Package: Rtractor (via r-universe)

July 12, 2026

Title Complexity and Nonlinear Time Series Analysis for Physiological Signals

Version 0.1.0

Description A staging- and signal-agnostic toolkit of complex systems measures for physiological time series: entropy (sample, permutation, approximate), fractal and multifractal analysis (Higuchi dimension, DFA, MFDFA), Lyapunov exponents, multiscale entropy, and recurrence quantification analysis. Core algorithms wrap established C/C++/Fortran reference implementations rather than reimplementing them in pure R, to preserve numerical parity with the original methods literature. Some wrapped reference implementations are GPL-licensed (see inst/COPYRIGHTS), which is why the package as a whole is licensed GPL (>= 2) rather than MIT. Designed to sit underneath other Circadia Lab and CoDe-Neuro Lab packages (mrpheus, hypnoR, zeitR, dynR) as a shared complexity-metrics layer, while remaining fully usable standalone on any numeric time series.

License GPL (>= 2)

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports Rcpp, stats

LinkingTo Rcpp

Suggests covr, ggplot2 (>= 3.4.0), knitr, pkgdown, rmarkdown, testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

URL https://rtractor.circadia-lab.uk, https://github.com/circadia-bio/Rtractor

BugReports https://github.com/circadia-bio/Rtractor/issues

Config/roxygen2/version 8.0.0

Repository https://circadia-bio.r-universe.dev

Date/Publication 2026-07-12 23:44:12 UTC

RemoteUrl https://github.com/circadia-bio/Rtractor

RemoteRef v0.1.0

RemoteSha 5243c7c85746652c0965aa94ea75f762eeccdda3

Contents

`chhabra_jensen` *Multifractal Spectrum via the Chhabra-Jensen Method*

Description

Estimates the multifractal spectrum f(alpha) and generalised dimension spectrum D(q) of a strictly positive time series using the direct box-counting method of Chhabra & Jensen (1989). Clean-room C++ reimplementation from the published algorithm (the reference MATLAB implementation consulted, `ChhabraJensen_Yuj_w0.m`, co-authored by L. Franca, had no license header; see `inst/COPYRIGHTS`). The moments core was validated against a Python transliteration of that reference on synthetic test data (exact match to displayed precision).

Usage

```
chhabra_jensen(
  x,
  q_values = c(seq(-10, -0.1, by = 0.1), seq(1, 10, by = 0.1)),
  scales = NULL
)
```

Arguments

x	Numeric vector, strictly positive (treat it as a measure — e.g. apply a sigmoid transform first if your data can be negative). `length(x)` must be evenly divisible by `2^scale` for every value in `scales`, i.e. dyadic lengths (powers of two) work best.
q_values	Numeric vector of multifractal orders. Per the original author's guidance, values strictly between 0 and 1 (exclusive) are numerically unstable for this method and best avoided – a warning is issued if any are supplied. Default skips that range: `c(seq(-10, -0.1, by = 0.1), seq(1, 10, by = 0.1))`.
scales	Integer vector of box-counting scale exponents; the box size at each scale is `2^scale`. Default `1:floor(log2(length(x)) - 2)`, which keeps at least 4 points per box at the coarsest scale.

Value

A list with:

alpha	Singularity strength alpha(q).
falpha	Multifractal spectrum f(alpha(q)).
Dq	Generalised dimension spectrum D(q).
r_squared_alpha, r_squared_falpha, r_squared_Dq	R-squared of the linear fit underlying each of the above, per q – inspect these before trusting a given q value's estimate.
q	The q_values used.
mu_scale, Ma, Mf, Md	The underlying regression inputs, included for completeness.

References

Chhabra A, Jensen RV. Direct determination of the f(alpha) singularity spectrum. Phys Rev Lett 1989;62:1327-1330.

Franca LGS, Miranda JGV, Leite M, Sharma NK, Walker MC, Lemieux L, Wang Y. Fractal and multifractal properties of electrographic recordings of human brain activity: toward its use as a signal feature for machine learning in clinical applications. Front Physiol 2018;9:1767. Compares MF-DFA, MF-DMA, and Chhabra-Jensen on simulated and human intracranial EEG data and finds Chhabra-Jensen the most stable/reliable of the three – the basis for implementing this method (rather than MF-DFA) alongside `mfdma()` in Rtractor.

Examples

```
set.seed(1)
x <- abs(rnorm(1024)) + 0.01
res <- chhabra_jensen(x, scales = 1:6)
plot(res$alpha, res$falpha, type = "b")
```

`dfa` *Detrended Fluctuation Analysis (DFA)*

Description

Estimates the scaling exponent alpha of a time series using Detrended Fluctuation Analysis (Peng et al. 1994). This is a direct C++ port of the reference `dfa.c` implementation distributed by PhysioNet (Mietus, Peng & Moody), validated to reproduce the original compiled binary's output exactly on synthetic test data — see `inst/COPYRIGHTS`.

Usage

```
dfa(
  x,
  order = 1,
  min_box = NULL,
  max_box = NULL,
  integrate = TRUE,
  sliding_window = FALSE
)
```

Arguments

`x`	Numeric vector. The time series to analyse.
`order`	Integer >= 1. Order of the polynomial detrending fit (1 = linear detrending, the DFA default; 2 = quadratic, etc.). Corresponds to `nfit - 1` in the original `dfa.c`.
`min_box, max_box`	Integer or `NULL`. Smallest/largest box size to evaluate. Defaults (as in the original): `min_box = 2 * (order + 1)`, `max_box = length(x) / 4`.
`integrate`	Logical. If `TRUE` (default), `x` is treated as an increment series and cumulatively summed before analysis (the usual DFA convention — e.g. pass RR-interval deviations, not a cumulative profile). Set `FALSE` if `x` is already an integrated/cumulative profile.
`sliding_window`	Logical. Use overlapping (sliding) windows instead of non-overlapping boxes. Default `FALSE`.

Value

A list with:

n	Box sizes evaluated (integer vector).
F	RMS fluctuation at each box size (numeric vector).
alpha	The DFA scaling exponent: the slope of log10(F) on log10(n).

References

Peng C-K, Buldyrev SV, Havlin S, Simons M, Stanley HE, Goldberger AL. Mosaic organization of DNA nucleotides. Phys Rev E 1994;49:1685-1689.

Examples

```
set.seed(1)
dfa(rnorm(2000))
```

---

higuchi_fd *Higuchi Fractal Dimension*

Description

Estimates the fractal dimension of a time series using Higuchi's (1988) curve-length algorithm. A clean-room C++ reimplementation, validated against a MATLAB reference implementation on synthetic test data — see inst/COPYRIGHTS.

Usage

```
higuchi_fd(x, k_max = 10L)
```

Arguments

x	Numeric vector. The time series to analyse.
k_max	Integer (>= 2). Maximum sub-series interval k. Choose based on where the log-log curve plateaus for your signal (see Doyle et al. for guidance in postural-sway / physiological contexts).

Value

A list with:

k	The k values evaluated, 1:k_max.
L	Average curve length at each k.
hfd	The Higuchi Fractal Dimension: negative slope of log(L) on log(1/k).

References

Higuchi T. Approach to an irregular time series on the basis of the fractal theory. Physica D 1988;31(2):277-283.

Examples

```
set.seed(1)
higuchi_fd(rnorm(1000), k_max = 10)
```

`hjorth_parameters` *Hjorth Mobility and Complexity*

Description

Computes the Hjorth parameters (Hjorth 1970): mobility, a proxy for mean frequency derived from the variance ratio of the first difference to the signal itself; and complexity, a proxy for bandwidth/irregularity derived from the second difference. Ported from Lucas Franca's own `mrpheus` package (AASM staging feature pipeline). Uses Bessel-corrected (`ddof = 1`) variance throughout, deliberately matching R's `var()` convention – this was mrpheus's original design choice, and differs slightly from the `antropy` Python library's population-variance (`ddof = 0`) convention. The two converge as `length(x)` grows but differ meaningfully for short series; see `inst/COPYRIGHTS`.

Usage

```
hjorth_parameters(x)
```

Arguments

`x`	Numeric vector. The time series to analyse.

Value

A named list with `mobility` and `complexity`.

References

Hjorth B. EEG analysis based on time domain properties. Electroencephalogr Clin Neurophysiol 1970;29(3):306-310.

Examples

```
set.seed(1)
hjorth_parameters(rnorm(1000))
```

`mfdma` *Multifractal Detrending Moving Average (MFDMA)*

Description

Estimates the multifractal scaling properties of a time series using the detrending moving average algorithm (Gu & Zhou 2010). Clean-room C++ reimplementation from the published algorithm (the reference MATLAB implementation consulted, `MFDMA_1D.m`, had no license header; see `inst/COPYRIGHTS`). The segment-fluctuation core was validated against a Python transliteration of that reference on synthetic test data (exact match to displayed precision).

Usage

```
mfdma(
  x,
  n_min = 10L,
  n_max = NULL,
  n_scales = 30L,
  theta = 0,
  q = seq(-4, 4, by = 0.1)
)
```

Arguments

`x`	Numeric vector. The time series to analyse.
`n_min`, `n_max`	Integer. Lower/upper bound of the segment size `n`. Following the reference implementation's guidance: `n_min` around `10`; `n_max` around 10% of `length(x)`. Defaults: `n_min` = `10`, `n_max` = `round(length(x) / 10)`.
`n_scales`	Integer. Number of segment sizes to evaluate (log-spaced between `n_min` and `n_max`). Default `30`.
`theta`	Numeric in `[0, 1]`. Position of the moving-average window: `0` (default, recommended) = backward MFDMA, `0.5` = centered, `1` = forward.
`q`	Numeric vector. Multifractal orders to evaluate. Default `seq(-4, 4, by = 0.1)`.

Value

A list with:

`n`	Segment sizes evaluated.
`Fq`	Matrix of the q-th order fluctuation function (segment size x q).
`tau`	Multifractal scaling exponent tau(q).
`alpha`	Singularity strength alpha(q) (trimmed at both ends by the local-slope smoothing window; shorter than `q`).
`f`	Multifractal spectrum f(alpha).
`q`	The `q` values corresponding to `alpha`/`f` (trimmed to match).

References

Gu GF, Zhou WX. Detrending moving average algorithm for multifractals. Phys Rev E 2010;82:011136.

Examples

```
set.seed(1)
x <- rnorm(4000)
res <- mfdma(x, n_min = 10, n_max = 400, n_scales = 20)
plot(res$alpha, res$f, type = "b")
```

---

multiscale_entropy *Multiscale Entropy (MSE)*

Description

Estimates the complexity of a time series across a range of temporal scales (Costa, Goldberger & Peng 2002): the series is coarse-grained (non-overlapping block-averaged) at each scale factor, and `sample_entropy()` is computed on each coarse-grained series, using a tolerance held *fixed* relative to the original (not the coarse-grained) series' standard deviation – this is the standard MSE convention and essential for entropy values to be comparable across scales. Direct C++ port of the coarse-graining + sample-entropy core in PhysioNet's reference `mse.c` (Costa), validated to reproduce the compiled reference binary's output exactly (to its own displayed precision) on synthetic test data. See `inst/COPYRIGHTS`.

Usage

```
multiscale_entropy(x, scale_max = 20L, m = 2L, r = 0.15)
```

Arguments

`x`	Numeric vector. The time series to analyse.
`scale_max`	Integer >= 1. Largest scale factor to evaluate; MSE is computed at every integer scale from 1 to `scale_max`. Default `20` (the standard MSE convention).
`m`	Integer >= 1. Template length, passed to `sample_entropy()`. Default `2` (the standard MSE convention).
`r`	Numeric > 0. Tolerance as a fraction of the *original* series' standard deviation, passed to `sample_entropy()`. Default `0.15` (the standard MSE convention).

Value

A list with:

`scale`	The scale factors evaluated, `1:scale_max`.
`mse`	Sample entropy at each scale (may contain `NA` at large scales, where the coarse-grained series becomes too short to estimate reliably).
`m`, `r`	The parameters used, echoed back for reference.

References

Costa M, Goldberger AL, Peng CK. Multiscale entropy analysis of complex physiologic time series. Phys Rev Lett 2002;89:068102.

Examples

```
set.seed(1)
res <- multiscale_entropy(rnorm(2000), scale_max = 10)
plot(res$scale, res$mse, type = "b")
```

---

num_zerocross *Number of Zero Crossings*

Description

Counts sign changes in a time series – a simple time-domain proxy for dominant frequency / signal roughness, commonly reported alongside Hjorth parameters and fractal dimension in EEG complexity work. Ported from Lucas Franca's own `mrpheus` package (AASM staging feature pipeline), itself validated for exact parity against the `antropy` Python library; re-validated here directly against `antropy` 0.2.2 on synthetic test data (exact match). See `inst/COPYRIGHTS`.

Usage

```
num_zerocross(x)
```

Arguments

`x`	Numeric vector. The time series to analyse.

Value

A length-1 integer: the number of zero crossings.

Examples

```
set.seed(1)
num_zerocross(rnorm(1000))
```

perm_entropy *Permutation Entropy*

Description

Estimates the complexity of a time series using permutation entropy (Bandt & Pompe 2002): the Shannon entropy of the distribution of ordinal patterns ("motifs") of length `order` found in the series. Ported from Lucas Franca's own `mrpheus` package (part of its AASM sleep-staging feature pipeline), itself validated for exact parity against the `antropy` Python library; re-validated here directly against `antropy` 0.2.2 on synthetic test data (exact match to displayed precision). See `inst/COPYRIGHTS`.

Usage

```
perm_entropy(x, order = 3L, delay = 1L, normalize = TRUE)
```

Arguments

`x`	Numeric vector. The time series to analyse.
`order`	Integer >= 2. Length of the ordinal pattern (embedding dimension). Default `3`.
`delay`	Integer >= 1. Time delay between pattern elements. Default `1`.
`normalize`	Logical. If `TRUE` (default), divide by `log(order!)` so the result falls in `[0, 1]`. If `FALSE`, return the raw Shannon entropy in nats (natural log). Note this differs from the `antropy` Python library, which computes the raw value in bits (log base 2); the normalized value is base-independent and matches `antropy` exactly, but the raw values are not directly comparable between the two. See `inst/COPYRIGHTS`.

Value

A length-1 numeric: the permutation entropy.

References

Bandt C, Pompe B. Permutation entropy: a natural complexity measure for time series. Phys Rev Lett 2002;88:174102.

Examples

```
set.seed(1)
perm_entropy(rnorm(1000))
```

`petrosian_fd` *Petrosian Fractal Dimension*

Description

Estimates fractal dimension from the rate of sign changes in a time series' first difference (a fast proxy for signal irregularity). Ported from Lucas Franca's own `mrpheus` package (AASM staging feature pipeline), itself validated for exact parity against the `antropy` Python library; re-validated here directly against `antropy` 0.2.2 on synthetic test data (exact match). See `inst/COPYRIGHTS`.

Usage

```
petrosian_fd(x)
```

Arguments

`x`	Numeric vector. The time series to analyse.

Value

A length-1 numeric: the Petrosian fractal dimension.

References

Petrosian A. Kolmogorov complexity of finite sequences and recognition of different preictal EEG patterns. Proceedings of the Eighth IEEE Symposium on Computer-Based Medical Systems 1995:212-217.

Examples

```
set.seed(1)
petrosian_fd(rnorm(1000))
```

`pmodel` *Simulate a Multifractal Time Series Using the p-Model*

Description

Generates a multiplicative binomial cascade ("p-model") time series (Meneveau & Sreenivasan 1987), optionally fractionally integrated in Fourier space to a prescribed power-spectrum slope (Davis et al. 1997) to produce a continuous, nonstationary series superposed on the multifractal cascade.

Usage

```
pmodel(n_values = 256L, p = 0.375, slope = NULL, seed = 42L)
```

Arguments

n_values	Integer. Desired length of the output series. Internally rounded up to the next power of two for the cascade construction, then truncated back to n_values. Default 256.
p	Numeric in (0, 1). The p-model parameter: values near 0 or 1 produce a very peaked (strongly multifractal) series; values near 0.5 produce a calm series (p = 0.5 gives an exactly constant unit vector – no multifractality at all). Default 0.375.
slope	Numeric or NULL. If supplied, the p-model series is fractionally integrated in Fourier space to have this power-spectrum slope, producing a continuous, non-stationary series (slopes between -1 and -3 give stationary increments; steeper than -3 is differentiable) – see Davis et al. (1997) for the regimes. If NULL (default), no fractional integration is applied.
seed	Integer or NULL. Seeds the random sign draws at *every* cascade level. The original MATLAB implementation reseeds to a fixed seed (42) at every level unconditionally, making the cascade fully deterministic by design (each level's random signs are literally a prefix of the next level's, since the RNG stream restarts from the same point every time). This port preserves that same reseed-every-level structure, with seed (default 42, matching the original) controlling it. Pass seed = NULL to draw fresh randomness at every level instead (no reseeding) – unlike the original, which has no way to disable this.

Details

Clean-room R reimplementation of pmodel.m, written by the late Victor Venema (no license header present in the source file; see inst/COPYRIGHTS). The underlying random sign draws use R's own RNG rather than attempting to replicate MATLAB's specific generator bit-for-bit (not feasible across languages, and not meaningful for a synthetic-data generator characterised by its cascade *rule* rather than particular RNG bytes) – see the seed argument below for how the original's reseed-every-level behaviour is preserved. Validated against two RNG-independent structural properties that hold for *any* random sign sequence: at p = 0.5 the cascade is provably an exact constant vector of 1s, and total mass is conserved exactly at every cascade level (sum(y) doubles per level) for any p – see tests/testthat/test-pmodel.R.

Value

If slope is NULL: a numeric vector of length n_values, the p-model series itself. If slope is supplied: a list with

x	The fractionally integrated series.
y	The underlying p-model series before integration.

References

Meneveau C, Sreenivasan KR. Simple multifractal cascade model for fully developed turbulence. Phys Rev Lett 1987;59:1424-1427.

Davis A, Marshak A, Cahalan R, Wiscombe W. The Landsat scale break in stratocumulus as a three-dimensional radiative transfer effect: implications for cloud remote sensing. J Atmos Sci 1997;54(2):241-260.

Venema V, Bachner S, Rust HW, Simmer C. Statistical characteristics of surrogate data based on geophysical measurements. Nonlin Processes Geophys 2006;13:449-466. (Requested citation for this code by its original author, Victor Venema.)

Examples

```
y <- pmodel(1024, p = 0.3)
range(y)

# p = 0.5 always gives a constant series, regardless of seed:
all(pmodel(64, p = 0.5) == 1)
```

`recurrence_microstate_entropy`

Recurrence Microstates Entropy (maximum-entropy threshold search)

Description

Estimates the complexity of a (windowed) time series using the entropy of recurrence microstates (Corso et al. 2018), searching over the vicinity threshold `epsilon` to find the value that *maximises* the Shannon entropy of the microstate distribution — a parameter-free alternative to fixing `epsilon` by hand, related to the optimal-threshold method of Prado et al. (2018).

Usage

```
recurrence_microstate_entropy(
  x,
  block = 3L,
  n_samples = 18000L,
  eps_min = 1e-09,
  eps_max = 0.499999999,
  frac = 10L,
  frac2 = 10L,
  seed = NULL
)
```

Arguments

`x`	Numeric vector. The (already windowed) time series segment to analyse. In the original study, `x` was min-max normalised to `[0, 1]` per window before calling this function — do the same if you want comparable `eps_max` values across windows/subjects.

`block` Integer. Side length of the square recurrence microstate block; the microstate space has `2^(block^2)` possible patterns. Default `3` (matching the reference implementation).

`n_samples` Integer. Number of random point-pairs sampled to estimate the recurrence microstate distribution. Default `18000`.

`eps_min`, `eps_max` Numeric. Search range for the vicinity threshold. Defaults match the reference implementation.

`frac`, `frac2` Integer. Number of steps in the coarse search pass and number of refinement iterations, respectively. Defaults match the reference implementation.

`seed` Optional integer. Seeds the random pair sampling for reproducibility. The original Julia script reseeds from system entropy on every run (`Random.seed!()` with no argument) and is therefore not reproducible run-to-run by design; pass a `seed` here if you need reproducible results, or leave `NULL` to match the original's fresh-randomness-every-call behaviour.

Details

Ported from `circadia-bio/maxEntropy` (`Entropy.jl`, MIT licensed), the script used to produce the entropy estimates in Ferre et al. (2024, PLOS ONE). The deterministic core loop was validated against a Python transliteration of the original on synthetic test data (exact match on `S_max` and `eps_max` given identical sampled index pairs). Only the reusable `Max_Entropy()` routine was ported — the original script's `main()` (file I/O over named subject directories) is study-specific scaffolding, not part of the general-purpose method.

Value

A list with:

`microstate_probs` Probability of each of the `2^(block^2)` microstates at the entropy-maximising threshold.

`entropy_max` The maximum Shannon entropy found (`S_max`).

`eps_max` The vicinity threshold at which `entropy_max` was achieved.

References

Corso G, Prado TL, dos Santos Lima GZ, Kurths J, Lopes SR. Quantifying entropy using recurrence matrix microstates. Chaos 2018;28(8):083108.

Prado TL, dos Santos Lima GZ, Lobao-Soares B, do Nascimento GC, Corso G, Fontenele-Araujo J, Kurths J, Lopes SR. Optimizing the detection of nonstationary signals by using recurrence analysis. Chaos 2018;28(8):085703.

Ferre IBS, Corso G, dos Santos Lima GZ, Lopes SR, Leocadio-Miguel MA, Franca LGS, de Lima Prado T, Araujo JF. Cycling reduces the entropy of neuronal activity in the human adult cortex. PLOS ONE 2024;19(10):e0298703.

Examples

```
set.seed(1)
x <- (sin(seq(0, 20 * pi, length.out = 300)) + 1) / 2
recurrence_microstate_entropy(x, seed = 1)
```

`rtractor_palette` *Retrieve an Rtractor palette*

Description

Returns a named character vector of hex colour codes for the requested palette. Suitable for direct use in `ggplot2::scale_fill_manual()` or `ggplot2::scale_colour_manual()`.

Usage

```
rtractor_palette(palette = "main", n = NULL, reverse = FALSE)
```

Arguments

`palette`	Name of the palette. One of `"main"`, `"core"`, `"diverging"`, `"seq_blue"`, `"seq_sage"`, `"seq_coral"`. Defaults to `"main"`.
`n`	Number of colours to return. If `NULL` (default) all colours in the palette are returned. If `n` is smaller than the palette length the first `n` colours are returned; if larger an error is thrown.
`reverse`	Logical. Reverse the order of the palette? Default `FALSE`.

Value

A named (or unnamed for gradient palettes) character vector of hex colour codes.

Examples

```
rtractor_palette()
rtractor_palette("core")
rtractor_palette("seq_blue", n = 3)
rtractor_palette("diverging", reverse = TRUE)
```

rtractor_palettes *List all available Rtractor palettes*

Description

Prints the names and sizes of all palettes defined in the package.

Usage

```
rtractor_palettes()
```

Value

A named integer vector of palette lengths, invisibly.

Examples

```
rtractor_palettes()
```

sample_entropy *Sample Entropy (SampEn)*

Description

Estimates the complexity of a time series using sample entropy (Richman & Moorman 2000): the negative log ratio of template matches of length `m + 1` to template matches of length `m`, using a fixed Chebyshev-distance tolerance. Direct C++ port of the counting core in PhysioNet's reference `mse.c` (Costa), validated to reproduce the compiled reference binary's output exactly (to its own displayed precision) on synthetic test data. See `inst/COPYRIGHTS`.

Usage

```
sample_entropy(x, m = 2L, r = 0.15)
```

Arguments

`x`	Numeric vector. The time series to analyse.
`m`	Integer >= 1. Template length. Default `2` (the standard MSE convention, per Costa et al.).
`r`	Numeric > 0. Tolerance, as a fraction of `x`'s own standard deviation (i.e. the actual Chebyshev-distance tolerance used is `sd(x) * r`). Default `0.15` (the standard MSE convention).

Details

This is also the building block `multiscale_entropy()` applies at each coarse-grained scale – see `R/multiscale.R`.

Value

A length-1 numeric: the sample entropy. If there are too few valid template pairs to compare (`length(x) - m < 2`), returns `NA`. If there are zero matches at either template length (undefined ratio), returns the conventional fallback `-log(1 / (N*(N-1)))` used by the reference implementation, where `N = length(x) - m`.

References

Richman JS, Moorman JR. Physiological time-series analysis using approximate entropy and sample entropy. Am J Physiol Heart Circ Physiol 2000;278(6):H2039-H2049.

Examples

```
set.seed(1)
sample_entropy(rnorm(1000))
```

`scale_colour_rtractor` *Rtractor discrete colour scale*

Description

Applies an Rtractor qualitative palette to the `colour` aesthetic. Defaults to `"core"` (excludes the dark `ink` colour, which is reserved for text/axis annotation rather than data encoding).

Usage

```
scale_colour_rtractor(palette = "core", reverse = FALSE, ...)

scale_color_rtractor(palette = "core", reverse = FALSE, ...)
```

Arguments

`palette`	Palette name. Default `"core"`.
`reverse`	Logical. Reverse the palette? Default `FALSE`.
`...`	Additional arguments passed to `ggplot2::discrete_scale()`.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
  geom_point(size = 3) +
  scale_colour_rtractor() +
  theme_rtractor()
```

`scale_colour_rtractor_c` *Rtractor continuous colour scale*

Description

Interpolates across an Rtractor palette for continuous `colour` data. The `"diverging"` palette is recommended for signed quantities (e.g. Lyapunov exponent sign, spectrum asymmetry); the `seq_*` palettes for unipolar data (e.g. entropy magnitude).

Usage

```
scale_colour_rtractor_c(palette = "seq_blue", reverse = FALSE, ...)

scale_color_rtractor_c(palette = "seq_blue", reverse = FALSE, ...)
```

Arguments

`palette`	Palette name. Default `"seq_blue"`.
`reverse`	Logical. Reverse the palette? Default `FALSE`.
`...`	Additional arguments passed to `ggplot2::scale_colour_gradientn()`.

`scale_fill_rtractor` *Rtractor discrete fill scale*

Description

Applies an Rtractor qualitative palette to the `fill` aesthetic.

Usage

```
scale_fill_rtractor(palette = "core", reverse = FALSE, ...)
```

Arguments

`palette`	Palette name. Default `"core"`.
`reverse`	Logical. Reverse the palette? Default `FALSE`.
`...`	Additional arguments passed to `ggplot2::discrete_scale()`.

Examples

```
library(ggplot2)
ggplot(mpg, aes(class, fill = drv)) +
  geom_bar() +
  scale_fill_rtractor() +
  theme_rtractor()
```

`scale_fill_rtractor_c` *Rtractor continuous fill scale*

Description

Interpolates across an Rtractor palette for continuous `fill` data.

Usage

```
scale_fill_rtractor_c(palette = "seq_blue", reverse = FALSE, ...)
```

Arguments

`palette`	Palette name. Default `"seq_blue"`.
`reverse`	Logical. Reverse the palette? Default `FALSE`.
`...`	Additional arguments passed to `ggplot2::scale_colour_gradientn()`.

`theme_rtractor` *Rtractor ggplot2 theme*

Description

A clean ggplot2 theme built on `ggplot2::theme_minimal()`, visually distinct from `theme_circadia()` (softer palette, same structural conventions) so that Rtractor figures are recognisable at a glance.

Usage

```
theme_rtractor(
  base_size = 14,
  base_family = "",
  grid = "none",
  legend_position = "right"
)
```

Arguments

base_size	Base font size in points. Default 14.
base_family	Base font family. Default "" (ggplot2 default).
grid	Which grid lines to show. One of "none" (default), "y" (horizontal only), "xy" (both), "x" (vertical only).
legend_position	Position of the legend passed to ggplot2::theme(). Default "right".

Value

A ggplot2::theme() object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
  geom_point(size = 2) +
  scale_colour_rtractor() +
  theme_rtractor()
```

Index