Package: tidyvisor (via r-universe)

November 27, 2026

Title A Statistical Advisor for Research: Design, Analysis and Interpretation

Version 0.2.0

Description A chief statistician for R. Before the data are collected, 'tidyvisor' advises on study design, the unit of replication, sample size (power analysis) and good research practice, with the reporting guideline for the field. Given data and a question, it audits the data, checks the assumptions of the candidate analyses and recommends the analysis, the test and the plots, with the code to run and what to do when assumptions fail. Given a fitted result, it explains the numbers in plain sentences, with its reasons, so the advice can be checked and overruled. Specialised knowledge lives in the packages that do the analyses, each of which registers an advisor: 'tidydiversity' for community ecology, 'tidymapr' for spatial ecology and spatial epidemiology, and 'tidyepidemio' for epidemiology.

License MIT + file LICENSE

URL https://tidyvisor.in4metrix.dev/, https://github.com/jmozez/tidyvisor

BugReports https://github.com/jmozez/tidyvisor/issues

Depends R (>= 4.1.0)

Imports methods, cli (>= 3.6.0), rlang (>= 1.1.0), stats, utils, tibble (>= 3.2.0)

Suggests brms, posterior, aods3, glmmTMB, lme4, mgcv, MASS, knitr, rmarkdown, testthat (>= 3.0.0), tidydiversity (>= 0.3.0), tidymapr (>= 0.3.0), tidyepidemio, vegan, withr

Remotes jmozez/tidydiversity, jmozez/tidymapr, jmozez/tidyepidemio

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-NZ

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

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

Date/Publication 2026-11-27 06:29:22 UTC

RemoteUrl https://github.com/jmozez/tidyvisor

RemoteRef HEAD

RemoteSha 8ebf9de067b6d3bbcc062bd1593c20b7a2e7862e

Contents

advise . . . 2
advisors . . . 3
brms-advisor . . . 4
compare-advisor . . . 5
interpret . . . 6
model-advisors . . . 7
power-advisor . . . 9
practice-advisor . . . 12
register_advisor . . . 14
regression-advisor . . . 15
tidyvisor . . . 17
tv_check_model . . . 18
tv_equivalence . . . 19
tv_evalue . . . 20
tv_save . . . 21
vegan-advisor . . . 22

Index **25**

`advise` *Advice on which analysis to run*

Description

`advise()` finds the advisor for a data frame and a question, and returns its advice: what the data show, the analysis to run and why, the test, the plots, and the code.

Usage

```
advise(x, question = "auto", ...)
```

Arguments

x	A data frame.
question	What is asked, or "auto". advisors() lists the questions of each advisor.
...	Passed to the advisor.

Details

The advisor is chosen by question. With question = "auto" (the default), each installed advisor is asked whether the data and the arguments look like its kind: community data given with abundance or a formula belong to tidydiversity, records with coordinates to tidymapr, and case counts, exposures and outcomes to tidyepidemio. When more than one advisor claims the data, give question.

Value

The advisor's advice.

Examples

```
cases <- data.frame(
  week = seq(as.Date("2024-01-01"), by = "week", length.out = 12),
  n = c(2, 3, 5, 4, 9, 12, 15, 24, 30, 41, 60, 77)
)
advise(cases, question = "epidemic", date = "week", cases = "n")
```

---

advisors *List the available advisors*

Description

List the available advisors

Usage

```
advisors()
```

Value

A tibble with one row per advisor: package, title, whether the package is installed, its questions and the result classes it can interpret.

Examples

```
advisors()
```

`brms-advisor` *Advisor for brms: Bayesian regression and mixed models*

Description

Built-in advisor for `brms::brm()`.

- **Before** (`question = "bayesian"`): chooses the family from the kind of response (including `trials()` for successes out of trials, and zero-inflated families for counts with excess zeros), adds random intercepts for the grouping columns in `random`, and gives the code with weakly informative priors, a prior predictive check and a posterior predictive check.
- **After**: explains each population-level effect on its natural scale with its credible interval and the posterior probability of its direction; the standard deviation of each group-level effect; and the sampling diagnostics (R-hat, effective sample sizes, divergent transitions) that decide whether the posterior can be trusted at all.

Usage

```
tv_advise_brms(
  x,
  question = "bayesian",
  response,
  predictors = NULL,
  random = NULL,
  trials = NULL,
  ...
)

tv_interpret_brms(x, level = 0.95, ...)
```

Arguments

`x`	A data frame (advice) or a `brmsfit` (interpretation).
`question`	The question; set by `advise()`.
`response`, `predictors`, `random`, `trials`	As for `tv_advise_mixed()`; `random` is optional.
`...`	Ignored.
`level`	Width of the credible intervals.

Value

A `tv_advice` or a `tv_interpretation` object.

References

Bürkner, P.-C. (2017). brms: an R package for Bayesian multilevel models using Stan. *Journal of Statistical Software* 80(1), 1-28.

Gabry, J. et al. (2019). Visualization in Bayesian workflow. *Journal of the Royal Statistical Society A* 182, 389-402.

Vehtari, A. et al. (2021). Rank-normalization, folding, and localization: an improved R-hat for assessing convergence of MCMC. *Bayesian Analysis* 16, 667-718.

Examples

```
tidyvisor(mtcars, question = "bayesian", response = "am", predictors = "wt")
```

`compare-advisor` *Advisor for comparing groups: assumptions and what to do when they fail*

Description

Built-in advisor for comparing a numeric response between groups: the t-test, one-way ANOVA and their alternatives in base R.

- **Before** (`question = "compare"`): checks the assumptions of the classical tests in each group (normality, equal spread, outliers, skewness, whether the spread grows with the mean) and recommends the test those checks support, with the code, and the alternatives:

What the checks show	Recommendation
roughly normal, equal spread	Student t-test or one-way ANOVA
roughly normal, unequal spread	Welch t-test or Welch ANOVA (`oneway.test()`)
positive, right-skewed, spread grows with the mean	log-transform, then Welch (compares geometric means)
outliers or heavy tails	robust test on trimmed means (Yuen; `WRS2`)
counts with unequal spread	Poisson or negative binomial GLM
otherwise non-normal, or very small groups	rank test: Wilcoxon or Kruskal-Wallis, or a permutation test

  With `paired`, the checks are on the within-pair differences and the tests are their paired versions (paired t, Wilcoxon signed-rank).

- **After**: explains any classical test result (`htest`: from `stats::t.test()`, `stats::wilcox.test()`, `stats::kruskal.test()`, `stats::oneway.test()`, `stats::shapiro.test()`, `stats::cor.test()`, `stats::chisq.test()` and others): what was tested, the statistic, the p-value, the estimate and its interval, what the test does not show, and what to do next.

Assumption tests answer "can the data rule out the assumption?", not "does the violation matter?": with large groups they flag trivial departures, with small groups they miss large ones. The advice therefore weighs them together with the skewness, the ratio of spreads and the group sizes, and always says to look at the plot.

Usage

```
tv_advise_compare(x, question = "compare", response, group, paired = NULL, ...)

tv_interpret_htest(x, ...)
```

Arguments

x	A data frame (advice) or an htest object (interpretation).
question	The question; set by advise().
response	Name of the numeric response column.
group	Name of the grouping column (two or more groups).
paired	Optional name of a column identifying the pairs (or blocks) for repeated measurements on the same units; two groups only.
...	Ignored.

Value

A tv_advice or a tv_interpretation object.

References

Delacre, M., Lakens, D. and Leys, C. (2017). Why psychologists should by default use Welch's t-test instead of Student's t-test. *International Review of Social Psychology* 30, 92-101.

Fagerland, M. W. (2012). t-tests, non-parametric tests, and large studies: a paradox of statistical practice? *BMC Medical Research Methodology* 12, 78.

O'Hara, R. B. and Kotze, D. J. (2010). Do not log-transform count data. *Methods in Ecology and Evolution* 1, 118-122.

Wilcox, R. R. (2017). *Introduction to Robust Estimation and Hypothesis Testing*, fourth edition. Academic Press.

Examples

```
tidyvisor(InsectSprays, response = "count", group = "spray")
tidyvisor(kruskal.test(count ~ spray, data = InsectSprays))
```

---

interpret *Plain-language interpretation of a fitted result*

Description

interpret() finds the advisor that knows the class of x and returns its interpretation: the numbers in sentences, what the tests say, and how to read the figure.

Usage

```
interpret(x, ...)
```

Arguments

x	A fitted result, such as an ordination from `tidydiversity::ord_fit()` or a growth rate from `tidymapr::epi_growth_rate()`.
...	Passed to the advisor (for example `permutations`).

Value

The advisor's interpretation.

Examples

```
cases <- data.frame(
  week = seq(as.Date("2024-01-01"), by = "week", length.out = 12),
  n = c(2, 3, 5, 4, 9, 12, 15, 24, 30, 41, 60, 77)
)
g <- tidymapr::epi_growth_rate(cases, date = week, cases = n, unit = "week")
interpret(g)
```

---

model-advisors *Advisors for mixed, additive and overdispersed models*

Description

Built-in advisors for four widely used modelling packages. Each gives advice before fitting (from a data frame) and a plain-language interpretation after it (from the fitted model).

- **lme4** (`question = "mixed"`, or a `random` argument): linear and generalised linear mixed models, `lme4::lmer()`, `lme4::glmer()` and `lme4::glmer.nb()`. The advice checks the grouping factors (number of levels, observations per level) and chooses the family. The interpretation reports fixed effects on their natural scale, the variance of each random effect, the intra-class correlation, and singular or non-converged fits.
- **glmmTMB** (`question = "zero_inflation"`): mixed models for overdispersed and zero-inflated counts with `glmmTMB::glmmTMB()`; the interpretation adds the zero-inflation part and the dispersion.
- **mgcv** (`question = "smooth"`, or a `smooth` argument): generalised additive models with `mgcv::gam()`. The interpretation explains each smooth by its effective degrees of freedom, checks the basis dimension (k) and concurvity.
- **aods3** (`question = "overdispersion"`): beta-binomial and negative binomial models with `aods3::aodml()` for proportions and counts more variable than the binomial or Poisson allows.

Usage

```
tv_advise_mixed(
  x,
  question = "mixed",
  response,
  predictors = NULL,
  random,
  trials = NULL,
  engine = c("lme4", "glmmTMB"),
  ...
)

tv_advise_gam(
  x,
  question = "smooth",
  response,
  predictors = NULL,
  smooth = NULL,
  random = NULL,
  trials = NULL,
  ...
)

tv_advise_aod(
  x,
  question = "overdispersion",
  response,
  predictors = NULL,
  trials = NULL,
  ...
)

tv_interpret_mixed(x, level = 0.95, ...)

tv_interpret_gam(x, level = 0.95, ...)

tv_interpret_aod(x, level = 0.95, ...)
```

Arguments

x	A data frame (advice) or a fitted model (interpretation).
question	The question; set by advise().
response	Name of the response column.
predictors	Names of the fixed-effect predictor columns (default: all other columns except random, smooth and trials).
random	Names of the grouping columns for random intercepts.
trials	Name of the column with the number of trials, for successes out of trials.

engine	Package to fit the mixed model with, "lme4" or "glmmTMB".
...	Ignored.
smooth	Names of the numeric predictors to model with smooths.
level	Confidence level for the intervals.

Value

A tv_advice or a tv_interpretation object.

References

Bolker, B.M. et al. (2009). Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution* 24, 127-135.

Brooks, M.E. et al. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal* 9, 378-400.

Harrison, X.A. et al. (2018). A brief introduction to mixed effects modelling and multi-model inference in ecology. *PeerJ* 6, e4794.

Nakagawa, S. & Schielzeth, H. (2013). A general and simple method for obtaining R2 from generalized linear mixed-effects models. *Methods in Ecology and Evolution* 4, 133-142.

Wood, S.N. (2017). *Generalized Additive Models: An Introduction with R*, 2nd ed. Chapman & Hall/CRC.

Lesnoff, M. & Lancelot, R. (2012). aods3: analysis of overdispersed data using S3 methods. R package.

Examples

```
tidyvisor(lme4::sleepstudy, response = "Reaction", predictors = "Days",
  random = "Subject")
m <- lme4::lmer(Reaction ~ Days + (1 | Subject), data = lme4::sleepstudy)
tidyvisor(m)
```

---

power-advisor *Power analysis: how many samples, sites or groups to collect*

Description

Advice before fieldwork: how large a study must be to detect the effect that matters, with a given power (the chance of detecting the effect if it is real) and significance level.

Give the design and the smallest effect worth detecting, from biology, earlier studies or a pilot:

design	Effect to give	Answer
"two_groups"	delta (difference in means), sd	samples per group (t-test)
"paired"	delta, sd of the within-pair differences	pairs
"groups"	means (expected group means), sd	samples per group (one-w

"proportions"	p1, p2	samples per group
"counts"	mean1 and ratio (rate ratio), theta (negative binomial size; Inf for Poisson)	samples per group
"regression"	r2 (partial R squared of the predictor(s) tested), predictors	observations
"correlation"	r	observations
"sites"	delta, sd, icc, per_site	sites per group, with samp

Each answer comes with a sensitivity table (the size needed for half and twice the effect, and for 90% power) because the effect is never known exactly, and with the code that reproduces the calculation.

With pilot data, `advise(pilot, question = "power", response = ..., group = ...)` estimates the standard deviation (and the intra-class correlation when `site` is given) from the pilot and asks only for the effect. Use the smallest effect of biological interest, not the pilot's observed difference: pilot effects are noisy, and powering a study on an overestimate leaves it too small.

Usage

```
tv_power(
 design = c("two_groups", "paired", "groups", "proportions", "counts", "regression",
    "correlation", "sites"),
  delta = NULL,
  sd = NULL,
  means = NULL,
  p1 = NULL,
  p2 = NULL,
  mean1 = NULL,
  ratio = NULL,
  theta = Inf,
  r2 = NULL,
  predictors = 1,
  r = NULL,
  icc = NULL,
  per_site = NULL,
  power = 0.8,
  alpha = 0.05
)

tv_advise_power(
  x,
  question = "power",
  response,
  group = NULL,
  site = NULL,
  ...
)
```

Arguments

design	The design (see the table).

`delta`, `sd`	Difference in means worth detecting, and the standard deviation within groups (in the same units).
`means`	Expected means of three or more groups.
`p1`, `p2`	Expected proportions in two groups.
`mean1`, `ratio`, `theta`	Expected mean count in the reference group, rate ratio worth detecting, and negative binomial size (dispersion; smaller is more overdispersed; `Inf` for Poisson).
`r2`, `predictors`	Partial R squared of the tested predictor(s) and the total number of predictors.
`r`	Correlation worth detecting.
`icc`, `per_site`	Intra-class correlation (share of the variance between sites) and the number of samples per site.
`power`	Target power (default 0.8).
`alpha`	Significance level (default 0.05).
`x`	Pilot data (a data frame).
`question`	Set by `advise()`.
`response`, `group`, `site`	Columns of the pilot data.
`...`	Passed to `tv_power()` (the effect: `delta`, `ratio`, ...).

Value

A `tv_advice` object: `checks` (inputs and sensitivity) and `advice` (the size needed and how to collect it), and `n`, the required size.

References

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences*, second edition. Lawrence Erlbaum.

Green, P. and MacLeod, C. J. (2016). SIMR: an R package for power analysis of generalized linear mixed models by simulation. *Methods in Ecology and Evolution* 7, 493-498.

Johnson, P. C. D., Barry, S. J. E., Ferguson, H. M. and Mueller, P. (2015). Power analysis for generalized linear mixed models in ecology and evolution. *Methods in Ecology and Evolution* 6, 133-142.

Zhu, H. and Lakkis, H. (2014). Sample size calculation for comparing two negative binomial rates. *Statistics in Medicine* 33, 376-387.

Examples

```
# Detect a 5 cm difference in tree height (SD 8 cm) between two habitats
tv_power("two_groups", delta = 5, sd = 8)

# Beetle counts: mean 4 per trap, detect a 1.5-fold difference
tv_power("counts", mean1 = 4, ratio = 1.5, theta = 2)

# Plots nested in sites: how many sites per treatment?
```

```
tv_power("sites", delta = 5, sd = 8, icc = 0.2, per_site = 5)

# From a pilot: the SD comes from the data, the effect from biology
advise(PlantGrowth, question = "power", response = "weight",
       group = "group", delta = 0.5)
```

`practice-advisor` *Research practice and study design: advice for any field*

Description

The general advice of tidyvisor, for any field and any stage of a study.

- `tv_practice()` gives a checklist of good research practice for a stage of the study (`"plan"`, `"collect"`, `"analyse"`, `"report"` or `"all"`) and a kind of study (`"experiment"`, `"observational"`, `"survey"`, `"monitoring"`, `"review"`), with the reporting guideline for the field where one exists (`field`).
- `advise(data, question = "design", ...)` audits a data set before the analysis: missing values, duplicated rows, constant and nearly empty columns, the unit of replication (pseudoreplication), treatments confounded with sites or time, unbalanced groups, and suggests the analysis for the response. It is also what `advise()` falls back on when no specialised advisor recognises the data, so `tidyvisor()` always answers.

Reporting guidelines by `field`:

`field`	Guideline
`"trial"`	CONSORT 2010 (randomised trials)
`"epidemiology"`, `"medicine"`	STROBE (observational studies)
`"animal"`	ARRIVE 2.0 (animal research)
`"review"`	PRISMA 2020 (systematic reviews and meta-analyses)
`"diagnostic"`	STARD 2015 (diagnostic accuracy)
`"sdm"`	ODMAP (species distribution models)
`"ecology"`	Tools for Transparency in Ecology and Evolution (TTEE)

Usage

```
tv_practice(
  stage = c("all", "plan", "collect", "analyse", "report"),
 design = c("any", "experiment", "observational", "survey", "monitoring", "review"),
  field = NULL
)

tv_advise_design(
  x,
  question = "design",
  response = NULL,
  group = NULL,
```

```
  unit = NULL,
  time = NULL,
  ...
)
```

Arguments

`stage`	Stage of the study.
`design`	Kind of study.
`field`	Field, to name its reporting guideline (see the table); `NULL` for general advice.
`x`	A data frame.
`question`	Set by `advise()`.
`response`	Optional name of the response column.
`group`	Optional name of the treatment or grouping column.
`unit`	Optional name of the column identifying the unit the treatment was applied to (plot, site, cage, patient, school).
`time`	Optional name of a time column (date, year, visit).
`...`	Ignored.

Value

A `tv_advice` object.

References

Hurlbert, S. H. (1984). Pseudoreplication and the design of ecological field experiments. *Ecological Monographs* 54, 187-211.

Nosek, B. A., Ebersole, C. R., DeHaven, A. C. and Mellor, D. T. (2018). The preregistration revolution. *Proceedings of the National Academy of Sciences* 115, 2600-2606.

Wasserstein, R. L. and Lazar, N. A. (2016). The ASA statement on p-values: context, process, and purpose. *The American Statistician* 70, 129-133.

Wilkinson, M. D. et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3, 160018.

Zuur, A. F., Ieno, E. N. and Elphick, C. S. (2010). A protocol for data exploration to avoid common statistical problems. *Methods in Ecology and Evolution* 1, 3-14.

Examples

```
tv_practice("plan", design = "experiment")
tv_practice("report", field = "sdm")

# Audit a data set: here, chicks weighed repeatedly
advise(ChickWeight, question = "design", response = "weight",
       group = "Diet", unit = "Chick", time = "Time")
```

`register_advisor` *Register an advisor with tidyvisor*

Description

A package that wants `tidyvisor()` to advise on its analyses and interpret its results registers an *advisor*, usually from its `.onLoad()`:

```
.onLoad <- function(libname, pkgname) {
  if (requireNamespace("tidyvisor", quietly = TRUE)) {
    tidyvisor::register_advisor(
      package   = pkgname,
      questions = c("explore", "explain"),
      classes   = c("mypkg_fit"),
      advise    = function(x, ...) my_advise(x, ...),
      interpret = function(x, ...) my_interpret(x, ...),
      claims    = function(x, args) "abundance" %in% args
    )
  }
}
```

Registering again under the same `package` replaces the earlier advisor, so a package's own registration takes precedence over the built-in one.

Usage

```
register_advisor(
  package,
  questions,
  classes,
  advise,
  interpret,
  claims = function(x, args) FALSE,
  title = package
)
```

Arguments

`package`	Name of the package that provides the advisor.
`questions`	The questions it answers (used by `advise()` with `question = "..."`).
`classes`	Classes of the results it can interpret.
`advise, interpret`	Functions `function(x, ...)` returning the advice or the interpretation, or names of functions exported by `package`.

`claims` A function `function(x, args)` returning `TRUE` when a data frame `x`, given with arguments named `args`, is this advisor's kind of data (used by `advise()` with `question = "auto"`). It must not evaluate the arguments, which may be bare column names.

`title` A short description shown by `advisors()`.

Value

The registry entry, invisibly.

`regression-advisor` *The regression advisor: linear and generalised linear models*

Description

The built-in advisor for `stats::lm()` and `stats::glm()`. It is registered under the name `"stats"`, so `tidyvisor()` uses it for `question = "regression"` and for any fitted `lm` or `glm`.

Before fitting (`tidyvisor(data, question = "regression", response = "y")`), it looks at the response and the predictors and recommends a model:

- the **family and link** from the kind of response: continuous (Gaussian), counts (Poisson, or negative binomial when overdispersed), binary (binomial, logit), proportions (binomial with the number of trials), positive and right-skewed (Gamma, log);
- whether counts show **overdispersion** or **more zeros** than a Poisson distribution with the same mean would give (a rough first check, before the predictors are taken into account);
- whether there are enough observations: about 10 to 20 per estimated parameter (Harrell 2015), and for a binary response at least 10 events of the rarer outcome per parameter (Peduzzi et al. 1996);
- **collinear** numeric predictors ($|r| > 0.7$; Dormann et al. 2013) and missing values.

After fitting (`tidyvisor(model)`), it explains:

- the fit: R-squared for `lm`, the share of deviance explained for `glm`, and the test against the null model;
- every coefficient on its natural scale, with a 95% confidence interval: a change per unit (identity link), a multiplicative change (log link) or an odds ratio (logit link);
- diagnostics: overdispersion (Pearson statistic / residual df), non-constant variance (studentised Breusch-Pagan test), non-normal residuals (Shapiro-Wilk, for Gaussian models), influential observations (Cook's distance > 1; Cook and Weisberg 1982), collinearity (variance inflation factors) and separation in binomial models.

Confidence intervals are Wald intervals (estimate +/- critical value x standard error) on the link scale, transformed to the response scale; they can be optimistic in small samples, where profile intervals (`stats::confint()`) are better.

Usage

```
tv_advise_regression(
  x,
  question = "regression",
  response,
  predictors = NULL,
  trials = NULL,
  ...
)

tv_interpret_regression(x, level = 0.95, ...)
```

Arguments

x	A data frame (advice) or a fitted lm or glm (interpretation).
question	"regression".
response	Name of the response column.
predictors	Names of the predictor columns; default all other columns.
trials	For a response of successes out of a number of trials, the name of the column holding the number of trials.
...	Not used.
level	Confidence level of the intervals.

Value

tv_advise_regression() returns a tv_advice object and tv_interpret_regression() a tv_interpretation object; both print as sentences.

References

Cook, R. D. and Weisberg, S. (1982). Residuals and Influence in Regression. Chapman and Hall.

Dormann, C. F. et al. (2013). Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. Ecography, 36, 27-46.

Harrell, F. E. (2015). Regression Modeling Strategies, 2nd ed. Springer.

Peduzzi, P., Concato, J., Kemper, E., Holford, T. R. and Feinstein, A. R. (1996). A simulation study of the number of events per variable in logistic regression analysis. Journal of Clinical Epidemiology, 49, 1373-1379.

Zuur, A. F., Ieno, E. N. and Elphick, C. S. (2010). A protocol for data exploration to avoid common statistical problems. Methods in Ecology and Evolution, 1, 3-14.

Examples

```
# Before: which model for counts of warp breaks?
tidyvisor(warpbreaks, question = "regression", response = "breaks")

# After: what does the model show?
```

```
m <- glm(breaks ~ wool + tension, family = poisson, data = warpbreaks)
tidyvisor(m)
```

`tidyvisor` *tidyvisor: advice before the analysis, interpretation after it*

Description

`tidyvisor()` is a statistical advisor shared by a family of packages. Give it **data** and a question, and it recommends the analysis, checks the data for the usual pitfalls, and names the plots and the code to run (`advise()`). Give it a **result**, and it explains the numbers and the figure in plain sentences (`interpret()`).

The knowledge lives in the packages that do the analyses. Each one registers an *advisor* with tidyvisor (`register_advisor()`): the questions it answers, the results it can interpret, and how to recognise its kind of data. tidyvisor knows three advisors out of the box:

- **tidydiversity** (community ecology): which ordination, constrained or unconstrained, linear or unimodal, and which test; interpretation of ordinations, principal response curves and variation partitioning.
- **tidymapr** (spatial ecology): species distribution models, spread of invasions, the spread of epidemics across a landscape, range shifts and spatial autocorrelation.
- **tidyepidemio** (epidemiology): outbreaks (incidence, growth, Rt), associations (risk, odds and rate ratios), survival, diagnostic tests, surveillance and causal effects.

Each is used when its package is installed; `advisors()` lists them.

The advice always gives its reasons, so it can be checked and overruled: the question, which only the analyst knows, comes first.

Usage

```
tidyvisor(x, ...)
```

Arguments

`x`	A data frame (to be advised) or a fitted result (to be interpreted).
`...`	Passed to `advise()` or `interpret()`, and on to the advisor: for example `question`, the columns to use, or `abundance` and `formula` for community data.

Value

The advisor's advice or interpretation, which prints as sentences.

See Also

`advise()`, `interpret()`, `advisors()`, `register_advisor()`

Examples

```
frogs <- tidydiversity::png_frogs

# Before: which analysis?
tidyvisor(frogs, abundance = abundance)

# After: what does it show?
ord <- tidydiversity::ord_fit(frogs,
  abundance = abundance, method = "pca", transform = "hellinger"
)
tidyvisor(ord)
```

---

tv_check_model *Check a fitted model with simulated residuals*

Description

Checks whether a fitted model describes the data well, using scaled residuals from simulations of the fitted model (Dunn and Smyth 1996; Hartig 2024, the method of the DHARMa package). For each observation the observed value is placed within the values simulated from the model; if the model is right these scaled residuals are uniform between 0 and 1, whatever the distribution (Poisson, binomial, negative binomial, mixed models). Four tests follow:

- uniformity (Kolmogorov-Smirnov): the overall distribution is wrong;
- dispersion: the data vary more (or less) than the model expects;
- zero inflation: more zeros than the model expects (counts only);
- outliers: more observations outside all simulated values than chance.

It works for any model with `stats::simulate()` and `stats::fitted()` methods: `lm()`, `glm()`, `MASS::glm.nb()`, `lme4::glmer()` and `glmmTMB::glmmTMB()`.

Usage

```
tv_check_model(model, nsim = 250, seed = NULL)
```

Arguments

`model`	A fitted model.
`nsim`	Number of simulations (250 is usually enough).
`seed`	Optional random seed, set with `set.seed()` before simulating, for reproducible residuals.

Value

A `tv_advice` object (question `"model check"`) with `checks` (each test, its p-value and what it means) and `advice`, plus `residuals`, a tibble with `observed`, `fitted` and `scaled` (the scaled residuals), for plots such as a uniform QQ plot.

References

Dunn, P. K. and Smyth, G. K. (1996). Randomized quantile residuals. Journal of Computational and Graphical Statistics, 5, 236-244. doi:10.1080/10618600.1996.10474708

Hartig, F. (2024). DHARMa: residual diagnostics for hierarchical (multi-level / mixed) regression models. R package. https://CRAN.R-project.org/package=DHARMa

Examples

```
# Counts with extra variation: Poisson is overdispersed
fit <- glm(count ~ spray, family = poisson, data = InsectSprays)
tv_check_model(fit, seed = 1)
```

---

tv_equivalence *Equivalence test: is the difference small enough to ignore?*

Description

A non-significant test does not show that two groups are the same. An equivalence test does: with two one-sided tests (TOST; Schuirmann 1987, Lakens 2017) it tests whether the difference in means lies inside the bounds `low` and `high`, the smallest differences that would matter. Equivalence is declared when both one-sided tests are significant, which is the same as the 90% confidence interval (for `alpha = 0.05`) lying inside the bounds. Uses Welch's t-test, or the paired or one-sample t-test.

Usage

```
tv_equivalence(x, y = NULL, low, high, paired = FALSE, alpha = 0.05)
```

Arguments

`x, y`	Numeric samples (`y = NULL` for one sample or paired differences already taken).
`low, high`	Equivalence bounds for the difference (`x - y`), in the units of the data, from the smallest effect of interest.
`paired`	Paired samples?
`alpha`	Significance level of each one-sided test.

Value

A `tv_advice` object (question `"equivalence"`) with the difference, the confidence interval, the two one-sided tests and the ordinary test in `checks`, the conclusion in `advice`, and `result`, a one-row tibble with `estimate`, `lower`, `upper`, `p_lower`, `p_upper`, `p_tost` and `equivalent`.

References

Schuirmann, D. J. (1987). A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability. Journal of Pharmacokinetics and Biopharmaceutics, 15, 657-680. doi:10.1007/BF01068419

Lakens, D. (2017). Equivalence tests: a practical primer for t tests, correlations, and meta-analyses. Social Psychological and Personality Science, 8, 355-362. doi:10.1177/1948550617697177

Examples

```
# Is sleep gain under the two drugs within 0.5 hours?
tv_equivalence(sleep$extra[1:10], sleep$extra[11:20], low = -0.5, high = 0.5, paired = TRUE)
```

`tv_evalue` *E-value: how strong would unmeasured confounding have to be?*

Description

For an observational study, the E-value (VanderWeele and Ding 2017) is the minimum strength of association, on the risk ratio scale, that an unmeasured confounder would need with both the exposure and the outcome to fully explain away the observed association. A small E-value (near 1) means weak confounding could explain the result; a large one means only strong confounding could. The E-value for the confidence limit closest to 1 says how much confounding would make the interval include 1.

Odds ratios and hazard ratios for a common outcome (more than about 15%) are first converted to approximate risk ratios with the square-root transformation and that of VanderWeele (2020); a standardised mean difference `d` with $RR \approx \exp(0.91d)$.

Usage

```
tv_evalue(
  estimate,
  lower = NULL,
  upper = NULL,
  type = c("rr", "or", "hr", "smd"),
  rare = FALSE
)
```

Arguments

`estimate`	The point estimate (a ratio, or a standardised mean difference for `type = "smd"`).
`lower, upper`	Confidence limits (optional).
`type`	`"rr"` (risk or rate ratio), `"or"` (odds ratio), `"hr"` (hazard ratio) or `"smd"` (standardised mean difference).
`rare`	For odds and hazard ratios: is the outcome rare (under about 15%)? Then the ratio approximates a risk ratio and is used as is.

Value

A `tv_advice` object (question `"confounding"`) with the E-values in `checks` and an interpretation in `advice`, plus `evalue` (point) and `evalue_ci`.

References

VanderWeele, T. J. and Ding, P. (2017). Sensitivity analysis in observational research: introducing the E-value. Annals of Internal Medicine, 167, 268-274. doi:10.7326/M162607

VanderWeele, T. J. (2020). Optimal approximate conversions of odds ratios and hazard ratios to risk ratios. Biometrics, 76, 746-752. doi:10.1111/biom.13197

Examples

```
# Risk ratio 3.9 (95% CI 1.8 to 8.7) for an exposure in a cohort study
tv_evalue(3.9, 1.8, 8.7)

# Odds ratio for a common outcome
tv_evalue(1.6, 1.1, 2.3, type = "or")
```

---

tv_save *Save advice and interpretations to a file*

Description

tidyvisor prints its advice with the cli package. In the RStudio console, cli writes straight to the console, so `capture.output()` and `sink()` record nothing there. `tv_save()` writes the advice as plain text instead, and `format()` returns it as a character vector, one element per line.

Usage

```
tv_save(..., file, append = FALSE, width = 80)

## S3 method for class 'tv_advice'
format(x, ..., width = 80)

## S3 method for class 'tv_interpretation'
format(x, ..., width = 80)
```

Arguments

`...`	Objects to save, such as the results of `tidyvisor()`.
`file`	Path of the file to write. A file ending in `.md` gets a heading per object and the text in code blocks, ready for a report or a Quarto / R Markdown document; any other file gets plain text.
`append`	Add to the end of `file` instead of replacing it.
`width`	Line width of the text.
`x`	An object printed by tidyvisor.

Details

`tv_save()` works on anything tidyvisor prints: advice from `tidyvisor()`, `advise()`, `tv_power()` and `tv_practice()`, interpretations from `interpret()`, and the advice of the packages that register an advisor (tidydiversity, tidymapr, tidyepidemio). Give several objects to save a whole consultation in one file.

Value

`tv_save()` returns `file`, invisibly. `format()` returns a character vector.

Examples

```
adv <- tv_power("two_groups", delta = 5, sd = 8)
res <- tidyvisor(lm(mpg ~ wt + hp, data = mtcars))
f <- tempfile(fileext = ".md")
tv_save(adv, res, file = f)
cat(readLines(f), sep = "\n")

# The text as a character vector
head(format(adv))
```

`vegan-advisor` *Advisor for vegan: ordination and multivariate tests*

Description

Built-in advisor for the vegan package, for community data in the wide format vegan uses (one row per site, one column per species).

- **Before** (`question = "vegan"`): reports the number of sites and species, the share of zeros, and the length of the main gradient. From these it recommends a linear or a unimodal method, NMDS for an unconstrained view, and PERMANOVA (`vegan::adonis2()`) with its companion check of dispersion (`vegan::betadisper()`).

The gradient-length rule. The gradient length is the length of the longest axis of a detrended correspondence analysis (`vegan::decorana()`, detrending by segments, on log(y + 1) abundances or presence-absence), in standard-deviation (SD) units of species turnover. This is the value CANOCO reports. Over about 4 SD the species composition turns over completely.

Gradient length	Species responses	Method
below 3 SD	mostly monotonic	linear: PCA, RDA (on Hellinger-transformed abundances)
3 to 4 SD	mixed	both appropriate; tidyvisor gives RDA, with CCA as the alternative
above 4 SD	unimodal (rise and fall)	unimodal: CA, CCA

The rule guides; it does not decide. Hellinger-transformed RDA (Legendre & Gallagher 2001) and distance-based methods (NMDS, db-RDA, PERMANOVA) are valid at any gradient length.

- **After**: explains `vegan::metaMDS()` (stress against Clarke's benchmarks), `vegan::rda()`, `vegan::cca()`, `vegan::capscale()` and `vegan::dbrda()` (variance explained, adjusted R-squared, permutation tests), `vegan::adonis2()` and `anova()` tables, `vegan::betadisper()` and `vegan::envfit()`.

For long-format data, tidydiversity offers the same analyses through `ord_fit()`; its advisor answers `tidyvisor(data, abundance = ...)`.

Usage

```
tv_advise_vegan(
  x,
  question = "vegan",
  species = NULL,
  env = NULL,
  blocks = NULL,
  time = NULL,
  lon = NULL,
  lat = NULL,
  traits = NULL,
  ...
)

tv_interpret_vegan(x, permutations = 999, ...)
```

Arguments

`x`	A wide data frame of sites (advice), or a vegan result (interpretation).
`question`	The question; set by `advise()`.
`species`	Names of the species columns (default: every numeric column not in `env`).
`env`	Names of the explanatory columns, if any.
`blocks`	Name of a column of blocks: permutations are restricted to within blocks (`tidydiversity::ord_design`
`time`	Name of a time (or transect position) column: permutations keep the order of the series.
`lon, lat`	Names of coordinate columns: the advice adds variation partitioning between environment and space (`tidymapr::geo_varpart()`).
`traits`	Name of a trait table, if traits were measured: the advice adds trait-environment analysis (dc-CA or RLQ).
`...`	Ignored.
`permutations`	Number of permutations for the tests run while interpreting a constrained ordination.

Value

A `tv_advice` or a `tv_interpretation` object.

References

Anderson, M.J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology* 26, 32-46.

Clarke, K.R. (1993). Non-parametric multivariate analyses of changes in community structure. *Australian Journal of Ecology* 18, 117-143.

Leps, J. & Smilauer, P. (2003). *Multivariate Analysis of Ecological Data using CANOCO*. Cambridge University Press. (Second edition for Canoco 5, 2014.)

Legendre, P. & Gallagher, E.D. (2001). Ecologically meaningful transformations for ordination of species data. *Oecologia* 129, 271-280.

Ter Braak, C.J.F. & Smilauer, P. (2012). *Canoco Reference Manual and User's Guide: Software for Ordination (version 5.0)*. Microcomputer Power, Ithaca.

Examples

```
data("dune", "dune.env", package = "vegan")
tidyvisor(cbind(dune, dune.env), question = "vegan", env = c("A1", "Management"))
m <- vegan::rda(vegan::decostand(dune, "hellinger") ~ A1 + Management, dune.env)
tidyvisor(m, permutations = 99)
```

Index

advise, 2
advise(), *4*, *6*, *8*, *11–15*, *17*, *22, 23*
advisors, 3
advisors(), *3*, *15*, *17*
aods3::aodml(), *7*

brms-advisor, 4
brms::brm(), *4*

compare-advisor, 5

format.tv_advice *(*tv_save*)*, 21
format.tv_interpretation *(*tv_save*)*, 21

glmmTMB::glmmTMB(), *7*

interpret, 6
interpret(), *17*, *22*

lme4::glmer(), *7*
lme4::glmer.nb(), *7*
lme4::lmer(), *7*

mgcv::gam(), *7*
model-advisors, 7

power-advisor, 9
practice-advisor, 12

register_advisor, 14
register_advisor(), *17*
regression-advisor, 15

set.seed(), *18*
stats::chisq.test(), *5*
stats::confint(), *15*
stats::cor.test(), *5*
stats::fitted(), *18*
stats::glm(), *15*
stats::kruskal.test(), *5*
stats::lm(), *15*
stats::oneway.test(), *5*
stats::shapiro.test(), *5*
stats::simulate(), *18*
stats::t.test(), *5*
stats::wilcox.test(), *5*

tidydiversity::ord_design(), *23*
tidyvisor, 17
tidyvisor(), *14, 15*, *21, 22*
tv_advise_aod *(*model-advisors*)*, 7
tv_advise_brms *(*brms-advisor*)*, 4
tv_advise_compare *(*compare-advisor*)*, 5
tv_advise_design *(*practice-advisor*)*, 12
tv_advise_gam *(*model-advisors*)*, 7
tv_advise_mixed *(*model-advisors*)*, 7
tv_advise_mixed(), *4*
tv_advise_power *(*power-advisor*)*, 9
tv_advise_regression
*(*regression-advisor*)*, 15
tv_advise_vegan *(*vegan-advisor*)*, 22
tv_check_model, 18
tv_equivalence, 19
tv_evalue, 20
tv_interpret_aod *(*model-advisors*)*, 7
tv_interpret_brms *(*brms-advisor*)*, 4
tv_interpret_gam *(*model-advisors*)*, 7
tv_interpret_htest *(*compare-advisor*)*, 5
tv_interpret_mixed *(*model-advisors*)*, 7
tv_interpret_regression
*(*regression-advisor*)*, 15
tv_interpret_vegan *(*vegan-advisor*)*, 22
tv_power *(*power-advisor*)*, 9
tv_power(), *22*
tv_practice *(*practice-advisor*)*, 12
tv_practice(), *22*
tv_save, 21

vegan-advisor, 22
vegan::adonis2(), *22, 23*
vegan::betadisper(), *22, 23*

vegan::capscale(), *23*
vegan::cca(), *23*
vegan::dbrda(), *23*
vegan::decorana(), *22*
vegan::envfit(), *23*
vegan::metaMDS(), *23*
vegan::rda(), *23*