Package: dynR (via r-universe)

July 2, 2026

Title Dynamic Connectivity Analysis for Neurophysiological Timeseries

Version 0.1.2

Description An R port of the Python dynfc library for computing dynamic connectivity (dynFC) representations from multivariate neurophysiological timeseries, including BOLD fMRI, EEG, LFP, and related signals. Implements sliding-window Pearson correlation (Hansen et al., 2015), edge-centric cofluctuation analysis (Esfahlani et al., 2020; Faskowitz et al., 2020), instantaneous phase-locking via the Hilbert transform, dynamic phase-locking matrices (dPL), the LEiDA leading-eigenvector framework (Cabral et al., 2017; Lord et al., 2019), and the Kuramoto order parameter with metastability and Shannon entropy. Part of the Circadia Lab R ecosystem.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 4.1.0)

Imports gsignal (>= 0.3.5), dplyr (>= 1.1.0), tidyr (>= 1.3.0), rlang (>= 1.1.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, pkgdown, withr, ggplot2 (>= 3.4.0), reticulate

Config/testthat/edition 3

Lifecycle experimental

URL https://dynr.circadia-lab.uk, https://github.com/circadia-bio/dynR

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

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-07-02 21:19:47 UTC

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

RemoteRef v0.1.2

RemoteSha c08c64a0b24a4689dbd92e6cf0bfc2697480a602

Contents

`bandpass_filter` *Butterworth bandpass filter*

Description

Design and apply a zero-phase Butterworth bandpass filter to a signal, using scipy-compatible steady-state initial conditions (equivalent to `scipy.signal.sosfiltfilt` / `scipy.signal.filtfilt` with `padtype = "odd"`).

Usage

```
bandpass_filter(x, flp, fhi, delt, order = 2)
```

Arguments

`x`	Numeric vector. Signal to be filtered.
`flp`	Numeric. Low-pass cutoff frequency (Hz).
`fhi`	Numeric. High-pass cutoff frequency (Hz).
`delt`	Numeric. Sampling interval in seconds (i.e. TR for fMRI).
`order`	Integer. Filter order. Default is 2.

Details

`gsignal::filtfilt()` uses zero initial conditions, which produces edge transients up to ~0.24 signal units on real fMRI data. This implementation replicates scipy's `lfilter_zi` approach: both the forward and backward passes are initialised at steady state scaled by the first sample of each pass, reducing edge error to machine precision.

Value

Numeric vector. Zero-phase filtered signal, same length as `x`.

Examples

```
set.seed(1)
x <- rnorm(200)
x_filt <- bandpass_filter(x, flp = 0.01, fhi = 0.1, delt = 2, order = 2)
```

---

cofluct *Edge-centric cofluctuation analysis*

Description

Compute edge time series and root-sum-square (RSS) cofluctuations from z-standardised BOLD signal. Implements the edge-centric FC framework of Esfahlani et al. (2020) and Faskowitz et al. (2020).

Usage

```
cofluct(timeseries, k = 1)
```

Arguments

`timeseries`	Numeric matrix [N × Tmax]. BOLD signal with N parcels as rows and Tmax timepoints as columns.
`k`	Integer. Upper-triangle offset. `k = 1` (default) excludes the diagonal; `k = 0` includes it.

Value

A list with:

`edge_ts`	Numeric matrix [n_edges × Tmax]. Edge time series, one row per unique parcel pair.
`rss`	Numeric vector [Tmax]. Root-sum-square cofluctuation at each timepoint.

References

Esfahlani, F. Z. et al. (2020). High-amplitude cofluctuations in cortical activity drive functional connectivity. *PNAS*, 117(45), 28393–28401. doi:10.1073/pnas.2005531117

Faskowitz, J. et al. (2020). Edge-centric functional network representations of human cerebral cortex reveal overlapping system-level architecture. *Nature Neuroscience*, 23(12), 1644–1654. doi:10.1038/s4159302000719y

Examples

```
set.seed(1)
ts <- matrix(rnorm(10 * 200), nrow = 10, ncol = 200)
res <- cofluct(ts)
dim(res$edge_ts)  # n_edges x 200
```

`corr_corr` *Correlation of correlations matrix*

Description

Compute the correlation between edge time series across all timepoints, producing a [Tmax × Tmax] "correlation of correlations" matrix (Hansen et al., 2015).

Usage

```
corr_corr(timeseries, k = 1)
```

Arguments

`timeseries`	Numeric matrix [N × Tmax]. BOLD signal.
`k`	Integer. Upper-triangle offset passed to `cofluct()`. Default 1.

Value

Numeric matrix [Tmax × Tmax].

References

Hansen, E. C. A. et al. (2015). Functional connectivity dynamics: Modeling the switching behavior of the resting state. *NeuroImage*, 105, 525–535. doi:10.1016/j.neuroimage.2014.11.001

Examples

```
set.seed(1)
ts <- matrix(rnorm(10 * 100), nrow = 10, ncol = 100)
cc <- corr_corr(ts)
dim(cc)  # 100 x 100
```

`corr_slide` *Sliding window correlation*

Description

Compute functional connectivity matrices over sliding windows of a BOLD timeseries (Hansen et al., 2015).

Usage

```
corr_slide(timeseries, window, step = NULL)
```

Arguments

`timeseries`	Numeric matrix [N × Tmax]. BOLD signal with N parcels as rows and Tmax timepoints as columns.
`window`	Integer. Window size in timepoints.
`step`	Integer. Step between window onsets. Defaults to `window` (non-overlapping windows).

Value

A list with:

`corr_mats`	Array [N, N, n_windows] of Pearson correlation matrices.
`idx`	Integer vector of 1-indexed window onset positions.

References

Hansen, E. C. A. et al. (2015). Functional connectivity dynamics: Modeling the switching behavior of the resting state. *NeuroImage*, 105, 525–535. doi:10.1016/j.neuroimage.2014.11.001

Examples

```
set.seed(1)
ts <- matrix(rnorm(10 * 200), nrow = 10, ncol = 200)
res <- corr_slide(ts, window = 20)
dim(res$corr_mats)  # 10 x 10 x 10
```

do_euclid *Euclidean distance between consecutive points*

Description

Compute the Euclidean distance between each consecutive pair of rows in a matrix — typically a trajectory through PC space.

Usage

```
do_euclid(x)
```

Arguments

x	Numeric matrix [n_points × n_dims].

Value

Numeric vector of length `nrow(x)`. The first element is always 0 (no previous point).

Examples

```
set.seed(1)
pcs <- matrix(rnorm(50 * 3), nrow = 50, ncol = 3)
d <- do_euclid(pcs)
length(d)  # 50
```

dyn_phase_lock *Dynamic phase-locking matrix (dPL)*

Description

Compute instantaneous phase-locking matrices from parcel-level phase time series and extract leading eigenvectors via `get_leida()`. The first and last 10 timepoints are discarded to avoid edge effects from the Hilbert transform.

Usage

```
dyn_phase_lock(phases)
```

Arguments

phases	Numeric matrix [N × Tmax]. Instantaneous phases in radians, as returned by `hilbert_phases()`.

Value

A list with:

`sync_conn`	Array [N, N, Tmax-20]. Instantaneous phase-locking matrices, one per (trimmed) timepoint.
`leida`	Matrix [Tmax-20, N]. Leading eigenvectors from `get_leida()`.

References

Cabral, J. et al. (2017). Cognitive performance in healthy older adults relates to spontaneous switching between states of functional connectivity during rest. *Scientific Reports*, 7(1), 5135. doi:10.1038/s41598017054257

Lord, L.-D. et al. (2019). Dynamical exploration of the repertoire of brain networks at rest is modulated by psilocybin. *NeuroImage*, 199, 127–142. doi:10.1016/j.neuroimage.2019.05.060

Examples

```
set.seed(1)
ts <- matrix(rnorm(10 * 200), nrow = 10, ncol = 200)
phases <- hilbert_phases(ts)
res <- dyn_phase_lock(phases)
dim(res$sync_conn)  # 10 x 10 x 180
dim(res$leida)      # 180 x 10
```

---

dyn_transitions *State transition probabilities (Markov analysis)*

Description

From a long-format data frame of cluster labels ordered in time, compute first-order Markov transition probabilities between brain states: for each source state, the fraction of transitions that lead to each target state.

Usage

```
dyn_transitions(tbl, vars, cVar, sortBy, groupBy, remIntra = FALSE)
```

Arguments

`tbl`	A data frame in long format (one row per timepoint).
`vars`	Character vector of grouping / covariate column names to preserve in the output (e.g. `c("sub", "ses", "age", "sex")`).
`cVar`	Character. Name of the column holding integer cluster labels.
`sortBy`	Character vector of column names used to sort rows within each group before transitions are computed (typically `c("sub", "ses", "ttime")`).

groupBy	Character vector of column names that define independent sequences (typically c("sub", "ses")). Transitions are never computed across group boundaries.
remIntra	Logical. If TRUE, self-transitions (state → same state) are removed before normalising probabilities. Default FALSE.

Details

Ported from clusters_markov() in the neonatal_dfc analysis pipeline (França et al., *Nat Commun*).

Value

A nested tibble with columns tag, source, target, and data. tag encodes the transition as "<source>_<target>". Each data element is a per-group tibble with columns inherited from vars plus:

n	Raw transition count.
tot	Total transitions out of source for that group.
nCount	Transition probability (n / tot).

Examples

```
set.seed(1)
df <- data.frame(
  sub   = rep(c("A", "B"), each = 50),
  ses   = 1L,
  ttime = rep(seq_len(50), 2),
  clus4 = sample(1:4, 100, replace = TRUE)
)
tr <- dyn_transitions(
  df,
  vars    = c("sub", "ses"),
  cVar    = "clus4",
  sortBy  = c("sub", "ses", "ttime"),
  groupBy = c("sub", "ses")
)
tr
```

---

fc *Functional connectivity matrix (200 parcels)*

Description

Static functional connectivity matrix derived from ts by computing the full-timeseries Pearson correlation across all 200 parcel pairs. Serves as a ground-truth reference for validating sliding-window and phase-based dynFC methods when the window spans the full timeseries.

Usage

```
fc
```

Format

A numeric matrix with 200 rows and 200 columns. Diagonal is 1; off-diagonal values are Pearson correlations in [-1, 1].

Source

Derived from ts.

`get_leida` *Leading eigenvector decomposition (LEiDA)*

Description

Extract the leading eigenvector from a series of instantaneous phase-locking matrices. The sign of each eigenvector is normalised so that its sum is non-positive, following the LEiDA convention.

Usage

```
get_leida(sync_conn)
```

Arguments

`sync_conn`	Numeric array [N, N, Tmax]. Instantaneous phase-locking matrices, as returned by `dyn_phase_lock()`.

Value

Numeric matrix [Tmax × N]. Leading eigenvectors, one per timepoint.

References

Cabral, J. et al. (2017). Cognitive performance in healthy older adults relates to spontaneous switching between states of functional connectivity during rest. *Scientific Reports*, 7(1), 5135. doi:10.1038/s41598017054257

Lord, L.-D. et al. (2019). Dynamical exploration of the repertoire of brain networks at rest is modulated by psilocybin. *NeuroImage*, 199, 127–142. doi:10.1016/j.neuroimage.2019.05.060

hilbert_phases *Hilbert transform phase extraction*

Description

Compute the instantaneous phase time series for all parcels/voxels via the analytic signal (Hilbert transform). Each parcel timeseries is demeaned before transformation.

Usage

```
hilbert_phases(timeseries)
```

Arguments

`timeseries`	Numeric matrix [N × Tmax]. BOLD signal with N parcels as rows and Tmax timepoints as columns.

Value

Numeric matrix [N × Tmax]. Instantaneous phases in radians.

References

Cabral, J. et al. (2017). Cognitive performance in healthy older adults relates to spontaneous switching between states of functional connectivity during rest. *Scientific Reports*, 7(1), 5135. doi:10.1038/s41598017054257

Examples

```
set.seed(1)
ts <- matrix(rnorm(10 * 200), nrow = 10, ncol = 200)
phases <- hilbert_phases(ts)
dim(phases)  # 10 x 200
```

kuramoto *Kuramoto order parameter and metastability*

Description

Compute the global Kuramoto order parameter time series, the metastability index (standard deviation of the order parameter), and Shannon entropy of synchrony from parcel-level instantaneous phase time series. The first and last 10 timepoints are discarded (matching `dyn_phase_lock()`).

Usage

```
kuramoto(phases, base = 2, n_bits = 8L)
```

Arguments

phases	Numeric matrix [N × Tmax]. Instantaneous phases in radians, as returned by hilbert_phases().
base	Numeric. Logarithm base for Shannon entropy. Default is 2.
n_bits	Integer or NULL. Bit depth for discretising the synchrony series before entropy estimation. Default is 8.

Value

A list with:

metastability	Numeric scalar. Standard deviation of the Kuramoto order parameter.
synchrony	Numeric vector [Tmax-20]. Kuramoto order parameter at each (trimmed) time-point.
entropy	Numeric scalar. Shannon entropy of the synchrony series.

Examples

```
set.seed(1)
ts <- matrix(rnorm(10 * 200), nrow = 10, ncol = 200)
phases <- hilbert_phases(ts)
res <- kuramoto(phases)
res$metastability
```

---

shannon_entropy *Shannon entropy*

Description

Estimate Shannon entropy from a numeric vector, with optional bit-depth discretisation (as used internally by kuramoto()).

Usage

```
shannon_entropy(x, base = 2, n_bits = NULL)
```

Arguments

x	Numeric vector.
base	Numeric. Logarithm base. Default is 2 (entropy in bits).
n_bits	Integer or NULL. If supplied, values are scaled by 2^{n_bits} and rounded before computing entropy. Default is NULL.

Value

Numeric scalar. Shannon entropy.

Examples

```
x <- sample(1:4, 100, replace = TRUE)
shannon_entropy(x)
```

`ts` *BOLD fMRI timeseries (200 parcels, 600 timepoints)*

Description

A resting-state BOLD fMRI timeseries from the edge-ts repository, parcellated into 200 regions of interest. Used in dynR vignettes and examples to demonstrate dynamic FC methods on realistic data.

Usage

```
ts
```

Format

A numeric matrix with 200 rows (parcels) and 600 columns (timepoints).

Source

https://github.com/brain-networks/edge-ts

Index