALSE,
```

```

    print = NULL,
    export = NULL,
    file = NULL,
    open = FALSE
  )

```

Arguments

outcome	Binary reference-standard outcome. Supply an unquoted variable name or a single character variable name.
...	One or more diagnostic tests/markers. Binary variables are analyzed as 2 x 2 tests. Numeric variables, dates/times, and ordered factors are analyzed by ROC.
data	Optional data frame. If omitted, the active R4VN data selected by <code>usedf()</code> or <code>opendata(..., active = TRUE)</code> is used.
vars	Optional diagnostic-marker selection created by <code>vars()</code> . This may be used instead of, or together with, markers supplied in ...
event	Positive level of the reference-standard outcome. If omitted, R4VN recognizes common positive encodings such as 1, TRUE, Yes, Positive, Case, or Co; otherwise it uses the second factor/observed level and reports the selected positive outcome explicitly.
positive	Positive level for binary diagnostic tests. It may be a single value used for all binary tests, an unnamed vector in test order, or a named vector such as <code>positive = c(rapid = "Positive", ct = "Abnormal")</code> .
direction	ROC direction: "auto" (default), "<", or ">". With R4VN conventions, "<" means higher marker values indicate the positive outcome and R4VN reports cutoffs as $\geq$; ">" means lower values indicate the positive outcome and R4VN reports cutoffs as $\leq$.
best	Optimal-cutoff methods. Default "youden". May contain one or more of "youden", "closest", "ruleout", and "rulein". "ruleout" finds the threshold with the highest specificity while sensitivity is at least target; "rulein" finds the threshold with the highest sensitivity while specificity is at least target. Set <code>best = NULL</code> to suppress automatically selected cutoffs.
cuts	Optional numeric cutoff(s) requested by the user. Every cutoff becomes a separate diagnostic-performance row for each quantitative marker. This is useful for clinically predefined thresholds.
target	Target sensitivity/specificity for "ruleout"/"rulein", default 0.95.
ci	TRUE (default), FALSE, or a character vector of measures for which confidence intervals are required. Examples include <code>c("sens", "spec")</code> , <code>c("sens", "spec", "ppv", "npv")</code> , and <code>c("auc", "lr+", "lr-", "dor")</code> . TRUE requests every currently implemented stable interval.
ci_level	Confidence level, default 0.95.
ci_method	Binomial confidence-interval method for 2 x 2 proportions: "auto"/"wilson" or "exact". AUC uses an internally implemented DeLong CI for an ordinary empirical ROC and stratified bootstrap CI for partial AUC.

cut_ci	Logical; calculate a bootstrap confidence interval for an automatically selected Youden/closest cutoff. Default FALSE because this can be computationally expensive.
boot	Number of bootstrap replicates for cutoff CI, partial-AUC CI, and bootstrap ROC comparison. Default 2000.
prevalence	Optional target disease prevalence, strictly between 0 and 1. When supplied, PPV and NPV are recalculated for that prevalence.
roc	Logical; calculate ROC analysis for eligible non-binary tests. Default TRUE.
partial_auc	Optional numeric vector of length two defining a partial AUC range, for example <code>c(1, 0.80)</code> for specificity 100%-80% when <code>partial_focus = "specificity"</code> .
partial_focus	"specificity" (default) or "sensitivity".
partial_correct	Logical; request corrected partial AUC where supported by the R4VN internal ROC engine.
compare	Logical; for two or more ROC-eligible markers, calculate pairwise comparisons of AUC. Default FALSE.
compare_method	ROC comparison method: "delong" (default) or "bootstrap". Both are implemented internally by R4VN.
adjust	Multiplicity adjustment for pairwise comparison p-values, passed to <code>stats::p.adjust()</code> . Examples: "none", "holm", "BH".
all_cuts	Logical; if TRUE, calculate full 2 x 2 diagnostic properties for every finite empirical ROC threshold and store them in <code>\$all_cutoffs</code> . FALSE by default because this table can be very large.
missing	Logical; include the marker-specific analysis sample table in printed/Viewer output. The sample table is always retained in <code>\$descriptive</code> . Default FALSE.
show	Logical; open the formatted result in the Viewer. Default TRUE. For backward compatibility, a character vector supplied to <code>show</code> is interpreted as the old diagnostic-column selection syntax.
measures	Diagnostic measures displayed in the selected-threshold table. Default "core" gives N, sensitivity, specificity, PPV, NPV, LR+, LR-, and accuracy. Other convenient profiles are "minimal", "publication", "counts", and "all"; or supply an explicit vector such as <code>c("sens", "spec", "ppv", "npv", "lr+", "lr-", "dor")</code> . Cell counts are presented as a conventional 2 x 2 diagnostic classification table rather than as separate TP/FP/FN/TN measures. The returned object always retains all calculated measures.
hide	Optional diagnostic columns to remove from the displayed cutoff table, for example <code>hide = c("n", "accuracy")</code> .
plot	FALSE (default), TRUE, "roc", "cutoff", or "both". TRUE is equivalent to "roc". Requested figures are saved internally, embedded in the Viewer report, and also sent to the RStudio Plot pane when running interactively.
plot_args	Named list of graphical arguments passed to <code>plot.r4vn_diag()</code> . Useful options include <code>color</code> , <code>lty</code> , <code>line_width</code> , <code>legend</code> , <code>legend_position</code> , <code>auc</code> , <code>diagonal</code> , <code>grid</code> , <code>xlim</code> , <code>ylim</code> , <code>xlab</code> , <code>ylab</code> , <code>main</code> , and <code>cutoff-plot</code> controls. ROC axes are constrained to 0-1 by default.

interpretation	Logical; add a cautious deterministic interpretation table. Default FALSE. Interpretation describes discrimination and selected thresholds but does not claim clinical utility or causality.
digit	Number of digits displayed for estimates. Default 2.
p_digit	Number of digits displayed for p-values. Default 3.
zero_correction	Continuity correction used only for a requested DOR confidence interval when a zero cell is present. Default 0.5.
title	Optional title printed above the output.
console	Logical; also print the traditional Console result. Default FALSE.
print	Deprecated compatibility argument. When supplied, it overrides console.
export	Optional export format accepted by <code>tabexport()</code> , such as "html", "docx", "xlsx", "pdf", or "png". Multiple formats may be requested in one call.
file	Optional export filename. Its extension may also determine the export format.
open	Logical; open the exported file when supported.

Details

Binary diagnostic tests:

A test with exactly two observed non-missing values is analyzed directly as a 2 x 2 table. `tabdiag()` and `tabdiagi()` use the same internal engine, so identical TP/FP/FN/TN counts give identical sensitivity, specificity, predictive values, likelihood ratios, DOR, accuracy, balanced accuracy, Youden index, F1 score, MCC, kappa, FPR, FNR, FDR, FOR, prevalence, detection rate, and detection prevalence.

Quantitative and ordered diagnostic markers:

ROC analysis is implemented inside R4VN using base R. `tabdiag()` therefore does not require `pROC`, `ggplot2`, or another ROC package for routine ROC, AUC, cutoff selection, DeLong inference, partial AUC, or paired AUC comparison. This keeps installation light while preserving a complete analysis object.

Numeric markers, Date/POSIX values, and ordered factors with more than two levels are analyzed by ROC. Unordered factors with more than two levels are rejected because a clinically meaningful ordering cannot safely be inferred. A marker that becomes constant after removal of missing values is retained in the sample description and omitted from ROC analysis with an explanatory note instead of crashing the whole report.

Cutoffs:

`best = "youden"` maximizes sensitivity + specificity. `"closest"` minimizes distance to the upper-left ROC corner. `"ruleout"` and `"rulein"` choose thresholds meeting a target sensitivity or specificity. Tied optimal thresholds are deliberately retained rather than silently choosing one. User-defined cuts are added as separate rows rather than replacing the automatic cutoff.

Confidence intervals:

Sensitivity, specificity, PPV, NPV, and accuracy use Wilson or exact binomial intervals. LR+/LR- and DOR use conventional log-scale intervals. Ordinary AUC uses DeLong CI. Partial-AUC CI and optimal-cutoff CI use stratified bootstrap. If prevalence-adjusted PPV/NPV are requested, simple binomial CIs are not attached to those adjusted predictive values because they would not correspond to the target-prevalence estimates.

AUC inference and paired ROC comparison:

For each ordinary empirical ROC, R4VN stores DeLong AUC variance, standard error, z statistic, and a two-sided large-sample p-value for $H_0: AUC = 0.5$. When `compare = TRUE`, every pair of eligible markers is compared on its own pairwise complete-case sample. This keeps the comparison paired and makes the reported N explicit when marker missingness differs.

Missing values:

Each marker is analyzed on its own complete cases with the outcome. Pairwise ROC comparison uses complete cases for both markers and the outcome. Therefore N may differ between marker-specific AUCs and pairwise comparisons. Use `missing = TRUE` to show the sample accounting table; it is always available as `$descriptive`.

Returned result contract:

Backward-compatible components (`$summary`, `$thresholds`, `$coordinates`, `$comparison`, `$all_cutoffs`, `$roc`) are retained. In addition, the object follows the common R4VN reporting contract with `$descriptive`, `$estimates`, `$tests`, `$diagnostics`, `$interpretation`, `$tables`, `$plots`, `$models`, `$metadata`, and `$call`.

Publication-oriented display:

The default `measures = "core"` keeps the Viewer readable. Use `measures = "publication"` to add DOR, `measures = "all"` for the full diagnostic panel, or `measures = "counts"` for the 2 x 2 diagnostic classification table only. Cell counts are never shown as a vertical TP/FP/FN/TN measure list. Calculations are never discarded by this display choice. When a full performance table contains only a few rows but many columns, the Viewer automatically presents measures vertically for easier reading.

ROC plotting:

ROC plots are drawn from false-positive rate (1-specificity) and sensitivity coordinates using ordinary base graphics. The default axes are exactly 0 to 1 (`xaxs = "i"`, `yaxs = "i"`), preventing the small negative/>1 axis extensions that can otherwise appear in automatic plotting systems.

Value

An object of class `r4vn_diag`. Important components include:

- `$summary`: marker-specific AUC results.
- `$thresholds`: full diagnostic metrics at binary tests and selected cutoffs.
- `$comparison`: pairwise AUC comparisons.
- `$descriptive`: marker-specific N/missing/case/control accounting.
- `$estimates`: structured AUC and threshold estimates.
- `$tests`: AUC-vs-0.5 and pairwise ROC tests.
- `$diagnostics`: ROC coordinates and optional all-cutoff table.
- `$tables`: publication-ready display tables used by Viewer/export, including `$tables$Confusion_matrix` for selected tests/cutoffs.
- `$interpretation`: NULL by default, or an interpretation table when requested.
- `$roc` / `$models$roc`: lightweight internal R4VN ROC objects.

Examples - 1. Simplest binary test

```
d_bin <- data.frame(
  disease = c(1,1,1,1,0,0,0,0),
  rapid    = c(1,1,1,0,1,0,0,0)
)
tabdiag(disease, rapid, data = d_bin)
```

Examples - 2. Explicit positive outcome/test level

```
d_txt <- data.frame(
  truth = factor(c("No", "Yes", "Yes", "No", "Yes", "No")),
  test  = factor(c("Neg", "Pos", "Pos", "Neg", "Neg", "Pos"))
)
tabdiag(truth, test, data = d_txt, event = "Yes", positive = "Pos")
```

Examples - 3. Several binary tests with named positive levels

```
d_multi_bin <- data.frame(
  disease = c(1,1,1,0,0,0,1,0),
  rapid = c("Positive", "Positive", "Negative", "Negative", "Positive", "Negative", "Positive", "Negative"),
  ct = c("Abnormal", "Normal", "Abnormal", "Normal", "Normal", "Normal", "Abnormal", "Normal")
)
tabdiag(
  disease, rapid, ct, data = d_multi_bin,
  positive = c(rapid = "Positive", ct = "Abnormal")
)
```

Examples - 4. One continuous biomarker

```
set.seed(101)
d <- data.frame(
  disease = rep(c(0,1), each = 100),
  crp = c(rnorm(100, 5, 2), rnorm(100, 10, 3))
)
z <- tabdiag(disease, crp, data = d, show = FALSE)
z$summary
z$thresholds
```

Examples - 5. Select markers with vars()

```
tabdiag(disease, data = d, vars = vars(crp), show = FALSE)
```

Examples - 6. Automatic and clinically specified cutoffs

```
tabdiag(
  disease, crp, data = d,
  best = "youden", cuts = c(5, 7.5, 10), show = FALSE
)
```

Examples - 7. Several optimal-cutoff definitions

```
tabdiag(  
  disease, crp, data = d,  
  best = c("youden", "closest", "ruleout", "rulein"),  
  target = 0.90, show = FALSE  
)
```

Examples - 8. Disable automatic cutoffs and use only clinical cutoffs

```
tabdiag(disease, crp, data = d, best = NULL, cuts = c(6, 8, 10), show = FALSE)
```

Examples - 9. Confidence-interval control

```
tabdiag(disease, crp, data = d, ci = c("auc", "sens", "spec"), show = FALSE)  
tabdiag(disease, crp, data = d, ci = TRUE, show = FALSE)  
tabdiag(disease, crp, data = d, ci = FALSE, show = FALSE)
```

Examples - 10. Exact binomial CI for diagnostic proportions

```
tabdiag(disease, rapid, data = d_bin, ci = TRUE, ci_method = "exact", show = FALSE)
```

Examples - 11. Bootstrap CI for the optimal cutoff

```
tabdiag(  
  disease, crp, data = d, cut_ci = TRUE,  
  boot = 200, show = FALSE  
)
```

Examples - 12. Predictive values at a target prevalence

```
tabdiag(disease, crp, data = d, prevalence = 0.10, show = FALSE)
```

Examples - 13. Pairwise DeLong comparison

```
set.seed(102)  
d$pct <- c(rnorm(100, 1, 0.8), rnorm(100, 3, 1.2))  
tabdiag(disease, crp, pct, data = d, compare = TRUE, show = FALSE)
```

Examples - 14. Multiple-comparison adjustment

```
d$score3 <- d$crp + rnorm(nrow(d), 0, 2)  
tabdiag(  
  disease, crp, pct, score3, data = d,  
  compare = TRUE, adjust = "holm", show = FALSE  
)
```

Examples - 15. Bootstrap ROC comparison

```
tabdiag(
  disease, crp, pct, data = d,
  compare = TRUE, compare_method = "bootstrap", boot = 200,
  show = FALSE
)
```

Examples - 16. Partial AUC

```
tabdiag(
  disease, crp, data = d,
  partial_auc = c(1, 0.80), partial_focus = "specificity",
  partial_correct = TRUE, show = FALSE
)
```

Examples - 17. Ordered diagnostic score

```
d_ord <- data.frame(
  disease = c(0,0,0,0,1,1,1,1,1,0),
  score = ordered(c("Low","Low","Medium","Low","Medium","High","High","Medium","High","Medium"),
    levels = c("Low","Medium","High"))
)
tabdiag(disease, score, data = d_ord, show = FALSE)
```

Examples - 18. Lower values indicate disease

```
d$low_marker <- -d$crp
tabdiag(disease, low_marker, data = d, direction = ">", show = FALSE)
```

Examples - 19. Marker-specific missing values

```
d_missing <- d
d_missing$crp[1:12] <- NA
d_missing$pct[21:35] <- NA
zmis <- tabdiag(
  disease, crp, pct, data = d_missing,
  compare = TRUE, missing = TRUE, show = FALSE
)
zmis$descriptive
zmis$comparison
```

Examples - 20. Compact, publication, counts, and full displays

```
tabdiag(disease, crp, data = d, measures = "minimal", show = FALSE)
tabdiag(disease, crp, data = d, measures = "publication", show = FALSE)
tabdiag(disease, crp, data = d, measures = "counts", show = FALSE)
tabdiag(disease, crp, data = d, measures = "all", show = FALSE)
```

Examples - 21. Explicit measure selection and hiding columns

```
tabdiag(
  disease, crp, data = d,
  measures = c("sens", "spec", "ppv", "npv", "lr+", "lr-", "dor"),
  hide = "npv", show = FALSE
)
```

Examples - 22. Full metrics at every empirical ROC threshold

```
zall <- tabdiag(disease, crp, data = d, all_cuts = TRUE, show = FALSE)
head(zall$all_cutoffs)
```

Examples - 23. Interpretation is explicit opt-in

```
zint <- tabdiag(disease, crp, data = d, interpretation = TRUE, show = FALSE)
zint$interpretation
```

Examples - 24. ROC graph with publication controls

```
tabdiag(
  disease, crp, pct, data = d, plot = "roc",
  plot_args = list(
    line_width = 2, diagonal = TRUE, grid = TRUE,
    legend_position = "bottomright",
    xlab = "1 - Specificity", ylab = "Sensitivity",
    main = "ROC curves"
  )
)
```

Examples - 25. Sensitivity/specificity across cutoffs

```
tabdiag(
  disease, crp, data = d, plot = "cutoff",
  plot_args = list(cutoff_mark = TRUE)
)
```

Examples - 26. Draw both graph families from a saved object

```
z <- tabdiag(disease, crp, pct, data = d, plot = FALSE, show = FALSE)
plot(z, what = "roc")
plot(z, what = "cutoff")
```

Examples - 27. Active R4VN data

```
\donttest{
  usedf(d)
  tabdiag(disease, crp, show = FALSE)
}
```

Examples - 28. Export publication tables

```
\donttest{
  tabdiag(
    disease, crp, data = d, show = FALSE,
    export = "xlsx", file = tempfile(fileext = ".xlsx")
  )
}
```

Examples - 29. Inspect the stable R4VN result contract

```
z <- tabdiag(disease, crp, data = d, show = FALSE)
names(z$tables)
z$estimates$auc
z$tests$auc_vs_0.5
z$diagnostics$coordinates
z$tables$Confusion_matrix
z$metadata
```

Examples - 30. Full confusion matrix and full performance panel

```
zfull <- tabdiag(
  disease, crp, data = d,
  measures = "all", missing = TRUE, show = FALSE
)
zfull$tables$Diagnostic_performance
zfull$tables$Confusion_matrix
```

Examples - 31. Save publication-ready ROC and cutoff plots

```
\donttest{
  zplot <- tabdiag(disease, crp, data = d, show = FALSE)
  plot(
    zplot, what = "roc",
    file = tempfile(fileext = ".png"),
    width = 7, height = 7, res = 300,
    line_width = 2, grid = TRUE
  )
  plot(
    zplot, what = "cutoff",
    file = tempfile(fileext = ".pdf"),
    width = 7, height = 7, cutoff_mark = TRUE
  )
}
```

Examples

```
set.seed(123)
dd <- data.frame(
  disease = rep(c(0, 1), each = 40),
```

```

marker = c(rnorm(40, 0, 1), rnorm(40, 1.5, 1))
)
z <- tabdiag(disease, marker, data = dd, show = FALSE)
z$summary

```

tabdiagi

Immediate diagnostic accuracy from a 2 x 2 table

Description

tabdiagi() calculates diagnostic test accuracy directly from the four cell counts of a 2 x 2 table. It is intended especially for teaching, checking hand calculations, and analyses where only aggregated counts are available.

Usage

```

tabdiagi(
  tp = NULL,
  fp = NULL,
  fn = NULL,
  tn = NULL,
  a = NULL,
  b = NULL,
  c = NULL,
  d = NULL,
  prevalence = NULL,
  ci = FALSE,
  ci_level = 0.95,
  ci_method = c("auto", "wilson", "exact"),
  zero_correction = 0.5,
  digit = 2,
  show = TRUE,
  console = FALSE
)

```

Arguments

tp	True positives. With positional input this is the first number.
fp	False positives. With positional input this is the second number.
fn	False negatives. With positional input this is the third number.
tn	True negatives. With positional input this is the fourth number.
a, b, c, d	Optional aliases for tp, fp, fn, and tn, respectively. Use either the TP/FP/FN/TN names or the a/b/c/d names, not both.
prevalence	Optional population disease prevalence, strictly between 0 and 1. When supplied, PPV and NPV are recalculated from sensitivity, specificity, and this prevalence. The observed predictive values remain available in the returned object.

<code>ci</code>	FALSE (default), TRUE, or a character vector specifying which confidence intervals to calculate. Examples include <code>ci = c("sens", "spec")</code> , <code>ci = c("ppv", "npv", "lr+", "lr-", "dor")</code> , or <code>ci = TRUE</code> .
<code>ci_level</code>	Confidence level, default 0.95.
<code>ci_method</code>	Method for binomial proportion confidence intervals: "auto" or "wilson" uses Wilson intervals; "exact" uses exact binomial intervals. LR+/LR- and DOR use log-scale intervals.
<code>zero_correction</code>	Continuity correction used only when a zero cell prevents a finite log-scale DOR confidence interval. Default 0.5.
<code>digit</code>	Number of digits displayed for estimates.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the teaching-oriented result in the Console. Default FALSE.

Details

The cell layout is:

	Reference standard	
	Positive	Negative
Test positive	TP	FP
Test negative	FN	TN

The function calculates the full set of commonly used 2 x 2 diagnostic measures: sensitivity, specificity, PPV, NPV, accuracy, balanced accuracy, LR+, LR-, diagnostic odds ratio (DOR), Youden index, F1 score, Matthews correlation coefficient (MCC), Cohen's kappa, false-positive rate (FPR), false-negative rate (FNR), false-discovery rate (FDR), false-omission rate (FOR), disease prevalence, detection rate, and detection prevalence.

Confidence intervals are optional because they can make console output much wider. `ci = TRUE` requests all currently supported intervals. Sensitivity, specificity, PPV, NPV, and accuracy use binomial intervals. LR+ and LR- use conventional log-scale intervals. DOR uses a log-scale interval; if any cell is zero, the requested `zero_correction` is applied for the DOR interval calculation and this is recorded in the returned object.

When prevalence is supplied, PPV and NPV in the main results are prevalence-adjusted. Because simple binomial confidence intervals no longer apply to these adjusted predictive values, their CI entries are returned as missing. The observed PPV and NPV remain in `result$estimates` as `ppv_observed` and `npv_observed`.

Value

Invisibly returns an object of class `r4vn_diagi` and `r4vn_stat` containing table, estimates, ci, settings, and notes.

Typical teaching use

Use four counts directly:

```
tabdiagi(80, 20, 10, 90)
```

or use explicit names:

```
tabdiagi(tp = 80, fp = 20, fn = 10, tn = 90)
```

Confidence intervals

Request only sensitivity and specificity confidence intervals:

```
tabdiagi(80, 20, 10, 90, ci = c("sens", "spec"))
```

Request all supported confidence intervals:

```
tabdiagi(80, 20, 10, 90, ci = TRUE)
```

Predictive values at a target prevalence

To show how PPV and NPV change when disease prevalence is 10 percent:

```
tabdiagi(80, 20, 10, 90, prevalence = 0.10)
```

Examples

```
tabdiagi(80, 20, 10, 90)
tabdiagi(tp = 80, fp = 20, fn = 10, tn = 90)
tabdiagi(80, 20, 10, 90, ci = c("sens", "spec"))
tabdiagi(80, 20, 10, 90, ci = TRUE, show = FALSE)
tabdiagi(80, 20, 10, 90, prevalence = 0.10)
```

tabexport

Export One or More R4VN Tables

Description

Accepts objects created by `tab()` or `tabmulti()`, as well as data frames and matrices, and returns their data or exports them to HTML, Word, Excel, PDF, or PNG.

Usage

```
tabexport(..., export = NULL, file = NULL, open = FALSE, title = NULL,
          sheet = NULL, overwrite = TRUE, quiet = FALSE)
```

Arguments

...	One or more <code>r4vn_tab</code> , <code>r4vn_tabmulti</code> , data-frame, or matrix objects. A list of tables is also accepted.
<code>export</code>	Output format: "html", "docx", "xlsx", "pdf", "png", or a vector of formats. NULL or FALSE returns data only. TRUE is equivalent to "html". Aliases "word" and "excel" are accepted.
<code>file</code>	Base output path. The extension is added automatically.
<code>open</code>	Logical. Open the last exported file.
<code>title</code>	Optional report title.
<code>sheet</code>	Optional Excel sheet names.
<code>overwrite</code>	Logical. Overwrite existing files.
<code>quiet</code>	Logical. Suppress export messages.

Details

Word and Excel use the flat `$data` component stored in R4VN table objects. HTML, PDF, and PNG retain the original HTML presentation.

Optional packages are required for some formats:

- Excel: **openxlsx** or **writexl**;
- Word: **officer** and **flextable**;
- PDF: **pagedown** and Chrome or Chromium;
- PNG: **webshot2** and a compatible browser.

With `export = NULL`, one table returns a data frame and multiple tables return a named list of data frames.

Value

When `export = NULL` or FALSE, returns a data frame or list. Otherwise invisibly returns an object of class `r4vn_export` containing data, files, objects, and call.

See Also

[tab](#) and [tabmulti](#).

Other R4VN tables: [tab\(\)](#), [tabforest\(\)](#), [tablong\(\)](#), [tabmeta\(\)](#), [tabmulti\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [tabsurvey\(\)](#), [vars\(\)](#)

Examples

```
set.seed(2026)
dat <- data.frame(
  age = round(rnorm(80, 45, 12)),
  sex = factor(sample(c("Female", "Male"), 80, TRUE)),
  bmi = round(rnorm(80, 23, 3), 1)
)
```

```

tb <- tab(
  dat,
  vars = vars(c.age, b2.sex, q.bmi),
  title = "Descriptive characteristics",
  show = FALSE
)

# Return a flat data frame without creating a file.
exported_data <- tabexport(tb)
head(exported_data)

# Export HTML to a temporary location.
html_result <- tabexport(
  tb,
  export = "html",
  file = file.path(tempdir(), "r4vn_example"),
  quiet = TRUE
)
html_result$files
unlink(html_result$files)

if (requireNamespace("officer", quietly = TRUE) &&
    requireNamespace("flextable", quietly = TRUE) &&
    requireNamespace("openxlsx", quietly = TRUE)) {
  ex <- tabexport(
    tb,
    export = c("html", "docx", "xlsx"),
    file = tempfile("R4VN_report_"),
    open = FALSE
  )
  unlink(ex$files)
}

# Extended usage examples

t1 <- tab(iris, vars = vars(c.Sepal.Length, c.Sepal.Width, Species),
  show = FALSE)
t2 <- tab(iris, vars = vars(c.Sepal.Length, c.Petal.Length), by = Species,
  test = TRUE, show = FALSE)

# Return the underlying data frame
tabexport(t1, export = NULL)

# One or several output formats, with several tables in one report
ex1 <- tabexport(t1, t2, export = "html", file = tempfile("Iris_tables_"), open = FALSE)
unlink(ex1$files)
if (requireNamespace("officer", quietly = TRUE) &&
    requireNamespace("flextable", quietly = TRUE) &&
    requireNamespace("openxlsx", quietly = TRUE)) {
  ex2 <- tabexport(t1, t2, export = c("html", "docx", "xlsx"),

```

```

        file = tempfile("Iris_analysis_"), title = "Iris analysis", open = FALSE)
    unlink(ex2$files)
}

# PDF/PNG export is available when its external rendering tools are installed.

```

tabforest

Flexible regression, multi-outcome, survival, and subgroup forest plots

Description

tabforest() is the common forest-plot engine for R4VN. It can (1) fit regression models directly from an outcome and focal predictors, (2) reuse a fitted R4VN or standard R model, (3) place several outcomes side by side using the same predictor structure, and (4) create subgroup-effect forests with a p-value for interaction. Numeric estimates are always stored without clipping; xmin and xmax affect only the drawing.

Usage

```

tabforest(
  outcome = NULL,
  predictors = NULL,
  data = NULL,
  time = NULL,
  event = NULL,
  failure = NULL,
  outcomes = NULL,
  subgroup = NULL,
  predictor = NULL,
  type = c("auto", "regression", "multioutcome", "subgroup"),
  crude = TRUE,
  adjusted = FALSE,
  multi = FALSE,
  or = FALSE,
  rr = FALSE,
  pr = FALSE,
  irr = FALSE,
  estimate = c("auto", "beta", "or", "rr", "pr", "irr", "hr"),
  ci = 0.95,
  sample = c("auto", "common", "model"),
  per = NULL,
  per_labels = NULL,
  select = NULL,
  xmin = NULL,
  xmax = NULL,
  ticks = NULL,

```

```
log = NULL,
arrows = TRUE,
row_layout = c("auto", "modelrows", "compact"),
layout = c("dodge", "stack"),
row_spacing = 1,
model_row_gap = 0.55,
group_gap = 0.25,
order = NULL,
reference = TRUE,
pvalue = TRUE,
global_p = FALSE,
show_n = FALSE,
show_events = FALSE,
show_model_label = TRUE,
show_interaction_p = TRUE,
p_layout = c("inline", "column"),
effect_digit = 2,
p_digit = 3,
labels = NULL,
level_labels = NULL,
lang = c("en", "vi"),
text = NULL,
model_labels = NULL,
label_title = NULL,
effect_title = NULL,
axis_title = NULL,
title = NULL,
subtitle = NULL,
caption = NULL,
note = TRUE,
template = c("journal", "clean", "minimal"),
grid = c("major", "none", "both"),
font_family = "",
base_size = 11,
label_cex = 1,
header_cex = 1,
axis_cex = 1,
model_cex = 0.86,
colors = NULL,
fills = NULL,
pch = NULL,
lty = NULL,
point_cex = 1.15,
point_lwd = 1,
ci_lwd = 1.2,
ref_lwd = 1,
ref_lty = 2,
ref_col = "gray45",
```

```

    arrow_length = 0.08,
    zebra = FALSE,
    zebra_fill = c("white", "gray94"),
    zebra_by = c("variable", "header"),
    label_width = 0.28,
    forest_width = 0.32,
    column_gap = 0.012,
    panel_gap = 0.012,
    panel_forest_ratio = 0.58,
    label_indent = 0.018,
    label_wrap = 38,
    file = NULL,
    width = 12,
    height = NULL,
    dpi = 300,
    show = TRUE,
    console = FALSE
  )

```

Arguments

outcome	Outcome variable for ordinary regression/subgroup analysis, or a supported fitted object (r4vn_surv, r4vn_tabmulti, r4vn_stat, lm, glm, coxph). For Cox regression, outcome is the event/status variable. May be omitted when outcomes is supplied.
predictors	Focal predictors that should appear in the forest. Prefer vars() so R4VN type/reference declarations are retained, for example vars(c.age, b2.sex, c.bmi, smoking).
data	Data frame. If omitted, the active R4VN data frame is used.
time	Follow-up time variable for Cox regression. Supplying time automatically selects HR unless another incompatible effect is requested.
event	Modeled event level for binary OR/RR/PR outcomes.
failure	Event value for Cox regression. With numeric 0/1 status, 1 is selected automatically when present.
outcomes	Optional named character vector or named list for a multi-outcome forest. Character example: c(HTN="hypertension", DEP="depression"). A list allows outcome-specific settings, e.g. list(HTN=list(outcome="hypertension", event="Yes"), Death=list(outcome="death", time="followup", failure=1, estimate="hr")).
subgroup	Optional categorical variables for subgroup analysis. When supplied, use predictor for the main exposure whose effect is estimated within each subgroup level.
predictor	Main exposure for subgroup mode. It may be continuous or a two-level categorical variable. Use vars(b2.treatment) when a non-default reference is required.
type	Analysis mode: "auto", "regression", "multioutcome", or "subgroup". "auto" chooses multi-outcome when outcomes is non-NULL, subgroup when subgroup is non-NULL, otherwise ordinary regression.

crude	For regression/multi-outcome mode, fit one crude model per focal predictor. Set FALSE when only adjusted/multivariable estimates are wanted.
adjusted	Regression mode: FALSE, TRUE, or vars(...). With vars(X1,X2), each focal predictor gets a separate model adjusted for X1/X2. With TRUE, each focal predictor is adjusted for all other focal predictors. Subgroup mode requires an explicit vars(...) adjustment set if adjustment is desired.
multi	Regression mode: FALSE, TRUE, or vars(...). TRUE fits one joint model containing all focal predictors. vars(A,B,C,X) fits that exact joint model but still displays only variables listed in predictors. In subgroup mode, an explicit vars(...) may also be used as the final adjustment set; the exposure and current subgroup variable are removed from the covariate set automatically.
or, rr, pr, irr	Logical shortcuts for OR, RR, PR, or IRR. Only one may be TRUE. Binary outcomes default to OR; numeric outcomes default to beta.
estimate	Explicit effect type: "auto", "beta", "or", "rr", "pr", "irr", or "hr".
ci	Confidence level, default 0.95.
sample	Missing-data strategy. "common" forces displayed regression models to use the same complete-case sample; "model" allows each model to use its own available cases; "auto" uses a common sample when several regression model groups are displayed. In subgroup mode, "auto" behaves like model-specific analysis so unrelated subgroup variables do not reduce one another's sample size.
per	Optional multiplier for continuous effects. Example c(age=10) reports the ratio/HR per 10 years or beta per 10 units.
per_labels	Optional display labels for per, e.g. c(age="per 10 years").
select	Model components when outcome is a fitted R4VN object. For tabsurv() this can include "crude", "adjusted", "multi"; for tabmulti() use stored model names such as "full", "backward".
xmin, xmax	Forest plotting limits. These never alter stored estimates. For ratio effects, if only xmax is supplied and is >1, xmin=1/xmax is used automatically. In multi-outcome mode these may be named vectors or lists keyed by outcome-panel name.
ticks	Optional axis ticks. In multi-outcome mode a named list can give different ticks to different outcome panels.
log	NULL or logical. Ratio effects default to logarithmic axes, beta to linear. In multi-outcome mode this may be a named logical vector/list.
arrows	Draw arrowheads when CIs extend beyond plotting limits. When the point estimate itself is outside the range, no false boundary point is drawn.
row_layout	"auto", "modelrows", or "compact". The default "auto" uses "modelrows" whenever more than one model group is displayed and "compact" when only one model is displayed. In "modelrows", Crude, Adjusted, and Multivariable are separate physical rows but share ONE effect column (for example, one OR (95% CI) column). Thus each CI, marker, numeric estimate, and p-value is aligned with its own row. Use "compact" only when several model estimates are deliberately wanted on the same labelled row.
layout	In compact mode, "dodge" or "stack" controls vertical offsets of multiple model markers/CI lines.

<code>row_spacing</code>	Baseline distance between ordinary rows.
<code>model_row_gap</code>	Distance between model rows belonging to the same variable/level when <code>row_layout="modelrows"</code> .
<code>group_gap</code>	Extra vertical separation between variable blocks.
<code>order</code>	Optional order of focal predictor variable names.
<code>reference</code>	Show categorical reference rows. Default TRUE.
<code>pvalue</code>	Show coefficient-level p-values.
<code>global_p</code>	Show categorical-variable omnibus Wald p-values. Default FALSE because forest plots are usually cleaner without these values.
<code>show_n, show_events</code>	Add model N and number of events after the estimate.
<code>show_model_label</code>	In <code>modelrows</code> , print the model label (Crude, Adjusted, Multivariable) beside the corresponding row.
<code>show_interaction_p</code>	In subgroup mode, show the p-value for interaction on the subgroup-variable header row.
<code>p_layout</code>	Regression display style: "inline" appends p to the numeric effect string; "column" uses a separate p-value column. <code>modelrows</code> works particularly well with either style.
<code>effect_digit, p_digit</code>	Decimal places for effects and p-values.
<code>labels</code>	Named character vector overriding variable labels.
<code>level_labels</code>	Named list overriding displayed categorical levels. This changes display only, not model coding/reference levels.
<code>lang</code>	Built-in language: "en" or "vi".
<code>text</code>	Named list overriding individual words. Useful keys include <code>characteristic</code> , <code>reference</code> , <code>crude</code> , <code>adjusted</code> , <code>multi</code> , <code>p</code> , <code>n</code> , <code>events</code> , <code>subgroup</code> , <code>interaction_p</code> , <code>overall</code> , <code>arrow_note</code> , and effect keys <code>OR</code> , <code>RR</code> , <code>PR</code> , <code>IRR</code> , <code>HR</code> , <code>beta</code> .
<code>model_labels</code>	Named character vector overriding model labels, e.g. <code>c(Crude="Unadjusted", Multivariable="Adjusted")</code> .
<code>label_title, effect_title, axis_title</code>	Optional column/axis titles.
<code>title, subtitle, caption</code>	Optional plot title, subtitle, caption.
<code>note</code>	TRUE for an automatic clipping note, FALSE for none, or custom text.
<code>template</code>	Visual preset: "journal", "clean", "minimal".
<code>grid</code>	"major", "none", or "both".
<code>font_family</code>	Base graphics font family.
<code>base_size, label_cex, header_cex, axis_cex, model_cex</code>	Text-size controls.
<code>colors</code>	Model line/marker colors. A named vector is recommended, e.g. <code>c(Crude="gray50", Adjusted="navy")</code> .
<code>fills</code>	Optional marker fill colors, useful with <code>pch 21:25</code> .

pch	Model marker symbols. Named vectors may assign different symbols to crude and adjusted estimates.
lty	Model CI line types.
point_cex	Model marker sizes. May be scalar or named vector by model.
point_lwd	Marker border widths. May be scalar or named vector.
ci_lwd	CI line widths. May be scalar or named vector by model.
ref_lwd, ref_lty, ref_col	Null-line appearance.
arrow_length	Arrowhead size in inches.
zebra	Draw alternating background blocks by predictor/subgroup.
zebra_fill	Two or more background colors, e.g. <code>c("white", "gray93")</code> .
zebra_by	"variable" or "header"; both currently alternate complete variable/subgroup blocks so all levels/models in a block share a background.
label_width, forest_width, column_gap	Horizontal layout controls for a single regression/subgroup forest.
panel_gap	Gap between panels in multi-outcome mode.
panel_forest_ratio	Fraction of each multi-outcome panel devoted to the CI forest; the remaining panel width is used for numeric estimates.
label_indent	Indentation of categorical levels.
label_wrap	Approximate wrapping width for long labels; Inf disables.
file	Optional PDF/PNG/SVG/JPG/TIFF output file.
width, height, dpi	Graphics dimensions. If height is NULL it grows with the actual number of drawn rows, so <code>modelrows</code> can produce a tall figure without compressing row spacing.
show	Draw immediately. Default TRUE.
console	Print the long standardized estimate table.

Details

1. Regression model semantics:

With `predictors=vars(A,B,C)`:

- `crude=TRUE`: $Y \sim A$, $Y \sim B$, $Y \sim C$.
- `adjusted=vars(X)`: $Y \sim A+X$, $Y \sim B+X$, $Y \sim C+X$.
- `adjusted=TRUE`: each focal variable is adjusted for the other focal vars.
- `multi=TRUE`: one joint model $Y \sim A+B+C$.
- `multi=vars(A,B,C,X)`: one joint model $Y \sim A+B+C+X$, but only A/B/C are shown.

Therefore `adjusted` and `multi` answer different scientific questions and may be requested together in the same forest. When two or more model groups are displayed, `row_layout="auto"` uses separate model rows and ONE shared effect column. For example, Crude and Multivariable ORs are both printed under the same OR (95% CI) header instead of being put in separate Crude-OR and Multivariable-OR columns.

2. Effect measure selected by outcome:

- numeric continuous outcome -> linear regression -> beta, null=0;
- binary outcome -> logistic regression -> OR, null=1;
- pr=TRUE -> robust modified Poisson -> PR, null=1;
- rr=TRUE -> robust modified Poisson -> RR, null=1;
- count outcome + irr=TRUE -> Poisson -> IRR, null=1;
- time= -> Cox proportional hazards -> HR, null=1.

3. Exact numbers when the forest is clipped:

Suppose OR=7.41 and 95% CI=1.56 to 35.20 while xmax=10. The printed number remains 7.41 (1.56-35.20). Only the graphical CI is truncated at 10 and an arrow is drawn. If OR itself exceeds 10, no marker is placed falsely at 10.

4. Multiple models: separate rows, one merged effect column:

row_layout="auto" is the default. If more than one model group is present, it automatically switches to the "modelrows" layout. Crude, Adjusted and/or Multivariable estimates are placed on separate physical rows, while the right side contains only ONE shared effect column such as OR (95% CI), HR (95% CI), or Beta (95% CI). The result, CI line, marker and p-value therefore stay on exactly the same row. This is the recommended publication layout when crude and adjusted estimates are presented together. Increase row_spacing, model_row_gap, group_gap, or leave height=NULL for a taller figure.

Set row_layout="compact" only when you intentionally want several model estimates on the same labelled row; compact mode retains separate numeric model columns because the rows are not expanded.

5. Multi-outcome forests:

outcomes= creates side-by-side panels sharing predictor labels. Every panel may have its own effect type, axis range and follow-up variable. This permits two binary outcomes (OR panels), several continuous outcomes (beta panels), or even mixed OR/HR panels in one figure. For very many panels, increase width.

6. Subgroup forests:

subgroup= estimates the effect of one main predictor separately within each subgroup level. A full model containing predictor*subgroup is fitted for the Wald interaction p-value. For RR/PR, the interaction Wald test uses the same robust covariance approach as the effect model. Cox subgroup forests use HR and a Cox interaction model. A subgroup variable must be categorical; make clinically meaningful groups before calling tabforest().

7. Styling:

colors, fills, pch, lty, point_cex, point_lwd, and ci_lwd accept named model vectors. zebra=TRUE shades complete variable blocks, closely matching journal forest-table layouts. lang="vi", text=, labels=, and level_labels= allow all visible wording to be translated without changing the model.

Value

An object of class r4vn_tabforest; multi-outcome and subgroup modes add subclasses r4vn_tabforest_multi and r4vn_tabforest_subgroup. \$data is publication-ready, \$table is the numeric long table, \$models stores fitted models, and plot() can redraw without refitting.

See Also

vars, tab, tabmulti, tabsurv, tabexport

Other R4VN tables: [tab\(\)](#), [tabexport\(\)](#), [tablong\(\)](#), [tabmeta\(\)](#), [tabmulti\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [tabsurvey\(\)](#), [vars\(\)](#)

Examples

```
data(tabforest_demo)
usedf(tabforest_demo, quiet = TRUE)

# Crude odds ratios.
f1 <- tabforest(
  hypertension,
  predictors = vars(c.age, sex, c.bmi, smoking),
  event = "Yes",
  show = FALSE
)
f1$data

# Crude plus one final multivariable model.
f2 <- tabforest(
  hypertension,
  predictors = vars(c.age, sex, c.bmi, smoking),
  event = "Yes",
  crude = TRUE, multi = TRUE,
  show = FALSE
)

# Re-drawing is intentionally interactive so CRAN examples do not depend
# on the graphics device or installed fonts.
if (interactive()) {
  plot(f2, row_layout = "modelrows", zebra = TRUE)
}

# Each focal predictor adjusted for the same confounders.
f3 <- tabforest(
  hypertension,
  predictors = vars(c.age, c.bmi, smoking),
  event = "Yes",
  adjusted = vars(sex, education),
  show = FALSE
)

# Vietnamese display text can be prepared without drawing during checks.
f_vi <- tabforest(
  hypertension,
  predictors = vars(c.age, sex, c.bmi, smoking),
  event = "Yes",
  multi = TRUE,
  lang = "vi",
  labels = c(
    age = "Tu\u1ed5i",
```

```

    sex = "Gi\u1edbi t\u00ednh",
    bmi = "Ch\u1ec9 s\u1ed1 kh\u1ed1i c\u01a1 th\u1ec3",
    smoking = "H\u00fat thu\u1ed1c"
  ),
  level_labels = list(
    sex = c(Female = "N\u1eeef", Male = "Nam"),
    smoking = c(No = "Kh\u00f4ng", Yes = "C\u00f3")
  ),
  text = list(reference = "Tham chi\u1ebfu"),
  title = "Bi\u1ec3u \u0111\u1ed3 forest",
  show = FALSE
)
if (interactive()) plot(f_vi)

# Modified-Poisson prevalence ratio.
f_pr <- tabforest(
  depression,
  predictors = vars(c.age, sex, smoking, alcohol),
  event = "Yes", pr = TRUE, multi = TRUE,
  show = FALSE
)

# Continuous outcome.
f_beta <- tabforest(
  sbp,
  predictors = vars(c.age, sex, c.bmi, smoking),
  multi = TRUE,
  show = FALSE
)

# Cox model, only when the suggested package is available.
if (requireNamespace("survival", quietly = TRUE)) {
  f_hr <- tabforest(
    death, time = followup,
    predictors = vars(c.age, sex, treatment, c.bmi),
    failure = 1, crude = TRUE, multi = TRUE,
    show = FALSE
  )
}

# Multi-outcome forest without drawing.
f_multi <- tabforest(
  outcomes = c(
    Hypertension = "hypertension",
    Depression = "depression"
  ),
  predictors = vars(c.age, sex, c.bmi, smoking),
  event = "Yes", crude = FALSE, multi = TRUE,
  show = FALSE
)

# Subgroup forest without drawing.

```

```
f_sub <- tabforest(
  hypertension,
  predictor = vars(treatment),
  subgroup = vars(age_group, sex, obesity, diabetes, smoking),
  event = "Yes", type = "subgroup",
  adjusted = vars(c.age, c.bmi),
  show = FALSE
)

# File output uses a temporary path and is cleaned up.
f_png <- tempfile(fileext = ".png")
tabforest(
  hypertension,
  predictors = vars(c.age, sex, c.bmi, smoking),
  event = "Yes", multi = TRUE,
  file = f_png, width = 8, height = 5, dpi = 120,
  show = FALSE
)
unlink(f_png)

usedf(clear = TRUE, quiet = TRUE)
```

tabforest_demo

Demonstration data for tabforest()

Description

A simulated health-research dataset designed specifically for the worked examples in `tabforest()`. It contains binary, continuous, count, and time-to-event outcomes, together with continuous and categorical predictors. A small amount of missing data is intentional so users can compare sample = "common" and sample = "model".

Usage

```
data(tabforest_demo)
```

Format

A data frame with 500 rows and 19 variables:

id Participant identifier.

age Age in years.

sex Sex: Female or Male.

bmi Body mass index in kg/m².

smoking Current smoking: No or Yes.

alcohol Current alcohol use: No or Yes.

education Primary, Secondary, or College.

treatment Control or Intervention.

hypertension Binary outcome: No or Yes.

depression Binary prevalence outcome: No or Yes.

sbp Systolic blood pressure in mmHg; continuous outcome.

visits Number of healthcare visits; count outcome.

followup Follow-up time in years.

death Survival event indicator: 0=censored, 1=death.

hospital Hospital cluster identifier H1-H5.

age_group Age group: <55 or >=55 years.

obesity Demo obesity grouping: No or Yes (BMI >=27.5 for teaching only).

diabetes Simulated diabetes mellitus: No or Yes.

dyslipidemia Simulated dyslipidemia: No or Yes.

Details

The dataset is entirely simulated and contains no real patient information. It is intended for package examples, teaching, testing, and screenshots.

Source

Simulated by R4VN for reproducible examples; seed 20260813.

Examples

```
data(tabforest_demo)
str(tabforest_demo)
table(tabforest_demo$hypertension, useNA = "ifany")
summary(tabforest_demo$sbp)

# A CSV copy is also installed for teaching/import demonstrations.
p <- system.file("extdata", "tabforest_demo.csv", package = "R4VN")
p
```

tabi

Immediate Contingency Tables

Description

Creates an immediate contingency table from typed counts. The immediate mode uses the same calculation engine as the corresponding R4VN analyses when values are supplied directly, for example `tab(sex, outcome, data = dat, chi = TRUE)`.

Usage

```

tabi(
  ...,
  row.names = NULL,
  col.names = NULL,
  percent = c("none", "row", "col", "total"),
  row = FALSE,
  col = FALSE,
  cell = FALSE,
  total = FALSE,
  exp = FALSE,
  chi = TRUE,
  fisher = FALSE,
  lr = FALSE,
  residual = FALSE,
  adjresidual = FALSE,
  correct = FALSE,
  digits = 1,
  p_digits = 3,
  workspace = 2e+05,
  show = TRUE,
  console = FALSE
)

```

Arguments

...	Numeric row vectors, a matrix, or a table. For <code>anovai()</code> , each argument is a group summary in the form <code>c(n, mean, sd)</code> .
<code>row.names</code> , <code>col.names</code>	Optional row and column labels.
<code>percent</code>	Percentage denominator: "none", "row", "col", or "total".
<code>row</code> , <code>col</code> , <code>cell</code> , <code>total</code>	Logical shortcuts for row, column, or total-cell percentages. Only one may be TRUE.
<code>exp</code>	Show expected counts.
<code>chi</code>	Show Pearson's chi-squared test.
<code>fisher</code>	Show Fisher's exact test.
<code>lr</code>	Show the likelihood-ratio chi-squared test.
<code>residual</code>	Show Pearson residuals.
<code>adjresidual</code>	Show adjusted residuals.
<code>correct</code>	Apply Yates's correction for a 2 by 2 Pearson test.
<code>digits</code>	Number of decimal places for estimates.
<code>p_digits</code>	Number of decimal places for p-values.
<code>workspace</code>	Workspace passed to <code>stats::fisher.test()</code> .
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional text result in the Console. Default FALSE.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
tabi(c(2, 3), c(5, 6), c(8, 9), row = TRUE,
     exp = TRUE, chi = TRUE, fisher = TRUE)
```

tablearn	<i>Learning-curve analysis for sequential clinical or procedural performance</i>
----------	--

Description

`tablearn()` analyzes performance across consecutive cases, procedures, or observations. It keeps individual case-level data for statistical inference while allowing visually stable rolling or block summaries for learning-curve display. A right-aligned rolling window such as cases 1-5, 2-6, 3-7, ... is particularly useful when the question is: "What is my current performance based on the most recent N cases?"

Usage

```
tablearn(
  outcome,
  data = NULL,
  order = NULL,
  operator = NULL,
  type = c("auto", "continuous", "binary", "count"),
  event = NULL,
  better = c("auto", "lower", "higher"),
  window = 5L,
  window_type = c("rolling", "block", "none", "cumulative"),
  window_align = c("right", "center", "left"),
  window_step = NULL,
  window_complete = TRUE,
  window_fun = c("auto", "mean", "median", "trimmed_mean", "proportion", "rate"),
  window_trim = 0.1,
  window_weight = c("equal", "linear", "exponential"),
  window_decay = 0.85,
  window_ci = TRUE,
  ci_level = 0.95,
  window_ci_method = c("auto", "t", "normal", "wilson", "exact"),
  interval = c("ci", "sd", "iqr", "range", "none"),
  ewma = FALSE,
  ewma_lambda = 0.2,
  ewma_init = c("first", "mean", "target"),
  smooth = TRUE,
```

```
smooth_method = c("loess", "spline", "lm", "gam", "none"),
smooth_on = c("window", "raw", "ewma"),
smooth_span = 0.6,
smooth_df = NULL,
smooth_ci = TRUE,
breakpoint = TRUE,
break_on = c("raw", "window"),
break_min_n = 8L,
break_grid = NULL,
break_boot = 0L,
phase = TRUE,
phase_breaks = NULL,
phase_names = c("Learning", "Consolidation", "Proficiency"),
phase_method = c("auto", "combined", "manual", "breakpoint", "proficiency"),
proficiency = TRUE,
proficiency_method = c("auto", "combined", "target", "lccusum", "breakpoint", "manual"),
proficiency_case = NULL,
proficiency_case_rule = c("confirmed", "first_stable"),
target_hold = 3L,
target_tolerance = 0,
plateau = TRUE,
plateau_ratio = 0.25,
plateau_slope = NULL,
stability = TRUE,
stability_metric = c("auto", "sd", "iqr", "cv", "none"),
stability_window = NULL,
stability_ratio = 0.75,
cusum = FALSE,
target = NULL,
cusum_method = c("deviation", "llr"),
cusum_alt = NULL,
cusum_reset = FALSE,
cusum_limit = NULL,
lccusum = FALSE,
p_acceptable = NULL,
p_unacceptable = NULL,
alpha = 0.05,
beta = 0.1,
racusum = FALSE,
expected = NULL,
adjust = NULL,
racusum_or = 2,
racusum_limit = NULL,
sensitivity = FALSE,
window_sensitivity = NULL,
plot = TRUE,
plot_type = c("auto", "learning", "cusum", "both"),
x_axis = c("case", "order"),
```

```

operator_display = c("facet", "overlay"),
show_raw = TRUE,
show_window = TRUE,
show_smooth = TRUE,
show_ewma = TRUE,
show_interval = TRUE,
show_break = TRUE,
show_proficiency = TRUE,
show_phase = TRUE,
show_target = TRUE,
show_current = TRUE,
title = NULL,
subtitle = NULL,
caption = NULL,
xlab = NULL,
ylab = NULL,
theme = c("publication", "minimal", "classic", "bw", "gray"),
legend = "bottom",
plot_opts = NULL,
lang = c("en", "vi"),
digit = 2,
p_digit = 3,
interpretation = FALSE,
viewer_plot_format = c("png", "svg"),
console = FALSE,
show = TRUE
)

```

Arguments

outcome	Outcome variable. May be a bare variable name, character name, or <code>vars(...)</code> containing multiple outcomes.
data	Data frame. If <code>NULL</code> , <code>tablearn()</code> attempts to use the R4VN active data frame.
order	Optional ordering variable such as case number or procedure date. If omitted, current row order is used. Data are sorted by this variable within each operator.
operator	Optional operator/surgeon/trainee variable. Curves and analyses are then calculated separately within operator.
type	Outcome type: "auto", "continuous", "binary", or "count". In auto mode, logical/factor/character variables with exactly two non-missing levels and numeric 0/1 variables are treated as binary. A numeric variable with two other observed values (for example, 50 and 100) is treated as continuous unless event is supplied or <code>type = "binary"</code> is requested. This conservative rule prevents a two-valued continuous measurement from being silently converted to a proportion.
event	Event level for a binary outcome. If omitted, the second factor level, <code>TRUE</code> , or the larger numeric level is used.

better	Direction of better performance: "lower", "higher", or "auto". Auto uses "lower" for numeric outcomes and for an event coded as failure/complication it should normally be set explicitly by the analyst.
window	Number of cases in a rolling or block window. Default 5.
window_type	"rolling" (1-5, 2-6, 3-7, ...), "block" (1-5, 6-10, ...), "cumulative" (1, 1-2, 1-3, ...), or "none".
window_align	Alignment for rolling windows: "right" (recommended and default for current-performance monitoring), "center", or "left".
window_step	Number of cases to advance each window. Default is 1 for rolling/cumulative and window for block summaries.
window_complete	If TRUE, only complete windows are retained. If FALSE, partial windows at the beginning/end are allowed.
window_fun	"auto", "mean", "median", "trimmed_mean", "proportion", or "rate". Auto uses a proportion for binary outcomes and mean otherwise.
window_trim	Trim proportion used by window_fun = "trimmed_mean".
window_weight	"equal", "linear", or "exponential". Equal weighting gives the ordinary rolling mean/proportion; alternatives give more weight to recent cases within each window.
window_decay	Decay in (0,1] for exponentially weighted windows.
window_ci	Show point-wise interval for each window.
ci_level	Confidence level, default 0.95.
window_ci_method	"auto", "t", "normal", "wilson", or "exact". Auto uses Wilson for binary proportions, t intervals for small continuous windows, and normal intervals for larger continuous windows.
interval	Window interval displayed/calculated: "ci", "sd", "iqr", "range", or "none".
ewma	Logical; calculate an exponentially weighted moving average.
ewma_lambda	EWMA smoothing parameter in (0,1]; larger values react more strongly to the latest case.
ewma_init	EWMA starting value: "first", "mean", or "target".
smooth	Logical; add a fitted smooth trend.
smooth_method	"loess", "spline", "lm", "gam", or "none". GAM requires package mgcv; otherwise loess is used as fallback.
smooth_on	Data used only for the descriptive smooth: "window", "raw", or "ewma".
smooth_span	LOESS span.
smooth_df	Optional degrees of freedom for smoothing spline.
smooth_ci	Show a point-wise confidence band where the smoothing method supports one.
breakpoint	Logical; estimate a one-change-point piecewise regression.
break_on	"raw" (recommended default for inference) or "window". Overlapping rolling windows are correlated, so fitting the inferential model to raw cases avoids treating overlapping windows as independent observations.

break_min_n	Minimum observations required on each side of a candidate breakpoint.
break_grid	Optional numeric vector of candidate case positions.
break_boot	Number of bootstrap replications for an exploratory percentile CI for the breakpoint. 0 disables bootstrap.
phase	Logical; create a phase summary table using the estimated or user supplied phase breaks.
phase_breaks	Optional numeric vector of manual phase boundaries. This is useful for 3+ named phases; automatic estimation currently provides one main change point.
phase_names	Names of phases. Defaults include Learning, Consolidation, and Proficiency. Manual names are used when phase_breaks is supplied.
phase_method	Phase classification rule: "auto"/"combined" uses the detected learning change point together with the estimated proficiency case; "manual" uses only phase_breaks; "breakpoint" uses the change point; and "proficiency" divides the series at the proficiency case. Automatic classification never labels a phase Proficiency unless proficiency is actually estimated.
proficiency	Logical; assess whether proficiency has been reached.
proficiency_method	"auto", "combined", "target", "lccusum", "breakpoint", or "manual". The recommended "auto" resolves to a combined rule. A supplied target and an enabled LC-CUSUM are treated as required criteria; plateau/stability strengthen the evidence.
proficiency_case	Manual case number used only with proficiency_method = "manual".
proficiency_case_rule	For sustained-target proficiency, report the first qualifying window ("first_stable") or the window in which the required run is confirmed ("confirmed", default).
target_hold	Number of consecutive window estimates that must satisfy target before target-based proficiency is confirmed. Default 3.
target_tolerance	Non-negative tolerance around target. For better = "lower", values $\leq$ target + tolerance qualify; for better = "higher", values $\geq$ target - tolerance qualify.
plateau	Logical; assess whether the post-breakpoint slope is sufficiently small to be interpreted as an operational plateau. A breakpoint alone is not automatically called proficiency.
plateau_ratio	Relative plateau threshold when plateau_slope is NULL. The post-breakpoint absolute slope must be $\leq$ plateau_ratio times the initial absolute slope. Default 0.25.
plateau_slope	Optional absolute slope threshold overriding plateau_ratio. This is outcome-scale specific and can be useful when a clinically meaningful slope threshold is known.
stability	Logical; assess whether performance variability has fallen.
stability_metric	"auto", "sd", "iqr", "cv", or "none". Auto uses SD for continuous/count outcomes and does not use variability as proficiency evidence for binary outcomes.

stability_window	Number of early and late raw cases used to compare variability. Default is at least the selected learning-curve window size.
stability_ratio	Late/early variability ratio required for stability. Default 0.75 means late variability must be at most 75% of early variability.
cusum	Logical; calculate a conventional CUSUM from individual cases.
target	Clinical/quality target for CUSUM, target/reference line, and optional EWMA initialization.
cusum_method	"deviation" or binary likelihood-ratio "llr".
cusum_alt	Alternative binary failure probability for LLR-CUSUM.
cusum_reset	If TRUE, use a one-sided tabular CUSUM reset at zero.
cusum_limit	Optional decision limit. If omitted, CUSUM is descriptive.
lccusum	Logical; calculate a binary LC-CUSUM designed to signal evidence that an acceptable failure rate has been reached.
p_acceptable	Acceptable failure probability for LC-CUSUM.
p_unacceptable	Unacceptable failure probability for LC-CUSUM; must be larger than p_acceptable.
alpha, beta	Type-I and Type-II error probabilities used in the LC-CUSUM decision boundary formula.
racusum	Logical; calculate a binary risk-adjusted CUSUM using likelihood scores and individual expected risks.
expected	Optional expected-risk variable or numeric vector for RA-CUSUM. Supplying externally validated or pre-operative expected risks is preferable.
adjust	Optional covariates used to fit a logistic expected-risk model if expected is not supplied. May be character names or vars(...).
racusum_or	Odds ratio representing deterioration to be detected. Must be greater than 1.
racusum_limit	Optional RA-CUSUM decision limit.
sensitivity	Logical; calculate window-size sensitivity summaries.
window_sensitivity	Numeric vector of window sizes. If omitted and sensitivity = TRUE, sensible values are chosen from the sample size.
plot	Logical; create figures. Standard figures and Viewer figures use base R and therefore require no add-on package. When ggplot2 is installed, an advanced ggplot object is also retained in \$plots.
plot_type	"auto", "learning", "cusum", or "both". Auto shows the learning curve and also the CUSUM panel whenever conventional CUSUM, LC-CUSUM, or RA-CUSUM has been requested.
x_axis	"case" (default) or "order". Case number is usually preferable for learning curves; date/time can be shown using "order".
operator_display	"facet" or "overlay" when operator is supplied.

show_raw, show_window, show_smooth, show_ewma, show_interval, show_break, show_proficiency, show_phase, show_target, show_current	High-level plot layer switches.
title, subtitle, caption, xlab, ylab	Plot labels. Defaults are generated from the outcome and selected language.
theme	Plot theme: "publication", "minimal", "classic", "bw", or "gray".
legend	Legend position: "bottom", "top", "left", "right", or "none".
plot_opts	Nested list for advanced plot customization. See the dedicated Plot options section below. Values supplied here override defaults.
lang	"en" or "vi" for generated labels/messages.
digit	Number of decimals for ordinary estimates in Viewer tables.
p_digit	Number of decimals for p-values in Viewer tables.
interpretation	Logical; include a short interpretation section in the Viewer/console report. Default is FALSE; the interpretation text is still retained in \$interpretation for programmatic use.
viewer_plot_format	Self-contained Viewer image format: "png" or "svg". Both are produced with base R graphics and need no extra package.
console	Print a concise analysis summary to the console. Default FALSE.
show	Open the complete HTML report in the RStudio Viewer (or browser) and display requested figures in the Plot pane. Default TRUE.

Value

An object of class `r4vn_tablearn`. For one outcome it contains at least `raw`, `window`, `ewma`, `smooth`, `breakpoint`, `proficiency`, `proficiency_evidence`, `phases`, `phase_classification`, `current`, `current_status`, `cusum`, `lccusum`, `racusum`, `sensitivity`, `plots`, `settings`, `plot_options`, `tables`, and `interpretation`. With `show = TRUE`, the returned object also carries the generated self-contained Viewer HTML/file path. Multiple outcomes return class `r4vn_tablearn_multi` containing one analysis per outcome.

Rolling-window interpretation

With `window = 5`, `window_type = "rolling"`, `window_align = "right"`, and `window_step = 1`, the first displayed point summarizes cases 1-5, the next summarizes 2-6, then 3-7, and so on. Therefore the point at case 100 represents performance in the most recent five cases (96-100). A newly observed case 101 updates the curve to cases 97-101. This is different from non-overlapping block summaries and from a cumulative mean.

Statistical inference versus visual smoothing

Overlapping rolling windows share observations and are therefore correlated. `tablearn()` can display rolling windows for a stable curve while fitting the change-point model and CUSUM on the original case sequence. The recommended default is `break_on = "raw"`; CUSUM, LC-CUSUM and RA-CUSUM always operate on individual sequential cases in this implementation.

Advanced plot options

plot_opts is a nested list. Every field is optional. Main groups are:

- raw: show, color, fill, shape, size, alpha, stroke.
- window: show, geom("line", "point", "point_line"), color, fill, shape, size, alpha, line_color, line_width, line_type.
- window_interval: show, geom("ribbon" or "errorbar"), color, fill, alpha, line_width, width.
- smooth: show, color, fill, line_width, line_type, alpha, ci_show, ci_alpha.
- ewma: show, color, line_width, line_type, alpha.
- breakpoint: show, color, line_width, line_type, alpha, label, label_text, label_size, label_angle, label_hjust, label_vjust.
- proficiency: independent proficiency-line controls: show, color, line_width, line_type, alpha, label, label_text, label_size, label_angle, label_hjust, label_vjust.
- phase: show, fills, alpha, border_color, border_width, label, label_size, label_position.
- target: show, color, line_width, line_type, alpha, label, label_text, label_size.
- current: show, color, fill, shape, size, alpha, stroke, label, label_text, label_size, hjust, vjust.
- axes: x_limits, y_limits, x_breaks, y_breaks, x_expand, y_expand, y_percent, percent_accuracy, x_reverse, y_reverse, x_trans, y_trans, x_date_format, x_date_breaks, clip.
- legend: show, position, title, direction.
- facet: ncol, nrow, scales.
- operator: colors, shapes, line_types; layout is selected by the high-level operator_display argument.
- cusum: CUSUM-figure controls including colors, line_types, line_width, alpha, point controls, zero-line controls, decision-limit controls, signal marker controls, title/subtitle/caption/axis labels, and CUSUM-specific axis limits/breaks.
- theme: name, base_size, base_family, grid_major, grid_minor, panel_border, axis_line, plot_title_face, legend_key_size.
- text: optional title/subtitle/caption/axis/legend text sizes.
- margins: top, right, bottom, left in points.
- panel: optional background/border customization.
- annotation: show_n, show_window_label.

This layered design lets the raw observations remain visible while the rolling curve, uncertainty, fitted smooth, target, breakpoint, estimated proficiency, phases, and current performance are styled independently.

Proficiency and phase classification

`tablearn()` deliberately distinguishes a statistical/descriptive change point from proficiency. A change point indicates a change in the learning trajectory; proficiency is estimated from one or more operational criteria. With the default combined rule, a supplied clinical target must be sustained for `target_hold` consecutive displayed windows, and an enabled LC-CUSUM must cross its competency decision boundary. A post-change plateau and reduced variability strengthen the evidence. If no target or LC-CUSUM is supplied, a clear plateau can provide a limited, data-driven proficiency estimate. Manual proficiency is also supported.

Automatic phase classification uses these results rather than forcing every dataset into three phases. When both an earlier learning change point and a later proficiency case are found, phases are Learning -> Consolidation -> Proficiency. If proficiency is not established, a post-change segment is labelled Consolidation rather than Proficiency. `phase_breaks` always allows complete manual control for study protocols with pre-specified phases.

The evidence label (Strong, Moderate, Limited) is an R4VN rule-based summary of concordant criteria, not a confidence probability and not a substitute for a clinically defined competency standard.

Viewer and dependency policy

With `show = TRUE` (default), `tablearn()` opens one self-contained HTML report containing the key publication-ready tables and every requested figure. The same figures are also sent to the Plot pane. Standard analysis, HTML rendering, learning curves, CUSUM, LC-CUSUM, and RA-CUSUM use only base/recommended R packages. `ggplot2` is optional: when already installed, an advanced ggplot object is retained in `$plots`; when it is absent, plotting still works through the base-R fallback. `mgcv` is needed only when the user explicitly selects `smooth_method = "gam"`; otherwise the default smoothing methods use base R.

Examples

```
# -----
# Reproducible demonstration data used by the examples below
# -----
set.seed(2026)
n <- 150
d <- data.frame(
  case = 1:n,
  date = as.Date("2025-01-01") + 0:(n - 1),
  surgeon = rep(c("A", "B", "C"), each = n / 3),
  complexity = rbinom(n, 1, 0.35),
  age = round(rnorm(n, 58, 12), 1)
)
d$time <- 115 - 48 * (1 - exp(-d$case / 30)) +
  9 * d$complexity + rnorm(n, 0, 9)
d$score <- 55 + 28 * (1 - exp(-d$case / 35)) + rnorm(n, 0, 5)
d$expected_risk <- plogis(-1.8 + 0.9 * d$complexity + 0.012 * (d$age - 58))
actual_risk <- plogis(qlogis(d$expected_risk) - 0.010 * d$case)
d$complication <- rbinom(n, 1, actual_risk)
d$errors <- rpois(n, pmax(0.15, 3.2 * exp(-d$case / 45)))
```

```

# The full catalogue is interactive so R CMD check stays fast.
if (interactive()) {

# 1. Simplest end-user command. In an interactive session this opens the
# complete Viewer report and sends the learning curve to the Plot pane.
if (interactive()) {
  m1 <- tablearn(time, data = d, order = case)
}

# 2. Right-aligned rolling window: 1-5, 2-6, 3-7, ...
m2 <- tablearn(time, data = d, order = case, window = 5,
               show = FALSE, plot = FALSE)
head(m2$tables$Window_performance)
m2$tables$Current_performance

# 3. Non-overlapping blocks: 1-10, 11-20, 21-30, ...
m3 <- tablearn(time, data = d, order = case, window = 10,
               window_type = "block", show = FALSE, plot = FALSE)
head(m3$window[, c(".start", ".end", ".value")])

# 4. Cumulative learning curve: 1, 1-2, 1-3, ...
m4 <- tablearn(time, data = d, order = case,
               window_type = "cumulative", show = FALSE, plot = FALSE)

# 5. Individual-case series without aggregation.
m5 <- tablearn(time, data = d, order = case,
               window_type = "none", show = FALSE, plot = FALSE)

# 6. Median and IQR for a skewed continuous outcome.
m6 <- tablearn(time, data = d, order = case, window = 7,
               window_fun = "median", interval = "iqr",
               show = FALSE, plot = FALSE)

# 7. Give recent cases greater weight within the rolling window.
m7 <- tablearn(time, data = d, order = case, window = 10,
               window_weight = "exponential", window_decay = 0.85,
               show = FALSE, plot = FALSE)

# 8. Add EWMA to the ordinary rolling curve.
m8 <- tablearn(time, data = d, order = case, window = 10,
               ewma = TRUE, ewma_lambda = 0.20,
               show = FALSE, plot = FALSE)
tail(m8$ewma)

# 9. Automatic piecewise change point. Inference uses raw cases by default.
m9 <- tablearn(time, data = d, order = case, window = 5,
               breakpoint = TRUE, break_on = "raw",
               show = FALSE, plot = FALSE)
m9$tables$Change_point

# 10. Sustained clinical target: <= 70 for 3 consecutive windows.
m10 <- tablearn(time, data = d, order = case, window = 5,
                better = "lower", target = 70, target_hold = 3,

```

```

        proficiency_method = "target",
        show = FALSE, plot = FALSE)
m10$tables$Proficiency
m10$tables$Proficiency_evidence

# 11. Report the first qualifying window rather than the confirmation window.
m11 <- tablearn(time, data = d, order = case, window = 5,
        better = "lower", target = 70, target_hold = 3,
        proficiency_method = "target",
        proficiency_case_rule = "first_stable",
        show = FALSE, plot = FALSE)

# 12. Manual proficiency and prespecified study phases.
m12 <- tablearn(time, data = d, order = case, window = 5,
        proficiency_method = "manual", proficiency_case = 60,
        phase_method = "manual", phase_breaks = c(25, 59),
        phase_names = c("Learning", "Consolidation", "Proficiency"),
        show = FALSE, plot = FALSE)
m12$tables$Phase_classification

# 13. Binary outcome. Numeric 0/1 is recognized automatically; event = 1 is
# explicit and makes the scientific meaning clear.
m13 <- tablearn(complication, data = d, order = case, event = 1,
        better = "lower", window = 20,
        window_ci_method = "wilson",
        show = FALSE, plot = FALSE)
m13$tables$Current_performance

# 14. Count outcome.
m14 <- tablearn(errors, data = d, order = case, type = "count",
        better = "lower", window = 10,
        show = FALSE, plot = FALSE)

# 15. Higher values can represent better performance.
m15 <- tablearn(score, data = d, order = case, better = "higher",
        target = 80, target_hold = 3,
        show = FALSE, plot = FALSE)

# 16. Conventional deviation CUSUM for a continuous outcome.
m16 <- tablearn(time, data = d, order = case, target = 75,
        better = "lower", cusum = TRUE,
        plot_type = "auto", show = FALSE, plot = FALSE)
m16$tables$Sequential_monitoring

# 17. Binary likelihood-ratio CUSUM.
m17 <- tablearn(complication, data = d, order = case, event = 1,
        better = "lower", target = 0.10,
        cusum = TRUE, cusum_method = "llr", cusum_alt = 0.20,
        show = FALSE, plot = FALSE)

# 18. LC-CUSUM: evidence that an acceptable failure rate has been reached.
m18 <- tablearn(complication, data = d, order = case, event = 1,
        better = "lower", window = 20,

```

```

        lccusum = TRUE, p_acceptable = 0.10,
        p_unacceptable = 0.25,
        proficiency_method = "lccusum",
        show = FALSE, plot = FALSE)
m18$tables$Proficiency

# 19. RA-CUSUM with externally supplied case-specific expected risks.
m19 <- tablearn(complication, data = d, order = case, event = 1,
        better = "lower", window = 20,
        racusum = TRUE, expected = expected_risk,
        racusum_or = 2,
        show = FALSE, plot = FALSE)

# 20. RA-CUSUM can estimate expected risk from covariates using base glm().
# For prospective monitoring, an external/pre-specified risk model is preferred.
m20 <- suppressWarnings(tablearn(
        complication, data = d, order = case, event = 1,
        racusum = TRUE, adjust = vars(complexity, c.age), racusum_or = 2,
        show = FALSE, plot = FALSE
))

# 21. Separate learning curves by operator/surgeon.
m21 <- tablearn(time, data = d, order = case, operator = surgeon,
        window = 8, operator_display = "facet",
        show = FALSE, plot = FALSE)
m21$tables$Current_performance

# 22. Overlay operators in the same graph.
m22 <- tablearn(time, data = d, order = case, operator = surgeon,
        window = 8, operator_display = "overlay",
        show = FALSE, plot = FALSE)

# 23. Window-size sensitivity analysis.
m23 <- tablearn(time, data = d, order = case, window = 5,
        sensitivity = TRUE,
        window_sensitivity = c(3, 5, 10, 20),
        show = FALSE, plot = FALSE)
m23$tables$Window_sensitivity

# 24. Multiple outcomes in one command.
m24 <- tablearn(vars(time, complication), data = d, order = case,
        event = 1, window = 10,
        show = FALSE, plot = FALSE)
names(m24$outcomes)

# 25. Use procedure date on the x-axis instead of consecutive case number.
m25 <- tablearn(time, data = d, order = date, x_axis = "order",
        window = 7, show = FALSE, plot = FALSE)

# 26. Interpretive prose is opt-in; default is FALSE.
m26 <- tablearn(time, data = d, order = case,
        interpretation = TRUE, show = FALSE, plot = FALSE)
m26$interpretation

```

```

# 27. Vietnamese generated interpretation/labels.
m27 <- tablearn(time, data = d, order = case, lang = "vi",
                interpretation = TRUE, show = FALSE, plot = FALSE)

# 28. Conservative auto-detection: a numeric variable with two values other
# than 0/1 remains continuous unless event/type explicitly says binary.
d2 <- data.frame(case = 1:20, value = c(rep(100, 10), rep(50, 10)))
m28 <- tablearn(value, data = d2, order = case,
                show = FALSE, plot = FALSE)
m28$settings$type

# 29. All publication-ready tables are directly accessible.
names(m10$tables)
summary(m10)$tables

# 30. Plot methods work even when ggplot2 is not installed because tablearn()
# has a base-R plotting fallback. These also appear inside the Viewer report.
if (interactive()) {
  plot(m10, type = "learning")
  m30 <- tablearn(complication, data = d, order = case, event = 1,
                  lccusum = TRUE, p_acceptable = .10,
                  p_unacceptable = .25, plot_type = "both")
  plot(m30, type = "cusum")
}

# 31. High-level publication styling. Advanced ggplot styling is used when
# ggplot2 is installed; the Viewer/base-R figure remains available otherwise.
if (interactive()) {
  m31 <- tablearn(
    time, data = d, order = case, window = 5, target = 70,
    plot_opts = list(
      raw = list(alpha = .15, size = 1.0),
      window = list(color = "#1F5A94", line_width = 1.2),
      smooth = list(color = "#B23A48", line_width = 1.4),
      proficiency = list(color = "#00796B"),
      phase = list(alpha = .08),
      current = list(fill = "#FFD166"),
      theme = list(base_size = 12, grid_minor = FALSE)
    )
  )
}

} # end full interactive example catalogue

```

Description

Performs publication-ready longitudinal or repeated-measures analysis from either long or wide data. `tablong()` is designed for the usual biomedical workflow: describe each time point, test overall time and group effects, test the time-by-group interaction, estimate clinically interpretable contrasts, retain fitted models for advanced use, and optionally create a longitudinal profile plot and a cautious interpretation table.

Usage

```
tablong(  
  data = NULL,  
  vars,  
  time = NULL,  
  id = NULL,  
  by = NULL,  
  ref = NULL,  
  event = NULL,  
  adjusted = NULL,  
  gee = FALSE,  
  ar1 = FALSE,  
  slope = FALSE,  
  or = FALSE,  
  rr = FALSE,  
  pr = FALSE,  
  count = FALSE,  
  exposure = NULL,  
  change = TRUE,  
  pairwise = FALSE,  
  adjust = "none",  
  missing = FALSE,  
  level = 0.95,  
  digit = 1,  
  p_digit = 3,  
  effect_digit = 2,  
  bold_p = TRUE,  
  p_bold = 0.05,  
  diagnostics = FALSE,  
  diagnosis = FALSE,  
  interpretation = FALSE,  
  plot = FALSE,  
  plot_args = list(),  
  name = FALSE,  
  title = NULL,  
  file = NULL,  
  raw = FALSE,  
  show = TRUE  
)
```

Arguments

<code>data</code>	Optional data frame. If NULL, the active data selected by <code>usedf()</code> are used.
<code>vars</code>	Outcome specification created by <code>vars()</code> . In long data, several outcomes can be analyzed in one call. Continuous outcomes use the <code>c.</code> prefix, <code>q.</code> requests median (IQR) descriptive display, and <code>f.</code> requests a fuller continuous summary. An unprefixed two-level variable is treated as binary. In wide data, the variables in <code>vars()</code> are repeated measurements of the same outcome.
<code>time</code>	In long data, an unquoted time variable. Use <code>c.timevar</code> for a continuous linear time effect or <code>b2.timevar</code> , <code>b3.timevar</code> , etc. to select the categorical reference level. In wide data, provide display labels such as <code>c("Baseline", "Month 3", "Month 6")</code> ; when omitted, the repeated variable names are used as time labels.
<code>id</code>	Optional subject identifier. In wide data it is optional because each source row represents one subject. In long data, repeated IDs trigger longitudinal analysis with within-subject correlation. If no ID is supplied, or IDs do not repeat across time, observations are analyzed as repeated cross-sectional samples.
<code>by</code>	Optional grouping variable, for example treatment group. Prefix <code>b2.</code> , <code>b3.</code> , etc. selects the reference group.
<code>ref</code>	Optional categorical reference time label. This overrides a <code>bN.</code> prefix supplied in <code>time</code> .
<code>event</code>	Event level for binary outcomes. A single value applies to every binary outcome; a named vector can specify a different event for each outcome, for example <code>event = c(controlled = "Yes", admitted = "Yes")</code> .
<code>adjusted</code>	Optional adjustment variables created by <code>vars()</code> . Use the same R4VN prefixes as elsewhere, for example <code>vars(c.age, sex, b2.site)</code> .
<code>gee</code>	Logical. Request a population-average marginal model for repeated data. By default R4VN uses working-independence regression with subject- clustered robust sandwich standard errors calculated internally. No extra package is required. Set <code>ar1 = TRUE</code> to request AR(1) GEE through optional package <code>geepack</code> when it is installed.
<code>ar1</code>	Logical. Request an AR(1) working correlation for a marginal GEE. This advanced option uses optional package <code>geepack</code> . If it is unavailable, <code>tablong()</code> warns and falls back to working-independence cluster-robust inference instead of stopping the analysis.
<code>slope</code>	Logical. With a mixed model and continuous time (<code>time = c.month</code>), include a subject-specific random linear time slope in addition to the random intercept.
<code>or, rr, pr</code>	Logical effect switches for binary outcomes. Odds ratio is the default when none is selected. <code>rr = TRUE</code> reports risk ratios and <code>pr = TRUE</code> reports prevalence ratios using modified Poisson regression with robust variance; repeated subjects use subject-clustered robust variance. No external sandwich/GEE package is needed unless <code>ar1 = TRUE</code> is requested. Only one switch may be TRUE.
<code>count</code>	Logical. Treat numeric outcomes as non-negative counts and fit Poisson models. Count outcomes report incidence-rate ratios (IRR).
<code>exposure</code>	Optional positive exposure/person-time variable for count models. In wide data it may also be <code>vars(exp0, exp1, ...)</code> , with one exposure variable per repeated count variable.

change	Logical. For categorical time, report change from the reference time. With exactly two groups, also report the difference in change, i.e. the usual difference-in-differences contrast. Default TRUE.
pairwise	Logical. Calculate all available time and group pairwise contrasts and retain them in <code>\$contrasts</code> and <code>\$contrasts_table</code> . The main publication table remains compact. Default FALSE.
adjust	Multiplicity adjustment applied to contrast p-values. Any method accepted by <code>p.adjust()</code> may be used, including "none", "holm", "bonferroni", and "BH".
missing	Logical. Append cell-specific n to continuous/count summary cells. Binary cells always show event/total. Detailed observed and missing counts are always available in <code>\$descriptive</code> .
level	Confidence level, default 0.95.
digit	Decimal places for descriptive summaries.
p_digit	Decimal places for p-values.
effect_digit	Decimal places for model effects and confidence intervals.
bold_p	Logical. Bold p-values below <code>p_bold</code> in the HTML Viewer.
p_bold	Threshold used by <code>bold_p</code> .
diagnostics	Logical. Include the compact model-diagnostics table in the HTML Viewer. Diagnostics are always retained in <code>\$diagnostics</code> ; default FALSE keeps the primary Viewer concise.
diagnosis	Logical singular alias for diagnostics, provided for consistency with other R4VN regression commands. Default FALSE. When explicitly supplied, it overrides diagnostics; omitting it preserves backward-compatible use of diagnostics.
interpretation	Logical. Add a cautious deterministic interpretation table. The default is FALSE. The interpretation emphasizes the time-by-group interaction when present and does not replace substantive or clinical interpretation by the researcher.
plot	Logical. Create an observed longitudinal profile plot with 95% confidence intervals, include the same plot directly in the HTML Viewer, display it in the Plot pane, and store its specification in <code>\$graph</code> and <code>\$plots\$trajectory</code> . Plotting uses base R graphics; ggplot2 is not required. Default FALSE.
plot_args	Named list controlling the profile plot. Supported entries include <code>title</code> , <code>xlab</code> , <code>ylab</code> , <code>ci</code> , <code>line_width</code> , <code>point_size</code> , <code>base_size</code> , <code>legend_position</code> , <code>font_family</code> , <code>colors</code> , <code>point_shapes</code> , <code>line_types</code> , and <code>grid</code> . Generic sans is the default font for reliable display in RStudio Viewer, browsers, Windows, macOS, and Linux.
name	Logical. Display the original variable name after its variable label in the main table.
title	Optional table title.
file	Optional HTML file path. When omitted, a temporary HTML file is created. This file is the formatted Viewer report, not a replacement for <code>tabexport()</code> .
raw	Logical retained for backward compatibility. Raw models, tests, contrasts, standardized long data, and reporting tables are always retained in the returned object.
show	Logical. Open the formatted HTML report in the Viewer/browser. Default TRUE.

Details

The interface follows the R4VN principle of keeping routine analysis simple. In most studies the essential call is only `vars()`, `time`, `id`, and optionally `by`. Repeated continuous outcomes use a random-intercept model when R's recommended `nlme` package is available. Repeated binary/count outcomes use marginal regression with subject-clustered robust standard errors calculated internally by R4VN, so routine analyses need no extra package.

Data format. If `time` names a column in `data`, input is treated as long. If `vars()` contains multiple repeated variables and `time` is a vector of labels (or omitted), input is treated as wide and is reshaped internally. The original data frame is never modified.

Continuous outcomes. Repeated subjects use a random-intercept linear mixed model through R's recommended `nlme` package when available. `slope = TRUE` adds a random linear time slope when time is continuous. If `nlme` is unavailable, R4VN falls back to a marginal linear model with subject-clustered robust standard errors. Repeated cross-sectional data use ordinary linear models. `gee = TRUE` explicitly requests the marginal model.

Binary outcomes. The default effect is an odds ratio from logistic regression. For repeated subjects, R4VN calculates subject-clustered robust standard errors internally. `rr = TRUE` and `pr = TRUE` use modified Poisson regression with robust variance and report RR or PR. This avoids requiring `lme4`, `sandwich`, or `geepack` for routine binary longitudinal analysis.

Count outcomes. `count = TRUE` fits a Poisson model and reports IRR. Repeated subjects use subject-clustered robust standard errors calculated internally. Supplying `exposure` adds `offset(log(exposure))` and the observed plot is an incidence rate per one person-time unit.

Categorical time. The model includes `time`, `group` when supplied, and the time-by-group interaction. The main table reports observed summaries at each time. With two groups it also reports the between-group effect at each time, within-group change from the reference time, and the difference in change. The omnibus `Time x group` p-value is the formal test that temporal changes differ between groups.

Continuous time. `time = c.month` estimates change per one time unit. With a `group` variable, group-specific slopes and their difference are returned.

More than two groups. The main table remains intentionally compact and shows omnibus tests. Set `pairwise = TRUE` to obtain all model-based pairwise comparisons in `$contrasts_table`; use `adjust` to control multiplicity.

Descriptive prefixes. `q.` and `f.` affect the observed descriptive summary only. Inferential effects remain based on the selected mean model; they do not fit median regression.

Missing values. Each model uses observations complete for that outcome, time, ID/group, exposure if required, and adjustment variables. Descriptive counts are retained separately so that missingness and attrition can be reviewed before publication.

Package dependencies. Routine `tablong()` analyses and plots are designed to run with base/recommended R only. `nlme` is used for continuous random-effects models and is bundled with standard R installations. `geepack` is optional and used only when an AR(1) GEE is explicitly requested. `lme4`, `sandwich`, `broom`, and `ggplot2` are not required by `tablong()`.

Returned reporting contract. For programmatic reuse, the object includes a flat main table plus descriptive, omnibus-test, contrast, diagnostics, interpretation, plot, model, and metadata components. The object also inherits from `r4vn_tab`, so existing `tabexport()` workflows continue to work.

Value

Invisibly returns an object of class `c("r4vn_tablong", "r4vn_tab")`. Important components are: `$data` (main publication table), `$descriptive`, `$tests` and `$tests_table`, `$contrasts` and `$contrasts_table`, `$diagnostics`, `$interpretation`, `$tables`, `$graph`, `$plots`, `$models`, `$long_data`, `$metadata`, `$html`, `$file`, `$results`, and `$call`.

See Also

`vars()`, `tab()`, `tabexport()`, `usedf()`

Other R4VN tables: [tab\(\)](#), [tabexport\(\)](#), [tabforest\(\)](#), [tabmeta\(\)](#), [tabmulti\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [tabsurvey\(\)](#), [vars\(\)](#)

Examples

```
# -----
# 1. Repeated cross-sectional continuous outcome: no optional package needed
# -----
set.seed(11)
d <- data.frame(
  period = factor(rep(c("Before", "After"), each = 80),
    levels = c("Before", "After")),
  group = factor(rep(rep(c("Control", "Intervention"), each = 40), 2)),
  age = rnorm(160, 45, 10)
)
d$score <- 50 + 2 * (d$period == "After") +
  5 * (d$group == "Intervention") +
  4 * (d$period == "After" & d$group == "Intervention") +
  0.15 * d$age + rnorm(160, 0, 8)

z <- tablong(
  d, vars = vars(c.score), time = period, by = group,
  adjusted = vars(c.age), show = FALSE
)
z$data
z$tests_table
z$contrasts_table

# -----
# 2. The same analysis with interpretation and a publication profile plot
# -----
z2 <- tablong(
  d, vars = vars(c.score), time = period, by = group,
  adjusted = vars(c.age), interpretation = TRUE,
  plot = TRUE, show = FALSE
)
z2$interpretation
z2$diagnostics
plot(z2)

# -----
```

```

# 3. No comparison group: change over time only
# -----
z_time <- tablong(
  d, vars = vars(c.score), time = period,
  adjusted = vars(c.age), show = FALSE
)
z_time$tests_table

# -----
# 4. Change the reference time by value or by bN. prefix
# -----
z_ref1 <- tablong(d, vars = vars(c.score), time = period,
  by = group, ref = "After", show = FALSE)
z_ref2 <- tablong(d, vars = vars(c.score), time = b2.period,
  by = group, show = FALSE)

# -----
# 5. Median/IQR or full descriptive display, while inference remains a
#    mean model
# -----
z_median <- tablong(d, vars = vars(q.score), time = period,
  by = group, show = FALSE)
z_full <- tablong(d, vars = vars(f.score), time = period,
  by = group, missing = TRUE, show = FALSE)

# -----
# 6. Long repeated data: linear mixed model
# -----
set.seed(12)
n_subject <- 60
dl <- expand.grid(
  id = seq_len(n_subject),
  visit = factor(c("Baseline", "Month 3", "Month 6"),
    levels = c("Baseline", "Month 3", "Month 6"))
)
dl <- dl[order(dl$id, dl$visit), ]
trt <- factor(sample(c("Control", "Intervention"), n_subject, TRUE),
  levels = c("Control", "Intervention"))
dl$treatment <- rep(trt, each = 3)
u <- rnorm(n_subject, 0, 6)
dl$sbp <- 140 + u[dl$id] - 3 * (dl$visit == "Month 3") -
  5 * (dl$visit == "Month 6") -
  4 * (dl$treatment == "Intervention" & dl$visit == "Month 6") +
  rnorm(nrow(dl), 0, 5)

mixed <- tablong(
  dl, vars = vars(c.sbp), time = visit, id = id,
  by = treatment, diagnostics = TRUE, show = FALSE
)
mixed$models[[1]]
mixed$diagnostics

# -----

```

```

# 7. Wide repeated data: internally converted to long format
# -----
dw <- data.frame(
  id = seq_len(n_subject), treatment = trt,
  sbp0 = rnorm(n_subject, 140, 10)
)
dw$sbp3 <- dw$sbp0 - 3 + rnorm(n_subject, 0, 4)
dw$sbp6 <- dw$sbp0 - 5 - 4 * (dw$treatment == "Intervention") +
  rnorm(n_subject, 0, 4)
wide <- tablong(
  dw, vars = vars(c.sbp0, c.sbp3, c.sbp6),
  time = c("Baseline", "Month 3", "Month 6"),
  id = id, by = treatment, show = FALSE
)
wide$input_format
head(wide$long_data)

# -----
# 8. Binary repeated outcome: marginal logistic model with clustered SE and OR
# -----
p <- plogis(-1 + 0.4 * (dl$visit == "Month 6") +
  0.6 * (dl$treatment == "Intervention"))
dl$controlled <- factor(rbinom(nrow(dl), 1, p),
  levels = c(0, 1), labels = c("No", "Yes"))
binary_or <- tablong(
  dl, vars = vars(controlled), time = visit, id = id,
  by = treatment, event = "Yes", show = FALSE
)
binary_or$contrasts_table

# -----
# 9. Continuous time and random slope
# -----
ds <- expand.grid(id = seq_len(50), month = c(0, 3, 6, 12))
ds <- ds[order(ds$id, ds$month), ]
ds$group <- factor(rep(sample(c("Control", "Intervention"), 50, TRUE), each = 4))
b0 <- rnorm(50, 0, 5)
b1 <- rnorm(50, 0, 0.15)
ds$score <- 50 + b0[ds$id] + (-0.3 + b1[ds$id]) * ds$month -
  0.25 * ds$month * (ds$group == "Intervention") + rnorm(nrow(ds), 0, 3)
slope_fit <- tablong(
  ds, vars = vars(c.score), time = c.month, id = id,
  by = group, slope = TRUE, show = FALSE
)
slope_fit$contrasts_table

# -----
# 10. Repeated count outcome with person-time offset -> IRR
# -----
dc <- dl
dc$person_time <- runif(nrow(dc), 0.8, 1.2)
rate <- exp(0.2 + 0.2 * (dc$visit == "Month 6") -
  0.3 * (dc$treatment == "Intervention" & dc$visit == "Month 6"))

```

```

dc$events <- rpois(nrow(dc), rate * dc$person_time)
count_fit <- tablong(
  dc, vars = vars(c.events), time = visit, id = id,
  by = treatment, count = TRUE, exposure = person_time,
  show = FALSE
)
count_fit$contrasts_table

# -----
# 11. RR/PR without extra packages; optional AR(1) GEE
# -----
set.seed(13)
n_subject <- 70
dg <- expand.grid(
  id = seq_len(n_subject),
  visit = factor(c("Baseline", "Month 6"),
    levels = c("Baseline", "Month 6"))
)
dg <- dg[order(dg$id, dg$visit), ]
dg$treatment <- factor(
  rep(sample(c("Control", "Intervention"), n_subject, TRUE), each = 2),
  levels = c("Control", "Intervention")
)
p <- plogis(-1 + 0.3 * (dg$visit == "Month 6") +
  0.4 * (dg$treatment == "Intervention"))
dg$controlled <- factor(rbinom(nrow(dg), 1, p),
  levels = c(0, 1), labels = c("No", "Yes"))

fit_rr <- tablong(
  dg, vars = vars(controlled), time = visit, id = id,
  by = treatment, event = "Yes", rr = TRUE, show = FALSE
)
fit_pr <- tablong(
  dg, vars = vars(controlled), time = visit, id = id,
  by = treatment, event = "Yes", pr = TRUE,
  show = FALSE
)
fit_rr$contrasts_table
fit_pr$contrasts_table

# AR(1) is advanced and uses geepack only when explicitly requested.
if (requireNamespace("geepack", quietly = TRUE)) {
  fit_pr_ar1 <- tablong(
    dg, vars = vars(controlled), time = visit, id = id,
    by = treatment, event = "Yes", pr = TRUE, gee = TRUE, ar1 = TRUE,
    show = FALSE
  )
}

# -----
# 12. Three or more groups: keep main table compact, request pairwise tests
# -----
set.seed(14)

```

```

dm <- data.frame(
  period = factor(rep(c("Baseline", "Follow-up"), each = 90),
    levels = c("Baseline", "Follow-up")),
  arm = factor(rep(rep(c("A", "B", "C"), each = 30), 2))
)
dm$score <- rnorm(nrow(dm), 50 + 2 * (dm$period == "Follow-up") +
  2 * (dm$arm == "B") + 4 * (dm$arm == "C"), 7)
multi_arm <- tablong(
  dm, vars = vars(c.score), time = period, by = arm,
  pairwise = TRUE, adjust = "holm", show = FALSE
)
multi_arm$tests_table
multi_arm$contrasts_table

# -----
# 13. Several outcomes in one long-data analysis
# -----
d$positive <- factor(
  rbinom(nrow(d), 1, plogis(-1 + 0.5 * (d$period == "After"))),
  levels = c(0, 1), labels = c("No", "Yes")
)
multi_outcome <- tablong(
  d, vars = vars(c.score, positive), time = period,
  by = group, event = "Yes", show = FALSE
)
multi_outcome$data
multi_outcome$descriptive

# -----
# 14. Export remains compatible with ordinary R4VN table workflows
# -----
export_data <- tabexport(z)
head(export_data)

# -----
# 15. Missing outcomes/attrition: inspect counts before publication
# -----
d_missing <- d
d_missing$score[c(2, 7, 21, 100)] <- NA
miss_fit <- tablong(
  d_missing, vars = vars(c.score), time = period, by = group,
  missing = TRUE, diagnostics = TRUE, show = FALSE
)
miss_fit$descriptive
miss_fit$diagnostics

# -----
# 16. Several binary outcomes can use a named event vector
# -----
d$admitted <- factor(
  rbinom(nrow(d), 1, plogis(-1.4 + 0.4 * (d$period == "After"))),
  levels = c(0, 1), labels = c("No", "Yes")
)

```

```

binary_set <- tablong(
  d, vars = vars(positive, admitted), time = period, by = group,
  event = c(positive = "Yes", admitted = "Yes"), show = FALSE
)
binary_set$tests_table

# -----
# 17. Continuous Gaussian GEE with AR(1) working correlation
# -----
if (requireNamespace("geepack", quietly = TRUE)) {
  set.seed(17)
  dg2 <- expand.grid(id = seq_len(60), month = c(0, 3, 6, 12))
  dg2 <- dg2[order(dg2$id, dg2$month), ]
  dg2$group <- factor(rep(sample(c("Control", "Intervention"), 60, TRUE), each = 4))
  dg2$score <- 55 - 0.2 * dg2$month -
    0.15 * dg2$month * (dg2$group == "Intervention") + rnorm(nrow(dg2), 0, 5)
  gee_cont <- tablong(
    dg2, vars = vars(c.score), time = c.month, id = id, by = group,
    gee = TRUE, ar1 = TRUE, show = FALSE
  )
  gee_cont$contrasts_table
}

# -----
# 18. Wide count data can supply one person-time variable per time point
# -----
set.seed(18)
nw <- 50
wc <- data.frame(
  id = seq_len(nw),
  group = factor(sample(c("Control", "Intervention"), nw, TRUE)),
  pt0 = runif(nw, 0.8, 1.2),
  pt6 = runif(nw, 0.8, 1.2)
)
wc$event0 <- rpois(nw, 1.2 * wc$pt0)
wc$event6 <- rpois(nw,
  exp(log(1.2) - 0.25 * (wc$group == "Intervention")) * wc$pt6)
wide_count <- tablong(
  wc, vars = vars(c.event0, c.event6),
  time = c("Baseline", "Month 6"), id = id, by = group,
  count = TRUE, exposure = vars(pt0, pt6), show = FALSE
)
wide_count$contrasts_table

# -----
# 19. Replot an existing result without refitting the statistical model
# -----
plot(z, ci = FALSE, title = "Observed longitudinal profile",
  base_size = 12, legend_position = "right")

```

tabmachine	<i>Comprehensive machine-learning analysis with publication-ready reporting</i>
------------	---

Description

`tabmachine()` is the comprehensive machine-learning command in R4VN. It is designed for health and biomedical research where the user needs a complete, reproducible workflow rather than only a fitted prediction model.

The function can detect the prediction task, split development data, preprocess predictors, handle missing values, encode categorical predictors, detect problematic predictors, standardize predictors when required, transform skewed numeric variables when explicitly requested, address class imbalance, select predictors, tune candidate algorithms, perform cross-validation, compare models, select a final model, determine a classification threshold using training data only, evaluate the untouched test set, calculate confidence intervals for performance measures whenever a defensible interval is implemented, assess calibration and clinical utility, calculate variable importance, and generate prediction-ready model objects.

The central R4VN principle is that automation must remain transparent. "auto" may choose an analysis action, but every action is stored in the returned object and shown in the report. Pre-processing, feature selection, class balancing, tuning, and threshold optimization are learned from training data only. The held-out test data are not used to make those decisions.

Usage

```
tabmachine(
  outcome,
  x = NULL,
  data = NULL,
  exclude = NULL,
  task = c("auto", "binary", "multiclass", "regression"),
  event = NULL,
  preprocess = c("auto", "none"),
  missing = c("auto", "median", "mean", "mode", "complete"),
  missing_max = 0.5,
  encode = c("auto", "dummy"),
  standardize = c("auto", "none", "z", "minmax", "robust"),
  transform = c("none", "auto", "log", "yeojohnson"),
  outlier = c("none", "detect", "winsor", "robust"),
  corr = "auto",
  feature = c("none", "auto", "polynomial", "interaction", "all"),
  degree = 2,
  reduce = c("none", "auto", "pca"),
  variance = 0.95,
  select = c("auto", "none", "filter", "lasso", "stepwise", "importance", "rfe",
    "boruta", "compare"),
  nfeatures = "auto",
```

```

simplify = TRUE,
simplify_tol = 0.01,
balance = c("auto", "none", "weight", "up", "down", "smote", "adasyn", "rose",
  "compare"),
balance_target = 0.5,
neighbors = 5,
split = 0.8,
folds = 10,
repeats = 1,
nested = FALSE,
seed = NULL,
method = "auto",
tune = TRUE,
tune_n = 10,
metric = "auto",
threshold = "auto",
target_sens = 0.9,
target_spec = 0.9,
ci = TRUE,
ci_level = 0.95,
boot = 1000,
calibration = TRUE,
decision = TRUE,
decision_thresholds = seq(0.01, 0.99, 0.01),
learning = FALSE,
importance = TRUE,
importance_repeats = 20,
explain = TRUE,
shap = FALSE,
pdp = FALSE,
validation = NULL,
predict = NULL,
id = NULL,
show = TRUE,
plot = TRUE,
plot_display = "auto",
plot_args = list(),
strict = FALSE,
console = FALSE,
digit = 3,
title = NULL,
ai = FALSE,
...
)

```

Arguments

outcome	Outcome variable. Supply an unquoted variable name or a one-element character name. Binary, multiclass, and continuous outcomes are supported.
---------	--

x	Candidate predictors. Use <code>vars(age, sex, bmi)</code> , a character vector, a single bare variable, or <code>.</code> to use every variable except outcome and exclude. If omitted, all other variables are used.
data	Optional data frame. If omitted, the active R4VN data selected by <code>usedf()</code> or <code>opendata(..., active = TRUE)</code> is used.
exclude	Optional predictors to exclude, supplied as <code>vars(...)</code> , a bare variable, or a character vector. Typical examples are identifiers, names, dates that would leak future information, or variables unavailable at the intended prediction time.
task	Prediction task: "auto" (default), "binary", "multiclass", or "regression". In auto mode, an outcome with two observed levels is binary; a factor/character outcome or a low-cardinality integer outcome is multiclass; otherwise a numeric outcome is regression.
event	Positive/event level for binary classification. If omitted, R4VN recognizes common positive encodings such as 1, TRUE, Yes, Positive, Case, or Co; otherwise the second observed level is used. The chosen event is always reported.
preprocess	Preprocessing policy. "auto" performs safe structural checks, imputation, factor-level alignment, dummy encoding, zero/near-zero variance removal, and optional correlation filtering. "none" disables automatic structural filtering but still creates a usable model matrix.
missing	Missing-value handling for predictors: "auto", "median", "mean", "mode", or "complete". "auto" uses median for numeric and mode for categorical predictors. Imputation values are estimated on each training fold and then applied to its validation fold. Outcome missingness is never imputed.
missing_max	Maximum allowed proportion missing in a candidate predictor before automatic structural filtering removes it. Default 0.50.
encode	Encoding of categorical predictors. Currently "auto" and "dummy" use treatment/dummy coding through a training-derived <code>model.matrix</code> blueprint. New/unseen levels in validation data are mapped to the training reference fallback and reported.
standardize	Standardization policy: "auto", "none", "z", "minmax", or "robust". In auto mode, z-standardization is applied only inside algorithms that materially benefit from scaling (penalized models, SVM, KNN, and multinomial neural optimization). Tree-based models receive the unscaled encoded matrix.
transform	Numeric transformation: "none" (default), "auto", "log", or "yeojohnson". "auto" is intentionally conservative and currently performs no transformation unless a future R4VN rule explicitly justifies one. "log" uses a training-derived shift when necessary. "yeojohnson" estimates lambda on the training data by profile likelihood.
outlier	Outlier policy: "none" (default), "detect", "winsor", or "robust". Detection uses training-data IQR rules and never deletes rows. "winsor" caps numeric values at the 1st and 99th training percentiles; "robust" caps at median +/- 5 MAD. Cut points are then reused unchanged for validation/test data.
corr	Correlation filtering for numeric candidate predictors. "auto" uses 0.95, a numeric value in (0,1) supplies a custom absolute-correlation threshold, and FALSE/"none" disables correlation filtering. Filtering is learned separately inside each training fold.

feature	Feature engineering: "none" (default), "auto", "polynomial", "interaction", or "all". "polynomial" adds powers of numeric predictors up to degree; "interaction" adds pairwise products among numeric predictors; "all" adds both. "auto" is intentionally conservative and currently behaves as "none". All engineered features are created from a fold-specific training blueprint.
degree	Highest polynomial degree for numeric feature engineering. Default 2; values 2 or 3 are supported.
reduce	Dimensionality reduction: "none" (default), "auto", or "pca". PCA is learned only on the training fold after encoding/feature engineering. "auto" uses PCA only in strongly high-dimensional settings (more encoded features than training observations and at least 50 encoded features); otherwise it remains off to preserve clinical interpretability.
variance	Target cumulative variance retained by PCA. Default 0.95.
select	Feature-selection strategy: "auto", "none", "filter", "lasso", "stepwise", "importance", "rfe", "boruta", or "compare". "auto" deliberately resolves to the native R4VN "filter" strategy so the default workflow never depends on an optional selection package. LASSO uses glmnet when explicitly requested. Boruta uses the optional Boruta package. "compare" compares available strategies by cross-validation on the development training set and does not inspect the held-out test set.
nfeatures	Number of predictors to retain for importance/RFE selection, or "auto". With "auto", R4VN chooses a compact candidate size based on the training sample size and number of available encoded predictors.
simplify	Logical; after choosing the best algorithm, search for a smaller predictor set whose development cross-validated performance is within simplify_tol of the larger selected model. This step is training- only. Default TRUE.
simplify_tol	Maximum acceptable loss in the primary metric when preferring a smaller model. For metrics where larger is better this is an absolute decrease; for RMSE/MAE it is an absolute increase. Default 0.01.
balance	Class-imbalance handling for binary classification: "auto", "none", "weight", "up", "down", "smote", "adasyn", "rose", or "compare". Balancing is applied only to the training portion of each resample. Test/validation observations are never resampled. "auto" uses class weighting when the minority class is below 20 percent and otherwise leaves the sample unchanged. "compare" compares available approaches on development cross-validation.
balance_target	Target minority proportion after sampling. Default 0.50.
neighbors	Number of nearest neighbors for native SMOTE/ADASYN. Default 5.
split	Development/test split. A single number such as 0.80 means 80% training and 20% untouched test data. FALSE uses all development data for model development (appropriate when a separate validation data set is supplied). A length-three vector such as c(.70, .15, .15) creates training, internal validation, and test partitions; the internal validation portion is combined with training for final refitting after model decisions are completed, while the test portion remains untouched.
folds	Number of cross-validation folds in the development training sample. Default 10. Classification folds are stratified when possible.

repeats	Number of repeated cross-validation repetitions. Default 1.
nested	Logical; if TRUE, hyperparameter tuning is repeated within each outer cross-validation fold. This is computationally expensive but gives a less optimistic development estimate. Regardless of this option, the final held-out test evaluation remains untouched by tuning.
seed	Optional random seed used for splitting, resampling, tuning, and bootstrap. The default NULL does not set a seed.
method	Algorithms to fit. "auto" deliberately uses a compact low-dependency R4VN core set so that a routine analysis does not require installation of an ML ecosystem: logistic regression plus a decision tree for binary outcomes, linear regression plus a decision tree for continuous outcomes, and multinomial regression plus a decision tree and R4VN-native KNN for multiclass outcomes. The native KNN fallback keeps multiclass auto usable even if recommended modelling packages are unavailable. "all" tries every supported engine that is already installed. Optional engines are never installed automatically. Methods may also be supplied explicitly: "logistic", "linear", "multinom", "lasso", "ridge", "elastic", "tree", "rf", "xgb", "svm", "knn", "naive".
tune	Hyperparameter tuning. TRUE/"grid" uses compact clinically practical grids; "random" samples candidate combinations; FALSE uses defaults. Penalized glmnet engines select lambda internally by CV.
tune_n	Maximum random-search combinations when tune = "random".
metric	Primary model-selection metric. "auto" uses ROC-AUC for most binary problems, PR-AUC when the event is uncommon, macro F1 for multiclass, and RMSE for regression. Other supported names include auc, pr_auc, accuracy, balanced_accuracy, sensitivity, specificity, f1, brier, logloss, macro_f1, weighted_f1, macro_auc, macro_pr_auc, rmse, mae, r2, and mape.
threshold	Binary classification threshold. A numeric value between 0 and 1 fixes the cut-off. "auto"/"youden" maximizes sensitivity + specificity - 1 using out-of-fold predictions from the development training set. "f1" maximizes F1. "sens" chooses the most specific threshold achieving at least target_sens; "spec" chooses the most sensitive threshold achieving at least target_spec. The test set is never used to choose the threshold.
target_sens	Target sensitivity used when threshold = "sens".
target_spec	Target specificity used when threshold = "spec".
ci	Logical; calculate 95% confidence intervals (or the level supplied by ci_level) for performance measures whenever an implemented interval is statistically meaningful. Default TRUE.
ci_level	Confidence level, default 0.95.
boot	Number of bootstrap replicates for performance measures whose interval has no preferred closed-form method. Default 1000. For final publication analyses, 2000 or more may be preferred when runtime permits.
calibration	Logical; for binary classification, calculate calibration intercept, calibration slope, Brier score, and calibration-curve data.
decision	Logical; for binary classification, calculate decision-curve net benefit for the final model over decision_thresholds.

decision_thresholds	Probability thresholds for decision-curve analysis. Default <code>seq(.01, .99, .01)</code> .
learning	Logical; calculate training-size learning-curve summaries for the final model. Default FALSE because it can be computationally expensive.
importance	Logical; calculate permutation importance for the final model. Default TRUE. Importance is grouped back to original predictor names when dummy variables were created.
importance_repeats	Number of repeated permutations used to stabilize permutation importance. Default 20.
explain	Logical; retain explanation data and display the principal importance/calibration information in the report. Default TRUE.
shap	Logical; calculate SHAP-like contribution output when a supported engine is available. Native XGBoost <code>predcontrib</code> is used for XGBoost; other engines are left as NULL unless a future R4VN explainer is added.
pdp	Logical or character vector. TRUE calculates partial-dependence data for up to the five most important original numeric predictors; a character vector requests specific predictors.
validation	Optional external validation data frame. It must contain the same outcome and required predictors. It is never used for preprocessing, selection, tuning, balancing, threshold selection, or final model fitting.
predict	Optional new data frame for predictions after the final model is fitted. Predictions are returned in <code>\$predictions</code> .
id	Optional identifier variable to copy into prediction output.
show	Logical; open the publication-style HTML report in the Viewer. Default TRUE.
plot	Logical or character vector controlling figures embedded in the HTML Viewer. TRUE embeds every principal figure available for the analysis. Character values may include "roc", "pr", "calibration", "threshold", "confusion", "importance", "decision", "learning", "observed", "residual", "pdp", and "shap"; "all" requests every available figure. Any figure sent to the Plot pane through <code>plot_display</code> is also retained in the Viewer whenever plotting is enabled.
plot_display	Figure type(s) also drawn in the interactive R/RStudio Plot pane and Plot history. "auto" (default) draws the most useful diagnostic figures for the task. Use "all" for every available figure, "none" to keep figures only in the Viewer, or a character vector of figure names. This option never changes model fitting or performance estimates.
plot_args	Named list of base-graphics options used for Viewer and Plot pane figures. Options may be common (for example <code>list(font_family="Arial")</code>) or nested, for example <code>list(all=list(font_family="Arial"), roc=list(lwd=3))</code> .
strict	Logical. If FALSE (default), a non-essential figure that cannot be drawn is skipped with a warning while the analysis result is retained. If TRUE, such a plotting error stops the call.
console	Logical; also print a compact console summary. Default FALSE.

<code>digit</code>	Number of digits displayed for estimates. Default 3.
<code>title</code>	Optional report title.
<code>ai</code>	FALSE, TRUE, or a named R4VN AI endpoint. When R4VN <code>aiask()</code> is available, interpretation is requested only after the quantitative analysis object has been created. <code>tabmachine()</code> never calls AI when <code>ai = FALSE</code> ; when enabling AI, the privacy behavior of the configured <code>aiask()</code> endpoint should be reviewed because fitted model objects may contain development information needed for prediction.
<code>...</code>	Reserved for future model-engine options.

Details

1. What `tabmachine()` regards as a complete ML workflow:

A typical call performs the following sequence:

1. resolve active/explicit data and variable labels;
2. validate the outcome and candidate predictors;
3. create an untouched test partition;
4. inside training resamples, learn imputation/transformation/encoding rules;
5. remove structural problems such as constant predictors;
6. optionally create polynomial/interaction features and/or training-fold PCA;
7. apply feature selection inside the training portion of each resample;
8. apply class balancing only inside the training portion of each resample;
9. tune and compare candidate algorithms;
10. choose the final algorithm using development data only;
11. choose a binary classification threshold from training out-of-fold predictions only;
12. refit the selected pipeline on the full development training sample;
13. evaluate the untouched test set and calculate confidence intervals;
14. optionally validate on a completely external data set;
15. assess calibration, decision-curve utility, and variable importance;
16. store a prediction blueprint for future `predict()` calls.

2. Confidence intervals:

`ci = TRUE` is the default because R4VN is intended for scientific reporting. The implementation does not attach a made-up CI to a quantity merely because a point estimate exists. Methods currently used are:

- sensitivity, specificity, PPV, NPV, accuracy, and prevalence: Wilson binomial intervals;
- ROC-AUC: DeLong interval through `pROC` when available; otherwise a stratified nonparametric bootstrap interval;
- an optimized binary classification threshold: stratified bootstrap interval; a user-fixed threshold has no sampling CI because it is specified rather than estimated;
- balanced accuracy, F1, MCC, kappa, PR-AUC, Brier score, log loss and other derived binary metrics: paired-observation nonparametric bootstrap;
- calibration intercept and slope: model-based Wald intervals, with bootstrap fallback when the calibration model is unstable;

- RMSE, MAE, R-squared and MAPE: nonparametric bootstrap over test subjects;
- multiclass accuracy/balanced accuracy/macro-F1/weighted-F1, macro one-vs-rest AUC/PR-AUC, and log loss: nonparametric bootstrap over test subjects;
- class-specific multiclass one-vs-rest sensitivity, specificity, PPV, NPV, accuracy and prevalence: Wilson intervals where the denominator is fixed; class-specific AUC uses DeLong through pROC when available, while PR-AUC, F1 and other derived class measures use nonparametric bootstrap;
- decision-curve net benefit: pointwise nonparametric bootstrap when `ci=TRUE`.

These intervals quantify uncertainty in performance on the evaluation sample conditional on the fitted development procedure. They do not replace full external validation or transportability assessment.

3. Leakage prevention:

The most important implementation rule is that no data-dependent preprocessing action is estimated on the test set. Imputation values, transformations, factor levels, scaling parameters, feature selection, balancing, tuning, and threshold optimization are fitted using training data. During CV, those operations are refitted inside each training fold before predictions are made for the corresponding validation fold.

4. Class imbalance:

`balance = "weight"` is the preferred automatic strategy because it does not fabricate observations. `up`, `down`, native numeric-space `smote`, native `adasyn`, and optional `ROSE` are available for explicit experiments. Synthetic sampling occurs after fold-specific numeric encoding and is therefore applied only to the analysis portion of a resample. The original test prevalence is preserved for evaluation, PPV/NPV, calibration, Brier score and decision curves.

5. Feature selection versus feature importance:

`select` controls which predictors are allowed into the fitted model. `importance` explains which predictors contribute most to a fitted final model. They are intentionally separate concepts. A variable may survive selection yet have weak final permutation importance, and correlated variables may share or exchange importance.

6. Model comparison:

Development CV estimates are useful for choosing an algorithm. The untouched test estimate is the primary internal-validation result. When several models are evaluated on the same test subjects, `$performance` includes a CI for each available metric. `$model_difference` additionally stores paired bootstrap differences in the primary metric between each model and the selected model.

7. External validation:

Supply `validation = external_data` to evaluate the finalized development pipeline without refitting it. External performance and its CIs are stored separately. If external validation is the principal evaluation, use `split = FALSE` to use all development observations for model development.

8. Optional packages and the low-dependency default:

R4VN deliberately does not require every ML engine for every user. The default `method = "auto"` is intentionally based on base/recommended R engines and does not require `caret`, `tidymodels`,

recipes, yardstick, or a collection of boosting/forest packages. Optional modelling engines are checked only when the user explicitly requests them or uses `method = "all"`. `glmnet` supplies penalized models; `ranger` random forests; `xgboost` gradient boosting; `e1071` SVM and naive Bayes; `Boruta` Boruta feature selection; `ROSE` ROSE sampling; and `pROC` DeLong ROC intervals. When `pROC` is absent, `R4VN` uses its native bootstrap AUC interval. The returned `$engines` table records what was requested, installed, and actually used.

Value

Invisibly returns an object of class `c("r4vn_machine", "r4vn_tab")` with major components:

overview Data/task overview.

engines Supported algorithms, package requirements, availability, and engines actually used.

preprocessing Auditable preprocessing decisions.

balance Chosen imbalance strategy and comparison when requested.

selection Chosen feature-selection strategy and selected predictors.

tuning Selected hyperparameters for each candidate algorithm.

cv_performance Development cross-validation performance.

performance Held-out test performance in long format with CI columns.

comparison Publication-ready model comparison table.

model_difference Paired difference in the primary metric versus the selected model, with bootstrap CI where available.

overfitting Development-versus-evaluation comparison for the primary metric.

coefficients For an unpenalized logistic or linear final model, model coefficients with 95% CI; logistic coefficients are exponentiated to OR.

best Name of the selected algorithm.

final Final fitted pipeline/model object.

threshold Training-derived classification threshold information.

confusion Final binary or multiclass confusion matrix counts.

class_performance For multiclass outcomes, one-vs-rest class-specific discrimination and classification measures with 95% CI.

calibration Calibration statistics and curve data.

decision Decision-curve data.

importance Grouped permutation importance.

explanation Convenience list collecting importance, SHAP and PDP outputs when `explain = TRUE`.

shap Native XGBoost contribution matrix when requested/supported.

pdp Partial-dependence data when requested.

external_performance External-validation performance when supplied.

external_confusion External binary or multiclass confusion matrix when applicable.

external_class_performance External multiclass one-vs-rest performance with CI.

external_calibration External binary calibration statistics/curve when requested.

external_decision External binary decision-curve data when requested.

predictions Predictions for predict= data when supplied.

plots Data required to replay publication plots.

plot_titles Stable publication titles for available figure types.

tables Named publication-ready tables used by R4VN/Studio/export.

html Finished HTML report.

settings Complete analysis settings.

notes Methodological notes and any optional-engine skips.

Examples - binary classification

```
set.seed(123)
n <- 500
d <- data.frame(
  patient_id = seq_len(n),
  age = rnorm(n, 45, 12),
  sex = factor(sample(c("Female", "Male"), n, TRUE)),
  bmi = rnorm(n, 23, 3.5),
  smoke = factor(sample(c("No", "Yes"), n, TRUE, c(.75, .25)))
)
lp <- -6 + .055*d$age + .10*d$bmi + .65*(d$smoke == "Yes")
d$hypertension <- factor(rbinom(n, 1, plogis(lp)), 0:1, c("No", "Yes"))

m <- tabmachine(
  hypertension,
  x = vars(age, sex, bmi, smoke),
  data = d,
  event = "Yes",
  method = "logistic",
  tune = FALSE,
  boot = 200,
  show = FALSE,
  plot = FALSE
)
m$comparison
m$performance
```

Examples - automatic ML comparison

```
\donttest{
# The automatic comparison uses the low-dependency R4VN core set.
# Optional engines join only when explicitly requested or method = "all".
m <- tabmachine(
  hypertension,
  x = vars(age, sex, bmi, smoke),
```

```

    data = d,
    event = "Yes",
    method = "auto",
    select = "auto",
    balance = "auto",
    ci = TRUE,
    boot = 2000
  )
  plot(m, "roc")
  plot(m, "calibration")
  plot(m, "importance")
}

```

Examples - class imbalance and feature selection

```

\donttest{
m2 <- tabmachine(
  hypertension,
  x = .,
  data = d,
  exclude = vars(patient_id),
  event = "Yes",
  balance = "compare",
  select = "compare",
  metric = "pr_auc",
  nested = TRUE
)
m2$balance
m2$selection
}

```

Examples - continuous outcome

```

set.seed(321)
r <- data.frame(
  age = rnorm(400, 50, 14),
  bmi = rnorm(400, 24, 4),
  sex = factor(sample(c("Female", "Male"), 400, TRUE))
)
r$sbp <- 75 + .65*r$age + 1.15*r$bmi + 4*(r$sex == "Male") + rnorm(400, 0, 9)

mr <- tabmachine(
  sbp,
  x = vars(age, bmi, sex),
  data = r,
  method = "linear",
  tune = FALSE,
  boot = 200,
  show = FALSE,

```

```

    plot = FALSE
  )
  mr$performance

```

Examples - active data and new predictions

```

\donttest{
  usedf(d)
  fit <- tabmachine(
    hypertension,
    x = vars(age, sex, bmi, smoke),
    event = "Yes",
    method = "logistic"
  )

  newpatients <- data.frame(
    age = c(35, 68),
    sex = factor(c("Female", "Male"), levels = levels(d$sex)),
    bmi = c(21, 31),
    smoke = factor(c("No", "Yes"), levels = levels(d$smoke))
  )
  predict(fit, newpatients)
}

```

Examples - external validation

```

\donttest{
  dev <- d[1:350, ]
  ext <- d[351:500, ]
  me <- tabmachine(
    hypertension,
    x = vars(age, sex, bmi, smoke),
    data = dev,
    validation = ext,
    split = FALSE,
    event = "Yes",
    method = "auto"
  )
  me$external_performance
}

```

Examples - Viewer and Plot pane together

```

\donttest{
  # Every requested figure remains in the Viewer. The selected figures are
  # also added to Plot history so Previous/Next can be used in RStudio.
  mv <- tabmachine(
    hypertension, vars(age, sex, bmi, smoke), data = d, event = "Yes",
    method = "logistic", tune = FALSE, boot = 200,

```

```

    plot = TRUE,
    plot_display = c("roc", "calibration", "confusion", "importance")
  )
  names(mv$plots)
  mv$settings$viewer_plots
  mv$settings$display_plots

# Keep all figures in Viewer but draw none in the Plot pane.
mv2 <- tabmachine(
  hypertension, vars(age, sex, bmi, smoke), data = d, event = "Yes",
  method = "logistic", tune = FALSE, boot = 200,
  plot = TRUE, plot_display = "none"
)
}
```

Examples - classification thresholds

```

\donttest{
myouden <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, threshold="youden",
  boot=200, show=FALSE, plot=FALSE)
mf1 <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, threshold="f1",
  boot=200, show=FALSE, plot=FALSE)
msens <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, threshold="sens",
  target_sens=.90, boot=200, show=FALSE, plot=FALSE)
mspec <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, threshold="spec",
  target_spec=.90, boot=200, show=FALSE, plot=FALSE)
mfixed <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, threshold=.20,
  boot=200, show=FALSE, plot=FALSE)
myouden$threshold
plot(myouden, "threshold")
}
```

Examples - missing data, correlation, outliers, and transformations

```

\donttest{
mp <- tabmachine(
  hypertension, vars(age, sex, bmi, smoke), data=d, event="Yes",
  method="logistic", tune=FALSE,
  missing="auto", corr=.90, outlier="winsor", transform="none",
  boot=200, show=FALSE, plot=FALSE
)
mp$preprocessing
}
```

Examples - imbalance without extra packages

```
\donttest{
# weight, up, down, SMOTE, and ADASYN are implemented inside R4VN.
mw <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, balance="weight",
  metric="pr_auc", boot=200, show=FALSE, plot=FALSE)
msmote <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, balance="smote",
  metric="pr_auc", boot=200, show=FALSE, plot=FALSE)
mw$balance
msmote$balance
}
```

Examples - feature engineering and PCA

```
\donttest{
mfeat <- tabmachine(hypertension, vars(age, bmi), data=d, event="Yes",
  method="logistic", tune=FALSE, feature="all", degree=2,
  boot=200, show=FALSE, plot=FALSE)
mfeat$selection

mpdp <- tabmachine(hypertension, vars(age, bmi, smoke), data=d, event="Yes",
  method="logistic", tune=FALSE, pdp=c("age", "bmi"), boot=200,
  plot=TRUE, plot_display="pdp")
mpdp$pdp
plot(mpdp, "pdp")

set.seed(11)
hd <- as.data.frame(matrix(rnorm(180*60), 180, 60))
names(hd) <- paste0("x", 1:60)
hd$y <- factor(rbinom(180, 1, plogis(hd$x1-.7*hd$x2+.5*hd$x3)), 0:1, c("No", "Yes"))
mpca <- tabmachine(y, x=., data=hd, event="Yes", method="logistic",
  tune=FALSE, reduce="pca", variance=.90, select="filter",
  simplify=FALSE, boot=100, show=FALSE, plot=FALSE)
mpca$selection
}
```

Examples - feature selection and parsimony

```
\donttest{
mfilter <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, select="filter",
  simplify=TRUE, boot=200, show=FALSE, plot=FALSE)
mfilter$selection
mfilter$simplify

# Penalized selection is optional and used only when glmnet is installed.
if (requireNamespace("glmnet", quietly=TRUE)) {
```

```

    mlasso <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
      event="Yes", method="lasso", select="lasso", boot=200,
      show=FALSE, plot=FALSE)
    mlasso$selection
  }
}

```

Examples - repeated and nested validation

```

\donttest{
mrep <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method=c("logistic","tree"), tune=TRUE,
  folds=5, repeats=3, boot=200, show=FALSE, plot=FALSE)
mrep$cv_performance

mnested <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method=c("logistic","tree"), tune=TRUE,
  folds=5, nested=TRUE, boot=200, show=FALSE, plot=FALSE)
mnested$cv_performance
}

```

Examples - three-way split

```

\donttest{
msplit <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", split=c(.70,.15,.15), method="auto", boot=200,
  show=FALSE, plot=FALSE)
msplit$overview
}

```

Examples - calibration, clinical utility, and learning curve

```

\donttest{
mclin <- tabmachine(hypertension, vars(age, sex, bmi, smoke), data=d,
  event="Yes", method="logistic", tune=FALSE,
  calibration=TRUE, decision=TRUE, learning=TRUE, boot=200,
  plot=TRUE, plot_display="all")
mclin$calibration$statistics
head(mclin$decision)
mclin$learning
}

```

Examples - regression diagnostics

```

\donttest{
mr2 <- tabmachine(sbp, vars(age,bmi,sex), data=r, method="auto",
  boot=200, plot=TRUE, plot_display=c("observed","residual","importance"))
mr2$comparison
plot(mr2,"observed")
}

```

```
plot(mr2,"residual")
}
```

Examples - multiclass classification

```
\donttest{
ir <- iris
mm <- tabmachine(Species, vars(Sepal.Length,Sepal.Width,Petal.Length,Petal.Width),
  data=ir, method="auto", folds=5, boot=200,
  plot=TRUE, plot_display=c("confusion","importance"))
mm$performance
mm$class_performance
mm$confusion
plot(mm,"confusion")
}
```

Examples - predictions for new patients

```
\donttest{
newpatients <- data.frame(
  patient_id=c("P001","P002"), age=c(35,68),
  sex=factor(c("Female","Male"),levels=levels(d$sex)),
  bmi=c(21,31), smoke=factor(c("No","Yes"),levels=levels(d$smoke))
)
mpred <- tabmachine(hypertension, vars(age,sex,bmi,smoke), data=d,
  event="Yes", method="logistic", tune=FALSE, boot=200,
  predict=newpatients, id=patient_id, show=FALSE, plot=FALSE)
mpred$predictions
predict(mpred,newpatients,type="prob")
predict(mpred,newpatients,type="class")
}
```

Examples - optional advanced engines

```
\donttest{
# None of these packages is needed for the ordinary R4VN auto workflow.
if (requireNamespace("ranger",quietly=TRUE)) {
  mrf <- tabmachine(hypertension, vars(age,sex,bmi,smoke), data=d,
    event="Yes", method="rf", boot=200, show=FALSE, plot=FALSE)
}
if (requireNamespace("xgboost",quietly=TRUE)) {
  mxgb <- tabmachine(hypertension, vars(age,sex,bmi,smoke), data=d,
    event="Yes", method="xgb", shap=TRUE, boot=200, show=FALSE, plot=FALSE)
  mxgb$shap
  plot(mxgb, "shap")
}
if (requireNamespace("e1071",quietly=TRUE)) {
  msvm <- tabmachine(hypertension, vars(age,sex,bmi,smoke), data=d,
    event="Yes", method="svm", boot=200, show=FALSE, plot=FALSE)
}
```

```

}
if (requireNamespace("Boruta",quietly=TRUE)) {
  mb <- tabmachine(hypertension, vars(age,sex,bmi,smoke), data=d,
    event="Yes", method="logistic", select="boruta", tune=FALSE,
    boot=200, show=FALSE, plot=FALSE)
}
if (requireNamespace("ROSE",quietly=TRUE)) {
  mrose <- tabmachine(hypertension, vars(age,sex,bmi,smoke), data=d,
    event="Yes", method="logistic", balance="rose", tune=FALSE,
    boot=200, show=FALSE, plot=FALSE)
}
}

```

Examples

```

set.seed(99)
n <- 110
dd <- data.frame(
  x1 = rnorm(n),
  x2 = rnorm(n),
  group = factor(sample(c("A", "B"), n, TRUE))
)
pp <- plogis(-0.4 + 0.9 * dd$x1 - 0.5 * dd$x2)
dd$y <- factor(rbinom(n, 1, pp), 0:1, c("No", "Yes"))

z <- tabmachine(
  y,
  x = vars(x1, x2, group),
  data = dd,
  event = "Yes",
  method = "logistic",
  tune = FALSE,
  folds = 2,
  ci = FALSE,
  boot = 50,
  calibration = FALSE,
  decision = FALSE,
  importance = FALSE,
  explain = FALSE,
  show = FALSE,
  plot = FALSE
)
z$comparison

```

Description

Fits fixed- and/or random-effects meta-analysis and returns a complete, publication-ready report. `tabmeta()` accepts a binary 2-by-2 table (a, b, c, d), event/total data, continuous summaries, rates, correlations, or study-level effect estimates with standard errors or confidence intervals. The default `profile="auto"` adds prediction, few-study inference, subgroup/moderator results when requested, small-study-effect diagnostics, sensitivity analyses, and figures.

Usage

```
tabmeta(
  data = NULL,
  study,
  effect = NULL,
  se = NULL,
  lower = NULL,
  upper = NULL,
  a = NULL,
  b = NULL,
  c = NULL,
  d = NULL,
  event1 = NULL,
  n1 = NULL,
  event0 = NULL,
  n0 = NULL,
  mean1 = NULL,
  sd1 = NULL,
  mean0 = NULL,
  sd0 = NULL,
  event = NULL,
  n = NULL,
  time = NULL,
  time1 = NULL,
  time0 = NULL,
  or = FALSE,
  rr = FALSE,
  rd = FALSE,
  hr = FALSE,
  irr = FALSE,
  md = FALSE,
  smd = FALSE,
  prop = FALSE,
  rate = FALSE,
  cor = FALSE,
  fixed = FALSE,
  random = TRUE,
  method = "REML",
  hk = NULL,
  small = c("auto", "adhoc", "knha", "t", "z"),
```

```

prediction = NULL,
by = NULL,
subgroup = NULL,
reg = NULL,
moderator = NULL,
bias = NULL,
bias_methods = "auto",
leaveout = NULL,
influence = NULL,
cumulative = NULL,
transform = NULL,
cc = 0.5,
zero = c("keep", "exclude"),
ci = 0.95,
digit = 2,
p_digit = 3,
profile = c("auto", "brief", "full", "custom"),
full = NULL,
plot = NULL,
plot_display = "forest",
plot_args = list(),
report = FALSE,
interpretation = FALSE,
title = NULL,
show = TRUE,
export = NULL,
file = NULL,
open = FALSE,
strict = FALSE
)

```

Arguments

<code>data</code>	Optional data frame. When omitted, active R4VN data are used.
<code>study</code>	Study label variable.
<code>effect</code>	Generic study-level effect estimate on the natural scale.
<code>se</code>	Standard error on the analysis scale. For ratio measures this is the standard error of the log effect.
<code>lower, upper</code>	Lower and upper confidence limits for effect.
<code>a, b, c, d</code>	Binary 2-by-2 cells: events and non-events in group 1 (a, b) and group 0 (c, d). Supplying these four arguments is equivalent to <code>event1=a</code> , <code>n1=a+b</code> , <code>event0=c</code> , <code>n0=c+d</code> .
<code>event1, n1, event0, n0</code>	Events and total sample sizes in groups 1 and 0.
<code>mean1, sd1, mean0, sd0</code>	Group means and standard deviations.
<code>event, n, time</code>	Single-group events, sample size, and person-time.

time1, time0	Person-time in groups 1 and 0 for incidence-rate ratios.
or, rr, rd, hr, irr, md, smd, prop, rate, cor	Logical effect selectors. With raw binary input, OR is inferred if none is selected; with continuous summaries, MD is inferred. Set a selector to request another measure.
fixed	Fit a fixed-effect model in addition to, or instead of, the random-effects model.
random	Fit a random-effects model; default TRUE.
method	Random-effects tau-squared estimator; default "REML".
hk	Backward-compatible logical shortcut. TRUE is small="knha"; FALSE is small="z". Prefer small in new code.
small	Random-effects inference: "auto", "adhoc", "knha", "t", or "z". "auto" uses modified Hartung-Knapp ("adhoc") when there are at most 10 studies and the normal approximation otherwise.
prediction	Add a prediction interval. NULL uses the profile default.
by, subgroup	One or more subgroup variables. Use one unquoted variable, a character vector, or vars(region, design). subgroup is a readable alias for by; both may be combined. Factor levels and value labels from labelled data are used in tables, interpretation, and subgroup figures. For every estimable subgroup, the result includes its pooled effect and confidence interval, effect test/df/p-value, prediction interval, Cochran's Q/df/p-value, I-squared and tau-squared with confidence intervals when estimable. The publication table is transposed: statistics are rows and subgroup labels are columns. Multiple subgroup variables produce one transposed table per variable.
reg, moderator	One or more meta-regression moderators created with vars() or supplied as character names. moderator is an alias for reg. The output includes the multi-variable model and a univariable moderator screen.
bias	Assess small-study effects/publication bias. NULL uses the profile default.
bias_methods	One or more of "egger", "begg", "trimfill", "failsafe", or "selection". "auto" runs Egger, Begg, and trim-and-fill; the "full" profile also requests fail-safe N and a selection model. Methods are used as sensitivity diagnostics, not as proof that publication bias is or is not present.
leaveout	Perform leave-one-out sensitivity analysis. NULL uses the profile default.
influence	Perform influence diagnostics. NULL uses the profile default.
cumulative	Optional variable defining the ordering for cumulative meta-analysis, usually publication year.
transform	Single-proportion transformation: "logit", "arcsine", "ft", or "none".
cc	Continuity correction for zero events; default 0.5.
zero	Handling of double-zero binary studies: "keep" or "exclude".
ci	Confidence level as a proportion.
digit	Number of decimals for effect estimates and confidence limits, including subgroup and meta-regression estimates. Default 2.
p_digit	Number of decimals for p-values.

profile	Analysis profile. "auto" (default) creates a complete appropriate analysis; "brief" creates the main model and forest plot; "full" also requests extended bias diagnostics and radial/L'Abbe plots; "custom" turns optional modules off unless explicitly requested.
full	Backward-compatible profile shortcut: TRUE selects "full" and FALSE selects "custom".
plot	Create publication-ready plots. NULL uses the profile default. With subgroup analysis, the HTML Viewer includes the overall forest plot and one clearly titled forest plot for every estimable subgroup level.
plot_display	Plot type(s) also drawn in the interactive R/RStudio Plot pane when plot=TRUE. Default "forest". Use a character vector for Plot history, "all" for every available figure, or "none" to suppress Plot pane drawing while retaining figures in the HTML Viewer. When subgroup analysis is present, selecting "forest" adds the overall forest plot and every subgroup forest plot to Plot history; use Previous/Next to review.
plot_args	Named list of plot options. Supply common options directly, or nested lists such as <code>list(all=list(color="navy"), forest=list(xlim=c(0.2, 2)))</code> .
report	Add manuscript-style results text.
interpretation	Add a detailed, sectioned interpretation covering the pooled effect, heterogeneity, prediction interval, subgroup differences, moderators, small-study effects, few-study inference, influence, leave-one-out robustness, and cumulative evidence whenever available. Default is FALSE.
title	Report title.
show	Open the complete HTML report; default TRUE.
export	Optional direct export format(s): "html", "docx", "xlsx", "pdf", or "png".
file	Export file or base path. The extension may determine the format.
open	Open the exported file.
strict	Stop when an optional diagnostic or figure cannot be created. The default FALSE retains the main analysis and issues a warning/note.

Value

Invisibly returns an object of class `r4vn_meta` and `r4vn_tab`. Stable publication components are available in `$estimates`, `$tests`, `$diagnostics`, `$models`, `$tables`, and `$metadata`. The table `$tables$Statistical_tests` gives the pooled-effect and Cochran's Q tests. Its significance and conclusion columns are added only when `interpretation=TRUE`. `$tables$Subgroup` is the transposed publication table for the first subgroup variable, `$subgroup_tables` contains every transposed subgroup table, and `$subgroup_long` retains tidy long output.

See Also

`tabexport`, `vars`

Other R4VN tables: `tab()`, `tabexport()`, `tabforest()`, `tablong()`, `tabmulti()`, `tabscale()`, `tabscore()`, `tabsurvey()`, `vars()`

Examples

```

if (requireNamespace("metafor", quietly = TRUE)) {
  dat <- read.csv(
    system.file("extdata", "meta_example.csv", package = "R4VN")
  )

  # 1. Binary outcome from a, b, c, d. OR is inferred automatically.
  dat$non_event_treat <- dat$n_treat - dat$event_treat
  dat$non_event_control <- dat$n_control - dat$event_control
  m_abcd <- tabmeta(
    data = dat, study = study,
    a = event_treat, b = non_event_treat,
    c = event_control, d = non_event_control,
    profile = "custom", plot = FALSE, show = FALSE
  )
  m_abcd$estimates$overall
  m_abcd$tables$Binary_2x2
  m_abcd$tables$Statistical_tests
  m_abcd$tests$overall
  m_abcd$tests$heterogeneity
  plot(
    m_abcd, type = "forest", show_abcd = TRUE,
    abcd_titles = c("Events T", "No event T", "Events C", "No event C")
  )

  # 2. Equivalent event/total syntax; request RR or RD with rr/rd=TRUE.
  m_or <- tabmeta(
    data = dat, study = study,
    event1 = event_treat, n1 = n_treat,
    event0 = event_control, n0 = n_control,
    or = TRUE, profile = "custom", plot = FALSE, show = FALSE
  )

  # 3. Complete automatic analysis: report, prediction, diagnostics, plots.
  m_all <- tabmeta(
    data = dat, study = study,
    a = event_treat, b = non_event_treat,
    c = event_control, d = non_event_control,
    show = FALSE
  )

  # 4. Subgroup analysis: one or several subgroup variables.
  m_sub <- tabmeta(
    data = dat, study = study,
    effect = OR, lower = LCI, upper = UCI, or = TRUE,
    subgroup = region,
    profile = "custom", plot = FALSE, show = FALSE
  )
  m_sub$tables$Subgroup
  m_sub$subgroup_long

```

```

m_sub$tables$Subgroup_test
# Each subgroup row includes its effect test and heterogeneity test.
names(m_sub$tables$Subgroup)
# If the subgroup variable carries value labels, R4VN prints those labels
# instead of numeric codes. With plot=TRUE, both Viewer and Plot history
# contain the overall plot and one clearly titled plot per subgroup level.
dat$risk_group <- rep(c(1, 2), length.out = nrow(dat))
attr(dat$risk_group, "label") <- "Baseline risk"
attr(dat$risk_group, "labels") <- c("Lower risk" = 1, "Higher risk" = 2)

m_labelled <- tabmeta(
  data = dat, study = study,
  effect = OR, lower = LCI, upper = UCI, or = TRUE,
  subgroup = risk_group, profile = "custom",
  plot = TRUE, plot_display = "forest", show = FALSE
)
m_labelled$plot_titles

# Multiple subgroup analyses can be requested together.
dat$period <- ifelse(dat$year < median(dat$year), "Earlier", "Later")
m_sub2 <- tabmeta(
  data = dat, study = study,
  effect = OR, lower = LCI, upper = UCI, or = TRUE,
  subgroup = vars(region, period),
  profile = "custom", plot = FALSE, show = FALSE
)

# 5. Meta-regression with numeric and categorical moderators.
attr(dat$year, "label") <- "Publication year"
attr(dat$region, "label") <- "Geographic region"
m_reg <- tabmeta(
  data = dat, study = study,
  effect = OR, lower = LCI, upper = UCI, or = TRUE,
  moderator = vars(c.year, region),
  profile = "custom", plot = FALSE, show = FALSE
)
m_reg$tables$Meta_regression
m_reg$tables$Moderator_univariable
# Intercept is written in full; moderator labels are used when available.
# Set digit=4, for example, when four decimal places are required.

# 6. Publication-bias and small-study-effect sensitivity analyses.
m_bias <- tabmeta(
  data = dat, study = study,
  effect = OR, lower = LCI, upper = UCI, or = TRUE,
  bias = TRUE,
  bias_methods = c("egger", "begg", "trimfill", "failsafe"),
  profile = "custom", plot = FALSE, show = FALSE
)
m_bias$tables$Publication_bias
m_bias$tables$Publication_bias_adjusted

```

```

# 7. Few-study inference. auto uses modified Hartung-Knapp at <=10 studies.
m_few <- tabmeta(
  data = dat[1:8, ], study = study,
  effect = OR, lower = LCI, upper = UCI, or = TRUE,
  small = "auto", profile = "custom", plot = FALSE, show = FALSE
)
m_few$tables$Small_sample_inference

# 8. Generic hazard ratios with confidence intervals.
m_hr <- tabmeta(
  data = dat, study = study,
  effect = OR, lower = LCI, upper = UCI, hr = TRUE,
  profile = "custom", plot = FALSE, show = FALSE
)

# 9. Continuous MD/SMD, single proportion/rate, correlation, and IRR.
cont <- data.frame(
  study = paste0("C", 1:5),
  m1 = c(12, 14, 13, 16, 15), s1 = c(3, 4, 3, 5, 4), n1 = rep(60, 5),
  m0 = c(15, 15, 16, 18, 16), s0 = c(4, 4, 5, 5, 4), n0 = rep(60, 5)
)
m_md <- tabmeta(
  cont, study, mean1 = m1, sd1 = s1, n1 = n1,
  mean0 = m0, sd0 = s0, n0 = n0, md = TRUE,
  profile = "custom", plot = FALSE, show = FALSE
)
m_smd <- tabmeta(
  cont, study, mean1 = m1, sd1 = s1, n1 = n1,
  mean0 = m0, sd0 = s0, n0 = n0, smd = TRUE,
  profile = "custom", plot = FALSE, show = FALSE
)

one <- data.frame(
  study = paste0("P", 1:5), events = c(8, 12, 15, 10, 14),
  total = c(100, 110, 120, 90, 105), person_time = c(80, 90, 95, 75, 88),
  correlation = c(.20, .28, .15, .31, .24)
)
m_prop <- tabmeta(
  one, study, event = events, n = total, prop = TRUE,
  profile = "custom", plot = FALSE, show = FALSE
)
m_rate <- tabmeta(
  one, study, event = events, time = person_time, rate = TRUE,
  profile = "custom", plot = FALSE, show = FALSE
)
m_cor <- tabmeta(
  one, study, effect = correlation, n = total, cor = TRUE,
  profile = "custom", plot = FALSE, show = FALSE
)
m_irr <- tabmeta(
  dat[1:5, ], study,
  event1 = event_treat, time1 = n_treat,
  event0 = event_control, time0 = n_control, irr = TRUE,

```

```

    profile = "custom", plot = FALSE, show = FALSE
  )

# 10. Cumulative meta-analysis ordered by publication year.
m_cum <- tabmeta(
  data = dat, study = study,
  effect = OR, lower = LCI, upper = UCI, or = TRUE,
  cumulative = year,
  profile = "custom", plot = FALSE, show = FALSE
)
m_cum$tables$Cumulative

# 11. Draw or save individual publication figures.
if (interactive()) {
  # plot=TRUE keeps all figures in the HTML Viewer and also draws the
  # selected figures in the R/RStudio Plot pane and Plot history.
  m_publication <- tabmeta(
    data = dat, study = study,
    a = event_treat, b = non_event_treat,
    c = event_control, d = non_event_control,
    interpretation = TRUE,
    plot = TRUE,
    plot_display = c("forest", "funnel", "trimfill"),
    plot_args = list(
      all = list(
        font_family = "Arial", background = "white",
        title_color = "#17365D", title_size = 1.1,
        text_size = 0.86, axis_size = 0.92
      ),
      forest = list(
        subtitle = "Random-effects model with 95% confidence intervals",
        caption = "Square size reflects study weight; diamond is pooled effect.",
        margins = c(5.5, 4.2, 5.0, 2.0),
        point_color = "#1F4E79", ci_color = "#5B9BD5",
        summary_color = "#C00000", summary_border = "#7F0000",
        point_shape = 15, row_shade = "zebra",
        shade_color = "#F5F7FA", show_weights = TRUE,
        weight_title = "Weight", estimate_title = "OR (95% CI)",
        show_prediction = FALSE,
        ref_color = "#666666", ref_type = 2,
        xlim = c(0.2, 2.0), ticks = c(0.25, 0.5, 1, 1.5, 2)
      ),
      funnel = list(
        point_shape = 21, point_color = "#1F4E79",
        point_bg = "#D9EAF7", point_size = 1.1,
        contour_levels = c(90, 95, 99),
        contour_colors = c("#FFF2CC", "#FCE4D6", "#E2F0D9"),
        funnel_label = "out", funnel_legend = "topright"
      )
    )
  )
  m_publication$tables$Interpretation

```

```

# Any figure can be redrawn or saved independently.
plot(m_publication, type = "forest")
plot(m_publication, type = "funnel", contour = TRUE)
plot(m_publication, type = "trimfill", contour = TRUE)
plot(m_reg, type = "bubble", moderator = "year")
plot(
  m_publication, type = "forest", file = "forest_publication.tiff",
  width = 2400, height = 1800, res = 300,
  font_family = "Arial", point_color = "#1F4E79",
  ci_color = "#5B9BD5", summary_color = "#C00000",
  show_weights = TRUE, show_prediction = FALSE,
  xlim = c(0.5, 1.5), ticks = c(0.5, 0.75, 1, 1.25, 1.5)
)
# Values outside xlim remain exact in the Estimate (95% CI) column;
# the graphical confidence interval is clipped with an arrow.
m_publication$overall[c("pi_lower", "pi_upper")]
tabmeta(
  data = dat, study = study,
  a = event_treat, b = non_event_treat,
  c = event_control, d = non_event_control,
  export = c("docx", "xlsx"), file = "meta_report",
  show = FALSE
)
}
}

```

tabmulti

Compare Multivariable Model-Building Strategies

Description

Builds and compares variable-selection strategies for a binary outcome. Each strategy selects complete variables or terms, after which the selected model is refitted using ordinary logistic regression or modified Poisson regression so that conventional OR, RR, or PR estimates, 95% confidence intervals, and p-values can be reported.

Usage

```

tabmulti(data = NULL, vars = NULL, by = NULL,
  methods = c("full", "forward", "backward", "purposeful"),
  digit = 1, p_digit = 3, effect_digit = 2, global = FALSE,
  pvalue = TRUE, rvrow = NULL, bold_p = TRUE, p_bold = 0.05,
  or = FALSE, rr = FALSE, pr = FALSE, event = NULL,
  criterion = c("AIC", "BIC"), force = NULL, entry = 0.20,
  stay = 0.05, confounding = 0.10, lasso_lambda = "lambda.1se",
  bma_pip = 0.50, max_subset_vars = 15L, max_subset_models = 100000L,
  template = c("journal", "clean", "minimal"), append = NULL,
  file = NULL, raw = FALSE, name = FALSE, title = NULL, show = TRUE)

```

Arguments

<code>data</code>	Optional data frame. When omitted or <code>NULL</code> , the active data frame set by <code>usedf()</code> or <code>opendata(..., active = TRUE)</code> is used.
<code>vars</code>	A variable specification created by <code>vars()</code> .
<code>by</code>	Binary outcome supplied without quotation marks.
<code>methods</code>	Model strategies: "full", "forward", "backward", "purposeful", "lasso", "bma", or "best".
<code>digit</code>	Retained for API consistency with <code>tab()</code> .
<code>p_digit</code>	Number of decimal places for p-values.
<code>effect_digit</code>	Number of decimal places for effect estimates and confidence limits.
<code>global</code>	Logical. Display global likelihood-ratio p-values.
<code>pvalue</code>	Logical. Display coefficient p-value columns.
<code>rvrow</code>	Categorical variables whose displayed level order should be reversed. This does not change model reference categories.
<code>bold_p</code>	Logical. Bold p-values smaller than <code>p_bold</code> .
<code>p_bold</code>	Threshold used when <code>bold_p = TRUE</code> .
<code>or</code>	Logical. Report odds ratios from logistic regression.
<code>rr</code>	Logical. Report risk ratios from modified Poisson regression.
<code>pr</code>	Logical. Report prevalence ratios from modified Poisson regression. Exactly one of <code>or</code> , <code>rr</code> , and <code>pr</code> must be <code>TRUE</code> .
<code>event</code>	Event level. The last observed outcome level is used when omitted.
<code>criterion</code>	Selection criterion, "AIC" or "BIC".
<code>force</code>	Variables forced into every selected model. Accepts <code>vars(...)</code> , <code>c(...)</code> , a character vector, <code>TRUE</code> , or "ALL".
<code>entry</code>	Univariate entry threshold for purposeful selection.
<code>stay</code>	Multivariable retention threshold for purposeful selection.
<code>confounding</code>	Relative coefficient-change threshold for identifying a confounder during purposeful selection.
<code>lasso_lambda</code>	Either "lambda.min" or "lambda.1se".
<code>bma_pip</code>	Posterior inclusion-probability threshold used by the BIC-weighted BMA strategy.
<code>max_subset_vars</code>	Maximum number of candidate variables for exhaustive subset methods.
<code>max_subset_models</code>	Maximum number of subset models to evaluate.
<code>template</code>	HTML style: "journal", "clean", or "minimal".
<code>append</code>	Optional previous <code>r4vn_tabmulti</code> object or existing HTML path.
<code>file</code>	Optional output HTML path.
<code>raw</code>	Logical. Retain unformatted coefficients and selection details.
<code>name</code>	Logical. Display original variable names beside labels.
<code>title</code>	Optional table title.
<code>show</code>	Logical. Open the HTML table in the Viewer or browser.

Details

Available strategies are:

- full: include every candidate variable;
- forward: forward stepwise selection using AIC or BIC;
- backward: backward stepwise selection from the full model;
- purposeful: univariate screening followed by significance and confounding assessment;
- lasso: selection with `glmnet::cv.glmnet()`, followed by ordinary-model refitting; requires the suggested package **glmnet**;
- bma: BIC-weighted subset averaging and inclusion-probability thresholding;
- best: select the subset with the smallest AIC or BIC.

All strategies use the same complete-case sample. Diagnostic rows include sample size, events, number of variables and parameters, AIC, BIC, pseudo-R-squared measures, goodness-of-fit tests, AUC where applicable, and the coefficient-level VIF range. Exhaustive methods grow exponentially with the number of candidate variables.

Value

Invisibly returns an object of class `r4vn_tabmulti`. Important components include `data`, `selected`, `models`, `diagnostics`, `file`, `html`, and `table_html`.

See Also

[vars](#), [tab](#), and [tabexport](#).

Other R4VN tables: [tab\(\)](#), [tabexport\(\)](#), [tabforest\(\)](#), [tablong\(\)](#), [tabmeta\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [tabsurvey\(\)](#), [vars\(\)](#)

Examples

```
set.seed(2026)
n <- 180
dat <- data.frame(
  age = round(rnorm(n, 45, 12)),
  sex = factor(sample(c("Female", "Male"), n, TRUE)),
  bmi = round(rnorm(n, 23, 3), 1),
  smoking = factor(sample(c("No", "Yes"), n, TRUE,
    prob = c(0.70, 0.30))),
  education = factor(sample(c("Primary", "Secondary", "College"),
    n, TRUE))
)
lp <- -3.1 + 0.045 * dat$age + 0.11 * (dat$bmi - 23) +
  0.45 * (dat$sex == "Male") + 0.70 * (dat$smoking == "Yes")
dat$hypertension <- factor(
  rbinom(n, 1, plogis(lp)),
  levels = c(0, 1), labels = c("No", "Yes")
)

models <- tabmulti(
```

```

    dat,
    vars = vars(c.age, b2.sex, c.bmi, b2.smoking, b2.education),
    by = hypertension,
    methods = c("full", "backward"),
    or = TRUE,
    event = "Yes",
    criterion = "AIC",
    global = TRUE,
    show = FALSE
  )
models$selected
models$diagnostics

# LASSO requires the suggested package glmnet.
if (requireNamespace("glmnet", quietly = TRUE)) {
models_lasso <- tabmulti(
  dat,
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking, b2.education),
  by = hypertension,
  methods = c("full", "lasso"),
  or = TRUE,
  event = "Yes",
  show = FALSE
)
}

# Extended usage examples

set.seed(2026)
n <- 400
d <- data.frame(
  sex = factor(sample(c("Female", "Male"), n, TRUE)),
  age = rnorm(n, 45, 12),
  bmi = rnorm(n, 23, 3),
  smoking = factor(sample(c("No", "Yes"), n, TRUE)),
  education = factor(sample(c("Primary", "Secondary", "College"), n, TRUE))
)
lp <- -3.2 + 0.045 * d$age + 0.10 * (d$bmi - 23) +
  0.45 * (d$sex == "Male") + 0.70 * (d$smoking == "Yes")
d$outcome <- factor(rbinom(n, 1, plogis(lp)),
  levels = 0:1, labels = c("No", "Yes"))

# Full multivariable logistic model
m1 <- tabmulti(
  d,
  vars = vars(b2.sex, c.age, c.bmi, b2.smoking, b2.education),
  by = outcome,
  methods = "full",
  or = TRUE,
  event = "Yes",
  show = FALSE
)

```

```

)

# Compare several model-building strategies in one table
m2 <- tabmulti(
  d,
  vars = vars(b2.sex, c.age, c.bmi, b2.smoking, b2.education),
  by = outcome,
  methods = c("full", "forward", "backward", "purposeful"),
  or = TRUE,
  event = "Yes",
  criterion = "AIC",
  global = TRUE,
  show = FALSE
)
m2$selected
m2$diagnostics

# Force variables into every selected model and tune purposeful selection
tabmulti(
  d,
  vars = vars(b2.sex, c.age, c.bmi, b2.smoking, b2.education),
  by = outcome,
  methods = "purposeful",
  force = vars(age, sex),
  entry = 0.20, stay = 0.05, confounding = 0.10,
  or = TRUE, event = "Yes", show = FALSE
)

# Modified Poisson models for RR or PR
tabmulti(d, vars = vars(b2.sex, c.age, c.bmi, b2.smoking),
  by = outcome, methods = "full", rr = TRUE,
  event = "Yes", show = FALSE)
tabmulti(d, vars = vars(b2.sex, c.age, c.bmi, b2.smoking),
  by = outcome, methods = "full", pr = TRUE,
  event = "Yes", show = FALSE)

# Display and output controls
tabmulti(d, vars = vars(b2.sex, c.age, c.bmi, b2.smoking),
  by = outcome, methods = c("full", "backward"),
  or = TRUE, event = "Yes", rrow = vars(smoking),
  pvalue = TRUE, bold_p = TRUE, p_bold = 0.05,
  template = "clean", raw = TRUE, name = TRUE,
  title = "Model-building comparison", show = FALSE)

# Active-data syntax
usedf(d)
tabmulti(vars = vars(b2.sex, c.age, c.bmi, b2.smoking),
  by = outcome, methods = "full", or = TRUE,
  event = "Yes", show = FALSE)

# LASSO requires glmnet; BMA/best are exhaustive and suit fewer candidates
if (requireNamespace("glmnet", quietly = TRUE)) {
  tabmulti(d, vars = vars(b2.sex, c.age, c.bmi, b2.smoking),

```

```

    by = outcome, methods = c("full", "lasso"), or = TRUE,
    event = "Yes", lasso_lambda = "lambda.1se", show = FALSE)
}
tabmulti(d, vars = vars(b2.sex, c.age, c.bmi, b2.smoking),
  by = outcome, methods = c("best", "bma"), or = TRUE,
  event = "Yes", criterion = "BIC", bma_pip = 0.50,
  max_subset_vars = 10, show = FALSE)

```

tabscale

Comprehensive Scale Analysis

Description

`tabscale()` provides a one-command, publication-ready psychometric report. It covers internal consistency, stability, equivalence, inter-rater reliability, measurement error, content validity, structural validity, convergent/discriminant and known-groups validity, criterion validity, measurement invariance, DIF screening, and responsiveness when the required data are supplied. Variable labels are used throughout whenever available.

Usage

```

tabscale(data = NULL, vars = NULL, factor = NULL, reverse = NULL,
  report = c("auto", "brief", "full", "custom"),
  range = NULL, score = c("mean", "sum"), min_valid = NULL,
  missing = c("pairwise", "complete"), cor_method = c("pearson", "spearman"),
  reliability = TRUE, bootstrap = 0, conf = 0.95,
  retest = NULL, parallel_form = NULL, raters = NULL,
  rater_type = c("auto", "continuous", "categorical"),
  content = NULL, content_cutoff = 3, content_max = 4,
  validity = TRUE,
  efa = NULL, efa_method = c("pa", "ml"), nfactor = NULL,
  rotation = c("varimax", "promax", "none"), parallel_iter = 100,
  cfa = NULL, ordered = FALSE, estimator = "auto",
  cfa_missing = "fiml", cfa_group = NULL,
  modification = FALSE, modification_min = 10,
  invariance = FALSE,
  invariance_levels = c("configural", "metric", "scalar", "strict"),
  dif = FALSE, convergent = NULL, discriminant = NULL,
  convergent_min = 0.50, discriminant_max = 0.30,
  known_groups = NULL,
  gold = NULL, event = NULL, direction = c("auto", "higher", "lower"),
  post = NULL,
  name = FALSE, digit = 2, p_digit = 3,
  template = c("journal", "clean", "minimal"),
  plot = TRUE, plot_types = "auto",
  viewer_plot_format = c("png", "svg"),
  append = NULL, file = NULL,

```

```
title = NULL, interpretation = FALSE,
raw = TRUE, show = TRUE, seed = NULL)
```

Arguments

<code>data</code>	Optional data frame. When omitted, the active R4VN data frame is used.
<code>vars</code>	Items created by <code>vars()</code> , a character vector, or a one-sided formula.
<code>factor</code>	Optional named list defining subscales/CFA factors.
<code>reverse</code>	Optional items to reverse-score.
<code>report</code>	Output profile. "auto" runs reliability plus EFA and any data-dependent modules requested by their arguments; "brief" omits factor models; "full" also runs CFA when a factor map and lavaan are available; "custom" follows the module switches exactly.
<code>range</code>	Two numeric values giving the minimum and maximum item score.
<code>score</code>	Calculate scale scores as the item "mean" or "sum".
<code>min_valid</code>	Minimum valid items. A value in (0,1] is treated as a proportion. A named vector can define separate minima for factors and Total.
<code>missing</code>	Correlation/covariance handling: pairwise or complete observations.
<code>cor_method</code>	Pearson or Spearman item correlations.
<code>reliability</code>	Logical; calculate reliability statistics.
<code>bootstrap</code>	Number of nonparametric bootstrap replicates for alpha and omega-total confidence intervals. Zero uses a Feldt interval for alpha.
<code>conf</code>	Confidence level for alpha and AUC intervals.
<code>retest</code>	Items measured again, in the same order as <code>vars()</code> , for test-retest reliability, ICC, SEM, MDC, and Bland-Altman analysis.
<code>parallel_form</code>	Items from an equivalent form, in the same order as <code>vars()</code> , for parallel-form reliability.
<code>raters</code>	Two or more variables containing ratings of the same subjects.
<code>rater_type</code>	Treat ratings as continuous or categorical; "auto" chooses categorical for variables with at most 10 observed levels.
<code>content</code>	Expert-by-item matrix/data frame of content-relevance ratings.
<code>content_cutoff</code>	Minimum rating counted as content-relevant.
<code>content_max</code>	Maximum possible content rating, retained in the report.
<code>validity</code>	Logical master switch for validity modules.
<code>efa</code>	Logical or NULL; run EFA, KMO, Bartlett, and parallel analysis.
<code>efa_method</code>	Principal-axis ("pa") or maximum-likelihood ("ml") EFA.
<code>nfactor</code>	Number of EFA factors. When NULL, parallel analysis is used.
<code>rotation</code>	EFA rotation.
<code>parallel_iter</code>	Number of Monte Carlo samples for parallel analysis.
<code>cfa</code>	Logical or NULL; run CFA using the optional lavaan package.

ordered	Logical or character item names treated as ordinal in CFA.
estimator	CFA estimator. "auto" uses WLSMV for ordered items and MLR otherwise.
cfa_missing	Missing-data option passed to lavaan for non-ordinal CFA.
cfa_group	Optional grouping variable name for multiple-group CFA.
modification	Logical; include large CFA modification indices.
modification_min	Minimum modification index displayed.
invariance	Logical; test configural, metric, scalar, and strict measurement invariance across cfa_group.
invariance_levels	Invariance levels to fit.
dif	Logical; screen uniform and non-uniform differential item functioning across a two-level cfa_group.
convergent	External variables used for convergent validity.
discriminant	External variables used for discriminant validity.
convergent_min	Prespecified minimum absolute convergent correlation.
discriminant_max	Prespecified maximum absolute discriminant correlation.
known_groups	Grouping variable for known-groups validity.
gold	Optional criterion or binary gold-standard variable.
event	Event level for binary gold-standard ROC analysis.
direction	Whether higher or lower scores predict the event; "auto" chooses the direction with AUC at least 0.5.
post	Post-intervention/follow-up items, in the same order as vars(), for responsiveness (effect size and standardized response mean).
name	FALSE does not modify data. TRUE creates scale_total and subscale variables; a character value supplies the score prefix.
digit	Decimal places for estimates.
p_digit	Decimal places for p-values.
template	HTML style.
plot	Logical; create all applicable graphics in both the R Plots pane and the HTML Viewer.
plot_types	"auto", "all", or any of "items", "distributions", "correlation", "reliability", "scores", "scree", "loadings", "cfa_loadings", "roc", "test_retest", "parallel_form", "inter_rater", "content", "external_validity", "known_groups", "invariance", and "responsiveness". The automatic set omits secondary plots that often add visual clutter: item completeness, item-response distributions, item-reliability diagnostics, external-validity correlations, known-groups boxplots, and individual responsiveness trajectories. Their statistical tables remain in the report. Request any of these names explicitly, or use plot_types = "all", when the graphic is needed.

<code>viewer_plot_format</code>	Format used to embed plots in the HTML Viewer. The default "png" uses high-resolution, self-contained images and gives consistent fonts in RStudio Viewer, Chrome, and saved HTML files. "svg" retains vector graphics but may render text differently across browsers because SVG font substitution is controlled by the local system.
<code>append</code>	Optional previous R4VN table object or HTML file.
<code>file</code>	Optional HTML output path.
<code>title</code>	Optional table title.
<code>interpretation</code>	Logical; add cautious automatic interpretation. The default is FALSE.
<code>raw</code>	Logical; retain numerical result components.
<code>show</code>	Logical; open the HTML result.
<code>seed</code>	Optional random seed used by parallel analysis. The default NULL does not set a seed.

Details

The default report = "auto" produces descriptive item distributions, missing/floor/ceiling effects, corrected item-total correlations, alpha with 95% CI, standardized alpha, omega total, split-half coefficients, all six Guttman lambdas, KMO, Bartlett's test, parallel analysis, EFA, score distributions, and matching graphics. KR-20 is added for binary items. Ordinal alpha, omega hierarchical, and the greatest lower bound are added when the optional package psych is installed.

Additional data activate stability/test-retest, parallel-form, inter-rater, content, convergent, discriminant, known-groups, criterion, responsiveness, measurement-invariance, and DIF sections. CFA and invariance use lavaan. Face validity is inherently qualitative and is therefore identified in the coverage table rather than assigned a spurious numeric coefficient.

The same plot specifications are rendered in the R graphics device and in the HTML Viewer. In RStudio, use the Plots pane arrows to review every graph, or rerun selected graphs with `plot(result, which = "roc")`.

Value

Invisibly returns an object inheriting from `r4vn_tabscale` and `r4vn_tab`. Components include `tables`, `coverage`, `descriptive`, `reliability`, `scores`, `test_retest`, `parallel_form`, `inter_rater`, `content_validity`, `efa`, `cfa`, `convergent_validity`, `discriminant_validity`, `known_groups`, `criterion_validity`, `invariance`, `dif`, `responsiveness`, `plots`, `interpretation`, and `file`.

See Also

`vars`, `tab`, `tabmulti`, `tabexport`

Other R4VN tables: `tab()`, `tabexport()`, `tabforest()`, `tablong()`, `tabmeta()`, `tabmulti()`, `tabscore()`, `tabsurvey()`, `vars()`

Examples

```

set.seed(2026)
n <- 120
f1 <- rnorm(n)
f2 <- 0.35 * f1 + rnorm(n, sd = 0.94)
make_item <- function(z) as.integer(cut(z, quantile(z, 0:5/5),
                                         include.lowest = TRUE, labels = FALSE))

latent <- list(0.8*f1, 0.7*f1, 0.9*f1, -0.7*f1,
              0.8*f2, 0.7*f2, 0.9*f2, 0.6*f2)
base_items <- lapply(latent, function(z) make_item(z + rnorm(n)))
dat <- as.data.frame(base_items)
names(dat) <- paste0("q", 1:8)
for (j in 1:8) {
  dat[[paste0("q", j, "_retest")]] <- pmax(
    1,
    pmin(
      5,
      dat[[paste0("q", j)]] + sample(-1:1, n, TRUE, c(.1, .8, .1))
    )
  )
  dat[[paste0("q", j, "_formb")]] <- make_item(latent[[j]] + rnorm(n))
  dat[[paste0("q", j, "_post")]] <- pmax(1, pmin(5, dat[[paste0("q", j)]] + rbinom(n,1,.35)))
  attr(dat[[paste0("q", j)],], "label") <- paste("Well-being item", j)
}
dat$convergent_measure <- f1 + f2 + rnorm(n, sd=.6)
dat$unrelated_measure <- rnorm(n)
dat$known_group <- factor(ifelse(f1+f2>0,"Higher expected score","Lower expected score"))
dat$gold <- factor(ifelse(f1 + f2 + rnorm(n) > 0, "Yes", "No"),
                  levels = c("No", "Yes"))
dat$rater1 <- sample(1:4,n,TRUE); dat$rater2 <- dat$rater1
dat$rater3 <- dat$rater1
dat$rater2[sample(n,30)] <- sample(1:4,30,TRUE)
dat$rater3[sample(n,35)] <- sample(1:4,35,TRUE)
attr(dat$known_group,"label") <- "Prespecified clinical group"
attr(dat$gold,"label") <- "Clinical gold standard"

# 1. One-command automatic report: reliability, factorability, EFA, plots.
tb <- tabscale(
  dat,
  vars = vars(q1, q2, q3, q4, q5, q6, q7, q8),
  factor = list(Domain1 = vars(q1, q2, q3, q4),
                Domain2 = vars(q5, q6, q7, q8)),
  reverse = vars(q4), range = c(1, 5),
  nfactor = 2, parallel_iter = 10,
  plot = FALSE, show = FALSE
)
tb$reliability_summary
tb$efa$loadings
if (interactive()) {
  plot(tb) # all plots in the Plots pane
  plot(tb, which = "reliability") # one selected plot
}

```

```

# 2. Stability, parallel forms, inter-rater reliability, and measurement error.
rel <- tabscale(
  dat, vars=vars(q1,q2,q3,q4,q5,q6,q7,q8), reverse=vars(q4), range=c(1,5),
  retest=vars(q1_retest,q2_retest,q3_retest,q4_retest,
    q5_retest,q6_retest,q7_retest,q8_retest),
  parallel_form=vars(q1_formb,q2_formb,q3_formb,q4_formb,
    q5_formb,q6_formb,q7_formb,q8_formb),
  raters=vars(rater1,rater2,rater3),
  plot=FALSE, show=FALSE)
rel$test_retest$table
rel$parallel_form$table
rel$inter_rater$table

# 3. Convergent, discriminant, known-groups, criterion validity,
#   responsiveness, and automatic ROC curves.
val <- tabscale(
  dat, vars=vars(q1,q2,q3,q4,q5,q6,q7,q8), reverse=vars(q4), range=c(1,5),
  convergent=vars(convergent_measure), discriminant=vars(unrelated_measure),
  known_groups=known_group, gold=gold, event="Yes",
  post=vars(q1_post,q2_post,q3_post,q4_post,q5_post,q6_post,q7_post,q8_post),
  interpretation=TRUE, plot=FALSE, show=FALSE)
val$convergent_validity
val$discriminant_validity
val$known_groups$tests
val$criterion_validity$table
val$responsiveness$table

# 4. Content validity: experts in rows and items in columns.
expert_ratings <- as.data.frame(matrix(sample(2:4, 6*8, TRUE,
  prob=c(.10,.30,.60)), nrow=6, dimnames=list(NULL,paste0("q",1:8))))
content_result <- tabscale(
  dat, vars=vars(q1,q2,q3,q4,q5,q6,q7,q8), range=c(1,5),
  content=expert_ratings, content_cutoff=3,
  plot=FALSE, show=FALSE)
content_result$content_validity$summary
content_result$content_validity$item

# 5. CFA, composite reliability, AVE, HTMT, Fornell-Larcker,
#   measurement invariance, and DIF screening.
if (interactive() && requireNamespace("lavaan", quietly = TRUE)) {
  cfa_result <- tabscale(
    dat, vars=vars(q1,q2,q3,q4,q5,q6,q7,q8),
    factor=list(Domain1=vars(q1,q2,q3,q4), Domain2=vars(q5,q6,q7,q8)),
    reverse=vars(q4), range=c(1,5), cfa=TRUE, ordered=TRUE,
    cfa_group=known_group, invariance=TRUE, dif=TRUE,
    modification=TRUE, plot=FALSE, show=FALSE)
  cfa_result$cfa$reliability
  cfa_result$cfa$htmt
  cfa_result$invariance$table
  cfa_result$dif
}

```

tabscore

Build, simplify, validate and present a clinical/statistical scorecard

Description

`tabscore()` converts a multivariable prediction model into a complete, publication-ready scorecard. The function is designed for the full workflow, not merely for rounding regression coefficients. In one call it can:

- develop or accept a final prediction model;
- optionally select predictors from a candidate set;
- create score-ready categories for continuous predictors;
- derive an exact/model score and a simpler clinical integer score;
- produce a complete Predictor–Category–Point table and theoretical total score range;
- map every possible total score to predicted risk (or expected count/rate);
- select clinically/statistically useful score cutoffs;
- report diagnostic/prognostic properties at the selected cutoff;
- compare the original model, the model score and the clinical score;
- perform bootstrap internal validation of the *entire development pipeline*;
- optionally evaluate an external validation data set; and
- retain plot-ready data and prediction methods for deployment in R4VN Studio.

Version 1 supports logistic regression, Cox proportional hazards regression, and Poisson regression. Logistic and Cox models receive the most complete discrimination/cutoff workflow. A Poisson model may be used for count/rate scores; when its outcome is binary, ROC/cutoff summaries are also available.

Usage

```
tabscore(
  outcome,
  predictors = NULL,
  data = NULL,
  family = c("auto", "logistic", "cox", "poisson"),
  event = NULL,
  time = NULL,
  times = NULL,
  cutoff_time = NULL,
  select = c("none", "full", "backward", "forward", "purposeful", "lasso"),
  force = NULL,
  exclude = NULL,
  entry = 0.25,
  stay = 0.1,
  confound = 0.15,
```

```

cuts = NULL,
continuous = c("easy", "auto", "quantile", "keep"),
bins = 4L,
points = "auto",
pdo = 20,
maxscore = "auto",
simplify = TRUE,
tolerance = 0.01,
riskonly = TRUE,
scoreref = "lowest",
cutoff = c("iu", "youden", "risk", "prevalence", "sens", "spec", "cost", "manual",
  "refprob", "none"),
riskcut = NULL,
cutoff_value = NULL,
sens = NULL,
spec = NULL,
cost_fp = 1,
cost_fn = 1,
refprob = NULL,
refcut = NULL,
validate = c("bootstrap", "none"),
bootstrap = 500L,
validation = NULL,
risktable = TRUE,
compare = TRUE,
calibration = TRUE,
decision = TRUE,
plot = TRUE,
show = TRUE,
console = FALSE,
seed = NULL,
ai = FALSE
)

```

Arguments

outcome	Outcome variable. It can be an unquoted variable name, a character variable name, an already fitted glm/coxph model, or an R4VN regression result containing <code>raw\$model</code> (for example from <code>logistic()</code> or <code>poisson()</code>). For Cox analysis this is the event/status indicator; supply follow-up time in <code>time=</code> . Binary outcomes may be numeric/logical/factor/character.
predictors	Candidate predictors. Accepts a character vector, unquoted variables inside <code>c()</code> , or an R4VN <code>vars()</code> expression. Predictors are treated as model <i>variables</i> rather than individual dummy coefficients. When <code>vars()</code> is used, its R4VN declarations are honored: no prefix and <code>b2.</code> , <code>b3.</code> , ... declare categorical predictors and select the corresponding factor reference; <code>c.</code> , <code>q.</code> and <code>f.</code> declare numeric predictors as continuous for model fitting. Numeric variables carrying complete named value labels are also treated as categorical scorecard variables. Omit this

	argument when outcome is an already fitted model.
data	Data frame. When omitted, <code>tabscore()</code> attempts to use the active R4VN data set created by <code>usedf()</code> . When outcome is an already fitted model, the stored model frame is the authoritative development sample; data is not used to refit or silently change that model.
family	Model family: <code>auto</code> , <code>logistic</code> , <code>cox</code> , or <code>poisson</code> . With <code>auto</code> , the presence of <code>time=</code> selects Cox; otherwise a binary outcome selects logistic and a non-negative integer count selects Poisson.
event	Event level for a binary outcome/status. For 0/1 outcomes the default is 1. For a two-level factor the default is its second factor level; for character outcomes it is the second observed non-missing value. Set this explicitly whenever the event direction matters, e.g. <code>event="Co"</code> .
time	Cox follow-up-time variable, supplied as an unquoted or character variable name. This is the observed time, not the prediction horizon.
times	Prediction horizons for Cox score-to-risk tables, e.g. <code>times=c(1,3,5)</code> . When omitted, useful event-time quantiles are selected.
cutoff_time	Time horizon at which a Cox cutoff is evaluated. Defaults to the largest value in <code>times</code> , which is often the main clinical horizon.
select	Predictor-selection strategy. <code>none</code> and <code>full</code> keep all supplied predictors. <code>backward/forward</code> use AIC stepwise selection. <code>purposeful</code> uses univariable screening, multivariable removal, confounding assessment and re-entry. <code>lasso</code> uses cross-validated <code>glmnet</code> at <code>lambda.1se</code> (falling back to <code>lambda.min</code> if necessary), then refits a standard model.
force	Predictors that must remain in model-selection procedures. Accepts character names, <code>c(...)</code> , or <code>vars(...)</code> , using the same naming conventions as predictors.
exclude	Candidate predictors to remove before model development. Accepts character names, <code>c(...)</code> , or <code>vars(...)</code> .
entry	Univariable screening p-value for purposeful selection; default 0.25.
stay	Multivariable retention/re-entry p-value for purposeful selection; default 0.10.
confound	Relative coefficient-change threshold used to retain a variable as a confounder during purposeful selection; default 0.15 (15 percent).
cuts	Optional named list of user-defined cut points for continuous predictors, e.g. <code>list(age=c(40,50,60), bmi=c(23,25,30))</code> . These are used in the score-ready model and therefore in the scorecard. As a convenience, a single character value <code>"easy"</code> , <code>"auto"</code> , <code>"quantile"</code> , or <code>"keep"</code> is accepted as an alias for <code>continuous=</code> . The original final model remains available for comparison.
continuous	Handling of continuous predictors when the clinical scorecard is built. <code>easy</code> (default) uses quantile-informed cut points snapped to easy-to-use numbers; <code>auto</code> is an alias; <code>quantile</code> uses unsnapped empirical quantiles; <code>keep</code> leaves continuous predictors continuous in the score-ready model. A complete bedside Predictor–Category–Point table nevertheless needs explicit categories, so use <code>cuts=</code> when <code>keep</code> is requested.
bins	Desired number of categories when automatic continuous-variable categorization is used. Default 4.

points	Point-construction method. <code>auto</code> (or <code>clinical/integer</code>) searches for a small integer score that preserves score-ready model discrimination within tolerance. <code>model</code> or <code>pdo</code> uses model/PDO scaling. A named list or a data frame with columns Predictor, Category and Point may be supplied for a completely user-defined integer score. With the default <code>riskonly=TRUE</code> , manual points are shifted within each predictor so its minimum becomes zero; therefore even a supplied negative protective point is converted to an equivalent add-only risk score. With <code>riskonly=FALSE</code> , negative manual points are retained. <code>tabscore()</code> still calibrates, tests, compares and validates the manual score. When manual points refer to categories of a continuous predictor, also supply explicit <code>cuts=</code> so the category definitions remain fixed during validation. Named point vectors are matched to displayed category labels after normalizing common typographic equivalents, so ASCII input such as 40-49 and >=60 also matches publication labels such as 40-49 and >=60. Unnamed vectors are matched in displayed category order.
pdo	Points to double the effect on the model's log scale. For logistic regression this is Points to Double the Odds: $\text{factor} = \text{pdo} / \log(2)$. For Cox the same number of points doubles hazard; for Poisson it doubles modeled rate. Default 20. The PDO/model score is retained separately from the clinical score.
maxscore	Maximum preferred theoretical clinical score. <code>auto</code> searches compact totals (approximately 5-30 points). A numeric value constrains the theoretical range. This range is based on all possible scorecard categories, not merely the observed sample minimum/maximum.
simplify	Logical. With <code>points="auto"</code> , <code>TRUE</code> (default) searches for a compact clinical integer score; <code>FALSE</code> uses rounded PDO/model points instead, which is useful when preserving model-scale resolution is more important than bedside compactness.
tolerance	Maximum tolerated decrease in the primary discrimination metric when simplifying to the clinical score. For binary models this is AUC; for Cox it is C-index. The same threshold is also used to flag excessive loss caused by automatic predictor categorization. Default 0.01. If no compact score meets the tolerance, the best-performing candidate is retained and a warning is stored.
riskonly	Logical. Default <code>TRUE</code> . Build the bedside score as a pure add-only risk score: each predictor is re-referenced to its lowest modeled risk category, all scoring effect ratios are at least 1, and all clinical points are non-negative. For example, if Male is the regression reference and Female has $\text{OR}=0.50$, the score representation becomes Female=0 points (score reference) and Male has risk-oriented $\text{OR}=2.00$ with positive points. This reparameterization does not change the fitted model, subject ranking, or predicted risks. Set <code>FALSE</code> only when a signed score with negative protective points is specifically desired.
scoreref	Scoring reference. Default "lowest" chooses the lowest-risk category independently within every categorical predictor. "model" uses each fitted model reference and is mainly useful with <code>riskonly=FALSE</code> . A named list/vector can set explicit scoring references, for example <code>list(sex="Female", exercise="Yes")</code> . With <code>riskonly=TRUE</code> , an explicitly requested reference must be one of the predictor's lowest-risk categories; otherwise <code>tabscore()</code> stops because satisfying that reference would require negative risk points.

cutoff	Selected cutoff principle: <i>iu</i> , <i>youden</i> , <i>risk</i> , <i>prevalence</i> , <i>sens</i> , <i>spec</i> , <i>cost</i> , <i>manual</i> , <i>refprob</i> , or <i>none</i> . All available methods are shown; this argument chooses the threshold used for final classification.
riskcut	One or more clinically meaningful probability thresholds. With <i>cutoff="risk"</i> , the first value is converted to the nearest integer score. Multiple values create <i>Low/Intermediate/High/etc.</i> groups in the risk table.
cutoff_value	User-specified integer score threshold for <i>cutoff="manual"</i> .
sens	Minimum desired sensitivity for <i>cutoff="sens"</i> , e.g. 0.90.
spec	Minimum desired specificity for <i>cutoff="spec"</i> , e.g. 0.90.
cost_fp	Relative cost assigned to a false positive. Default 1.
cost_fn	Relative cost assigned to a false negative. Example: <i>cost_fn=5</i> makes a false negative five times as costly as a false positive when <i>cost_fp=1</i> .
refprob	Optional reference predicted probability for binary-outcome workflows: a numeric vector or a probability-variable name in data. <i>tabscore()</i> reports correlation, MAE, RMSE and mean difference between reference and score-derived probabilities. It is not used as a surrogate outcome and is not called a gold standard unless it truly is one.
refcut	Probability threshold applied to <i>refprob</i> . With <i>cutoff="refprob"</i> , the score threshold that best reproduces this reference classification is selected, then evaluated against the actual outcome.
validate	Internal validation: <i>bootstrap</i> (default) or <i>none</i> . Bootstrap validation reruns the whole development process inside each resample, including selection, automatic cuts and point simplification.
bootstrap	Number of bootstrap resamples. Default 500. Use 1000 or more for a final analysis when feasible; small values are for code testing only.
validation	Optional external validation data frame. The frozen final scorecard is applied without re-estimating cuts, points, calibration or the chosen score cutoff. Binary validation reports AUC, Brier, calibration and fixed-cutoff performance; Cox validation reports C-index, time-specific IPCW Brier and fixed-cutoff IPCW performance; count Poisson reports prediction error.
risktable	Logical; create score-to-risk/score-to-expected-value table.
compare	Logical; compare original model, model score and clinical score.
calibration	Logical; prepare calibration data for plots.
decision	Logical; prepare decision-curve net-benefit data for binary outcomes.
plot	Logical. Default TRUE. Prepare all graphics supported by the chosen model family, embed every available graph directly in the HTML Viewer, and draw them into the interactive R/RStudio Plot history. Set FALSE when only tables are wanted. Plotting uses base R and does not add a required package.
show	Logical. If TRUE, write a self-contained publication-oriented HTML result and open it in the RStudio Viewer (or the default browser).
console	Logical. If TRUE, print a concise console summary.
seed	Optional random seed used for LASSO, bootstrap fallback and validation. The default NULL does not set a seed.
ai	Logical or endpoint name. If R4VN <i>aiask()</i> is available, request an optional AI interpretation after the statistical object is complete.

Details

Missing data and development sample:

When `tabscore()` develops a model from candidate predictors, it uses one complete-case development sample across the outcome/status, Cox time (when applicable), and all candidate predictors supplied before model selection. This keeps candidate models comparable but can reduce sample size when many predictors have missing values. Perform the intended imputation or missing-data strategy before `tabscore()` when complete-case analysis is inappropriate.

R4VN vars() declarations and reference categories:

`tabscore()` understands the R4VN variable culture rather than merely stripping prefixes. For example, `vars(c.age, b2.sex, smoking)` fits age continuously, treats sex as categorical with its second factor level as model reference, and treats smoking as categorical with its first factor level as reference. This affects the fitted model table and model-selection calculations. Point assignment itself is then shifted within each predictor so the lowest-risk category receives zero automatic points; consequently the zero-point category does not have to be the regression reference category. The final prediction-model table preserves the original statistical reference and may therefore legitimately show OR/HR/RR below 1. The separate `risk_orientation` table shows the scoring contrast after re-referencing; with `riskonly=TRUE`, every displayed scoring ratio is ≥ 1 and the clinical score contains only zero or positive points. With ordinary `c(age, sex)` syntax, data type is inferred from the columns instead of imposing R4VN categorical declarations.

Original model, model score and clinical score:

`tabscore()` distinguishes three objects. The **original model** is the selected or fixed model using the original predictor representation. The **model score** is a monotone point transformation of the score-ready model linear predictor. With `pdo=20`, a 20-point increase doubles odds (logistic), hazard (Cox), or modeled rate (Poisson). The **clinical score** uses small integer points and is the score shown in the clean Predictor–Category–Point table.

Intercepts are never artificially divided among predictors; they remain in the risk mapping. Within each predictor, the lowest modeled contribution is shifted to zero before automatic non-negative points are assigned. Thus a zero-point category need not be the regression reference if another category has lower risk.

Protective factors and add-only risk scoring:

With `riskonly=TRUE`, categorical contributions are transformed predictor by predictor as `beta_score = beta_category - min(beta_categories)`. Therefore `exp(beta_score) >= 1`. For a binary predictor with an original protective contrast `OR=0.50`, reversing the scoring contrast gives $1/0.50=2.00$ for the higher-risk category. For multi-level predictors the same minimum-risk rebasing is used; the procedure is not `abs(beta)`, which can distort category ordering. Cox HR and Poisson RR/IRR are handled identically on their log-effect scales. For a continuous coefficient retained without categories, a negative coefficient is described technically as risk per unit decrease; a complete bedside integer score still requires explicit/automatic categories. The transformation changes only the score origin/reference: the original fitted model and its absolute-risk predictions remain untouched.

Starting from an already fitted model:

A base `glm/coxph` model or an R4VN regression result containing `raw$model` may be passed as outcome. In that workflow the fitted predictor set is fixed, so `select`, `force` and `exclude` are not used. This version deliberately rejects interactions, transformed/spline terms, no-intercept

models, non-unit analysis weights, and non-zero offsets/exposures rather than silently changing the fitted model during score simplification. Represent required transformed predictors as explicit columns and refit before calling `tabscore()`, or provide a manual point system.

Automatic and manual cut points:

Automatic scorecard categorization is a simplification step rather than part of the original continuous model. `continuous="easy"` uses empirical quantiles and snaps thresholds toward simple numbers. Prefer clinically established cuts= when available. Because automatic cuts are data-driven, bootstrap validation repeats the cut-selection step inside each resample.

Theoretical total score:

The displayed range is calculated from all category combinations implied by the scorecard, not from the smallest/largest observed subject score. This keeps the bedside score stable in new data.

Score-to-risk conversion:

For logistic models, the clinical score is recalibrated by $\text{logit}(P)=a+b*\text{Score}$; every possible total score receives a predicted probability and 95 percent CI. This recalibration is important because categorization and integer rounding mean the clinical score is no longer exactly the original LP. Cox scorecards use a one-predictor Cox calibration model to provide risk at each times=horizon. Poisson count scores provide expected count/rate and CI.

Cutoff methods:

- `youden`: maximize sensitivity + specificity - 1.
- `iu`: minimize $\text{abs}(\text{sensitivity}-\text{AUC})+\text{abs}(\text{specificity}-\text{AUC})$.
- `risk`: map a clinical probability in `riskcut` to an integer score.
- `prevalence`: use development event prevalence as probability threshold, reproducing a common legacy workflow; it is not automatically clinically best.
- `sens`: among thresholds meeting `sens`, maximize specificity.
- `spec`: among thresholds meeting `spec`, maximize sensitivity.
- `cost`: minimize $\text{cost_fn}*FN + \text{cost_fp}*FP$.
- `manual`: use `cutoff_value`.
- `refprob`: use `refprob` and `refcut` to reproduce a reference probability rule.

Cox support in this version is for standard right-censored proportional-hazards models. Start-stop/time-dependent, multi-state, competing-risk and other complex survival structures are not silently reduced to a simple integer score. Cox ROC/cutoff calculations at `cutoff_time` use cumulative/dynamic IPCW sensitivity and specificity with the censoring distribution estimated by Kaplan-Meier. IPCW cutoff sensitivity, specificity, PPV, NPV, accuracy and likelihood ratios are point estimates in the apparent cutoff table; unlike the ordinary binary-outcome table, simple binomial confidence intervals are not reported because censoring weights make them inappropriate. Full-pipeline bootstrap validation supplies optimism-corrected cutoff performance and the empirical stability interval of the selected score threshold. If an intervention probability is known, `riskcut` is generally easier to interpret clinically than a purely statistical cutoff.

Model versus score comparison:

A score is not accepted merely because an AUC difference is non-significant. Binary comparisons include AUC with 95 percent CI, Brier score, calibration intercept/slope, and paired AUC difference. Cox comparisons include C-index and time-specific IPCW Brier scores. Decision-curve data compare net benefit across probability thresholds.

Bootstrap internal validation:

Each bootstrap resample repeats predictor selection, automatic categorization, point derivation, and the requested cutoff-selection rule. Apparent performance is compared with performance when the bootstrap-derived score and cutoff are applied to the original sample. For binary outcomes the validation table includes optimism-corrected AUC, Brier score, calibration intercept/slope, sensitivity, specificity, PPV, NPV, accuracy, and bootstrap cutoff stability. For Cox models the validation table includes optimism-corrected C-index and, when a cutoff is requested, time-dependent IPCW sensitivity, specificity, PPV, NPV and accuracy at `cutoff_time` plus bootstrap cutoff stability. Mean optimism is subtracted from the apparent final performance. This is more rigorous than bootstrapping a fixed already-developed score.

External validation and deployment:

Supply `validation=` to apply the *frozen* developed score to a separate data set. Predictor cut points, integer points, risk mapping, and the selected score threshold are not re-optimized in the external data. This avoids turning an external validation into a second development exercise. After development, `predict(score_object, newdata, type="all")` returns bedside score, predicted risk/value and risk group where applicable. New factor values that were absent from the development scorecard cannot be scored and therefore yield missing score/risk rather than being silently assigned zero points.

Viewer, plots and package requirements:

With `show=TRUE`, the Viewer contains the publication tables and, when `plot=TRUE`, every graph that is actually available for the fitted family. Logistic/binary scorecards can show score-to-risk, ROC, calibration, decision curve and score-distribution plots. Cox scorecards show score-to-risk across requested horizons, time-dependent ROC at `cutoff_time`, and score distribution. Poisson count scorecards show score-to-expected-value and the observed-count distribution. The Viewer plots are rendered with base R, using a system sans-serif font and an embedded raster image when possible; this keeps the report self-contained and avoids a `ggplot2/htmlwidgets` dependency. Use `plot(result)` to draw all available graphs in the RStudio Plots pane, or `plot(result, which="roc")`, for example, to draw one.

The core logistic and Poisson workflows use base/recommended R only. `survival` is needed only for Cox scorecards, `glmnet` only for `select="lasso"`, and `pROC` is optional because `R4VN` has a base-R fallback for AUC calculations and paired AUC comparison. Thus users do not need to install a large collection of packages for ordinary `tabscore()` analyses.

Modeling cautions:

Prediction modeling is not equivalent to retaining only $p < 0.05$ predictors. Use subject-matter knowledge and `force=` for essential variables. Data-driven cuts and cutoffs can overfit and should be validated. Interactions, spline bases, time-varying Cox effects, competing risks and machine-learning distillation are not silently converted into a bedside integer score in this version.

Value

An object of class `r4vn_tabscore` with fitted models, score rules, development scores/predictions, selected cutoff, theoretical range, publication tables, technical tables, validation results and plot-ready data. Important elements include `models`, `scores`, `tables`, `publication_tables`, `selected_predictors`, `selected_cutoff`, `score_range`, `riskonly`, `effect_measure`, `plots`, `plot_titles` and `plot_data`. `tables$risk_orientation` explicitly compares the score-ready model-reference effect ratio with

the risk-oriented scoring ratio. The `publication_tables` list contains only ready-to-export non-NULL tables and can be passed directly to `tabexport()`. `predict()` can then score new patients without re-estimating the scorecard.

See Also

`predict.r4vn_tabscore`, `plot.r4vn_tabscore`

Other R4VN tables: `tab()`, `tabexport()`, `tabforest()`, `tablong()`, `tabmeta()`, `tabmulti()`, `tabscale()`, `tabsurvey()`, `vars()`

Examples

```
set.seed(2026)
n <- 220
d <- data.frame(
  age = round(rnorm(n, 52, 12)),
  bmi = round(rnorm(n, 24, 4), 1),
  hypertension = factor(rbinom(n, 1, .30), 0:1, c("No", "Yes")),
  smoking = factor(rbinom(n, 1, .25), 0:1, c("No", "Yes")),
  alcohol = factor(rbinom(n, 1, .20), 0:1, c("No", "Yes"))
)
lp <- -4.2 + .04*d$age + .06*(d$bmi - 24) +
  .8*(d$hypertension == "Yes") + .6*(d$smoking == "Yes")
d$event <- rbinom(n, 1, plogis(lp))

# 1. Simplest publication-ready logistic scorecard.
# show=TRUE and plot=TRUE are the user-facing defaults.
s1 <- tabscore(
  event, c(age, bmi, hypertension, smoking), data=d,
  validate="none", show=FALSE, plot=FALSE
)
s1$tables$scorecard
s1$tables$risk
s1$tables$comparison

# 2. R4VN variable declarations: continuous variables and chosen references.
s2 <- tabscore(
  event, vars(c.age, c.bmi, b2.hypertension, b2.smoking), data=d,
  validate="none", show=FALSE, plot=FALSE
)
s2$tables$model
s2$tables$risk_orientation

# 3. Clinically prespecified cut points.
s3 <- tabscore(
  event, c(age, bmi, hypertension, smoking), data=d,
  cuts=list(age=c(40,50,60), bmi=c(23,25,30)),
  validate="none", show=FALSE, plot=FALSE
)

# 4. Apply the frozen scorecard to new patients.
newp <- data.frame(
```

```

    age=c(45,68), bmi=c(24,29),
    hypertension=factor(c("No","Yes"), levels=c("No","Yes")),
    smoking=factor(c("Yes","No"), levels=c("No","Yes"))
  )
  predict(s3, newp, type="all")

# 5. Purposeful selection; force variables that must remain clinically.
s5 <- tabscore(
  event, c(age,bmi,hypertension,smoking,alcohol), data=d,
  select="purposeful", force=c("age","hypertension"),
  validate="none", show=FALSE, plot=FALSE
)

# 6. Backward or forward AIC selection.
s6a <- tabscore(event, c(age,bmi,hypertension,smoking,alcohol), data=d,
  select="backward", validate="none", show=FALSE, plot=FALSE)
s6b <- tabscore(event, c(age,bmi,hypertension,smoking,alcohol), data=d,
  select="forward", validate="none", show=FALSE, plot=FALSE)

# 7. LASSO is optional and only needs glmnet for this selection method.
if (requireNamespace("glmnet", quietly=TRUE)) {
  s7 <- tabscore(event, c(age,bmi,hypertension,smoking,alcohol), data=d,
    select="lasso", validate="none", show=FALSE, plot=FALSE)
}

# 8. Compact score versus PDO/model-scale points.
s8a <- tabscore(event, c(age,hypertension,smoking), data=d,
  cuts=list(age=c(40,50,60)), maxscore=10,
  validate="none", show=FALSE, plot=FALSE)
s8b <- tabscore(event, c(age,hypertension,smoking), data=d,
  cuts=list(age=c(40,50,60)), points="pdo", pdo=20,
  validate="none", show=FALSE, plot=FALSE)

# 9. Completely manual bedside points; R4VN still calibrates and validates it.
s9 <- tabscore(
  event, c(age,hypertension,smoking), data=d,
  cuts=list(age=c(40,50,60)),
  points=list(
    age=c("<40"=0, "40-49"=1, "50-59"=2, ">=60"=3),
    hypertension=c("No"=0,"Yes"=2),
    smoking=c("No"=0,"Yes"=1)
  ), validate="none", show=FALSE, plot=FALSE
)

# 10. Common cutoff rules.
s10_iu <- tabscore(event, c(age,hypertension,smoking), data=d,
  cutoff="iu", validate="none", show=FALSE, plot=FALSE)
s10_youden <- tabscore(event, c(age,hypertension,smoking), data=d,
  cutoff="youden", validate="none", show=FALSE, plot=FALSE)
s10_sens <- tabscore(event, c(age,hypertension,smoking), data=d,
  cutoff="sens", sens=.90, validate="none", show=FALSE, plot=FALSE)
s10_cost <- tabscore(event, c(age,hypertension,smoking), data=d,

```

```

        cutoff="cost", cost_fn=5, cost_fp=1,
        validate="none", show=FALSE, plot=FALSE)

# 11. Clinically meaningful probability threshold and multiple risk groups.
s11 <- tabscore(event, c(age,hypertension,smoking), data=d,
               riskcut=c(.05,.10,.20), cutoff="risk",
               validate="none", show=FALSE, plot=FALSE)
s11$tables$risk

# 12. Compare the score with an existing/reference probability.
d$reference_risk <- plogis(-4 + .04*d$age + .7*(d$hypertension == "Yes"))
s12 <- tabscore(event, c(age,hypertension,smoking), data=d,
               refprob=reference_risk, refcut=.10, cutoff="refprob",
               validate="none", show=FALSE, plot=FALSE)
s12$tables$reference_probability

# 13. Convert an already fitted logistic model.
m13 <- glm(event ~ age + hypertension + smoking, data=d, family=binomial())
s13 <- tabscore(m13, validate="none", show=FALSE, plot=FALSE)

# 14. External validation with a frozen scorecard.
dev <- d[1:150, ]
val <- d[151:nrow(d), ]
s14 <- tabscore(event, c(age,hypertension,smoking), data=dev,
               cuts=list(age=c(40,50,60)), validation=val,
               validate="none", show=FALSE, plot=FALSE)
s14$tables$external_validation

# 15. Full-pipeline bootstrap validation. B=20 is only a quick code check;
# use bootstrap=500 or more for the final report.
s15 <- tabscore(event, c(age,hypertension,smoking), data=d,
               cuts=list(age=c(40,50,60)),
               validate="bootstrap", bootstrap=20,
               show=FALSE, plot=FALSE)
s15$tables$validation

# 16. Cox scorecard: survival is the only package required for this family.
if (requireNamespace("survival", quietly=TRUE)) {
  ds <- d
  true_t <- rexp(nrow(ds), rate=exp(-3 + .02*ds$age +
    .6*(ds$hypertension == "Yes")))
  censor_t <- rexp(nrow(ds), rate=.08)
  ds$status <- as.integer(true_t <= censor_t)
  ds$ftime <- pmin(true_t, censor_t)
  sc <- tabscore(status, c(age,hypertension,smoking), data=ds,
               family="cox", time=ftime, times=c(1,3,5), cutoff_time=5,
               cuts=list(age=c(40,50,60)),
               validate="none", show=FALSE, plot=FALSE)
  sc$tables$risk
  sc$tables$time_brier
}

# 17. Poisson count scorecard.

```

```

dp <- d
dp$count <- rpois(nrow(dp), exp(-1 + .015*dp$age + .35*(dp$smoking == "Yes")))
sp <- tabscore(count, c(age,smoking), data=dp, family="poisson",
               cuts=list(age=c(40,50,60)), validate="none",
               show=FALSE, plot=FALSE)
sp$tables$risk

# 18. Every available plot; Viewer includes the same figures when show=TRUE.
sv <- tabscore(event, c(age,hypertension,smoking), data=d,
               cuts=list(age=c(40,50,60)), validate="none",
               show=FALSE, plot=TRUE)
plot(sv, which="risk")
plot(sv, which="roc")
plot(sv, which="calibration")
plot(sv, which="decision")
plot(sv, which="distribution")
plot(sv) # all available plots in Plot history

# 19. Protective predictors: default risk-only coding versus a signed score.
dr <- d
dr$exercise <- factor(rbinom(nrow(dr), 1, .55), 0:1, c("No", "Yes"))
dr$event2 <- rbinom(nrow(dr), 1,
                   plogis(-2.5 + .04*dr$age - .8*(dr$exercise == "Yes")))
srisk <- tabscore(event2, c(age,exercise), data=dr,
                  cuts=list(age=c(40,50,60)), riskonly=TRUE,
                  validate="none", show=FALSE, plot=FALSE)
ssigned <- tabscore(event2, c(exercise), data=dr,
                    points=list(exercise=c("No"=0,"Yes"=-2)),
                    riskonly=FALSE, scoreref="model",
                    validate="none", show=FALSE, plot=FALSE)

# 20. Additional cutoff strategies.
s20_prev <- tabscore(event, c(age,hypertension,smoking), data=d,
                    cutoff="prevalence", validate="none",
                    show=FALSE, plot=FALSE)
s20_spec <- tabscore(event, c(age,hypertension,smoking), data=d,
                    cutoff="spec", spec=.90, validate="none",
                    show=FALSE, plot=FALSE)
s20_manual <- tabscore(event, c(age,hypertension,smoking), data=d,
                      cutoff="manual", cutoff_value=3, validate="none",
                      show=FALSE, plot=FALSE)
s20_none <- tabscore(event, c(age,hypertension,smoking), data=d,
                    cutoff="none", validate="none",
                    show=FALSE, plot=FALSE)

# 21. Quantile-based automatic categorization.
s21 <- tabscore(event, c(age,bmi,hypertension,smoking), data=d,
                continuous="quantile", bins=4,
                validate="none", show=FALSE, plot=FALSE)

# 22. An R4VN logistic() result can be converted directly as well.
rfit <- logistic(event, vars=vars(c.age, hypertension, smoking),
                 data=d, show=FALSE)

```

```
s22 <- tabscore(rfit, validate="none", show=FALSE, plot=FALSE)

# 23. Prediction outputs after the scorecard is frozen.
predict(s3, newp, type="score")
predict(s3, newp, type="risk")
predict(s3, newp, type="group")
predict(s3, newp, type="model")

# 24. Export all publication-ready tables without another tabscore-specific
# dependency. Word/Excel writers are only needed when those formats are chosen.
# tabexport(sv$publication_tables, export=c("html","docx","xlsx"),
#           file="tabscore_report", open=FALSE)
```

tabsurv	<i>Comprehensive Survival Analysis Table</i>
---------	--

Description

Performs descriptive survival analysis, Kaplan-Meier/Aalen-Johansen estimates, optional life tables, cumulative incidence at selected times, incidence rate, log-rank tests, Cox regression, proportional-hazards diagnostics, RMST, competing-risk Fine-Gray models, and counting-process/recurrent-event Cox models.

Usage

```
tabsurv(
  time,
  event,
  vars = NULL,
  by = NULL,
  data = NULL,
  failure = NULL,
  compete = NULL,
  id = NULL,
  start = NULL,
  unit = NULL,
  followup = NULL,
  km = NULL,
  lifetable = FALSE,
  at = NULL,
  risk = NULL,
  cuminc = NULL,
  rate = NULL,
  scale = 100,
  logrank = NULL,
  rr = NULL,
  rd = NULL,
```

```

irr = NULL,
cox = NULL,
adjusted = NULL,
multi = NULL,
strata = NULL,
cluster = NULL,
frailty = NULL,
finegray = NULL,
recurrent = FALSE,
rmst = NULL,
tau = NULL,
ph = NULL,
interaction = FALSE,
superby = NULL,
ci = 0.95,
digit = 2,
p_digit = 3,
effect_digit = 2,
missing = FALSE,
plot = NULL,
title = NULL,
show = TRUE,
console = FALSE,
ai = FALSE,
ties = c("efron", "breslow", "exact"),
report = c("auto", "brief", "full", "custom"),
plot_args = list(),
interpretation = FALSE,
export = NULL,
file = NULL,
open = FALSE,
strict = FALSE
)

```

Arguments

time	Follow-up or stop-time variable, supplied without quotes.
event	Event/status variable, supplied without quotes.
vars	Optional predictor specification created by <code>vars()</code> .
by	Optional grouping variable for survival curves and comparisons. Hierarchical syntax is supported: in <code>by = vars(province, sex, treatment)</code> , <code>treatment</code> is the innermost curve/comparison group and <code>province > sex</code> are ordered outer strata.
data	Optional data frame. When omitted, active R4VN data are used.
failure	Value of event representing the event of interest. For a binary event it defaults to the second factor level or larger numeric value.
compete	Optional competing-event value(s). When supplied, <code>risk = TRUE</code> uses the Aalen-Johansen cumulative incidence function.

id	Optional subject identifier for counting-process/recurrent data.
start	Optional start/entry time. When supplied, time is treated as stop time.
unit	Optional display unit such as "day", "month", or "year".
followup	Estimate median follow-up using reverse Kaplan-Meier when possible. With report = "auto", this is enabled unless explicitly set to FALSE.
km	Fit Kaplan-Meier (ordinary survival) or Aalen-Johansen (competing risks).
lifetable	Show a detailed life table at every observed time. The default is FALSE. For ordinary survival, the table reports numbers at risk, events, censoring, conditional survival, cumulative Kaplan-Meier survival, cumulative risk, standard error, and confidence limits. With competing risks, it reports the corresponding Aalen-Johansen event-history table and cumulative incidence.
at	Optional time points for survival/risk/rate summaries. With report = "auto" or "full", three representative follow-up times are selected automatically when at is omitted.
risk	Report cumulative risk at at. For ordinary survival this is 1-S(t); with competing risks it is the cumulative incidence function.
cuminc	Optional numeric time points at which cumulative incidence is required, for example cuminc = c(6, 12, 24). This directly activates cumulative-risk output without also requiring risk = TRUE. Ordinary survival uses 1-KM; competing-risk analysis uses the Aalen-Johansen CIF.
rate	FALSE, TRUE, "overall", "cumulative", "interval", or "all". "all" reports overall, cumulative, and interval-specific rates. The automatic profile uses the overall incidence rate.
scale	Rate multiplier, e.g. 100 for events per 100 person-time units.
logrank	Perform a log-rank test when by is supplied and no competing risk exists.
rr, rd	Compare cumulative risks between two by groups using approximate risk ratio or risk difference inference based on survival-estimate standard errors.
irr	Compare incidence rates between two by groups.
cox	Fit crude Cox models for variables in vars.
adjusted	FALSE/NULL, TRUE (adjust each focal predictor for all other focal predictors), or a vars()/character set of adjustment covariates.
multi	FALSE/NULL, TRUE (all vars in one model), or a vars()/character set defining the final multivariable Cox model.
strata	Optional stratification variable for Cox regression.
cluster	Optional clustering variable for robust Cox variance.
frailty	Optional shared-frailty variable. Do not combine with cluster.
finegray	Fit a Fine-Gray subdistribution hazards model when compete is supplied.
recurrent	FALSE/TRUE or "ag". TRUE is Andersen-Gill and requires id and start. Automatic profiles suppress ordinary KM/RMST modules for recurrent-event data unless the user explicitly requests them.
rmst	Compute restricted mean survival time.

<code>tau</code>	Restriction time for RMST. Defaults to the largest common curve time.
<code>ph</code>	Test the proportional-hazards assumption with <code>cox.zph()</code> for the final Cox model.
<code>interaction</code>	Optional <code>vars(a, b)</code> containing exactly two variables to include their interaction in the final multivariable Cox model.
<code>superby</code>	Optional outer subgroup variable retained for backward compatibility. For new code, multiple ordered outer strata can be supplied directly in <code>by = vars(stratum1, stratum2, group)</code> .
<code>ci</code>	Confidence level, default 0.95.
<code>digit, p_digit, effect_digit</code>	Display digits.
<code>missing</code>	Show missing/exclusion information when printing.
<code>plot</code>	Draw a survival/CIF curve using <code>gsurv()</code> after analysis. In the automatic profiles, the plot includes confidence limits, the log-rank p-value when available, and a number-at-risk table.
<code>title</code>	Optional title.
<code>show</code>	Logical; open the formatted result in the Viewer. Default TRUE.
<code>console</code>	Logical; also print the traditional result in the Console. Default FALSE.
<code>ai</code>	Prepare a compact de-identified interpretation payload in <code>\$ai_text</code> .
<code>ties</code>	Cox tie method: "efron", "breslow", or "exact".
<code>report</code>	Reporting profile: "auto" (context-sensitive comprehensive output), "brief" (descriptive survival summary), "full" (all valid modules), or "custom" (backward-compatible concise defaults plus explicitly requested modules).
<code>plot_args</code>	Named list of additional arguments passed to <code>gsurv()</code> .
<code>interpretation</code>	Add a cautious, deterministic interpretation table. The default is FALSE; use <code>interpretation = TRUE</code> when narrative output is wanted.
<code>export</code>	Optional export format accepted by <code>tabexport()</code> , such as "docx", "xlsx", or "html".
<code>file</code>	Optional export filename. Its extension may also determine the export format.
<code>open</code>	Open the exported file when supported.
<code>strict</code>	If TRUE, an unavailable optional module stops the analysis. The default FALSE keeps the main report and records a warning instead.

Value

An object of class `r4vn_surv`. Backward-compatible components are retained, with a consistent reporting contract in `$descriptive`, `$estimates`, `$tests`, `$diagnostics`, `$interpretation`, `$tables`, `$plots`, `$models`, `$metadata`, and `$call`.

Examples

```

if (requireNamespace("survival", quietly = TRUE)) {
  # Reproducible two-group data from the survival package.
  d <- survival::lung
  d$death <- as.integer(d$status == 2)
  d$group <- factor(d$sex, levels = c(1, 2),
                    labels = c("Male", "Female"))

  # 1. Complete two-group report. This includes the log-rank test.
  km <- tabsurv(
    time, death, by = group, data = d, failure = 1,
    unit = "day", at = c(90, 180, 365, 540),
    report = "auto", plot = FALSE, show = FALSE
  )
  km$logrank
  km$tests$logrank
  km$logrank$p

  # 2. Cumulative incidence at 6, 12, and 24 months.
  d$month <- d$time / 30.4375
  ci_month <- tabsurv(
    month, death, by = group, data = d, failure = 1,
    cuminc = c(6, 12, 24), report = "custom", show = FALSE
  )
  ci_month$cuminc

  # 3. Detailed life table and interpretation are both opt-in.
  km_detail <- tabsurv(
    time, death, by = group, data = d, failure = 1,
    at = c(90, 180, 365, 540),
    lifetable = TRUE, interpretation = TRUE, show = FALSE
  )
  head(km_detail$lifetable)
  km_detail$interpretation

  # 4. A compact KM plus log-rank analysis without automatic extras.
  km_simple <- tabsurv(
    time, death, by = group, data = d, failure = 1,
    report = "custom", km = TRUE, logrank = TRUE,
    plot = FALSE, show = FALSE
  )

  # 5. Explicit two-group effect measures and RMST.
  km_compare <- tabsurv(
    time, death, by = group, data = d, failure = 1,
    at = c(90, 180, 365, 540), risk = TRUE,
    rr = TRUE, rd = TRUE, rate = "all", irr = TRUE,
    rmst = TRUE, tau = 365, show = FALSE
  )
  km_compare$risk_compare
  km_compare$irr

```

```

km_compare$rmst

# 6. Publication graphs, including risk tables, are documented in ?gsurv.
# Keeping graphics out of this example also keeps tabsurv() examples fast
# and executable on non-interactive CRAN check devices.

# 7. With competing risks, use the Aalen-Johansen CIF, not 1-KM.
set.seed(2026)
n <- 180
t1 <- rexp(n, 0.07)
t2 <- rexp(n, 0.05)
tc <- runif(n, 4, 30)
tm <- pmin(t1, t2, tc)
dcr <- data.frame(
  time = tm,
  status = ifelse(tm == t1, 1L, ifelse(tm == t2, 2L, 0L)),
  group = factor(rep(c("A", "B"), each = n / 2))
)
cif <- tabsurv(
  time, status, by = group, data = dcr,
  failure = 1, compete = 2, cuminc = c(6, 12, 24),
  report = "custom", show = FALSE
)
cif$cuminc
# See ?gsurv for publication CIF graphs and risk tables.

}

```

Description

Performs R4VN-style descriptive analysis, hypothesis testing, and effect estimation for complex survey data. The interface deliberately mirrors `tab()` while adding survey weights, strata, clusters, replicate weights, domain analysis, design-based standard errors, weighted and unweighted results, and optional population totals.

Usage

```

tabsurvey(
  data = NULL,
  vars = NULL,
  by = NULL,
  design = NULL,
  weight = NULL,
  strata = NULL,
  cluster = NULL,
  fpc = NULL,

```

```
repweights = NULL,
rep_type = NULL,
weightscale = c("relative", "population"),
nest = TRUE,
subpop = NULL,
result = c("weighted", "unweighted", "both"),
bothstyle = c("columns", "rows"),
statcols = c("separate", "compact"),
rawn = TRUE,
digit = 1,
p_digit = 3,
effect_digit = 2,
level = 0.95,
missing = c("ifany", "no", "always"),
row = FALSE,
col = TRUE,
cell = FALSE,
overall = c("first", "last", "none"),
descriptive = TRUE,
rvrow = NULL,
rvcol = FALSE,
test = TRUE,
pvalue = TRUE,
survey_test = c("F", "Chisq", "Wald", "adjWald"),
or = FALSE,
rr = FALSE,
pr = FALSE,
event = NULL,
adjusted = NULL,
multi = NULL,
effect_ref = NULL,
ci = TRUE,
cimethod = c("logit", "likelihood", "beta", "mean", "asin", "xlogit"),
quantile_method = c("mean", "beta", "xlogit", "asin", "score", "quantile"),
se = FALSE,
deff = FALSE,
cv = FALSE,
population = FALSE,
lonely = NULL,
bold_p = TRUE,
p_bold = 0.05,
test_note = TRUE,
template = c("journal", "clean", "minimal"),
append = NULL,
file = NULL,
raw = FALSE,
name = FALSE,
title = NULL,
```

```

report = c("auto", "brief", "full", "custom"),
interpretation = FALSE,
show = TRUE
)

```

Arguments

<code>data</code>	Optional data frame. Normally omitted when a stored <code>surveyset()</code> design is used.
<code>vars</code>	Variables to summarize, created with <code>vars()</code> . R4VN prefixes are supported: unprefixed or <code>b2./b3.</code> categorical variables, <code>c.</code> mean/SD, <code>q.</code> median/IQR, and <code>f.</code> full continuous summaries. Deferred selectors such as <code>vars(.)</code> , wildcards, and exclusions are resolved against the survey data.
<code>by</code>	Optional outcome/grouping variable. An unprefixed variable is treated as categorical. Use <code>by = c.outcome</code> for a continuous outcome with mean-oriented inference or <code>by = q.outcome</code> for a continuous outcome with rank-oriented descriptive tests.
<code>design</code>	Survey design. May be an <code>r4vn_survey</code> object, a stored design name, or a design object from the survey package. If omitted, the active design created by <code>surveyset()</code> is used.
<code>weight, strata, cluster, fpc</code>	Direct design arguments for one-off analyses. These are alternatives to <code>design=</code> and have the same meaning as in <code>surveyset()</code> .
<code>repweights</code>	Optional replicate weights for a one-off design.
<code>rep_type</code>	Replicate design type when <code>repweights</code> is used.
<code>weightscale</code>	"relative" or "population" for a one-off design. Stored designs retain the value declared in <code>surveyset()</code> .
<code>nest</code>	Logical for a one-off multistage design.
<code>subpop</code>	Optional logical domain/subpopulation expression, for example <code>subpop = age >= 60 & sex == "Female"</code> . Domain estimation preserves the original survey design rather than naively rebuilding it after row deletion.
<code>result</code>	Which analysis system to show: "weighted" (default), "unweighted", or "both". "both" applies to descriptive statistics, tests, effect estimates, and confidence intervals, not only to percentages.
<code>bothstyle</code>	When <code>result = "both"</code> , "columns" puts weighted and unweighted results in parallel columns; "rows" stacks them using an Analysis column.
<code>statcols</code>	Presentation of descriptive statistics and effect estimates. "separate" (default) places sample n, estimate, confidence interval, SE, DEFF, CV, population N, model effect, model confidence interval, and p-value in separate publication-ready columns. "compact" keeps the older compact style in which estimates and confidence intervals are combined in one cell.
<code>rawn</code>	Include the actual unweighted sample n in descriptive cells. This is especially important beside weighted estimates and also keeps n visible for unweighted continuous summaries. The default is TRUE.
<code>digit</code>	Decimal places for descriptive estimates.

<code>p_digit</code>	Decimal places for p-values.
<code>effect_digit</code>	Decimal places for OR, PR, RR, and beta estimates.
<code>level</code>	Confidence level. The default is 0.95.
<code>missing</code>	"ifany", "no", or "always" for categorical missing-value rows.
<code>row, col, cell</code>	Percentage denominator for categorical variables when <code>by</code> is categorical. Exactly one should be TRUE. The default is column percentage, matching a conventional Table 1/Table 2 layout.
<code>overall</code>	Position of the overall descriptive column: "first" (default), "last", or "none".
<code>descriptive</code>	Logical. Include descriptive statistics.
<code>rvrow</code>	Optional categorical row reversal, matching <code>tab()</code> . Use TRUE to reverse every categorical variable, or identify selected variables with <code>vars(...)</code> , <code>c(...)</code> , or a character vector. Reversing display order does not silently change the regression reference.
<code>rvcol</code>	Logical. Reverse the displayed levels of a categorical by variable, matching <code>tab()</code> . Event selection still follows the original outcome order unless <code>event=</code> is supplied.
<code>test</code>	Logical. Include omnibus/group-comparison tests.
<code>pvalue</code>	Logical. Include coefficient-level p-values beside effect estimates.
<code>survey_test</code>	Statistic for categorical design-adjusted association tests passed to <code>survey::svychisq()</code> . The default "F" is the Rao-Scott second-order F correction. Other useful choices include "Chisq", "Wald", and "adjWald".
<code>or</code>	Logical. For a binary categorical outcome, estimate odds ratios using logistic regression.
<code>rr</code>	Logical. For a binary outcome, estimate risk/prevalence ratios with a log-link modified Poisson model. In cross-sectional surveys this is interpreted as a prevalence ratio.
<code>pr</code>	Logical. Estimate prevalence ratios with a log-link modified Poisson model. Weighted models use <code>survey::svyglm()</code> with <code>quasipoisson(link="log")</code> ; unweighted models use Poisson regression with a sandwich/robust covariance estimate.
<code>event</code>	Event level for a binary categorical outcome. By default the last observed outcome level is the event.
<code>adjusted</code>	Optional adjustment set. Supply <code>vars(...)</code> , a character vector, TRUE, or "ALL". A separate adjusted model is fitted for each focal predictor.
<code>multi</code>	Optional multivariable set. Supply <code>vars(...)</code> , a character vector, TRUE, or "ALL". Each reported focal effect comes from a model containing the complete requested multivariable set; this is equivalent to reporting coefficients from the common model. Reference prefixes inside <code>multi = vars(...)</code> are respected even when they differ from the descriptive/crude reference.
<code>effect_ref</code>	Optional backward-compatible explicit reference mapping for crude and separately adjusted categorical effects, for example <code>effect_ref = list(sex = "Male", smoking = "No")</code> . A named character vector is also accepted. <code>b2./b3.</code> prefixes remain the preferred compact R4VN syntax. Multivariable references come from <code>multi=</code> when that specification supplies its own prefix.

ci	Logical. Show confidence intervals at the selected level where they are available.
cimethod	Confidence-interval method for weighted proportions: "logit" (default), "likelihood", "beta", "mean", "asin", or "xlogit". If a method cannot handle an observed proportion of exactly 0 or 1, R4VN falls back to a design-based Wald interval and constrains displayed limits to the interval from 0 to 1.
quantile_method	Interval method used by <code>survey::svyquantile()</code> . The default is "mean"; alternatives supported by the installed survey version include "beta", "xlogit", and "asin". "score" is for ordinary survey designs; "quantile" is for replicate-weight designs and is not appropriate for jackknife quantile SEs.
se	Logical. Add a separate standard-error column for descriptive estimates. When unweighted results are requested, their conventional SE is also reported where defined.
deff	Logical. Add a separate with-replacement design-effect column for weighted statistics where the underlying survey statistic supports it.
cv	Logical. Add a separate coefficient-of-variation/relative-SE column where defined for weighted and unweighted descriptive estimates.
population	Logical. Append estimated population N and its confidence interval for categorical cells. This requires a design declared with <code>weightscale = "population"</code> . R4VN will not relabel normalized weights as population totals.
lonely	Optional lonely-PSU rule for this analysis. If omitted, the rule stored in the design is used.
bold_p	Logical. Bold p-values smaller than <code>p_bold</code> in HTML.
p_bold	Threshold used when <code>bold_p = TRUE</code> .
test_note	Logical. Add footnotes describing the tests used.
template	HTML style: "journal", "clean", or "minimal".
append	Optional previous R4VN table object to place before this table in the generated HTML page.
file	Optional HTML output path. A temporary file is used when omitted.
raw	Logical. Use raw variable names instead of variable labels.
name	Logical. When labels exist, append the raw variable name in square brackets.
title	Optional table title.
report	Reporting profile: "auto" (simple publication-ready survey output), "brief" (weighted descriptives only unless the user explicitly requests more), "full" (weighted and unweighted results stacked by rows with SE, DEFF, CV, tests, and model details where available), or "custom" (legacy defaults plus exactly the options requested by the user).
interpretation	Logical. Add a cautious deterministic interpretation table. The default is FALSE.
show	Logical. Open the generated HTML report in the Viewer/browser.

Details

Dependency-light implementation. Beyond R4VN itself, `tabsurvey()` requires only the **survey** package for complex-survey estimation. Publication HTML is generated with base R; **ggplot2**, **plotly**, **htmlwidgets**, **flextable**, and similar presentation packages are not required. `tabsurvey()` is a table/inference function and does not create a plot, so it deliberately adds no plotting dependency. R4VN functions that do create plots should embed every requested plot directly in their Viewer/HTML report.

Weighted and unweighted are complete analysis modes. With `result = "both"`, R4VN computes two parallel analyses. The unweighted side uses ordinary sample descriptions and conventional tests or regressions. The weighted side uses the declared survey design for descriptive estimates, standard errors, confidence intervals, Rao-Scott or design-based tests, and survey-weighted regression. This is intentionally more comprehensive than merely displaying a raw `n` beside a weighted percentage.

Default publication display. The default `statcols = "separate"` uses distinct columns for sample `n`, estimate, and confidence interval instead of combining them in one long cell. Optional SE, DEFF, CV, population totals, model effects, model confidence intervals, and model `p`-values are also separate columns. The default `result = "weighted"`, `rawn = TRUE` shows the actual sample `n` together with the survey-weighted estimate. For categorical variables the weighted statistic is a percentage with a design-based confidence interval. For `c.` variables the weighted mean and weighted population SD are shown, with a design-based CI for the mean. For `q.` variables the weighted median and weighted IQR are shown, with a median CI when available.

Full summaries. A variable declared with `f.` produces separate mean (SD), median (IQR), and range rows so weighted and unweighted summaries can be compared without compressing incompatible statistics into one number.

Tests. For categorical predictor by categorical outcome, weighted inference uses `survey::svychisq()` and defaults to the second-order Rao-Scott F correction. Weighted continuous comparisons use design-based `t`/Wald tests for mean-oriented variables and `survey::svyranktest()` for median/rank-oriented variables.

Regression estimates. OR uses survey-weighted logistic regression. PR and RR use a log-link survey-weighted quasi-Poisson model. A continuous `by = c.outcome` or `by = q.outcome` automatically reports unstandardized beta coefficients; the `q.` prefix changes the descriptive/group test but beta remains a linear-regression coefficient, consistent with R4VN `tab()` conventions.

Reference categories. Categorical references follow `vars()` prefixes. For example `b2.sex` makes the second observed/displayed level the model reference. The same requested reference is used in weighted and unweighted models.

Domain analysis. Use `subpop=` instead of physically deleting observations and rebuilding a complex design. The **survey** domain/subset machinery keeps the design information needed for valid variance estimation.

Population totals. `population = TRUE` is intentionally blocked unless `weightscale = "population"`. Weighted percentages, means, tests and regressions remain valid with normalized/relative survey weights, but their sum must not automatically be interpreted as the represented population.

Continuous outcomes. When `by` is continuous, predictor descriptions remain available and association tests/effect columns concern the continuous outcome. Categorical predictors are compared with `t`/ANOVA or rank tests as appropriate; numeric predictors are assessed by the slope test. The effect is an unstandardized beta coefficient with a confidence interval.

Replicate-weight designs. Replicate weights may be defined in `surveyset()` or directly in `tabsurvey()`. All statistics are then delegated to the corresponding **survey** replicate-design methods.

Reporting profiles. `report = "auto"` is the recommended default: it keeps the main table compact and weighted, automatically includes design-based tests when a `by` variable is present, and shows supporting design/test/effect tables in the Viewer. `"brief"` is deliberately descriptive. `"full"` adds the unweighted comparison plus SE, DEFF, and CV and stacks weighted/unweighted results by rows to avoid excessively wide tables. `"custom"` preserves the older option-by-option behavior. Interpretation is never automatic; set `interpretation = TRUE`.

Value

Invisibly returns an object of classes `r4vn_tabsurvey`, `r4vn_tab`, and `list`. Important components include:

- `data`: flat publication-ready table, compatible with `tabexport()`;
- `html`, `table_html`, and `file`: rendered table;
- `design`: the R4VN survey design metadata;
- `survey_design`: the underlying **survey** design used after any domain restriction;
- `metadata`: resolved R4VN variable specifications;
- `tests`: long-form machine-friendly test results;
- `effects`: long-form machine-friendly OR/PR/RR/beta results;
- `notes`: table footnotes;
- `tables`: named end-user report tables including Main, Design, Tests, Effects, Precision, and Interpretation when available;
- `diagnostics`: survey-design and precision diagnostics;
- `models`: fitted survey/unweighted regression models used for reported effects;
- `interpretation`: optional deterministic interpretation table;
- `subpop`: domain expression, when used.

Recommended reporting

For a publication or survey report, describe the sampling design and source of the final analytic weight, identify strata and PSU variables, state any domain/subpopulation restriction, and report the actual sample `n` together with survey-weighted estimates and design-based confidence intervals. When a hypothesis test is reported, the survey-adjusted test should normally be treated as the inferential result for a complex probability sample.

When `result = "both"`, the unweighted analysis is useful for data checking, transparency, and showing how weighting/design affects the result; it does not replace the design-based inference.

Common mistakes avoided by R4VN

- Do not interpret the sum of normalized/relative weights as a population size. Use `weightscale = "population"` only when the survey documentation supports an expansion-weight interpretation.

- Do not create a survey domain by deleting all observations outside the target subgroup and rebuilding the design. Prefer `subpop = . . .`.
- Do not assume one weight is correct for every variable in a public survey. When different analytic components require different weights, create multiple named designs with `surveyset()`.
- Do not silently treat propensity-score IPTW, frequency weights, or analytic regression weights as sampling/design weights.

References

Lumley T. Complex Surveys: A Guide to Analysis Using R. Wiley; 2010.

Lumley T. Analysis of complex survey samples. Journal of Statistical Software. 2004;9(1):1-19.

See Also

[surveyset](#), [tab](#), [vars](#), [tabexport](#)

Other R4VN survey: [surveyset\(\)](#)

Other R4VN tables: [tab\(\)](#), [tabexport\(\)](#), [tabforest\(\)](#), [tablong\(\)](#), [tabmeta\(\)](#), [tabmulti\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [vars\(\)](#)

Examples

```
# Reproducible complex-survey data used throughout the examples.
set.seed(2026)
d <- expand.grid(
  person = 1:2, household = 1:5, psu = 1:6, strata = 1:4,
  KEEP.OUT.ATTRS = FALSE
)
n <- nrow(d)
d$sex <- factor(sample(c("Female", "Male"), n, TRUE),
  levels = c("Female", "Male"))
d$age <- pmin(85, pmax(18, round(rnorm(n, 46, 14))))
d$bmi <- round(rnorm(n, 23.5, 3.4), 1)
d$income <- round(exp(rnorm(n, log(8), .5)), 1)
d$smoking <- factor(sample(c("No", "Yes"), n, TRUE, c(.72, .28)),
  levels = c("No", "Yes"))
d$education <- factor(
  sample(c("Primary", "Secondary", "College+"), n, TRUE),
  levels = c("Primary", "Secondary", "College+")
)
d$wt <- exp(.15 * (d$sex == "Male") + rnorm(n, 0, .3))
d$labwt <- d$wt * exp(rnorm(n, 0, .12))
d$popwt <- d$wt * 5000
d$fpc1 <- 30
d$fpc2 <- 100
lp <- -5 + .055 * d$age + .08 * (d$bmi - 23) +
  .45 * (d$sex == "Male") + .55 * (d$smoking == "Yes")
d$hypertension <- factor(rbinom(n, 1, plogis(lp)),
  levels = 0:1, labels = c("No", "Yes"))
d$sbp <- 82 + .72 * d$age + .85 * d$bmi +
  5 * (d$sex == "Male") + rnorm(n, 0, 13)
```

```

# Declare the survey design once; later tabsurvey() calls can stay short.
usedf(d)
surveyset(weight = wt, strata = strata, cluster = psu, nest = TRUE)

# 1. Simplest weighted publication table. report="auto" is the default.
s1 <- tabsurvey(vars = vars(c.age, sex, c.bmi, smoking), show = FALSE)
s1$tables$Main
s1$tables$Design

# 2. R4VN continuous prefixes: c.=mean, q.=median, f.=full summary.
s2 <- tabsurvey(vars = vars(c.age, q.income, f.bmi, sex), show = FALSE)

# 3. Table by a binary outcome; design-based tests are automatic.
s3 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi, smoking, education),
  by = hypertension, show = FALSE
)
s3$tables$Tests

# 4. Compare complete unweighted and weighted analyses side by side.
s4 <- tabsurvey(
  vars = vars(c.age, sex, q.income, c.bmi, smoking),
  by = hypertension, result = "both", bothstyle = "columns",
  show = FALSE
)

# 5. Full profile: both analyses stacked by rows plus SE, DEFF, and CV.
s5 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi, smoking),
  by = hypertension, report = "full", show = FALSE
)
s5$tables$Precision

# 6. Brief profile: weighted descriptive summary only unless overridden.
s6 <- tabsurvey(
  vars = vars(c.age, sex, q.income, c.bmi),
  report = "brief", show = FALSE
)

# 7. Row or cell percentages instead of the default column percentages.
s7_row <- tabsurvey(
  vars = vars(sex, smoking, education), by = hypertension,
  row = TRUE, col = FALSE, cell = FALSE, show = FALSE
)
s7_cell <- tabsurvey(
  vars = vars(sex, smoking, education), by = hypertension,
  row = FALSE, col = FALSE, cell = TRUE, show = FALSE
)

# 8. Crude survey-weighted odds ratios in the same publication table.
s8 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking, education),

```

```

    by = hypertension, or = TRUE, event = "Yes", show = FALSE
  )
  s8$tables$Effects

# 9. Separately adjusted OR for every focal predictor.
s9 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
  by = hypertension, or = TRUE, event = "Yes",
  adjusted = vars(c.age, b2.sex), show = FALSE
)

# 10. One common multivariable model containing all requested predictors.
s10 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking, education),
  by = hypertension, or = TRUE, event = "Yes",
  multi = TRUE, show = FALSE
)
s10$models

# 11. Prevalence ratio via survey-weighted modified Poisson regression.
s11 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
  by = hypertension, pr = TRUE, event = "Yes",
  multi = TRUE, show = FALSE
)

# 12. Explicit named reference levels; b2./b3. are also supported.
s12 <- tabsurvey(
  vars = vars(sex, smoking, education, c.age),
  by = hypertension, or = TRUE, event = "Yes",
  effect_ref = list(sex = "Male", smoking = "Yes",
                    education = "Secondary"),
  show = FALSE
)

# 13. Continuous outcome: unstandardized beta is reported automatically.
s13 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
  by = c.sbp, multi = TRUE, result = "both", show = FALSE
)

# 14. q. continuous outcome requests rank-oriented group tests; effect is beta.
s14 <- tabsurvey(
  vars = vars(b2.sex, b2.smoking, education),
  by = q.sbp, result = "both", show = FALSE
)

# 15. Correct domain/subpopulation analysis; do not rebuild a reduced design.
s15 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi, smoking),
  subpop = age >= 60 & sex == "Female", show = FALSE
)
s15$diagnostics$domain

```

```

# 16. Missing rows can be shown if present, always, or never.
d$smoking[1:4] <- NA
surveyset(d, name = "missing_demo", weight = wt, strata = strata,
          cluster = psu)
s16 <- tabsurvey(
  vars = vars(smoking, sex), design = "missing_demo",
  missing = "ifany", show = FALSE
)

# 17. Confidence level is fully dynamic, including the displayed CI label.
s17 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi), by = hypertension,
  or = TRUE, event = "Yes", level = .90, show = FALSE
)
names(s17$data) # contains "90% CI"

# 18. Request SE, design effect, and CV explicitly in a custom report.
s18 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi, smoking),
  report = "custom", se = TRUE, deff = TRUE, cv = TRUE,
  show = FALSE
)
s18$tables$Precision

# 19. Population totals require declared expansion/population weights.
surveyset(d, name = "population", weight = popwt, strata = strata,
          cluster = psu, weightscale = "population", active = FALSE)
s19 <- tabsurvey(
  vars = vars(sex, education, hypertension), design = "population",
  population = TRUE, show = FALSE
)

# 20. One-off design: no prior surveyset() call is required.
s20 <- tabsurvey(
  d, vars = vars(c.age, sex, c.bmi, hypertension),
  weight = wt, strata = strata, cluster = psu, show = FALSE
)

# 21. Multiple named weight systems can coexist.
surveyset(d, name = "laboratory", weight = labwt,
          strata = strata, cluster = psu, active = FALSE)
s21 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi), design = "laboratory", show = FALSE
)

# 22. Multistage clusters and finite-population corrections.
surveyset(
  d, name = "multistage", weight = wt, strata = strata,
  cluster = vars(psu, household), fpc = vars(fpc1, fpc2),
  nest = TRUE, active = FALSE
)
s22 <- tabsurvey(

```

```

    vars = vars(c.age, sex, c.bmi), design = "multistage", show = FALSE
  )

# 23. Compact legacy cells and display-order controls.
s23 <- tabsurvey(
  vars = vars(sex, smoking), by = hypertension,
  statcols = "compact", rvrow = TRUE, rvcol = TRUE,
  report = "custom", show = FALSE
)

# 24. Interpretation is opt-in and remains separate from statistical output.
s24 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
  by = hypertension, pr = TRUE, event = "Yes", multi = TRUE,
  report = "full", interpretation = TRUE, show = FALSE
)
s24$tables$Interpretation

# 25. Consistent result contract for custom reporting and downstream code.
names(s24$tables)
s24$descriptive
s24$tests
s24$effects
s24$diagnostics
s24$models
s24$interpretation

# 26. tabsurvey objects inherit from r4vn_tab and export with tabexport().
h <- tabexport(
  s3, s8, s11, s24, export = "html",
  file = tempfile("survey_report_"), quiet = TRUE
)
unlink(h$files)

# Additional syntax catalogue. These examples are intentionally not run by
# automatic checks, but are kept in ?tabsurvey for copy/paste use.

# 27. Risk ratio using the same modified-Poisson engine.
s27 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
  by = hypertension, rr = TRUE, event = "Yes", multi = TRUE
)

# 28. Alternative CI methods for proportions and weighted quantiles.
s28_prop <- tabsurvey(
  vars = vars(sex, smoking, hypertension), cimethod = "beta"
)
s28_quantile <- tabsurvey(
  vars = vars(q.income, q.bmi), quantile_method = "beta"
)

```

```

# 29. Choose another design-adjusted categorical test.
s29 <- tabsurvey(
  vars = vars(sex, smoking, education), by = hypertension,
  survey_test = "Wald"
)

# 30. Display controls: no CI, no raw n, two decimals, Overall last.
s30 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi, smoking), by = hypertension,
  ci = FALSE, rawn = FALSE, digit = 2, overall = "last"
)

# 31. Variable-name and HTML presentation controls.
s31_raw <- tabsurvey(
  vars = vars(c.age, sex, c.bmi), raw = TRUE,
  template = "clean", title = "Raw variable names"
)
s31_name <- tabsurvey(
  vars = vars(c.age, sex, c.bmi), name = TRUE,
  template = "minimal", title = "Labels plus names"
)

# 32. Inference/model-only table with descriptive cells suppressed.
s32 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi, b2.smoking),
  by = hypertension, or = TRUE, event = "Yes", multi = TRUE,
  descriptive = FALSE, report = "custom"
)

# 33. Explicitly suppress tests, coefficient p-values, and test notes.
s33 <- tabsurvey(
  vars = vars(c.age, b2.sex, c.bmi), by = hypertension,
  or = TRUE, event = "Yes", test = FALSE, pvalue = FALSE,
  test_note = FALSE, bold_p = FALSE, report = "custom"
)

# 34. Append two R4VN survey tables into one HTML page.
a34 <- tabsurvey(vars = vars(c.age, sex), show = FALSE)
f34 <- tempfile(fileext = ".html")
b34 <- tabsurvey(
  vars = vars(c.bmi, smoking), append = a34,
  file = f34, show = FALSE
)
unlink(f34)

# 35. A survey-package replicate design can be passed directly.
base35 <- survey::svydesign(
  ids = ~psu, strata = ~strata, weights = ~wt, data = d, nest = TRUE
)
rep35 <- survey::as.svrepdesign(base35, type = "bootstrap", replicates = 40)
s35 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi, hypertension),
  design = rep35, report = "full"
)

```

```

)

# 36. Override the lonely-PSU rule for one analysis only.
s36 <- tabsurvey(
  vars = vars(c.age, sex, c.bmi), lonely = "average"
)

```

tabts

Comprehensive time-series analysis with publication-ready output

Description

tabts() provides a single R4VN-style interface for descriptive time-series analysis, decomposition, stationarity assessment, ACF/PACF, ARIMA/SARIMA/ ARIMAX, ETS, model comparison, validation, forecasting, interrupted time series (ITS), controlled ITS, count ITS, residual diagnostics, flexible graphics, and evidence-linked interpretation.

Usage

```

tabts(
  outcome,
  time,
  data = NULL,
  model = "auto",
  period = "auto",
  order = NULL,
  seasonal = NULL,
  xreg = NULL,
  group = NULL,
  intervention = NULL,
  population = NULL,
  rate = 1e+05,
  family = c("auto", "gaussian", "poisson", "quasipoisson", "negativebinomial"),
  correlation = c("auto", "none", "ar1", "nw"),
  decompose = c("auto", "stl", "classical", "none"),
  stationarity = TRUE,
  acf = TRUE,
  pacf = TRUE,
  diagnostic = TRUE,
  forecast = 0,
  level = c(0.8, 0.95),
  future_xreg = NULL,
  test = NULL,
  criterion = c("aicc", "rmse", "mae", "mape"),
  missing_time = c("warn", "error", "NA", "zero", "interpolate"),
  duplicate_time = c("error", "mean", "sum", "first"),

```

```

plot = TRUE,
plots = "auto",
theme = c("r4vn", "minimal", "classic", "bw"),
title = NULL,
subtitle = NULL,
xlab = NULL,
ylab = NULL,
legend = TRUE,
legend_position = "bottom",
observed_color = NULL,
fitted_color = NULL,
forecast_color = NULL,
counterfactual_color = NULL,
intervention_color = NULL,
linewidth = 0.8,
point = TRUE,
point_size = 1.8,
pi = TRUE,
pi_alpha = 0.15,
width = 8,
height = 5,
dpi = 300,
interpret = FALSE,
language = "en",
detail = c("full", "brief"),
show = TRUE,
console = FALSE,
digits = 2,
p_digits = 3,
model_options = list(),
diagnostic_options = list(),
forecast_options = list(),
its_options = list(),
table_options = list(),
interpret_options = list(),
plot_options = list(),
ai = FALSE,
...
)

```

Arguments

outcome	Outcome variable. A bare column name or a single character column name. The outcome must be numeric.
time	Time variable. A bare column name or a single character column name. Date, POSIXct, integer/numeric index, and ordered time values are supported. Date-like variables are recommended.
data	Data frame. If NULL, tabts() attempts to use the active R4VN data frame.

model	Analysis engine. One of "auto", "arima", "ets", "compare", "its", or "regression". A vector such as <code>c("arima", "ets")</code> is treated as a candidate-model comparison.
period	Seasonal period. Use "auto" to infer common periods from the time variable, or supply a positive integer such as 12 for monthly annual seasonality, 4 for quarterly data, or 52 for weekly data.
order	ARIMA order <code>c(p,d,q)</code> . If NULL, <code>forecast::auto.arima()</code> is used when forecast is installed; otherwise <code>tabts()</code> performs a compact AICc-based search with <code>stats::arima()</code> so the default command still works.
seasonal	Seasonal ARIMA order <code>c(P,D,Q)</code> . The seasonal period comes from period. Ignored for non-ARIMA engines.
xreg	Optional external regressors. Accepts <code>vars(x1, x2)</code> , <code>c("x1", "x2")</code> , or a single bare column name.
group	Optional grouping variable. For ordinary time-series models, separate models are fitted for each group. For ITS, a group variable produces a controlled ITS when <code>its_options\$controlled = TRUE</code> (the default when a group is supplied).
intervention	Intervention time(s) for ITS. May be a Date/POSIXct, a value comparable with time, a vector of intervention times, or the name/bare name of a 0/1 intervention indicator column.
population	Optional population/exposure variable for count ITS. When supplied with a count family, <code>log(population)</code> is used as an offset.
rate	Display rate multiplier used for descriptive rate calculations, e.g. 100000. It does not change the count-model offset.
family	ITS family: "auto", "gaussian", "poisson", "quasipoisson", or "negativebinomial".
correlation	ITS residual correlation handling: "auto", "none", "ar1", or "nw". <code>AR(1)</code> via <code>nlme::gls()</code> is available for Gaussian ITS. Newey-West robust covariance requires sandwich.
decompose	Decomposition: "auto", "stl", "classical", or "none". "auto" uses STL when at least two seasonal cycles are present.
stationarity	Logical; run stationarity assessment when possible. ADF and KPSS require <code>tseries</code> ; suggested differencing additionally uses forecast when available.
acf, pacf	Logical; calculate ACF/PACF tables and plots.
diagnostic	Logical; calculate residual diagnostics.
forecast	Number of future periods to forecast. Zero disables forecasting. Forecast intervals are prediction intervals for ARIMA/ETS.
level	Forecast interval levels, e.g. <code>c(.80, .95)</code> .
future_xreg	Optional data frame/matrix of future xreg values for ARIMAX or time-series regression forecasts. It must contain at least forecast rows and the same xreg variables used for fitting.
test	Optional holdout size for validation. An integer means the number of final observations; a value between 0 and 1 means that proportion of observations.
criterion	Model-selection criterion: "aicc", "rmse", "mae", or "mape". Out-of-sample criteria require test.

missing_time	Handling of missing time points: "warn", "error", "NA", "zero", or "interpolate". Missing time points are never silently converted to zero.
duplicate_time	Handling of duplicate time values within a series: "error", "mean", "sum", or "first".
plot	Logical; create plots. Publication-ready base-R plots are always available; if ggplot2 is installed, tabts() uses ggplot2 automatically.
plots	Character vector selecting plots. "auto" creates the relevant set; "all" is an explicit synonym and "none" suppresses plot creation. Other values include "series", "decomposition", "acf", "pacf", "residual", "forecast", "its", and "counterfactual".
theme	Plot theme: "r4vn", "minimal", "classic", or "bw".
title, subtitle, xlab, ylab	Common plot labels.
legend	Logical; show legends where relevant.
legend_position	Legend position such as "bottom", "top", "left", "right", or "none".
observed_color, fitted_color, forecast_color, counterfactual_color	Optional common layer colors. Leave NULL to use R4VN defaults.
intervention_color	Optional intervention-layer color. Leave NULL to use the R4VN default.
linewidth	Default line width for main series layers.
point	Logical; display observed points on the main series plot.
point_size	Default observed point size.
pi	Logical; display prediction-interval ribbons on forecast plots.
pi_alpha	Prediction-interval ribbon transparency.
width, height, dpi	Default figure width/height in inches and raster resolution used when plot(result, file = ...) saves a figure.
interpret	Logical or one of "brief"/"full". Generate deterministic rule-based interpretation linked to exact numerical evidence from the output. The default is FALSE, keeping the routine report concise; request TRUE, "brief", or "full" when needed.
language	Output language. Currently only "en" is supported; all tables, plots, diagnostics, warnings, and interpretations are produced in English.
detail	Interpretation detail: "full" or "brief".
show	Logical; show a publication-style HTML report. In RStudio the report opens in the Viewer and contains all tables and every generated plot; the primary plot is also sent to the Plots pane. No HTML package is required for this Viewer report.
console	Logical; also print tables to the console.
digits	Number of digits for estimates.
p_digits	Number of digits for p-values.

<code>model_options</code>	Named list of advanced model controls. Important entries include <code>auto_arima</code> , <code>ets</code> , <code>diagnostic_gate</code> , <code>candidate</code> , <code>include_drift</code> , and <code>include_mean</code> . For fixed ARIMA/SARIMA models, <code>tabts()</code> automatically disables a drift term when $d + D \neq 1$ and disables a mean term when differencing is present.
<code>diagnostic_options</code>	Named list controlling diagnostics. Important entries include <code>ljung_lag</code> , <code>normality</code> , <code>acf_lag</code> , and <code>alpha</code> .
<code>forecast_options</code>	Named list controlling forecasts. Entries may include <code>bootstrap</code> , <code>biasadj</code> , and <code>history</code> .
<code>its_options</code>	Named list controlling ITS. Entries include <code>controlled</code> , <code>reference_group</code> , <code>time_scale</code> , <code>effect_at</code> , <code>counterfactual</code> , <code>include_season</code> , <code>seasonal_harmonics</code> , <code>nw_lag</code> , and <code>post_min</code> . Seasonal adjustment uses sine/cosine Fourier pairs; <code>seasonal_harmonics</code> controls how many pairs are included. For controlled ITS, the first factor level is the reference unless <code>reference_group</code> is supplied.
<code>table_options</code>	Named list controlling output tables. Entries include <code>ci_level</code> , <code>show_model_selection</code> , <code>show_stationarity</code> , <code>show_diagnostics</code> , and <code>show_interpretation</code> .
<code>interpret_options</code>	Named list controlling interpretation, including <code>alpha</code> , <code>include_assumptions</code> , <code>include_model</code> , <code>include_forecast</code> , <code>include_limitations</code> , and <code>evidence</code> .
<code>plot_options</code>	Deep list controlling plots. This is the main extension point for colors, line types/widths, point shapes/sizes, interval ribbons, axes, date breaks/labels, limits, legends, fonts, grids, reference lines, annotations, intervention/counterfactual layers, facets, and component plots. See Details and examples.
<code>ai</code>	FALSE, TRUE, or a named R4VN AI endpoint. AI is an optional additional layer and is used only when <code>aiask()</code> is available. For a deterministic evidence-linked interpretation, set <code>interpret = TRUE</code> .
<code>...</code>	Reserved for future compatible extensions.

Details

The function is intentionally simple for routine use:

```
tabts(cases, time = month)
```

while advanced behavior can be changed through `model_options`, `diagnostic_options`, `forecast_options`, `its_options`, `table_options`, `interpret_options`, and `plot_options`. This design keeps the public API stable while allowing future extensions.

Core workflow:

`tabts()` follows the R4VN workflow:

1. validate and regularize the time index;
2. describe the series;
3. assess seasonality/stationarity;
4. fit candidate models;
5. validate/select the model;
6. diagnose residuals;

7. forecast when requested;
8. produce publication-ready tables and plots;
9. generate evidence-linked interpretation at the end.

Optional packages and dependency-light defaults:

Routine ARIMA/SARIMA, forecasting from fixed/base-selected ARIMA models, regression, decomposition, ACF/PACF, Gaussian ITS, Poisson/quasi-Poisson ITS, Viewer tables, and Viewer figures can run without extra analysis or reporting packages. Optional packages add specialized methods: forecast for Hyndman-Khandakar auto-ARIMA and ETS; tseries for ADF/KPSS; nlme for Gaussian AR(1) ITS; sandwich for Newey-West covariance; MASS for negative-binomial ITS; and ggplot2 for editable ggplot objects.

Flexible plot options:

Advanced graphics are changed with nested `plot_options`. For example:

```
plot_options = list(
  observed = list(color = "black", linewidth = .8,
                  linetype = "solid", point = TRUE,
                  point_shape = 16, point_size = 2),
  fitted = list(color = "steelblue", linewidth = 1,
                linetype = "dashed"),
  forecast = list(color = "firebrick", linewidth = 1.1),
  pi = list(show = TRUE, alpha = .15, border = FALSE),
  axis = list(
    xlim = NULL, ylim = NULL,
    date_breaks = "6 months", date_labels = "%b %Y",
    y_breaks = NULL, y_log = FALSE
  ),
  legend = list(position = "bottom", title = NULL),
  grid = list(major = TRUE, minor = FALSE),
  intervention = list(line = TRUE, label = TRUE,
                     linetype = "dashed", linewidth = .8),
  counterfactual = list(show = TRUE, linetype = "dotted"),
  reference = list(xline = NULL, yline = NULL),
  facet = list(show = TRUE, ncol = NULL, scales = "fixed"),
  font = list(family = NULL, base_size = 11),
  annotation = NULL
)
```

When `ggplot2` is installed, plots are returned as ordinary `ggplot` objects and may be edited with standard `ggplot2` syntax. Without `ggplot2`, `tabts()` returns lightweight `r4vn_tabts_plot` objects drawn with base R; this keeps the default analysis and Viewer graphics dependency-light.

Interpretation:

Every rule-based interpretation row contains a finding, the exact evidence used to create it, a status, and a source. Thus statements about stationarity, model selection, residual adequacy, ITS effects, or forecast uncertainty can always be traced back to a specific result.

Value

An object of class `r4vn_tabts`. Important components include `data`, `summary`, `stationarity`, `decomposition`, `acf`, `pacf`, `model`, `models`, `model_info`, `model_selection`, `coefficients`, `diagnostics`, `validation`, `forecast`, `counterfactual`, `its`, `interpretation`, `tables`, `plots`, and `metadata`. The `its` component stores the family, correlation structure, intervention timing, and pre/post counts used by ITS. Grouped analyses return class `r4vn_tabts_grouped`.

Examples

```
data(dengue_ts)

# 1. Simplest command. Automatic ARIMA works with base R; when forecast is
# installed, ARIMA/ETS comparison becomes available automatically.
m1 <- tabts(cases, time = month, data = dengue_ts)
m1$tables
m1$plots$series

# 2. Forecast six future months with 80% and 95% prediction intervals.
m2 <- tabts(cases, time = month, data = dengue_ts, forecast = 6)
m2$forecast
plot(m2, "forecast")

# 3. Fixed ARIMA using base R only.
m3 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1), forecast = 6)
m3$coefficients
m3$diagnostics

# 4. Seasonal ARIMA/SARIMA.
m4 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 1, 1),
            seasonal = c(0, 1, 1), period = 12, forecast = 12)

# 5. ARIMAX with external regressors.
future_weather <- tail(dengue_ts[c("rainfall", "temperature")], 6)
m5 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1),
            xreg = vars(rainfall, temperature), forecast = 6,
            future_xreg = future_weather)

# 6. Time-series regression with trend and seasonal terms; base R only.
m6 <- tabts(cases, time = month, data = dengue_ts,
            model = "regression", forecast = 6)

# 7. Compare ARIMA and ETS when forecast is installed.
if (requireNamespace("forecast", quietly = TRUE)) {
  m7 <- tabts(cases, time = month, data = dengue_ts,
              model = c("arima", "ets"), test = 12,
              criterion = "rmse", forecast = 12)
  m7$model_selection
  m7$validation
}
```

```

# 8. Decomposition, ACF and PACF are available in the result.
m8 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1))
m8$decomposition
m8$acf
m8$pacf
plot(m8, "decomposition")
plot(m8, "acf")
plot(m8, "pacf")

# 9. Select only the plots needed in a report.
m9 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1), forecast = 6,
            plots = c("series", "forecast", "residual"))

# 10. Missing time points are never silently converted to zero.
dmiss <- dengue_ts[-20, ]
m10 <- tabts(cases, time = month, data = dmiss,
            missing_time = "interpolate", model = "arima",
            order = c(1, 0, 1), forecast = 6)

# 11. Duplicate time points can be handled explicitly.
ddup <- rbind(dengue_ts, dengue_ts[1, ])
m11 <- tabts(cases, time = month, data = ddup,
            duplicate_time = "mean", model = "arima",
            order = c(1, 0, 1))

# 12. Gaussian interrupted time series using base R.
m12 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "gaussian", correlation = "none")
m12$coefficients
m12$effect_at
plot(m12, "counterfactual")

# 13. Poisson ITS with a population offset; also base R.
m13 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "poisson", population = population,
            rate = 100000, correlation = "none")

# 14. Request effects at clinically meaningful post-intervention times.
m14 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "poisson", population = population,
            correlation = "none",
            its_options = list(effect_at = c(1, 3, 6, 12, 24)))
m14$effect_at

# 15. Intervention may be supplied as a 0/1 indicator column.
dind <- dengue_ts
dind$policy <- as.integer(dind$month >= as.Date("2023-01-01"))

```

```

m15 <- tabts(cases, time = month, data = dind, model = "its",
             intervention = policy, family = "poisson",
             population = population, correlation = "none")

# 16. Controlled ITS.
data(dengue_its_control)
m16 <- tabts(cases, time = month, group = group,
             data = dengue_its_control, model = "its",
             intervention = as.Date("2023-01-01"), family = "poisson",
             population = population, correlation = "none",
             its_options = list(reference_group = "Control"))
m16$coefficients
m16$effect_at

# 17. Fit separate ordinary time-series models by group.
m17 <- tabts(cases, time = month, group = group,
             data = dengue_its_control, model = "arima",
             order = c(1, 0, 1), forecast = 3)
names(m17$results)

# 18. Interpretation is OFF by default.
m18 <- tabts(cases, time = month, data = dengue_ts,
             model = "arima", order = c(1, 0, 1))
m18$interpretation

# 19. Turn on evidence-linked interpretation when desired.
m19 <- tabts(cases, time = month, data = dengue_ts,
             model = "arima", order = c(1, 0, 1),
             forecast = 6, interpret = TRUE)
m19$interpretation

# 20. Brief interpretation.
m20 <- tabts(cases, time = month, data = dengue_ts,
             model = "arima", order = c(1, 0, 1),
             interpret = "brief")

# 21. Publication-ready plot customization.
m21 <- tabts(cases, time = month, data = dengue_ts,
             model = "arima", order = c(1, 0, 1), forecast = 12,
             title = "Monthly dengue cases",
             subtitle = "Observed, fitted and forecast values",
             xlab = "Month", ylab = "Cases",
             plot_options = list(
               observed = list(point = TRUE, point_size = 1.6),
               forecast = list(linewidth = 1),
               pi = list(show = TRUE, alpha = .12),
               axis = list(date_breaks = "1 year", date_labels = "%Y"),
               legend = list(position = "bottom")
             ))

# 22. The Viewer report contains every generated table and plot when show=TRUE.
m22 <- tabts(cases, time = month, data = dengue_ts,
             model = "arima", order = c(1, 0, 1),

```

```

        forecast = 6, show = TRUE)

# 23. Save a publication figure without another export package.
plot(m22, "forecast",
     file = file.path(tempdir(), "tabts_forecast_300dpi.png"),
     width = 8, height = 5, dpi = 300)

# 24. Use the active R4VN data frame.
usedf(dengue_ts, quiet = TRUE)
m24 <- tabts(cases, time = month, model = "arima",
            order = c(1, 0, 1), forecast = 3)
usedf(clear = TRUE, quiet = TRUE)

# 25. Optional enhancements only when needed:
# tseries -> ADF/KPSS; nlme -> Gaussian AR(1) ITS;
# sandwich -> Newey-West ITS; MASS -> negative-binomial ITS;
# forecast -> auto.arima/ETS; ggplot2 -> editable ggplot objects.

# 26. Explicit ETS when forecast is available.
if (requireNamespace("forecast", quietly = TRUE)) {
  m26 <- tabts(cases, time = month, data = dengue_ts,
              model = "ets", forecast = 6)
  m26$model_info
  m26$forecast
}

# 27. ADF/KPSS stationarity tests are added when tseries is installed.
m27 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1),
            stationarity = TRUE)
m27$stationarity

# 28. Hold out the final 20% of observations for validation.
m28 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1), test = .20)
m28$validation

# 29. Keep tables but suppress plot creation completely.
m29 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1),
            plot = FALSE, show = TRUE)
m29$tables

# 30. Request all relevant figures explicitly.
m30 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1),
            forecast = 6, plots = "all")
names(m30$plots)

# 31. Gaussian ITS: correlation = "auto" uses AR(1) only when nlme is
# available and the correlated model improves AIC sufficiently.
m31 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),

```

```

        family = "gaussian", correlation = "auto")
m31$its

# 32. Newey-West covariance is an optional enhancement via sandwich.
m32 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "poisson", population = population,
            correlation = "nw")
m32$coefficients

# 33. Automatic count-family choice: Poisson, quasi-Poisson, or
# negative-binomial when MASS is available and overdispersion is marked.
m33 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "auto", population = population,
            correlation = "none")
m33$its$family

# 34. Explicit quasi-Poisson ITS requires no additional package.
m34 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "quasipoisson", population = population,
            correlation = "none")
m34$coefficients

# 35. Multiple intervention dates in one segmented model.
m35 <- tabts(cases, time = month, data = dengue_ts,
            model = "its",
            intervention = as.Date(c("2022-01-01", "2023-01-01")),
            family = "poisson", population = population,
            correlation = "none")
m35$coefficients
m35$its

# 36. Add dependency-light Fourier seasonal terms to ITS when required.
m36 <- tabts(cases, time = month, data = dengue_ts,
            model = "its", intervention = as.Date("2023-01-01"),
            family = "poisson", population = population,
            correlation = "none", period = 12,
            its_options = list(include_season = TRUE, seasonal_harmonics = 2))
m36$coefficients

# 37. Customize which tables are shown in the Viewer without deleting the
# underlying result components.
m37 <- tabts(cases, time = month, data = dengue_ts,
            model = "arima", order = c(1, 0, 1),
            table_options = list(show_stationarity = FALSE,
                                show_validation = FALSE))
names(m37$tables)
m37$stationarity

# 38. If ggplot2 is installed, edit a returned plot as an ordinary ggplot.
if (requireNamespace("ggplot2", quietly = TRUE)) {

```

```

p38 <- m22$plots$forecast + ggplot2::labs(caption = "R4VN tabts")
print(p38)
}

# 39. Save TIFF or vector PDF directly through plot().
plot(m22, "forecast",
      file = file.path(tempdir(), "tabts_forecast.tiff"),
      width = 8, height = 5, dpi = 300)
plot(m22, "forecast",
      file = file.path(tempdir(), "tabts_forecast.pdf"),
      width = 8, height = 5)

# 40. Inspect reusable components for a custom manuscript/report workflow.
names(m22)
names(m22$tables)
names(m22$plots)
m22$metadata
summary(m22)

```

ttesti

Student and Welch t Tests

Description

`ttesti()` calculates one- or two-sample t tests from summary statistics. `ttest()` performs the same analysis from variables and reports group sample size, mean, standard error, standard deviation, and confidence interval. `ttest(..., effect = TRUE)` additionally reports Cohen's d and Hedges' g. With `by = vars(province, sex, treatment)`, province and sex are nested strata and treatment is the innermost two-group comparison.

Usage

```

ttesti(
  n1,
  mean1,
  sd1,
  n2 = NULL,
  mean2 = NULL,
  sd2 = NULL,
  mu = 0,
  equal = FALSE,
  paired = FALSE,
  r = NULL,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,

```

```

    show = TRUE,
    console = FALSE
  )

  ttesti(
    n1, mean1, sd1, n2 = NULL, mean2 = NULL, sd2 = NULL, mu = 0, equal = FALSE,
    paired = FALSE, r = NULL, alternative = c("two.sided", "less", "greater"),
    level = 0.95, digits = 3, p_digits = 3, show = TRUE, console = FALSE
  )

  ttest(
    x, y = NULL, by = NULL, data = NULL, mu = 0, equal = FALSE, paired = FALSE,
    alternative = c("two.sided", "less", "greater"), level = 0.95,
    effect = FALSE, digits = 3, p_digits = 3, show = TRUE, console = FALSE
  )

  ttest(
    x,
    y = NULL,
    by = NULL,
    data = NULL,
    mu = 0,
    equal = FALSE,
    paired = FALSE,
    alternative = c("two.sided", "less", "greater"),
    level = 0.95,
    effect = FALSE,
    digits = 3,
    p_digits = 3,
    show = TRUE,
    console = FALSE
  )

```

Arguments

n1, mean1, sd1	Sample size, mean, and standard deviation for group 1.
n2, mean2, sd2	Optional sample size, mean, and standard deviation for group 2.
mu	Null mean or null mean difference.
equal	Use the equal-variance two-sample t test.
paired	Perform a paired analysis.
r	Correlation between paired measurements when only summaries are available.
alternative	Alternative hypothesis: "two.sided", "less", or "greater".
level	Confidence level as a proportion.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.

console	Logical; also print the traditional text result in the Console. Default FALSE.
x, y	Numeric variables. y is optional.
by	Optional two-level grouping variable. <code>by = vars(a, b, group)</code> performs the test within nested <code>a > b</code> strata using <code>group</code> as the innermost comparison.
data	Data frame. If NULL, the active data set is used.
effect	Logical; for <code>ttest()</code> , add standardized effect sizes.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
ttesti(40, 12, 3, mu = 10)
ttesti(40, 12, 3, 35, 14, 4)

d <- data.frame(score = c(10,12,11,18,17,20,9,13,12,19,21,18),
                 treatment = rep(c("Control","Intervention"), each = 6),
                 province = rep(c("A","B"), each = 3, times = 2))
ttest(score, by = treatment, data = d, effect = TRUE)
ttest(score, by = vars(province, treatment), data = d, effect = TRUE)
```

usedf	<i>Set or inspect the active data frame</i>
-------	---

Description

Makes a data frame the default data source for R4VN commands. When a bare object name is supplied, such as `usedf(patient)`, the active data remains linked to that object. R4VN editing commands such as `genvar()`, `replacevar()`, `labvar()`, `renvar()`, `dropvar()`, `keepvar()`, and `ordervar()` then update the object itself as well as the active data.

Usage

```
usedf(data, clear = FALSE, quiet = FALSE)
```

Arguments

data	Optional data frame to make active.
clear	Logical; clear active data.
quiet	Logical; suppress status messages.

Details

If an expression rather than a bare object name is supplied, R4VN stores an unlinked working copy. Use `newdata <- usedf()` to retrieve that copy.

Value

The active data frame invisibly, or NULL after clearing.

Examples

```
patient <- data.frame(id = 1:3, age = c(20, 30, 40))
usedf(patient, quiet = TRUE)

# Editing active data also updates patient
genvar(age2 = age^2)
names(patient)

# Manual changes to patient are seen by later active-data commands
patient$age[1] <- 21
usedf(quiet = TRUE)$age

usedf(clear = TRUE, quiet = TRUE)

# Extended usage examples

patient <- data.frame(id = 1:4, age = c(20, 30, 40, 50))

# Set a visible object as active data. The object remains in Environment.
usedf(patient)
usedf()

# R4VN editing commands update both active data and patient.
genvar(age10 = age / 10, label = "Age in decades")
labvar(age, label = "Age in years")
names(patient)
if (interactive()) View(patient)

# Manual changes to patient are visible to later active-data commands.
patient$age[1] <- 21
sum1(age)

# An expression creates an unlinked working copy.
usedf(subset(patient, age >= 30))
genvar(age2 = age^2)
working_copy <- usedf()

# Return to the linked object or clear active data.
usedf(patient)
usedf(clear = TRUE)
```

Description

Compares the nine Tukey ladder transformations using histograms with fitted normal curves and reports n, mean, SD, median, range, skewness, kurtosis and Shapiro-Wilk diagnostics.

Usage

```
varform(x = NULL,
  vars = NULL,
  data = NULL,
  by = NULL,
  shift = c("none",
    "auto"),
  bins = "Sturges",
  color = NULL,
  palette = "default",
  alpha = 0.8,
  normal_color = "black",
  normal_lty = 1,
  normal_lwd = 2,
  theme = "journal",
  size = 10,
  file = NULL,
  width = 9,
  height = 8,
  dpi = 300,
  show = TRUE,
  bg = "white",
  digits = 3,
  p_digits = 3)
```

Arguments

x	One numeric variable; optional when vars is supplied.
vars	Optional selection of several numeric variables.
data	Data frame or active R4VN data.
by	Optional grouping specification. All selected grouping combinations receive their own transformation ladder.
shift	Whether to leave nonpositive data unchanged or add the minimum positive shift required for log/reciprocal transformations.
bins	Histogram break specification.
color, palette, alpha	Histogram appearance.
normal_color, normal_lty, normal_lwd	Normal-curve appearance.
theme, size	Graph theme and text size.

file, width, height, dpi, show, bg
Graph export/display controls. Multiple outputs receive numbered file names.

digits, p_digits
Formatting digits.

Details

The transformation ladder contains cubic, square, identity, square-root, logarithm, inverse square-root, inverse, inverse-square and inverse-cubic transformations. Logarithmic and reciprocal transformations require positive values unless `shift = "auto"`.

Value

An R4VN result object, invisibly.

See Also

[ghist](#), [normtest](#)

Examples

```
d <- data.frame(weight = c(29,26,13,23,23,25,17,22,17,19,12,26,30,30,18,14,12,26,17,18))
varform(weight, data = d)
varform(vars = vars(weight), data = d, shift = "auto")
```

vars	<i>Specify Variables for R4VN Tables</i>
------	--

Description

Captures variable specifications without evaluating them immediately. Prefixes determine how variables are summarized and which observed categorical level is used as the model reference category. The `i.` prefix is accepted as an explicit categorical declaration so the same syntax can be reused in regression and survival commands.

Usage

```
vars(...)
```

Arguments

... One or more unquoted variable specifications or selectors. Examples include `sex`, `i.sex`, `b2.age_group`, `c.age`, `q.weight`, `f.gestational_age`, `..`kt*``, ``*score``, ``*kt*``, and `vars(., -id)`.

Details

The function also supports deferred selectors: `.` for all variables, wildcard selectors using `*`, and exclusions using unary `-`. Deferred selectors are expanded only after the calling analysis function knows which data frame is being used.

Supported prefixes are:

- no prefix: automatic typing from the data. Numeric/integer variables use mean and standard deviation; factor/character/logical variables are categorical using their first observed level as reference;
- `i.`: force categorical treatment and use the first observed level as reference. This is equivalent to `b1.` and makes `vars(i.sex)` consistent with regression-model syntax;
- `b1.`, `b2.`, `b3.`, ...: force categorical treatment and use the first, second, third, or corresponding observed level as reference;
- `c.`: numeric variable summarized by mean and standard deviation;
- `q.`: numeric variable summarized by median and interquartile range;
- `f.`: numeric variable summarized by mean, median, and range.

Selector syntax:

- `vars(.):` select all variables intentionally;
- `vars(`kt*`):` names beginning with `kt`;
- `vars(`*kt`):` names ending with `kt`;
- `vars(`*kt*`):` names containing `kt`;
- `vars(., -id):` all variables except `id`;
- `vars(`kt*`, -kt_total):` wildcard selection except `kt_total`;
- `vars(., -`id*`):` all variables except names beginning with `id`.

Because `*` is an R operator, wildcard specifications must be written inside backticks. Thus use `vars(`kt*`)`, not `vars(kt*)`. A selector consisting only of asterisks is deliberately rejected; use `vars(.)` when all variables are intended.

Prefixes can be combined with wildcard selectors, for example `vars(`c.lab*`)`, `vars(`q.score*`)`, or `vars(`b2.item*`)`.

Exact specifications are more specific than wildcard specifications, and wildcard specifications are more specific than `..`. Therefore an exact specification can override a broader selector. For example, `vars(`c.lab*`, q.lab_crp)` declares all `lab*` variables as mean/SD except `lab_crp`, which is median/IQR. When two selectors have the same specificity, the later one wins. Exclusions are applied last and always win.

Unprefixed variables and deferred selectors such as `.` and ``kt*`` are stored with type `"default"` until they are resolved against a data frame. With `default_type = "auto"` in `.r4vn_resolve_vars()`, factor/character/logical columns become categorical and numeric/integer columns become mean/SD variables. Use an explicit `b1.`, `b2.`, ... prefix when a numeric-coded variable should be treated as categorical instead.

Prefixes are declaration syntax only. For example, `c.age` refers to the `age` column; the data do not need a column named `c.age`. For factors, observed-level order follows `levels()`. Set factor levels before calling `tab()` or `tabmulti()` when exact ordering or reference categories are important.

`vars()` with no arguments remains an error by design. This avoids accidentally selecting every variable.

Value

A data frame of class `r4vn_vars` with columns `variable`, `type`, `specification`, and `reference_index`. Deferred selectors are expanded by `.r4vn_resolve_vars()` inside R4VN analysis functions.

See Also

[tab](#) and [tabmulti](#).

Other R4VN tables: [tab\(\)](#), [tabexport\(\)](#), [tabforest\(\)](#), [tablong\(\)](#), [tabmeta\(\)](#), [tabmulti\(\)](#), [tabscale\(\)](#), [tabscore\(\)](#), [tabsurvey\(\)](#)

Examples

```
# Existing declaration syntax.
specification <- vars(i.sex, b3.education, c.age, q.bmi, f.sbp)
specification
# i.sex is an explicit categorical declaration with the first level as reference.
vars(i.sex)

# Deferred selectors are captured by vars() and resolved by public
# R4VN analysis functions once a data frame is supplied.
vars(.)
vars(`kt*`)
vars(`*score`)
vars(`*kt*`)
vars(., -id)
vars(`c.lab*`, q.lab_crp)

dat <- data.frame(
  id = 1:5,
  age = c(31, 42, 38, 50, 46),
  sex = factor(c("F", "M", "F", "M", "F")),
  kt1 = 1:5,
  kt2 = 6:10,
  kt_total = 11:15,
  score_kt = 16:20
)

# Unprefixed variables are typed automatically from the actual data:
# age is numeric -> mean (SD); sex is a factor -> categorical.
t_auto <- tab(dat, vars = vars(age, sex), show = FALSE)

# Select all variables.
t_all <- tab(dat, vars = vars(.), show = FALSE)

# Prefix wildcard.
t_kt <- tab(dat, vars = vars(`kt*`), show = FALSE)

# Select all except id.
```

```

t_no_id <- tab(dat, vars = vars(., -id), show = FALSE)

# Typed wildcard with an exact override.
t_typed <- tab(
  dat,
  vars = vars(`c.kt*`, q.kt_total),
  show = FALSE
)

t_kt$data

```

ztesti

z Tests with Known Standard Deviations

Description

Performs one- or two-sample z tests when population standard deviations are known.

Usage

```

ztesti(
  n1,
  mean1,
  sd1,
  n2 = NULL,
  mean2 = NULL,
  sd2 = NULL,
  mu = 0,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,
  show = TRUE,
  console = FALSE
)

ztest(
  x,
  sigma1,
  y = NULL,
  sigma2 = NULL,
  by = NULL,
  data = NULL,
  mu = 0,
  alternative = c("two.sided", "less", "greater"),
  level = 0.95,
  digits = 3,
  p_digits = 3,

```

```
    show = TRUE,  
    console = FALSE  
  )
```

Arguments

n1, mean1, sd1	Summary statistics for sample 1.
n2, mean2, sd2	Optional summary statistics for sample 2.
mu	Null mean or mean difference.
alternative	Alternative hypothesis.
level	Confidence level.
digits, p_digits	Decimal places for estimates and p-values.
show	Logical; open the formatted result in the Viewer. Default TRUE.
console	Logical; also print the traditional text result in the Console. Default FALSE.
x, y	Numeric variables.
sigma1, sigma2	Known population standard deviations.
by	Optional two-level grouping variable.
data	Data frame. If NULL, active data is used.

Value

Invisibly returns an object of class `r4vn_stat`.

Examples

```
ztesti(100, 52, 10, mu = 50)
```

Index

- * **R4VN datasets**
 - tabforest_demo, 169
- * **R4VN survey**
 - surveyset, 132
 - tabsurvey, 248
- * **R4VN tables**
 - tab, 135
 - tabexport, 157
 - tabforest, 160
 - tablong, 184
 - tabmeta, 211
 - tabmulti, 220
 - tabscale, 225
 - tabscore, 231
 - tabsurvey, 248
 - vars, 277
- * **datasets**
 - dengue_its_control, 17
 - dengue_ts, 18
 - tabforest_demo, 169
- * **prediction models**
 - tabscore, 231
- * **scorecards**
 - tabscore, 231
- aiask, 4
- aisetup, 6
- anova (anovai), 8
- anova_r4vn (anovai), 8
- anovai, 8
- appenddata, 10
- as.data.frame.r4vn_machine, 11
- as.data.frame.r4vn_meta, 12
- at, 12, 71
- bitest (prop), 108
- bitesti (prop), 108
- cc (epi), 27
- cci (epi), 27
- ci (cii), 13
- ci_r4vn (cii), 13
- cii, 13
- corr, 14
- cox, 16
- cs (epi), 27
- csi (epi), 27
- dengue_its_control, 17
- dengue_ts, 18
- describe, 18
- design, 20
- dict, 21
- distdata, 22
- distdata(), 24, 88
- distlearn, 24
- distlearn(), 22
- dropvar, 25
- egenvar, 26
- epi, 27
- epii (epi), 27
- epitool, 30
- esize (esize), 31
- esize, 31
- friedman, 32
- gbar, 34
- gbox, 37
- gdensity, 39
- genvar, 41
- genvar(), 26
- gforest, 42
- ghist, 45, 277
- gline, 47
- gpie, 49
- groc, 51
- gscatter, 53
- gsurv, 55

- ir, 58
- iri (ir), 58
- keepvar, 59
- kwallis, 60, 80
- labvar, 62
- lincom, 64, 71
- logistic, 66
- logistic(), 69, 70, 99
- lrtest, 69
- lrtest(), 67, 68, 99, 118
- margins, 13, 65, 70, 73, 77
- marginsplot, 71, 71
- mcc, 73
- mcci (mcc), 73
- mergedata, 75
- nlregress, 76, 112
- normtest, 78, 135, 277
- nptrend, 79
- opendata, 80
- ordervar, 85
- plot.r4vn_diag, 86
- plot.r4vn_distdata, 88
- plot.r4vn_graph, 89
- plot.r4vn_graph_set (plot.r4vn_graph), 89
- plot.r4vn_machine, 89
- plot.r4vn_meta, 90
- plot.r4vn_tabforest, 94
- plot.r4vn_tablong, 95
- plot.r4vn_tabscale, 95
- plot.r4vn_tabscore, 96
- plot.r4vn_tabts, 97
- poisson, 98
- poisson(), 68–70
- predict, 65, 71, 100
- predict.r4vn_machine, 103
- predict.r4vn_tabscore, 104
- print.r4vn_graph, 104
- print.r4vn_graph_set, 105
- print.r4vn_machine, 105
- print.r4vn_meta, 106
- print.r4vn_quick, 106
- print.r4vn_stat, 107
- print.r4vn_tabscale, 107
- prop (propi), 108
- propi, 108
- proportion_tests (propi), 108
- prtest (propi), 108
- prtesti (propi), 108
- qregress, 77, 111
- r4fun, 113
- R4VN_examples, 114
- ranksum, 115
- regress, 112, 117
- renvar, 119
- replacevar, 120
- savedata, 121
- sdtest (sdtesti), 123
- sdtesti, 123
- shapevar, 124
- shiny::runApp(), 20, 24, 113
- signrank, 126
- stats::anova(), 8, 9
- stats::fisher.test(), 171
- stats::p.adjust(), 9, 61
- stats::poisson(), 98
- sum1, 127
- summary.r4vn_survey, 129
- summary.r4vn_tabsurvey, 130
- surv_tables, 130
- survexport, 131
- surveyset, 132, 255
- surveyset(), 255
- swilk, 79, 134
- tab, 135, 158, 222, 255, 279
- tab(), 158, 167, 189, 215, 222, 228, 239, 255, 279
- tab1, 142
- tabconti (tab), 135
- tabdiag, 144
- tabdiagi, 155
- tabexport, 140, 157, 222, 255
- tabexport(), 140, 167, 189, 215, 222, 228, 239, 255, 279
- tabforest, 160
- tabforest(), 140, 158, 169, 189, 215, 222, 228, 239, 255, 279
- tabforest_demo, 169
- tabi, 170

tablearn, 172
tablong, 184
tablong(), 140, 158, 167, 215, 222, 228, 239,
255, 279
tabmachine, 195
tabmeta, 211
tabmeta(), 140, 158, 167, 189, 222, 228, 239,
255, 279
tabmulti, 140, 158, 220, 279
tabmulti(), 140, 158, 167, 189, 215, 228,
239, 255, 279
tabscale, 225
tabscale(), 140, 158, 167, 189, 215, 222,
239, 255, 279
tabscore, 231
tabscore(), 140, 158, 167, 189, 215, 222,
228, 255, 279
tabsurv, 243
tabsurvey, 134, 248
tabsurvey(), 134, 140, 158, 167, 189, 215,
222, 228, 239, 279
tabts, 261
ttest (ttesti), 272
ttesti, 272

usedf, 134, 274
usedf(), 26, 27

varform, 79, 135, 275
vars, 134, 140, 222, 255, 277
vars(), 140, 158, 167, 189, 215, 222, 228,
239, 255

ztest (ztesti), 280
ztesti, 280