

Package: PhysioNIRS (via r-universe)

October 3, 2026

Title Governed Near-Infrared Spectroscopy Data and SNIRF Input/Output

Version 0.6.1

Description Provides strict input, output, measurement identity, probe geometry, optical-density, modified Beer-Lambert, motion-correction, and short-separation nuisance, signal-quality, channel-pruning, and live neurofeedback contracts for continuous-wave functional near-infrared spectroscopy data stored in 'PhysioExperiment' objects. Shared Near Infrared Spectroscopy Format ('SNIRF') files are read and written through structured 'HDF5' APIs while preserving time bases, measurement order, probe coordinates, metadata tags, stimulus tables, and processing provenance.

Depends R (>= 4.2), PhysioExperiment (>= 2.0.0)

Imports digest, methods, PhysioStream (>= 0.9.0), rhdf5, S4Vectors, signal, SummarizedExperiment, stats

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

License MIT + file LICENSE

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/testthat/edition 3

Collate 'PhysioNIRS-package.R' 'examples-data.R' 'snirf-contract.R' 'io-snirf.R' 'probe.R' 'optical-density.R' 'dpf.R' 'extinction.R' 'mbl.R' 'motion-contract.R' 'motion-detect.R' 'motion-tddr.R' 'motion-wavelet.R' 'motion-spline.R' 'short-separation-contract.R' 'short-separation.R' 'physiology-band.R' 'nuisance-design.R' 'quality-contract.R' 'quality.R' 'neurofeedback.R' 'activation-glm.R' 'zzz.R'

URL <https://github.com/x-biosignal/PhysioNIRS>,
<https://x-biosignal.r-universe.dev/PhysioNIRS>,
<https://x-biosignal.github.io/PhysioNIRS/>

BugReports <https://github.com/x-biosignal/PhysioNIRS/issues>

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs libxml2-dev libssl-dev zlib1g-dev

Repository <https://x-biosignal.r-universe.dev>

Date/Publication 2026-10-03 13:11:43 UTC

RemoteUrl <https://github.com/x-biosignal/PhysioNIRS>

RemoteRef HEAD

RemoteSha f9c150833ba0c3456101fdeff60533f3359391ec

Contents

PhysioNIRS-package	3
dpf	3
extinctionCoefficients	4
identifyShortChannels	5
intensityToOD	6
mbll	7
measurementList	8
motionArtifactDetect	8
nirsActivationContrast	9
nirsActivationGLM	10
nirsNeurofeedback	11
nirsNeurofeedback-lifecycle	13
physiologyBandpass	14
print.nirs_quality	15
print.NIRSNeurofeedback	15
pruneChannels	16
readSNIRF	17
scalpCouplingIndex	18
shortSeparationDesign	19
shortSeparationRegress	20
signalQualityIndex	21
sourceDetectorDistances	22
splineMotionCorrect	23
tddr	24
waveletMotionCorrect	25
writeSNIRF	26
Index	27

Description

PhysioNIRS maps continuous-wave SNIRF data into the `PhysioExperiment::PhysioExperiment()` contract without changing measurement order, probe geometry, stimulus tables, or the exact time base. Governed optical-density, modified Beer-Lambert, motion-correction, short-separation nuisance, quality/pruning, and live-neurofeedback operations retain their log, distance, pathlength, geometry, filter, identity, and reference conventions.

Details

SNIRF metadata may contain identifying subject and acquisition fields. Callers are responsible for de-identification before sharing a file.

Author(s)

Maintainer: Yusuke Matsui <mail.to.matsui@gmail.com>

References

Shared Near Infrared Spectroscopy Format specification: <https://fnirs.github.io/snirf/>

See Also

Useful links:

- <https://github.com/x-biosignal/PhysioNIRS>
- <https://x-biosignal.r-universe.dev/PhysioNIRS>
- <https://x-biosignal.github.io/PhysioNIRS/>
- Report bugs at <https://github.com/x-biosignal/PhysioNIRS/issues>

Description

Implements published population equations. These values are population estimates, not subject-specific pathlength measurements.

Usage

```
dpf(
  wavelength_nm,
  age_years,
  model = c("scholkmann2013", "duncan1996"),
  extrapolate = FALSE
)
```

Arguments

wavelength_nm	Positive wavelength values in nanometres.
age_years	Non-negative age values in years.
model	Exactly "scholkmann2013" or "duncan1996".
extrapolate	Whether to allow age/wavelength extrapolation where the selected model defines an unambiguous equation.

Value

A positive numeric vector with model and extrapolation attributes.

Examples

```
# Age-dependent differential pathlength factors for a 30-year-old adult
dpf(c(760, 850), age_years = 30)
```

extinctionCoefficients

Get governed haemoglobin extinction coefficients

Description

Coefficients are linearly interpolated from the package-owned table. Stored coefficients use the base-10 molar convention in $\text{cm}^{-1} \text{ M}^{-1}$; natural-log coefficients in $\text{m}^{-1} \text{ M}^{-1}$ include the explicit $100 * \log(10)$ conversion.

Usage

```
extinctionCoefficients(
  wavelength_nm,
  unit = c("m-1 M-1", "cm-1 M-1"),
  extrapolate = FALSE
)
```

Arguments

wavelength_nm Positive wavelengths in nanometres.
 unit Exactly "m-1 M-1" or "cm-1 M-1".
 extrapolate Whether to permit terminal-slope linear extrapolation.

Value

A numeric matrix with exact columns HbO and HbR.

Examples

```
# Molar extinction (m^-1 M^-1) at two typical fNIRS wavelengths
extinctionCoefficients(c(760, 850))
# The same coefficients expressed per centimetre
extinctionCoefficients(c(690, 830), unit = "cm-1 M-1")
```

identifyShortChannels *Identify short source-detector channels*

Description

Calculates governed source-detector distances and midpoints in metres. Channels strictly below threshold_m are classified as short. A returned map is fingerprint-bound to the complete source object.

Usage

```
identifyShortChannels(x, threshold_m = 0.01)
```

Arguments

x A governed measurement-level or MBLL pair-level PhysioExperiment.
 threshold_m Positive short-separation threshold in metres.

Value

A source-bound nirs_short_channels data frame.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
identifyShortChannels(od, threshold_m = 0.015)
```

intensityToOD

Convert continuous-wave fNIRS intensity to optical density

Description

Optical density is calculated with the natural-log convention $-\log(I / I_0)$. The reference intensity and any replacement of non-positive cells are explicit and recorded in provenance.

Usage

```
intensityToOD(
  x,
  assay_name = NULL,
  baseline = c("mean", "median"),
  baseline_interval = NULL,
  reference = NULL,
  nonpositive = c("error", "replace"),
  replacement = NULL,
  output_assay = "OD"
)
```

Arguments

x	A governed continuous-wave PhysioExperiment.
assay_name	Exact input assay name. NULL is allowed only when one eligible intensity assay exists.
baseline	Exactly "mean" or "median".
baseline_interval	Optional closed interval in seconds.
reference	Optional positive scalar or vector with one value per measurement.
nonpositive	Exactly "error" or "replace".
replacement	Explicit positive scalar used only with nonpositive = "replace".
output_assay	Exact name for the new optical-density assay.

Value

x with a unitless optical-density assay appended.

Examples

```
pe <- PhysioNIRS:::nirs_demo_object()
od <- intensityToOD(pe)
SummarizedExperiment::assayNames(od)
```

mbll

*Convert optical density to haemoglobin concentration change***Description**

Solves the modified Beer-Lambert system independently for each exact source-detector pair. The returned `PhysioExperiment` has one column per pair and assays HbO, HbR, and HbT in micromolar.

Usage

```
mbll(
  x,
  assay_name = "OD",
  pathlength_factor = 6,
  age_years = NULL,
  dpf_model = "scholkmann2013",
  distance_m = NULL,
  output_unit = "uM"
)
```

Arguments

<code>x</code>	A governed <code>PhysioExperiment</code> carrying natural-log optical density.
<code>assay_name</code>	Exact optical-density assay name.
<code>pathlength_factor</code>	Positive scalar/vector, or <code>NULL</code> to derive DPF from <code>age_years</code> .
<code>age_years</code>	Age used only when <code>pathlength_factor</code> = <code>NULL</code> .
<code>dpf_model</code>	Exact DPF model name passed to <code>dpf()</code> .
<code>distance_m</code>	Optional positive scalar, per-measurement vector, or per-pair vector overriding probe geometry.
<code>output_unit</code>	Exactly "uM".

Value

A pair-collapsed `PhysioExperiment` with concentration assays.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
hb <- mbll(od)
SummarizedExperiment::assayNames(hb)
```

measurementList	<i>Return the governed SNIRF measurement list</i>
-----------------	---

Description

Return the governed SNIRF measurement list

Usage

```
measurementList(x)
```

Arguments

x	A <code>PhysioExperiment</code> imported from SNIRF or carrying the same governed metadata contract.
---	--

Value

A defensive copy of the normalized measurement `DataFrame`.

Examples

```
pe <- PhysioNIRS:::nirs_demo_object()
measurementList(pe)[, c("channel_label", "wavelength_nm")]
```

motionArtifactDetect	<i>Detect fNIRS motion artifacts by channel</i>
----------------------	---

Description

Uses strict amplitude and derivative-SD thresholds over a finite lag window. Detections are expanded in time and can be unioned across wavelengths from the same source-detector pair.

Usage

```
motionArtifactDetect(
  x,
  assay_name = "OD",
  t_motion = 0.5,
  t_mask = 1,
  sd_threshold = 20,
  amplitude_threshold = 0.5,
  group_wavelengths = TRUE
)
```

Arguments

x	A governed PhysioExperiment.
assay_name	Exact source assay name.
t_motion	Positive lag window in seconds.
t_mask	Non-negative expansion on each side in seconds.
sd_threshold	Positive multiple of channel derivative SD.
amplitude_threshold	Positive absolute-change threshold.
group_wavelengths	Whether to union measurements sharing an exact source-detector pair.

Value

A typed nirs_motion_mask aligned to assay rows and columns.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
mask <- motionArtifactDetect(od)
class(mask)
```

nirsActivationContrast

Contrast of an fNIRS activation GLM

Description

Applies a linear contrast over the condition betas of a [nirsActivationGLM\(\)](#) fit, per channel: the estimate $c' \beta$, its standard error $\sqrt{\sigma^2 * c' (X'X)^{-1} c}$, t-statistic, degrees of freedom and p-value.

Usage

```
nirsActivationContrast(fit, contrast)
```

Arguments

fit	A nirs_activation_glm.
contrast	A numeric contrast over conditions: named by condition, or an unnamed vector with one weight per condition (e.g. <code>c(task = 1, rest = -1)</code>).

Value

A per-channel data.frame: channel, estimate, se, t, df, p.

See Also

[nirsActivationGLM\(\)](#)

Examples

```
hb <- mbll(intensityToOD(PhysioNIRS:::nirs_demo_object()))
fit <- nirsActivationGLM(hb, assay_name = "HbO")
nirsActivationContrast(fit, c(task = 1))
```

nirsActivationGLM	<i>First-level fNIRS cortical-activation GLM</i>
-------------------	--

Description

Fits a general linear model of every haemoglobin channel against HRF-convolved task regressors: for each condition, a boxcar over its stimulus blocks convolved with a canonical double-gamma HRF, plus a discrete-cosine drift basis and optional short-separation nuisance regressors. Returns the per-channel activation amplitude (beta), standard error and t-statistic for each condition. Run it separately on the HbO and HbR assays.

Usage

```
nirsActivationGLM(
  x,
  events = NULL,
  assay_name = "HbO",
  conditions = NULL,
  basis = c("canonical", "derivative"),
  drift = 6,
  nuisance = NULL,
  hrf = list(peak = 6, undershoot = 16, ratio = 6),
  prewhiten = c("none", "ar1")
)
```

Arguments

x	A <code>PhysioExperiment</code> of haemoglobin concentration (e.g. the HbO assay from mbll()).
events	A <code>data.frame</code> with numeric onset and duration (seconds) and a condition column; if <code>NULL</code> , the object's events (PhysioExperiment::getEvents onset/duration/value) are used, with value as the condition.
assay_name	Assay to model (default "HbO").
conditions	Optional character vector selecting/ordering conditions.
basis	"canonical" (HRF only) or "derivative" (HRF + temporal derivative per condition).
drift	Number of discrete-cosine drift terms (default 6).

nuisance	Optional numeric matrix of extra nuisance regressors (e.g. <code>shortSeparationDesign()</code> output).
hrf	List of HRF parameters (peak, undershoot, ratio).
prewhiten	"none" or "ar1" (per-channel AR(1) prewhitening; strongly recommended for fNIRS to avoid inflated t-statistics).

Value

An S3 `nirs_activation_glm` with coefficients (a channel x condition data.frame of beta, se, t, df, p), the beta/se matrices, per-channel sigma2/rho, and the design (for `nirsActivationContrast()`).

References

Friston KJ et al. (1994) Statistical parametric maps. *Hum Brain Mapp* 2:189-210. Huppert TJ et al. (2009) HomER. *Appl Opt* 48:D280.

See Also

`nirsActivationContrast()`, `shortSeparationDesign()`, `mbll()`

Examples

```
hb <- mbll(intensityToOD(PhysioNIRS:::nirs_demo_object()))
fit <- nirsActivationGLM(hb, assay_name = "HbO")
fit$coefficients
```

nirsNeurofeedback	<i>Construct a governed live fNIRS neurofeedback controller</i>
-------------------	---

Description

The controller builds a public `PhysioStream` pipeline and `BiofeedbackScope`. It computes frozen-baseline regional HbO contrasts in a committed pipeline operation and publishes each eligible multi-target value through one atomic `biofeedbackUpdate()` call. Construction is side-effect-free.

Usage

```
nirsNeurofeedback(
  stream,
  regions,
  contrast,
  assay_name = "HbO",
  baseline_seconds = 20,
  update_seconds = 0.1,
  smoothing_seconds = 2
)
```

Arguments

stream	A created or open regular-rate numeric StreamSource.
regions	A named list of non-overlapping exact HbO channel IDs.
contrast	A region-named numeric vector or region-by-target matrix.
assay_name	Exact live assay name, currently "HbO".
baseline_seconds	Positive frozen-baseline duration.
update_seconds	Positive scheduled update interval.
smoothing_seconds	Positive trailing time-weighted mean duration.

Details

The stream must declare metadata\$nirs fields assay_name, assay_kind = "haemoglobin_concentration", identity_kind = "pair", channel_id, source_index, and detector_index; ordered channel names and units must be the exact HbO identities and "uM".

Value

A side-effect-free NIRSNeurofeedback runtime.

References

Kober et al. (2017), DOI: 10.3389/fnhum.2017.00081.

Examples

```
# Offline demonstration on a deterministic loopback stream. In practice
# `stream` is a live regular-rate HbO inlet (e.g. an LSL inlet from a device).
info <- PhysioStream::streamInfo(
  "nirs-demo", type = "NIRS", channel_names = c("S1_D1", "S2_D2"),
  nominal_srate = 10, dtype = "float64", source_id = "nirs-demo",
  clock_domain = "demo-clock", channel_units = c("uM", "uM"),
  metadata = list(nirs = list(
    assay_name = "HbO", assay_kind = "haemoglobin_concentration",
    identity_kind = "pair", channel_id = c("S1_D1", "S2_D2"),
    source_index = 1:2, detector_index = 1:2)))
stream <- PhysioStream::streamOpen(PhysioStream::loopbackSource(info, 8192L))
controller <- nirsNeurofeedback(
  stream, regions = list(motor = "S1_D1", control = "S2_D2"),
  contrast = c(motor = 1, control = -1),
  baseline_seconds = 1, update_seconds = 0.1, smoothing_seconds = 0.5)
nirsNeurofeedbackStart(controller)
time <- (0:19) / 10
PhysioStream::loopbackFeed(stream, cbind(
  S1_D1 = 1 + 0.2 * sin(2 * pi * 0.5 * time),
  S2_D2 = 1 + 0.1 * cos(time)), time)
while (nirsNeurofeedbackStep(controller, 4L)$updated) {}
nirsNeurofeedbackState(controller)$lifecycle
```

```
nirsNeurofeedbackStop(controller)
```

```
nirsNeurofeedback-lifecycle
```

Control and inspect live NIRS neurofeedback

Description

`nirsNeurofeedbackStep()` processes at most `max_chunks` synchronously, flushing committed target values after each chunk so the external update timestamp remains the latest scope signal time.

Usage

```
nirsNeurofeedbackStart(controller)

nirsNeurofeedbackStep(controller, max_chunks = 16L)

nirsNeurofeedbackState(controller)

nirsNeurofeedbackStop(controller)

nirsNeurofeedbackScope(controller)
```

Arguments

<code>controller</code>	A NIRSNeurofeedback runtime.
<code>max_chunks</code>	Positive exact per-call work bound.

Value

Lifecycle functions return `controller` invisibly. Step and state functions return portable plain lists. `nirsNeurofeedbackScope()` returns the governed `BiofeedbackScope` for a viewer.

Examples

```
# A controller is driven by an open regular-rate HbO stream; here a loopback
# stream stands in for a live device inlet.
info <- PhysioStream::streamInfo(
  "nirs-demo", type = "NIRS", channel_names = c("S1_D1", "S2_D2"),
  nominal_srate = 10, dtype = "float64", source_id = "nirs-demo",
  clock_domain = "demo-clock", channel_units = c("uM", "uM"),
  metadata = list(nirs = list(
    assay_name = "HbO", assay_kind = "haemoglobin_concentration",
    identity_kind = "pair", channel_id = c("S1_D1", "S2_D2"),
    source_index = 1:2, detector_index = 1:2)))
stream <- PhysioStream::streamOpen(PhysioStream::loopbackSource(info, 8192L))
controller <- nirsNeurofeedback(
```

```

    stream, regions = list(motor = "S1_D1", control = "S2_D2"),
    contrast = c(motor = 1, control = -1), baseline_seconds = 1)
nirsNeurofeedbackStart(controller)
time <- (0:19) / 10
PhysioStream::loopbackFeed(stream, cbind(
  S1_D1 = 1 + 0.2 * sin(2 * pi * 0.5 * time), S2_D2 = 1), time)
while (nirsNeurofeedbackStep(controller, 4L)$updated) {}
nirsNeurofeedbackScope(controller)
nirsNeurofeedbackState(controller)$lifecycle
nirsNeurofeedbackStop(controller)

```

physiologyBandpass	<i>Extract an explicit fNIRS physiology frequency band</i>
--------------------	--

Description

Applies a zero-phase Butterworth band-pass independently to every source assay column. Band names identify common nuisance ranges; they are not physiological diagnoses.

Usage

```

physiologyBandpass(
  x,
  assay_name = "OD",
  band = c("mayer", "respiration", "cardiac"),
  range_hz = NULL,
  output_assay = paste0(assay_name, "_", band[[1L]]),
  order = 3L
)

```

Arguments

x	A governed PhysioExperiment.
assay_name	Exact source assay name.
band	Exactly "mayer", "respiration", or "cardiac".
range_hz	Optional explicit positive passband below Nyquist.
output_assay	Exact name for the added filtered assay.
order	Positive integer Butterworth order.

Details

Default passbands in Hz are Mayer 0.07--0.13, respiration 0.15--0.40, and cardiac 0.70--2.00. Filtering uses an order-order Butterworth band-pass through `signal::filtfilt()`. Explicit ranges must remain strictly below the governed Nyquist frequency.

Value

A clone of x with one zero-phase filtered assay.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
# Pass an explicit band; defaults warn that adult ranges may not transfer
filt <- physiologyBandpass(od, band = "cardiac", range_hz = c(0.8, 2))
SummarizedExperiment::assayNames(filt)
```

```
print.nirs_quality      Display a governed NIRS quality result
```

Description

Display a governed NIRS quality result

Usage

```
## S3 method for class 'nirs_quality'
print(x, ...)
```

Arguments

x	A nirs_quality result.
...	Unused.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
print(signalQualityIndex(od))
```

```
print.NIRSNeurofeedback
      Display a governed NIRS neurofeedback runtime
```

Description

Display a governed NIRS neurofeedback runtime

Usage

```
## S3 method for class 'NIRSNeurofeedback'
print(x, ...)
```

Arguments

`x` A NIRSNeurofeedback runtime.
`...` Unused.

Examples

```
info <- PhysioStream::streamInfo(
  "nirs-demo", type = "NIRS", channel_names = c("S1_D1", "S2_D2"),
  nominal_srate = 10, dtype = "float64", source_id = "nirs-demo",
  clock_domain = "demo-clock", channel_units = c("uM", "uM"),
  metadata = list(nirs = list(
    assay_name = "HbO", assay_kind = "haemoglobin_concentration",
    identity_kind = "pair", channel_id = c("S1_D1", "S2_D2"),
    source_index = 1:2, detector_index = 1:2)))
stream <- PhysioStream::streamOpen(PhysioStream::loopbackSource(info, 8192L))
controller <- nirsNeurofeedback(
  stream, regions = list(motor = "S1_D1", control = "S2_D2"),
  contrast = c(motor = 1, control = -1), baseline_seconds = 1)
print(controller)
```

pruneChannels

Mark or drop channels using governed NIRS quality results

Description

Window decisions are collapsed conservatively per metric. Measurement-level optical-density drops propagate to the complete source-detector pair. Inputs and quality results are fingerprint-bound and are never modified.

Usage

```
pruneChannels(x, quality, action = c("mark", "drop"), require_all = TRUE)
```

Arguments

`x` The exact governed source PhysioExperiment.
`quality` One nirs_quality result or a non-empty named list of distinct results.
`action` Exactly "mark" or "drop".
`require_all` Whether all metrics, rather than any metric, must pass.

Value

A cloned marked or consistently subset PhysioExperiment.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
q <- signalQualityIndex(od)
pruneChannels(od, q, action = "mark")
```

readSNIRF

Read a Shared Near Infrared Spectroscopy Format file

Description

Read a Shared Near Infrared Spectroscopy Format file

Usage

```
readSNIRF(path, nirs_index = 1L, data_index = 1L)
```

Arguments

path	Path to one readable .snirf file.
nirs_index	Positive index of the NIRS root.
data_index	Positive index of the data block.

Value

A `PhysioExperiment::PhysioExperiment()` with a time-by-measurement raw assay.

Examples

```
# Round-trip through a temporary SNIRF/HDF5 file (needs a .snirf on disk)
pe <- PhysioNIRS:::nirs_demo_object()
pe <- PhysioExperiment::setEvents(pe, PhysioExperiment::PhysioEvents(
  onset = 6, duration = 4, type = "task", value = "1"))
path <- tempfile(fileext = ".snirf")
writeSNIRF(pe, path)
back <- readSNIRF(path)
dim(back)
unlink(path)
```

scalpCouplingIndex	<i>Calculate the governed fNIRS scalp-coupling index</i>
--------------------	--

Description

Optical-density measurements are grouped by exact source-detector identity, zero-phase filtered in the cardiac band, and scored by the minimum pairwise wavelength correlation. Scores are copied to every wavelength in the group. The default threshold is a project default, not a clinical cutoff.

Usage

```
scalpCouplingIndex(
  x,
  assay_name = "OD",
  l_freq = 0.7,
  h_freq = 1.5,
  window_seconds = NULL,
  step_seconds = NULL,
  threshold = 0.8,
  order = 3L
)
```

Arguments

x	A governed measurement-level optical-density <code>PhysioExperiment</code> .
assay_name	Exact optical-density assay name.
l_freq, h_freq	Cardiac-band limits in Hz.
window_seconds	Optional positive complete-window duration.
step_seconds	Optional positive window step. <code>NULL</code> uses the window.
threshold	Finite pass threshold in $[-1, 1]$.
order	Positive Butterworth filter order.

Value

A source-bound `nirs_quality` result.

References

Pollonini et al. (2014), DOI: 10.1117/1.JBO.19.8.086007.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
scalpCouplingIndex(od)
```

shortSeparationDesign *Construct a governed short-separation nuisance design*

Description

Returns short-channel signals, optionally aggregated by compatibility class, with requested physiology-band copies. No intercept, task, or drift regressor is added.

Usage

```
shortSeparationDesign(
  x,
  assay_name = "OD",
  short = NULL,
  aggregation = c("all", "mean"),
  physiology_bands = character(),
  band_ranges = NULL,
  standardize = TRUE
)
```

Arguments

x	A governed OD or pair-collapsed haemoglobin PhysioExperiment.
assay_name	Exact source assay.
short	A compatible nirs_short_channels map returned by identifyShortChannels(), or NULL for the strict 10-mm default.
aggregation	Exactly "all" or "mean".
physiology_bands	Unique exact band names to append.
band_ranges	Optional named list providing one explicit range per requested band.
standardize	Whether to centre and sample-SD standardize every column.

Value

A time-aligned nirs_nuisance_design numeric matrix.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
short <- identifyShortChannels(od, threshold_m = 0.015)
design <- shortSeparationDesign(od, short = short)
dim(design)
```

shortSeparationRegress

Regress governed short-separation nuisance signals

Description

Fits a centred short-channel regression separately for each long channel. Short channels are copied unchanged. Positive lag means that the short signal at $t - \text{lag_seconds}$ predicts the long signal at t .

Usage

```
shortSeparationRegress(
  x,
  assay_name = "OD",
  short = NULL,
  method = c("nearest", "all"),
  lag_seconds = 0,
  output_assay = paste0(assay_name, "_ssr"),
  condition_limit = 1e+10
)
```

Arguments

<code>x</code>	A governed OD or pair-collapsed haemoglobin PhysioExperiment.
<code>assay_name</code>	Exact source assay name.
<code>short</code>	A compatible <code>nirs_short_channels</code> map returned by <code>identifyShortChannels()</code> , or <code>NULL</code> to use the strict 10-mm default.
<code>method</code>	Exactly "nearest" or "all".
<code>lag_seconds</code>	A finite lag that corresponds to an exact sample count.
<code>output_assay</code>	Exact name for the added corrected assay.
<code>condition_limit</code>	Finite regression condition-number limit greater than one.

Details

For a target y and selected short signals X , the fitted intercept is retained and only $(X - \text{colMeans}(X))$ beta is subtracted. Regression uses a deterministic SVD solve and rejects rank-deficient or over-limit designs. Short columns and lag-edge rows remain unchanged.

Value

A clone of `x` with one short-separation-corrected assay.

References

Saager and Berger (2005), DOI: 10.1364/JOSAA.22.001874.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
short <- identifyShortChannels(od, threshold_m = 0.015)
corrected <- shortSeparationRegress(od, short = short)
SummarizedExperiment::assayNames(corrected)
```

signalQualityIndex	<i>Calculate governed fNIRS signal quality</i>
--------------------	--

Description

method = "peak_power" implements the pinned MNE-NIRS PHOEBE-style cross-correlation peak-power metric. It is not the original PHOEBE equation. method = "snr" returns a channel-specific cardiac-to-noise spectral power ratio in decibels.

Usage

```
signalQualityIndex(
  x,
  assay_name = "OD",
  method = c("peak_power", "snr"),
  window_seconds = 10,
  step_seconds = NULL,
  cardiac_range_hz = c(0.7, 1.5),
  noise_range_hz = NULL,
  threshold = NULL,
  order = 3L
)
```

Arguments

x	A governed measurement-level optical-density PhysioExperiment.
assay_name	Exact optical-density assay name.
method	Exactly "peak_power" or "snr".
window_seconds	Positive complete-window duration.
step_seconds	Optional positive window step. NULL uses the window.
cardiac_range_hz	Finite increasing cardiac band below Nyquist.
noise_range_hz	For SNR, NULL or a disjoint two-column band matrix.
threshold	Optional finite threshold. Defaults to 0.1 for peak power and 0 dB for SNR.
order	Positive Butterworth filter order.

Value

A source-bound nirs_quality result.

References

Pollonini et al. (2016), DOI: 10.1364/BOE.7.005104.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
signalQualityIndex(od)
```

sourceDetectorDistances

Calculate source-detector distances for each SNIRF measurement

Description

Calculate source-detector distances for each SNIRF measurement

Usage

```
sourceDetectorDistances(
  x,
  dimension = c("auto", "3d", "2d"),
  unit = c("m", "native")
)
```

Arguments

x	A governed PhysioExperiment.
dimension	Geometry selection: exactly "auto", "3d", or "2d".
unit	Output unit: exactly "m" or "native".

Value

A DataFrame with one row per measurement and distance identity.

Examples

```
pe <- PhysioNIRS:::nirs_demo_object()
sourceDetectorDistances(pe)
```

splineMotionCorrect *Correct detected fNIRS motion intervals with cubic smoothing splines*

Description

Uses the MATLAB csaps smoothing-parameter definition and stitches corrected artifact and clean segments with short local means.

Usage

```
splineMotionCorrect(
    x,
    mask = NULL,
    assay_name = "OD",
    output_assay = paste0(assay_name, "_spline"),
    p = 0.99,
    t_motion = 0.5,
    t_mask = 1,
    sd_threshold = 20,
    amplitude_threshold = 0.5,
    group_wavelengths = TRUE
)
```

Arguments

x	A governed PhysioExperiment.
mask	A compatible nirs_motion_mask, or NULL to detect once.
assay_name	Exact source assay name.
output_assay	Exact new assay name.
p	Cubic smoothing parameter in $[0, 1]$.
t_motion	Positive detection lag window in seconds, used only when mask = NULL.
t_mask	Non-negative detection-mask expansion in seconds, used only when mask = NULL.
sd_threshold	Positive derivative-SD detection multiplier, used only when mask = NULL.
amplitude_threshold	Positive absolute-change detection threshold, used only when mask = NULL.
group_wavelengths	Whether detection unions wavelengths from each source-detector pair, used only when mask = NULL.

Value

A clone of x with one motion-corrected assay.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
corrected <- splineMotionCorrect(od)
SummarizedExperiment::assayNames(corrected)
```

tddr

*Repair fNIRS motion artifacts with TDDR***Description**

Implements temporal derivative distribution repair with a third-order Butterworth low-pass and Tukey bisquare derivative weights.

Usage

```
tddr(
  x,
  assay_name = "OD",
  output_assay = paste0(assay_name, "_tddr"),
  cutoff_hz = 0.5,
  tune = 4.685,
  max_iter = 50L
)
```

Arguments

x	A governed PhysioExperiment.
assay_name	Exact source assay name.
output_assay	Exact new assay name.
cutoff_hz	Positive low-pass cutoff below Nyquist.
tune	Positive Tukey bisquare tuning constant.
max_iter	Positive integer maximum robust iterations.

Value

A clone of x with one motion-corrected assay.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())
corrected <- tddr(od)
SummarizedExperiment::assayNames(corrected)
```

waveletMotionCorrect	<i>Correct fNIRS motion with a shift-invariant db2 wavelet transform</i>
----------------------	--

Description

Outlying detail coefficients are removed using a per-block IQR rule and the signal is reconstructed at its original length.

Usage

```
waveletMotionCorrect(  
  x,  
  assay_name = "OD",  
  output_assay = paste0(assay_name, "_wavelet"),  
  iqr = 1.5,  
  wavelet = "db2",  
  min_level = 4L  
)
```

Arguments

x	A governed PhysioExperiment.
assay_name	Exact source assay name.
output_assay	Exact new assay name.
iqr	Positive IQR multiplier.
wavelet	Exact wavelet name; currently only "db2".
min_level	Positive integer lowest analysis scale.

Value

A clone of x with one motion-corrected assay.

Examples

```
od <- intensityToOD(PhysioNIRS:::nirs_demo_object())  
corrected <- waveletMotionCorrect(od)  
SummarizedExperiment::assayNames(corrected)
```

`writeSNIRF`*Write a Shared Near Infrared Spectroscopy Format file*

Description

Write a Shared Near Infrared Spectroscopy Format file

Usage

```
writeSNIRF(x, path, assay_name = NULL, overwrite = FALSE, compact_time = FALSE)
```

Arguments

<code>x</code>	A governed <code>PhysioExperiment</code> .
<code>path</code>	Destination ending exactly in <code>.snirf</code> .
<code>assay_name</code>	Assay to write. <code>NULL</code> uses the default assay.
<code>overwrite</code>	Whether to replace an existing destination.
<code>compact_time</code>	Whether to use <code>[start, spacing]</code> time encoding.

Value

`path`, invisibly.

Examples

```
# Write a governed object to a temporary SNIRF/HDF5 file
pe <- PhysioNIRS:::nirs_demo_object()
pe <- PhysioExperiment::setEvents(pe, PhysioExperiment::PhysioEvents(
  onset = 6, duration = 4, type = "task", value = "1"))
path <- tempfile(fileext = ".snirf")
writeSNIRF(pe, path)
file.exists(path)
unlink(path)
```

Index

dpf, [3](#)
dpf(), [7](#)

extinctionCoefficients, [4](#)

identifyShortChannels, [5](#)
intensityToOD, [6](#)

mbll, [7](#)
mbll(), [10](#), [11](#)
measurementList, [8](#)
motionArtifactDetect, [8](#)

nirsActivationContrast, [9](#)
nirsActivationContrast(), [11](#)
nirsActivationGLM, [10](#)
nirsActivationGLM(), [9](#), [10](#)
nirsNeurofeedback, [11](#)
nirsNeurofeedback-lifecycle, [13](#)
nirsNeurofeedbackScope
 (nirsNeurofeedback-lifecycle),
 [13](#)
nirsNeurofeedbackStart
 (nirsNeurofeedback-lifecycle),
 [13](#)
nirsNeurofeedbackState
 (nirsNeurofeedback-lifecycle),
 [13](#)
nirsNeurofeedbackStep
 (nirsNeurofeedback-lifecycle),
 [13](#)
nirsNeurofeedbackStop
 (nirsNeurofeedback-lifecycle),
 [13](#)

PhysioExperiment::getEvents, [10](#)
PhysioExperiment::PhysioExperiment(),
 [3](#), [17](#)
physiologyBandpass, [14](#)
PhysioNIRS (PhysioNIRS-package), [3](#)
PhysioNIRS-package, [3](#)

print.nirs_quality, [15](#)
print.NIRSNeurofeedback, [15](#)
pruneChannels, [16](#)

readSNIRF, [17](#)

scalpCouplingIndex, [18](#)
shortSeparationDesign, [19](#)
shortSeparationDesign(), [11](#)
shortSeparationRegress, [20](#)
signalQualityIndex, [21](#)
sourceDetectorDistances, [22](#)
splineMotionCorrect, [23](#)

tddr, [24](#)

waveletMotionCorrect, [25](#)
writeSNIRF, [26](#)