Package: crimlab (via r-universe)

November 13, 2026

Title Criminology and Sociolegal Statistics

Version 0.0.0.9000

Description Statistical methods for courts, corrections, policing, forensics and sociolegal research, computed in a compiled core with a thin R layer. Where an identity or bound has a machine-checked proof ('Lean'), the package names it. This development version holds the disparity tests: the veil-of-darkness contrast of Grogger and Ridgeway (2006) <doi:10.1198/016214506000000168>, the outcome test and its infra-marginality problem, benchmark decomposition of resident disparities, dyadic offending rates, and the bounds a relative risk obeys given an odds ratio; and the first forensic methods: genotype match probabilities with the correction of Balding and Nichols (1994) <doi:10.1016/0379-0738(94)90222-4>, the log-likelihood-ratio cost of Brummer and du Preez (2006) <doi:10.1016/j.csl.2005.08.001> with its calibrated minimum, mixture likelihood ratios, kinship and paternity indices, database-search corrections, and Tippett and empirical-cross-entropy plots.

License Apache License (== 2.0) | file LICENSE

URL https://github.com/rootcoder007/crimlab, https://rootcoder007.github.io/crimlab/

BugReports https://github.com/rootcoder007/crimlab/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports graphics

Suggests igraph, knitr, rmarkdown, testthat (>= 3.1.5), circular, DescTools, geosphere, geepack, MASS, sandwich, sf, spatstat.explore, spatstat.geom, spdep

VignetteBuilder knitr

SystemRequirements C++17

Config/testthat/edition 3

Roxygen list(markdown = TRUE, roclets = c(``namespace'', ``rd'', ``srr::srr_stats_roclet''))

Config/roxygen2/version 8.0.0

Repository https://rootcoder007.r-universe.dev

Date/Publication 2026-11-13 08:45:19 UTC

RemoteUrl https://github.com/rootcoder007/crimlab

RemoteRef main

RemoteSha 3a04fa3e4ea19e49220e20edc13dd0bc0db42f65

Contents

aorwt . . . 4
bdig . . . 6
bjtc . . . 7
calaud . . . 8
catchk . . . 9
cgmedn . . . 11
cgstat . . . 12
chmb . . . 13
cllr . . . 14
cllrmin . . . 15
clprep . . . 15
clrsmp . . . 17
cntreg . . . 18
coofg . . . 20
crwalk . . . 21
ctabx . . . 22
cvhull . . . 24
dastd . . . 25
dbsrch . . . 26
dirmn . . . 27
dmlx . . . 28
dspbm . . . 31
dyadrt . . . 32
eceplt . . . 33
enpath . . . 34
eqproj . . . 36
expbnd . . . 37
fair . . . 38
fairdf . . . 39
fairsim . . . 41
fzmode . . . 42
gearc . . . 43
gorg . . . 44

grprr . . . 45
gstr . . . 46
gtmp . . . 47
hdbang . . . 48
hvdist . . . 50
hzdsel . . . 51
impfpr . . . 52
incap . . . 53
incapr . . . 54
jtacc . . . 55
jtcal . . . 56
jtdecay . . . 57
jtsurf . . . 58
kinidx . . . 59
kmpart . . . 60
lgtreg . . . 62
lipaud . . . 63
lmorn . . . 64
locgi . . . 65
losbnd . . . 66
lrcal . . . 66
lrpost . . . 67
lwss . . . 68
mantst . . . 68
misrd . . . 70
mixlr . . . 71
mvtll . . . 72
nnhclu . . . 73
nnidx . . . 75
nordet . . . 76
nsbnd . . . 77
ntbrk . . . 78
ntbtw . . . 79
ntchg . . . 80
ntcls . . . 81
ntcmp . . . 82
ntcore . . . 83
ntdeg . . . 84
ntevc . . . 85
ntlvn . . . 86
ntmod . . . 87
oaxbd . . . 88
orrisk . . . 90
outtst . . . 90
patidx . . . 91
proof . . . 92
relchk . . . 92
rfor . . . 94

ripenv 96
ripk 97
rnkres 98
rnnhcl 99
rptc 100
rswp 101
rtmhs 102
scrdsp 103
sdell 104
sellab 106
sirex 107
skp 108
sntbnd 109
sptst 110
spwt 111
stacel 112
stmavg 114
tfrp 115
tipplt 116
tlvc 117
tmpaud 118
vodtst 119
webpt 120
zgll 121

Index **123**

`aorwt` *Aoristic weights*

Description

Spreads each event known only to have happened in an interval `[from, to)` with uniform density over that interval and counts each instant in the bin of its local clock time (Ratcliffe 2002): with the defaults, the 168 hours of the week from Sunday 00:00. A change of clock (daylight saving) moves the remaining time to the right hours: skipped hours get nothing and a repeated hour gets both. Missing values follow stated rules: an event with no `from` is left out; with no `to`, a reversed interval or one shorter than `lo` seconds, its whole weight goes to the start's bin; one of at least `hi` seconds is spread evenly over all bins.

Usage

```
aorwt(
  from,
  to,
  bw = 3600,
  nb = 168,
  origin = 259200,
```

```
  off = 0,
  tt = NULL,
  toff = NULL,
  lo = 120,
  hi = 604800,
  each = FALSE
)
```

Arguments

from, to	The start and end of each event's interval: date-times (POSIXct) or seconds since 1970-01-01 UTC; NA allowed as above, not infinite values.
bw	The bin width in seconds, positive.
nb	The number of bins in a period: one whole number from 1.
origin	The local clock time at which bin 1 starts, in the same units (the default is Sunday 1970-01-04 00:00).
off	The offset of the local clock from UTC in seconds, before tt[1] (for a fixed offset, the only one).
tt, toff	NULL, or the instants at which the local clock's offset changes (strictly increasing) and the new offsets.
lo, hi	The lengths in seconds below which an interval counts at its start and from which it counts everywhere, 0 <= lo < hi (lo = 0, hi = Inf spread every interval exactly).
each	TRUE to return each event's weights.

Value

A list: tot (the summed weights by bin), w (NULL, or a matrix with one row per event and one column per bin, rows summing to 1, 0 for events left out) and cnt (events left out, without end, reversed, short and long).

References

Ratcliffe, J. H. (2002). Aoristic signatures and the spatio-temporal analysis of high volume crime patterns. *Journal of Quantitative Criminology*, 18(1), 23-43. doi:10.1023/A:1013240828824

Examples

```
f <- as.POSIXct(c("2024-03-04 22:00", "2024-03-05 09:30"), tz = "UTC")
t <- as.POSIXct(c("2024-03-05 02:00", "2024-03-05 10:30"), tz = "UTC")
a <- aorwt(f, t, bw = 3600, nb = 24, origin = 0)
a$tot
```

bdig *Benford digit screens*

Description

`bdig()` takes the first `nd` significant digits of each value. With `dp = -1` they are the digits of the shortest decimal that gives back the double, decided exactly, so 0.29 gives 29 where truncating `x` times a power of ten gives 28; with `dp` from 0 to 22 the data are taken as recorded to `dp` decimals. `bnft()` tests the digits against Benford's law, `P(d) = log10(1 + 1 / d)`, by the chi-square statistic with its exact upper tail, a Monte Carlo p-value from samples of Benford digits of the same size, and Nigrini's mean absolute deviation. A departure from Benford's law is a reason to look, not a finding of fabrication: many honest data sets (bounded amounts, assigned numbers) do not follow it.

Usage

```
bdig(x, nd = 1, dp = -1, sgn = "pos")

bnft(dig, nd = 1, nsim = 0, seed = NULL)
```

Arguments

`x`	Numbers. Zeros, non-finite values (including `NA`), values of the excluded sign and values outside the exact range are set aside and counted.
`nd`	The number of leading digits: from 1 to 9 for `bdig()`, 1 to 6 for `bnft()`.
`dp`	`-1`, or the number of decimals the data were recorded to.
`sgn`	Which values to keep: `"pos"`, `"neg"` or `"both"` (by absolute value).
`dig`	Digits as `bdig()` returns them (0 is skipped).
`nsim`	The number of Monte Carlo samples: a whole number from 0.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.

Value

`bdig()`: a list of `dig` (an integer per value, 0 when set aside) and `cnt` (zeros, non-finite, excluded sign, out of range). `bnft()`: a list of `obs` (a data frame of `digit`, observed `count` and Benford `expected` count) and `stat` (`m` digits used, `chisq`, `df`, `p`, `pmc` (Monte Carlo, `NaN` without samples) and `mad`).

References

Benford, F. (1938). The law of anomalous numbers. *Proceedings of the American Philosophical Society*, 78(4), 551-572.

Nigrini, M. J. (2012). *Benford's Law: Applications for Forensic Accounting, Auditing, and Fraud Detection*. Wiley. doi:10.1002/9781119203094

Examples

```
set.seed(1)
x <- exp(runif(500, 0, 10))
d <- bdig(x)$dig
bnft(d, nsim = 199, seed = 1)$stat
```

`bjtc` *Bayesian journey to crime*

Description

Combines a journey-to-crime surface over origin zones with an origin-destination matrix of solved offences (Levine 2009; Levine and Block 2011): `P(Jtc)`, the surface; `P(O)`, the origins of all solved offences; `P(O|Jtc)`, the origins of offences in the zones where the series happened; their product, Bayes' numerator with the surface as prior; and the risk, the product over `P(O)`, renormalised. Each block sums to 1. The matrix describes past solved offences and inherits their selection.

Usage

```
bjtc(pj, od, dest)
```

Arguments

`pj`	The journey-to-crime value of each origin zone: numbers from 0, not all 0.
`od`	The origin-destination counts: a matrix with one row per origin zone and one column per destination zone, numbers from 0.
`dest`	The destination zone (column of `od`) of each incident of the series: whole numbers from 1; repeats count.

Value

A data frame with one row per origin zone: `pj`, `po`, `pod`, `prod` and `risk` (0 where `po` is 0).

References

Levine, N. and Block, R. (2011). Bayesian journey-to-crime estimation: an improvement in geographic profiling methodology. *The Professional Geographer*, 63(2), 213-229. doi:10.1080/00330124.2010.547152

Examples

```
od <- matrix(c(5, 1, 0, 2, 4, 1, 0, 1, 6), 3)
bjtc(pj = c(0.5, 0.3, 0.2), od = od, dest = c(1, 2, 2))
```

`calaud` *Calibration audit of a predictive-policing ranking*

Description

Does the order in which an algorithm ranks areas by predicted risk match the order of what was later recorded there, and does the disagreement follow the areas' demographic make-up? `calaud()` ranks the areas both ways, measures the agreement by Spearman's correlation, and gives each group the mean gap between its areas' two ranks, with a permutation test. It generalises the district-level analysis of the SciencesPo predictive-policing study of Chicago (Lacherade et al. 2021) to any place: the five columns of the audit are `area`, `risk`, `outcome`, `group` and `population`.

Usage

```
calaud(
  area,
  risk = NULL,
  outcome = NULL,
  group = NULL,
  population = NULL,
  nperm = 999,
  seed = NULL
)
```

Arguments

`area`	The area of each record: labels; or a data frame with columns `area`, `risk`, `outcome`, `group` and optionally `population`, the other arguments then left out. A `clpre` list from `clprep()` is taken too: its elements fill the arguments by name.
`risk`	The risk the algorithm predicts for each record: finite numbers.
`outcome`	The recorded outcome of each record (a 0 or 1, or a count): finite numbers.
`group`	The group of each record (or area): labels.
`population`	`NULL`, or the population of each area: positive, one per record (the same within an area) or one per area named by area.
`nperm`	The number of permutations: a whole number from 0 (none).
`seed`	`NULL` to draw the seed of the bootstrap (or the permutations) from R's generator, so that `set.seed()` fixes it, or one whole number from 0.

Details

Records are rolled up to areas first: an area's risk is the mean over its records, its rate the mean outcome (or, with `population`, the summed outcome per head), and its group the most frequent label among its records (the first in order on ties). One record per area is the per-area audit itself. Rank 1 is the highest; tied values share the mean of their ranks. An area's gap is its outcome

rank minus its risk rank: positive when the algorithm ranks it as riskier than its recorded outcome warrants, negative when less risky.

Spearman's rho is Pearson's correlation of the two rankings (Spearman 1904). `pex` is its two-sided p-value by algorithm AS 89 (Best and Roberts 1975): exact for nine areas or fewer, from the Edgeworth series up to 1289, and `NA` when either ranking has ties; `pt` is the large-sample one from the t distribution, as `stats::cor.test()` gives them. A group's mean gap is tested against shuffling the groups across the areas, `nperm` times; `between` tests all groups at once by the between-group sum of squares of the gaps (the gaps sum to 0).

The audit compares the algorithm with the records, not with crime: a recorded outcome follows where the police looked as well as what happened, so a group's areas can look over-predicted because their incidents were recorded less, or under-predicted because they were recorded more (see `fairsim()`). A gap says the two orders disagree for that group's areas; it does not say which order is right or why.

Value

A list: `areas`, a data frame with one row per area (sorted by label): `area`, `n` (records), `risk`, `rate`, `group`, `riskrk`, `raterk` and `gap`; `groups`, one row per group: `group`, `n` (areas), `gap` (mean) and `p` (`NaN` without areas or permutations); `spear`, a named vector: `rho`, `s` (Spearman's statistic, the sum of squared rank differences), `pex`, `t` and `pt`; `ties` (`TRUE` when either ranking has ties); `between`, `ss` and `p`; and `nperm`.

References

Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15(1), 72-101. doi:10.2307/1412159

Best, D. J. and Roberts, D. E. (1975). Algorithm AS 89: the upper tail probabilities of Spearman's rho. *Journal of the Royal Statistical Society, Series C*, 24(3), 377-379. doi:10.2307/2347111

Lacherade, P., Szabo, B., Krikava, J. and Aeby, M. (2021). Predictive policing in the City of Chicago. Group project, Decoding Biases in AI, Sciences Po. https://github.com/marquito95/Predictive-policing-Chicago

Examples

```
set.seed(3)
d <- fairsim(4000, nareas = 24, bias = 0.5, seed = 1)
# the recorded outcome is `det`: the incidents that were detected
r <- calaud(d$area, d$risk, d$det, d$group, nperm = 499, seed = 1)
r$groups
r$spear
```

`catchk` *Audit categorical inputs before an analysis*

Description

Reports, for each variable, what the functions of this package would make of it: its storage type, its levels and their counts, missing values, unused levels, levels counted fewer than `nmin` times, labels that differ only by case or white space, a 0/1 or logical coding, integer codes that look like categories, value labels (as haven makes) that do not match the values, and which level a default would take as the reference, and why. Nothing is changed.

Usage

```
catchk(x, ref = NULL, nmin = 5)
```

Arguments

`x`	A vector, a data frame, or a list of vectors (each one a variable, named by its name or position).
`ref`	`NULL`, or the reference level the analysis intends: one level for every variable, or levels named by variable (a named vector or list). Matched exactly to the levels as text; a factor by its label.
`nmin`	Levels counted fewer times (but at least once) are reported as low: one whole number from 1.

Details

Levels are those of a factor, in level order; `FALSE` and `TRUE` for a logical, and `0` and `1` for numbers that are all 0 or 1, both always listed; otherwise the distinct values as text, in order of first appearance, and for a labelled vector then the labelled values that do not occur. Doubles that are not all whole numbers, dates, times, lists and matrices are not categorical: they are reported with their type and missing values only.

The default reference: `FALSE` for a logical and `0` for 0/1 numbers (`sntbnd()` takes `TRUE` and 1 as the treatment), the first level of a factor (`ref = 1` on a table of it), and none otherwise: character values and codes have no order the package uses, so the reference must be stated, and a labelled vector is refused until it is decoded. Labels are compared after trimming white space, making each run of it one space and ASCII letters lower case; other letters are compared as they are.

Value

An object of class `catchk`, a list: `ok` (`TRUE` when no variable has a problem), `nmin`, `vars`, a data frame with one row per variable, and `levels`, one row per level. `vars`: `var`, `type`, `n` (values), `nna` (missing), `nlev` (levels), `unused` (levels never seen), `low` (levels seen fewer than `nmin` times), `twin` (levels with a label that differs from another only by case or white space), `bin` (logical or 0/1), `codes` (whole numbers, not 0/1, with at most 10 distinct values: likely category codes), `nolab` (values seen with no value label), `dlab` (value labels shared by two values), `ref` (the default reference, `NA` for none), `why`, `want` (the stated reference), `state` (`"default"`: the stated level is the default; `"other"`: present, but must be passed explicitly; `"low"`, `"empty"` (never seen), `"case"` (present only up to case or white space), `"absent"`; `NA` when none is stated) and `ok`. `nlev`, `unused`, `low` and `twin` are `NA` for a variable that is not categorical. `levels`: `var`, `level`, `label` (the value label, `NA` where none), `n`, `low`, `twin` (the level it differs from only by case or white space, `NA` where none) and `ref` (`TRUE` for the default reference). A variable has a problem when it has missing values, unused or

low levels, twins, value labels, likely codes, fewer than two levels seen, or a stated reference in state `"low"`, `"empty"`, `"case"` or `"absent"`.

Examples

```
d <- data.frame(sex = c("Male", "male", "Female", "Female", NA),
                custody = c(TRUE, FALSE, TRUE, TRUE, FALSE),
                court = c(1L, 2L, 2L, 3L, 1L))
catchk(d, nmin = 2)
catchk(factor(c("b", "a", "b"), levels = c("a", "b", "c")), ref = "b")
```

---

cgmedn *Median centre of points*

Description

The median of the `x` coordinates and the median of the `y` coordinates (the mean of the two middle values for an even number of points). It depends on the orientation of the axes: rotating the points does not rotate it. `webpt()` gives the rotation-invariant centre of minimum distance.

Usage

```
cgmedn(x, y = NULL, lonlat = FALSE)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y` `NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`lonlat`	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A named numeric vector `c(x, y)`, in degrees when `lonlat = TRUE` (the medians are taken of the projected coordinates).

References

Neft, D. S. (1966). *Statistical Analysis for Areal Distributions*. Regional Science Research Institute, Philadelphia.

Examples

```
cgmedn(c(1, 2, 3, 10), c(5, 1, 2, 2))
```

`cgstat` *Mean centre and dispersion of points*

Description

The (weighted) mean centre of a point pattern, the standard deviations of the two coordinates, the standard distance deviation (the root mean squared distance from the mean centre), and the geometric and harmonic means of the coordinates (Bachi 1957; Neft 1962). The mean centre is the point that minimises the sum of squared distances to the points. The dispersions divide by the total weight less `k`: `k = 1` makes the squared values unbiased for the variances; `k = 2` is the convention of some crime-mapping texts; `k = 0` divides by the total weight. That the mean centre minimises the weighted sum of squared distances is proved in Lean: see `proof("cgstat")`.

Usage

```
cgstat(x, y = NULL, w = NULL, k = 1, lonlat = FALSE)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`w`	`NULL` for equal weights, or one non-negative weight per point (counts at a location, say), with a positive sum.
`k`	The divisor offset: 0, 1 or 2; the total weight must exceed it.
`lonlat`	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A named numeric vector: `w` (the total weight), `mx`, `my` (the mean centre, in degrees when `lonlat = TRUE`), `sx`, `sy` (standard deviations of the coordinates), `sdd` (the standard distance deviation), `gx`, `gy` (geometric means) and `hx`, `hy` (harmonic means). The last four are `NaN` for a coordinate with a non-positive value under positive weight, and for `lonlat = TRUE` (they depend on the origin of the coordinates). Lengths are in the unit of the coordinates, or metres.

References

Bachi, R. (1957). Statistical analysis of geographical series. *Bulletin de l'Institut International de Statistique*, 36(2), 229-240.

Neft, D. S. (1966). *Statistical Analysis for Areal Distributions*. Regional Science Research Institute, Philadelphia.

Examples

```
set.seed(1)
p <- cbind(x = rnorm(50, 100, 10), y = rnorm(50, 200, 5))
cgstat(p)
cgstat(p[, 1], p[, 2], w = rep(1:2, 25))
cgstat(p, k = 0)  # dispersions over n, not n - 1
```

---

chmb *The Chumbley score for two tool-mark profiles*

Description

Compares two striated tool-mark profiles by the deterministic Chumbley score (Chumbley et al. 2010; Hadler and Morris 2018): each profile is trimmed and detrended by its lowess smooth, the pair of windows of `wopt` points with the largest correlation fixes the shift, and windows of `wval` points at that shift are compared with windows at other shifts by a standardised Wilcoxon rank-sum statistic.

Usage

```
chmb(a, b, wopt, wval, coarse = 0.25)
```

Arguments

`a`, `b`	The two profiles: finite numbers (heights along the mark).
`wopt`, `wval`	The window lengths of the optimisation and validation steps: whole numbers from 2 with `wval` <= `wopt`.
`coarse`	The lowess span of the detrending: positive.

Value

A named numeric vector: `nsame`, `ndiff` (window pairs used), `u` (the statistic) and `p` (its upper normal tail; `NaN` when either set is empty).

References

Chumbley, L. S., Morris, M. D., Kreiser, M. J. et al. (2010). Validation of tool mark comparisons obtained using a quantitative, comparative, statistical algorithm. *Journal of Forensic Sciences*, 55(4), 953-961. doi:10.1111/j.15564029.2010.01424.x

Hadler, J. R. and Morris, M. D. (2018). An improved version of a tool mark comparison algorithm. *Journal of Forensic Sciences*, 63(3), 849-855. doi:10.1111/15564029.13640

Examples

```
set.seed(1)
m <- cumsum(rnorm(700))
a <- m[1:600] + rnorm(600, sd = 0.2)
b <- m[51:650] + rnorm(600, sd = 0.2)
chmb(a, b, wopt = 120, wval = 30)
```

`cllr` *Log-likelihood-ratio cost*

Description

Scores a set of likelihood ratios against the truth. Ratios from same-source comparisons should be large and ratios from different-source comparisons small. The cost is 1 for a system that always answers 1. It is above 0 for any set of ratios, and tends to 0 as the ratios of a system that separates the two kinds perfectly grow without bound. See `proof("cllr")` and `cllrmin()`.

Usage

```
cllr(same, diff)
```

Arguments

`same`	Likelihood ratios from same-source comparisons: positive numbers.
`diff`	Likelihood ratios from different-source comparisons: positive numbers.

Value

One number, the cost in bits.

References

Brummer, N. and du Preez, J. (2006). Application-independent evaluation of speaker detection. *Computer Speech and Language*, 20(2-3), 230-275. doi:10.1016/j.csl.2005.08.001

Examples

```
cllr(same = c(50, 200, 8), diff = c(0.01, 0.2, 1.5))
```

cllrmin — *Log-likelihood-ratio cost after recalibration*

Description

Splits the cost of a set of likelihood ratios into `min`, their cost after recalibration with the pool-adjacent-violators algorithm, which keeps their order and gives each block of pooled ratios the value that costs that block least, and `cal`, the rest (`cllr - min`). That `min` is the least cost of every order-keeping recalibration, so that `cal` is never negative, is not proved. A system can separate the two kinds of comparison well and still report ratios that are too large or too small. See `proof("cllrmin")` and `proof("lrcal")`.

Usage

```
cllrmin(same, diff)
```

Arguments

`same`	Likelihood ratios from same-source comparisons: positive numbers.
`diff`	Likelihood ratios from different-source comparisons: positive numbers.

Value

A named numeric vector: `cllr`, `min` and `cal`, with `cllr = min + cal`.

References

Brummer, N. and du Preez, J. (2006). Application-independent evaluation of speaker detection. *Computer Speech and Language*, 20(2-3), 230-275. doi:10.1016/j.csl.2005.08.001

Examples

```
cllrmin(same = c(50, 200, 8), diff = c(0.01, 0.2, 1.5))
```

clprep — *Presets: a published dataset's columns onto a generic function's arguments*

Description

A preset says how the columns of one published dataset (an Ontario segregation table, the Special Investigations Unit's cases, a use-of-force file) feed the arguments of one of crimlab's generic functions, so that `generic(clprep(data, preset))` gives the published analysis while the same generic runs on anyone's data given as plain vectors. A preset describes a layout and ships no data: the user supplies the table.

Usage

```
clprep(data = NULL, preset = NULL)
```

Arguments

data	The dataset: a data frame with the columns the preset names. NULL (with preset NULL) lists the presets.
preset	The preset's name, as clprep() lists them; NULL for the list.

Details

The presets are rows of presets.tsv (in the installed package's presets folder), one per preset and argument: preset, generic, arg, column (the dataset's column), recode, filter, units, note, source. Python reads the same file. A recode is one of: empty (the column as it is); fac (a factor, one effect per value); num (numbers); one (a 1 for every row); sub:a,b (characters a to b, as a year from an ISO date); const:v (the argument is v, no column); or a mapping x|y=1;has:abc=1;*=0 from labels, compared without case or surrounding space (has: a label containing the text), to values, * covering every other label and *=NA dropping those rows. A filter col=value keeps the rows whose column has that value. An arg of recipe derives columns first: long stacks the two count columns named in recode as count with with 1 for the first; acvm and pyr reduce one row per placement to one per person and year (see Details of the returned attributes).

Rows with a missing value in any column the preset reads are dropped, as the source analyses drop them; their number is the attribute drop.

Value

With a preset, an object of class clpre: a list of the generic's arguments, which the generic takes as its first argument, with attributes preset, generic, source, units, note and drop. Without one, a data frame of the presets: preset, generic, args and source.

Examples

```
head(clprep())
# a table laid out as the published OTIS c01 table (numbers made up)
d <- data.frame(EndFiscalYear = rep(2020:2022, each = 2),
                Gender = rep(c("Female", "Male"), 3),
                NumberIndividuals_RestrictiveConfinement =
                  c(40, 310, 52, 335, 47, 360))
p <- clprep(d, "otis-c01-cnt")
cntreg(p)
# the same model on plain vectors
cntreg(d$NumberIndividuals_RestrictiveConfinement, d$Gender == "Female",
       adj = factor(d$EndFiscalYear))
```

`clrsmp` *Clearance-rate decomposition and Simpson's paradox*

Description

Splits the change in an aggregate clearance rate between two periods, areas or services (A and B) into a rate effect, from each offence type clearing more or less, and a mix effect, from the workload moving between types that clear at different rates (Kitagawa's decomposition). The two always add up to the change. When every type clears better in B and the aggregate still falls, the fall is all mix: Simpson's paradox, and the reason an aggregate clearance rate cannot be read as performance without the split by type. See `proof("clrsmp")`.

Usage

```
clrsmp(clra, reca, clrb, recb)
```

Arguments

`clra`	Offences cleared in A, by type: non-negative numbers, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame.
`reca`	Offences recorded in A, by type: positive numbers, in the forms of `clra`; named inputs are matched to `clra` by name, unnamed ones by position (only names protect against a mismatch).
`clrb`	Offences cleared in B, by type: non-negative numbers, in the forms of `clra`, matched by name.
`recb`	Offences recorded in B, by type: positive numbers, in the forms of `clra`, matched by name.

Details

Each part's mix term is measured from the mean of the two aggregate rates, so a part's term is negative when work moved toward a part that clears below average; the centre changes how the mix effect is shared among the parts, never its total.

Value

A list. `parts`, a data frame with one row per type, in the order of `clra`: `part`, `ratea`, `rateb` (clearance rates), `sharea`, `shareb` (shares of the recorded workload), `rateeff` and `mixeff`. `total`, a named numeric vector: `ratea`, `rateb` (aggregate rates), `change`, `rateeff`, `mixeff` (with `change` = `rateeff` + `mixeff`), `up` and `down` (types whose rate rose or fell) and `simpson`: 2 when every type moved one way and the aggregate the other, 1 when the aggregate change and the rate effect differ in sign, 0 otherwise.

References

Kitagawa, E. M. (1955). Components of a difference between two rates. *Journal of the American Statistical Association*, 50(272), 1168-1194. doi:10.1080/01621459.1955.10501299

Examples

```
# both types clear better in B, yet the aggregate falls
clrsmp(clra = c(minor = 90, serious = 20), reca = c(100, 50),
       clrb = c(19, 60), recb = c(20, 140))
```

`cntreg` *Count regression: rate ratios between groups*

Description

Fits a log-linear model of a count on a group, with adjustment columns, an exposure and clusters, by Poisson and negative binomial maximum likelihood, and reports the incidence rate ratio (IRR) of each group against the reference with its interval. The same model serves counts of people, placements, days or deaths in the cells of a published table and counts per person or per area.

Usage

```
cntreg(
  y,
  grp = NULL,
  adj = NULL,
  expo = NULL,
  clus = NULL,
  ref = NULL,
  gee = FALSE,
  cadj = TRUE,
  level = 0.95
)
```

Arguments

`y`	The counts: finite numbers from 0, not all 0; or a `clpre` list from `clprep()`, whose elements fill the arguments by name (arguments given here override them).
`grp`	The group of each count: labels (a factor by its levels, text or numbers by their sorted values), at least two groups.
`adj`	`NULL` or adjustment columns: a data frame or list of columns, a numeric matrix, or one vector, one value per count. Factor, text and logical columns enter as indicators of each level but the first; give a number such as a year as a factor for one effect per value.
`expo`	`NULL` or the exposure of each count (person-days, population): positive numbers; its logarithm is the offset.
`clus`	`NULL` or the cluster of each count (institution, region, year): labels, at least two clusters.
`ref`	The reference group: a label or a position; the first group by default.

gee	TRUE to fit the exchangeable GEE as well (needs clus).
cadj	TRUE for the G / (G - 1) factor on the cluster-robust variances.
level	The confidence level of the intervals: one number strictly between 0 and 1.

Details

The Poisson fit takes the steps of stats::glm() (iteratively reweighted least squares from y + 0.1, stopping when the relative change of the deviance is below 1e-8), and the negative binomial fit (variance mu + mu^2 / theta) those of MASS::glm.nb(): theta by maximum likelihood and the coefficients in turn. A column of the design that the columns before it explain (an adjustment level the group determines, say) is aliased and dropped, as R does: its coefficients are NA.

With clus, each fit also gets cluster-robust (sandwich) standard errors, rows "poiscl" and "nbcl": (X'WX)^-1 (sum_g U_g U_g') (X'WX)^-1 over the clusters' scores U_g, times G / (G - 1) for G clusters when cadj (the default of sandwich::vcovCL(); cadj = FALSE gives the independence GEE variance of 'geepack'). With gee = TRUE the Poisson model is also fitted by generalised estimating equations with an exchangeable working correlation (row "gee"), as geepack::geeglm(corstr = "exchangeable") estimates it.

Intervals are Wald intervals on the log scale, exp(est +- z se), with z the normal quantile for level. sep flags a ratio whose group or reference has no counts at all: the estimate then runs off towards 0 or infinity and its interval means nothing.

Counts need not be whole numbers (a median duration, say): the estimating equations are those of the Poisson pseudo-likelihood, and the log-likelihood uses lgamma(y + 1), where stats::glm() reports an infinite AIC.

Value

An object of class cntreg, a list: irr, a data frame with one row per fit and group (fit, term, irr, lo, hi, p, sep); coef, every coefficient of every fit on the log scale (fit, term, est, se, z, p); fit, one row per model (fit, loglik, aic, theta, setheta, alpha, iter, conv, rank); ref, the reference group; n, the number of counts; ncl, the number of clusters (0 without clus); and level.

References

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*, 4th ed. Springer. doi:10.1007/9780387217062

Liang, K.-Y. and Zeger, S. L. (1986). Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1), 13-22. doi:10.1093/biomet/73.1.13

Cameron, A. C. and Miller, D. L. (2015). A practitioner's guide to cluster-robust inference. *Journal of Human Resources*, 50(2), 317-372. doi:10.3368/jhr.50.2.317

Examples

```
# placements by group in eight institutions over three years
d <- data.frame(
  n = c(12, 30, 8, 25, 14, 41, 9, 22, 11, 35, 7, 28),
  female = rep(c("no", "yes"), 6),
  year = factor(rep(c(2021, 2022, 2023), each = 4)),
  inst = rep(c("A", "B", "C", "D"), each = 2, length.out = 12)
```

```
)
cntreg(d$n, d$female, adj = d["year"], clus = d$inst)
# rates: a count per person-days of exposure
cntreg(c(5, 12), c("a", "b"), expo = c(1000, 1500))
```

`coofg` *Co-offending graph from an incident table*

Description

The one-mode projection of offenders onto incidents: two offenders are joined when they are recorded in the same incident, and the weight of the tie is the number of incidents they share. An offender recorded twice in one incident counts once there. Offenders who never offend with anyone appear in `nodes` but in no edge; give `nodes` to the network functions to keep them as isolated nodes.

Usage

```
coofg(inc, off)
```

Arguments

`inc`	Incident identifiers, one per row: a vector of labels with no missing values.
`off`	Offender identifiers, one per row: labels, same length.

Details

Ties are listed in the order each pair is first met: incidents in the order they first appear, and within an incident the pairs of its offenders in the order of the rows; `from` is the offender who appears first in `off`. Labels are compared as text (`as.character()`; a factor by its labels).

Value

A list: `edges`, a data frame with `from`, `to` (offender labels) and `w` (incidents shared), and `nodes`, every offender in order of first appearance.

References

Reiss, A. J. (1988). Co-offending and criminal careers. *Crime and Justice*, 10, 117-170. doi:10.1086/449145

Examples

```
g <- coofg(inc = c(1, 1, 1, 2, 2, 3), off = c("a", "b", "c", "a", "b", "d"))
g$edges
ntdeg(g$edges$from, g$edges$to, w = g$edges$w, nodes = g$nodes)
```

`crwalk` *Correlated random walk of a series of events*

Description

Treats the successive locations of a series (incidents attributed to one offender, in time order) as a correlated random walk (Kareiva and Shigesada 1983): steps of independent lengths and changes of heading (turning angles) independent of each other and of the steps. The fit gives the mean and mean squared step and the mean cosine and sine of the turns; the expected squared displacement after `k` moves is Kareiva and Shigesada's for symmetric turning (mean sine 0), `k E l^2 + 2 (E l)^2 c / (1 - c) (k - (1 - c^k) /` next to the observed mean over every start. The walk is then continued `nsim` times for `h` steps from the last location and heading, each turn and step drawn independently and uniformly from the observed ones (an empirical correlated random walk), and the simulated positions summarise where the next events may fall. It describes movement; it does not identify anyone, and a series with few events gives a rough walk.

Usage

```
crwalk(x, y = NULL, h = 1, nsim = 999, seed = NULL, lonlat = FALSE)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`h`	The number of steps ahead: a whole number from 1.
`nsim`	The number of simulated continuations: a whole number from 1.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.
`lonlat`	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A list. `fit`, a named vector: `steps`, `ml` (mean step), `ml2` (mean squared step), `c` and `s` (mean cosine and sine of the turning angles) and `head` (the last heading, radians anticlockwise from east). `msd`, a data frame with one row per step ahead `k`: `obs` (the observed mean squared displacement over `k` moves, `NA` beyond the series), `exp` (its expectation; it assumes `s` near 0), and `x`, `y`, `sdx`, `sdy` (the mean

and standard deviations of the simulated positions; the means in degrees when `lonlat = TRUE`, the rest in metres). `sim`, the simulated positions: matrices `x` and `y`, one row per walk and one column per step (planar, metres when `lonlat = TRUE`).

References

Kareiva, P. M. and Shigesada, N. (1983). Analyzing insect movement as a correlated random walk. *Oecologia*, 56(2-3), 234-238. doi:10.1007/BF00379695

Examples

```
set.seed(1)
a <- cumsum(rnorm(20, 0, 0.5))
p <- cbind(x = cumsum(cos(a)), y = cumsum(sin(a)))
r <- crwalk(p, h = 3, nsim = 200, seed = 1)
r$fit
r$msd
```

---

ctabx *Cross-tabulation with tests and measures of association*

Description

Builds a two-way table from labels (counting rows, summing a count column of an aggregate table, or counting distinct ids per cell) and gives Pearson's chi-square test, the likelihood-ratio test, Cramer's V with an interval and its bias-corrected form, the contingency coefficient, phi for a 2 x 2 table, the share on the diagonal (the rows whose label is also the column's, as a region at placement against the region now), and on request Fisher's exact test with, for a 2 x 2 table, the conditional odds ratio and its exact interval.

Usage

```
ctabx(
  row,
  col = NULL,
  w = NULL,
  agg = NULL,
  correct = FALSE,
  exact = FALSE,
  level = 0.95
)
```

Arguments

`row`	The row label of each record (a factor by its levels, text or numbers by their sorted values); or a whole table: a numeric matrix or 2-d table of counts, with `col` `NULL`; or a `clpre` list from `clprep()`.
`col`	The column label of each record.

w	NULL, or the value of each record that agg combines into its cell: a count ("sum"), a measure ("max", "min", "mean"), or an id ("dist", the distinct ids per cell).
agg	How the records of a cell combine: "n" (count them, the default without w), "sum" (the default with w), "max", "min", "mean" or "dist". An empty cell is 0.
correct	TRUE for the continuity correction of a 2 x 2 chi-square, as stats::chisq.test()'s default; FALSE (here the default) for Pearson's statistic.
exact	TRUE for Fisher's exact test.
level	The confidence level of the intervals: one number strictly between 0 and 1.

Details

Rows or columns with no counts are dropped before the statistics (their number is drop), as an unused factor level carries no information; the table tab keeps them. Cramer's V is sqrt(chisq / (n (k - 1))), k the smaller dimension, from the uncorrected statistic even when correct is TRUE; its interval inverts the noncentral chi-square distribution, sqrt(ncp / (n (k - 1))), as the default of DescTools::CramerV(); the bias-corrected V is Bergsma's. The exact test enumerates every table with the observed margins and refuses one that would take more than 1e7 tables. With agg other than "n", "sum" or "dist" the table holds values, not counts, and the statistics are NA.

Value

An object of class ctabx, a list: tab, the table (all labels); expd and res, the expected counts and Pearson residuals of the table without empty rows and columns; stat, a one-row data frame: n, chisq, df, p, g2, g2p (the likelihood-ratio statistic and its p-value), v, vlo, vhi, vbc (bias-corrected V), cc (the contingency coefficient), phi, diag, fp (Fisher's p-value), or, orlo, orhi; and drop, agg, correct and level.

References

Agresti, A. (2013). *Categorical Data Analysis*, 3rd ed. Wiley. ISBN 978-0-470-46363-5.

Bergsma, W. (2013). A bias-correction for Cramer's V and Tschuprow's T. *Journal of the Korean Statistical Society*, 42(3), 323-328. doi:10.1016/j.jkss.2012.10.002

Examples

```
# force type by weapon, one row per incident
wp <- c("none", "none", "knife", "gun", "none", "knife", "gun", "none")
fc <- c("hands", "hands", "CEW", "firearm", "CEW", "hands", "firearm",
        "hands")
ctabx(wp, fc)
# region at placement against region now: the diagonal share
ctabx(c("N", "N", "E", "W", "E"), c("N", "E", "E", "W", "E"))
# a published aggregate table: sum its count column
d <- data.frame(race = c("A", "A", "B", "B"), sex = c("F", "M", "F", "M"),
                n = c(12, 40, 9, 51))
ctabx(d$race, d$sex, w = d$n, exact = TRUE)
```

cvhull *Convex hull of points*

Description

The smallest convex polygon holding every point, by Andrew's monotone chain with an exact orientation predicate (Shewchuk 1997), so near-collinear points never make a wrong turn. Points on an edge are not vertices; of coincident points the first is reported.

Usage

```
cvhull(x, y = NULL, lonlat = FALSE)
```

Arguments

x, y	The points: x and y numeric vectors of equal length; or x a matrix or data frame with columns named x and y (or X and Y, as sf::st_coordinates() names them, or lon and lat with lonlat = TRUE) and y NULL; or x an sf or sfc object of POINT geometries, whose coordinate reference system must agree with lonlat. An unnamed two-column matrix is read as x then y; other columns are never taken by position. Vectors given separately (x, y, and w or t where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A list: idx, the row numbers of the vertices, anticlockwise from the point with the lowest x (then lowest y); area and perim, the hull's area and perimeter (in the square and the unit of the coordinates, or square metres and metres when lonlat = TRUE). One distinct point gives one vertex and zeros; collinear points give their two ends, area 0 and twice the segment's length. Each coordinate must be 0 or of magnitude from 1e-140 to 1e150, where the predicate is exact.

References

Andrew, A. M. (1979). Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5), 216-219. doi:10.1016/00200190(79)900723

Shewchuk, J. R. (1997). Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete and Computational Geometry*, 18(3), 305-363. doi:10.1007/PL00009321

Examples

```
cvhull(c(0, 1, 1, 0, 0.5), c(0, 0, 1, 1, 0.5))
```

dastd *Directly standardised rates*

Description

The rate each group would have with the stratum make-up (age bands, say) of a standard population: the sum over strata of the standard's weights times the group's stratum rates. The interval is Fay and Feuer's gamma interval; the ratio of two standardised rates has a log-normal interval and their difference a normal one.

Usage

```
dastd(x, n, strat, grp = NULL, std, ref = NULL, per = 1, level = 0.95)
```

Arguments

x	The events: numbers from 0; or a clpre list from clprep().
n	The exposure of each row: positive numbers.
strat	The stratum of each row: labels.
grp	NULL (one group) or the group of each row: labels. Each group has one row per stratum.
std	The standard population (or weights) of each stratum: numbers from 0 with a positive sum, named by stratum or in the order of the strata (a factor's levels, or the sorted labels).
ref	The reference group: a label or a position; the first group by default.
per	The rates and differences are given per per units of n (1e5 for rates per 100,000, 100 for percentages): one positive number.
level	The confidence level of the intervals: one number strictly between 0 and 1.

Value

An object of class dastd, a list: rates (group, x the events, rate, lo, hi, var the variance of the rate), cmp (one row per group but the reference: group, ratio, lo, hi, diff, dlo, dhi), ref, per and level.

References

Fay, M. P. and Feuer, E. J. (1997). Confidence intervals for directly standardized rates: a method based on the gamma distribution. *Statistics in Medicine*, 16(7), 791-801. doi:10.1002/(SICI)1097-0258(19970415)16:7<791::AIDSIM500>3.0.CO;2#

Examples

```
d <- data.frame(age = rep(c("18-29", "30-49", "50+"), 2),
                grp = rep(c("A", "B"), each = 3),
                x = c(30, 22, 5, 41, 30, 12),
                n = c(5000, 9000, 6000, 3000, 8000, 7000))
dastd(d$x, d$n, d$age, d$grp, std = c(30, 45, 25), per = 1e4)
```

`dbsrch` *Database search*

Description

What a match found by trawling a database is worth. `np` is the bound of the second National Research Council report: the match probability times the number of profiles searched. `any` is the exact chance of at least one chance match among that many unrelated profiles, never above `np`. `post` takes the other view (Balding and Donnelly): everyone else in the database was excluded, so among `pop` possible sources with equal prior the matching person's probability of being the source goes up, not down. See `proof("dbsrch")`.

Usage

```
dbsrch(mp, n, pop = NULL)
```

Arguments

`mp`	The profile match probability: one number above 0 and at most 1.
`n`	The number of profiles searched, the match included.
`pop`	`NULL`, or the number of possible sources, at least `n`.

Value

A named numeric vector: `np`, `any` and, when `pop` is given, `post`.

References

Balding, D. J. and Donnelly, P. (1996). Evaluating DNA profile evidence when the suspect is identified through a database search. *Journal of Forensic Sciences*, 41(4), 603-607.

Examples

```
dbsrch(mp = 1e-9, n = 5e6, pop = 3e7)
```

`dirmn` *Directional mean and circular variance*

Description

The mean direction of a set of directions, the mean resultant length `R` (from 0, directions spread evenly, to 1, all the same), the circular variance `1 - R` and the circular standard deviation `sqrt(-2 log R)` (Mardia and Jupp 2000). Directions are given as angles, or as points seen from an origin (the directions of incidents from an offender's base, say). The mean direction of points depends on the origin chosen. That `R` lies in `[0, 1]` is proved in Lean: see `proof("dirmn")`.

Usage

```
dirmn(
  x = NULL,
  y = NULL,
  origin = c(0, 0),
  w = NULL,
  angle = NULL,
  lonlat = FALSE
)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y` `NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`origin`	For points: the origin `c(x, y)` (in degrees when `lonlat = TRUE`); no point may lie on it.
`w`	`NULL`, or one weight from 0 per direction, positive sum.
`angle`	Instead of points: angles in radians, anticlockwise from east (the x axis).
`lonlat`	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A named numeric vector: `theta` (the mean direction in radians anticlockwise from east, in `[0, 2 pi)`; `NaN` when `R` is 0), `bearing` (the same as a compass bearing, degrees clockwise from north), `r`, `var`, `sd`, `w` (the total weight) and `n`.

References

Mardia, K. V. and Jupp, P. E. (2000). *Directional Statistics*. Wiley. doi:10.1002/9780470316979

Examples

```
dirmn(c(1, 2, 0), c(1, 1, 3), origin = c(0, 0))
dirmn(angle = c(0.1, 0.2, 6.2))
```

`dmlx` *Double machine learning by cross-fitting*

Description

Estimates a causal parameter from orthogonal scores whose nuisance functions (outcome regressions, propensities) are learned out of fold: each fold's predictions come from learners fitted on the other folds, so that the estimate keeps root-n inference while the learners are flexible (Chernozhukov et al. 2018). The learning, the scores, the estimate and its standard error are all compiled; the folds are the caller's, so results can be matched with other software fold for fold.

Usage

```
dmlx(
  y,
  d,
  x,
  model = "plr",
  fold,
  z = NULL,
  s = NULL,
  t = NULL,
  score = NULL,
  lrn = "lasso",
  lrnd = lrn,
  alpha = 0.5,
  nlam = 100,
  nfin = 5,
  tol = 1e-07,
  scr = TRUE,
  hon = FALSE,
  ntree = 500,
  mtry = NULL,
  trim = NULL,
  clu = NULL,
  pred = NULL,
  seed = 1
)
```

Arguments

y	The outcome: one number per row of x (for "did", the change between the periods; for "ssm", NA allowed where s is 0).
d	The treatment: numbers ("plr", "pliv") or 0 and 1.
x	The covariates: a numeric matrix, one row per observation.
model	One of "plr", "pliv", "irm", "iivm", "aipw", "tmle", "did", "didcs", "ssm".
fold	The fold of each row, whole numbers from 1 to the number of folds (at least 2), each used; or a matrix of them, one column per repetition of the cross-fitting.
z	The instrument ("pliv"; 0 and 1 for "iivm"), else NULL.
s	For "ssm", 1 where the outcome is observed and 0 where not.
t	For "didcs", the period of each row, 0 or 1.
score	NULL for the default ("po", "ate" or "obs"), or "po" or "iv" for "plr" and "pliv", "ate" or "atte" for "irm", "obs" or "exp" for "did" and "didcs": the observational score, or the experimental one of 'DoubleML' for 'Python' for a treatment assigned at random (in-sample normalised; nuisances g0, g1 on a panel, the four cell regressions on cross-sections, no propensity).
lrn	The learner of the outcome nuisances: "glm" (least squares; logistic regression for a probability), "lasso", "ridge" or "enet" (the elastic net of enpath() with mixing alpha, the penalty chosen by cross-validation within the training folds), or "forest" (the random forest of rfor() with ntree trees and mtry columns per split, nodes of 5 rows, or 10 for a probability, left unsplit).
lrnd	The learner of the treatment, instrument and selection nuisances, from the same choices.
alpha	The mixing parameter of "enet", from 0 to 1.
nlam	The number of penalties on each elastic net path.
nfin	The number of inner folds that choose the penalty, at least 2.
tol, scr	The convergence threshold and the screening of the elastic net learners, as in enpath().
hon	TRUE for honest forests, on half-samples without replacement with imb = 0.05, as 'grf' grows them (see rfor()).
ntree	The number of trees of "forest": a whole number, at least 1.
mtry	The columns tried at each split of "forest": NULL for the default of rfor(), or a whole number (more than the columns means all of them).
trim	Propensities are kept within trim of 0 and 1: NULL for the default of the reference implementation (1e-12 for "irm", "iivm" and "ssm", as 'DoubleML'; 0.01 for "did" and "didcs", as 'DoubleML' for 'Python'; 0.025 for "aipw", as 'AIPW'; for "tmle", 5 / sqrt(n) / log(n), a lower bound only, as 'tmle').
clu	NULL, or the cluster of each row (whole numbers from 1 to the number of clusters, each used), for "plr" and "pliv"; folds must then be constant within clusters. For two-way clustering, a matrix of two such columns, one per kind of cluster; fold is then a matrix of two columns, the fold of each row's cluster of each kind (constant within it), the test set of each pair of folds is the rows in both and its training set the rows in neither, and the variance is the two-way form of Chiang et al. (2022) as 'DoubleML' computes it.

pred — NULL to learn the nuisances, or their predictions: a matrix with one row per observation and one column per nuisance, in the order listed above (an array with a third dimension for several repetitions). The estimate is then a closed form of these.

seed — The seed of the inner folds and of the forests: one number, at least 0.

Details

The models are those of 'DoubleML', with its scores and its defaults:

- "plr", the partially linear regression y = theta d + g(x) + e, score "po" (partialling out: nuisances l = E[y | x], m = E[d | x]) or "iv" (IV-type: also g, learned on y - theta0 d with theta0 from the partialling-out score);
- "pliv", the partially linear IV model with one instrument z (nuisances l, m = E[z | x], r = E[d | x], and g for "iv");
- "irm", the interactive regression model for a 0/1 treatment, score "ate" or "atte" (nuisances g0, g1, the outcome regressions among the untreated and the treated, and the propensity m);
- "iivm", the interactive IV model (local average effect) for a 0/1 treatment and a 0/1 instrument (g0, g1, m = P(z = 1 | x), r0, r1, the treatment probabilities given z);
- "did", difference in differences on a panel: y is the change in the outcome, d the 0/1 group (nuisances g0, the change among the untreated, and m), with the in-sample normalised observational score of 'DoubleML' for 'Python' (Sant'Anna and Zhao 2020; Chang 2020);
- "didcs", the same on repeated cross-sections, t the 0/1 period (nuisances g00, g01, g10, g11, the outcome regressions in each group and period, and m);
- "ssm", the average effect when the outcome is observed only where s is 1, missing at random given d and x (Bia, Huber and Laffers 2023; nuisances pi = P(s = 1 | d, x), m, g0, g1);
- "aipw" and "tmle", the average effect of a 0/1 treatment from the nuisances of "irm": augmented inverse probability weighting, with the standard error of the 'AIPW' package (the standard deviation of the influence function), and targeted maximum likelihood with the fluctuation of the 'tmle' package (Gruber and van der Laan 2012).

The partialling-out score is Neyman-orthogonal, the AIPW term is unbiased when either nuisance is right, and the estimate is the root of the empirical score: see proof("dmlx").

The estimate solves the mean of the score over all folds (DoubleML's "dml2"); its variance is mean(psi^2) / mean(psi_a)^2 / n, or, with clusters (one-way, models "plr" and "pliv"), the cluster-robust form of Chiang et al. (2022). With several repetitions of the folds (columns of fold), the estimate is the median over repetitions and the variance the median of var + (theta_r - theta)^2.

Value

A list: coef and se, the estimate and its standard error; reps, a data frame with each repetition's coef and se; psia and psib, the score elements (one column per repetition; the estimate solves mean(psia * coef + psib) = 0); pred, the out-of-fold predictions, an array named by nuisance.

References

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W. and Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1), C1-C68. doi:10.1111/ectj.12097

Chiang, H. D., Kato, K., Ma, Y. and Sasaki, Y. (2022). Multiway cluster robust double/debiased machine learning. *Journal of Business and Economic Statistics*, 40(3), 1046-1056. doi:10.1080/07350015.2021.1895815

Sant'Anna, P. H. C. and Zhao, J. (2020). Doubly robust difference-in-differences estimators. *Journal of Econometrics*, 219(1), 101-122. doi:10.1016/j.jeconom.2020.06.003

Chang, N.-C. (2020). Double/debiased machine learning for difference-in-differences models. *The Econometrics Journal*, 23(2), 177-191. doi:10.1093/ectj/utaa001

Bia, M., Huber, M. and Laffers, L. (2023). Double machine learning for sample selection models. *Journal of Business and Economic Statistics*, 42(3), 958-969. doi:10.1080/07350015.2023.2271071

Gruber, S. and van der Laan, M. J. (2012). tmle: an R package for targeted maximum likelihood estimation. *Journal of Statistical Software*, 51(13), 1-35. doi:10.18637/jss.v051.i13

Examples

```
set.seed(1)
n <- 500
x <- matrix(rnorm(n * 5), n)
d <- x[, 1] + rnorm(n)
y <- 0.5 * d + x[, 1] - x[, 2] + rnorm(n)
fold <- sample(rep(1:5, length.out = n))
r <- dmlx(y, d, x, "plr", fold, lrn = "glm", lrnd = "glm")
c(r$coef, r$se)
```

---

dspbm *Benchmark decomposition of a disparity*

Description

Splits each group's disparity in force per resident, against a reference group, into a disparity in contact per resident times a disparity in force per contact. The split is a product, never a sum. See `proof("dspbm")`.

Usage

```
dspbm(pop, contact, force, ref = 1)
```

Arguments

pop	Residents by group: positive numbers, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame.

contact	Police contacts by group: positive numbers, in the forms of pop; named inputs are matched to pop by name, unnamed ones by position (only names protect against a mismatch).
force	Uses of force by group: positive numbers, in the forms of pop, matched by name.
ref	The reference group, by name (matched exactly, case included; a factor by its label) or by position in pop as given.

Value

A data frame with one row per group, in the order of pop: group, ref (TRUE for the reference group), cd (contact per resident), fd (force per contact) and rd (force per resident), each against the reference group, with rd = cd * fd.

Examples

```
dspbm(pop = c(a = 9000, b = 1000), contact = c(a = 900, b = 300),
      force = c(a = 9, b = 6))
```

---

dyadrt *Dyadic offending rates*

Description

Offences by one group against another, as a rate per member of the offending group and as a rate per offender-victim pair exposed. The rate per offender rises with the size of the victim group even when offenders choose victims at random; the rate per pair does not. See proof("dyadrt").

Usage

```
dyadrt(count, pop)
```

Arguments

count	A square matrix (or 2-d table) of offence counts, offending group in rows and victim group in columns: non-negative numbers. When it names both its rows and its columns, the columns are matched to the rows by name, and both to pop when pop is named.
pop	Group sizes: positive numbers, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame. Unnamed, they are taken in the order of the rows: only names protect against a mismatch.

Value

A data frame with one row per dyad: off, vic, count, rtoff, rtpair, nulloff (the rate per offender under random mixing) and ratio (rtpair over the pooled pair rate; 1 under random mixing).

Examples

```
pop <- c(a = 800, b = 200)
dyadrt(count = outer(pop, pop) / 1000, pop = pop)
```

`eceplt` *Empirical cross-entropy plot*

Description

The information a set of likelihood ratios still leaves missing, across prior odds: for the ratios as reported, for the same ratios after calibration, and for a system that always reports 1. Reported ratios above the last curve do harm at that prior. At even prior odds the first two curves pass through the values of `cllrmin()`. `proof()` records no theorem for this plot.

Usage

```
eceplt(same, diff, lo = seq(-2.5, 2.5, by = 0.05), ...)
```

Arguments

`same`	Likelihood ratios from same-source comparisons: positive numbers.
`diff`	Likelihood ratios from different-source comparisons: positive numbers.
`lo`	The prior log10 odds to draw at.
`...`	Passed to `graphics::plot()`.

Value

Invisibly, a data frame: `lo`, `ece`, `min` and `ref`.

References

Ramos, D. and Gonzalez-Rodriguez, J. (2013). Reliable support: measuring calibration of likelihood ratios. *Forensic Science International*, 230(1-3), 156-169. doi:10.1016/j.forsciint.2013.04.014

Examples

```
eceplt(same = c(50, 200, 8, 0.5), diff = c(0.01, 0.2, 1.5, 0.001))
```

enpath *Elastic net by coordinate descent, with cross-validation*

Description

Fits the lasso, ridge and elastic net path by cyclical coordinate descent, for a numeric (gaussian) or a 0/1 (binomial) outcome, and cross-validates it over folds the caller supplies. The objective, the standardisation of the columns and the scale of `lambda` are those of 'glmnet', so that with the same `lambda` and the same folds the two give the same coefficients, up to their convergence thresholds: the gaussian family minimises `(1 / 2n) |y - a0 - x b|^2 + lambda pen(b)`, the binomial family the mean negative log-likelihood plus `lambda pen(b)`, with `pen(b) = sum((1 - alpha) / 2 (s b)^2 + alpha |s b|)`, `s` the population standard deviations of the columns. For the gaussian family the outcome is scaled to unit variance while fitting, as 'glmnet' does. The intercept is never penalised; a constant column keeps a zero coefficient.

Usage

```
enpath(
  x,
  y,
  fam = "gaussian",
  alpha = 1,
  lambda = NULL,
  nlam = 100,
  lmr = NULL,
  fold = NULL,
  tol = 1e-07,
  maxit = 1e+05,
  scr = TRUE
)
```

Arguments

`x`	The predictors: a numeric matrix, one row per observation, with no missing values.
`y`	The outcome: one finite number per row of `x`; 0 or 1 for the binomial family.
`fam`	`"gaussian"` or `"binomial"`.
`alpha`	The mixing parameter, from 0 (ridge) to 1 (lasso).
`lambda`	`NULL`, or the penalties to fit: non-negative numbers in decreasing order.
`nlam`	When `lambda` is `NULL`, the number of penalties on the path: one whole number, at least 1.
`lmr`	When `lambda` is `NULL`, the smallest penalty as a share of the largest: one number strictly between 0 and 1, or `NULL` for 1e-4 when `x` has more rows than columns and 0.01 otherwise. The path runs from the smallest penalty that zeroes every coefficient down to `lmr` times it, evenly spaced on the log scale, and stops

early as 'glmnet' does: from the fifth penalty on, once the share of the deviance explained grows by less than 1e-5 of itself or exceeds 0.999.

`fold` — `NULL`, or the fold of each row: whole numbers from 1 to the number of folds (at least 2), each fold used.

`tol` — The convergence threshold: a pass that moves no standardised coefficient by more than `sqrt(tol)` (weighted by the curvature, for the binomial family) ends the descent. One positive number.

`maxit` — The most passes over the columns, in all: one whole number.

`scr` — `TRUE` to screen the columns of a binomial path by the sequential strong rule of Tibshirani et al. (2012) before each penalty (faster; the optimality conditions are still checked over every column, so the solution does not change), `FALSE` to cycle over the active columns only. Ignored for the gaussian family.

Details

Speed against accuracy: `tol` and `maxit` set how far the descent goes. The default `tol = 1e-7` is glmnet's; `tol = 1e-20, maxit = 1e7` is the exact setting, which reproduces glmnet run at the same threshold to 1e-8 in every coefficient (tests). Between the two, the penalised objective at each penalty differs by less than 1e-6 of its value (tested on the gaussian and binomial paths). `nlam` and `lmr` set the length and the end of a path chosen here, and `scr` the screening of a binomial path.

Each coordinate step is the minimiser of its one-dimensional objective, by soft-thresholding, the proximal map of the absolute value: see `proof("enpath")`.

The binomial fit is a proximal Newton method: iteratively reweighted least squares around coordinate descent over the active columns, with the optimality conditions checked over every column before a `lambda` is left. The gaussian fit updates the gradient through the column inner products, formed only for the columns that enter the model.

Value

A list: `lambda`, the penalties fitted; `a0`, the intercepts; `beta`, a matrix of coefficients with one column per penalty and one row per column of `x`; and, when `fold` is given, `cvm` and `cvsd`, the mean held-out loss (squared error, or the binomial deviance with probabilities kept within 1e-5 of 0 and 1) and its standard error over folds, computed as 'glmnet' computes them, `lmin`, the penalty at the minimum, and `l1se`, the largest penalty within one standard error of it; these four are `NULL` without folds.

References

Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 1-22. doi:10.18637/jss.v033.i01

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J. and Tibshirani, R. J. (2012). Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B*, 74(2), 245-266. doi:10.1111/j.14679868.2011.01004.x

Examples

```
set.seed(1)
x <- matrix(rnorm(200 * 5), 200)
```

```
y <- x[, 1] - 0.5 * x[, 2] + rnorm(200)
f <- enpath(x, y, nlam = 20, fold = rep(1:5, 40))
f$lmin
f$beta[, f$lambda == f$lmin]
```

---

eqproj *Equal-area projection of longitude and latitude, and its inverse*

Description

`eqproj()` maps longitude and latitude to planar metres by the Lambert azimuthal equal-area projection of the sphere about a centre; `eqinv()` maps them back. Areas are exact, so intensities, nearest-neighbour expectations and K functions computed on the result are right; lengths are scaled by `cos(c / 2)` along the radius from the centre and its inverse across it (`c` the central angle), so within 100 km of the centre every length is right to 3.1e-5 relative. Every planar function of the package takes the projected coordinates; those with a `lonlat` argument call this projection themselves.

Usage

```
eqproj(x, y = NULL, lon0 = NULL, lat0 = NULL, r = 6371008.8)

eqinv(x, y = NULL, lon0, lat0, r = 6371008.8)
```

Arguments

`x`, `y`	The points: `x` and `y` numeric vectors, or `x` a matrix or data frame with columns named `lon` and `lat` for `eqproj()` and `x` and `y` for `eqinv()` (an unnamed two-column matrix is read in that order) and `y` `NULL`. Longitudes from -180 to 180, latitudes from -90 to 90; no missing values.
`lon0`, `lat0`	The centre of the projection in degrees. For `eqproj()`, `NULL` (both) takes the circular mean of the longitudes and the mean of the latitudes.
`r`	The radius of the sphere, positive; the distances come back in its unit. The default is the mean Earth radius in metres.

Value

A data frame with one row per point, in input order: `x` and `y` (east and north, in the unit of `r`) from `eqproj()`, with the centre used as the attribute `centre`; `lon` and `lat` from `eqinv()`.

References

Snyder, J. P. (1987). *Map Projections: A Working Manual*. U.S. Geological Survey Professional Paper 1395, chapter 24. doi:10.3133/pp1395

Examples

```
p <- eqproj(c(-79.40, -79.38, -79.35), c(43.64, 43.66, 43.65))
p
a <- attr(p, "centre")
eqinv(p, lon0 = a[1], lat0 = a[2])
```

---

expbnd *Rates and disparities when exposure is measured by a proxy*

Description

Recorded counts (stops, searches, uses of force) are rates per unit of exposure to police, and exposure is seldom observed. If each group's true exposure lies within a factor `gamma` of an observed proxy (residents, drivers on the road, calls for service), each group's true rate lies within a factor `gamma` of its proxy rate, and the ratio of two groups' true rates within a factor `gamma^2` of the proxy ratio. The direction of a disparity is settled only when the proxy ratio lies outside that band. See `proof("expbnd")`.

Usage

```
expbnd(y, m, gamma, ref = 1)
```

Arguments

`y`	Recorded counts by group: positive numbers, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame.
`m`	The exposure proxy by group: positive numbers, in the forms of `y`; named inputs are matched to `y` by name, unnamed ones by position (only names protect against a mismatch).
`gamma`	The factor within which true exposure lies around the proxy: one number, at least 1.
`ref`	The reference group, by name (matched exactly, case included; a factor by its label) or by position in `y` as given.

Value

A data frame with one row per group, in the order of `y`: `group`, `ref` (`TRUE` for the reference group), `rate` (`y / m`), `lo` and `hi` (the bounds on the true rate), `ratio` (the proxy rate against the reference group), `rlo` and `rhi` (the bounds on the true ratio; every end is reached by some admissible exposure), and `sign`: 1 when the true ratio exceeds 1 for every admissible exposure, -1 when it is below 1 for every one, 0 when the data leave it open.

References

Knox, D., Lowe, W. and Mummolo, J. (2020). Administrative records mask racially biased policing. *American Political Science Review*, 114(3), 619-637. doi:10.1017/S0003055420000039

Examples

```
expbnd(y = c(white = 100, black = 300), m = c(1000, 1000), gamma = 1.5)
```

`fair` *Fairness metrics of a binary risk tool by group*

Description

The confusion matrix of a risk score cut at `cut` within each group, and the usual rates computed from it: accuracy, sensitivity (true positive rate), specificity, false positive and false negative rates, positive and negative predictive values, the share and number predicted positive and the Matthews correlation, each with its ratio to the base group and, for the first eight, a Wilson score interval (Wilson 1927). A case is predicted positive when its score exceeds `cut`. No single metric defines fairness, and when base rates differ between groups the error rates and the predictive values cannot all be equal at once (Chouldechova 2017); the table shows which differ, not which should not.

Usage

```
fair(score, y, grp, cut = 0.5, base = 1, conf = 0.95)
```

Arguments

`score`	Risk scores: finite numbers, one per case.
`y`	Outcomes: 0 or 1 (or `FALSE`/`TRUE`), one per case.
`grp`	Group labels: numbers, text or a factor, one per case, no missing values. Groups are ordered as the factor's levels, or sorted.
`cut`	The cut: one number.
`base`	The group the ratios are taken against: its label, or its position from 1.
`conf`	The confidence level of the intervals, strictly between 0 and 1.

Value

A list of data frames, one row per group with its label in `group`: `cnt` (`tp`, `fp`, `tn`, `fn`, `n`), `met` (`acc`, `tpr`, `tnr`, `fpr`, `fnr`, `ppv`, `npv`, `ppos`, `npos`, `mcc`; `NaN` where a rate has an empty denominator), `rat` (each metric over the base group's), `lo` and `hi` (the interval ends of the first eight metrics; the last two columns `NaN`).

References

Wilson, E. B. (1927). Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158), 209-212. doi:10.1080/01621459.1927.10502953

Chouldechova, A. (2017). Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. *Big Data*, 5(2), 153-163. doi:10.1089/big.2016.0047

Examples

```
set.seed(1)
g <- rep(c("a", "b"), each = 100)
y <- rbinom(200, 1, ifelse(g == "a", 0.3, 0.5))
s <- plogis(rnorm(200, ifelse(y == 1, 0.8, -0.8)))
fair(s, y, g)$met
```

`fairdf` *Differences between groups in a risk tool's decisions*

Description

How the share of cases a tool flags (predicts positive) differs between groups, as a difference and as a ratio against a base group, each with an interval; with the outcomes, how its true and false positive rates differ; and the bias amplification score, the largest parity gap times the Gini coefficient of the groups' shares.

Usage

```
fairdf(
  pred,
  grp = NULL,
  y = NULL,
  base = NULL,
  conf = 0.95,
  nboot = 0,
  seed = NULL
)
```

Arguments

`pred`	The tool's decisions: 1 (or `TRUE`) for a case it flags, 0 (or `FALSE`) otherwise, one per case. A `clpre` list from `clprep()` is taken too: its elements fill the arguments by name.
`grp`	Group labels: numbers, text or a factor, one per case, no missing values. Groups are ordered as the factor's levels, or sorted.
`y`	`NULL`, or the outcomes: 0 or 1 (or `FALSE`/`TRUE`), one per case.
`base`	The base group: `NULL` for the group with the largest share flagged (the first in order on ties), its label, or its position from 1.
`conf`	The confidence level of the intervals, strictly between 0 and 1.
`nboot`	The bootstrap replicates for the intervals of the Gini coefficient and the score: a whole number from 0 (none).
`seed`	`NULL` to draw the seed of the bootstrap (or the permutations) from R's generator, so that `set.seed()` fixes it, or one whole number from 0.

Details

`fair()` reports each group's rates and their ratios; `fairdf()` reports the comparisons the fairness literature names. The parity gap `dp` is a group's share flagged minus the base group's (demographic parity holds when every gap is 0), with Newcombe's (1998) hybrid score interval. The ratio `di` is the disparate impact ratio, with the log interval of Katz et al. (1978); under the four-fifths rule a ratio below 0.8 against the group with the highest selection rate is the usual sign of adverse impact, so by default that group is the base. With the outcomes `y`, `dtpr` and `dfpr` are the gaps in true and false positive rates (both 0 is equalized odds), `aod` their mean (the average odds difference) with the interval of Zou and Donner (2008) built from the two Newcombe intervals, and `eo` the larger of the two gaps in size. The bias amplification score of Barman and Barman (2026) is large only when a sizeable gap comes with unequal shares across all the groups; its bootstrap interval, and the Gini coefficient's, are percentile intervals from redrawing each group's decisions with replacement (Efron and Tibshirani 1993), the base group held.

These numbers describe what the tool did to each group. They do not say why: a gap in shares flagged can come from a gap in the outcomes the tool was built to predict, from how those outcomes were recorded, or from the tool; and when the groups' base rates differ, equal error rates and equal predictive values cannot all hold at once (Chouldechova 2017, see `impfpr()`). A ratio's interval is `NaN` when either share is 0, and a ratio against a base group with no case flagged is `Inf` (or `NaN` for 0 over 0).

Value

A list: `tab`, a data frame with one row per group (in the order of `grp`): `group`, `n`, `npos` (cases flagged), `ppos` (their share) with `lo` and `hi`, `dp` with `dplo` and `dphi`, `di` with `dilo` and `dihi`, and with `y` also `tpr`, `fpr`, `dtpr` (`dtprlo`, `dtprhi`), `dfpr` (`dfprlo`, `dfprhi`), `aod` (`aodlo`, `aodhi`) and `eo`; the base group's own intervals are `NaN`. `sum`, a named vector: `di` (the smallest finite ratio of another group), `dp` (the gap of largest size, signed), `gini`, `bas`, `ginilo`, `ginihi`, `baslo`, `bashi` (`NaN` when `nboot` is 0) and with `y` also `aod` and `eo` (those of largest size, signed). `base`, the base group's label.

References

Newcombe, R. G. (1998). Interval estimation for the difference between independent proportions: comparison of eleven methods. *Statistics in Medicine*, 17(8), 873-890. doi:10.1002/(SICI)1097-0258(19980430)17:8<873::AIDSIM779>3.0.CO;2I

Katz, D., Baptista, J., Azen, S. P. and Pike, M. C. (1978). Obtaining confidence intervals for the risk ratio in cohort studies. *Biometrics*, 34(3), 469-474. doi:10.2307/2530610

Zou, G. Y. and Donner, A. (2008). Construction of confidence limits about effect measures: a general approach. *Statistics in Medicine*, 27(10), 1693-1702. doi:10.1002/sim.3095

Barman, P. K. and Barman, P. K. (2026). Spatial disparities in simulated patrol detection: a reproducible generative-model audit across Baltimore and Chicago. doi:10.48550/arXiv.2603.18987

Efron, B. and Tibshirani, R. J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall. doi:10.1007/9781489945419

Examples

```
set.seed(1)
g <- rep(c("a", "b"), each = 200)
```

```
y <- rbinom(400, 1, 0.3)
p <- rbinom(400, 1, ifelse(g == "a", 0.4, 0.25))
r <- fairdf(p, g, y, nboot = 199, seed = 1)
r$tab[, c("group", "ppos", "dp", "dplo", "dphi", "di", "dilo", "dihi")]
r$sum
```

`fairsim` *Simulated records with a known disparity*

Description

Incidents in segregated areas, whose true occurrence is the same for every group but whose recording, and the risk an algorithm predicts for them, differ by group by a set amount: data on which an audit's answer is known (after Barman and Barman 2026). The first group is the reference.

Usage

```
fairsim(
  n = 2000,
  groups = c("A", "B"),
  props = NULL,
  nareas = 20,
  base = 0.3,
  bias = 0.5,
  seed = NULL
)
```

Arguments

`n`	The number of records: a whole number from 1.
`groups`	The group labels, at least two, the reference first.
`props`	`NULL` for equal shares, or one weight from 0 per group (not all 0).
`nareas`	The number of areas: a whole number, at least the number of groups.
`base`	The true rate, and the reference group's recorded rate: one number from 0 to 1.
`bias`	The disparity injected: one number from -1 to 1.
`seed`	`NULL` to draw the seed from R's generator, or one whole number from 0; the records come from crimlab's own generator, the same on every platform.

Details

Each record draws its group by `props`, then an area among those its group lives in (with g groups, areas 1, g + 1, `2 g + 1`, ... hold the first group only, areas 2, g + 2, ... the second, and so on), then `truth`, 1 with probability `base` whatever the group; `det`, the recorded outcome, 1 with probability `base` in the reference group and `base (1 - bias)` in the others (clamped to 0 and 1), so that the other groups' incidents are recorded less when `bias` is positive; and `risk`, normal with mean 250 in the reference group and `250 + 100 bias` in the others and standard deviation 40, clamped to 0 and 500. So `fairdf()` on `det` should find a ratio near `1 - bias`, `scrdsp()` on `risk` a gap near `100 bias`, and `calaud()` the other groups' areas ranked as riskier than their recorded outcomes.

Value

A data frame with one row per record: `area` (a number from 1), `group`, `truth`, `det` and `risk`.

References

Barman, P. K. and Barman, P. K. (2026). Spatial disparities in simulated patrol detection: a reproducible generative-model audit across Baltimore and Chicago. doi:10.48550/arXiv.2603.18987

Examples

```
d <- fairsim(2000, seed = 1)
head(d)
# the injected ratio is 1 - 0.5
fairdf(d$det, d$group, base = "A")$tab[, c("group", "di", "dilo", "dihi")]
```

`fzmode` *Fuzzy mode*

Description

For each point, the number of points (itself and coincident ones included) within distance `r` of it. With `r = 0` it counts coincident points, the mode of exact locations; with `r > 0` it is the fuzzy mode of the crime-mapping literature, a fixed-radius scan centred on each incident. Counts of nearby points overlap by design: each point counts its own neighbourhood.

Usage

```
fzmode(x, y = NULL, r, count = NULL, metric = "euclid", lonlat = FALSE)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`r`	The radius: one number from 0, in the unit of the coordinates (metres when `lonlat = TRUE`). Distances up to and including `r` count.
`count`	`NULL` for points, or for zones: one whole count from 0 per zone (`x`, `y` the zones' centroids); the result is then each zone's total count within `r` of it, its own included.
`metric`	The distance: `"euclid"` or `"manhattan"` (`|dx| + |dy|`), as in `spwt()` and `jtsurf()`.

`lonlat` — `FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

An integer vector, one count per point, in input order (with `count`, a numeric vector of zone totals).

References

Eck, J. E., Chainey, S., Cameron, J. G., Leitner, M. and Wilson, R. E. (2005). *Mapping Crime: Understanding Hot Spots*. National Institute of Justice, NCJ 209393.

Examples

```
fzmode(c(0, 0, 1, 5), c(0, 0, 0, 5), r = 1)
```

---

gearc *Geary's C*

Description

Geary's (1954) contiguity ratio of values on units with spatial weights `w`: `C = (n - 1) sum w_ij (x_i - x_j)^2 / (2 S0 sum` 1 under no autocorrelation, below 1 when neighbours are alike. Its variance is given under normality and under randomisation (Cliff and Ord 1981), with z statistics `(1 - C) / sd` and their upper normal tails, and, with `nsim > 0`, a permutation p-value against positive autocorrelation. That C has expectation 1 under randomisation is proved in Lean: see `proof("gearc")`.

Usage

```
gearc(x, w, nsim = 0, seed = NULL)
```

Arguments

`x`	The values: finite numbers, one per unit.
`w`	The weights: `spwt()`'s list, or an spdep `listw` object, for as many units as `x`.
`nsim`	The number of permutations: a whole number from 0.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.

Value

A named numeric vector: `c`, `vn` and `vr` (its variance under normality and randomisation), `zn`, `zr`, `pn`, `pr`, `pperm` (`(1 + #{C* <= C}) / (nsim + 1)`, `NaN` without permutations) and `s0` (the sum of the weights).

References

Geary, R. C. (1954). The contiguity ratio and statistical mapping. *The Incorporated Statistician*, 5(3), 115-145. doi:10.2307/2986645

Cliff, A. D. and Ord, J. K. (1981). *Spatial Processes: Models and Applications*. Pion, London.

Examples

```
set.seed(1)
p <- cbind(x = runif(50), y = runif(50))
gearc(p[, 1] + p[, 2] + rnorm(50, sd = 0.1), spwt(p, k = 4))
```

`gorg` *The general G of Getis and Ord*

Description

The global concentration statistic of Getis and Ord (1992) for non-negative values: `G = sum_ij w_ij x_i x_j / sum_{i != j` its expectation `S0 / (n (n - 1))` and variance under randomisation, the z statistic, its upper normal tail (high values near high values) and, with `nsim > 0`, a permutation p-value. Unlike Moran's I and Geary's C it tells high-value clusters from low-value ones. Its expectation under randomisation is proved in Lean: see `proof("gorg")`.

Usage

```
gorg(x, w, nsim = 0, seed = NULL)
```

Arguments

`x`	The values: finite numbers from 0, at least two positive.
`w`	The weights: `spwt()`'s list, or an spdep `listw` object, for as many units as `x`.
`nsim`	The number of permutations: a whole number from 0.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.

Value

A named numeric vector: `g`, `e`, `v`, `z`, `p` and `pperm` (`(1 + #{G* >= G}) / (nsim + 1)`).

References

Getis, A. and Ord, J. K. (1992). The analysis of spatial association by use of distance statistics. *Geographical Analysis*, 24(3), 189-206. doi:10.1111/j.15384632.1992.tb00261.x

Examples

```
set.seed(1)
p <- cbind(x = runif(50), y = runif(50))
gorg(exp(3 * p[, 1]), spwt(p, type = "band", d = 0.2, style = "raw"))
```

grprr *Rates and proportions by group: ratios and differences*

Description

The rate (events over exposure) or the proportion (events over trials) of each group, and its ratio to and difference from a reference group, with approximate and exact intervals. Rows that share a group are pooled: give one row per group, or one per case (0/1 outcomes with `n = 1`, the default for proportions), or anything between.

Usage

```
grprr(
  x,
  n = NULL,
  grp = NULL,
  ref = NULL,
  type = "rate",
  per = 1,
  level = 0.95
)
```

Arguments

`x`	The events: numbers from 0; or a `clpre` list from `clprep()`.
`n`	The exposure (person-time, population) of each row for rates, or the trials for proportions: positive numbers; `NULL` for one trial per row (proportions only).
`grp`	`NULL` (each row is a group, named by the names of `x`) or the group of each row: labels.
`ref`	The reference group: a label or a position; the first group by default.
`type`	`"rate"` (events over exposure) or `"prop"` (a proportion).
`per`	The rates and differences are given per `per` units of `n` (1e5 for rates per 100,000, 100 for percentages): one positive number.
`level`	The confidence level of the intervals: one number strictly between 0 and 1.

Details

For a rate `x / n` the approximate interval is `rate * exp(-+z / sqrt(x))` and the exact one the Poisson interval through the gamma quantiles, as `stats::poisson.test()`; the ratio's approximate interval is log-normal with variance `1 / x + 1 / x0`, its exact interval the one conditional on the total count (the Clopper-Pearson interval of `x / (x + x0)`, as the two-sample `stats::poisson.test()`), and the difference's is Wald. For a proportion the intervals are Wilson's score interval and the exact Clopper-Pearson one (`stats::binom.test()`); the ratio's is log-normal (Katz et al.), with no exact interval (`NA`), and the difference's is Newcombe's hybrid score interval. With a zero count the log-normal intervals are `NA`: no continuity correction is made.

`rep` is a group's share of the events over its share of the exposure (or the trials): over 1 when the group is over-represented among the events, as in a comparison of who is placed in segregation with who is in custody.

Value

An object of class `grprr`, a list: `rates`, a data frame with one row per group (`group`, `x`, `n`, `rate`, `lo`, `hi` the approximate interval, `elo`, `ehi` the exact one, `rep`); `cmp`, one row per group but the reference (`group`, `ratio`, `lo`, `hi`, `elo`, `ehi`, `diff`, `dlo`, `dhi`); `ref`, `type`, `per` and `level`.

References

Katz, D., Baptista, J., Azen, S. P. and Pike, M. C. (1978). Obtaining confidence intervals for the risk ratio in cohort studies. *Biometrics*, 34(3), 469-474. doi:10.2307/2530610

Newcombe, R. G. (1998). Interval estimation for the difference between independent proportions: comparison of eleven methods. *Statistics in Medicine*, 17(8), 873-890. doi:10.1002/(SICI)1097-0258(19980430)17:8<873::AIDSIM779>3.0.CO;2I

Examples

```
# use-of-force incidents per 1,000 contacts by perceived race
grprr(c(white = 120, black = 95, other = 30),
      c(white = 60000, black = 21000, other = 15000), per = 1000)
# proportions from one row per case
grprr(c(1, 0, 0, 1, 1, 0, 1, 1), grp = c(1, 1, 1, 1, 2, 2, 2, 2),
      type = "prop")
```

---

gstr *Local Getis-Ord Gi* on a grid of counts*

Description

The local statistic Gi* of Getis and Ord (1992; Ord and Getis 1995) for each cell of a grid of counts, with binary weights on the cells within `rad` cell widths (`rad = 1.5`: the 3 by 3 block, cell included), its normal p-value and Benjamini-Hochberg q-value, a conditional permutation p-value, and an exposure-aware version: given the total, the count of a neighbourhood is binomial with the neighbourhood's share of the exposure under complete spatial randomness, so cells overhanging the study area neither inflate the mean nor get flagged. No threshold is applied.

Usage

```
gstr(cnt, ex = NULL, rad = 1.5, nsim = 0, seed = NULL)
```

Arguments

cnt	Counts per cell: a matrix of whole numbers from 0.
ex	NULL for equal exposure, or a matrix of the same dimensions: each cell's exposure (its area inside the study window, say), 0 for a cell outside (its results are NaN).
rad	The neighbourhood radius in cell widths, from 0 to 1000 (1: the cell and its four rook neighbours; 1.5: the queen neighbourhood).
nsim	The number of permutations: a whole number from 0.
seed	NULL to draw the simulations' seed from R's generator (so set.seed() fixes them), or one whole number from 0.

Value

A data frame with one row per cell (down the columns of cnt): row, col, gz (the classic Gi* z), pz, qz, pp (the permutation p-value, NaN without permutations), ze (the exposure-aware z), pe (its exact two-sided binomial p-value) and qe.

References

Getis, A. and Ord, J. K. (1992). The analysis of spatial association by use of distance statistics. *Geographical Analysis*, 24(3), 189-206. doi:10.1111/j.15384632.1992.tb00261.x

Ord, J. K. and Getis, A. (1995). Local spatial autocorrelation statistics: distributional issues and an application. *Geographical Analysis*, 27(4), 286-306. doi:10.1111/j.15384632.1995.tb00912.x

Examples

```
set.seed(1)
m <- matrix(rpois(100, 2), 10)
m[4:5, 4:5] <- 9
g <- gstr(m)
g[g$qe < 0.05, ]
```

---

gtmp *Genotype match probability*

Description

The chance that an unrelated person from the same population shares a DNA profile, locus by locus and overall. With theta = 0 this is the product rule (p^2 for a homozygote, 2pq for a heterozygote). A positive theta applies the Balding-Nichols correction for shared ancestry. That never makes a homozygote match rarer than the product rule says; it can make a heterozygote match rarer when both alleles are common. See proof("gtmp").

Usage

```
gtmp(p, q = NULL, hom = FALSE, theta = 0)
```

Arguments

p	Frequency of the first allele at each locus: numbers above 0 and at most 1, optionally named by locus, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame.
q	Frequency of the second allele at each locus; not read where hom is TRUE. Named inputs are matched to p by locus name, unnamed ones by position (only names protect against a mismatch).
hom	TRUE where the genotype is homozygous: one value, or one per locus, matched to p by name when both are named.
theta	The coancestry coefficient: one number from 0 up to, not including, 1.

Value

A list: locus, a data frame with locus, hom and mp; log10mp, the base-10 logarithm of the profile match probability; and log10lr, the same with the sign changed, the weight of a single-source match against an unrelated alternative.

References

Balding, D. J. and Nichols, R. A. (1994). DNA profile match probability calculation. *Forensic Science International*, 64(2-3), 125-140. doi:10.1016/03790738(94)902224

Examples

```
gtmp(p = c(D3 = 0.1, vWA = 0.2), q = c(0.3, NA), hom = c(FALSE, TRUE),
     theta = 0.03)
```

---

hdbang *Head-bang smoothing*

Description

Smooths values at points or areal units (rates by area centroid, say) by Hansen's (1991) head-bang rule, which removes isolated peaks and troughs while keeping edges between regions of different level. For each unit i the rule takes its nn nearest neighbours and every pair (j, k) of them whose angle at i is at least ang degrees, so that i lies roughly between j and k (a triple); with ntrip > 0 only the ntrip triples whose ends are nearest (smallest |p_j - p_i| + |p_k - p_i|) are kept. Each triple gives a low value min(v_j, v_k) and a high value max(v_j, v_k); the low screen is the median of the lows, the high screen the median of the highs, and the new value is the median of the low screen, v_i and the high screen, so v_i moves only if it lies outside both. With weights w (populations at risk, say) the screens are weighted medians, each end weighted by its own unit's weight, after Mungiole, Pickle and Simonson (1999): the smallest value whose cumulative weight reaches half the total. A unit with no triple (on the outer edge of the points) keeps its value; Hansen's extrapolated end triples are not formed. Each of iter passes starts from the previous pass's values.

Usage

```
hdbang(
  x,
  y = NULL,
  v,
  w = NULL,
  nn = 8,
  ang = 135,
  ntrip = 0,
  iter = 1,
  lonlat = FALSE
)
```

Arguments

x, y	The points: x and y numeric vectors of equal length; or x a matrix or data frame with columns named x and y (or X and Y, as sf::st_coordinates() names them, or lon and lat with lonlat = TRUE) and y NULL; or x an sf or sfc object of POINT geometries, whose coordinate reference system must agree with lonlat. An unnamed two-column matrix is read as x then y; other columns are never taken by position. Vectors given separately (x, y, and w or t where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
v	The values to smooth: finite numbers, one per point.
w	NULL, or positive weights, one per point.
nn	The number of nearest neighbours searched: from 2 to the number of points less 1.
ang	The smallest angle at i of a triple, in degrees, from 90 to 180 (135 by convention).
ntrip	The most triples used per unit, nearest first; 0 for all.
iter	The number of passes: a whole number from 0.
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A numeric vector of smoothed values, one per point, in input order.

References

Hansen, K. M. (1991). Head-banging: robust smoothing in the plane. *IEEE Transactions on Geoscience and Remote Sensing*, 29(3), 369-378.

Mungiole, M., Pickle, L. W. and Simonson, K. H. (1999). Application of a weighted head-banging algorithm to mortality data maps. *Statistics in Medicine*, 18(23), 3201-3209.

Examples

```
g <- expand.grid(x = 1:6, y = 1:6)
v <- rep(1, 36)
v[15] <- 50
hdbang(g, v = v)[15]
```

`hvdist` *Great-circle distance*

Description

The distance along the sphere between pairs of points given in longitude and latitude. The central angle is computed as the `atan2` of the cross and dot products of the two unit vectors, written so that nothing cancels: accurate to a few units in the last place from coincident points to antipodes (the spherical case of Vincenty 1975). The Earth is taken as a sphere of mean radius `r`; distances on the ellipsoid differ by up to about 0.5 per cent.

Usage

```
hvdist(from, to, r = 6371008.8)
```

Arguments

`from`, `to`	The two ends of each pair, in degrees: a matrix or data frame with columns `lon` and `lat` (an unnamed two-column matrix is read as longitude then latitude), or one point as `c(lon, lat)`. Both have the same number of points, or one of them has one point, which is paired with every point of the other. Longitudes from -180 to 180, latitudes from -90 to 90; no missing values.
`r`	The radius of the sphere, positive; the distances come back in its unit. The default is the mean Earth radius in metres.

Value

A numeric vector of distances, one per pair, in the order of the input.

References

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 23(176), 88-93. doi:10.1179/sre.1975.23.176.88

See Also

`eqproj()` to project longitude and latitude to metres.

Examples

```
# Toronto to Montreal, metres
hvdist(c(-79.38, 43.65), c(-73.57, 45.50))
pts <- data.frame(lon = c(0, 90, 180), lat = c(0, 0, 0))
hvdist(pts, c(0, 0))
```

`hzdsel` *Period-two hazards after selective depletion*

Description

A share `s` of a cohort is at high risk of reconviction, `h` a period, and the rest at low risk, `l`. If a programme changes who survives the first period without reconviction, the period-two hazard among the survivors is a weighted average of `h` and `l` with weight the share of high-risk survivors, and it rises with that share. An arm that keeps more of its high-risk people out of reconviction in period one therefore shows the higher period-two hazard even if it has no effect on anyone in period two: a period-by-period hazard ratio is not a causal contrast; a risk difference at a fixed follow-up is. See `proof("hzdsel")`.

Usage

```
hzdsel(s, h, l, surh, surl)
```

Arguments

`s`	The high-risk share at the start, strictly between 0 and 1.
`h`	The period hazard of the high-risk type: at most 1.
`l`	The period hazard of the low-risk type: from 0, below `h`.
`surh`	Period-one survival of the high-risk type in the control and treated arms: two numbers above 0 and at most 1, the control arm first, or named `control` and `treated` in either order (with `surl` named too). A list of single numbers or a one-row data frame is accepted.
`surl`	Period-one survival of the low-risk type, the same way; named inputs are matched to `surh` by name.

Value

A named numeric vector: `share0`, `share1` (the high-risk share among the survivors of each arm), `haz0`, `haz1` (the period-two hazard in each arm) and `ratio` (`haz1 / haz0`).

References

Hernan, M. A. (2010). The hazards of hazard ratios. *Epidemiology*, 21(1), 13-15. doi:10.1097/EDE.0b013e3181c1ea43

Examples

```
hzdsel(s = 0.5, h = 0.5, l = 0.1, surh = c(0.5, 0.8), surl = c(0.9, 0.9))
```

`impfpr` *The false positive rate the other rates imply*

Description

Chouldechova's (2017) identity: in any confusion table, the false positive rate is `p / (1 - p) (1 - ppv) / ppv (1 - fnr)`, from the base rate `p`, the positive predictive value `ppv` and the false negative rate `fnr`. So two groups whose scores are equally calibrated (the same `ppv`) and miss equally (the same `fnr`) but whose base rates differ must have different false positive rates: no score, however it is built, can equalise all three. The identity is arithmetic, true of any tool on any data; it says which equalities cannot hold together, not which one should give way. See `proof("impfpr")`.

Usage

```
impfpr(p, ppv, fnr)
```

Arguments

`p`	Base rates, strictly between 0 and 1.
`ppv`	Positive predictive values, above 0 and at most 1.
`fnr`	False negative rates, from 0 and below 1. The three have one length (or length 1); named ones are matched by name.

Value

The false positive rates, one per triple.

References

Chouldechova, A. (2017). Fair prediction with disparate impact: a study of bias in recidivism prediction instruments. *Big Data*, 5(2), 153-163. doi:10.1089/big.2016.0047

Examples

```
# the same ppv and fnr, different base rates: different fpr
impfpr(p = c(a = 0.3, b = 0.5), ppv = 0.6, fnr = 0.25)
```

incap *Incapacitation: crime rate, share prevented and the marginal year*

Description

An offender commits crimes at rate `lam` a year while free, and each crime leads to a custodial sentence of `s` years with probability `q`. Over a career the crime rate is `lam / (1 + lam q s)`: the share of offending prevented, `lam q s / (1 + lam q s)`, never reaches 1, longer or surer sentences lower the rate, and one more year of sentence prevents `lam^2 q / ((1 + lam q s) (1 + lam q (s + 1)))` crimes a year, less for each year added. A sentence uniform across offenders prevents a larger share of the offending of those with the higher rate. The model holds the rate fixed over the career and replaces nobody; see `incapr()` for both. See `proof("incap")`.

Usage

```
incap(lam, q, s, share = NULL)
```

Arguments

`lam`	Offending rates while free (crimes a year): positive numbers, one per group, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame.
`q`	The chance a crime leads to a custodial sentence: numbers in the interval from 0 (excluded) to 1, recycled to the length of `lam`; one per group, in the forms of `lam` and matched to it by name when both are named (unnamed, by position: only names protect against a mismatch).
`s`	Sentence lengths in years: non-negative numbers, recycled; one per group, in the forms of `lam`, matched by name.
`share`	`NULL`, or one non-negative population share per group, summing to 1, in the forms of `lam`, matched by name.

Value

A data frame with one row per group: `group`, `lam`, `q`, `s`, `rate` (the long-run crime rate), `prev` (the share prevented) and `marg` (the crimes a year one more year of sentence prevents). With `share`, a last row `all` holds the weighted free rate in `lam`, the weighted rate in `rate` and the share of all offending prevented in `prev`.

References

Avi-Itzhak, B. and Shinnar, R. (1973). Quantitative models in crime control. *Journal of Criminal Justice*, 1(3), 185-217. doi:10.1016/00472352(73)900615

Examples

```
incap(lam = c(low = 2, high = 10), q = 0.1, s = 1, share = c(0.8, 0.2))
```

`incapr` *Incapacitation under desistance and replacement*

Description

Crimes prevented by a sentence of `s` periods served from career age `t0`, when the offending rate falls with career age and a share `r` of the crimes prevented is committed by someone else instead. The count is at most `s` times the rate at the start of the sentence and at least `s` times the rate at its end; the same sentence served one period later prevents no more; and, net of replacement, the estimate from the rate at the start is still an upper bound. The usual constant-rate estimate therefore overstates what a sentence buys under both assumptions. See `proof("incapr")`.

Usage

```
incapr(lam, t0, s, r = 0)
```

Arguments

`lam`	Offending rates by career age, `lam[1]` at age 0: non-negative numbers that never increase.
`t0`	The career age at which the sentence starts: a whole number from 0, with `t0 + s` at most `length(lam) - 1`.
`s`	The sentence in periods: a whole number from 1.
`r`	The replacement share: one number from 0 to 1.

Value

A named numeric vector: `prev` (crimes prevented), `net` (net of replacement), `upper` (`s * lam` at age `t0`), `lower` (`s * lam` at age `t0 + s`) and `later` (crimes prevented by the same sentence from age `t0 + 1`).

References

Shinnar, S. and Shinnar, R. (1975). The effects of the criminal justice system on the control of crime: a quantitative approach. *Law and Society Review*, 9(4), 581-611. doi:10.2307/3053340

Examples

```
incapr(lam = c(12, 10, 8, 6, 5, 4, 3, 2, 2, 1), t0 = 2, s = 3, r = 0.25)
```

`jtacc` *Accuracy of a journey-to-crime surface on a solved series*

Description

Given a surface over the grid of `jtsurf()` and the offender's known base: the surface's value in the base's cell, the hit score (the share of the grid searched, in decreasing order of the surface, up to and including the base's cell), in the best and the worst order of ties (Rossmo 2000; Canter et al. 2000), and the error distance from the peak cell's centre to the base.

Usage

```
jtacc(prob, win, base)
```

Arguments

`prob`	The surface: a matrix of values from 0, `nx` rows by `ny` columns, as `jtsurf()` returns.
`win`	The grid's rectangle, as given to `jtsurf()`.
`base`	The known base, `c(x, y)`, inside `win`.

Value

A named numeric vector: `p` (the value in the base's cell), `hitbest` and `hitworst` (shares from 0 to 1), `err` (the error distance), `peak` and `cell` (the peak's and the base's cell, numbered down the columns).

References

Rossmo, D. K. (2000). *Geographic Profiling*. CRC Press.

Examples

```
s <- jtsurf(c(2, 3, 4), c(2, 4, 3), win = c(0, 6, 0, 6), nx = 30,
            ny = 30, type = "negexp", par = c(1, 1), mode = "lik")
jtacc(s$prob, win = c(0, 6, 0, 6), base = c(3.2, 2.9))
```

`jtcal` *Empirical distance decay from solved trips*

Description

A kernel density estimate of trip lengths from known offender trips, to calibrate a decay function (Levine 2009; Silverman 1986). With `refl = TRUE` the kernels are reflected at 0, so no mass falls below 0 and the estimate integrates to 1 over `[0, Inf)`. Tabulated at evenly spaced distances it is a `"table"` decay for `jtdecay()` and `jtsurf()`.

Usage

```
jtcal(d, t, h, w = NULL, kern = "normal", refl = TRUE)
```

Arguments

`d`	Trip lengths: numbers from 0.
`t`	The distances to evaluate at (from 0 when `refl = TRUE`).
`h`	The bandwidth: one positive number.
`w`	`NULL` for equal weights, or one weight from 0 per trip, with a positive sum.
`kern`	The kernel: `"normal"`, `"quartic"`, `"triangular"`, `"uniform"` or `"laplace"` (double exponential).
`refl`	`TRUE` to reflect at 0.

Value

A numeric vector of densities, one per value of `t`.

References

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall. doi:10.1201/9781315140919

Levine, N. (2009). Introduction to the special issue on Bayesian journey-to-crime modelling. *Journal of Investigative Psychology and Offender Profiling*, 6(3), 167-185. doi:10.1002/jip.107

Examples

```
set.seed(1)
trips <- rexp(200, 1 / 2)
jtcal(trips, t = seq(0, 10, 0.5), h = 0.5)
```

`jtdecay` *Distance-decay functions for journey to crime*

Description

The decay of the chance of offending with distance from an offender's base, in the forms of Brantingham and Brantingham (1981), Rossmo (2000) and Levine (2009): `"linear"`, `f = max(0, A + B d)`, `par = c(A, B)`; `"negexp"`, `f = A exp(-b d)`, `par = c(A, b)` with `b >= 0`; `"normal"`, `f = A dnorm(d, mu, s)`, `par = c(A, mu, s)`; `"lognormal"`, `f = A dlnorm(d, mu, s)`, `par = c(A, mu, s)` (the lognormal density of `d`; some texts use `log d^2` and `1 / d^2`, which is not a density); `"trunc"`, the truncated negative exponential, `f = P d / dp` up to the peak `dp` and `P exp(-C (d - dp))` beyond, `par = c(P, dp, C)`; and `"table"`, values `tab` at distances `0, delta, 2 delta, ...`, linear in between and 0 beyond, `par = delta` (as `jtcal()` tabulates).

Usage

```
jtdecay(d, type, par, tab = NULL)
```

Arguments

`d`	Distances: numbers from 0.
`type`	The form: one of the names above.
`par`	The parameters of the form, of the length given above.
`tab`	For `type = "table"`: at least two values from 0.

Value

A numeric vector of `f(d)`, one per distance.

References

Brantingham, P. J. and Brantingham, P. L. (1981). *Environmental Criminology*. Sage.

Levine, N. (2009). Introduction to the special issue on Bayesian journey-to-crime modelling. *Journal of Investigative Psychology and Offender Profiling*, 6(3), 167-185. doi:10.1002/jip.107

Examples

```
jtdecay(c(0, 0.5, 1, 2, 5), type = "trunc", par = c(1, 0.5, 0.8))
```

`jtsurf` *Journey-to-crime surface*

Description

Ranks the cells of a grid as candidate bases of an offender from the locations of a series of incidents. `mode = "sum"` gives the classic aggregate surface, the sum over incidents of the decay of their distance to the cell (Canter et al. 2000; Levine 2009); `mode = "lik"` gives the sum of the logarithms, the log-likelihood of the cell as base when the decay function is the planar density of an incident about the base and the incidents are independent (O'Leary 2009), so that `prob` is the posterior over cells under a uniform prior. The sum surface is not a likelihood.

Usage

```
jtsurf(
  x,
  y = NULL,
  win,
  nx,
  ny,
  type,
  par,
  tab = NULL,
  metric = "euclid",
  mode = "sum"
)
```

Arguments

`x, y`	The points in planar coordinates (project longitude and latitude with `eqproj()` first): `x` and `y` numeric vectors of equal length, or `x` a matrix or data frame with columns named `x` and `y` (an unnamed two-column matrix is read as `x` then `y`). At least two points, all inside `win`; no missing values.
`win`	The grid's rectangle `c(xmin, xmax, ymin, ymax)`.
`nx, ny`	The numbers of cells across and up: whole numbers from 1, with at most 1e9 cells.
`type`	The form: one of the names above.
`par`	The parameters of the form, of the length given above.
`tab`	For `type = "table"`: at least two values from 0.
`metric`	`"euclid"` (straight-line distance) or `"manhattan"` (`|dx| + |dy|`, a street grid aligned with the axes).
`mode`	`"sum"` or `"lik"`.

Details

What this does not license: the surface ranks places to search under the assumptions that the series is one offender's, that the offender travelled from one fixed base inside the grid, and that the decay function, which describes offenders in general, fits this one. It is a search priority, not evidence that any person lives anywhere. Accuracy on solved series in the same place (`jtacc()`) is the only guide to its worth.

Value

A list. `raw` and `prob`, matrices of `nx` rows (across) and `ny` columns (up), cell `[i, j]` centred at `(xmin + (i - 1/2) dx, ymin + (j - 1/2) dy)`: the surface and its normalised version (summing to 1). `peak`, a named vector: `cell` (the first cell of maximum value, numbered down the columns of the matrix), `x` and `y` (its centre).

References

Canter, D., Coffey, T., Huntley, M. and Missen, C. (2000). Predicting serial killers' home base using a decision support system. *Journal of Quantitative Criminology*, 16(4), 457-478. doi:10.1023/A:1007551316253

O'Leary, M. (2009). The mathematics of geographic profiling. *Journal of Investigative Psychology and Offender Profiling*, 6(3), 253-265. doi:10.1002/jip.111

Examples

```
s <- jtsurf(c(2, 3, 4), c(2, 4, 3), win = c(0, 6, 0, 6), nx = 30,
            ny = 30, type = "negexp", par = c(1, 1), mode = "lik")
s$peak
```

`kinidx` *Kinship index*

Description

The weight of two DNA profiles for a stated relationship against no relationship, locus by locus and overall. Loci are taken as independent; there is no allowance for mutation or coancestry. See `proof("kinidx")`.

Usage

```
kinidx(g1, g2, freq, rel = "parent")
```

Arguments

`g1, g2`	The two genotypes: two-column matrices of allele labels, one row per locus (row names name the loci), or two alleles for one locus.

`freq` The population frequencies of the alleles of `g1`, same shape. When `g1`, `g2` and `freq` all have row names, their rows are matched to `g1` by locus name (an error lists loci that differ); otherwise by position, so only names protect against a mismatch.

`rel` The relationship: `"parent"`, `"sibling"`, `"half"` (half siblings, grandparent or uncle) or `"cousin"`, or the three probabilities of sharing 0, 1 and 2 alleles by descent. Names are matched exactly, in lower case.

Value

A list: `locus`, a data frame with `locus` and `idx`; `log10idx`, the base-10 logarithm of the combined index; and `post`, the probability of the relationship at even prior odds.

Examples

```
kinidx(g1 = c(12, 14), g2 = c(12, 14), freq = c(0.1, 0.2), rel = "sibling")
```

---

kmpart *K-means partition of points*

Description

Partitions the points into `k` groups by Lloyd's iterations: each point joins its nearest centre (the lowest-numbered on ties), then each centre moves to its points' mean (`upd = "mean"`, K-means, minimising the within-group sum of squares) or to their centre of minimum distance (`upd = "weber"`, a spatial K-medians minimising the sum of distances), until no point changes group or after `maxit` moves. An emptied group keeps its centre. The result depends on the start: k-means++ seeding (Arthur and Vassilvitskii 2007), the centres of the busiest grid cells at least a given distance apart, or centres given.

Usage

```
kmpart(
  x,
  y = NULL,
  k,
  init = "pp",
  cell = NULL,
  upd = "mean",
  maxit = 100,
  seed = NULL,
  count = NULL,
  lonlat = FALSE
)
```

Arguments

x, y	The points: x and y numeric vectors of equal length; or x a matrix or data frame with columns named x and y (or X and Y, as sf::st_coordinates() names them, or lon and lat with lonlat = TRUE) and y NULL; or x an sf or sfc object of POINT geometries, whose coordinate reference system must agree with lonlat. An unnamed two-column matrix is read as x then y; other columns are never taken by position. Vectors given separately (x, y, and w or t where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
k	The number of groups: one whole number from 1 to the number of points.
init	The start: "pp" (k-means++, drawn from seed), "cells" (square cells of side cell[1] from the lower-left corner of the points' bounding box, taken by decreasing count, a cell's centre becoming a seed when it is at least cell[2] from every earlier one), or the k starting centres as a matrix or data frame of coordinates (degrees when lonlat = TRUE).
cell	For init = "cells": c(side, separation), the side positive and the separation from 0, in the unit of the coordinates.
upd	The centre update: "mean" or "weber".
maxit	The largest number of iterations: one whole number from 0.
seed	NULL to draw the simulations' seed from R's generator (so set.seed() fixes them), or one whole number from 0.
count	NULL for points, or for zones: one whole count from 0 per zone, the points x, y being the zones' centroids. A zone is taken as count incidents at its centroid, which these hot-spot rules treat exactly as coincident points. Zones with count 0 get NA.
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A list. cl, the group of each point (1 to k, input order). cen, a data frame with one row per group: x, y (its centre, in degrees when lonlat = TRUE), n (points), ss (sum of squared distances to the centre) and sd (sum of distances). fit, a named vector: ss and sd (totals), iter (moves made) and conv (1 if no point changed group at the end).

References

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129-137. doi:10.1109/TIT.1982.1056489

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027-1035.

Examples

```
set.seed(1)
x <- c(rnorm(30, 0), rnorm(30, 5))
y <- c(rnorm(30, 0), rnorm(30, 5))
kmpart(x, y, k = 2, seed = 1)$cen
```

`lgtreg` *Logistic regression*

Description

Maximum likelihood for a 0/1 outcome and a logit link, by Newton's method on the design with centred columns, the method `glm()` uses with `family = binomial()` when the columns are not collinear. Standard errors come from the inverse Fisher information.

Usage

```
lgtreg(x, y, tol = 1e-10, maxit = 50)
```

Arguments

`x`	The predictors: a numeric matrix with more rows than columns plus one, with no missing values; an intercept is always added.
`y`	The outcome: 0 or 1, one per row of `x`, both present.
`tol`	Newton steps stop when none moves a coefficient by more than `tol (1 + |coefficient|)`: one positive number.
`maxit`	The most Newton steps: one whole number.

Value

A list: `coef` and `se`, named numeric vectors with the intercept first, and `dev`, the residual deviance.

References

McCullagh, P. and Nelder, J. A. (1989). *Generalized Linear Models* (2nd ed.). Chapman and Hall. doi:10.1007/9781489932426

Examples

```
set.seed(1)
x <- matrix(rnorm(300 * 2), 300)
y <- rbinom(300, 1, plogis(0.5 + x[, 1] - x[, 2]))
lgtreg(x, y)$coef
```

`lipaud` *Individual fairness: a Lipschitz audit*

Description

Dwork et al. (2012) ask that similar individuals be treated similarly: that the gap between two people's scores be at most `lip` times the distance between them, for a distance that says who is alike for the task. `lipaud()` checks the given pairs against that rule and reports the smallest constant the scores satisfy on them.

Usage

```
lipaud(h, d = NULL, pairs = NULL, lip = 1)
```

Arguments

`h`	The scores: finite numbers, one per person.
`d`	The distances: with `pairs` `NULL`, a `stats::dist()` object or a symmetric matrix between all the people (every pair is audited); with `pairs`, `NULL` (all 1) or one number from 0 per pair.
`pairs`	`NULL`, or the pairs to audit: a matrix of two columns of person numbers from 1.
`lip`	The constant: one number from 0.

Details

A pair at distance 0 is two people the distance cannot tell apart: any gap between their scores breaks the rule, and no constant repairs it (its ratio is `Inf`). The audit is only as good as the distance: it says whether the scores treat alike those the distance calls alike, not whether the distance is the right one.

Value

A list: `lreq` (the smallest constant the scores satisfy on these pairs), `nviol` and `rate` (the pairs breaking the rule at `lip`, and their share), `ok` (`TRUE` when none does), `maxgap` (the largest gap), `worst` (the first pair with the largest ratio), and `pairs`, a data frame with one row per pair: `i`, `j`, `d`, `gap`, `ratio` and `viol`.

References

Dwork, C., Hardt, M., Pitassi, T., Reingold, O. and Zemel, R. (2012). Fairness through awareness. *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 214-226. doi:10.1145/2090236.2090255

Examples

```
x <- c(1, 2, 2.5, 6)
h <- c(0.10, 0.20, 0.60, 0.65)
# scores on a scale of 0 to 1 against distances in years
lipaud(h, dist(x), lip = 0.2)
```

`lmorn` *Local Moran (LISA)*

Description

Anselin's (1995) local indicator of spatial association for each unit: with `z = x - mean(x)` and `m2 = mean(z^2)`, `I_i = z_i / m2 sum_j w_ij z_j`, its expectation and variance under total randomisation, the two-sided normal p-value, and a conditional permutation p-value: unit `i` keeps its value and its neighbours' values are drawn without replacement from the others, `nsim` times from `seed`. The quadrant tells high-high and low-low clusters from high-low and low-high outliers. The p-values are per unit and not adjusted for the many tests made (use `stats::p.adjust()`); with row-standardised weights the sum of the `I_i` is `S0` times global Moran's I. That identity is proved in Lean: see `proof("lmorn")`.

Usage

```
lmorn(x, w, nsim = 0, seed = NULL)
```

Arguments

`x`	The values: finite numbers, one per unit.
`w`	The weights: `spwt()`'s list, or an spdep `listw` object, for as many units as `x`.
`nsim`	The number of permutations: a whole number from 0.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.

Value

A data frame with one row per unit: `ii` (I_i), `e`, `v`, `z`, `p`, `pperm` (`(1 + min(#{I* >= I}, #{I* <= I})) / (nsim + 1)`, `NaN` without permutations) and `quad` (1 high-high, 2 low-low, 3 low-high, 4 high-low, 0 neither).

References

Anselin, L. (1995). Local indicators of spatial association: LISA. *Geographical Analysis*, 27(2), 93-115. doi:10.1111/j.15384632.1995.tb00338.x

Examples

```
set.seed(1)
p <- cbind(x = runif(60), y = runif(60))
v <- ifelse(p[, 1] > 0.7 & p[, 2] > 0.7, 5, 1) + rnorm(60, sd = 0.3)
l <- lmorn(v, spwt(p, k = 5), nsim = 199, seed = 1)
head(l)
```

`locgi` *Local G and G* of Getis and Ord*

Description

The local statistics of Getis and Ord (1992) as z values (Ord and Getis 1995): `G_i` compares the weighted sum of the neighbours' values with its expectation from the other `n - 1` values; `G*_i` (`star = TRUE`) includes the unit itself with weight 1 and uses all `n`. Large positive values mark concentrations of high values, large negative ones of low values. With `star = TRUE` the weights are best binary (`style = "raw"`).

Usage

```
locgi(x, w, star = TRUE)
```

Arguments

`x`	The values: finite numbers, one per unit.
`w`	The weights: `spwt()`'s list, or an spdep `listw` object, for as many units as `x`.
`star`	`TRUE` for G*, `FALSE` for G.

Value

A numeric vector of z values, one per unit; `NaN` for a unit without neighbours.

References

Ord, J. K. and Getis, A. (1995). Local spatial autocorrelation statistics: distributional issues and an application. *Geographical Analysis*, 27(4), 286-306. doi:10.1111/j.15384632.1995.tb00912.x

Examples

```
set.seed(1)
p <- cbind(x = runif(60), y = runif(60))
v <- rpois(60, ifelse(p[, 1] > 0.7, 6, 2))
locgi(v, spwt(p, type = "band", d = 0.2, style = "raw"))
```

losbnd *Mean time in custody from completed and current stays*

Description

The mean length of stay of the people released (or of the cases disposed) leaves out everyone still inside. Their elapsed time is a lower bound on their stay, so the mean over everyone admitted is at least `lower`, the mean of the completed stays and the elapsed times together; the released-only mean falls short of it by `m / (n + m)` times the excess of the mean elapsed time over the released mean, and understates the truth for certain when that excess is positive. No upper bound exists without a cap on how long the current stays will last. See `proof("losbnd")`.

Usage

```
losbnd(done, open)
```

Arguments

`done`	Completed stays: non-negative numbers.
`open`	Elapsed time of the stays still open: non-negative numbers.

Value

A named numeric vector: `mean` (of the completed stays), `age` (the mean elapsed time of the open ones), `lower` (the lower bound on the mean over all) and `bias` (`lower - mean`, the least by which the completed mean understates when it is positive).

References

Manski, C. F. (2003). *Partial Identification of Probability Distributions*. Springer. doi:10.1007/b97478

Examples

```
losbnd(done = c(30, 45, 60, 90, 120), open = c(100, 150, 200))
```

lrcal *Calibrated likelihood ratios*

Description

Recalibrates a set of likelihood ratios against the truth with the pool-adjacent-violators algorithm, keeping their order. The result can hold 0 and `Inf`: the data never contradicted those ratios. See `proof("lrcal")`.

Usage

```
lrcal(same, diff)
```

Arguments

same	Likelihood ratios from same-source comparisons: positive numbers.
diff	Likelihood ratios from different-source comparisons: positive numbers.

Value

A list: same and diff, the calibrated ratios in the order given.

Examples

```
lrcal(same = c(50, 200, 8), diff = c(0.01, 0.2, 1.5))
```

---

lrpost *Posterior probability from a likelihood ratio*

Description

Turns a likelihood ratio into the probability of the first hypothesis, which needs a prior. A large ratio with a small prior can still leave that probability low; reading the ratio as the probability is the prosecutor's fallacy. See `proof("lrpost")`.

Usage

```
lrpost(lr, prior)
```

Arguments

lr	Likelihood ratios: positive numbers.
prior	The prior probability of the first hypothesis, strictly between 0 and 1: one value, or one per ratio, matched to lr by name when both are named with unique names.

Value

The posterior probabilities.

Examples

```
lrpost(lr = 1e6, prior = 1 / 5e6)
```

lwss *Lowess smoothing*

Description

Robust locally weighted regression (Cleveland 1979) with the conventions of the classic implementation, so that the fit equals that of `stats::lowess()`; the fitted values come back in the order of the input, not sorted.

Usage

```
lwss(x, y, f = 2/3, iter = 3, delta = NULL)
```

Arguments

`x, y`	The points: finite numbers of equal length.
`f`	The share of points in each neighbourhood: positive.
`iter`	The number of robustness steps: a whole number from 0.
`delta`	Points within `delta` of the last fitted one are interpolated: `NULL` for 1 per cent of the range of `x`, or a number from 0.

Value

A numeric vector of fitted values, one per point in input order.

References

Cleveland, W. S. (1979). Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368), 829-836. doi:10.1080/01621459.1979.10481038

Examples

```
lwss(c(3, 1, 2, 5, 4), c(2, 1, 3, 6, 4))
```

mantst *Mantel's space-time test*

Description

The Pearson correlation, over all pairs of events, between their spatial dissimilarity (the distance, or `1 / (distance + cs)` when `cs > 0`, Mantel's reciprocal transform) and their temporal one (`|t_i - t_j|`, or `1 / (|t_i - t_j| + ct)`), with a permutation test that permutes the times over the fixed locations (Mantel 1967). Drawing distances and intervals independently instead, as some descriptions do, does not keep the dependence among the pairs of one distance matrix and gives no test of stated size. Memory grows with the number of events, not pairs; time with pairs times permutations.

Usage

```
mantst(
  x,
  y = NULL,
  t,
  cs = 0,
  ct = 0,
  nsim = 999,
  seed = NULL,
  metric = "euclid",
  lonlat = FALSE
)
```

Arguments

x, y	The points: x and y numeric vectors of equal length; or x a matrix or data frame with columns named x and y (or X and Y, as sf::st_coordinates() names them, or lon and lat with lonlat = TRUE) and y NULL; or x an sf or sfc object of POINT geometries, whose coordinate reference system must agree with lonlat. An unnamed two-column matrix is read as x then y; other columns are never taken by position. Vectors given separately (x, y, and w or t where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
t	The time of each event: finite numbers (for dates, as.numeric() of them, in days or seconds).
cs, ct	The constants of the reciprocal transforms: numbers from 0 (0 uses the raw distance or interval).
nsim	The number of permutations: one whole number from 0.
seed	NULL to draw the simulations' seed from R's generator (so set.seed() fixes them), or one whole number from 0.
metric	The spatial distance: "euclid" or "manhattan"; the test permutes times and needs no expectation, so both are exact.
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A named numeric vector: r (the correlation), pairs, and the permutation p-values pup ((1 + #{r_perm >= r}) / (nsim + 1) large r: events close in space are close in time when both are raw distances), plo (r_perm <= r) and p2 (|r_perm| >= |r|); NaN without permutations. 3 to 46340 events.

References

Mantel, N. (1967). The detection of disease clustering and a generalized regression approach. *Cancer Research*, 27(2), 209-220.

Examples

```
set.seed(1)
x <- runif(60)
y <- runif(60)
mantst(x, y, t = x + rnorm(60, sd = 0.1), nsim = 199, seed = 1)
```

`misrd` *Risk difference when group membership is misrecorded*

Description

When the group a person is recorded in (an officer's perception of race, say) differs at random from the true one, with the same error rates whatever the outcome, the measured groups are mixtures of the true ones. The measured risk difference is then the true one times the factor `(se + sp - 1) p (1 - p) / (d1 d0)`, where `d1` and `d0` are the shares recorded in each group. For any sensitivity and specificity it is never larger in size than the true difference, so a disparity in recorded data understates the true one; with `se + sp >= 1` it never has the opposite sign, while with `se + sp < 1` a difference other than 0 has it. See `proof("misrd")`.

Usage

```
misrd(r1, r0, prev, se, sp)
```

Arguments

`r1`	The true risk in group 1: one number from 0 to 1.
`r0`	The true risk in group 0: one number from 0 to 1.
`prev`	The true share in group 1, strictly between 0 and 1.
`se`	Sensitivity: the chance a member of group 1 is recorded in group 1, from 0 to 1.
`sp`	Specificity: the chance a member of group 0 is recorded in group 0, from 0 to 1.

Value

A named numeric vector: `true` (`r1 - r0`), `measured` (the difference between the recorded groups), `factor` (`measured / true` whenever `true` is not 0), `d1` and `d0`.

References

Bross, I. (1954). Misclassification in 2 x 2 tables. *Biometrics*, 10(4), 478-486. doi:10.2307/3001619

Examples

```
misrd(r1 = 0.3, r0 = 0.1, prev = 0.4, se = 0.8, sp = 0.9)
```

`mixlr` *Mixture likelihood ratio*

Description

The weight of a DNA mixture at one locus for one account of who contributed against another. Each account names the alleles of the mixture its known contributors do not explain and the number of unknown contributors who must explain them. Peak heights, drop-out and coancestry are not modelled. See `proof("mixlr")`.

Usage

```
mixlr(p, np, xp, nd, xd)
```

Arguments

`p`	The population frequencies of the alleles in the mixture: named numbers above 0 with a sum of at most 1.
`np`, `nd`	The alleles left unexplained under the first and the second account: names of `p` (or `character(0)`), at most 20.
`xp`, `xd`	The number of unknown contributors under each account.

Details

With no alleles to explain and one unknown, the denominator is the chance a random person is not excluded from the mixture.

Value

A named numeric vector: `num`, `den` and `lr`.

References

Weir, B. S., Triggs, C. M., Starling, L., Stowell, L. I., Walsh, K. A. J. and Buckleton, J. (1997). Interpreting DNA mixtures. *Journal of Forensic Sciences*, 42(2), 213-222.

Examples

```
# victim explains a and b; suspect would explain c and d
p <- c(a = 0.1, b = 0.2, c = 0.05, d = 0.15)
mixlr(p, np = character(0), xp = 0, nd = c("c", "d"), xd = 1)
```

`mvtll` *Group-based multivariate trajectories*

Description

The group-based trajectory model with several normal indicators per unit (Nagin 1999; Magrini 2022): given group `k`, the `r` indicators of a unit at period `t` are multivariate normal with mean `B_k' x_t` and covariance `S_k`, independently over periods. `mvtll()` evaluates the log-likelihood and posterior memberships; `mvtem()` fits by EM with closed-form steps from the values given. A period with any missing indicator is left out for that unit.

Usage

```
mvtll(y, x, beta, sig, prob)

mvtem(y, x, beta, sig, prob, maxit = 500, tol = 1e-08)
```

Arguments

`y`	The indicators: an array of units by periods by indicators; `NA` allowed, not infinite values.
`x`	The design: a matrix with one row per period.
`beta`	The coefficients: an array of design columns by indicators by groups.
`sig`	The covariances: an array of indicators by indicators by groups, each positive definite.
`prob`	The group shares: positive numbers summing to 1.
`maxit`	The most iterations: a whole number from 1.
`tol`	Stop when an iteration gains at most `tol (1 + |ll|)`.

Value

`mvtll()`: a list of `ll` and `post`. `mvtem()`: a list of the fitted `beta`, `sig`, `prob`, `post`, `ll` and `iter`.

References

Magrini, A. (2022). Assessment of agricultural sustainability in European Union countries: a group-based multivariate trajectory approach. *AStA Advances in Statistical Analysis*, 106. doi:10.1007/s10182022004379

Examples

```
set.seed(1)
x <- cbind(1, 1:5)
y <- array(rnorm(40 * 5 * 2), c(40, 5, 2))
y[1:20, , 1] <- y[1:20, , 1] + 3
b <- array(c(0, 0, 0, 0, 3, 0, 0, 0), c(2, 2, 2))
s <- array(diag(2), c(2, 2, 2))
mvtem(y, x, b, s, prob = c(0.5, 0.5))$prob
```

nnhclu *Nearest-neighbour hierarchical clustering*

Description

Groups points that lie closer together than a threshold distance, then groups the groups, order by order (after Johnson 1967, as used for crime hot spots). At the first order two points are linked when their distance is below the threshold: `fixd`, or the mean nearest-neighbour distance expected under complete spatial randomness in an area `area` plus `z` standard errors, `0.5 sqrt(A / m) + z sqrt((4 - pi) A / (4 pi)) / m` for `m` units, with `z = qnorm(p)` (`p = 0.5` gives the expectation itself, smaller `p` a stricter threshold). Points with a linked neighbour are taken in decreasing number of links; each one not yet clustered takes every unclustered point linked to it; groups of at least `minpts` points are kept. Their centres of minimum distance become the units of the next order, with the threshold recomputed for their number, while at least four remain, up to `maxord` orders. The iterative reassignment some descriptions add after the first pass is not done: its rule is not stated unambiguously.

Usage

```
nnhclu(
  x,
  y = NULL,
  area = NULL,
  fixd = NULL,
  p = 0.5,
  minpts = 10,
  minhi = minpts,
  maxord = 3,
  k = 1,
  c = 1,
  count = NULL,
  lonlat = FALSE
)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`area`	The study area for the random-expectation threshold, in the square of the coordinates' unit (square metres when `lonlat = TRUE`). Give `area` or `fixd`, not both.

`fixd`	A fixed threshold distance instead, positive.
`p`	The one-tailed probability that a pair is as close by chance: one number strictly between 0 and 1.
`minpts`, `minhi`	The fewest points (order 1) and member units (higher orders) a cluster keeps: whole numbers from 2.
`maxord`	The highest order: one whole number from 1 to 20.
`k`	The divisor offset: 0, 1 or 2; the total weight must exceed it.
`c`	The multiplier of each cluster's standard deviational ellipse (see `sdell()`).
`count`	`NULL` for points, or for zones: one whole count from 0 per zone, the points `x`, `y` being the zones' centroids. A zone is taken as `count` incidents at its centroid, which these hot-spot rules treat exactly as coincident points. Zones with count 0 get `NA`; the random-expectation threshold counts incidents, not zones.
`lonlat`	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A list. `cl`, an integer matrix with one row per point (input order) and one column per order: the cluster (a row of `clusters`) the point belongs to at that order, or `NA`. `clusters`, a data frame with one row per cluster: `order`, `points`, `units` (member units), `mx`, `my` (mean centre), `wx`, `wy` (centre of minimum distance; both in degrees when `lonlat = TRUE`), `a`, `b`, `angle` (the ellipse of the member units, `NaN` when too few for the divisor) and `thresh` (the threshold distance used).

References

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3), 241-254. doi:10.1007/BF02289588

Eck, J. E., Chainey, S., Cameron, J. G., Leitner, M. and Wilson, R. E. (2005). *Mapping Crime: Understanding Hot Spots*. National Institute of Justice, NCJ 209393.

Examples

```
set.seed(1)
x <- c(rnorm(40, 0, 0.3), rnorm(40, 6, 0.3), runif(40, -5, 10))
y <- c(rnorm(40, 0, 0.3), rnorm(40, 6, 0.3), runif(40, -5, 10))
nnhclu(x, y, area = 225, p = 0.05, minpts = 5)$clusters
```

nnidx *Nearest-neighbour index*

Description

The mean distance from each point to its k-th nearest neighbour, against its expectation under complete spatial randomness (CSR) in the rectangle `win`: the nearest-neighbour index of Clark and Evans (1954) and its k-order extension (Thompson 1956), with Donnelly's (1978) edge-corrected expectation for the first order and, when `nsim > 0`, a Monte Carlo test against `nsim` patterns of as many uniform points in the rectangle. The Monte Carlo p-values have exact size; the z statistics ignore that nearest-neighbour distances are dependent and, uncorrected, the edge. The index tests only the k-th neighbour distances, not complete spatial randomness as a whole.

Usage

```
nnidx(
  x,
  y = NULL,
  win,
  kmax = 1,
  nsim = 0,
  seed = NULL,
  dist = FALSE,
  metric = "euclid"
)
```

Arguments

`x, y`	The points in planar coordinates (project longitude and latitude with `eqproj()` first): `x` and `y` numeric vectors of equal length, or `x` a matrix or data frame with columns named `x` and `y` (an unnamed two-column matrix is read as `x` then `y`). At least two points, all inside `win`; no missing values.
`win`	The study rectangle `c(xmin, xmax, ymin, ymax)`, in the unit of the coordinates. The intensity is the number of points over its area.
`kmax`	The highest neighbour order: one whole number from 1 to the number of points less 1.
`nsim`	The number of simulated patterns for the Monte Carlo test: one whole number from 0 (no test).
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.
`dist`	`TRUE` to return every point's neighbour distances.
`metric`	The distance: `"euclid"` or `"manhattan"` (`|dx| + |dy|`, a street grid aligned with the axes), as in `spwt()` and `jtsurf()`. Under Manhattan distance the expectations use the area of the Manhattan ball, `2 r^2`: the mean k-th distance is `sqrt(pi / 2)` times the Euclidean expectation and `E R_k^2 = k / (2 lambda)`;

no edge-corrected (Donnelly) expectation is published for it, so `ce` holds `NaN` except `z`, about the uncorrected expectation with `Var R_1 = (4 - pi) / (8 lambda)`. The Monte Carlo test is unchanged.

Value

A list. `order`, a data frame with one row per order `k`: `mean` (the mean k-th neighbour distance), `exp` (its expectation under CSR without edge effects, `c_k / sqrt(lambda)`), `ratio`, `pclu` and `preg` (the Monte Carlo p-values against clustering and against regularity, `NaN` without simulations) and `sd` (the standard deviation of one k-th distance under CSR). `ce`, a named vector for the first order: `donnelly` (the edge-corrected expectation), `ratio` (the index against it), `z` and `zc` (the Clark-Evans z statistics about the uncorrected and the corrected expectation). `dist`, `NULL` or a matrix with one row per point (input order) and one column per order. Distances are in the unit of the coordinates.

References

Clark, P. J. and Evans, F. C. (1954). Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35(4), 445-453. doi:10.2307/1931034

Thompson, H. R. (1956). Distribution of distance to nth neighbour in a population of randomly distributed individuals. *Ecology*, 37(2), 391-394. doi:10.2307/1933159

Donnelly, K. P. (1978). Simulations to determine the variance and edge effect of total nearest-neighbour distance. In I. Hodder (ed.), *Simulation Methods in Archaeology*, 91-95. Cambridge University Press.

Examples

```
set.seed(1)
x <- runif(200)
y <- runif(200)
nnidx(x, y, win = c(0, 1, 0, 1), kmax = 3, nsim = 99)
```

`nordet` *Noisy-OR detection by patrols*

Description

The chance that an incident is seen by patrols when each officer within `radius` of it sees it independently with probability `p`: with `k` officers in range, `1 - (1 - p)^k` (the detection model of Barman and Barman 2026). With `draw = TRUE`, one detection outcome is drawn for each incident. A model of who is seen, under stated assumptions; how patrols are actually placed and how much they see are matters of fact the model does not settle.

Usage

```
nordet(x, off, radius, p = 0.85, draw = FALSE, seed = NULL)
```

Arguments

x	The incidents: a numeric matrix of two columns, or a data frame with columns x and y (or two columns), planar coordinates.
off	The officers, in the same forms; NULL or no rows for none.
radius	The range: one positive number, in the coordinates' units (distances are Euclidean; an officer exactly at radius counts).
p	The chance one officer in range sees an incident: one number above 0 and at most 1.
draw	TRUE to draw whether each incident is detected.
seed	NULL to draw the seed of the detections from R's generator, or one whole number from 0; read only with draw.

Value

A data frame with one row per incident: k (officers in range), prob and, with draw, det (1 detected, 0 not).

Examples

```
set.seed(1)
inc <- matrix(runif(20), ncol = 2)
off <- matrix(runif(10), ncol = 2)
nordet(inc, off, radius = 0.3, p = 0.7)
```

---

nsbnd *A true base rate behind a noisily recorded one*

Description

A recorded outcome is often a proxy: a rearrest for a reoffence, a recorded crime for one committed. With false-positive noise a, the chance a case without the outcome is recorded with it, and false-negative noise b, the chance a case with it is not, a true base rate p is recorded as pobs = p (1 - b) + (1 - p) a. Given only the boxes the noise rates lie in, nsbnd() gives the interval of true rates consistent with each recorded rate, and says for each pair of groups whether the order of their true rates follows from the records whatever the noise.

Usage

```
nsbnd(pobs, a, b)
```

Arguments

pobs	Recorded rates: numbers from 0 to 1, named or not.
a	The box of the false-positive noise rate: one number a (the box from 0 to a) or two, its ends; from 0.
b	The box of the false-negative noise rate, in the same way. The upper ends of a and b must add to less than 1.

Details

The true rate `(pobs - a) / (1 - a - b)` falls as `a` grows and rises as `b` grows, so its ends are reached at corners of the boxes: the lower at the largest `a` (capped at `pobs`) and smallest `b`, the upper at the smallest `a` and largest `b` (capped at `1 - pobs`). Both ends are attained, so no narrower interval follows from the boxes alone; with known noise rates (each box one point) the interval is the one true rate. A recorded rate that no admissible noise can produce (below the smallest `a`, or above `1 -` the smallest `b`) has no interval (`NaN`).

Two groups' true rates are ordered whatever the noise exactly when their intervals do not overlap; when they overlap there is admissible noise under which either is the larger. `bd` is the largest upper end of the `b` box, `a` held, at which the lower recorded rate's interval stays below the other's: how much under-recording the order survives. The boxes are the analyst's claim about how the records were made; the bounds are only as good as that claim. See `proof("nsbnd")`.

Value

A list: `bnd`, a data frame with one row per rate: `rate` (its name, or its position), `pobs`, `lo`, `hi` and `width`; `pairs`, one row per pair `i` before `j`: `i`, `j`, `ord` (`"lower"`: `i`'s true rate is below `j`'s for every admissible noise; `"higher"`; `"undecided"`; `NA` when either has no interval) and `bd` (`NA` when the rates are equal or no box keeps them ordered).

References

Fogliato, R., Chouldechova, A. and G'Sell, M. (2020). Fairness evaluation in presence of biased noisy labels. *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, PMLR 108, 2325-2336. doi:10.48550/arXiv.2003.13808

Examples

```
# rearrest rates of two groups; up to 10% false positives, 20% missed
nsbnd(c(a = 0.35, b = 0.55), a = 0.10, b = 0.20)
# with up to 40% missed, the order is no longer identified
nsbnd(c(a = 0.35, b = 0.55), a = 0.05, b = 0.40)$pairs
# known noise rates give the true rate itself
nsbnd(0.4 * 0.8 + 0.6 * 0.1, a = c(0.1, 0.1), b = c(0.2, 0.2))$bnd
```

---

ntbrk *Brokerage roles*

Description

Counts, for each node v, the two-paths `i -> v -> j` with `i != j` and no tie `i -> j` (v brokers between them), by the roles of Gould and Fernandez (1989), given each node's group: `coord` (coordinator: i, v and j in one group), `itin` (itinerant: i and j in one group, v in another), `repr` (representative: v in i's group, j outside), `gate` (gatekeeper: v in j's group, i outside) and `liais` (liaison: three groups). Undirected, each tie runs both ways, so each unordered pair of neighbours counts twice and `repr` equals `gate`. Weights are not taken. Graph rules as in `ntdeg()`. Raw counts only: the expected counts and z-scores of Gould and Fernandez are not given.

Usage

```
ntbrk(from, to, grp, nodes = NULL, dir = FALSE)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
grp	The group of each node: labels, named by node or one per node in node order, as memb in ntmod().
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.
dir	TRUE for a directed graph (from sends a tie to to).

Value

A data frame: node, grp (as text), coord, itin, repr, gate, liais and total.

References

Gould, R. V. and Fernandez, R. M. (1989). Structures of mediation: a formal approach to brokerage in transaction networks. *Sociological Methodology*, 19, 89-126. doi:10.2307/270949

Examples

```
ntbrk(c("a", "b", "b"), c("b", "c", "d"),
      grp = c(a = "x", b = "y", c = "y", d = "z"), dir = TRUE)
```

---

ntbtw *Betweenness of each node*

Description

The sum, over pairs of other nodes, of the share of their shortest paths that pass through the node (Freeman 1977), by Brandes' algorithm in O(nodes x edges) time. Paths are counted in steps: weights are tie strengths, not lengths, and are not taken. An undirected pair is counted once. Graph rules as in ntdeg().

Usage

```
ntbtw(from, to, nodes = NULL, dir = FALSE, norm = FALSE)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.
dir	TRUE for a directed graph (from sends a tie to to).
norm	TRUE to divide by the number of pairs of other nodes, (n - 1)(n - 2) directed or (n - 1)(n - 2) / 2 undirected, as igraph::betweenness(normalized = TRUE) does; NaN for fewer than 3 nodes.

Value

A data frame: node, btw.

References

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2), 163-177. doi:10.1080/0022250X.2001.9990249

Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, 40(1), 35-41. doi:10.2307/3033543

Examples

```
ntbtw(c(1, 2, 3), c(2, 3, 4))
```

---

ntchg *Cheeger bounds on a network's bottleneck*

Description

The conductance of a set of nodes S is the weight of the edges leaving it over the smaller of its volume and that of the rest (volume: summed strength); the graph's conductance h is the least over all sets, a measure of how cleanly the network splits in two. Finding h is hard, but the second-smallest eigenvalue lambda2 of the normalised Laplacian I - D^-1/2 A D^-1/2 brackets it: lambda2 / 2 <= h <= sqrt(2 lambda2) (Cheeger's inequality; the lower half is proof("ntchg")). The sweep over the eigenvector finds a set whose conductance (sweep) is at most sqrt(2 lambda2); h is at most sweep.

Usage

```
ntchg(from, to, w = NULL, nodes = NULL, tol = 1e-10, maxit = 1e+05)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
w	NULL, or one positive weight per edge (a tie's strength, such as the number of co-offences).
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.
tol	The stopping tolerance: one number between 0 and 1.
maxit	The most steps: one whole number, at least 1.

Details

lambda2 is found by power iteration on (I + D^-1/2 A D^-1/2) / 2 kept orthogonal to D^1/2 1, from a fixed pseudo-random start, until the residual is below tol or after maxit steps (conv FALSE). Convergence is slow when lambda2 and the next eigenvalue are close (long paths, large sparse graphs). The graph is undirected and must be connected; weights are tie strengths. Graph rules as in ntdeg().

Value

A list: bound, a named vector lambda2, lower (lambda2 / 2), upper (sqrt(2 lambda2)) and sweep; set, the labels of the sweep set; iter; conv.

References

Chung, F. R. K. (1997). *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics 92. American Mathematical Society. doi:10.1090/cbms/092

Examples

```
# two triangles joined by one edge
ntchg(c(1, 1, 2, 4, 4, 5, 3), c(2, 3, 3, 5, 6, 6, 4))
```

---

ntcls *Closeness of each node*

Description

Distances are counted in steps (weights are not taken), along edges' directions when dir = TRUE (out-distances). A disconnected graph is handled in the two standard ways, both returned: cls and ncls count only the nodes a node reaches (as igraph::closeness() does), so a node close to the few it reaches scores high; hrm, the harmonic centrality, sums 1 / distance over every other node, an unreachable one adding 0, so reaching more nodes always scores more. Graph rules as in ntdeg().

Usage

```
ntcls(from, to, nodes = NULL, dir = FALSE)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.
dir	TRUE for a directed graph (from sends a tie to to).

Value

A data frame: node, reach (nodes reached), cls (1 / the summed distance to them), ncls (reach / that sum, the inverse mean distance; cls and ncls are NaN when nothing is reached) and hrm (the harmonic centrality).

References

Freeman, L. C. (1978). Centrality in social networks: conceptual clarification. *Social Networks*, 1(3), 215-239. doi:10.1016/03788733(78)900217

Marchiori, M. and Latora, V. (2000). Harmony in the small-world. *Physica A*, 285(3-4), 539-546. doi:10.1016/S03784371(00)003113

Examples

```
ntcls(c(1, 2, 3), c(2, 3, 4), nodes = 1:5)
```

---

ntcmp *Connected components*

Description

Weak components join nodes linked by edges in either direction; strong components (Tarjan's algorithm, without recursion) join nodes that reach each other along the edges' directions. Undirected, the two coincide. Components are numbered 1, 2, ... in the order of their first node. Graph rules as in ntdeg().

Usage

```
ntcmp(from, to, nodes = NULL, dir = FALSE)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.
dir	TRUE for a directed graph (from sends a tie to to).

Value

A data frame: node, weak, strong.

References

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2), 146-160. doi:10.1137/0201010

Examples

```
ntcmp(c("a", "b", "c", "x"), c("b", "c", "a", "y"), dir = TRUE)
```

---

ntcore *k-core of each node*

Description

The coreness of a node is the largest k such that it belongs to a subgraph in which every node has at least k neighbours (Seidman 1983), computed in O(nodes + edges) by the algorithm of Batagelj and Zaversnik. The cores are nested: the (k + 1)-core lies inside the k-core (proof("ntcore")). Edges are taken as undirected, without weights. Graph rules as in ntdeg().

Usage

```
ntcore(from, to, nodes = NULL)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.

Value

A data frame: node, core.

References

Seidman, S. B. (1983). Network structure and minimum degree. *Social Networks*, 5(3), 269-287. doi:10.1016/03788733(83)90028X

Batagelj, V. and Zaversnik, M. (2003). An O(m) algorithm for cores decomposition of networks. doi:10.48550/arXiv.cs/0310049

Examples

```
ntcore(c(1, 1, 1, 2, 2, 3, 4), c(2, 3, 4, 3, 4, 4, 5))
```

`ntdeg` *Degree and strength of each node*

Description

A graph is an edge list: `from[k]` is tied to `to[k]`. The rules below hold for every network function in crimlab.

Usage

```
ntdeg(from, to, w = NULL, nodes = NULL, dir = FALSE)
```

Arguments

`from`, `to`	Labels of the two ends of each edge: vectors of the same length (zero when `nodes` gives every node) with no missing values.
`w`	`NULL`, or one positive weight per edge (a tie's strength, such as the number of co-offences).
`nodes`	`NULL`, or every node's label once: the node order of the results and the way to include isolated nodes.
`dir`	`TRUE` for a directed graph (`from` sends a tie to `to`).

Details

- Nodes are identified by their labels, compared as text (`as.character()`; a factor by its labels, unused levels ignored). Results have one row per node, in the order nodes first appear: each edge's `from`, then its `to`, edge by edge. With `nodes`, the order is that of `nodes`, which must hold every label of the edges once; labels in `nodes` that no edge names are isolated nodes.
- A self-loop (`from[k] == to[k]`) is dropped; its node stays.
- Repeated edges (the same pair, in either direction when undirected) are merged into one edge whose weight is the sum of their weights (1 each when `w` is `NULL`). So `deg` counts distinct neighbours and `wdeg` counts edges or their weight.
- A missing value in any argument is an error.

Value

A data frame with one row per node: `node`, `deg` (distinct neighbours) and `wdeg` (summed weight); directed, `node`, `in`, `out`, `deg` (`in + out`), `win`, `wout` and `wdeg`. The degrees sum to twice the number of distinct edges: see `proof("ntdeg")`.

References

Wasserman, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications*. Cambridge University Press. doi:10.1017/CBO9780511815478

Examples

```
ntdeg(c("a", "a", "b", "a"), c("b", "c", "c", "b"))
ntdeg(c("a", "a"), c("b", "c"), nodes = c("c", "b", "a", "z"), dir = TRUE)
```

---

ntevc *Eigenvector centrality of each node*

Description

The leading eigenvector of the (weighted) adjacency matrix of the undirected graph, scaled to a maximum of 1 (Bonacich 1972). It is found by power iteration on `A + c I`, `c` the largest weight, which has the same eigenvectors and whose leading eigenvalue is strictly dominant even on a bipartite graph (a co-offending network between two roles, a star), where plain power iteration oscillates. The iteration starts from 1 for every node and stops when no value changes by `tol` or more between two steps, or after `maxit` steps (then `conv` is `FALSE`).

Usage

```
ntevc(from, to, w = NULL, nodes = NULL, tol = 1e-13, maxit = 1e+05)
```

Arguments

`from`, `to`	Labels of the two ends of each edge: vectors of the same length (zero when `nodes` gives every node) with no missing values.
`w`	`NULL`, or one positive weight per edge (a tie's strength, such as the number of co-offences).
`nodes`	`NULL`, or every node's label once: the node order of the results and the way to include isolated nodes.
`tol`	The stopping tolerance: one number between 0 and 1.
`maxit`	The most steps: one whole number, at least 1.

Details

On a disconnected graph the leading eigenvector lives on the components with the largest leading eigenvalue; every other component gets 0 (each component is iterated and scaled on its own, and a component whose eigenvalue falls short of the largest by more than 1e-10 of it is set to 0). When several components share the largest eigenvalue each is scaled to a maximum of 1: any mix of them is an eigenvector too, so compare nodes within a component. A graph without edges gives 1 for every node. Graph rules as in `ntdeg()`.

Value

A list: `cent`, a data frame with `node` and `evc`; `value`, the leading eigenvalue; `iter`, the steps taken; `conv`, whether the tolerance was met.

References

Bonacich, P. (1972). Factoring and weighting approaches to status scores and clique identification. *Journal of Mathematical Sociology*, 2(1), 113-120. doi:10.1080/0022250X.1972.9989806

Examples

```
ntevc(c("hub", "hub", "hub", "a"), c("a", "b", "c", "b"))
```

`ntlvn` *Communities by the Louvain method*

Description

Groups nodes to raise the modularity (see `ntmod()`) of the undirected, weighted graph (Blondel et al. 2008). Each level visits the nodes in node order when `seed` is `NULL`, or in a random order drawn from `seed` (one stream for the whole run), and moves each to the neighbouring community with the largest gain in modularity, the first met among equal gains, when that gain beats staying by more than 1e-12 of the node's strength; passes repeat until none moves a node. The communities then become the nodes of the next level, and a level that moves nothing ends the run. The result is the same on every platform for the same input and seed.

Usage

```
ntlvn(from, to, w = NULL, nodes = NULL, res = 1, seed = NULL)
```

Arguments

`from`, `to`	Labels of the two ends of each edge: vectors of the same length (zero when `nodes` gives every node) with no missing values.
`w`	`NULL`, or one positive weight per edge (a tie's strength, such as the number of co-offences).
`nodes`	`NULL`, or every node's label once: the node order of the results and the way to include isolated nodes.

res	The resolution: one positive number; above 1 favours smaller communities.
seed	NULL for node order, or one whole number from 0.

Details

The method is a heuristic: different orders reach different local optima (on Zachary's karate club, modularity from about 0.39 to the optimum 0.4198), so run several seeds and keep the best. Graph rules as in ntdeg(); isolated nodes stay alone.

Value

A list: memb, a data frame with node and comm (communities numbered 1, 2, ... in the order of their first node); q, its modularity; levels, the levels that moved a node.

References

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics*, P10008. doi:10.1088/17425468/2008/10/P10008

Examples

```
e <- c(1, 1, 2, 4, 4, 5, 3)
f <- c(2, 3, 3, 5, 6, 6, 4)
ntlvn(e, f)
```

---

ntmod *Modularity of a partition*

Description

Q = sum over communities of in / 2m - res (tot / 2m)^2, where in is the weight of edges inside the community (counted from both ends), tot the summed strength of its nodes and 2m the total strength (Newman and Girvan 2004), on the undirected, weighted graph. At res = 1 it lies between -1/2 and 1 (proof("ntmod")). NaN for a graph without edges. Graph rules as in ntdeg().

Usage

```
ntmod(from, to, memb, w = NULL, nodes = NULL, res = 1)
```

Arguments

from, to	Labels of the two ends of each edge: vectors of the same length (zero when nodes gives every node) with no missing values.
memb	The community of each node: labels, either named by node (every node named once; other names are ignored) or unnamed, one per node in the order the network functions report nodes.

w	NULL, or one positive weight per edge (a tie's strength, such as the number of co-offences).
nodes	NULL, or every node's label once: the node order of the results and the way to include isolated nodes.
res	The resolution: one number, at least 0.

Value

The modularity, one number.

References

Newman, M. E. J. and Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, 69(2), 026113. doi:10.1103/PhysRevE.69.026113

Examples

```
ntmod(c(1, 1, 2, 4, 4, 5, 3), c(2, 3, 3, 5, 6, 6, 4),
      memb = c(1, 1, 1, 2, 2, 2))
```

---

oaxbd *Oaxaca-Blinder decomposition of a gap between two groups*

Description

Splits the gap in mean outcome between two groups (a sentence length between two groups of defendants, say) into the part their covariates' means account for and the part they do not, by least squares in each group (Oaxaca 1973; Blinder 1973), with delta-method standard errors (Jann 2008).

Usage

```
oaxbd(y, x, grp, ref = 1, conf = 0.95)
```

Arguments

y	The outcomes: finite numbers, one per case.
x	The covariates: NULL (the constant only), a numeric vector, a numeric matrix with one row per case, or a data frame of numeric columns; a constant is added. Factors are refused: code them as indicator columns.
grp	The group of each case: labels, exactly two groups, each with more cases than coefficients.
ref	Group B, whose coefficients price the explained part of the first decomposition: its label, or its position from 1 among the two (in the order of grp); the other is group A.
conf	The confidence level of the intervals, strictly between 0 and 1.

Details

With `D` the difference of the covariate means (group A minus group B, the constant first) and `Delta` the difference of the coefficients, the gap is exactly `D'b_B + xbar_A'Delta` (explained at B's coefficients, `exp`, and unexplained, `unexp`), and equally `D'b_A + xbar_B'Delta` (`expa`, `unexpa`), and `D'b_B + xbar_B'Delta + D'Delta` (endowments, coefficients and their interaction). The two explained parts differ by exactly the interaction, so how much of a gap is "explained" depends on which group's coefficients are taken as the standard: `oaxbd()` reports both. `by` splits the first decomposition by covariate; its unexplained lines are not identified one by one: moving a covariate's origin by `c` for everyone moves `c` times its `shift` between that line and the constant's, leaving the total alone (Oaxaca and Ransom 1999), and for a factor coded as indicators the split depends on the category left out. See `proof("oaxbd")`.

The standard errors treat the two groups, and the means and coefficients within each, as independent, and leave out a term of smaller order, as Jann (2008) does. The unexplained part is what the model's covariates do not account for. It is not a measure of discrimination: it holds every influence the covariates miss, and the covariates can themselves carry earlier unequal treatment.

Value

A list: `groups`, the labels of `a` and `b`; `means`, a data frame with one row per term (`const` first): `term`, `a`, `b`; `coef`: `term`, `a`, `ase`, `b`, `bse`; `dec`, one row per part (`gap`, `exp`, `unexp`, `expa`, `unexpa`, `endow`, `coef`, `inter`): `part`, `est`, `se` and the normal interval `lo`, `hi`; `by`: `term`, `exp`, `expse`, `unexp`, `unexpse` and `shift`.

References

Oaxaca, R. (1973). Male-female wage differentials in urban labor markets. *International Economic Review*, 14(3), 693-709. doi:10.2307/2525981

Blinder, A. S. (1973). Wage discrimination: reduced form and structural estimates. *Journal of Human Resources*, 8(4), 436-455. doi:10.2307/144855

Oaxaca, R. L. and Ransom, M. R. (1999). Identification in detailed wage decompositions. *Review of Economics and Statistics*, 81(1), 154-157. doi:10.1162/003465399767923908

Jann, B. (2008). The Blinder-Oaxaca decomposition for linear regression models. *The Stata Journal*, 8(4), 453-479. doi:10.1177/1536867X0800800401

Examples

```
set.seed(3)
g <- rep(c("a", "b"), each = 200)
sev <- rnorm(400, ifelse(g == "a", 5, 4))
pri <- rpois(400, ifelse(g == "a", 2, 1))
len <- 2 + 0.8 * sev + 0.5 * pri + ifelse(g == "a", 1, 0) + rnorm(400)
r <- oaxbd(len, data.frame(sev, pri), g, ref = "b")
r$dec
```

orrisk *Risks behind an odds ratio*

Description

An odds ratio is not a relative risk: it lies further from 1, and by more the more common the outcome. Given the odds ratio, the overall rate of the outcome and the share exposed, this returns the two risks and the relative risk they imply. See `proof("orrisk")`.

Usage

```
orrisk(or, rate, share)
```

Arguments

`or`	The odds ratio: one positive number.
`rate`	The overall rate of the outcome, strictly between 0 and 1.
`share`	The exposed share of the population, strictly between 0 and 1.

Value

A named numeric vector: `exposed` and `unexposed` (the two risks), `rr` (their ratio) and `or`. A risk closer to 1 than a double can hold is returned as 1.

Examples

```
orrisk(or = 3, rate = 0.4, share = 0.5)
```

outtst *Outcome test*

Description

The stop rate and the hit rate of a group searched whenever the chance of guilt exceeds a threshold. Hit rates compare averages, not thresholds: two groups held to the same threshold can show different hit rates, and a group held to a lower threshold can show the higher one (infra-marginality). See `proof("outtst")`.

Usage

```
outtst(q, t, w = NULL)
```

Arguments

q	The chances of guilt in the group: numbers in the interval 0 to 1.
t	The search threshold: one number.
w	NULL for equal weights, or one non-negative weight per element of q with a positive total; when both are named with unique names, w is matched to q by name.

Value

A named numeric vector: stop (the weighted share searched) and hit (the weighted mean chance of guilt among them; NaN when nobody is searched).

Examples

```
outtst(q = c(0.1, 0.5, 0.9), t = 0.3)
```

---

patidx *Paternity index*

Description

The weight of the DNA evidence that an alleged father is the father of a child, against an unrelated man, locus by locus and overall. With the mother's genotype the index uses the allele the child must have received from the father; without it the index is the parent-child kinidx(). There is no allowance for mutation, so one inconsistent locus gives an index of 0. See proof("patidx").

Usage

```
patidx(child, father, freq, mother = NULL)
```

Arguments

child, father	Genotypes: two-column matrices of allele labels, one row per locus, or two alleles for one locus.
freq	The population frequencies of the child's alleles, same shape.
mother	NULL, or the mother's genotype in the same form. When all the tables have row names, their rows are matched to child by locus name; otherwise by position.

Value

A list: locus, a data frame with locus and idx (NaN where the mother's genotype rules her out as mother); log10idx; and post, the probability of paternity at even prior odds.

Examples

```
patidx(child = c(12, 14), father = c(14, 15), freq = c(0.1, 0.2),
       mother = c(12, 12))
```

proof *Machine-checked proofs behind the package*

Description

Names the 'Lean' theorem each function rests on. The theorems ship with the package: `file` is the path below `system.file(package = "crimlab")`, and each file compiles against 'Mathlib'.

Usage

```
proof(fn = NULL)
```

Arguments

`fn`	`NULL` for every function, or function names.

Value

A data frame: `fn`, `theorem` and `file`.

Examples

```
proof("vodtst")
```

relchk *Which labelling reproduces a reported contrast?*

Description

A reported odds ratio, risk ratio or risk difference depends on which group is the reference, which outcome is the event and which margin of the table holds the groups. `relchk()` recomputes the contrast from the table under every such reading and says which ones reproduce the reported value: the labelling the report states, another one, or none.

Usage

```
relchk(
  x,
  value,
  type = "or",
  group = NULL,
  ref = NULL,
  event = NULL,
  digits = NULL,
  tol = NULL
)
```

Arguments

x	The table of counts, groups in rows and outcomes in columns: a matrix, a 2-d table, a data frame of numeric columns, or a list of numeric vectors of one length (the columns). At least 2 x 2; counts finite and at least 0 with no NA, no row or column all 0, and no label missing or given twice. Rows and columns without names are labelled "1", "2", ...
value	The reported value: one finite number.
type	The kind of contrast the report claims: "or" (odds ratio), "rr" (risk ratio) or "rd" (risk difference).
group, ref, event	The labelling the report claims, each NULL (not stated) or a label (matched exactly, case included; a factor by its label) or a position: group and ref name rows of x, event a column.
digits	The decimals value was rounded to: a reading matches when it lies within half a unit of the last decimal, at either end, so that both rounding conventions at the half are accepted. By default the fewest decimals that print value exactly (give digits = 2 for a value reported as 2.30). Give digits or tol, not both.
tol	Instead of digits, the largest difference allowed: one number from 0. Either way, a reading also matches when it differs only in the last few bits.

Details

A reading compares group group with group ref for outcome event, the other outcomes pooled as non-events: with a and b the events and non-events of group, c and d those of ref, the odds ratio is a d / (b c), the risk ratio (a / (a + b)) / (c / (c + d)) and the risk difference a / (a + b) - c / (c + d). Every ordered pair of groups and every outcome is read, with the groups in the rows and, the table transposed, in the columns, and all three kinds of contrast.

No correction is made for zero cells: a reading whose denominator is 0 is Inf (a positive numerator) or NaN (0 over 0) and matches nothing. Counts need not be whole numbers (weighted totals are read as they are).

For a 2 x 2 table the odds ratio inverts when the reference group or the event changes and is unchanged when both change or the table is transposed; the risk ratio inverts with the reference group, and the risk difference changes sign with either; proof("relchk") names the Lean theorems.

Value

An object of class relchk, a list: verdict ("holds": a reading the claim allows matches; "other": only readings it does not allow match; "none"), ok (TRUE when the claim holds), value, tol (the difference allowed), digits (NA when tol was given or no decimals print value exactly; then tol is 0), type, claim (the stated group, ref and event, NA where not stated) and reads, a data frame with one row per reading: grp ("rows" or "cols": the margin holding the groups), type, group, ref, event, value, match and claim (TRUE for the readings the claim allows: groups in rows, the kind type, and the stated labels).

Examples

```
tab <- matrix(c(20, 10, 80, 90), 2,
              dimnames = list(c("A", "B"), c("yes", "no")))
# 2.25 is the odds ratio of A against B for "yes"
relchk(tab, 2.25, ref = "B", event = "yes")
# 0.44 is the same contrast read with the reference group swapped
relchk(tab, 0.44, ref = "B", event = "yes")
# a risk ratio reported as an odds ratio
relchk(tab, 2.0, type = "or", digits = 1)
```

`rfor` *Random forest for regression and probabilities*

Description

Grows a random forest (Breiman 2001) on `x` and `y` and returns its mean prediction at the rows of `newx` and its out-of-bag prediction at each row of `x`. Each tree is grown on a bag of rows drawn from `x`, as deep as `minnode` allows, choosing at each node the best of `mtry` columns drawn at random by the reduction in the sum of squares; for a 0/1 outcome this is the Gini split, and the forest's mean is a probability, as in a probability forest of 'ranger'.

Usage

```
rfor(
  x,
  y,
  newx = x,
  ntree = 500,
  mtry = NULL,
  minnode = 5,
  frac = 1,
  repl = TRUE,
  hon = FALSE,
  imb = 0,
  seed = 1
)
```

Arguments

`x`	The predictors: a numeric matrix with at least two rows, no missing values.
`y`	The outcome: one finite number per row of `x`; 0 or 1 for a probability.
`newx`	The rows to predict: a numeric matrix with the columns of `x`.
`ntree`	The number of trees: one whole number, at least 1.
`mtry`	The columns tried at each split: `NULL` for `min(ceiling( sqrt(p) + 20), p)`, `p` the number of columns (the rule of 'grf'), or a whole number from 1 to the number of columns.

`minnode`	Nodes of at most this many rows are not split: one whole number, at least 1 (5 is the default of 'ranger' for regression, 10 for probabilities).
`frac`	The size of each bag as a share of the rows of `x`: above 0, at most 1.
`repl`	`TRUE` to draw bags with replacement, `FALSE` without.
`hon`	`TRUE` for honest trees (Wager and Athey 2018): each bag is shuffled, its first half places the splits and its second half sets the leaf means (a leaf without such rows takes the mean of its deepest ancestor that has some). Needs bags of at least two rows.
`imb`	Each child of a split must hold at least this share of its node's rows (and one row): one number from 0 to below 0.5 ('grf' uses 0.05).
`seed`	The seed of the draws: one number, at least 0. The same seed and data give the same forest.

Details

Each column is cut once, at its own order statistics, into at most 256 bins: every split between neighbouring values when a column has at most 256 distinct values (as an exact search finds), and splits at 255 quantiles otherwise. A node's best split then comes from one pass over its rows, and the forest depends on each column only through the order of its values: a strictly increasing transform of a column changes nothing. The forest is not bit-for-bit that of 'ranger' (it draws its own random numbers and uses quantile bins on columns of many values); tests compare their errors on known functions instead.

Value

A list: `pred`, the predictions at the rows of `newx`, and `oob`, the out-of-bag prediction at each row of `x` (`NaN` for a row that every tree's bag holds).

Defaults

The defaults were chosen by test error against the true function over four data-generating functions (a sparse linear one, Friedman's, a step-and-curve one and a probability), `p` = 20 and 200, ten seeds each (`dev/rftune.R`). Like for like (the same trees, `mtry`, node size, half-samples without replacement, `imb` = `0.05`, honest or not), `rfor()` and 'grf' have test errors within 1 to 3 per cent of each other, inside two standard errors in almost every cell. The one setting that mattered was `mtry`: the rule of 'grf' lowers the error against the square root of `p` by 30 to 50 per cent on the sparse and step functions, so it is the default. Bags of all rows drawn with replacement gave the lowest error with that `mtry` in most cells; honest trees and half-samples raised the error of the predictions (honesty serves inference, not prediction), so both are options, off by default. Nodes of 5 rows are left unsplit, as both packages do.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32. doi:10.1023/A:1010933404324

Wager, S. and Athey, S. (2018). Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523), 1228-1242. doi:10.1080/01621459.2017.1319839

Examples

```
set.seed(1)
x <- matrix(rnorm(400 * 4), 400)
y <- sin(2 * x[, 1]) + x[, 2] + rnorm(400, sd = 0.3)
f <- rfor(x, y, ntree = 100)
mean((f$oob - y)^2)
```

`ripenv` *Envelopes of Ripley's L function and a test of complete spatial randomness*

Description

`L(r) - r` for the data, its pointwise minimum, maximum and mean over `nsim` patterns of as many uniform points in the rectangle, and a global Monte Carlo test on the maximum absolute deviation `T = max_r |L(r) - r|`, whose p-value `(1 + #{T_sim >= T}) / (nsim + 1)` has exact size (Ripley 1977; Diggle 2013). The pointwise envelopes are not a test of the whole curve: each radius alone falls outside them with probability `2 / (nsim + 1)` under randomness.

Usage

```
ripenv(x, y = NULL, win, r, corr = "iso", nsim = 99, seed = NULL)
```

Arguments

`x, y`	The points in planar coordinates (project longitude and latitude with `eqproj()` first): `x` and `y` numeric vectors of equal length, or `x` a matrix or data frame with columns named `x` and `y` (an unnamed two-column matrix is read as `x` then `y`). At least two points, all inside `win`; no missing values.
`win`	The study rectangle `c(xmin, xmax, ymin, ymax)`, in the unit of the coordinates. The intensity is the number of points over its area.
`r`	The radii: non-negative numbers in strictly increasing order, in the unit of the coordinates.
`corr`	The edge correction: `"iso"`, `"border"`, `"modif"` or `"none"`.
`nsim`	The number of simulated patterns: one whole number from 1.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.

Details

Euclidean distance only; `ripk()` takes `metric`.

Value

A list. `env`, a data frame with one row per radius: `r`, `obs` (`L(r) - r` of the data), `lo`, `hi` (the pointwise envelope) and `mean`. `test`, a named vector: `t` (the maximum absolute deviation) and `p` (its Monte Carlo p-value).

References

Diggle, P. J. (2013). *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns* (3rd ed.), chapter 6. CRC Press.

Ripley, B. D. (1977). Modelling spatial patterns. *Journal of the Royal Statistical Society, Series B*, 39(2), 172-212. doi:10.1111/j.25176161.1977.tb01615.x

Examples

```
set.seed(1)
ripenv(runif(100), runif(100), win = c(0, 1, 0, 1),
       r = seq(0.01, 0.25, 0.02), nsim = 39)$test
```

`ripk` *Ripley's K function*

Description

Ripley's K function of a point pattern in the rectangle `win` at radii `r`, and Besag's `L = sqrt(K / pi)`. Under complete spatial randomness `K(r) = pi r^2` and `L(r) = r`. Four estimators: `"none"` (no edge correction), `"border"` (only points at least `r` from the edge are centres), `"modif"` (the modified border estimator, with the area of the eroded window; `NaN` once `r` reaches half a side) and `"iso"` (Ripley's isotropic correction, each pair weighted by the inverse of the share of the circle through the other point that lies in the rectangle, computed exactly). The normalisation is `n (n - 1)`, as in spatstat.

Usage

```
ripk(x, y = NULL, win, r, corr = "iso", metric = "euclid")
```

Arguments

`x, y`	The points in planar coordinates (project longitude and latitude with `eqproj()` first): `x` and `y` numeric vectors of equal length, or `x` a matrix or data frame with columns named `x` and `y` (an unnamed two-column matrix is read as `x` then `y`). At least two points, all inside `win`; no missing values.
`win`	The study rectangle `c(xmin, xmax, ymin, ymax)`, in the unit of the coordinates. The intensity is the number of points over its area.
`r`	The radii: non-negative numbers in strictly increasing order, in the unit of the coordinates.
`corr`	The edge correction: `"iso"`, `"border"`, `"modif"` or `"none"`.
`metric`	The distance: `"euclid"` or `"manhattan"` (`|dx| + |dy|`, a street grid aligned with the axes), as in `spwt()` and `jtsurf()`. Under Manhattan distance `K(r) = 2 r^2` and `L = sqrt(K / 2)` under randomness; the border and modified border corrections carry over (a Manhattan ball of radius `r` lies in the rectangle exactly when its centre is at least `r` from every edge), while the isotropic correction, the share of a circle, has no Manhattan form and is refused.

Value

A data frame with one row per radius: `r`, `k` (in the square of the coordinates' unit) and `l` (`sqrt(k / pi)`, or `sqrt(k / 2)` under Manhattan distance).

References

Ripley, B. D. (1976). The second-order analysis of stationary point processes. *Journal of Applied Probability*, 13(2), 255-266. doi:10.2307/3212829

Ripley, B. D. (1977). Modelling spatial patterns. *Journal of the Royal Statistical Society, Series B*, 39(2), 172-212. doi:10.1111/j.25176161.1977.tb01615.x

Examples

```
set.seed(1)
ripk(runif(100), runif(100), win = c(0, 1, 0, 1), r = seq(0, 0.25, 0.05))
```

---

rnkres *Which pairs a noisy ranking orders*

Description

People (or areas, or courts) ranked by estimated scores are ordered correctly whatever the noise only when their estimates differ by more than their two interval half-widths together: below that, admissible errors can reverse them, above it none can. `rnkres()` says which pairs that holds for, and the share it fails for; the resolution of the ranking is twice the largest half-width. A league table whose intervals are wide orders few of its pairs (Goldstein and Spiegelhalter 1996). See `proof("rnkres")`.

Usage

```
rnkres(est, hw)
```

Arguments

`est`	The estimates: finite numbers, named or not, at least two.
`hw`	The half-width of each estimate's interval: one number from 0, or one per estimate.

Value

A list: `id`, a logical matrix, `TRUE` where the pair's order is identified (`NA` on the diagonal), with the estimates' names; `npair`, `nunid` (the pairs not identified), `share` (their share) and `res`.

References

Goldstein, H. and Spiegelhalter, D. J. (1996). League tables and their limitations: statistical issues in comparisons of institutional performance. *Journal of the Royal Statistical Society, Series A*, 159(3), 385-443. doi:10.2307/2983325

Examples

```
rnkres(c(a = 0.2, b = 0.35, c = 0.8), hw = c(0.1, 0.1, 0.05))
```

`rnnhcl` *Risk-adjusted nearest-neighbour hierarchical clustering*

Description

Nearest-neighbour clustering (as `nnhclu()`, first order) with the threshold distance adjusted for a baseline population at risk, so that incidents dense only because people are dense there are not called a cluster. In `nnhclu()` the threshold is the mean nearest-neighbour distance under complete spatial randomness plus `z` standard errors, `0.5 sqrt(A / n) + z sqrt((4 - pi) A / (4 pi)) / n`, which is `f / sqrt(lambda)` with `lambda = n / A` and `f = 0.5 + z sqrt((4 - pi) / (4 pi n))`. Here the uniform intensity is replaced by the intensity the `n` incidents would have if they followed the baseline: `lambda_i = n b(p_i) / B`, where `b` is the normal-kernel density (bandwidth `h`) of the baseline units with sizes `bw` and `B` is their total, and two incidents `i` and `j` are linked when `|p_i - p_j| < f / sqrt((lambda_i + lambda_j) / 2)`. Taking the mean of the two intensities is this function's reading of "the threshold at the pair's baseline density" (the published description does not say how the two ends combine); a uniform baseline gives back `nnhclu()`'s threshold. Grouping is `nnhclu()`'s first order; higher orders are not formed.

Usage

```
rnnhcl(
  x,
  y = NULL,
  base,
  bw = NULL,
  h,
  p = 0.5,
  minpts = 10,
  k = 1,
  c = 1,
  lonlat = FALSE
)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.

base	The baseline units: coordinates as for x (zone centroids, or points), in the same system as x (degrees when lonlat = TRUE).
bw	NULL for size 1 each, or the baseline sizes (populations), numbers from 0 with a positive sum.
h	The bandwidth of the baseline's normal kernel, positive, in the unit of the coordinates (metres when lonlat = TRUE).
p	The one-tailed probability that a pair is as close by chance: one number strictly between 0 and 1.
minpts	The fewest points a cluster keeps: a whole number from 2.
k	The divisor offset: 0, 1 or 2; the total weight must exceed it.
c	The multiplier of each cluster's standard deviational ellipse (see sdell()).
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A list: cl, each point's cluster (a row of clusters) or NA; clusters, a data frame of points, mx, my, wx, wy, a, b and angle as in nnhclu(); thr, each point's own threshold f / sqrt(lambda_i).

References

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3), 241-254. doi:10.1007/BF02289588

Clark, P. J. and Evans, F. C. (1954). Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35(4), 445-453. doi:10.2307/1931034

Examples

```
set.seed(1)
town <- cbind(x = rnorm(400, 0, 1), y = rnorm(400, 0, 1))
inc <- rbind(town[1:60, ], cbind(x = rnorm(15, 3, 0.1),
                                 y = rnorm(15, 3, 0.1)))
r <- rnnhcl(inc, base = town, h = 0.5, minpts = 5)
r$clusters
```

---

rptc *Constant reporters of a reporting panel*

Description

The units (agencies reporting to a national crime statistics programme, say) that report at least minrep months in every year from y0 to y1. A year without a record of the unit, or with too few months, disqualifies it, whatever other units do that year. Counts over time are comparable only on such a constant panel; a trend from all reporters mixes change in crime with change in who reports.

Usage

```
rptc(unit, year, rep, y0, y1, minrep = 12)
```

Arguments

unit	The unit of each record: labels, no missing values.
year	The year of each record: whole numbers.
rep	The months reported in each record: whole numbers.
y0, y1	The first and last years, y0 <= y1, from 0.
minrep	The fewest months that count as reporting: a whole number from 0.

Value

A data frame with one row per unit (sorted): unit and const (TRUE for a constant reporter).

References

Maltz, M. D. (1999). *Bridging Gaps in Police Crime Data*. Bureau of Justice Statistics, NCJ 176365.

Examples

```
rptc(unit = c("a", "a", "b", "b", "b"), year = c(2020, 2021, 2020, 2021,
     2021), rep = c(12, 12, 12, 6, 12), y0 = 2020, y1 = 2021)
```

---

rswp *Rosenbaum sensitivity bounds for matched pairs*

Description

How strong would hidden bias have to be to explain away a matched-pairs result? gam is Gamma, the most the odds of treatment can differ within a pair because of an unobserved covariate (1: no hidden bias). rswp() gives the bounds on the one-sided p-value of Wilcoxon's signed-rank test at each Gamma (Rosenbaum 1987), from the normal approximation, and the Gamma at which the upper bound reaches alpha. rshl() gives the bounds on the Hodges-Lehmann estimate of the effect (Rosenbaum 1993), computed exactly from the Walsh averages, with no grid.

Usage

```
rswp(d, gam = 1, alpha = 0.05)

rshl(d, gam = 1)
```

Arguments

d	The treated-minus-control differences: finite numbers, one per pair.
gam	Values of Gamma: numbers from 1.
alpha	The level for the critical Gamma, strictly between 0 and 0.5.

Value

`rswp()`: a list of `p` (a data frame of `gam`, `plo`, `pup`), `gcrit` (the critical Gamma, below 1 when the test is not significant without bias) and `t` (the signed-rank statistic). `rshl()`: a data frame of `gam`, `lo`, `hi`. `NaN` when every difference is 0.

References

Rosenbaum, P. R. (1987). Sensitivity analysis for certain permutation inferences in matched observational studies. *Biometrika*, 74(1), 13-26. doi:10.1093/biomet/74.1.13

Rosenbaum, P. R. (1993). Hodges-Lehmann point estimates of treatment effect in observational studies. *Journal of the American Statistical Association*, 88(424), 1250-1253. doi:10.1080/01621459.1993.10476405

Examples

```
set.seed(1)
d <- rnorm(50, 0.5)
rswp(d, gam = c(1, 1.5, 2))
rshl(d, gam = c(1, 1.5, 2))
```

`rtmhs` *Regression to the mean at selected hot spots*

Description

Places chosen because their count in a first period exceeds a cut show a fall in the next period even with no treatment and no trend: if the two periods are exchangeable, the selected places' mean change is never positive, and places chosen for being low never fall on average. This reports the change at the selected places next to the same statistic on the data together with its period-swapped copy, which is exchangeable by construction and so never positive; the excess of the first over the second is the part of the fall that selection alone does not explain. A treatment effect needs a control group selected the same way. See `proof("rtmhs")`.

Usage

```
rtmhs(x1, x2, cut, w = NULL)
```

Arguments

x1	Counts by place in the first period: finite numbers.
x2	Counts by place in the second period: same length; when x1, x2 (and w) are all named with unique names, they are matched by name.
cut	The selection cut: places with x1 > cut are selected.
w	NULL for equal weights, or one non-negative weight per place.

Value

A named numeric vector: nsel (the weight selected), change (the weighted mean of x2 - x1 at the selected places), mirror (the mean of x1 - x2 at places with x2 > cut), sym (the change in the data with its swapped copy; never positive), excess (change - sym), low (the mean change at places with x1 < cut) and lowsym (its swapped-copy version; never negative). NaN where no place qualifies.

References

Barnett, A. G., van der Pols, J. C. and Dobson, A. J. (2005). Regression to the mean: what it is and how to deal with it. *International Journal of Epidemiology*, 34(1), 215-220. doi:10.1093/ije/dyh299

Examples

```
set.seed(5)
mu <- rgamma(300, 2, 0.5)
x1 <- rpois(300, mu)
x2 <- rpois(300, mu)
rtmhs(x1, x2, cut = quantile(x1, 0.9))
```

---

scrdsp *A risk score's spread across groups*

Description

Each group's number of cases, mean, median and standard deviation of a score, each mean's gap to a reference group with its interval, and the one-way analysis of variance of the score on group, with Welch's (1951) test, which does not assume equal variances.

Usage

```
scrdsp(score, group = NULL, ref = NULL, conf = 0.95)
```

Arguments

score	The scores: finite numbers, one per case. A clpre list from clprep() is taken too: its elements fill the arguments by name.
group	The group of each case: labels.
ref	The reference group: NULL for the group with the lowest mean (the first on ties), its label, or its position from 1.
conf	The confidence level of the intervals, strictly between 0 and 1.

Details

The analysis of variance is the standard one (stats::anova() of stats::lm()): a group with one case adds to the between-group sum of squares and nothing to the within-group one. The gaps' intervals use the pooled within-group variance on its degrees of freedom. Welch's test needs two or more cases and some spread in every group, and is NaN otherwise.

A gap in mean scores is not by itself a sign of bias: a score built to predict an outcome differs between groups whose rates of that outcome differ. Read it with calaud(), which compares the scores with what was recorded, or fair(), which compares them with outcomes case by case.

Value

A list: tab, a data frame with one row per group: group, n, mean, median, sd (NaN for one case), gap, se, lo and hi (NaN for the reference); anova and welch, named vectors f, df1, df2 and p; spread (the largest mean minus the smallest); ref.

References

Welch, B. L. (1951). On the comparison of several mean values: an alternative approach. *Biometrika*, 38(3-4), 330-336. doi:10.1093/biomet/38.34.330

Examples

```
set.seed(7)
g <- rep(c("x", "y", "z"), c(40, 40, 20))
s <- rnorm(100, ifelse(g == "x", 0, ifelse(g == "y", 0.5, 1)))
scrdsp(s, g)
```

---

sdell *Standard deviational ellipse*

Description

The ellipse whose axes lie along the eigenvectors of the weighted covariance matrix of the points about their mean centre, with semi-axes c times the square roots of its eigenvalues (Wang, Shi and Miao 2015). It is the ellipse of the rotated-axes formulas of Lefever (1926) and Ebdon (1988) once the same divisor is used; texts differ only in the divisor (k) and in the multiplier (some multiply by sqrt(2)). The curve of the standard deviation by direction that Lefever drew is not this ellipse

(Gong 2002); the ellipse is returned. For bivariate normal points the ellipse with `c = sqrt(-2 * log(1 - p))` covers a share `p` of them: `c = 1` covers 39.3 per cent, `c = 2` 86.5 per cent. The parallel-axis identity behind the ellipse's centre is proved in Lean: see `proof("sdell")`.

Usage

```
sdell(x, y = NULL, w = NULL, k = 1, c = 1, lonlat = FALSE)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y` `NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`w`	`NULL` for equal weights, or one non-negative weight per point (counts at a location, say), with a positive sum.
`k`	The divisor offset: 0, 1 or 2; the total weight must exceed it.
`c`	The multiplier of the standard deviations: one positive number.
`lonlat`	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A named numeric vector: `mx`, `my` (the mean centre, in degrees when `lonlat = TRUE`), `a` and `b` (the semi-major and semi-minor axes, `a >= b`, in the unit of the coordinates or metres), `angle` (of the major axis, in radians anticlockwise from the x axis (east), in `[0, pi)`; 0 when `a == b`) and `area` (`pi * a * b`).

References

Wang, B., Shi, W. and Miao, Z. (2015). Confidence analysis of standard deviational ellipse and its extension into higher dimensional Euclidean space. *PLoS ONE*, 10(3), e0118537. doi:10.1371/journal.pone.0118537

Gong, J. (2002). Clarifying the standard deviational ellipse. *Geographical Analysis*, 34(2), 155-167. doi:10.1111/j.15384632.2002.tb01082.x

Examples

```
set.seed(2)
x <- rnorm(200)
sdell(x, 0.5 * x + rnorm(200, sd = 0.3), c = 2)
```

`sellab` *Evaluating a release rule from the released alone*

Description

Failures (a missed court date, a new arrest) are seen only for the defendants a judge released. A proposed rule that releases only people the judge released is scored exactly from their outcomes; one that releases people the judge detained has a failure rate known only up to the worst and best cases for them, an interval whose width is the share of the rule's releases the judge detained. Both ends can occur. The naive score on the rule's observed releases alone always lies inside the interval, but reports one point of it as if it were known. See `proof("sellab")`.

Usage

```
sellab(y, rel, rule, w = NULL)
```

Arguments

`y`	Failures: 1 or 0 for each released defendant; values for detained defendants are ignored and may be `NA`.
`rel`	Released by the judge: `TRUE` or `FALSE` for each defendant. When `y`, `rel`, `rule` (and `w`) are all named with unique names, they are matched by name; otherwise by position.
`rule`	Released by the rule: `TRUE` or `FALSE`, same length.
`w`	`NULL` for equal weights, or one non-negative weight per defendant.

Value

A named numeric vector: `obs` (the failure rate among the judge's releases), `unobs` (the share of the rule's releases the judge detained), `lower` and `upper` (the bounds on the rule's failure rate), `width` (`upper - lower`, equal to `unobs`) and `naive` (the failure rate on the rule's releases that were observed; `NaN` when none were).

References

Lakkaraju, H., Kleinberg, J., Leskovec, J., Ludwig, J. and Mullainathan, S. (2017). The selective labels problem. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 275-284. doi:10.1145/3097983.3098066

Examples

```
set.seed(4)
risk <- runif(500)
y <- rbinom(500, 1, risk)
sellab(y, rel = risk < 0.7, rule = risk < 0.8)
```

`sirex` *Indirect standardisation: standardised ratios with shrinkage*

Description

The events observed in each group (an area, a police service) against those expected from stratum rates, the standardised incidence ratio (SIR) with its exact Poisson interval, funnel-plot limits for a ratio of 1 at its expected count, and the ratio shrunk towards the overall one by the empirical Bayes estimate of Marshall (a gamma prior fitted by moments), so that groups with little exposure are not ranked on noise.

Usage

```
sirex(x, n, strat, grp, rates = NULL, level = 0.95)
```

Arguments

`x`	The events: numbers from 0; or a `clpre` list from `clprep()`.
`n`	The exposure of each row: positive numbers.
`strat`	The stratum of each row: labels.
`grp`	The group of each row: labels.
`rates`	`NULL` for the stratum rates pooled over all groups, or the reference rate of each stratum: numbers from 0, named by stratum or in the order of the strata.
`level`	The confidence level of the intervals: one number strictly between 0 and 1.

Value

An object of class `sirex`, a list: `sir`, a data frame (`group`, `obs`, `expd`, `sir`, `lo`, `hi`, `eb` the shrunk ratio, `shr` the share of the prior in it, `flo`, `fhi` the funnel limits, `out` `TRUE` outside them); `prior`, the prior's mean and variance (a variance of 0 when the ratios vary no more than chance, and then every `eb` is the mean); and `level`.

References

Marshall, R. J. (1991). Mapping disease and mortality rates using empirical Bayes estimators. *Journal of the Royal Statistical Society, Series C*, 40(2), 283-294. doi:10.2307/2347593

Examples

```
d <- data.frame(area = rep(c("N", "S", "E"), each = 2),
                age = rep(c("young", "old"), 3),
                x = c(12, 3, 30, 9, 4, 1),
                n = c(4000, 2000, 8000, 3000, 1500, 900))
sirex(d$x, d$n, d$age, d$area)
```

skp *Scott-Knott grouping of measurements*

Description

Splits sorted measurements into groups that differ (Scott and Knott 1974) when the variance of one measurement is known, as in the forensic glass version of Triggs, Curran, Buckleton and Walsh (1997): a segment is split at the cut with the largest between-groups sum of squares when that is significant at `alpha`, judged against `nsim` simulated samples of independent normal values with the known variance (a test of exact size under that null).

Usage

```
skp(x, s2, alpha = 0.05, nsim = 999, seed = NULL)
```

Arguments

`x`	Measurements: finite numbers (refractive indices, say).
`s2`	The variance of one measurement, on the scale of `x`: positive.
`alpha`	The level of each split's test, strictly between 0 and 1.
`nsim`	The number of simulated samples per test: a whole number from 1.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.

Value

An integer vector of group labels, one per measurement in input order, numbered from 1 in increasing order of the values.

References

Scott, A. J. and Knott, M. (1974). A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3), 507-512. doi:10.2307/2529204

Triggs, C. M., Curran, J. M., Buckleton, J. S. and Walsh, K. A. J. (1997). The grouping problem in forensic glass analysis: a divisive approach. *Forensic Science International*, 85(1), 1-14. doi:10.1016/S03790738(96)020373

Examples

```
set.seed(1)
x <- c(rnorm(5, 1.5180, 4e-5), rnorm(5, 1.5195, 4e-5))
skp(x, s2 = 4e-5^2, nsim = 199, seed = 1)
```

`sntbnd` *Bounds on a sentence's effect without assumptions*

Description

Each person receives one of two sentences and shows the outcome (say, reconviction) under that one only. With no assumption about who receives which, the chance of the outcome under a sentence is known only up to an interval whose width is the share who received the other sentence, and the effect of one sentence against the other lies in an interval of width exactly 1 that always contains 0: observational data alone cannot even sign it. See `proof("sntbnd")`.

Usage

```
sntbnd(y, z, treat = NULL, w = NULL)
```

Arguments

`y`	Outcomes: 0 or 1 for each person.
`z`	Sentences: a vector with exactly two distinct values, same length: character, numbers, logical or a factor (its labels; level order and whether it is ordered are ignored), not dates nor a labelled vector of codes. When `y`, `z` (and `w`) are all named with unique names, they are matched by name.
`treat`	The value of `z` taken as the treatment, matched exactly. `NULL` is allowed only where the treatment is explicit in `z`: `TRUE` for a logical `z`, `1` for a `z` of 0 and 1; otherwise it must be given, since a treatment chosen by sorting the values would be arbitrary.
`w`	`NULL` for equal weights, or one non-negative weight per person with a positive total.

Value

A data frame with three rows, the comparison sentence, the treatment and `effect` (treatment minus comparison): `term`, `share` (the weighted share sentenced; `NA` for the effect), `lower`, `upper` and `observed` (the observed rate under each sentence and their difference).

References

Manski, C. F. and Nagin, D. S. (1998). Bounding disagreements about treatment effects: a case study of sentencing and recidivism. *Sociological Methodology*, 28(1), 99-137. doi:10.1111/0081-1750.00043

Examples

```
set.seed(1)
z <- sample(c("community", "custody"), 500, TRUE, c(0.6, 0.4))
y <- rbinom(500, 1, ifelse(z == "custody", 0.55, 0.35))
sntbnd(y, z, treat = "custody")
```

`sptst` *Andresen's spatial point pattern test*

Description

Compares where two point patterns fall over the same areal units (Andresen 2009): for each unit, the difference between its share of the test pattern `b` and its share of the base pattern `a`, with a bootstrap p-value from multinomial resamples of each pattern; the S index is the share of units whose difference is not significant (1: the patterns are alike everywhere).

Usage

```
sptst(a, b, nboot = 200, seed = NULL, alpha = 0.05, scale = 100)
```

Arguments

`a`, `b`	The counts of the base and the test pattern by unit: whole numbers from 0; `NA` in either leaves the unit out (counted, never taken as 0).
`nboot`	The number of bootstrap resamples: a whole number from 1.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.
`alpha`	The level of each unit's test, strictly between 0 and 1.
`scale`	The multiplier of the differences (100: percentage points).

Value

A list: `units`, a data frame with one row per unit of `diff`, `p` and `flag` (the sign of a significant difference, 0 otherwise; `NaN` without a p-value); `glob`, a named vector: units `left` out, `empty` in both, `eligible` and `flagged`, the S index `s`, its `z` against `1 - alpha`, the exact `p` of so many flags and `nboot`.

References

Andresen, M. A. (2009). Testing for similarity in area-based spatial patterns: a nonparametric Monte Carlo approach. *Applied Geography*, 29(3), 333-345. doi:10.1016/j.apgeog.2008.12.004

Examples

```
set.seed(1)
a <- rpois(30, 10)
b <- rpois(30, 10)
sptst(a, b, nboot = 199, seed = 1)$glob
```

`spwt` *Spatial weights*

Description

Builds the neighbours and weights the autocorrelation statistics (`gearc()`, `gorg()`, `lmorn()`, `locgi()`) use: the `k` nearest neighbours of each point (`type = "knn"`, ties broken by the lower row number), the points within distance `d` (`"band"`, the distance itself included), or a contiguity list `nb` (`"contig"`, one element per unit giving its neighbours' numbers, as spdep's `poly2nb()` returns). Weights are 1, or the inverse distance to the power `idw`; `style = "row"` divides each unit's weights by their sum (a unit without neighbours keeps none), `"raw"` leaves them.

Usage

```
spwt(
  x = NULL,
  y = NULL,
  type = "knn",
  k = 4,
  d = NULL,
  nb = NULL,
  idw = 0,
  style = "row",
  metric = "euclid",
  lonlat = FALSE
)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y` `NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
`type`	`"knn"`, `"band"` or `"contig"`.
`k`	For `"knn"`: the number of neighbours, from 1 to the number of points less 1.
`d`	For `"band"`: the distance, positive, in the unit of the coordinates (metres when `lonlat = TRUE`).
`nb`	For `"contig"`: a list with one element per unit, each a vector of neighbour numbers (from 1; `0` alone for none).
`idw`	The power of the inverse distance: 0 (binary weights) or more; above 0 it needs the points.

style	"row" or "raw".
metric	"euclid" or "manhattan" (|dx| + |dy|).
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A list: n (the number of units), and one element per neighbour pair, in order of unit then neighbour: i, j (integer unit numbers), d (their distance, NA for contiguity without points) and w (the weight). x may be NULL for contiguity with binary weights.

References

Cliff, A. D. and Ord, J. K. (1981). *Spatial Processes: Models and Applications*. Pion, London.

Examples

```
set.seed(1)
p <- cbind(x = runif(20), y = runif(20))
w <- spwt(p, k = 3)
str(w)
spwt(nb = list(2, c(1, 3), 2), type = "contig")$w
```

---

stacel *Search-circle hot-spot areas*

Description

Hot-spot areas from search circles (Block and Block 1995): circles of radius r are centred on the nodes of an nx by ny grid spanning the rectangle win (corners included), the points in each are counted, the circles holding at least two points are ranked by count and the first top are kept; kept circles that share a point are merged, repeatedly, and a hot-spot area is the set of points in a merged group, kept if it has at least minpts. Descriptions differ on the circle's radius and on the grid; both are the caller's choice here.

Usage

```
stacel(
  x,
  y = NULL,
  win,
  nx = 20,
  ny = 20,
  r,
  top = 25,
```

```
  minpts = 5,
  k = 1,
  c = 1,
  count = NULL
)
```

Arguments

x, y	The points in planar coordinates (project longitude and latitude with eqproj() first): x and y numeric vectors of equal length, or x a matrix or data frame with columns named x and y (an unnamed two-column matrix is read as x then y). At least two points, all inside win; no missing values.
win	The rectangle the grid spans, c(xmin, xmax, ymin, ymax); points may lie outside it.
nx, ny	The numbers of grid nodes across and up: whole numbers from 2 to 10000.
r	The search radius, positive, in the unit of the coordinates.
top	The number of busiest circles kept: one whole number from 1.
minpts	The fewest points a hot-spot area keeps: a whole number from 2.
k	The divisor offset: 0, 1 or 2; the total weight must exceed it.
c	The multiplier of each cluster's standard deviational ellipse (see sdell()).
count	NULL for points, or for zones: one whole count from 0 per zone, the points x, y being the zones' centroids. A zone is taken as count incidents at its centroid, which these hot-spot rules treat exactly as coincident points. Zones with count 0 get NA; the random-expectation threshold counts incidents, not zones.

Value

A list. cl, the hot-spot area of each point (zone) (a row of clusters) or NA, in input order. clusters, a data frame with one row per area, by size: points, mx, my (mean centre), wx, wy (centre of minimum distance), a, b and angle (the standard deviational ellipse of its points).

References

Block, R. L. and Block, C. R. (1995). Space, place and crime: hot spot areas and hot places of liquor-related crime. In J. E. Eck and D. Weisburd (eds.), *Crime and Place*, Crime Prevention Studies 4, 145-183. Criminal Justice Press.

Examples

```
set.seed(1)
x <- c(rnorm(30, 2, 0.2), runif(60, 0, 10))
y <- c(rnorm(30, 2, 0.2), runif(60, 0, 10))
stacel(x, y, win = c(0, 10, 0, 10), r = 0.75, top = 5, minpts = 5)$clusters
```

stmavg *Spatial-temporal moving average*

Description

For each event, the mean centre of the span events centred on it in time order (ties in input order), the window shortened at the ends of the series. It traces how the centre of a series of incidents moves over time. It is a descriptive device of the crime-mapping literature with no original paper to cite; the mean centre is that of `cgstat()`.

Usage

```
stmavg(x, y = NULL, t, span = 5, lonlat = FALSE)
```

Arguments

x, y	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y` `NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
t	The time of each event: finite numbers (for dates, `as.numeric()` of them, in days or seconds).
span	The window: one odd whole number from 1.
lonlat	`FALSE` (the default) when `x` and `y` are planar coordinates in one length unit; `TRUE` when they are longitude and latitude in degrees, which are then projected to metres by `eqproj()` about their own centre, and centres are returned in degrees. It is never guessed.

Value

A data frame with one row per event, in input order: `mx`, `my` (the window's mean centre, in degrees when `lonlat = TRUE`) and `n` (events in the window).

Examples

```
stmavg(x = 1:7, y = c(1, 1, 2, 3, 5, 8, 13), t = 7:1, span = 3)
```

`tfrp` *Glass fragments recovered under transfer and persistence*

Description

The probability of recovering 0, 1, ... fragments of glass from a suspect under the transfer, persistence and recovery model of Curran, Triggs, Buckleton, Walsh and Hicks (1998), with uncertain parameters averaged over: the transfer amount and the distance deterministically (by quadrature), the time since the offence exactly, and the loss and recovery rates by `nsim` Monte Carlo draws (exactly when every rate is fixed).

Usage

```
tfrp(
  par,
  deffect = TRUE,
  tdist = "nb",
  ymax = 20,
  nsim = 1000,
  seed = NULL,
  hl = 0.02,
  nq = 256
)
```

Arguments

`par`	The 17 parameters, in order: `lam`, `lsd` (transfer amount N(lam, lsd^2), used as its absolute value), `d` (distance scale), `Q` (share of high-persistence fragments), `l0`, `u0` (first-hour loss of the others, uniform), `ls0`, `us0` (of the high-persistence ones), `lj`, `uj` and `lsj`, `usj` (each later hour), `lR`, `uR` (share recovered), `lt`, `ut` (hours since the offence: fixed when equal) and `r` (the negative binomial probability of the time).
`deffect`	`TRUE` for the distance effect, `FALSE` without it.
`tdist`	The law of the time when `lt < ut`: `"nb"` (negative binomial with size `(lt + ut) / 2` and probability `r`), `"nbtrunc"` (that law on `floor(lt)` to `ceiling(ut)`) or `"unif"` (uniform on those hours).
`ymax`	The largest count reported: a whole number from 0.
`nsim`	The Monte Carlo draws of the rates: a whole number from 1.
`seed`	`NULL` to draw the simulations' seed from R's generator (so `set.seed()` fixes them), or one whole number from 0.
`hl`, `nq`	The quadrature's grid step (1e-4 to 1) and panels (1 to 100000); the defaults resolve the model's usual settings.

Value

A list: `prob`, a data frame of `y` (0 to `ymax`), `p` and `se` (the Monte Carlo standard error, 0 when exact); `cut`, the time mass left out and the intensity mass on the grid.

References

Curran, J. M., Triggs, C. M., Buckleton, J. S., Walsh, K. A. J. and Hicks, T. (1998). Assessing transfer probabilities in a Bayesian interpretation of forensic glass evidence. *Science and Justice*, 38(1), 15-21. doi:10.1016/S13550306(98)720684

Examples

```
par <- c(80, 50, 0.5, 0.05, 0.8, 0.9, 0.1, 0.2, 0.4, 0.6, 0.05, 0.1,
         0.5, 0.7, 1, 1, 0.5)
tfrp(par, ymax = 10, nsim = 100, seed = 1)$prob
```

---

tipplt *Tippett plot*

Description

For each value on the axis, the share of same-source and of different-source likelihood ratios above it. A good system keeps the same-source curve high to the right of 0 and the different-source curve low; where each crosses 0 shows the rate of misleading evidence. `proof()` records no theorem for this plot.

Usage

```
tipplt(same, diff, ...)
```

Arguments

`same`	Likelihood ratios from same-source comparisons: positive numbers.
`diff`	Likelihood ratios from different-source comparisons: positive numbers.
`...`	Passed to `graphics::plot()`.

Value

Invisibly, a list of two data frames, `same` and `diff`, each with `log10lr` and `share` (the share of ratios at or above it).

Examples

```
tipplt(same = c(50, 200, 8, 0.5), diff = c(0.01, 0.2, 1.5, 0.001))
```

tlvc *Two-level model for multivariate trace evidence*

Description

`tlvc()` estimates the background of the two-level normal model of Aitken and Lucy (2004) from a reference collection, `x_ij = theta_i + e_ij`, items `theta_i ~ N(mu, C)` and measurements within an item `e_ij ~ N(0, U)`: the overall mean, the pooled within-item covariance and the ANOVA between-item covariance (unbiased for any pattern of replication). `tlvlr()` gives the likelihood ratio of same source against different sources for two items (a control and a recovered fragment, say) from their mean vectors, computed in logarithms from Cholesky factors so that atypical or distant items give a finite value.

Usage

```
tlvc(x, item)

tlvlr(xa, xb, na, nb, bg)
```

Arguments

`x`	The reference measurements: a numeric matrix, one row per observation and one column per variable.
`item`	The item of each row: labels, every item at least once, some item more than once.
`xa`, `xb`	The mean vectors of the two items compared: one number per variable.
`na`, `nb`	Their numbers of replicates: positive.
`bg`	The background, as `tlvc()` returns it.

Value

`tlvc()`: a list of `mu` (the mean), `u` and `c` (the within- and between-item covariance matrices; `c` need not be positive definite, which `tlvlr()` then refuses). `tlvlr()`: a named vector, `ln` and `log10`, the natural and base-10 logarithms of the ratio.

References

Aitken, C. G. G. and Lucy, D. (2004). Evaluation of trace evidence in the form of multivariate data. *Journal of the Royal Statistical Society, Series C*, 53(1), 109-122. doi:10.1046/j.0035-9254.2003.05271.x

Examples

```
set.seed(1)
th <- matrix(rnorm(20, 10, 2), 10, 2)
x <- th[rep(1:10, each = 3), ] + matrix(rnorm(60, 0, 0.2), 30, 2)
bg <- tlvc(x, rep(1:10, each = 3))
```

```
tlvlr(th[1, ], th[1, ] + 0.05, 3, 3, bg)
tlvlr(th[1, ], th[2, ], 3, 3, bg)
```

---

tmpaud *Disparity metrics over time and place*

Description

The disparate impact ratio, the parity gap, the Gini coefficient of the groups' shares flagged and the bias amplification score of `fairdf()`, for every city and period, and how far each city's ratio moves over the periods (after Barman and Barman 2026).

Usage

```
tmpaud(pred, grp = NULL, period = NULL, city = NULL, base = NULL)
```

Arguments

`pred`	The tool's decisions: 1 (or `TRUE`) for a case it flags, 0 (or `FALSE`) otherwise, one per case. A `clpre` list from `clprep()` is taken too: its elements fill the arguments by name.
`grp`	Group labels: numbers, text or a factor, one per case, no missing values. Groups are ordered as the factor's levels, or sorted.
`period`	The period of each case: labels (text, numbers or a factor), sorted as `fair()` sorts groups; ISO dates as text sort in time order.
`city`	`NULL` (one place, labelled `"all"`), or the city of each case: labels.
`base`	The base group: `NULL` for the group with the largest share flagged (the first in order on ties), its label, or its position from 1.

Details

A cell (one city in one period) is audited when it holds at least two groups and the base group; otherwise it is reported and skipped. The base group is the same in every cell: by default the group with the largest share flagged over all the data. For each city, the means of the four values over its audited cells (finite values only), the smallest and largest ratio, their range, and how many periods have a ratio above

1. A ratio that moves a lot from one period to the next says that one audit at one time describes that time only; a spread between cities says the same tool can treat groups differently in different places. Neither says why. For intervals on one cell, call `fairdf()` on it.

Value

A list: `cells`, a data frame with one row per city and period that occurs: `city`, `period`, `n`, `ok` (`TRUE` when audited), `di`, `dp`, `gini` and `bas` (`NaN` when skipped); `cities`, one row per city: `city`, `ncell` (cells audited), `di`, `dp`, `gini`, `bas` (their means), `dimin`, `dimax`, `dirng` and `ngt1`; `sum`, `rng` (the largest range) and `spread` (largest minus smallest mean ratio of the cities; `NaN` with fewer than two); `base`.

Examples

```
set.seed(2)
n <- 600
g <- sample(c("a", "b"), n, TRUE)
per <- sample(c("2024-1", "2024-2", "2024-3"), n, TRUE)
cty <- sample(c("north", "south"), n, TRUE)
p <- rbinom(n, 1, ifelse(g == "a", 0.4, ifelse(cty == "north", 0.3, 0.2)))
tmpaud(p, g, per, cty)$cities
```

---

vodtst *Veil-of-darkness test*

Description

Compares each group's share of stops made in daylight, when a driver's race can be seen, with its share in darkness, when it cannot. The ratio of the two odds against a reference group is the factor by which visibility changes that group's chance of being stopped; 1 means no change.

Usage

```
vodtst(day, night = NULL, ref = 1, shift = NULL)
```

Arguments

day	Stops by group in daylight: non-negative numbers, as a numeric vector, a 1-d table, a list of single numbers or a one-row data frame. With night left out, one table of groups by stops: a matrix, 2-d table or data frame whose two columns are named day and night, in either order (columns are never taken by position).
night	Stops by group in darkness: positive numbers, in the forms of day; named inputs are matched to day by name, and an error lists any group named in one and not the other. Unnamed inputs are matched by position: only names protect against a mismatch.
ref	The reference group, by name (matched exactly, case included; a factor by its label) or by position in day as given.
shift	NULL, or one positive number per group: daylight exposure over darkness exposure, in the forms of day, matched by name.

Details

The ratio identifies the visibility effect only when the mix of drivers on the road is the same in daylight and darkness. shift states how much each group's exposure differs between the two, and adj is the ratio with that difference taken out. See proof("vodtst").

Value

A data frame with one row per group, in the order of day: group, ref (TRUE for the reference group), shday, shnight, ratio, sign (the sign of shday - shnight) and, when shift is given, adj.

References

Grogger, J. and Ridgeway, G. (2006). Testing for racial profiling in traffic stops from behind a veil of darkness. *Journal of the American Statistical Association*, 101(475), 878-887. doi:10.1198/016214506000000168

Examples

```
vodtst(day = c(white = 600, black = 400),
       night = c(white = 700, black = 300))
# one table of groups by day and night
stops <- data.frame(day = c(600, 400), night = c(700, 300),
                    row.names = c("white", "black"))
vodtst(stops)
```

`webpt` *Centre of minimum distance*

Description

The point `c` that minimises the weighted sum of distances `sum(w * |p - c|)` to the points: the Weber point or spatial median, which, unlike the median centre, does not depend on the axes. It is found by Weiszfeld's iteration with the modification of Vardi and Zhang (2000), which keeps every step defined and the sum non-increasing when an iterate lands on a data point. The optimality gap certifies the answer: it is 0 exactly at a minimiser. The minimiser is unique unless all the points lie on one line with the weight split evenly.

Usage

```
webpt(
  x,
  y = NULL,
  w = NULL,
  start = NULL,
  tol = 1e-10,
  maxit = 1000,
  lonlat = FALSE
)
```

Arguments

`x, y`	The points: `x` and `y` numeric vectors of equal length; or `x` a matrix or data frame with columns named `x` and `y` (or `X` and `Y`, as `sf::st_coordinates()` names them, or `lon` and `lat` with `lonlat = TRUE`) and `y` `NULL`; or `x` an sf or sfc object of POINT geometries, whose coordinate reference system must agree with `lonlat`. An unnamed two-column matrix is read as `x` then `y`; other columns are never taken by position. Vectors given separately (`x`, `y`, and `w` or `t` where a function takes them) are matched by name when every one is named and no

	name repeats, as elsewhere in the package; otherwise by position. No missing or infinite values.
w	NULL for equal weights, or one non-negative weight per point (counts at a location, say), with a positive sum.
start	NULL to start at the mean centre, or c(x, y) (in degrees when lonlat = TRUE).
tol	Stop when a step is at most tol times the root mean squared distance to the mean centre: one positive number.
maxit	The largest number of steps: one whole number from 0.
lonlat	FALSE (the default) when x and y are planar coordinates in one length unit; TRUE when they are longitude and latitude in degrees, which are then projected to metres by eqproj() about their own centre, and centres are returned in degrees. It is never guessed.

Value

A named numeric vector: x, y (the centre, in degrees when lonlat = TRUE), obj (the weighted sum of distances), gap (the optimality gap, 0 at a minimiser), iter (steps taken) and code: 2 at an exact minimiser, 1 when the step fell below tol, 0 when maxit was reached.

References

Weiszfeld, E. and Plastria, F. (2009). On the point for which the sum of the distances to n given points is minimum. *Annals of Operations Research*, 167, 7-41 (translation of the 1937 paper). doi:10.1007/s104790080352z

Vardi, Y. and Zhang, C.-H. (2000). The multivariate L1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97(4), 1423-1426. doi:10.1073/pnas.97.4.1423

Examples

```
webpt(c(0, 1, 0, 10), c(0, 0, 1, 10))
```

---

zgll *Group-based trajectory model for counts*

Description

The finite mixture of Nagin (1999) for counts over time, zero-inflated Poisson within groups (Lambert 1992; Nielsen et al. 2014): unit i belongs to group k with probability prob[k]; given k, its counts are independent Poisson with log mean off + x_t' beta_k and, when z is given, zero with extra probability plogis(z_t' gamma_k). zgll() evaluates the log-likelihood and the posterior memberships; zgem() fits by a generalised EM from the values given (the log-likelihood never decreases). One start per call: the model has local maxima, so several starts are the caller's loop, and the number of groups is the caller's choice (compare the log-likelihoods with a penalty such as BIC).

Usage

```
zgll(y, x, z = NULL, par, prob, off = NULL)

zgem(y, x, z = NULL, par, prob, off = NULL, maxit = 500, tol = 1e-08)
```

Arguments

`y`	Counts: a matrix with one row per unit and one column per period; `NA` or a negative value is a missing period.
`x`	The design of the Poisson part: a matrix with one row per period (columns 1, t, t^2, say).
`z`	`NULL` for a Poisson model, or the design of the zero-inflation part, one row per period.
`par`	The coefficients: a matrix with one row per column of `x` (then of `z`) and one column per group.
`prob`	The group shares: positive numbers summing to 1.
`off`	`NULL`, or log exposures with the dimensions of `y`.
`maxit`	The most iterations: a whole number from 1.
`tol`	Stop when an iteration gains at most `tol (1 + |ll|)`.

Value

`zgll()`: a list of `ll` and `post` (a matrix of posterior memberships, one row per unit). `zgem()`: a list of the fitted `par`, `prob`, `post`, `ll` and `iter` (iterations made).

References

Nagin, D. S. (1999). Analyzing developmental trajectories: a semiparametric, group-based approach. *Psychological Methods*, 4(2), 139-157. doi:10.1037/1082989X.4.2.139

Lambert, D. (1992). Zero-inflated Poisson regression, with an application to defects in manufacturing. *Technometrics*, 34(1), 1-14. doi:10.2307/1269547

Examples

```
set.seed(1)
x <- cbind(1, (1:6 - 3.5) / 2.5)
b <- cbind(c(0.5, -1), c(1.5, 0.5))
y <- t(sapply(rep(1:2, each = 50),
              function(k) rpois(6, exp(x %*% b[, k]))))
f <- zgem(y, x, par = b + 0.2, prob = c(0.5, 0.5))
f$par
```

Index

aorwt, 4

bdig, 6
bjtc, 7
bnft *(*bdig*)*, 6

calaud, 8
calaud(), *41*, *104*
catchk, 9
cgmedn, 11
cgstat, 12
cgstat(), *114*
chmb, 13
cllr, 14
cllrmin, 15
cllrmin(), *14*, *33*
clprep, 15
clprep(), *8*, *18*, *22*, *25*, *39*, *45*, *104*, *107*, *118*
clrsmp, 17
cntreg, 18
coofg, 20
crwalk, 21
ctabx, 22
cvhull, 24

dastd, 25
dbsrch, 26
dirmn, 27
dmlx, 28
dspbm, 31
dyadrt, 32

eceplt, 33
enpath, 34
enpath(), *29*
eqinv *(*eqproj*)*, 36
eqproj, 36
eqproj(), *11*, *12*, *21*, *24*, *27*, *43*, *49*, *50*, *58*, *61*, *69*, *74*, *75*, *96*, *97*, *100*, *105*, *112–114*, *121*

expbnd, 37

fair, 38
fair(), *104*, *118*
fairdf, 39
fairdf(), *41*, *118*
fairsim, 41
fairsim(), *9*
fzmode, 42

gearc, 43
gearc(), *111*
gorg, 44
gorg(), *111*
graphics::plot(), *33*, *116*
grprr, 45
gstr, 46
gtmp, 47

hdbang, 48
hvdist, 50
hzdsel, 51

impfpr, 52
impfpr(), *40*
incap, 53
incapr, 54
incapr(), *53*

jtacc, 55
jtacc(), *59*
jtcal, 56
jtcal(), *57*
jtdecay, 57
jtdecay(), *56*
jtsurf, 58
jtsurf(), *42*, *55*, *56*, *75*, *97*

kinidx, 59
kinidx(), *91*
kmpart, 60

lgtreg, 62
lipaud, 63
lmorn, 64
lmorn(), *111*
locgi, 65
locgi(), *111*
losbnd, 66
lrcal, 66
lrpost, 67
lwss, 68

mantst, 68
misrd, 70
mixlr, 71
mvtem *(*mvtll*)*, 72
mvtll, 72

nnhclu, 73
nnhclu(), *99, 100*
nnidx, 75
nordet, 76
nsbnd, 77
ntbrk, 78
ntbtw, 79
ntchg, 80
ntcls, 81
ntcmp, 82
ntcore, 83
ntdeg, 84
ntdeg(), *78, 79*, *81–83*, *86, 87*
ntevc, 85
ntlvn, 86
ntmod, 87
ntmod(), *79*, *86*

oaxbd, 88
orrisk, 90
outtst, 90

patidx, 91
proof, 92

relchk, 92
rfor, 94
rfor(), *29*
ripenv, 96
ripk, 97
ripk(), *96*
rnkres, 98
rnnhcl, 99
rptc, 100
rshl *(*rswp*)*, 101
rswp, 101
rtmhs, 102

scrdsp, 103
scrdsp(), *41*
sdell, 104
sdell(), *74*, *100*, *113*
sellab, 106
set.seed(), *6*, *8*, *21*, *39*, *43, 44*, *47*, *61*, *64*, *69*, *75*, *96*, *108*, *110*, *115*
sf::st_coordinates(), *11, 12*, *21*, *24*, *27*, *42*, *49*, *61*, *69*, *73*, *99*, *105*, *111*, *114*, *120*
sirex, 107
skp, 108
sntbnd, 109
sptst, 110
spwt, 111
spwt(), *42–44*, *64, 65*, *75*, *97*
stacel, 112
stats::anova(), *104*
stats::binom.test(), *45*
stats::chisq.test(), *23*
stats::cor.test(), *9*
stats::dist(), *63*
stats::glm(), *19*
stats::lm(), *104*
stats::lowess(), *68*
stats::p.adjust(), *64*
stats::poisson.test(), *45*
stmavg, 114

tfrp, 115
tipplt, 116
tlvc, 117
tlvlr *(*tlvc*)*, 117
tmpaud, 118

vodtst, 119

webpt, 120
webpt(), *11*

zgem *(*zgll*)*, 121
zgll, 121