Package ‘vismeteor’

September 12, 2026

Type Package

Title Analysis of Visual Meteor Data

Version 3.1.0

Author Janko Richter [aut, cre]

Maintainer Janko Richter <janko@richtej.de>

Description Provides a suite of analytical functionalities to process and analyze visual meteor observations from the Visual Meteor Database of the International Meteor Organization <https://www.imo.net/>.

License MIT + file LICENSE

URL https://github.com/jankorichter/vismeteor

BugReports https://github.com/jankorichter/vismeteor/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.1.0)

Imports methods, stats, httr2 (>= 1.0.0)

Suggests testthat (>= 3.2.0), httptest2, knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-09-12 17:20:02 UTC

Contents

`vismeteor-package` *vismeteor: Analysis of Visual Meteor Data*

Description

Provides a suite of analytical functionalities to process and analyze visual meteor observations from the Visual Meteor Database of the International Meteor Organization `https://www.imo.net/`.

Details

The data used in this package can created and provided by imo-vmdb.

Author(s)

Maintainer: Janko Richter `<janko@richtej.de>`

Authors:

- Janko Richter `<janko@richtej.de>`

See Also

Useful links:

- `https://github.com/jankorichter/vismeteor`
- Report bugs at `https://github.com/jankorichter/vismeteor/issues`

`freq_quantile` *Quantiles with a minimum frequency*

Description

This function generates quantiles with a minimum frequency. These quantiles are formed from a vector `freq` of frequencies. Each quantile then has the minimum total frequency `min`.

Usage

```
freq_quantile(freq, min)
```

Arguments

`freq`	integer; A vector of frequencies.
`min`	integer; Minimum total frequency per quantile.

Details

The frequencies `freq` are grouped in the order in which they are passed as a vector. The minimum `min` must be greater than `0`.

Value

A factor of indices is returned. The index references the corresponding passed frequency `freq`.

Examples

```
freq <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
cumsum(freq)
(f <- freq_quantile(freq, 10))
tapply(freq, f, sum)
```

`load_vmdb` *Loading visual meteor observations via the imo-vmdb REST API*

Description

Loads visual meteor observations from an imo-vmdb web server via its REST API.

Usage

```
load_vmdb_rates(
  base_url,
  shower = NULL,
  period = NULL,
  sl = NULL,
  lim_magn = NULL,
  sun_alt_max = NULL,
  moon_alt_max = NULL,
  session_id = NULL,
  rate_id = NULL,
  with_sessions = FALSE,
  with_magnitudes = FALSE
)

load_vmdb_magnitudes(
  base_url,
  shower = NULL,
  period = NULL,
  sl = NULL,
  lim_magn = NULL,
  session_id = NULL,
  magn_id = NULL,
  with_sessions = FALSE,
  with_magnitudes = TRUE
)
```

Arguments

`base_url`	character; base URL of the imo-vmdb API, e.g. `"http://localhost:8000/api/v1"`.
`shower`	character; selects by meteor shower codes. `NA` loads sporadic meteors.
`period`	selects an observation time range by minimum/maximum. Accepts `POSIXct`, `Date`, or character. Character elements may be `"YYYY-MM-DDTHH:MM:SS"`, `"YYYY-MM-DD HH:MM:SS"`, or date-only `"YYYY-MM-DD"`. Date-only inputs are expanded to midnight (lower bound) and 23:59:59 (upper bound) of the given day, in UTC. IMO data is UTC by convention; the timezone marker is omitted on the wire.
`sl`	numeric; selects a range of solar longitudes by minimum/maximum.
`lim_magn`	numeric; selects a range of limiting magnitudes by minimum/maximum.
`sun_alt_max`	numeric; selects the maximum altitude of the sun (rates only).
`moon_alt_max`	numeric; selects the maximum altitude of the moon (rates only).
`session_id`	integer; selects by session ids.
`rate_id`	integer; selects rate observations by ids.
`with_sessions`	logical; if `TRUE`, also load the corresponding session data.
`with_magnitudes`	logical; if `TRUE`, also load the corresponding magnitude observations.
`magn_id`	integer; selects magnitude observations by ids.

Details

sl, period and lim_magn expect a vector with successive minimum and maximum values. sun_alt_max and moon_alt_max are expected to be scalar values.

Note: Unlike the previous DBI-based version, only a single range per filter parameter is supported. If you previously passed a matrix with multiple rows to period, sl, or lim_magn, flatten them to a single min/max pair or issue multiple calls and combine with rbind().

Value

Both functions return a list, with

observations	data frame, rate or magnitude observations,
sessions	data frame; session data of observations,
magnitudes	table; contingency table of meteor magnitude frequencies.

observations depends on the function call. load_vmdb_rates returns a data frame with columns:

rate_id	unique identifier of the rate observation,
shower	IAU code of the shower. NA for sporadic.
period_start	POSIXct (UTC); start of observation,
period_end	POSIXct (UTC); end of observation,
sl_start	solar longitude at start,
sl_end	solar longitude at end,
session_id	reference to the session,
freq	count of observed meteors,
lim_magn	limiting magnitude,
t_eff	net observed time in hours,
f	correction factor of cloud cover,
sidereal_time	sidereal time,
sun_alt	altitude of the sun,
sun_az	azimuth of the sun,
moon_alt	altitude of the moon,
moon_az	azimuth of the moon,
moon_illum	illumination of the moon (0.0 .. 1.0),
field_alt	altitude of the field of view (optional),
field_az	azimuth of the field of view (optional),
rad_alt	altitude of the radiant (optional),
rad_az	azimuth of the radiant (optional),
magn_id	reference to the magnitude observations (optional),
magn_solo	TRUE if this rate is the sole contributor to its linked magnitude observation; FALSE if the magnitude aggrega

load_vmdb_magnitudes returns an observations data frame with:

magn_id	unique identifier of the magnitude observation,
shower	IAU code of the shower. NA for sporadic.
period_start	POSIXct (UTC); start of observation,
period_end	POSIXct (UTC); end of observation,

sl_start	solar longitude at start,
sl_end	solar longitude at end,
session_id	reference to the session,
freq	count of observed meteors,
magn_mean	mean magnitude,
lim_magn	limiting magnitude (optional).

The sessions data frame contains

session_id	unique identifier of the session,
longitude	location's longitude,
latitude	location's latitude,
elevation	height above mean sea level in km,
country	country name,
location_name	location name,
observer_id	observer id (optional),
observer_name	observer name (optional).

magnitudes is a contingency table of meteor magnitude frequencies. Row names are magnitude observation IDs; column names are magnitude classes.

Note

Angle values are expected and returned in degrees.

References

https://pypi.org/project/imo-vmdb/

Examples

```
## Not run:
# Load rate observations including session and magnitude data
data <- load_vmdb_rates(
    base_url = "http://localhost:8000/api/v1",
    shower = "PER",
    sl = c(135.5, 145.5),
    period = c("2015-08-01", "2015-08-31"),
    lim_magn = c(5.3, 6.7),
    with_magnitudes = TRUE,
    with_sessions = TRUE
)

# Load magnitude observations
data <- load_vmdb_magnitudes(
    base_url = "http://localhost:8000/api/v1",
    shower = "PER",
    sl = c(135.5, 145.5),
    period = c("2015-08-01", "2015-08-31"),
    lim_magn = c(5.3, 6.7),
```

```
    with_sessions = TRUE
)

# Period can also be given as POSIXct or as full ISO 8601 datetime
# strings - useful for narrowing to a specific time window within a day.
data <- load_vmdb_rates(
    base_url = "http://localhost:8000/api/v1",
    shower   = "PER",
    period   = c("2015-08-12T20:00:00", "2015-08-13T04:00:00")
)

## End(Not run)
```

`mideal` *Ideal Distribution of Meteor Magnitudes*

Description

Density, distribution function, quantile function and random generation for the ideal distribution of meteor magnitudes.

Usage

```
dmideal(m, psi = 0, log = FALSE)

pmideal(m, psi = 0, lower.tail = TRUE, log = FALSE)

qmideal(p, psi = 0, lower.tail = TRUE)

rmideal(n, psi = 0)
```

Arguments

`m`	numeric; meteor magnitude.
`psi`	numeric; the location parameter of a probability distribution. It is the only parameter of the distribution.
`log`	logical; if `TRUE`, probabilities p are given as `log(p)`.
`lower.tail`	logical; if `TRUE` (default) probabilities are $P[M \le m]$, otherwise, $P[M > m]$.
`p`	numeric; probability.
`n`	numeric; count of meteor magnitudes.

Details

The density of the ideal distribution of meteor magnitudes is

$$\frac{\mathrm{d}p}{\mathrm{d}m} = \frac{3}{2}\,\log(r)\sqrt{\frac{r^{3\,\psi+2\,m}}{(r^{\psi}+r^{m})^{5}}}$$

where m is the continuous (real-valued) meteor magnitude, $r = 10^{0.4} \approx 2.51189\dots$ is a constant and ψ is the only parameter of this magnitude distribution.

Value

`dmideal` gives the density, `pmideal` gives the distribution function, `qmideal` gives the quantile function and `rmideal` generates random deviates.

The length of the result is determined by `n` for `rmideal`, and is the maximum of the lengths of the numerical vector arguments for the other functions.

`qmideal` returns `NA` with a warning for probabilities outside $[0, 1]$.

References

Richter, J. (2018) *About the mass and magnitude distributions of meteor showers*. WGN, Journal of the International Meteor Organization, vol. 46, no. 1, p. 34-38

Examples

```
old_par <- par(mfrow = c(2, 2))
psi <- 5.0
plot(
    \(m) dmideal(m, psi, log = FALSE),
    -5, 10,
    main = paste0("Density of the Ideal Meteor Magnitude\nDistribution (psi = ", psi, ")"),
    col = "blue",
    xlab = "m",
    ylab = "dp/dm"
)
abline(v = psi, col = "red")

plot(
    \(m) dmideal(m, psi, log = TRUE),
    -5, 10,
    main = paste0("Density of the Ideal Meteor Magnitude\nDistribution (psi = ", psi, ")"),
    col = "blue",
    xlab = "m",
    ylab = "log( dp/dm )"
)
abline(v = psi, col = "red")

plot(
    \(m) pmideal(m, psi),
    -5, 10,
    main = paste0("Probability of the Ideal Meteor Magnitude\nDistribution (psi = ", psi, ")"),
    col = "blue",
    xlab = "m",
    ylab = "p"
)
abline(v = psi, col = "red")

plot(
```

```
    \(p) qmideal(p, psi),
    0.01, 0.99,
    main = paste("Quantile of the Ideal Meteor Magnitude\nDistribution (psi = ", psi, ")"),
    col = "blue",
    xlab = "p",
    ylab = "m"
)
abline(h = psi, col = "red")

# generate random meteor magnitudes
m <- rmideal(1000, psi)

# log likelihood function
llr <- function(psi) {
    -sum(dmideal(m, psi, log = TRUE))
}

# maximum likelihood estimation (MLE) of psi
est <- optim(2, llr, method = "Brent", lower = 0, upper = 8, hessian = TRUE)

# estimations
est$par # mean of psi
sqrt(1 / est$hessian[1][1]) # standard deviation of psi

par(old_par)
```

`PER_2015_magn` *Visual magnitude observations of Perseids from 2015*

Description

Visual magnitude observations of the Perseid shower from 2015.

Format

A list with the same structure as returned by load_vmdb_magnitudes.

Details

`PER_2015_magn` are magnitude observations loaded with load_vmdb_magnitudes.

See Also

load_vmdb

PER_2015_rates — *Visual rate observations of Perseids from 2015*

Description

Visual rate observations of the Perseid shower from 2015.

Format

A list with the same structure as returned by load_vmdb_rates.

Details

`PER_2015_rates` are rate observations loaded with load_vmdb_rates.

See Also

load_vmdb

select_knots — *Greedy stepwise knot selection for a regression spline*

Description

Selects a parsimonious subset of `knot_candidates` as interior knots for a regression spline by greedy forward addition or backward elimination, scored by a user-supplied criterion (BIC, AIC, or any cross-validation function). The function is model-agnostic: it only chooses which knot positions to include and passes that vector to your `score_fun`; your `score_fun` decides which basis (`splines::ns`, `splines::bs` of any degree, `...`) and which model family to fit. Typical use: fitting a smooth, parsimonious trend through noisy data — for example an activity profile of a meteor shower across solar longitude — where the number and position of knots should remain small and inspectable.

Usage

```
select_knots(
  data,
  knot_candidates,
  score_fun,
  backward = FALSE,
  n_steps = NULL,
  bulk_gap = 4L,
  fixed_knots = numeric(0),
  verbose = FALSE,
  n_cores = 1L
)
```

Arguments

data
: Data passed unchanged to score_fun; typically a data.frame. The function itself does not inspect or mutate it.

knot_candidates
: Numeric vector of candidate interior-knot positions. Duplicates and unsorted input are tolerated (sorted/uniq'd internally).

score_fun
: \(data, knots) -> numeric scalar. Lower is better. The caller is responsible for fitting whatever model they want and returning a single criterion value (BIC, AIC, cross-validation error, ...).

backward
: Logical. FALSE (default) = forward selection, TRUE = backward elimination.

n_steps
: NULL (default) or a positive integer. NULL runs the greedy search until the next move would not improve the score and stops there — so the end state of the search IS the score optimum (under the current direction and constraints). A positive integer N runs exactly N iterations regardless of whether each one improves the score; this is a deliberate exploration mode, intended for inspecting the score landscape just past a previously-found optimum. The canonical recipe is to first call with n_steps = NULL to find the optimum, then re-call with fixed_knots = prev$knots and n_steps = 1L (or 2L, 3L) to take one or more controlled steps onward and see how rapidly the score deteriorates. The value 0L is not allowed (use NULL for "no steps past the optimum"); non-integer or negative values error out.

bulk_gap
: Integer (>= 0). Minimum index gap between knots removed in the same round of backward elimination. Ignored when backward = FALSE. 0L disables bulk removal. The default 4L matches the support of a cubic spline basis (degree + 1 for cubic B-splines; the same value also fits splines::ns, which is cubic by construction); for higher-degree B-splines pick degree + 1 accordingly. Note that bulk_gap = 1L imposes no real separation constraint and therefore removes *every* improving knot in one round – a very aggressive setting that defeats the per-round verify-fit's role as a safety net.

fixed_knots
: Numeric vector of knot positions that must be present in every fitted model during the search. In forward mode they are set from the start, and further knots may be added on top; in backward mode they are never proposed for removal, neither singly nor in bulk. Need not be a subset of knot_candidates – included regardless. Duplicates and unsorted input are tolerated. Default numeric(0) (no fixed knots).

verbose
: Logical. If TRUE, prints per-round progress (cat() to stdout). Default FALSE.

n_cores
: Integer (>= 1). Performance-only knob: the per-round candidate scoring runs in parallel across n_cores workers. Default 1L (serial, no extra dependency). When n_cores > 1L the base-R package **parallel** is loaded and parallel::mclapply() is used (fork-based on macOS/Linux; falls back to serial on Windows). Quick recipes:
  - n_cores = 1L – safe default, no extra package loaded.
  - n_cores = max(1L, parallel::detectCores() - 1L) – use all cores except one; good default for interactive multi-tasking.
  - n_cores = parallel::detectCores() – use every core; fastest but resource-hungry (the whole machine is busy).

The function errors out with a clear message if `n_cores > 1L` but **parallel** is not installed.

Reproducibility deserves a note when `n_cores > 1L`: `mclapply()` is fork-based and inherits the parent RNG state. If your `score_fun` uses randomness (e.g. cross-validation splits), set `RNGkind("L'Ecuyer-CMRG")` and seed via `set.seed()` *before* calling `select_knots()` to get reproducible per-worker streams; otherwise results can differ run to run and across `n_cores`.

Details

The typical use case — meteor shower rates over the solar longitude or time, for example — is a smooth, slowly varying signal whose shape is not known in advance and whose curvature changes locally: too rigid for a low-order polynomial, too noisy for a histogram. A regression spline (typically cubic, e.g. `splines::ns` or `splines::bs(degree = 3)`) with a modest number of well-placed interior knots gives a smooth, locally flexible fit with interpretable degrees of freedom. Picking that number is a bias/variance trade-off: too many knots overfit (ringing, unstable derivatives, slow fits), too few introduce bias. `select_knots()` automates the trade-off by greedy forward addition or backward elimination, scored by a user-defined criterion; you control which knots are even allowed (`knot_candidates`) and what counts as "better" (`score_fun`).

Your `score_fun` must take `(data, knots)` and return a single `numeric` value; *lower is better*. Typically it fits a model with these interior knots (`length(knots) == 0L` means "no interior knots") and reports an information criterion (`stats::BIC`, `stats::AIC`) or a held-out score. `knots` arrives sorted, so the fit code does not need to sort it again. A divergent or failed fit should ideally return `Inf`; the function additionally wraps each call in `tryCatch()` and treats errors as `Inf` so the search continues robustly. See `Examples` for a runnable template.

Backward elimination can drop multiple knots per round in "bulk" mode. When `backward = TRUE` and `bulk_gap >= 1L`, each round removes several well-separated knots at once (minimum index gap `bulk_gap` in the current sorted knot list). For a B-spline basis of degree `d`, each basis function has support over `d + 1` consecutive knot intervals, so removing knots that are at least `d + 1` positions apart has nearly additive effect on the score — giving roughly a `bulk_gap`-fold speed-up at the cost of a small approximation (mitigated by a verify-fit after each bulk round). The default `4L` matches `d + 1` for cubic bases such as `splines::ns` or `splines::bs(degree = 3)`; for other bases pick `bulk_gap = degree + 1`, or set `bulk_gap = 0L` for strict one-knot-per-round behaviour.

The result splits the final knot set into two disjoint vectors: `knots` (positions the algorithm itself selected) and `fixed_knots` (the user-supplied anchors, echoed back, deduplicated and sorted). The full vector to fit a model on is `c(knots, fixed_knots)` (or its sorted form). With the default `n_steps = NULL` the loop stops as soon as no further move improves the score, so the end state *is* the score-optimum; with `n_steps > 0L` the loop runs that many iterations regardless of improvement, and the score-best point along the trajectory is recoverable from `history` via the row at `which.min(history$score)`.

If the starting state already is (locally) optimal — typical when `fixed_knots` alone overfits in forward mode, or when `knot_candidates` is so small that the full pool already minimises the score in backward mode — the `n_steps = NULL` run terminates immediately with `history` containing only the initial row and `score` equal to the starting score; an explicit `n_steps = N` run will instead take `N` worsening steps so the post-optimum landscape is still observable.

Knots sit next to extrema, not on them. `select_knots()` chooses knots for a *good fit*, not for detecting extrema of the response. Knots typically land next to peaks or troughs — where the curvature is highest — and not on them. Read local extrema off the *shape* of the fitted curve, not from `knots`. A knot supplied through `fixed_knots` is the exception: it sits wherever the user puts it, since it is a constraint imposed on the search rather than a finding produced by it.

Value

A list with elements (in this order):

- `backward` Logical — the direction used.
- `knots` Sorted numeric vector — the interior knots the algorithm itself selected, with `fixed_knots` *excluded*. Disjoint from `fixed_knots` by construction. The full knot vector to fit a model on is `c(knots, fixed_knots)` (or its sorted form). With the default `n_steps = NULL` this is the score-optimal selection; with `n_steps > 0L` it can be a state past the optimum.
- `fixed_knots` Sorted numeric vector — the user-supplied fixed knots echoed back (deduplicated and sorted). Empty vector if the caller did not supply any.
- `score` Numeric — the score at the end state, i.e. at `c(knots, fixed_knots)`.
- `n_steps` The `n_steps` value used (`NULL` or a positive integer).
- `history` `data.frame` of per-round records: `step`, `n_knots` (total interior knot count including `fixed_knots`), `changed_knot` (the knot added or removed in that round; `NA` for the initial state and for bulk-removal rounds), `score`, `extra` (boolean: `TRUE` when that step worsened the score). For `n_steps = NULL` runs no worsening step is taken, so `extra` is always `FALSE`.

When to use this

`select_knots()` is for situations where knot *positions* are meaningful and you want a small, inspectable set of interior knots scored under a criterion you choose. If you instead want a continuous roughness penalty over a dense knot grid, the **mgcv** package's penalised B-splines (`bs = "ps"`) or adaptive smoothers (`bs = "ad"`) are usually a better fit; for greedy stepwise selection on a hinge-function basis, see the **earth** package.

Like every greedy stepwise procedure, `select_knots()` performs a *local* search: in each round it commits to the locally best add/drop. It therefore returns a local optimum of the score, which is usually but not provably the global optimum — a knot *combination* that beats every individually-best move can be unreachable once an earlier, locally-attractive knot has been picked. The only way to guarantee globality would be exhaustive enumeration over all $2^{|knot\_candidates|}$ subsets, which is exponential and impractical for any realistic candidate pool. In practice the greedy optimum is close; repeating the search in the opposite direction (`backward = TRUE`) is a cheap robustness check.

See Also

`ns`, `bs`, `BIC`, `AIC`

Examples

```
## Not run:
# Greedy knot selection on a simple synthetic signal.
set.seed(1)
n <- 200
x <- seq(0, 10, length.out = n)
```

```
y <- stats::rpois(n, lambda = 50 + 30 * sin(x))
dat <- data.frame(x = x, y = y)

# score_fun: fit a Poisson GLM with a natural cubic spline, return BIC.
# Lower is better.
fit <- \(d, knots) {
    f <- if (length(knots) == 0L) {
        y ~ splines::ns(x)
    } else {
        y ~ splines::ns(x, knots = knots)
    }
    stats::glm(f, family = stats::poisson(), data = d)
}
score_bic <- \(d, knots) stats::BIC(fit(d, knots))

cand <- seq(1, 9, by = 0.5)
res <- select_knots(dat, cand, score_bic, verbose = TRUE)
# Full knot vector to fit the final model on:
final_knots <- sort(c(res$knots, res$fixed_knots))

## End(Not run)
```

---

vmgeom *Geometric Model of Visual Meteor Magnitudes*

Description

Density, distribution function, quantile function, and random generation for the geometric model of visual meteor magnitudes.

Usage

```
dvmgeom(m, lm, r, log = FALSE, perception_fun = vmperception)

pvmgeom(
  m,
  lm,
  r,
  lower.tail = TRUE,
  log = FALSE,
  perception_fun = vmperception
)

qvmgeom(p, lm, r, lower.tail = TRUE, perception_fun = vmperception)

rvmgeom(n, lm, r, perception_fun = vmperception)
```

Arguments

`m`	integer; the meteor magnitude.
`lm`	numeric; limiting magnitude.
`r`	numeric; the population index.
`log`	logical; if `TRUE`, probabilities p are given as `log(p)`.
`perception_fun`	function; perception probability function (optional). Default is vmperception.
`lower.tail`	logical; if `TRUE` (default) probabilities are $P[M < m]$, otherwise, $P[M \geq m]$.
`p`	numeric; probability.
`n`	numeric; count of meteor magnitudes.

Details

In visual meteor observations, magnitudes are estimated as integer values. Consequently, the distribution of observed magnitudes is discrete, and its probability mass function is given by

$$P[M = m] \sim \begin{cases} f(m_{\text{lim}} - m)\, r^m, & \text{if } m_{\text{lim}} - m > -0.5, \\ 0 & \text{otherwise,} \end{cases}$$

where m_{lim} denotes the limiting (non-integer) magnitude of the observation, and m the integer meteor magnitude. The function $f(\cdot)$ denotes the perception probability function.

Thus, the distribution is the product of the perception probabilities and the underlying geometric distribution of meteor magnitudes. Therefore, the parameter `p` of the geometric distribution is given by `p = 1 - 1/r`.

The parameter `lm` specifies the limiting magnitude for the meteor magnitude `m`. `m` must be an integer meteor magnitude. The length of the vector `lm` must either equal the length of the vector `m`, or `lm` must be a scalar value. In the case of `rvmgeom`, the length of the vector `lm` must equal `n`, or `lm` must be a scalar value.

If a different perception probability function `perception_fun` is provided, it must have the signature `function(x)` and return the perception probability of the difference `x` between the limiting magnitude and the meteor magnitude. If `x >= 15.0`, the function `perception_fun` should return a perception probability of `1.0`. The argument `perception_fun` is resolved using match.fun.

Value

- `dvmgeom`: density
- `pvmgeom`: distribution function
- `qvmgeom`: quantile function
- `rvmgeom`: random generation

The length of the result is determined by `n` for `rvmgeom`, and by the maximum of the lengths of the numeric vector arguments for the other functions. All arguments are vectorized; standard R recycling rules apply.

Since the distribution is discrete, `qvmgeom` and `rvmgeom` always return integer values. `qvmgeom` returns `NA` with a warning for probabilities outside $[0, 1]$.

See Also

vmperception stats::Geometric

Examples

```
N <- 100
r <- 2.0
limmag <- 6.5
(m <- seq(6, -7))

# discrete density of `N` meteor magnitudes
(freq <- round(N * dvmgeom(m, limmag, r)))

# log likelihood function
lld <- function(r) {
    -sum(freq * dvmgeom(m, limmag, r, log = TRUE))
}

# maximum likelihood estimation (MLE) of r
est <- optim(2, lld, method = "Brent", lower = 1.1, upper = 4)

# estimations
est$par # mean of r

# generate random meteor magnitudes
m <- rvmgeom(N, r, lm = limmag)

# log likelihood function
llr <- function(r) {
    -sum(dvmgeom(m, limmag, r, log = TRUE))
}

# maximum likelihood estimation (MLE) of r
est <- optim(2, llr, method = "Brent", lower = 1.1, upper = 4, hessian = TRUE)

# estimations
est$par # mean of r
sqrt(1 / est$hessian[1][1]) # standard deviation of r

m <- seq(6, -4, -1)
p <- vismeteor::dvmgeom(m, limmag, r)
barplot(
    p,
    names.arg = m,
    main = paste0("Density (r = ", r, ", limmag = ", limmag, ")"),
    col = "blue",
    xlab = "m",
    ylab = "p",
    border = "blue",
    space = 0.5
)
axis(side = 2, at = pretty(p))
```

`vmgeom_glm` *Generalized Linear Model for the Geometric Model of Visual Meteor Magnitudes*

Description

Fits the geometric model of visual meteor magnitudes as a generalized linear model, so that the population index `r` can be estimated as a function of covariates, and returns those estimates on the scale of `r`.

Usage

```
vmgeom_glm(
  formula,
  data,
  ...,
  perception_fun = vmperception,
  r_range = c(1.01, 20)
)

## S3 method for class 'vmgeom_glm'
predict(
  object,
  newdata,
  type = c("r", "inv_r", "log_r", "link"),
  se.fit = FALSE,
  level = 0.95,
  ...
)
```

Arguments

`formula`	an object of class stats::formula; the response must be a two-column matrix `cbind(magn, lim_magn)`.
`data`	a data.frame containing the variables of the model.
`...`	further arguments passed to stats::glm respectively stats::predict.glm.
`perception_fun`	function; perception probability function (optional). Default is vmperception.
`r_range`	numeric; the range of population indices considered when inverting the mean. Used to bracket the numerical inversion and to keep the moments of the distribution finite.
`object`	an object returned by `vmgeom_glm`.
`newdata`	optional data.frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`type`	character; the scale on which to return predictions. One of `"r"`, `"inv_r"`, `"log_r"` or `"link"`.

`se.fit`	logical; if `TRUE`, standard errors and confidence limits are returned in addition to the predictions.
`level`	numeric; the confidence level of the returned interval.

Details

In the geometric model of visual meteor magnitudes, the population index `r` is usually estimated as a single global value. `vmgeom_glm` allows `r` to depend on covariates instead, such as the solar longitude or limiting magnitude.

The response must be given as a two-column matrix, where the first column contains the integer meteor magnitudes and the second column the corresponding limiting magnitudes:

```
vmgeom_glm(cbind(magn, lim_magn) ~ x, data = magnitudes)
```

The limiting magnitude is a fixed property of each observation and is never estimated. Passing it as part of the response ensures that it is subsetted together with the remaining data whenever stats::glm drops rows, for example through `subset` or `na.action`.

The link function is

$$\eta = \mathrm{logit}(1/r) = -\log(r-1),$$

with the inverse $r = 1 + \exp(-\eta)$. The quantity $1/r$ is the factor by which the observable rate drops when the limiting magnitude is lowered by one, and thus lies between `0` and `1`, which is what a logit link is meant for. Since $1 + \exp(-\eta) > 1$ holds for every finite η, this link also enforces the constraint $r > 1$ required by dvmgeom.

This is deliberately not the canonical link. As stated for dvmgeom, the distribution of the integer meteor magnitude m is

$$P[M = m] \sim f(m_{\mathrm{lim}} - m)\, r^m,$$

where the perception probability $f(\cdot)$ does not depend on `r`. Writing $r^m = \exp\{m \log(r)\}$ exhibits this as an exponential family in which $\log(r)$ is the natural parameter and the magnitude m the sufficient statistic. The canonical link is therefore $\eta = -\log(r)$, the sign following the convention that a larger η means a smaller `r`. It is what makes the mean magnitude a sufficient statistic for `r`, so that the fit is an exact maximum likelihood estimate. It is not used here because it leaves $r > 1$ unenforced: $\eta > 0$ yields $r < 1$, which dvmgeom does not admit. Over the population indices met in practice, roughly `1.5` to `3.0`, the two scales are near-perfectly correlated, so little is given up. `predict` returns $\log(r)$ through `type = "log_r"` for those who prefer that scale.

The fit is an ordinary stats::glm object, so `summary`, `anova`, `AIC`/`BIC` and select_knots apply unchanged. `predict` additionally returns the population index itself rather than the linear predictor.

Visual magnitude observations are usually reported as counts per magnitude class rather than as individual meteors. Such data can be passed in aggregated form by supplying the counts as `weights`:

```
vmgeom_glm(cbind(magn, lim_magn) ~ x, data = magnitudes, weights = Freq)
```

This is equivalent to repeating every row `Freq` times, and yields identical coefficients, standard errors, deviance and `AIC`. Fractional counts should be rounded with vmtable beforehand.

Internally the distribution is evaluated through $1/r$ rather than `r` itself, which keeps every quantity bounded. The population index is nevertheless bracketed by `r_range`, because the mean of the distribution diverges as $r \to 1$: a population index of 1 implies arbitrarily many bright meteors and therefore has no finite mean. The default lower bound of 1.01 lies far below any population index encountered in practice, which is typically between 1.5 and 3.0.

Predictions and their uncertainties are obtained on the scale of the linear predictor and transformed afterwards. The confidence limits are the transformed limits of the linear predictor. Since every transformation offered here is monotonic, these limits are exact rather than an approximation, they are asymmetric about the estimate, and they cannot fall below 1. The reported se.fit is a first-order delta-method value, $|g'(\eta)|\,\mathrm{se}(\eta)$, and is intended as a measure of scale; use the limits themselves for inference. A second-order correction is deliberately omitted, as it stays below one percent for $\mathrm{se}(\eta) \leq 0.1$, which covers the sample sizes occurring in practice.

Value

- vmgeom_glm: an object of class vmgeom_glm inheriting from stats::glm.
- predict: a numeric vector of predictions, or a list with the components fit, se.fit, lwr and upr if se.fit = TRUE.

See Also

vmgeom vmperception vmgeom_vst_lm stats::glm stats::predict.glm

Examples

```
# Simulate magnitudes whose population index depends on a covariate
set.seed(1)
lim_magn <- rep(c(5.8, 6.1, 6.4), each = 120)
x <- rep(seq(-1.5, 1.5, length.out = 12), each = 30)
r <- 1 + exp(-(0.2 + 0.4 * x))
magn <- mapply(\(lim_magn, r) rvmgeom(1L, lim_magn, r), lim_magn, r)
obs <- data.frame(x = x, lim_magn = lim_magn, magn = magn)

# Fit the model
fit <- vmgeom_glm(cbind(magn, lim_magn) ~ x, data = obs)
summary(fit)

# Population index at selected covariate values
newdata <- data.frame(x = c(-1, 0, 1))
predict(fit, newdata) # r
predict(fit, newdata, type = "inv_r") # 1/r
predict(fit, newdata, type = "log_r") # log(r)

# ... including confidence limits
predict(fit, newdata, se.fit = TRUE)
```

vmgeom_vst *Variance-Stabilizing Transformation for Geometric Visual Meteor Magnitudes*

Description

Applies a variance-stabilizing transformation to visual meteor magnitudes under the geometric model.

Usage

```
vmgeom_vst_from_magn(m, lm)

vmgeom_vst_to_r(tm, log = FALSE, deriv_degree = 0L)
```

Arguments

`m`	integer; meteor magnitude.
`lm`	numeric; limiting magnitude.
`tm`	numeric; transformed magnitude.
`log`	logical; if `TRUE`, the logarithm of the population index `r` is returned.
`deriv_degree`	integer; the order of the derivative at `tm` to return instead of `r` or `log(r)`. Must be `0`, `1`, or `2`.

Details

`vmgeom_vst_from_magn` maps visual meteor magnitudes onto a scale on which their variance no longer depends on the population index `r`, provided the magnitudes follow the geometric model. The limiting magnitude `lm` enters the mapping, so it needs no further attention in the subsequent analysis. The variance of the transformed magnitudes is close to `1.0`, and their mean estimates `r` through `vmgeom_vst_to_r`.

The transformation is a monotone rescaling of the underlying quantity used by the rate-based estimator, which is formed from the perception probabilities of two magnitudes one class apart. `vmgeom_vst_to_r` maps the mean of the transformed magnitudes back onto `r`. See `vignette("vmgeom")` for the derivation, the properties of the resulting estimator and a comparison with the other methods the package offers.

Two consequences are relevant when calling these functions. Since that quantity lies between `0` and `1`, the transformed magnitudes are bounded as well, and the upper bound corresponds to $r = 1$. And once both perception probabilities are close to `1.0`, it no longer distinguishes how bright a meteor actually was, so bright meteors are pooled into a single class.

The variance is close to `1.0` over $1.4 \le r \le 4.0$, which covers the $1.7 \le r \le 3.3$ met in practice, and degrades outside that window. This matters when the transformed magnitudes are used as a response in a linear model, as vmgeom_vst_lm does; it does not affect reading a single `r` off their mean.

The back-transformation carries a small systematic deviation that does not shrink as the sample grows. In the practical range it stays below `0.8%` of `r`, and is negligible against the random error, but it grows for larger `r` and comes to dominate in very large samples. Where this matters, the rate-based estimator shown in `vignette("vmgeom")` is unbiased for `1/r`, and `vmgeom_glm` fits the exact likelihood.

The numerical form of the transformation is version-specific and may change substantially in future releases. Do not rely on equality of transformed values across package versions.

Value

- `vmgeom_vst_from_magn`: numeric value, the transformed meteor magnitude.
- `vmgeom_vst_to_r`: numeric value of the population index `r`, derived from the mean of `tm`.

The argument `deriv_degree` can be used to apply the delta method. If `log = TRUE`, the logarithm of `r` is returned.

`vmgeom_vst_to_r` is defined for every `tm` the transformation can produce and returns `NA` only for values it cannot, that is negative ones and those above the upper bound; `tm = 0` yields `Inf`. The recovered `r` is strictly monotone in `tm`, so an implausible estimate — as sparse data may produce in a predictive model — comes back too large, but never as `NA`.

Note

The transformation is based on the perception probabilities produced by vmperception.

The transformation uses the perception probability one magnitude class below `m`, which vanishes for `lm - m <= 0.5`. Such magnitudes are transformed to `0`, including the faintest class still inside the support of the model. This is the correct value — the ratio is genuinely zero there — and those meteors belong in the mean; do not exclude them. Magnitudes outside the support, that is `lm - m <= -0.5`, likewise yield `0` and should as usual be removed beforehand.

See Also

vmgeom vmperception vmgeom_glm vmgeom_vst_lm `vignette("vmgeom")`

Examples

```
N <- 100
r <- 2.0
limmag <- 6.3

# Simulate magnitudes
m <- rvmgeom(N, limmag, r)

# Variance-stabilizing transformation
tm <- vmgeom_vst_from_magn(m, limmag)
tm_mean <- mean(tm)
tm_var <- var(tm)

# Estimator for r from the transformed mean
r_hat <- vmgeom_vst_to_r(tm_mean)
```

```
# Derivative dr/d(tm) at tm_mean (needed for the delta method)
dr_dtm <- vmgeom_vst_to_r(tm_mean, deriv_degree = 1L)

# Variance of the sample mean of tm
var_tm.mean <- tm_var / N

# Delta method: variance and standard error of r_hat
var_r.hat <- (dr_dtm^2) * var_tm.mean
se_r.hat <- sqrt(var_r.hat)

# Results
print(r_hat)
print(se_r.hat)

# The transformation depends on the magnitude only through the difference
# to the limiting magnitude. Bright meteors run into the upper bound,
# where the transformed value no longer resolves their brightness.
old_par <- par(mfrow = c(1, 1))
plot(
    function(dm) vmgeom_vst_from_magn(0.0, dm),
    -0.5, 10.0,
    main = paste(
        "variance-stabilizing transformation of",
        "visual meteor magnitudes"
    ),
    col = "blue",
    xlab = "dm",
    ylab = "tm"
)
abline(h = vmgeom_vst_from_magn(0.0, 100.0), lty = "dashed")

par(old_par)
```

`vmgeom_vst_lm` *Linear Model of Variance-Stabilized Visual Meteor Magnitudes*

Description

Fits visual meteor magnitudes as an ordinary linear model after applying the variance-stabilizing transformation of the geometric model, and returns the estimates on the scale of the population index `r`.

Usage

```
vmgeom_vst_lm(formula, data, ...)

## S3 method for class 'vmgeom_vst_lm'
predict(
  object,
  newdata,
```

```
  type = c("r", "inv_r", "log_r", "tm"),
  bias_correction = FALSE,
  ...
)
```

Arguments

`formula`	an object of class stats::formula; the response must be a two-column matrix `cbind(magn, lim_magn)`.
`data`	a data.frame containing the variables of the model.
`...`	further arguments passed to stats::lm respectively stats::predict.lm.
`object`	an object returned by `vmgeom_vst_lm`.
`newdata`	optional data.frame in which to look for variables with which to predict. If omitted, the fitted values are used.
`type`	character; the scale on which to return predictions. One of `"r"`, `"inv_r"`, `"log_r"` or `"tm"`.
`bias_correction`	logical; if `TRUE`, the second-order term of the delta method is applied to correct the bias of the non-linear back-transformation.

Details

The variance-stabilizing transformation maps visual meteor magnitudes onto a scale on which their variance no longer depends on the population index `r`. The transformed magnitudes are therefore homoscedastic by construction, which is precisely what an ordinary linear model assumes, and `vmgeom_vst_lm` fits them as such.

The response must be given as a two-column matrix, where the first column contains the integer meteor magnitudes and the second column the corresponding limiting magnitudes:

```
vmgeom_vst_lm(cbind(magn, lim_magn) ~ x, data = magnitudes)
```

Both are needed because the transformation depends on their difference. The limiting magnitude is a fixed property of each observation and is never estimated. Passing it as part of the response also ensures that it is subsetted together with the remaining data whenever stats::lm drops rows.

Estimating a single global `r` is the intercept-only case:

```
vmgeom_vst_lm(cbind(magn, lim_magn) ~ 1, data = magnitudes, weights = Freq)
```

Since the back-transformation is a power relation, effects combine additively on the logarithmic scale, and a coefficient translates into a change of $\log r$ up to a constant factor.

Visual magnitude observations are usually reported as counts per magnitude class rather than as individual meteors. Such data are passed in aggregated form by supplying the counts as `weights`. One consequence deserves attention: stats::lm treats weights as precision weights and normalizes the residual variance by the number of rows, whereas the counts state how many meteors each row stands for. `predict` corrects for this by deriving the residual scale from `sum(weights)` instead, so that standard errors and confidence limits refer to the number of meteors. Without that correction they come out too wide. Unlike vmgeom_glm, fractional counts need not be rounded beforehand.

Predictions are obtained on the transformed scale and mapped back afterwards with vmgeom_vst_to_r. The confidence limits are the transformed limits of that scale. Since every transformation offered here is monotonic, these limits are exact rather than an approximation and are asymmetric about the estimate. They are formed by stats::predict.lm and therefore rest on the t-distribution, which is exact for a linear model, rather than on the normal approximation used by vmgeom_glm.

The reported `se.fit` is a first-order delta-method value, $|g'(t)|\,\mathrm{se}(t)$, matching the behaviour of stats::predict.glm and of vmgeom_glm. Setting `bias_correction = TRUE` additionally applies the second-order term, which corresponds to `deriv_degree = 2L` of vmgeom_vst_to_r. It corrects the curvature of the back-transformation and is small at the sample sizes met in practice. It does *not* address the systematic deviation of the back-transformation itself, which does not shrink as the sample grows; see `vignette("vmgeom")`. Where that deviation matters, vmgeom_glm fits the exact likelihood.

Value

- `vmgeom_vst_lm`: an object of class `vmgeom_vst_lm` inheriting from stats::lm.
- `predict`: a numeric vector of predictions, or the list returned by stats::predict.lm with its components transformed, if `se.fit = TRUE` or `interval` is given.

See Also

vmgeom_vst vmgeom vmgeom_glm stats::lm stats::predict.lm `vignette("vmgeom")`

Examples

```
# Simulate magnitudes with a constant population index
set.seed(1)
lim_magn <- rep(c(5.8, 6.1, 6.4), each = 200)
magn <- mapply(\(lim_magn) rvmgeom(1L, lim_magn, r = 2.0), lim_magn)
obs <- data.frame(lim_magn = lim_magn, magn = magn)

# Fit the model
fit <- vmgeom_vst_lm(cbind(magn, lim_magn) ~ 1, data = obs)

# The model has no covariates, so any single row predicts the global r-value
newdata <- data.frame(row.names = "")
predict(fit, newdata) # r
predict(fit, newdata, type = "inv_r") # 1/r
predict(fit, newdata, type = "log_r") # log(r)

# ... including confidence limits
predict(fit, newdata, interval = "confidence")
```

vmideal *Ideal Distribution of Visual Meteor Magnitudes*

Description

Density, distribution function, quantile function, and random generation for the ideal distribution of visual meteor magnitudes.

Usage

```
dvmideal(m, lm, psi, log = FALSE, perception_fun = vmperception)

pvmideal(
  m,
  lm,
  psi,
  lower.tail = TRUE,
  log = FALSE,
  perception_fun = vmperception
)

qvmideal(p, lm, psi, lower.tail = TRUE, perception_fun = vmperception)

rvmideal(n, lm, psi, perception_fun = vmperception)

cvmideal(lm, psi, log = FALSE, perception_fun = vmperception)
```

Arguments

`m`	integer; visual meteor magnitude.
`lm`	numeric; limiting magnitude.
`psi`	numeric; the location parameter of the probability distribution. `Inf` is permitted and denotes the geometric limit the distribution converges to as `psi` grows without bound, with a population index of $r = 10^{0.4}$. It is the value predict.vmideal_glm reports where the data no longer determine `psi`, and may be passed on unchanged. `-Inf` is not permitted, as no limiting distribution exists there.
`log`	logical; if `TRUE`, probabilities are returned as `log(p)`.
`perception_fun`	function; optional perception probability function. The default is vmperception.
`lower.tail`	logical; if `TRUE` (default), probabilities are $P[M < m]$; otherwise, $P[M \geq m]$.
`p`	numeric; probability.
`n`	numeric; count of meteor magnitudes.

Details

The density of the ideal distribution of meteor magnitudes is

$$f(m) = \frac{\mathrm{d}p}{\mathrm{d}m} = \frac{3}{2}\,\log(r)\sqrt{\frac{r^{3\,\psi + 2\,m}}{(r^\psi + r^m)^5}}$$

where m denotes the continuous (real-valued) meteor magnitude, $r = 10^{0.4} \approx 2.51189\ldots$ is a constant, and ψ is the only parameter of this magnitude distribution.

In visual meteor observations, magnitudes are usually estimated as integer values. Hence, this distribution is discrete and its probability mass function is given by

$$P[M = m] \sim \begin{cases} g(m_{\mathrm{lim}} - m) \int\limits_{m-0.5}^{m+0.5} f(u)\,\mathrm{d}u, & \text{if } m_{\mathrm{lim}} - m > -0.5,\\ 0 & \text{otherwise,} \end{cases}$$

where m_{lim} denotes the limiting (non-integer) magnitude of the observation, and m the integer meteor magnitude. The function $f(\cdot)$ is the continuous density of the ideal magnitude distribution, and $g(\cdot)$ denotes the perception probability function.

If a different perception probability function `perception_fun` is supplied, it must have the signature `function(x)` and return the perception probabilities of the difference `x` between the limiting magnitude and the meteor magnitude. If `x >= 15.0`, the `perception_fun` function should return a perception probability of `1.0`. The argument `perception_fun` is resolved using match.fun.

Value

- `dvmideal`: density
- `pvmideal`: distribution function
- `qvmideal`: quantile function
- `rvmideal`: random generation
- `cvmideal`: partial convolution of the ideal distribution of meteor magnitudes with the perception probabilities.

The length of the result is determined by `n` for `rvmideal`, and is the maximum of the lengths of the numeric vector arguments for the other functions. All arguments are vectorized; standard R recycling rules apply.

Since the distribution is discrete, `qvmideal` and `rvmideal` always return integer values. `qvmideal` returns `NA` with a warning for probabilities outside $[0, 1]$. `cvmideal` returns `NA` with a warning if `lm` and `psi` are both `Inf` or both `-Inf`, as the convolution is then indeterminate.

References

Richter, J. (2018) *About the mass and magnitude distributions of meteor showers*. WGN, Journal of the International Meteor Organization, vol. 46, no. 1, p. 34-38

See Also

mideal vmperception

Examples

```
N <- 100
psi <- 5.0
limmag <- 6.5
(m <- seq(6, -4))

# discrete density of `N` meteor magnitudes
(freq <- round(N * dvmideal(m, limmag, psi)))

# log likelihood function
lld <- function(psi) {
    -sum(freq * dvmideal(m, limmag, psi, log = TRUE))
}

# maximum likelihood estimation (MLE) of psi
est <- optim(2, lld, method = "Brent", lower = 0, upper = 8, hessian = TRUE)

# estimations
est$par # mean of psi

# generate random meteor magnitudes
m <- rvmideal(N, limmag, psi)

# log likelihood function
llr <- function(psi) {
    -sum(dvmideal(m, limmag, psi, log = TRUE))
}

# maximum likelihood estimation (MLE) of psi
est <- optim(2, llr, method = "Brent", lower = 0, upper = 8, hessian = TRUE)

# estimations
est$par # mean of psi
sqrt(1 / est$hessian[1][1]) # standard deviation of psi

m <- seq(6, -4, -1)
p <- vismeteor::dvmideal(m, limmag, psi)
barplot(
    p,
    names.arg = m,
    main = paste0("Density (psi = ", psi, ", limmag = ", limmag, ")"),
    col = "blue",
    xlab = "m",
    ylab = "p",
    border = "blue",
    space = 0.5
)
axis(side = 2, at = pretty(p))

plot(
    \(lm) vismeteor::cvmideal(lm, psi, log = TRUE),
    -5, 10,
```

```
    main = paste0(
        "Partial convolution of the ideal meteor magnitude distribution\n",
        "with the perception probabilities (psi = ", psi, ")"
    ),
    col = "blue",
    xlab = "lm",
    ylab = "log(rate)"
)
```

---

vmideal_glm *Generalized Linear Model for the Ideal Distribution of Visual Meteor Magnitudes*

Description

Fits the ideal distribution of visual meteor magnitudes as a generalized linear model, so that the location parameter `psi` can be estimated as a function of covariates.

Usage

```
vmideal_glm(
  formula,
  data,
  ...,
  perception_fun = vmperception,
  psi_range = c(-Inf, Inf)
)

## S3 method for class 'vmideal_glm'
predict(
  object,
  newdata,
  type = c("psi", "link"),
  se.fit = FALSE,
  level = 0.95,
  ...
)
```

Arguments

`formula`	an object of class stats::formula; the response must be a two-column matrix `cbind(magn, lim_magn)`.
`data`	a data.frame containing the variables of the model.
`...`	further arguments passed to stats::glm respectively stats::predict.glm.
`perception_fun`	function; perception probability function (optional). Default is vmperception.
`psi_range`	numeric; the range of location parameters considered when inverting the mean. The default leaves `psi` unrestricted.

`object`	an object returned by `vmideal_glm`.
`newdata`	optional data.frame in which to look for variables with which to predict. If omitted, the fitted linear predictors are used.
`type`	character; the scale on which to return predictions. `"psi"` (the default) returns the location parameter, `"link"` the linear predictor. The link is the identity, so both agree, except where the data no longer determine `psi`: `"psi"` is `Inf` there, while `"link"` remains the finite value the fit stopped at.
`se.fit`	logical; if `TRUE`, standard errors and confidence limits are returned in addition to the predictions.
`level`	numeric; the confidence level of the returned interval.

Details

In the ideal distribution of visual meteor magnitudes, the location parameter `psi` is usually estimated as a single global value. `vmideal_glm` allows `psi` to depend on covariates instead, such as the solar longitude or the limiting magnitude.

The response must be given as a two-column matrix, where the first column contains the integer meteor magnitudes and the second column the corresponding limiting magnitudes:

```
vmideal_glm(cbind(magn, lim_magn) ~ x, data = magnitudes)
```

The limiting magnitude is a fixed property of each observation and is never estimated. Passing it as part of the response ensures that it is subsetted together with the remaining data whenever stats::glm drops rows, for example through `subset` or `na.action`.

The link is the identity, since `psi` is unrestricted on the real line. The fit is an ordinary stats::glm object, so `summary`, `anova`, `AIC`/`BIC` and select_knots apply unchanged.

Visual magnitude observations are usually reported as counts per magnitude class rather than as individual meteors. Such data can be passed in aggregated form by supplying the counts as `weights`:

```
vmideal_glm(cbind(magn, lim_magn) ~ x, data = magnitudes, weights = Freq)
```

This is equivalent to repeating every row `Freq` times, and yields identical coefficients, standard errors, deviance and `AIC`. Fractional counts should be rounded with vmtable beforehand.

Unlike vmgeom_glm, the fit is not an exact maximum likelihood estimate but a quasi-likelihood one: the mean of the ideal distribution is not a sufficient statistic for `psi`. It is consistent, and for a few thousand meteors with `psi` at or below the limiting magnitude it differs from the maximum likelihood estimate by a few hundredths. For small samples, or a `psi` well above the limiting magnitude, the difference can reach a whole magnitude, and the standard errors are then no longer conservative. Use stats::optim on dvmideal directly when an exact maximum likelihood estimate of a single global `psi` is required. `vignette("vmideal")` sets out the reason and the size of the effect.

Value

- `vmideal_glm`: an object of class `vmideal_glm` inheriting from stats::glm.
- `predict`: a numeric vector of predictions, or a list with the components `fit`, `se.fit`, `lwr` and `upr` if `se.fit = TRUE`.

Range of psi

psi is unrestricted, and the default psi_range leaves it that way. What the data can resolve is governed by the distance between psi and the limiting magnitude: psi is estimated reliably as long as it does not exceed the limiting magnitude by more than about two to three magnitudes. For visual meteor observations, where psi is typically around 4 to 6 and limiting magnitudes lie near 6, this leaves ample room.

Beyond that the mean of the distribution flattens, and once it has reached the limit it converges to, no data can distinguish a larger psi from a smaller one. predict(type = "psi") then returns Inf. That is a result rather than a failure: the magnitudes are geometric with $r = 10^{0.4}$, and Inf says that psi is bounded below by the fit rather than determined by it. It also arises when the mean magnitude of the data lies above anything the model can produce, whether by chance in a small sample or because the ideal distribution does not describe them. The linear predictor stays finite throughout, so type = "link", summary and anova remain usable. Just short of that point the iteration may run out of steps; such a fit is reported with a warning rather than returned as if it had converged.

An infinite psi_range bound is permitted, and so is the Inf that predict(type = "psi") returns: it may be passed straight to dvmideal and its companions, which evaluate the geometric limit there.

Predictions and their uncertainties are obtained on the scale of the linear predictor. Since the link is the identity, the confidence limits are the ordinary symmetric limits fit +/- z * se.fit.

See Also

vmideal vmgeom_glm vmperception stats::glm stats::predict.glm

Examples

```
# Simulate magnitudes whose location parameter depends on a covariate
set.seed(1)
lim_magn <- rep(c(5.8, 6.1, 6.4), each = 120)
x <- rep(seq(-1.5, 1.5, length.out = 12), each = 30)
psi <- 4.0 + 0.8 * x
magn <- mapply(\(lim_magn, psi) rvmideal(1L, lim_magn, psi), lim_magn, psi)
obs <- data.frame(x = x, lim_magn = lim_magn, magn = magn)

# Fit the model
fit <- vmideal_glm(cbind(magn, lim_magn) ~ x, data = obs)
summary(fit)

# Location parameter at selected covariate values
newdata <- data.frame(x = c(-1, 0, 1))
predict(fit, newdata) # psi

# ... including confidence limits
predict(fit, newdata, se.fit = TRUE)
```

vmideal_vst — *Variance-stabilizing Transformation for the Ideal Distribution of Visual Meteor Magnitudes*

Description

Applies a variance-stabilizing transformation to meteor magnitudes under the assumption of the ideal magnitude distribution.

Usage

```
vmideal_vst_from_magn(m, lm)

vmideal_vst_to_psi(tm, lm, deriv_degree = 0L)
```

Arguments

`m`	integer; the meteor magnitude.
`lm`	numeric; limiting magnitude.
`tm`	numeric; transformed magnitude.
`deriv_degree`	integer; the degree of the derivative at `tm` to return instead of `psi`. Must be `0`, `1` or `2`.

Details

Many linear models require the variance of visual meteor magnitudes to be homoscedastic. The function `vmideal_vst_from_magn` applies a transformation that produces homoscedastic distributions of visual meteor magnitudes if the underlying magnitudes follow the ideal magnitude distribution. In this sense, the transformation acts as a normalization of meteor magnitudes and yields a variance close to `1.0`.

The ideal distribution of visual meteor magnitudes depends on the parameter psi and the limiting magnitude `lm`, resulting in a two-parameter distribution. Without detection probabilities, the magnitude distribution reduces to a pure ideal magnitude distribution, which depends only on the parameter `psi`. Since the limiting magnitude `lm` is a fixed parameter and never estimated statistically, the magnitudes can be transformed such that, for example, the mean of the transformed magnitudes directly provides an estimate of `psi` using the function `vmideal_vst_to_psi`.

What the transformation resolves is the difference between `psi` and `lm`, so its range follows the limiting magnitude: `psi` is recovered from about $\texttt{lm} - 16.5$ up to about $\texttt{lm} + 3$. At a limiting magnitude of around `6.0` this amounts to $-10 \le \texttt{psi} \le 9$. Within that range the estimate is accurate to a few hundredths of a magnitude, degrading towards the upper end, where the mean of the distribution begins to flatten. Outside it `vmideal_vst_to_psi` returns `NA` or `Inf` rather than an unreliable value; see below. The numerical form of the transformation is version-specific and may change substantially in future releases. Do not rely on equality of transformed values across package versions.

Value

- vmideal_vst_from_magn: a numeric value, the transformed meteor magnitude.
- vmideal_vst_to_psi: a numeric value of the parameter psi, derived from the mean of tm. The argument deriv_degree can be used to apply the delta method.

vmideal_vst_to_psi returns Inf for a tm below 0.02. A vanishing mean of the transformed magnitudes is what an unbounded psi produces, and the magnitudes are then geometric with $r = 10^{0.4}$, which dvmideal evaluates for an infinite psi. The derivatives do not exist there and are NA. Above the range the transformation covers, at a tm beyond 8.22, the estimate is NA as well: a psi that far below the limiting magnitude is not a limit but simply out of range.

Note

The internal approximations used here are derived from the perception probabilities produced by vmperception. For details on the derivation, see the script inst/derivation/vmideal_vst.R in the package's source code.

See Also

vmideal mideal vmperception

Examples

```
N <- 100
psi <- 5.0
limmag <- 6.3

# Simulate magnitudes
m <- rvmideal(N, limmag, psi)

# Variance-stabilizing transformation
tm <- vmideal_vst_from_magn(m, limmag)
tm_mean <- mean(tm)
tm_var <- var(tm)

# Estimator for psi from the transformed mean
psi_hat <- vmideal_vst_to_psi(tm_mean, limmag)

# Derivative d(psi)/d(tm) at tm_mean (needed for the delta method)
dpsi_dtm <- vmideal_vst_to_psi(tm_mean, limmag, deriv_degree = 1L)

# Variance of the sample mean of tm
var_tm.mean <- tm_var / N

# Delta method: variance and standard error of psi_hat
var_psi.hat <- (dpsi_dtm^2) * var_tm.mean
se_psi.hat <- sqrt(var_psi.hat)

# Results
print(psi_hat)
print(se_psi.hat)
```

`vmperception` *Perception Probabilities of Visual Meteor Magnitudes*

Description

Provides the perception probability of visual meteor magnitudes.

Usage

```
vmperception(dm)
```

Arguments

`dm` numeric; difference between the limiting magnitude and the meteor magnitude.

Details

The perception probabilities of *Koschack R., Rendtel J., 1990b* are estimated with the formula

$$p(dm) = \begin{cases} 1.0 - \exp\left(-z(dm + 0.5)\right) & \text{if } dm > -0.5, \\ 0.0 & \text{otherwise,} \end{cases}$$

where

$$z(x) = 0.0037\,x + 0.0019\,x^2 + 0.00271\,x^3 + 0.0009\,x^4$$

and `dm` is the difference between the limiting magnitude and the meteor magnitude.

Value

This function returns the visual perception probabilities.

References

Koschack R., Rendtel J., 1990b *Determination of spatial number density and mass index from visual meteor observations (II).* WGN 18, 119–140.

Examples

```
# Perception probability of visually estimated meteor of magnitude 3.0
# with a limiting magnitude of 5.6.
vmperception(5.6 - 3.0)

# plot
old_par <- par(mfrow = c(1, 1))
plot(
    vmperception,
    -0.5, 8,
    main = paste(
        "perception probability of",
```

```
        "visual meteor magnitudes"
    ),
    col = "blue",
    xlab = "dm",
    ylab = "p"
)

par(old_par)
```

---

vmtable *Rounds a contingency table of meteor magnitude frequencies*

Description

The meteor magnitude contingency table of VMDB contains half meteor counts (e.g. `3.5`). This function converts these frequencies to integer values.

Usage

```
vmtable(mt)
```

Arguments

`mt`	table; A two-dimensional contingency table of meteor magnitude frequencies.

Details

The contingency table of meteor magnitudes `mt` must be two-dimensional. The row names refer to the magnitude observations. Column names must be integer meteor magnitude values. Also, the columns must be sorted in ascending or descending order of meteor magnitude.

A sum-preserving algorithm is used for rounding. It ensures that the total frequency of meteors per observation is preserved. The marginal frequencies of the magnitudes are also preserved with the restriction that the deviation is at most ± 0.5. If the total sum of a meteor magnitude is integer, then the deviation is ± 0.

The algorithm is unbiased: for a fixed observation order it preserves the original totals without introducing systematic drift, even though each run follows the deterministic sequence dictated by the observed counts and their ordering.

Value

A rounded contingency table of meteor magnitudes is returned.

Note

Internally the counts are doubled to half-meteor units, leftover halves are alternated between rows so column margins stay within ± 0.5, and when the grand total is odd the matrix is temporarily mirrored so the unavoidable surplus meteor originates from the opposite end of the magnitude scale rather than always favouring the faintest bin. The mirroring is only the initial condition; the loop then processes the table cell by cell so the rounding direction alternates between bright and faint magnitudes depending on the current row and column state.

Examples

```
# For example, create a contingency table of meteor magnitudes
mt <- as.table(matrix(
    c(
        0.0, 0.0, 2.5, 0.5, 0.0, 1.0,
        0.0, 1.5, 2.0, 0.5, 0.0, 0.0,
        1.0, 0.0, 0.0, 3.0, 2.5, 0.5
    ),
    nrow = 3, ncol = 6, byrow = TRUE
))
colnames(mt) <- seq(6)
rownames(mt) <- c("A", "B", "C")
mt
margin.table(mt, 1)
margin.table(mt, 2)

# contingency table with integer values
(mt_int <- vmtable(mt))
margin.table(mt_int, 1)
margin.table(mt_int, 2)
```

Index