

# Package: PhysioExperiment (via r-universe)

October 3, 2026

**Title** Shared Data Model for Physiological Signal Analysis

**Version** 2.3.1

**Description** Provides the shared data model for the Physio ecosystem: the PhysioExperiment class, a 'SummarizedExperiment' extension for multi-modal physiological signal data (such as electroencephalography, electromyography, and electrocardiography), together with the containers for multiple simultaneously recorded streams, repeated sessions of one subject, and cohorts of subjects. Includes core accessors, channel and event management, provenance, and basic 'S4' methods. Domain packages apply specialized methods to these shared containers and are installed separately.

**biocViews** Software, Infrastructure, DataRepresentation, TimeCourse

**Depends** R (>= 4.2)

**Imports** methods, SummarizedExperiment, S4Vectors, stats, lifecycle, BiocGenerics, digest

**Suggests** testthat (>= 3.2.0), abind, ggplot2, farver, colorspace, withr, knitr, rmarkdown, BiocStyle, MultiAssayExperiment, PhysioMSKNet

**VignetteBuilder** knitr

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**Collate** 'PhysioExperiment-class.R' 'PhysioExperiment-accessors.R' 'PhysioExperiment-methods.R' 'provenance.R' 'MultiPhysioExperiment-class.R' 'container-migration.R' 'trials.R' 'multirate-accessors.R' 'PhysioLongitudinal-class.R' 'PhysioCohort-class.R' 'channels.R' 'harmonize.R' 'change-scores.R' 'events.R' 'event-query.R' 'utils-signal.R' 'utils-mvar.R' 'utils-na.R' 'clinimetrics.R' 'analysis-result.R' 'normative.R' 'biomarker.R' 'vis-theme.R' 'registry.R' 'lifecycle.R' 'stats-circular.R' 'stats-reliability.R' 'stats-fda.R' 'zzz.R'

**RoxygenNote** 7.3.3

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

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

**Config/pak/sysreqs** zliblg-dev

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

**Date/Publication** 2026-10-03 13:11:53 UTC

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

**RemoteRef** HEAD

**RemoteSha** e65059e55edbd9bf0568f094cdeb97124805084c

## Contents

|                                                      |    |
|------------------------------------------------------|----|
| [,MultiPhysioExperiment,ANY,ANY,ANY-method . . . . . | 5  |
| [,PhysioExperiment,ANY,ANY,ANY-method . . . . .      | 6  |
| addEvents . . . . .                                  | 7  |
| addProvenance . . . . .                              | 8  |
| addSession . . . . .                                 | 9  |
| addSubject . . . . .                                 | 10 |
| aggregatePhysioFeatures . . . . .                    | 11 |
| aggregateTrials . . . . .                            | 13 |
| aggregateTrialWaveforms . . . . .                    | 14 |
| alignStreams . . . . .                               | 16 |
| AnalysisResult . . . . .                             | 17 |
| AnalysisResult-accessors . . . . .                   | 18 |
| AnalysisResult-carrier . . . . .                     | 19 |
| AnalysisResult-class . . . . .                       | 20 |
| appendProvenance . . . . .                           | 21 |
| applyMontage . . . . .                               | 22 |
| as.data.frame,PhysioExperiment-method . . . . .      | 23 |
| as.data.frame.PhysioBiomarker . . . . .              | 24 |
| asMSKTracker . . . . .                               | 24 |
| asMultiAssayExperiment . . . . .                     | 25 |
| assaySamplingRates . . . . .                         | 26 |
| blandAltman . . . . .                                | 27 |
| cbindPhysio . . . . .                                | 28 |
| changeScores . . . . .                               | 29 |
| channelInfo . . . . .                                | 30 |
| channelInfo<- . . . . .                              | 31 |
| channelNames . . . . .                               | 32 |
| channelNames<- . . . . .                             | 33 |
| checkNA . . . . .                                    | 34 |
| circularLinearCorrelation . . . . .                  | 35 |
| circularSummary . . . . .                            | 36 |

|                                            |    |
|--------------------------------------------|----|
| cohensD . . . . .                          | 36 |
| cohortData . . . . .                       | 38 |
| cohortDesign . . . . .                     | 38 |
| cohortTrials . . . . .                     | 39 |
| commonClock . . . . .                      | 40 |
| defaultAssay . . . . .                     | 40 |
| deprecate_physio . . . . .                 | 41 |
| design,PhysioLongitudinal-method . . . . . | 42 |
| detectGaps . . . . .                       | 43 |
| dim,PhysioExperiment-method . . . . .      | 44 |
| dropChannels . . . . .                     | 45 |
| duration . . . . .                         | 46 |
| elapsedBetweenTrials . . . . .             | 47 |
| etaSquared . . . . .                       | 47 |
| eventQuery . . . . .                       | 49 |
| EventQuery-class . . . . .                 | 49 |
| extractTrialFeatures . . . . .             | 50 |
| extractWindow . . . . .                    | 51 |
| fillEdgeNA . . . . .                       | 52 |
| filterType . . . . .                       | 53 |
| filterValue . . . . .                      | 54 |
| format.PhysioBiomarker . . . . .           | 55 |
| fPCA . . . . .                             | 55 |
| getChannelsByType . . . . .                | 57 |
| getElectrodePositions . . . . .            | 58 |
| getEvents . . . . .                        | 59 |
| getReference . . . . .                     | 60 |
| handleNA . . . . .                         | 61 |
| harmonize . . . . .                        | 62 |
| harmonizeChannels . . . . .                | 63 |
| harmonizeMontage . . . . .                 | 64 |
| harmonizeReference . . . . .               | 64 |
| harmonizeReport . . . . .                  | 65 |
| hasMeasuredTimes . . . . .                 | 65 |
| hasNA . . . . .                            | 66 |
| icc . . . . .                              | 67 |
| is.PhysioBiomarker . . . . .               | 68 |
| length,PhysioExperiment-method . . . . .   | 69 |
| logStep . . . . .                          | 70 |
| mdc . . . . .                              | 71 |
| migrateContainer . . . . .                 | 72 |
| multiphysio-legacy-accessors . . . . .     | 73 |
| MultiPhysioExperiment . . . . .            | 74 |
| MultiPhysioExperiment-class . . . . .      | 76 |
| MultiPhysioExperiment-methods . . . . .    | 77 |
| MultiRatePhysioExperiment-class . . . . .  | 77 |
| mvarFit . . . . .                          | 78 |
| mvarOrderSelect . . . . .                  | 79 |

|                                      |     |
|--------------------------------------|-----|
| mvarTransfer . . . . .               | 79  |
| naSummary . . . . .                  | 80  |
| nChannels . . . . .                  | 81  |
| nEvents . . . . .                    | 82  |
| normativeLookup . . . . .            | 83  |
| NormativeReference . . . . .         | 83  |
| NormativeReference-class . . . . .   | 84  |
| NormativeReference-compare . . . . . | 85  |
| nStreams . . . . .                   | 86  |
| physioBiomarker . . . . .            | 86  |
| PhysioBiomarker . . . . .            | 88  |
| PhysioBiomarker-class . . . . .      | 88  |
| PhysioCohort . . . . .               | 89  |
| PhysioCohort-class . . . . .         | 90  |
| PhysioCohort-methods . . . . .       | 91  |
| PhysioEvents . . . . .               | 92  |
| PhysioEvents-class . . . . .         | 93  |
| PhysioExperiment . . . . .           | 93  |
| PhysioExperiment-class . . . . .     | 95  |
| PhysioLongitudinal . . . . .         | 95  |
| PhysioLongitudinal-class . . . . .   | 96  |
| PhysioLongitudinal-methods . . . . . | 97  |
| physioPalette . . . . .              | 98  |
| pickChannels . . . . .               | 99  |
| print.fPCA_result . . . . .          | 100 |
| provenance . . . . .                 | 100 |
| provenanceHash . . . . .             | 102 |
| rayleighTest . . . . .               | 102 |
| rbindPhysio . . . . .                | 103 |
| readPhysioRDS . . . . .              | 104 |
| reconstructFPCA . . . . .            | 105 |
| recordActivity . . . . .             | 106 |
| recordingMap . . . . .               | 107 |
| recordings . . . . .                 | 108 |
| registerCurves . . . . .             | 108 |
| registerOperation . . . . .          | 110 |
| registerReader . . . . .             | 111 |
| registerWriter . . . . .             | 112 |
| removeEvents . . . . .               | 113 |
| renameChannels . . . . .             | 114 |
| replaceNA . . . . .                  | 115 |
| resampleToCommon . . . . .           | 116 |
| resolveQuery . . . . .               | 116 |
| samplesToTime . . . . .              | 117 |
| samplingRate . . . . .               | 118 |
| scale_color_physio . . . . .         | 119 |
| sem . . . . .                        | 120 |
| session . . . . .                    | 121 |

|                                 |     |
|---------------------------------|-----|
| sessions                        | 121 |
| setAssaySamplingRate            | 122 |
| setChannelTypes                 | 123 |
| setChannelUnits                 | 124 |
| setElectrodePositions           | 125 |
| setEvents                       | 126 |
| setReference                    | 127 |
| show,PhysioEvents-method        | 128 |
| show,PhysioExperiment-method    | 128 |
| streamCoverage                  | 129 |
| streamNames                     | 130 |
| streamRates                     | 131 |
| streams                         | 131 |
| streamTimeIndex                 | 132 |
| subject                         | 133 |
| subjectData                     | 133 |
| subjectIds                      | 134 |
| subjects                        | 134 |
| subjectTrials                   | 135 |
| subsetCohort                    | 136 |
| summary,PhysioExperiment-method | 136 |
| theme_physio                    | 137 |
| timeIndex                       | 138 |
| timeToSamples                   | 139 |
| timeWindow                      | 140 |
| trial                           | 141 |
| trialCoverage                   | 142 |
| trialIntervals                  | 143 |
| trials                          | 144 |
| trialSet                        | 144 |
| watsonWilliamsTest              | 145 |
| waveformCMC                     | 146 |
| waveformICC                     | 147 |
| waveformReliability             | 148 |
| withProvenance                  | 149 |
| wpliEstimate                    | 150 |

**Index****152**

[,MultiPhysioExperiment,ANY,ANY,ANY-method

*Subset a MultiPhysioExperiment by time and stream***Description**

`x[c(start, end), streams]` is shorthand for `timeWindow(x, start, end, streams)`.

**Usage**

```
## S4 method for signature 'MultiPhysioExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]
```

**Arguments**

|      |                                                       |
|------|-------------------------------------------------------|
| x    | A MultiPhysioExperiment.                              |
| i    | Numeric vector of length 2: c(start, end) in seconds. |
| j    | Character vector of stream names.                     |
| ...  | Ignored.                                              |
| drop | Ignored; present for signature compatibility.         |

**Details**

This is the former **PhysioCrossModal** [ method. Its meaning is preserved, with one deliberate correction: the old implementation computed indices as  $\text{floor}(t * \text{rate}) + 1$  from the rate alone, which ignored each stream's start offset and could admit one sample lying before start. Selection is now made on each sample's actual time on the shared clock. For windows whose bounds land on sample times - including all zero-offset data with grid-aligned bounds - the selection is unchanged.

**Value**

A MultiPhysioExperiment.

---

[,PhysioExperiment,ANY,ANY,ANY-method]

*Subset PhysioExperiment by time indices*

---

**Description**

Extracts a subset of the PhysioExperiment by row (time) and/or column (channel) indices, preserving metadata and appending a processing-history entry with the supplied indices and input/output dimensions. Missing indices select the full corresponding axis.

**Usage**

```
## S4 method for signature 'PhysioExperiment,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]
```

**Arguments**

|      |                                                        |
|------|--------------------------------------------------------|
| x    | A PhysioExperiment object.                             |
| i    | Time indices (rows).                                   |
| j    | Channel indices (columns in first non-time dimension). |
| ...  | Additional arguments (not used).                       |
| drop | Logical. If TRUE, drops dimensions of size 1.          |

**Value**

A `PhysioExperiment` object containing only the selected time points and/or channels, with updated `rowData` and `colData`.

**References**

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:10.1038/nmeth.3252

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

[extractWindow](#) for subsetting by time in seconds, [pickChannels](#) for subsetting by channel name, [dropChannels](#) for removing specific channels

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)

# Subset by time
pe_subset <- pe[1:50, ]
dim(pe_subset) # 50 4

# Subset by channels
pe_channels <- pe[, 1:2]
dim(pe_channels) # 100 2
```

addEvents

*Add events to a PhysioExperiment object***Description**

Add events to a `PhysioExperiment` object

**Usage**

```
addEvents(x, onset, duration = 0, type = "event", value = "")
```

**Arguments**

|                       |                                                 |
|-----------------------|-------------------------------------------------|
| <code>x</code>        | A <code>PhysioExperiment</code> object.         |
| <code>onset</code>    | Numeric vector of event onset times in seconds. |
| <code>duration</code> | Numeric vector of event durations in seconds.   |
| <code>type</code>     | Character vector of event types.                |
| <code>value</code>    | Character vector of event values/labels.        |

**Value**

The modified `PhysioExperiment` object.

**References**

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[setEvents](#) for replacing all events, [removeEvents](#) for removing events, [getEvents](#) for retrieving events, [nEvents](#) for event count

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 100)),
  samplingRate = 100
)

# Add stimulus events
pe <- addEvents(pe, onset = c(1, 2, 3), type = "stimulus")

# Add response events
pe <- addEvents(pe, onset = c(1.5, 2.5), type = "response", value = c("hit", "hit"))
nEvents(pe) # 5 events total
```

---

addProvenance

*Append a provenance step, capturing an optional seed*

---

**Description**

A thin convenience over [logStep](#) that records an append-only provenance entry on the object's metadata log (never dropping prior entries), optionally capturing an integer seed in the entry parameters so the step is reproducible under a fixed seed.

**Usage**

```
addProvenance(
  pe,
  step,
  params = list(),
  seed = NULL,
  package = NA_character_,
  version = NA_character_
)
```

**Arguments**

pe                    A `PhysioExperiment` object.  
 step                 Character step / activity name.  
 params               Named list of parameters recorded with the step.  
 seed                 Optional integer seed captured in the entry parameters.  
 package, version    Optional producing-package name and version.

**Value**

pe with the entry appended to its provenance log.

**See Also**

[logStep](#), [provenance](#)

**Examples**

```

pe <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
pe <- addProvenance(pe, "conformalInterval", seed = 42)
provenance(pe)

```

---

addSession

---

*Add a session to a `PhysioLongitudinal`*


---

**Description**

Appends a session and re-sorts the container into chronological order by `days_from_baseline`.

**Usage**

```

addSession(
  x,
  label,
  pe,
  days_from_baseline,
  condition = NA_character_,
  session_id = label
)

```

Arguments

|                    |                                                                            |
|--------------------|----------------------------------------------------------------------------|
| x                  | A <code>PhysioLongitudinal</code> .                                        |
| label              | The visit label (e.g. "followup").                                         |
| pe                 | A <code>PhysioExperiment</code> / <code>MultiRatePhysioExperiment</code> . |
| days_from_baseline | Numeric days since the baseline visit.                                     |
| condition          | Optional condition string.                                                 |
| session_id         | Session id (defaults to label).                                            |

Value

The updated `PhysioLongitudinal`, re-sorted chronologically.

See Also

[PhysioLongitudinal](#)

---

|            |                                                     |
|------------|-----------------------------------------------------|
| addSubject | <i>Add a subject to a <code>PhysioCohort</code></i> |
|------------|-----------------------------------------------------|

---

Description

Add a subject to a `PhysioCohort`

Usage

```
addSubject(x, id, subject, group = NA_character_, meta = list())
```

Arguments

|         |                                                                         |
|---------|-------------------------------------------------------------------------|
| x       | A <code>PhysioCohort</code> .                                           |
| id      | The new subject's id (must be unique).                                  |
| subject | A <code>PhysioLongitudinal</code> or bare experiment (auto-wrapped).    |
| group   | Optional group/arm label for the new subject.                           |
| meta    | Optional named list of extra subject-level <code>colData</code> values. |

Value

The updated `PhysioCohort`.

See Also

[PhysioCohort](#)

---

 aggregatePhysioFeatures

*Aggregate features through trial, session, subject and cohort levels*


---

## Description

Summarizes scalar features while preserving observation identity. Higher targets are reached one level at a time: cycles to trials, trials to sessions, sessions to subjects, and subjects to a cohort. With the default mean, each immediate child receives equal weight, regardless of how many observations contributed to that child. These are descriptive summaries, not models or estimates of sampling uncertainty.

## Usage

```
aggregatePhysioFeatures(
  x,
  level,
  features = NULL,
  strata = NULL,
  FUN = mean,
  na.rm = FALSE,
  ...
)
```

## Arguments

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x        | A nonempty data.frame or S4Vectors DataFrame of features, or an unmodified result of this function or <code>extractTrialFeatures()</code> . Extracted features retain their trial selections and sample references in extraction; <code>source_rows</code> indexes the rows of <code>extraction\$data</code> . Tables require <code>subject_id</code> and the consecutive keys down to their observation level: <code>session_id</code> , <code>trial_id</code> , <code>cycle_id</code> . For example, trial-level input has <code>subject</code> , <code>session</code> and <code>trial</code> keys, with one row per trial and stratum. The deepest key determines the input level. Keys must be nonmissing, nonempty character, factor or finite numeric columns. Only keys, strata and selected features are retained. |
| level    | Required target: "trial", "session", "subject" or "cohort". Must be strictly above the input level.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| features | Names of numeric scalar feature columns, required for a new table. Inherited when continuing a previous aggregation result.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| strata   | Additional columns to keep separate at every level, such as <code>c("side", "condition")</code> . Defaults to no strata for a new table; inherited for a previous result. Strata cannot be changed during a chained operation. Include modality/channel identifiers here when using a long feature table.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| FUN      | Function, or function name, reducing a nonempty numeric vector to one finite numeric value. Defaults to mean; median and sum are other examples. Applied separately to each feature at each intervening level.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|                    |                                                                                                                                                                                                   |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>na.rm</code> | Logical. Missing features cause an error by default. If TRUE, omit missing children separately for each feature; an all-missing group is retained with NA. Infinite values always cause an error. |
| <code>...</code>   | Additional arguments to FUN. Weight vectors tied to original rows are not supported: vector lengths change at successive levels.                                                                  |

## Details

Repeated session/trial names in different parents remain distinct. Duplicate full observation keys, including strata, are rejected. Identifiers are normalized to character. Group order follows first appearance in the input.

To give trials equal weight, the mean of a session with trial means 1 and 9 is 5 even if the trials contain different numbers of cycles. Requesting a subject or cohort directly performs the same intervening steps as chaining results with the same FUN and arguments. For nonlinear FUN, this is a nested summary (e.g. a median of trial medians), not a pooled summary. Custom functions should be deterministic and depend only on their supplied inputs; recorded function text does not capture external state or closure values.

Features must have compatible units and meaning within each group; this function cannot infer those from numeric columns. It does not align or average raw waveforms, construct trial containers, infer missing IDs, or establish statistical independence. Absent trials are not counted; only supplied rows and their missing feature values can be audited.

For a modality returned by `PhysioMoCap`'s `summarizeCycleFeatures()`, combine its key table and feature block with `cbind(z$keys$emg, z$blocks$emg)` and specify the feature names and `strata = "side"`. Different modalities can be summarized separately using the same hierarchy and strata.

## Value

A `physio_aggregation` list with `data` (one row per target unit and stratum), `features`, `strata`, `level`, `source` (the retained original input table), and `source_rows` (a list mapping each output row to rows of source, including rows with missing features). `steps` records each reached level's data, source mapping, method text, arguments and missing-value policy. Each step's counts has output-row and feature identifiers, `n_units` (immediate children), `n_used`, `n_missing`, and `n_source` (mapped original rows, not an effective sample size). `counts` also exposes the final step. A checksum detects changed data or lineage before further aggregation.

## Examples

```
cycles <- data.frame(subject_id = "person1", session_id = "session1",
  trial_id = c("trial1", "trial1", "trial2"),
  cycle_id = c("cycle1", "cycle2", "cycle1"), amplitude = c(0, 2, 9))
trials <- aggregatePhysioFeatures(cycles, "trial", "amplitude")
sessions <- aggregatePhysioFeatures(trials, "session")
sessions$data                # mean(c(1, 9)) = 5
sessions$counts              # two trials, three original cycles
aggregatePhysioFeatures(cycles, "session", "amplitude")$data
```

---

|                 |                                                                      |
|-----------------|----------------------------------------------------------------------|
| aggregateTrials | <i>Aggregate per-trial features, choosing what counts as a trial</i> |
|-----------------|----------------------------------------------------------------------|

---

## Description

Selects rows from the trial table and hands them to [aggregatePhysioFeatures](#). The aggregation itself - the equal weighting of immediate children at each level, the link back to source rows, the checksum - is unchanged: choosing which trials take part is a selection of input rows, not a reweighting.

## Usage

```
aggregateTrials(
  x,
  level,
  features = NULL,
  include = "complete",
  rule = "nominal",
  gap_factor = 1.5,
  ...
)
```

## Arguments

|                  |                                                                                                                                                                                 |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x                | A MultiPhysioExperiment carrying a trial table.                                                                                                                                 |
| level            | Target level, passed to <a href="#">aggregatePhysioFeatures</a> .                                                                                                               |
| features         | Feature column names on the trial table.                                                                                                                                        |
| include          | Which states take part: any of "complete", "partial", "unknown", "undescribed". Defaults to "complete" – only trials shown to be whole take part unless another state is named. |
| rule, gap_factor | Passed to <a href="#">streamCoverage</a> when judging completeness.                                                                                                             |
| ...              | Passed to <a href="#">aggregatePhysioFeatures</a> .                                                                                                                             |

## Details

Each trial is classified before anything is averaged, because averaging a truncated trial together with whole ones silently mixes different amounts of evidence:

"complete" Every stream the trial names covers it fully.

"partial" A stream covers less of the interval than it asks for. The interval was cut by a window, or it runs past the end of the recording, or [streamCoverage](#) reports a dropout inside it.

"unknown" Completeness cannot be decided here. Coverage is undecidable; or a gap an earlier window recorded cannot be re-derived under this query's rate or gap\_factor, so the dropout can be neither confirmed nor ruled out; or the stream holds more samples in the interval than its declared rate predicts – nothing is missing, but the grid does not describe it. None of these is the same as complete.

"undescribed" No interval is recorded for the trial, so nothing describes its extent. Excluded by default: a trial table of per-trial values with no intervals is a job for [aggregatePhysioFeatures](#) directly, or ask for `include = "undescribed"` here.

Every judgement comes from [streamCoverage](#) – the shortfall, the dropout it reports, and what it says it could not re-derive – so a trial is excluded by the same rule that reports the problem rather than by a second definition of it.

The states are decided in that order: shown to be short, then unable to tell, then shown to be whole. Losing the evidence for a gap therefore never promotes a trial to "complete" – a question that can no longer be answered is not an answer of yes – while a shortfall this query can still demonstrate outranks a record it cannot reuse.

### Value

A list with `aggregate` (the aggregation result), counts (one entry per state plus `n_included / n_excluded`), states (the state of every trial) and excluded.

### See Also

[aggregatePhysioFeatures](#), [trialIntervals](#), [streamCoverage](#)

---

aggregateTrialWaveforms

*Align and average trial waveforms through the experimental hierarchy*

---

### Description

Evaluates one stream on an explicitly supplied trial-relative grid, then averages trials, sessions and subjects with equal weight at each level. Measured sample times take priority over reconstructed sampling times.

### Usage

```
aggregateTrialWaveforms(
  x,
  stream,
  grid,
  align = c("start", "phase"),
  level = c("session", "subject", "cohort"),
  assay = "raw",
  channels = NULL,
  strata = NULL,
  anchors = NULL,
  include = "complete",
  gap_factor = 1.5,
  na.rm = FALSE
)
```

**Arguments**

|            |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x          | A MultiPhysioExperiment, a named list of these recordings, a PhysioLongitudinal, or a PhysioCohort. Each recording must carry trial and interval tables for one subject and session. For a trial set, pass recordings(x) with these tables populated; map rows alone do not define waveform intervals. Duplicate full trial keys are rejected.                                                                                              |
| stream     | One stream name, evaluated separately from other modalities.                                                                                                                                                                                                                                                                                                                                                                                |
| grid       | Finite, strictly increasing evaluation points. For start, seconds relative to the alignment start; for phase, fractions in $[0, 1]$ . No output grid or sampling rate is inferred.                                                                                                                                                                                                                                                          |
| align      | Either start or phase. Phase maps each trial's supplied start and end to zero and one; this changes its time scale.                                                                                                                                                                                                                                                                                                                         |
| level      | Target: session, subject, or cohort.                                                                                                                                                                                                                                                                                                                                                                                                        |
| assay      | Name of the two-dimensional numeric signal assay.                                                                                                                                                                                                                                                                                                                                                                                           |
| channels   | Unique channel names. By default all channels are used and their sets must agree across included recordings. Column order may differ.                                                                                                                                                                                                                                                                                                       |
| strata     | Trial-table columns, such as condition, kept separate at every aggregation level. Modalities and channels are never averaged together.                                                                                                                                                                                                                                                                                                      |
| anchors    | Optional data frame with subject_id, session_id, trial_id, start and end, keyed uniquely by the first three columns. Times use each recording's local shared clock. Without this table, the union of all stream intervals defines each trial's anchors, preserving inter-stream offsets. Windowed inputs require explicit anchors: a cropped boundary is not an original trial onset. A supplied start may also be an event alignment time. |
| include    | Trial states to include: complete by default, or explicitly partial. Selection uses the coverage of all intervals named by the trial, as in <a href="#">aggregateTrials</a> . Unknown or undescribed trials cannot be included.                                                                                                                                                                                                             |
| gap_factor | Gap criterion passed to <a href="#">streamCoverage</a> .                                                                                                                                                                                                                                                                                                                                                                                    |
| na.rm      | If FALSE, any unavailable aligned value causes an error. TRUE omits it point-wise; an all-missing group is retained with mean NA and zero contributing trials. This can change contributing units along the waveform.                                                                                                                                                                                                                       |

**Details**

Linear interpolation uses adjacent finite samples inside the selected stream's trial interval. There is no extrapolation, interpolation across reported gaps, or skipping over missing signal values. Exact samples at gap boundaries remain usable. Included trials with unusable retained gap evidence are refused. The output is an interpolated representation, not antialias filtering; preprocess signals appropriately before choosing a coarser grid.

Channel names establish correspondence, not compatible units or physiology. The caller must provide comparable assays and units. Source hashes include times, selected signal values, channel metadata and interval/trial tables. Interpolation indices refer to rows in the input recording, including when that recording was previously windowed. Inputs themselves are not modified.

[aggregatePhysioFeatures](#) performs the hierarchy and preserves intermediate results and source-row links. The waveform SD or a confidence interval is not estimated: trials are not treated as independent participants.

**Value**

A list with data (mean waveform by output unit, channel and grid point), counts (immediate-child counts and n\_trials, the number of finite contributing original trial values), aggregate (the full hierarchical result), aligned (selected trial values), sample\_map (record, stream, channel, bracketing original row indices, interpolation weight and missingness reason for each aligned row), trials (all trial states and selections), anchors, settings, source hashes, and a result checksum.

**See Also**

[aggregateTrials](#), [streamTimeIndex](#)

**Examples**

```
pe <- PhysioExperiment(assays = list(raw = matrix(0:30, ncol = 1,
  dimnames = list(NULL, "signal"))), samplingRate = 10)
m <- MultiPhysioExperiment(sensor = pe)
trials(m) <- data.frame(subject_id = "P1", session_id = "S1",
  trial_id = c("T1", "T2"))
trialIntervals(m) <- data.frame(trials(m)[1:3], recording_id = "R1",
  stream = "sensor", start = c(0, 2), end = c(1, 3))
z <- aggregateTrialWaveforms(m, "sensor", grid = c(0, 0.5, 1))
z$data
z$counts
```

---

alignStreams

---

Align all streams to the reference rate

---

**Description**

Convenience wrapper for `resampleToCommon(x, reference_rate)` using the shared clock's reference rate.

**Usage**

```
alignStreams(x)
```

**Arguments**

x                      A MultiPhysioExperiment.

**Value**

A single-rate PhysioExperiment.

**See Also**

[resampleToCommon](#)

---

|                |                                    |
|----------------|------------------------------------|
| AnalysisResult | <i>Construct an AnalysisResult</i> |
|----------------|------------------------------------|

---

**Description**

Construct an AnalysisResult

**Usage**

```
AnalysisResult(  
  type,  
  result = list(),  
  parameters = list(),  
  provenance = data.frame(),  
  estimate = NULL,  
  uncertainty = list(),  
  method = NA_character_,  
  estimand = list()  
)
```

**Arguments**

|             |                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------|
| type        | Character analysis tag.                                                                                   |
| result      | Named list payload.                                                                                       |
| parameters  | Named list of parameters.                                                                                 |
| provenance  | Optional lineage data.frame.                                                                              |
| estimate    | Optional point estimate carried by the result.                                                            |
| uncertainty | Optional named list describing the interval (with a type of none/conformal/bayes/bootstrap/ncp/analytic). |
| method      | Optional character estimation-method label.                                                               |
| estimand    | Optional named list of ICH E9(R1) estimand attributes.                                                    |

**Value**

An AnalysisResult object.

**Examples**

```
AnalysisResult("hrv_time", result = list(sdn = 42, rmssd = 30))  
AnalysisResult("rom", estimate = 118,  
  uncertainty = list(type = "conformal", level = 0.9, lower = 104,  
    upper = 132),  
  estimand = list(population = "post-op knee", summary_measure = "median"))
```

---

**AnalysisResult-accessors***Accessors for analysis results*

---

**Description**

Accessors for analysis results

**Usage**

```
resultType(x)

## S4 method for signature 'AnalysisResult'
resultType(x)

resultValue(x)

## S4 method for signature 'AnalysisResult'
resultValue(x)

## S4 method for signature 'PhysioBiomarker'
resultValue(x)

biomarkerValue(x)

## S4 method for signature 'PhysioBiomarker'
biomarkerValue(x)
```

**Arguments**

x                      An AnalysisResult or PhysioBiomarker.

**Value**

resultType() the type tag; resultValue() the payload (or scalar value for a biomarker); biomarkerValue() the scalar value.

**Examples**

```
res <- AnalysisResult("hrv_time", result = list(sdn = 42, rmssd = 30))
resultType(res)
resultValue(res)

bm <- PhysioBiomarker("SDNN", 42, unit = "ms")
biomarkerValue(bm)
```

---

`AnalysisResult-carrier`*Estimand / uncertainty carrier accessors*

---

**Description**

Estimand / uncertainty carrier accessors

**Usage**

```
estimateOf(x)

## S4 method for signature 'AnalysisResult'
estimateOf(x)

uncertaintyOf(x)

## S4 method for signature 'AnalysisResult'
uncertaintyOf(x)

provenanceOf(x)

## S4 method for signature 'AnalysisResult'
provenanceOf(x)

estimandOf(x)

## S4 method for signature 'AnalysisResult'
estimandOf(x)
```

**Arguments**

`x`                      An AnalysisResult.

**Value**

`estimateOf()` the point estimate; `uncertaintyOf()` the interval list; `provenanceOf()` the provenance; `estimandOf()` the ICH E9(R1) estimand list.

**Examples**

```
r <- AnalysisResult("rom", estimate = 118,
  uncertainty = list(type = "conformal", level = 0.9, lower = 104,
                    upper = 132))
estimateOf(r)
uncertaintyOf(r)
```

---

AnalysisResult-class    *AnalysisResult: a uniform container for ecosystem analysis outputs*

---

### Description

A lightweight S4 container so every analysis (HRV, synergies, gait indices, ...) returns a consistently-shaped object that reporting and downstream code can consume uniformly.

### Usage

```
## S4 method for signature 'AnalysisResult'  
show(object)
```

### Arguments

object                    An object to display.

### Value

The show methods return NULL invisibly and are called for the side effect of printing a compact summary.

### Slots

type    Character tag identifying the analysis (e.g. "hrv\_time").  
result    Named list holding the payload.  
parameters    Named list of parameters used to produce the result.  
provenance    Optional data.frame lineage (see [provenance](#)).  
estimate    The point estimate the result carries (any type).  
uncertainty    Named list describing the interval, with a type in none, conformal, bayes, bootstrap, ncp or analytic, plus (typically) level, lower and upper.  
method    Character label of the estimation method.  
estimand    Named list of ICH E9(R1) estimand attributes (population, treatment, endpoint / variable, summary measure, intercurrent-event handling).

### See Also

[PhysioBiomarker](#)

---

|                  |                                     |
|------------------|-------------------------------------|
| appendProvenance | <i>Append a provenance activity</i> |
|------------------|-------------------------------------|

---

## Description

Appends one PROV activity to the object's audit trail. Two forms are supported: a high-level form that builds a full PROV-O record from named fields, and a low-level form that appends a pre-built entry list.

## Usage

```
appendProvenance(
  x,
  entry = NULL,
  activity = NULL,
  params = list(),
  input_assay = NA_character_,
  output_assay = NA_character_,
  agent = NA_character_,
  software_version = NA_character_,
  package = NA_character_
)
```

## Arguments

|                                        |                                                                                                                                           |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                         | A <code>PhysioExperiment</code> .                                                                                                         |
| <code>entry</code>                     | Either a pre-built entry list (low-level form) or a character scalar naming the activity (equivalent to passing <code>activity=</code> ). |
| <code>activity</code>                  | Character scalar naming the PROV activity (the operation).                                                                                |
| <code>params</code>                    | Named list of parameters to record.                                                                                                       |
| <code>input_assay, output_assay</code> | The assay(s) the activity used / generated (PROV used / wasGeneratedBy).                                                                  |
| <code>agent</code>                     | The responsible agent (defaults to <code>user@host</code> ).                                                                              |
| <code>software_version, package</code> | Software agent name and version.                                                                                                          |

## Value

`x` with the activity appended to its provenance log.

## Examples

```
pe <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
pe <- appendProvenance(pe, activity = "filterSignals",
  params = list(low = 1, high = 40),
```

```

                                input_assay = "raw", output_assay = "filtered")
provenance(pe)$activity

```

---

applyMontage

*Apply standard montage*

---

## Description

Applies a standard electrode montage (e.g., 10-20 system).

## Usage

```
applyMontage(x, system = c("10-20", "10-10", "10-5"))
```

## Arguments

**x**                      A `PhysioExperiment` object.

**system**                Montage system: "10-20", "10-10", or "10-5".

## Value

Modified `PhysioExperiment` object with electrode positions.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[setElectrodePositions](#) for custom positions, [getElectrodePositions](#) for reading positions, [channelNames](#) for channel labels, [setReference](#) for reference electrode

## Examples

```

pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)

# Apply 10-20 system positions
pe <- applyMontage(pe, "10-20")
getElectrodePositions(pe)

```

---

```
as.data.frame,PhysioExperiment-method
```

*Coerce to data.frame*

---

## Description

Converts the default assay of a `PhysioExperiment` to a `data.frame` with a time column (in seconds) followed by one column per channel.

## Usage

```
## S4 method for signature 'PhysioExperiment'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

## Arguments

|                        |                                         |
|------------------------|-----------------------------------------|
| <code>x</code>         | A <code>PhysioExperiment</code> object. |
| <code>row.names</code> | Unused.                                 |
| <code>optional</code>  | Unused.                                 |
| <code>...</code>       | Additional arguments.                   |

## Value

A `data.frame` with a time column and one column per channel. For 3D arrays, only the first sample (third dimension index 1) is returned. Returns an empty `data.frame` if no assays are present.

## References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](https://doi.org/10.1038/nmeth.3252)

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

## See Also

[summary,PhysioExperiment-method](#) for summary statistics, [timeIndex](#) for the time vector, [channelNames](#) for column names

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(12), nrow = 3, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)
df <- as.data.frame(pe)
head(df)
```

---

```
as.data.frame.PhysioBiomarker
```

*Coerce a PhysioBiomarker to a one-row data.frame*

---

### Description

Flattens every field - value, unit, CI, reference range, reliability (ICC/SEM/MDC), and provenance (assay/band/method/software version) - into a single row so biomarkers tabulate and round-trip losslessly.

### Usage

```
## S3 method for class 'PhysioBiomarker'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

### Arguments

|           |                                                                |
|-----------|----------------------------------------------------------------|
| x         | A <a href="#">PhysioBiomarker</a> .                            |
| row.names | Optional row names passed to <code>base::data.frame()</code> . |
| optional  | Ignored; present for S3 generic compatibility.                 |
| ...       | Ignored.                                                       |

### Value

A one-row data.frame with the flattened fields.

---

```
asMSKTracker
```

*Bridge a longitudinal container to a PhysioMSKNet tracker*

---

### Description

Builds the per-visit metric matrix and wraps it as an MSKLongitudinalTracker (its `activation_series`), so **PhysioMSKNet**'s `mskMinimalDetectableChange()` and `mskRecoveryTrajectoryFit()` can consume the container output directly (metrics play the role of "muscles", visits the role of time-points).

### Usage

```
asMSKTracker(long, metric_fn, metric_name = "metric")
```

### Arguments

|                        |                                        |
|------------------------|----------------------------------------|
| long                   | A <a href="#">PhysioLongitudinal</a> . |
| metric_fn, metric_name | As in <a href="#">changeScores</a> .   |

**Value**

An object of class "MSKLongitudinalTracker" carrying the metric-by-visit matrix.

**See Also**

[changeScores](#)

**Examples**

```
mk <- function(m) PhysioExperiment(  
  S4Vectors::SimpleList(raw = matrix(m, 10, 2)), samplingRate = 100)  
pl <- PhysioLongitudinal(baseline = mk(1), mid = mk(2), discharge = mk(3))  
tr <- asMSKTracker(pl, function(e) mean(SummarizedExperiment::assay(e, "raw")))  
dim(tr$activation_series)
```

---

asMultiAssayExperiment

*Coerce a PhysioLongitudinal to a MultiAssayExperiment*

---

**Description**

Requires the **MultiAssayExperiment** package. Each session becomes an experiment (named by session\_id); the design schema and subject metadata are stored in the result's metadata().

**Usage**

```
asMultiAssayExperiment(x)
```

**Arguments**

x                    A PhysioLongitudinal.

**Value**

A MultiAssayExperiment.

**See Also**

[PhysioLongitudinal](#)

---

|                    |                                 |
|--------------------|---------------------------------|
| assaySamplingRates | <i>Per-assay sampling rates</i> |
|--------------------|---------------------------------|

---

## Description

Returns the sampling rate associated with each assay of a `PhysioExperiment`. If no per-assay rates have been recorded, all assays are assumed to share the object's main `samplingRate`.

## Usage

```
assaySamplingRates(x)
```

## Arguments

|   |                                         |
|---|-----------------------------------------|
| x | A <code>PhysioExperiment</code> object. |
|---|-----------------------------------------|

## Details

For genuinely multi-rate acquisitions (streams of different length) prefer the `MultiRatePhysioExperiment` container; `assaySamplingRates()` is the lightweight per-assay tag for single-object, equal-length assays.

## Value

A named numeric vector of sampling rates (Hz), one per assay.

## References

Crochiere, R. E. & Rabiner, L. R. (1983). Multirate Digital Signal Processing. Prentice Hall.

## See Also

[setAssaySamplingRate](#), [MultiRatePhysioExperiment](#)

## Examples

```
pe <- PhysioExperiment(  
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)  
assaySamplingRates(pe)
```

blandAltman

*Bland-Altman Analysis for Method Agreement***Description**

Performs a Bland-Altman analysis comparing two measurement methods or two time points. Computes the bias (mean difference), limits of agreement, and confidence interval for the bias.

**Usage**

```
blandAltman(x, y, confidence = 0.95)
```

**Arguments**

**x** Numeric vector of measurements from method/time 1.  
**y** Numeric vector of measurements from method/time 2.  
**confidence** Numeric; confidence level for limits of agreement (default 0.95).

**Details**

The Bland-Altman method assesses agreement between two measurements by plotting their difference against their mean. The limits of agreement are:

$$LoA = \bar{d} \pm z \times SD_d$$

where  $\bar{d}$  is the mean difference (bias) and  $SD_d$  is the SD of differences.

**Value**

A list with components:

**bias** Mean difference (x - y)

**lower\_loa** Lower limit of agreement

**upper\_loa** Upper limit of agreement

**sd\_diff** Standard deviation of differences

**ci\_bias** Two-element vector with lower and upper CI for the bias

**References**

Bland JM, Altman DG (1986). Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, 327(8476), 307-310.

**See Also**

[icc\(\)](#) for intraclass correlation reliability analysis, [benchmarkAgreement\(\)](#) for benchmark validation agreement metrics, and [plotBlandAltman\(\)](#) in the *PhysioMoCap* package to visualize the agreement.

## Examples

```
set.seed(42)
method1 <- rnorm(30, mean = 50, sd = 10)
method2 <- method1 + rnorm(30, mean = 0, sd = 3)
result <- blandAltman(method1, method2)
result$bias
result$lower_loa
result$upper_loa
```

---

cbindPhysio

Combine PhysioExperiment objects by channels

---

## Description

Combines two PhysioExperiment objects by concatenating along the channel (column) dimension. Both objects must have the same number of time points and matching sampling rates.

## Usage

```
cbindPhysio(x, y)
```

## Arguments

|   |                                       |
|---|---------------------------------------|
| x | A PhysioExperiment object.            |
| y | A PhysioExperiment object to combine. |

## Value

A PhysioExperiment object with channels from both x and y, combined colData, and metadata from x.

## References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](https://doi.org/10.1038/nmeth.3252)

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

## See Also

[rbindPhysio](#) for combining along the time axis, [pickChannels](#) for selecting specific channels, [dropChannels](#) for removing channels

## Examples

```
pe1 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(200), nrow = 100, ncol = 2)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz")),
  samplingRate = 100
)
pe2 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(200), nrow = 100, ncol = 2)),
  colData = S4Vectors::DataFrame(label = c("Pz", "Oz")),
  samplingRate = 100
)

# Combine channels
pe_combined <- cbindPhysio(pe1, pe2)
nChannels(pe_combined) # 4
```

---

|              |                                                                |
|--------------|----------------------------------------------------------------|
| changeScores | <i>Per-visit change scores across a longitudinal container</i> |
|--------------|----------------------------------------------------------------|

---

## Description

Applies a user metric to every session of a [PhysioLongitudinal](#) and reports each visit's change from the baseline visit, optionally flagged against a minimal-detectable-change (MDC) threshold.

## Usage

```
changeScores(
  long,
  metric_fn,
  baseline = NULL,
  method = c("absolute", "percent", "z"),
  mdc = NULL,
  metric_name = "metric"
)
```

## Arguments

|           |                                                                                                                                                                                                             |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| long      | A <a href="#">PhysioLongitudinal</a> .                                                                                                                                                                      |
| metric_fn | A function taking one session (a <a href="#">PhysioExperiment</a> ) and returning a numeric scalar, or a named numeric vector of several metrics. It must return the same metric name(s) for every session. |
| baseline  | The baseline visit label or session id. If <code>NULL</code> (default) the visit labelled "baseline" is used, else the earliest visit.                                                                      |
| method    | Delta reported: "absolute" (value - baseline), "percent" (percent of baseline) or "z" (change divided by the metric's across-visit SD).                                                                     |

|             |                                                                                                                                                                                                                                                                                                                          |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| mdc         | Optional MDC threshold: a numeric scalar (applied to all metrics), a numeric vector named by metric, or a vector matching the metric count. The exceeds_mdc flag compares the <i>absolute</i> change to it. If NULL, exceeds_mdc is NA. See <a href="#">asMSKTracker</a> for obtaining an MDC from <b>PhysioMSKNet</b> . |
| metric_name | Name used for the metric when metric_fn returns an unnamed scalar (default "metric").                                                                                                                                                                                                                                    |

**Value**

A tidy DataFrame with columns subject, visit, days\_from\_baseline, metric, value, delta\_from\_baseline, exceeds\_mdc.

**References**

Beckerman, H., et al. (2001). Smallest real difference, a link between reproducibility and responsiveness. *Quality of Life Research*, 10(7), 571-578.

**See Also**

[asMSKTracker](#), [PhysioLongitudinal](#)

**Examples**

```
mk <- function(m) PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(m, 10, 2)), samplingRate = 100)
pl <- PhysioLongitudinal(
  baseline = mk(1), discharge = mk(3),
  design = S4Vectors::DataFrame(
    session_id = c("baseline", "discharge"),
    visit_label = c("baseline", "discharge"),
    days_from_baseline = c(0, 42)))
changeScores(pl, function(e) mean(SummarizedExperiment::assay(e, "raw")))
```

---

channelInfo

---

*Channel information management for PhysioExperiment*


---

**Description**

Functions for managing channel metadata including labels, types, units, and electrode positions. Get channel information

**Usage**

```
channelInfo(x)
```

**Arguments**

x                      A PhysioExperiment object.

**Details**

Returns channel metadata as a DataFrame. Channel information is stored in colData (columns = channels).

**Value**

A DataFrame with channel information.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[channelNames](#) for channel labels, [nChannels](#) for channel count, [setChannelTypes](#) for assigning channel types

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(
    label = c("Fz", "Cz", "Pz", "Oz"),
    type = rep("EEG", 4)
  ),
  samplingRate = 100
)

# Get channel information
channelInfo(pe)

# Get channel names
channelNames(pe)

# Get number of channels
nChannels(pe)
```

---

|               |                                |
|---------------|--------------------------------|
| channelInfo<- | <i>Set channel information</i> |
|---------------|--------------------------------|

---

**Description**

Updates channel metadata. Channel information is stored in colData (columns = channels).

**Usage**

```
channelInfo(x) <- value
```

**Arguments**

`x` A `PhysioExperiment` object.

`value` A `DataFrame` with channel information.

**Value**

Modified `PhysioExperiment` object.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[channelInfo](#) for reading channel metadata, [channelNames](#) for channel labels, [setChannelTypes](#) for assigning channel types

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)

# Set channel info
channelInfo(pe) <- S4Vectors::DataFrame(
  label = c("Fz", "Cz", "Pz", "Oz"),
  type = rep("EEG", 4)
)
```

---

channelNames

*Get channel names/labels*


---

**Description**

Get channel names/labels

**Usage**

```
channelNames(x)
```

**Arguments**

`x` A `PhysioExperiment` object.

**Value**

Character vector of channel names.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[channelInfo](#) for full channel metadata, [renameChannels](#) for renaming channels, [nChannels](#) for channel count

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)
channelNames(pe) # c("Fz", "Cz", "Pz", "Oz")
```

---

|                |                                 |
|----------------|---------------------------------|
| channelNames<- | <i>Set channel names/labels</i> |
|----------------|---------------------------------|

---

**Description**

Set channel names/labels

**Usage**

```
channelNames(x) <- value
```

**Arguments**

|       |                                    |
|-------|------------------------------------|
| x     | A PhysioExperiment object.         |
| value | Character vector of channel names. |

**Value**

Modified PhysioExperiment object.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[channelNames](#) for reading channel labels, [renameChannels](#) for renaming specific channels, [channelInfo](#) for full channel metadata

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)
channelNames(pe) <- c("Fz", "Cz", "Pz", "Oz")
channelNames(pe)
```

---

checkNA

---

*Check for NA values in assay data*


---

**Description**

Validates assay data for NA values and reports statistics.

**Usage**

```
checkNA(x, action = c("warn", "error", "none"))
```

**Arguments**

**x** A PhysioExperiment object or numeric array.

**action** Action to take: "warn" (default), "error", or "none".

**Value**

Invisibly returns a list with NA statistics.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[hasNA](#) for quick NA presence check, [naSummary](#) for per-assay NA statistics, [handleNA](#) for NA replacement strategies

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(c(1, NA, 3, 4), nrow = 2)),
  samplingRate = 100
)
checkNA(pe)
```

---

circularLinearCorrelation

*Mardia's circular-linear correlation*


---

**Description**

The association between an angular variable (e.g. a coupling angle or relative phase) and a linear one (e.g. speed, force, a clinical score). Symmetric under rotation of the angle.

**Usage**

```
circularLinearCorrelation(theta, x, units = c("degrees", "radians"))
```

**Arguments**

|       |                                                 |
|-------|-------------------------------------------------|
| theta | Numeric angular variable.                       |
| x     | Numeric linear variable (same length as theta). |
| units | "degrees" (default) or "radians" for theta.     |

**Value**

an htest: the circular-linear correlation  $r$  (estimate), the test statistic  $n * r^2 \sim \text{chi-square}(2)$  and its  $p$ .value.

**References**

Mardia KV (1976); Zar JH (1999) eq. 27.19.

**See Also**

[circularSummary\(\)](#)

**Examples**

```
set.seed(3); th <- runif(60, 0, 360); y <- cos(th * pi / 180) + rnorm(60, 0, 0.3)
circularLinearCorrelation(th, y)$estimate           # strong association
```

---

|                 |                                           |
|-----------------|-------------------------------------------|
| circularSummary | <i>Circular mean direction and spread</i> |
|-----------------|-------------------------------------------|

---

**Description**

The mean direction of a set of angles and its circular spread, computed via the resultant vector (so the wrap-around at 0/360 is handled correctly).

**Usage**

```
circularSummary(angles, units = c("degrees", "radians"))
```

**Arguments**

|        |                                   |
|--------|-----------------------------------|
| angles | Numeric vector of angles.         |
| units  | "degrees" (default) or "radians". |

**Value**

a circular\_summary list: mean (mean direction), R (resultant length), rbar (mean resultant length in [0,1]), variance (1 - rbar), sd (circular SD), kappa (von Mises concentration), all in the input units where applicable.

**References**

Zar JH (1999) Biostatistical Analysis, ch. 26-27.

**See Also**

[rayleighTest\(\)](#), [circularLinearCorrelation\(\)](#)

**Examples**

```
circularSummary(c(10, 20, 350, 355))$mean      # ~ 0, not ~180
```

---

|         |                              |
|---------|------------------------------|
| cohensD | <i>Cohen's d Effect Size</i> |
|---------|------------------------------|

---

**Description**

Computes Cohen's d effect size for comparing two groups or conditions, with confidence intervals and qualitative interpretation.

**Usage**

```
cohensD(x, y, paired = FALSE, pooled = TRUE)
```

**Arguments**

|                     |                                                                                                                                                                                                  |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>      | Numeric vector for group 1 (or condition 1 if paired).                                                                                                                                           |
| <code>y</code>      | Numeric vector for group 2 (or condition 2 if paired).                                                                                                                                           |
| <code>paired</code> | Logical; if TRUE, computes effect size for paired data using the SD of differences as the denominator.                                                                                           |
| <code>pooled</code> | Logical; if TRUE (default), uses pooled SD as denominator. If FALSE, uses the SD of <code>y</code> (Glass's delta, treating <code>y</code> as control). Ignored when <code>paired</code> = TRUE. |

**Details**

For independent groups with `pooled` = TRUE, the pooled SD is:

$$SD_{pooled} = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

For paired data, the denominator is the SD of the within-pair differences.

Interpretation thresholds follow Cohen (1988):

- $|d| < 0.2$ : negligible
- $0.2 \leq |d| < 0.5$ : small
- $0.5 \leq |d| < 0.8$ : medium
- $|d| \geq 0.8$ : large

**Value**

A list with components:

**d** Cohen's d value

**ci\_lower** Lower bound of 95 percent confidence interval

**ci\_upper** Upper bound of 95 percent confidence interval

**interpretation** Qualitative label: "negligible", "small", "medium", or "large"

**References**

Cohen J (1988). Statistical Power Analysis for the Behavioral Sciences. Lawrence Erlbaum Associates.

**See Also**

[etaSquared\(\)](#) for ANOVA-based effect sizes, [plotEffectSizeForest\(\)](#) for forest plot visualization of effect sizes.

Examples

```
set.seed(42)
x <- rnorm(30, mean = 10, sd = 2)
y <- rnorm(30, mean = 8, sd = 2)
result <- cohensD(x, y)
result$d
result$interpretation
```

---

|            |                                                       |
|------------|-------------------------------------------------------|
| cohortData | <i>Subject-level metadata table of a PhysioCohort</i> |
|------------|-------------------------------------------------------|

---

Description

Subject-level metadata table of a PhysioCohort

Usage

```
cohortData(x)

## S4 method for signature 'PhysioCohort'
cohortData(x)

cohortData(x) <- value

## S4 replacement method for signature 'PhysioCohort'
cohortData(x) <- value
```

Arguments

- x                   A PhysioCohort.
- value               A subject-level DataFrame (setter); one row per subject.

Value

cohortData() the subject-level DataFrame.

---

|              |                                                            |
|--------------|------------------------------------------------------------|
| cohortDesign | <i>Long design table of a cohort (subjects x sessions)</i> |
|--------------|------------------------------------------------------------|

---

Description

Flattens the whole cohort into one tidy data.frame with a row per (subject, session): subject\_id, any subject-level colData (e.g. group), and the session's session\_id, visit\_label, days\_from\_baseline and condition. This is the iteration unit for pipelines, longitudinal models and trial analyses.

**Usage**

```
cohortDesign(x)
```

**Arguments**

x                    A `PhysioCohort`.

**Value**

A `data.frame`, one row per subject-session.

**See Also**

[PhysioCohort](#), [cohortData](#)

---

|              |                                |
|--------------|--------------------------------|
| cohortTrials | <i>Every trial in a cohort</i> |
|--------------|--------------------------------|

---

**Description**

Lists the trials of every subject, carrying both the subject and the session. The same `trial_id` under two subjects has to stay distinguishable, which is why the subject travels with the row rather than being inferred.

**Usage**

```
cohortTrials(x)
```

**Arguments**

x                    A `PhysioCohort`.

**Value**

A data frame of trials with subject and session columns.

**See Also**

[subjectTrials](#)

---

|             |                         |
|-------------|-------------------------|
| commonClock | <i>The shared clock</i> |
|-------------|-------------------------|

---

**Description**

The shared clock

**Usage**

```
commonClock(x)
```

**Arguments**

x                    A MultiPhysioExperiment.

**Value**

The clock list (t0, reference\_rate, offsets, offsets\_assumed, schema).

---

|              |                                        |
|--------------|----------------------------------------|
| defaultAssay | <i>Retrieve the default assay name</i> |
|--------------|----------------------------------------|

---

**Description**

Returns the name of the first assay in the PhysioExperiment object, which is treated as the default assay for operations that do not specify an assay explicitly.

**Usage**

```
defaultAssay(x)
```

**Arguments**

x                    A PhysioExperiment instance.

**Value**

Character scalar naming the first assay, or NA\_character\_ when no assays are present.

**References**

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:10.1038/nmeth.3252

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

[PhysioExperiment](#) for the constructor, [samplingRate](#) for the sampling rate accessor, [timeIndex](#) for time point vector

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(100), nrow = 10), filtered = matrix(0, 10, 10)),
  samplingRate = 250
)
defaultAssay(pe)
```

deprecate\_physio

*Emit a Physio-ecosystem deprecation warning or error***Description****[Stable]****Usage**

```
deprecate_physio(
  when,
  what,
  with = NULL,
  details = NULL,
  severity = c("warn", "stop"),
  id = NULL
)
```

**Arguments**

|          |                                                                                                                                                                 |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| when     | Version string in which the deprecation was introduced, e.g. "0.3.0".                                                                                           |
| what     | String naming the deprecated function or argument, e.g. "old_fn()" or "old_fn(arg)". Prefix with the package ("PhysioECG:old_fn()") to attribute it explicitly. |
| with     | Optional string naming the replacement, e.g. "new_fn()".                                                                                                        |
| details  | Optional character vector of extra guidance shown to the user.                                                                                                  |
| severity | "warn" (default) to emit a deprecation warning, or "stop" to raise a defunct error.                                                                             |
| id       | Optional deprecation id used to de-duplicate warnings within a session; ignored when severity = "stop".                                                         |

**Details**

A thin, ecosystem-wide wrapper over [lifecycle::deprecate\\_warn\(\)](#) and [lifecycle::deprecate\\_stop\(\)](#) so every Physio package signals deprecations the same way. Use severity = "warn" during the deprecation window and switch to severity = "stop" once the item is defunct. See the ecosystem `DEPRECATION.md` policy for the lifecycle stages and the minimum window.

**Value**

Called for its side effect (a warning or an error); invisibly returns NULL.

**See Also**

[lifecycle::deprecate\\_warn\(\)](#), [lifecycle::deprecate\\_stop\(\)](#)

**Examples**

```
old_fn <- function(x) {
  deprecate_physio("0.3.0", "old_fn()", with = "new_fn()")
  x
}
# Warnings are shown once per session by default; force them for testing with
# options(lifecycle_verbosity = "warning").
```

---

design,PhysioLongitudinal-method

*The design schema of a PhysioLongitudinal*

---

**Description**

A method for the **BiocGenerics** design generic, so it composes cleanly with the rest of the Bioconductor ecosystem.

**Usage**

```
## S4 method for signature 'PhysioLongitudinal'
design(object, ...)

## S4 replacement method for signature 'PhysioLongitudinal'
design(object, ...) <- value
```

**Arguments**

|        |                              |
|--------|------------------------------|
| object | A PhysioLongitudinal.        |
| ...    | Ignored.                     |
| value  | A design DataFrame (setter). |

**Value**

design() the design DataFrame.

---

|            |                                         |
|------------|-----------------------------------------|
| detectGaps | <i>Gaps in a series of sample times</i> |
|------------|-----------------------------------------|

---

### Description

A step between consecutive sample times larger than `gap_factor` times the nominal period counts as a gap. The factor is an adopted criterion – it separates acquisition jitter from a dropped sample; it does not prove that no gap exists – and it is the same criterion **PhysioCrossModal** uses to refuse interpolating across one.

### Usage

```
detectGaps(times, rate, gap_factor = 1.5, window = NULL, tolerance = NULL)
```

### Arguments

|                         |                                                                                                   |
|-------------------------|---------------------------------------------------------------------------------------------------|
| <code>times</code>      | Sample times in seconds, ascending.                                                               |
| <code>rate</code>       | Nominal sampling rate in Hz.                                                                      |
| <code>gap_factor</code> | Multiple of the nominal period above which a step counts as a gap.                                |
| <code>window</code>     | Optional <code>c(start, end)</code> : report only gaps overlapping it.                            |
| <code>tolerance</code>  | Overlap tolerance; defaults to the tolerance <a href="#">timeWindow</a> uses for the same bounds. |

### Details

Detection runs on the whole series **before** any window is applied, and the window then selects which gaps to report. Selecting samples first and differencing afterwards loses exactly the gaps that matter most: a dropout straddling a window's edge has one of its two bounding samples outside the window, so the step across it is never formed and the window looks intact.

A gap that only touches an endpoint of the window is not in it. On a closed interval the two share a single instant, and the recording does cover that instant – the same reading of "touching is not overlapping" that [trialIntervals](#) uses.

### Value

A data frame of `start` and `end` – the sample times the gap lies between, in the original series, not clipped to window.

### See Also

[streamCoverage](#), [timeWindow](#)

### Examples

```
detectGaps(c(0, 0.1, 0.2, 0.4, 0.5), rate = 10)
# the same gap, found from a window whose edge cuts through it
detectGaps(c(0, 0.1, 0.2, 0.4, 0.5), rate = 10, window = c(0.25, 0.45))
```

---

`dim,PhysioExperiment-method`*Dim method for PhysioExperiment*

---

## Description

Returns the dimensions of the default assay.

## Usage

```
## S4 method for signature 'PhysioExperiment'  
dim(x)
```

## Arguments

`x`                      A PhysioExperiment object.

## Value

An integer vector of dimensions (time points by channels by samples for 3D data), or NULL if no assays are present.

## References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](https://doi.org/10.1038/nmeth.3252)

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

## See Also

[length,PhysioExperiment-method](#) for time point count, [nChannels](#) for channel count, [defaultAssay](#) for the assay being queried

## Examples

```
pe <- PhysioExperiment(  
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),  
  samplingRate = 100  
)  
dim(pe) # 100 4
```

---

|              |                      |
|--------------|----------------------|
| dropChannels | <i>Drop channels</i> |
|--------------|----------------------|

---

## Description

Creates a new `PhysioExperiment` without specified channels.

## Usage

```
dropChannels(x, channels)
```

## Arguments

|                       |                                                         |
|-----------------------|---------------------------------------------------------|
| <code>x</code>        | A <code>PhysioExperiment</code> object.                 |
| <code>channels</code> | Integer indices or character names of channels to drop. |

## Value

A new `PhysioExperiment` without specified channels.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[pickChannels](#) for keeping specific channels, [getChannelsByType](#) for finding channels by type, [nChannels](#) for channel count

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)

# Drop by index
pe_dropped <- dropChannels(pe, 1)
nChannels(pe_dropped) # 3

# Drop by name
pe_dropped <- dropChannels(pe, c("Fz", "Oz"))
```

---

|          |                            |
|----------|----------------------------|
| duration | <i>Get signal duration</i> |
|----------|----------------------------|

---

## Description

Computes the total duration of the signal in seconds from the number of time points and the sampling rate.

## Usage

```
duration(x)
```

## Arguments

`x`                      A `PhysioExperiment` object.

## Value

Numeric scalar giving the signal duration in seconds, or `NA_real_` if the sampling rate is not set.

## References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](https://doi.org/10.1038/nmeth.3252)

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

## See Also

[samplingRate](#) for the sampling rate, [length,PhysioExperiment-method](#) for time point count, [timeIndex](#) for the time vector, [extractWindow](#) for time-based subsetting

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 1000, ncol = 4)),
  samplingRate = 100
)
duration(pe) # 10 seconds
```

---

elapsedBetweenTrials    *Elapsed time between two trials*

---

### Description

Returns the seconds between the starts of two independently recorded trials.

### Usage

```
elapsedBetweenTrials(x, from, to)
```

### Arguments

|          |                          |
|----------|--------------------------|
| x        | A MultiPhysioExperiment. |
| from, to | Trial identifiers.       |

### Details

This is refused unless the recordings are actually placed relative to one another. Declaring `time_correspondence = "measured"` is not enough: the rows must also carry a reference and an offset, and both trials must be measured against the *same* reference, since offsets on different references are not comparable. "order\_only" and "unknown" are refused outright - the order in which recordings were made is not a duration, and turning one into the other would fabricate a measurement.

An absent absolute origin (`clock$t0` of NA) is a different matter and does not block this: what is needed is the correspondence between recordings, not an absolute time.

### Value

Numeric seconds from from to to.

### See Also

[recordingMap](#)

---

etaSquared    *Eta-Squared Effect Size*

---

### Description

Computes eta-squared, partial eta-squared, and omega-squared from a one-way between-subjects comparison.

### Usage

```
etaSquared(x, groups)
```

**Arguments**

|                     |                                                 |
|---------------------|-------------------------------------------------|
| <code>x</code>      | Numeric vector of values.                       |
| <code>groups</code> | Factor or character vector of group membership. |

**Details**

Eta-squared is the proportion of total variance explained by group membership:

$$\eta^2 = \frac{SS_{between}}{SS_{total}}$$

Omega-squared provides a less biased estimate:

$$\omega^2 = \frac{SS_{between} - df_{between} \cdot MS_{within}}{SS_{total} + MS_{within}}$$

For one-way designs, partial eta-squared equals eta-squared.

**Value**

A list with components:

**eta\_sq** Eta-squared ( $SS_{between} / SS_{total}$ )

**partial\_eta\_sq** Partial eta-squared ( $SS_{between} / (SS_{between} + SS_{within})$ )

**omega\_sq** Omega-squared (bias-corrected effect size)

**References**

Cohen J (1988). "Statistical Power Analysis for the Behavioral Sciences." Lawrence Erlbaum Associates.

**See Also**

[cohensD\(\)](#) for pairwise effect sizes, [plotEffectSizeForest\(\)](#) for forest plot visualization.

**Examples**

```
set.seed(42)
x <- c(rnorm(20, 10, 2), rnorm(20, 12, 2), rnorm(20, 14, 2))
groups <- rep(c("A", "B", "C"), each = 20)
result <- etaSquared(x, groups)
result$eta_sq
```

---

|            |                                              |
|------------|----------------------------------------------|
| eventQuery | Create an EventQuery from a PhysioExperiment |
|------------|----------------------------------------------|

---

**Description**

Create an EventQuery from a PhysioExperiment

**Usage**

```
eventQuery(x)
```

**Arguments**

x                      A PhysioExperiment object

**Value**

An EventQuery object

**Examples**

```
pe <- PhysioExperiment(assays = list(raw = matrix(rnorm(400), 100, 4)),
                        samplingRate = 100)
pe <- addEvents(pe, onset = c(1, 2, 3), type = "stimulus")
q <- eventQuery(pe)
```

---

|                  |                                                 |
|------------------|-------------------------------------------------|
| EventQuery-class | EventQuery class for composable event filtering |
|------------------|-------------------------------------------------|

---

**Description**

EventQuery class for composable event filtering

**Slots**

source   PhysioExperiment object  
filters   List of filter functions to apply  
resolved   Cached resolved events (NULL if not yet resolved)

---

extractTrialFeatures    *Extract scalar trial features across sessions and participants*

---

### Description

Applies named feature functions to each selected trial and channel. Accepts the same recording and hierarchy inputs as `aggregateTrialWaveforms()`. Trial selection, time-based sample selection, channel matching and assembly of the feature table are performed together. Pass the returned object to `aggregatePhysioFeatures()` to obtain session, subject or cohort summaries.

### Usage

```
extractTrialFeatures(
  x,
  stream,
  FUN,
  assay = "raw",
  channels = NULL,
  strata = NULL,
  include = "complete",
  gap_factor = 1.5,
  ...
)
```

### Arguments

|                         |                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>          | A <code>MultiPhysioExperiment</code> , uniquely named list of these, <code>PhysioLongitudinal</code> , or <code>PhysioCohort</code> . Each recording must describe one subject and session, with unique full trial keys across recordings.                                                                                                          |
| <code>stream</code>     | Name of the stream to measure.                                                                                                                                                                                                                                                                                                                      |
| <code>FUN</code>        | Named, nonempty list of feature functions, for example <code>list(amplitude = mean)</code> . Each receives the numeric sample vector for one trial and channel, followed by <code>...</code> , and must return one numeric value (NA is allowed). Functions are responsible for their scientific definition and treatment of missing signal values. |
| <code>assay</code>      | Assay name, default "raw".                                                                                                                                                                                                                                                                                                                          |
| <code>channels</code>   | Channel names. Defaults to all channels, requiring the same names across selected recordings; column order may differ.                                                                                                                                                                                                                              |
| <code>strata</code>     | Trial-table columns to preserve as separate groups in subsequent aggregation. Channels are always kept separate.                                                                                                                                                                                                                                    |
| <code>include</code>    | Trial states to process: "complete" (default) or explicitly "partial", or both. Classification uses every stream described for the trial, as in <code>aggregateTrials()</code> .                                                                                                                                                                    |
| <code>gap_factor</code> | Gap criterion passed to <code>streamCoverage()</code> .                                                                                                                                                                                                                                                                                             |
| <code>...</code>        | Additional fixed arguments passed to every feature function.                                                                                                                                                                                                                                                                                        |

## Details

Sample membership uses the measured time index and the same boundary tolerance as `timeWindow()`. Samples are neither interpolated nor resampled. For explicitly included partial trials, features describe the available samples inside the retained interval; these can differ in duration between trials. Included trials with unusable retained gap evidence are refused. Trials lacking the requested stream or interval are recorded as excluded. A recording without a trial table, ambiguous identity, an incompatible assay/channel set, an empty selected interval, or a failed feature function causes an error.

Temporal coverage does not establish signal quality. Missing values are passed to the feature function; returned NA values remain in the table and require explicit handling when aggregating. Infinite input samples and infinite feature results are rejected. Feature-function text and supplied arguments are recorded, but arbitrary external state used by a callback is not captured; use self-contained functions and explicit arguments.

Correspondence is established by channel names. Comparable units and feature definitions remain the caller's responsibility. The returned checksum detects modification of the extraction result before aggregation, and references are preserved when continuing aggregation through further levels.

## Value

A `physio_trial_features` list containing data (one row per selected trial and channel), `trials` (all trial selections and reasons), `source_map` (one entry per feature-table row: recording, interval, original sample indices and sample times), `source_hashes` (full input recording hashes), `features`, `strata`, `settings` and `checksum`. Input containers are not modified.

## See Also

[aggregatePhysioFeatures](#), [aggregateTrials](#), [aggregateTrialWaveforms](#)

## Examples

```
pe <- PhysioExperiment(assays = list(raw = matrix(0:30, ncol = 1,
  dimnames = list(NULL, "signal"))), samplingRate = 10)
m <- MultiPhysioExperiment(sensor = pe)
trials(m) <- data.frame(subject_id = "P1", session_id = "S1",
  trial_id = c("T1", "T2"))
trialIntervals(m) <- data.frame(trials(m)[1:3], recording_id = "R1",
  stream = "sensor", start = c(0, 2), end = c(1, 3))
f <- extractTrialFeatures(m, "sensor", FUN = list(amplitude = mean))
f$data
aggregatePhysioFeatures(f, "session")$data # mean(c(5, 25)) = 15
```

---

extractWindow

*Extract time window*

---

## Description

Extracts a time window from the signal based on start and end times in seconds.

**Usage**

```
extractWindow(x, tmin, tmax)
```

**Arguments**

|      |                                         |
|------|-----------------------------------------|
| x    | A <code>PhysioExperiment</code> object. |
| tmin | Start time in seconds.                  |
| tmax | End time in seconds.                    |

**Value**

A `PhysioExperiment` object containing only the samples within the specified time window, with preserved channel and event metadata.

**References**

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](https://doi.org/10.1038/nmeth.3252)

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

[duration](#) for total signal duration, [timeIndex](#) for the time vector, `[]` for index-based subsetting, [timeToSamples](#) for converting times to indices

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 1000, ncol = 4)),
  samplingRate = 100
)

# Extract 2 to 5 seconds
pe_window <- extractWindow(pe, tmin = 2, tmax = 5)
duration(pe_window) # approximately 3 seconds
```

---

fillEdgeNA

---

*Fill NA values at edges*


---

**Description**

Fills NA values at the beginning and end of a signal that may result from filtering operations.

**Usage**

```
fillEdgeNA(x, method = c("extend", "zero"))
```

**Arguments**

|        |                                                                                 |
|--------|---------------------------------------------------------------------------------|
| x      | Numeric vector.                                                                 |
| method | Fill method: "extend" (extend nearest valid value) or "zero" (fill with zeros). |

**Value**

Vector with edge NA values filled.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[handleNA](#) for general NA handling strategies, [replaceNA](#) for NA handling in PhysioExperiment assays

**Examples**

```
x <- c(NA, NA, 1, 2, 3, NA, NA)
fillEdgeNA(x, method = "extend")
```

---

filterType

*Filter events by type*


---

**Description**

Filter events by type

**Usage**

```
filterType(q, types)
```

**Arguments**

|       |                                         |
|-------|-----------------------------------------|
| q     | An EventQuery object                    |
| types | Character vector of event types to keep |

**Value**

Modified EventQuery

Examples

```
pe <- PhysioExperiment(assays = list(raw = matrix(rnorm(400), 100, 4)),
                           samplingRate = 100)
pe <- addEvents(pe, onset = c(1, 2, 3),
                type = c("stimulus", "response", "stimulus"))
q <- filterType(eventQuery(pe), "stimulus")
resolveQuery(q)
```

---

|             |                               |
|-------------|-------------------------------|
| filterValue | <i>Filter events by value</i> |
|-------------|-------------------------------|

---

Description

Filter events by value

Usage

```
filterValue(q, values)
```

Arguments

- q                    An EventQuery object
- values             Character vector of event values to keep

Value

Modified EventQuery

Examples

```
pe <- PhysioExperiment(assays = list(raw = matrix(rnorm(400), 100, 4)),
                           samplingRate = 100)
pe <- addEvents(pe, onset = c(1, 2, 3), type = "stimulus",
                value = c("target", "distractor", "target"))
q <- filterValue(eventQuery(pe), "target")
resolveQuery(q)
```

---

|                        |                                                     |
|------------------------|-----------------------------------------------------|
| format.PhysioBiomarker | <i>Format a PhysioBiomarker as a compact string</i> |
|------------------------|-----------------------------------------------------|

---

**Description**

Format a PhysioBiomarker as a compact string

**Usage**

```
## S3 method for class 'PhysioBiomarker'  
format(x, ...)
```

**Arguments**

- x                   A [PhysioBiomarker](#).
- ...                 Ignored.

**Value**

A length-1 character string "name = value unit [ci]".

---

|      |                                                        |
|------|--------------------------------------------------------|
| fPCA | <i>Functional Data Analysis (FDA) for Biomechanics</i> |
|------|--------------------------------------------------------|

---

**Description**

Functions for functional data analysis including functional PCA (fPCA), functional regression, and curve registration. These methods treat biomechanical waveforms as continuous functions. Functional Principal Component Analysis (fPCA)

**Usage**

```
fPCA(x, n_components = 5, smooth = FALSE, smooth_param = 10)
```

**Arguments**

- x                   A PhysioExperiment object or matrix (time x observations).
- n\_components       Number of principal components to retain.
- smooth             Logical; if TRUE, smooths the data before analysis.
- smooth\_param       Smoothing parameter (higher = smoother).

## Details

Performs functional PCA on waveform data to identify the main modes of variation in movement patterns.

fPCA decomposes waveform variability into orthogonal modes. In gait analysis, PC1 often represents overall amplitude, PC2 timing/phase shifts, and subsequent PCs capture more subtle shape variations.

## Value

A list of class "fpca\_result" containing:

|                     |                                                            |
|---------------------|------------------------------------------------------------|
| scores              | PC scores for each observation (observations x components) |
| loadings            | PC loadings/eigenfunctions (time x components)             |
| variance_explained  | Proportion of variance explained by each PC                |
| cumulative_variance | Cumulative variance explained                              |
| mean_function       | Mean waveform across observations                          |

## References

Ramsay JO, Silverman BW (2005). "Functional Data Analysis." 2nd ed. Springer.

## See Also

[reconstructFPCA\(\)](#) for waveform reconstruction from fPCA results, [plotFPCA\(\)](#) (in *PhysioMoCap*) for visualization of fPCA results, [registerCurves\(\)](#) for separating phase and amplitude variation.

## Examples

```
# Simulate gait angle data (100 time points x 30 subjects)
set.seed(123)
t <- seq(0, 100, length.out = 100)
base_curve <- sin(2 * pi * t / 100) * 30

# Add subject variability
data <- sapply(1:30, function(i) {
  amplitude <- rnorm(1, 1, 0.2)
  phase <- rnorm(1, 0, 5)
  base_curve * amplitude + rnorm(100, 0, 2)
})

pe <- PhysioExperiment(assays = list(values = data), samplingRate = 100)
fpca_result <- fPCA(pe, n_components = 4)
```

---

|                   |                             |
|-------------------|-----------------------------|
| getChannelsByType | <i>Get channels by type</i> |
|-------------------|-----------------------------|

---

## Description

Returns indices of channels matching specified types.

## Usage

```
getChannelsByType(x, types)
```

## Arguments

|       |                                             |
|-------|---------------------------------------------|
| x     | A <code>PhysioExperiment</code> object.     |
| types | Character vector of channel types to match. |

## Value

Integer vector of matching channel indices.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[setChannelTypes](#) for assigning channel types, [pickChannels](#) for subsetting by channel, [dropChannels](#) for removing channels

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(
    label = c("Fz", "EOG1", "EMG1", "Oz"),
    type = c("EEG", "EOG", "EMG", "EEG")
  ),
  samplingRate = 100
)

# Get EEG channels
eeg_idx <- getChannelsByType(pe, "EEG") # c(1, 4)
```

---

getElectrodePositions *Get electrode positions*

---

## Description

Get electrode positions

## Usage

```
getElectrodePositions(x)
```

## Arguments

x                      A `PhysioExperiment` object.

## Value

A `data.frame` with x, y, z columns or `NULL` if not set.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[setElectrodePositions](#) for setting positions, [applyMontage](#) for standard electrode layouts, [channelNames](#) for channel labels

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(300), nrow = 100, ncol = 3)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz")),
  samplingRate = 100
)
pe <- applyMontage(pe, "10-20")
getElectrodePositions(pe)
```

---

`getEvents`*Get events from a `PhysioExperiment` object*

---

### Description

Get events from a `PhysioExperiment` object

### Usage

```
getEvents(x, type = NULL)
```

### Arguments

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <code>x</code>    | A <code>PhysioExperiment</code> object.             |
| <code>type</code> | Optional character vector of event types to filter. |

### Value

A `PhysioEvents` object or `DataFrame` of events.

### References

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

### See Also

[setEvents](#) for attaching events, [addEvents](#) for appending events, [removeEvents](#) for removing events, [nEvents](#) for event count

### Examples

```
# Create PhysioExperiment with events
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 100)),
  samplingRate = 100
)
events <- PhysioEvents(
  onset = c(1, 2, 3),
  type = c("stimulus", "response", "stimulus")
)
pe <- setEvents(pe, events)

# Get all events
getEvents(pe)
```

```
# Get only stimulus events
getEvents(pe, type = "stimulus")
```

---

|              |                                |
|--------------|--------------------------------|
| getReference | <i>Get reference electrode</i> |
|--------------|--------------------------------|

---

**Description**

Get reference electrode

**Usage**

```
getReference(x)
```

**Arguments**

x                      A `PhysioExperiment` object.

**Value**

Character string of reference electrode or `NULL`.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[setReference](#) for setting the reference, [channelInfo](#) for full channel metadata

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)
pe <- setReference(pe, "Cz")
getReference(pe) # "Cz"
```

---

|          |                                        |
|----------|----------------------------------------|
| handleNA | <i>Handle NA values in signal data</i> |
|----------|----------------------------------------|

---

**Description**

Provides various strategies for handling NA values in signal data.

**Usage**

```
handleNA(  
  x,  
  method = c("interpolate", "omit", "zero", "mean", "locf", "none"),  
  ...  
)
```

**Arguments**

|        |                                                                                                                                                                                         |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x      | Numeric vector or matrix.                                                                                                                                                               |
| method | Method for handling NA: "omit" (remove), "interpolate" (linear), "zero" (replace with 0), "mean" (replace with mean), "locf" (last observation carried forward), or "none" (no action). |
| ...    | Additional arguments passed to interpolation methods.                                                                                                                                   |

**Value**

Data with NA values handled according to the specified method.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[replaceNA](#) for handling NA in PhysioExperiment assays, [fillEdgeNA](#) for edge-specific NA filling, [checkNA](#) for NA validation

**Examples**

```
x <- c(1, NA, 3, NA, 5)  
  
# Linear interpolation  
handleNA(x, method = "interpolate")  
  
# Replace with mean  
handleNA(x, method = "mean")
```

```
# Last observation carried forward
handleNA(x, method = "locf")
```

---

harmonize

*Harmonize channels, reference and montage across sessions*


---

## Description

Convenience wrapper that runs [harmonizeChannels](#), [harmonizeReference](#) and [harmonizeMontage](#) in turn. Pass `ref = NULL` or `system = NULL` to skip a step.

## Usage

```
harmonize(
  long,
  target_labels = NULL,
  rename = NULL,
  ref = "average",
  system = "10-20"
)
```

## Arguments

`long`                    A `PhysioLongitudinal`.

`target_labels`, `rename`                    As in [harmonizeChannels](#).

`ref`                    Common reference label, or `NULL` to skip (default "average").

`system`                    Montage system, or `NULL` to skip (default "10-20").

## Value

A fully harmonized `PhysioLongitudinal`.

## See Also

[harmonizeReport](#)

---

|                   |                                           |
|-------------------|-------------------------------------------|
| harmonizeChannels | <i>Harmonize channels across sessions</i> |
|-------------------|-------------------------------------------|

---

## Description

Intersects and reorders the channels of every session of a `PhysioLongitudinal` so they share one common channel set in identical order. Channels absent from any session are dropped; an optional rename map first unifies differing label conventions.

## Usage

```
harmonizeChannels(long, target_labels = NULL, rename = NULL)
```

## Arguments

|                            |                                                                                                                                                                                                           |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>long</code>          | A <code>PhysioLongitudinal</code> .                                                                                                                                                                       |
| <code>target_labels</code> | Optional character vector giving the desired channel set and order. Restricted to channels common to all sessions. If <code>NULL</code> , the common intersection (in the first session's order) is used. |
| <code>rename</code>        | Optional named character vector <code>c(old = new, ...)</code> applied to every session before intersecting (e.g. <code>c(T7 = "T3")</code> ).                                                            |

## Value

The `PhysioLongitudinal` with harmonized sessions; each session records the kept/dropped/renamed channels and a provenance activity.

## See Also

[harmonize](#), [harmonizeReport](#), [pickChannels](#)

## Examples

```
mk <- function(labs) PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(50 * length(labs)), 50, length(labs))),
  colData = S4Vectors::DataFrame(label = labs), samplingRate = 100)
pl <- PhysioLongitudinal(
  baseline = mk(c("Fz", "Cz", "Pz", "Oz")),
  discharge = mk(c("Cz", "Fz", "Pz")))
h <- harmonizeChannels(pl)
channelNames(session(h, "baseline"))
```

---

|                  |                                              |
|------------------|----------------------------------------------|
| harmonizeMontage | <i>Harmonize the montage across sessions</i> |
|------------------|----------------------------------------------|

---

**Description**

Applies a consistent electrode montage to every session (building on [applyMontage](#)).

**Usage**

```
harmonizeMontage(long, system = c("10-20", "10-10", "10-5"))
```

**Arguments**

|        |                                                  |
|--------|--------------------------------------------------|
| long   | A <code>PhysioLongitudinal</code> .              |
| system | Montage system: one of "10-20", "10-10", "10-5". |

**Value**

The `PhysioLongitudinal` with a common montage and a provenance activity on each session.

**See Also**

[harmonize](#), [applyMontage](#)

---

|                    |                                                |
|--------------------|------------------------------------------------|
| harmonizeReference | <i>Harmonize the reference across sessions</i> |
|--------------------|------------------------------------------------|

---

**Description**

Records a common reference on every session (building on [setReference](#)), so that [getReference](#) is identical across sessions.

**Usage**

```
harmonizeReference(long, ref)
```

**Arguments**

|      |                                                   |
|------|---------------------------------------------------|
| long | A <code>PhysioLongitudinal</code> .               |
| ref  | Character reference label (e.g. "average", "Cz"). |

**Value**

The `PhysioLongitudinal` with a common reference and a provenance activity on each session.

**See Also**

[harmonize](#), [setReference](#)

---

|                 |                                                                           |
|-----------------|---------------------------------------------------------------------------|
| harmonizeReport | <i>Report the channels dropped / renamed per session by harmonization</i> |
|-----------------|---------------------------------------------------------------------------|

---

**Description**

Report the channels dropped / renamed per session by harmonization

**Usage**

```
harmonizeReport(long)
```

**Arguments**

|      |                                  |
|------|----------------------------------|
| long | A harmonized PhysioLongitudinal. |
|------|----------------------------------|

**Value**

A named list (one entry per session) with kept, dropped and renamed channels.

**See Also**

[harmonizeChannels](#)

---

|                  |                                                   |
|------------------|---------------------------------------------------|
| hasMeasuredTimes | <i>Does a stream carry measured sample times?</i> |
|------------------|---------------------------------------------------|

---

**Description**

Regular streams put sample  $i$  at offset  $+ (i - 1) / \text{rate}$ . A stream that carries measured timestamps - because the acquisition reported them, or because it has gaps or drift - is not on that grid, and operations that can only work with a regular grid must say so rather than approximate.

**Usage**

```
hasMeasuredTimes(x, stream = NULL)
```

**Arguments**

|        |                                       |
|--------|---------------------------------------|
| x      | A MultiPhysioExperiment.              |
| stream | Stream name, or NULL for all streams. |

**Value**

Logical, named when stream is NULL.

**See Also**

[streamTimeIndex](#)

---

|       |                                             |
|-------|---------------------------------------------|
| hasNA | <i>Check if data contains any NA values</i> |
|-------|---------------------------------------------|

---

### Description

Quick check for NA presence in `PhysioExperiment` data.

### Usage

```
hasNA(x, assay_name = NULL)
```

### Arguments

|            |                                         |
|------------|-----------------------------------------|
| x          | A <code>PhysioExperiment</code> object. |
| assay_name | Optional specific assay to check.       |

### Value

Logical indicating presence of NA values.

### References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

### See Also

[checkNA](#) for detailed NA statistics, [naSummary](#) for per-assay NA summary, [replaceNA](#) for handling NA values

### Examples

```
pe <- PhysioExperiment(  
  assays = list(raw = matrix(1:4, nrow = 2)),  
  samplingRate = 100  
)  
hasNA(pe)
```

icc

*Intraclass Correlation Coefficient (ICC)***Description**

Computes the ICC using ANOVA-based methods following Shrout & Fleiss (1979). Supports one-way and two-way random/mixed models, agreement and consistency types, and single or average unit measures.

**Usage**

```
icc(
  ratings,
  model = c("twoway", "oneway"),
  type = c("agreement", "consistency"),
  unit = c("single", "average")
)
```

**Arguments**

|         |                                                                                                                                                                        |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ratings | Numeric matrix with subjects as rows and raters/sessions as columns.                                                                                                   |
| model   | Character; ICC model type:<br><b>"oneway"</b> One-way random effects (ICC(1,1) or ICC(1,k))<br><b>"twoway"</b> Two-way random/mixed effects (default)                  |
| type    | Character; agreement type:<br><b>"agreement"</b> Absolute agreement (ICC(2,1) for twoway)<br><b>"consistency"</b> Consistency/relative agreement (ICC(3,1) for twoway) |
| unit    | Character; unit of measurement:<br><b>"single"</b> Reliability for a single rater/measurement<br><b>"average"</b> Reliability for the mean of k raters/measurements    |

**Details**

The function implements the six ICC forms from Shrout & Fleiss (1979):

- ICC(1,1): One-way random, single measures
- ICC(1,k): One-way random, average measures
- ICC(2,1): Two-way random, absolute agreement, single measures
- ICC(2,k): Two-way random, absolute agreement, average measures
- ICC(3,1): Two-way mixed, consistency, single measures
- ICC(3,k): Two-way mixed, consistency, average measures

**Value**

A list with components:

**icc** ICC value

**ci\_lower** Lower bound of 95 percent confidence interval

**ci\_upper** Upper bound of 95 percent confidence interval

**f\_value** F statistic from ANOVA

**p\_value** p-value for testing ICC = 0

**model** Model used

**type** Agreement type used

**References**

Shrout PE, Fleiss JL (1979). Intraclass correlations: Uses in assessing rater reliability. Psychological Bulletin, 86(2), 420-428.

**See Also**

[sem\(\)](#) for standard error of measurement based on ICC, [mdc\(\)](#) for minimal detectable change, [blandAltman\(\)](#) for agreement analysis between methods.

**Examples**

```
# Test-retest reliability
ratings <- matrix(c(
  9, 2, 5, 8,
  6, 1, 3, 2,
  8, 4, 6, 8,
  7, 1, 2, 6,
  10, 5, 6, 9,
  6, 2, 4, 7
), nrow = 6, ncol = 4, byrow = TRUE)

result <- icc(ratings, model = "twoway", type = "agreement")
result$icc
```

---

|                    |                                                    |
|--------------------|----------------------------------------------------|
| is.PhysioBiomarker | <i>Test whether an object is a PhysioBiomarker</i> |
|--------------------|----------------------------------------------------|

---

**Description**

Test whether an object is a PhysioBiomarker

**Usage**

```
is.PhysioBiomarker(x)
```

**Arguments**

x                      Any object.

**Value**

TRUE if x is a [PhysioBiomarker](#), otherwise FALSE.

**See Also**

[physioBiomarker\(\)](#)

**Examples**

```
is.PhysioBiomarker(physioBiomarker(1, "x"))
is.PhysioBiomarker(42)
```

---

length,PhysioExperiment-method

*Length method for PhysioExperiment*

---

**Description**

Returns the number of time points (rows) in the default assay.

**Usage**

```
## S4 method for signature 'PhysioExperiment'
length(x)
```

**Arguments**

x                      A PhysioExperiment object.

**Value**

Integer scalar giving the number of time points (rows) in the default assay, or 0L if no assays are present.

**References**

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](#)

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

[dim, PhysioExperiment-method](#) for full dimensions, [nChannels](#) for the number of channels, [duration](#) for duration in seconds

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)
length(pe) # 100
```

logStep

*Record an analysis step in the provenance log***Description**

Appends a timestamped, agent-attributed PROV activity. Call this from any operation that returns a modified PhysioExperiment.

**Usage**

```
logStep(
  x,
  step,
  params = list(),
  package = NA_character_,
  version = NA_character_
)
```

**Arguments**

**x** A PhysioExperiment.

**step** Character scalar naming the operation (e.g. "filterSignals").

**params** Named list of parameters to record.

**package, version** Optional originating package name and version.

**Value**

x with the step appended (the input is unchanged; a modified copy is returned).

**Examples**

```
pe <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
pe <- logStep(pe, "filterSignals", params = list(low = 1, high = 40))
provenance(pe)
```

mdc

*Minimal Detectable Change (MDC)***Description**

Computes the minimal detectable change from the standard error of measurement.

**Usage**

```
mdc(sem_value, confidence = 0.95)
```

**Arguments**

`sem_value`      Numeric; the standard error of measurement.  
`confidence`      Numeric; confidence level (default 0.95).

**Details**

MDC is computed as:

$$MDC = SEM \times z \times \sqrt{2}$$

where  $z = \Phi^{-1}(1 - (1 - confidence)/2)$ .

At 95\

**Value**

Numeric MDC value.

**References**

Shrout PE, Fleiss JL (1979). "Intraclass Correlations: Uses in Assessing Rater Reliability." *Psychological Bulletin*, 86(2), 420-428.

**See Also**

[sem\(\)](#) for computing standard error of measurement, [icc\(\)](#) for computing intraclass correlation coefficients.

**Examples**

```
sem_val <- 2.5
mdc(sem_val, confidence = 0.95)
```

---

|                  |                                                       |
|------------------|-------------------------------------------------------|
| migrateContainer | <i>Convert a pre-unification multimodal container</i> |
|------------------|-------------------------------------------------------|

---

## Description

Rebuilds a container saved when the multimodal class carried experiments / alignment / sampleMap / couplingResults into the unified [MultiPhysioExperiment](#).

## Usage

```
migrateContainer(object, verbose = TRUE)

## S4 method for signature 'MultiPhysioExperiment'
updateObject(object, ..., verbose = FALSE)
```

## Arguments

|         |                                                                             |
|---------|-----------------------------------------------------------------------------|
| object  | A legacy container (or any object; non-legacy input is returned unchanged). |
| verbose | Emit a message describing what was carried over.                            |
| ...     | Ignored.                                                                    |

## Details

Two things are deliberately not converted:

- The old model had **no absolute origin**, so clock\$t0 is set to NA rather than to a fabricated zero.
- sampleMap and the cached couplingResults have no counterpart in the unified model. They are carried into clock\$migrated\_from with their origin rather than dropped, and the cache is **not** reused: it was keyed without reference to the signals or the clock, so nothing here can show it still matches its inputs.

The declared sampling rates in the old alignment table are checked against the real streams; a disagreement is reported rather than silently resolved.

## Value

A [MultiPhysioExperiment](#).

## See Also

[readPhysioRDS](#)

## Examples

```
# round-trips a current object unchanged
m <- MultiPhysioExperiment(a = PhysioExperiment(
  assays = list(raw = matrix(0, 10, 2)), samplingRate = 10))
identical(migrateContainer(m), m)
```

---

`multiphysio-legacy-accessors`*Legacy accessors for the multimodal container*

---

## Description

These are the names the container carried when it lived in **PhysioCrossModal**. They are kept so existing code runs unchanged and return the same types they always did: `experiments()` a plain named list (not a `SimpleList`), `modalities()` a character vector, `samplingRates()` a named numeric, `nModalities()` an integer.

## Usage

```
experiments(x)

## S4 method for signature 'MultiPhysioExperiment'
experiments(x)

experiments(x) <- value

## S4 replacement method for signature 'MultiPhysioExperiment'
experiments(x) <- value

modalities(x)

## S4 method for signature 'MultiPhysioExperiment'
modalities(x)

samplingRates(x)

## S4 method for signature 'MultiPhysioExperiment'
samplingRates(x)

nModalities(x)

## S4 method for signature 'MultiPhysioExperiment'
nModalities(x)

alignment(x)

## S4 method for signature 'MultiPhysioExperiment'
alignment(x)

alignment(x) <- value

## S4 replacement method for signature 'MultiPhysioExperiment'
alignment(x) <- value
```

**Arguments**

|                    |                                                                                                                                          |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>     | A MultiPhysioExperiment.                                                                                                                 |
| <code>value</code> | For <code>experiments&lt;-</code> a named list of streams; for <code>alignment&lt;-</code> a DataFrame with modality and offset columns. |

**Details**

`alignment()` is a *view* computed from the clock, not a stored second copy. Assigning to it writes back into the clock, so the two can never disagree.

**Value**

As described above; setters return the modified object.

---

MultiPhysioExperiment    *Construct a MultiPhysioExperiment*

---

**Description**

Holds several PhysioExperiment streams on one shared clock. Nothing is resampled and no synchronisation is estimated.

MultiRatePhysioExperiment() is the former name, kept as a compatibility entry point. It builds the same container and returns the legacy subclass so that existing `is()` tests keep passing.

**Usage**

```
MultiPhysioExperiment(
  streams = list(),
  ...,
  t0 = 0,
  offsets = NULL,
  reference_rate = NULL,
  clock = NULL,
  experiments = NULL,
  alignment = NULL
)

MultiRatePhysioExperiment(
  streams = list(),
  ...,
  t0 = 0,
  offsets = NULL,
  reference_rate = NULL,
  clock = NULL,
  experiments = NULL,
  alignment = NULL
)
```

**Arguments**

|                             |                                                                                                                                                                                                                            |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>streams</code>        | A named list of <code>PhysioExperiment</code> streams (or pass them as named ... arguments).                                                                                                                               |
| <code>...</code>            | Additional named <code>PhysioExperiment</code> streams.                                                                                                                                                                    |
| <code>t0</code>             | Origin of the shared clock in seconds (default 0). Use NA when the recording has no known absolute origin.                                                                                                                 |
| <code>offsets</code>        | Named numeric of per-stream start offsets in seconds relative to <code>t0</code> . Negative values are allowed. NULL (default) assumes a simultaneous start.                                                               |
| <code>reference_rate</code> | Default grid rate in Hz for <code>resampleToCommon</code> . Defaults to the highest stream rate.                                                                                                                           |
| <code>clock</code>          | Optional pre-built clock list, used instead of <code>t0</code> / <code>offsets</code> / <code>reference_rate</code> .                                                                                                      |
| <code>experiments</code>    | Legacy alias for <code>streams</code> .                                                                                                                                                                                    |
| <code>alignment</code>      | Legacy alignment <code>DataFrame</code> (columns <code>modality</code> , <code>samplingRate</code> , <code>offset</code> ). Its <code>offsets</code> are adopted; its declared rates are checked against the real streams. |

**Details**

Omitting `offsets` declares that the streams started together; that assumption is recorded in the clock (`offsets_assumed`) rather than being indistinguishable from a measured alignment. A partially specified `offsets` is an error rather than being completed with zeros.

**Value**

A `MultiPhysioExperiment`.

**See Also**

[streamRates](#), [timeWindow](#), [resampleToCommon](#)

**Examples**

```
kin <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(100 * 3), 100, 3)), samplingRate = 100)
emg <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(2000 * 2), 2000, 2)), samplingRate = 2000)
mpe <- MultiPhysioExperiment(kinematics = kin, emg = emg)
streamRates(mpe)

# a measured alignment: EMG started 137 ms after the clock origin
MultiPhysioExperiment(streams = list(kinematics = kin, emg = emg),
  offsets = c(kinematics = 0, emg = 0.137))
```

---

MultiPhysioExperiment-class

*Multimodal container for PhysioExperiment streams*


---

## Description

MultiPhysioExperiment holds several PhysioExperiment streams recorded together - the classic motion-capture situation of, say, 100 Hz kinematics alongside 1000 Hz force-plate analog and 2000 Hz EMG, or the NWB TimeSeries / LSL-XDF / C3D POINT+ANALOG pattern. Streams that happen to share a sampling rate use the same class; nothing about it is specific to mixed rates.

## Details

Each stream keeps its own sampling rate and length. The container adds one thing: a shared clock that says how the streams line up in time. There is deliberately no second alignment table - the clock is the single source of truth, and [alignment](#) is a view derived from it.

Construction never resamples and never estimates synchronisation. Holding the time correspondence and converting the signals onto a common grid ([resampleToCommon](#)) are separate steps.

## Slots

`streams` A SimpleList of named PhysioExperiment objects.

`clock` A list describing the shared clock (see above).

## The clock

`t0` Origin of the shared clock, in seconds. NA means the recording has no known absolute origin; that absence is preserved rather than replaced by a fabricated zero.

`offsets` Named numeric, one entry per stream, giving where each stream starts in seconds relative to `t0`. Negative values are allowed (a stream that started before the origin).

`reference_rate` Default grid rate used when the signals are explicitly converted to a common rate. It never overrides a stream's own sampling rate.

`offsets_assumed` TRUE when the offsets were not supplied and simultaneous start was assumed. A known common clock and an assumed one are different claims, so they are recorded differently.

`schema` Integer schema version, for migration.

Extra elements are preserved untouched, which is where measured timestamps, gap records or drift corrections belong.

Sample `i` of a stream sits at `offset + (i - 1) / samplingRate` seconds *relative to* `t0`. [streamTimeIndex](#) returns exactly that and does not add `t0`.

## See Also

[MultiPhysioExperiment](#) for the constructor, [timeWindow](#), [resampleToCommon](#), [alignStreams](#)

---

MultiPhysioExperiment-methods*Display, size and stream-extraction methods for MultiPhysioExperiment*

---

**Description**

Display, size and stream-extraction methods for MultiPhysioExperiment

**Usage**

```
## S4 method for signature 'MultiPhysioExperiment'
length(x)

## S4 method for signature 'MultiPhysioExperiment'
names(x)

## S4 method for signature 'MultiPhysioExperiment,ANY,ANY'
x[[i, j, ...]]

## S4 method for signature 'MultiPhysioExperiment'
dim(x)

## S4 method for signature 'MultiPhysioExperiment'
show(object)
```

**Arguments**

|                        |                                |
|------------------------|--------------------------------|
| <code>i</code>         | Stream name or index (for []). |
| <code>j, ...</code>    | Ignored.                       |
| <code>object, x</code> | A MultiPhysioExperiment.       |

**Value**

`length()` the number of streams; `dim()` a matrix of per-stream `c(nsamples, nchannels)`; `[[` the selected PhysioExperiment stream; `show()` is called for its side effect.

---

MultiRatePhysioExperiment-class*Legacy multi-rate container*

---

**Description**

MultiRatePhysioExperiment is the former name of what is now [MultiPhysioExperiment-class](#). It survives as an empty subclass so that objects saved under the old name keep loading, validating and dispatching, and so that code testing `is(x, "MultiRatePhysioExperiment")` keeps working. It adds no slots and no second clock model.

## Details

New code should use [MultiPhysioExperiment](#).

---

|         |                                        |
|---------|----------------------------------------|
| mvarFit | <i>Fit an MVAR model (user-facing)</i> |
|---------|----------------------------------------|

---

## Description

Convenience wrapper around [.fitMVAR](#) that optionally selects the model order automatically via [mvarOrderSelect](#).

## Usage

```
mvarFit(
  X,
  order = NULL,
  method = c("ols", "yulewalker", "nuttall-strand"),
  max_order = 20L,
  criterion = c("bic", "aic")
)
```

## Arguments

|           |                                                           |
|-----------|-----------------------------------------------------------|
| X         | A numeric matrix (time x channels).                       |
| order     | Integer order, or NULL (default) to select automatically. |
| method    | Estimator (see <a href="#">.fitMVAR</a> ).                |
| max_order | Maximum order for automatic selection (default: 20).      |
| criterion | Selection criterion when order is NULL (default "bic").   |

## Value

An object of class "mvar": a list with A, Sigma, order, method, and n\_channels.

## See Also

[mvarTransfer](#), [mvarOrderSelect](#)

## Examples

```
set.seed(1)
X <- matrix(rnorm(600), ncol = 3)
fit <- mvarFit(X, order = 2)
dim(fit$A)
```

---

|                 |                                                |
|-----------------|------------------------------------------------|
| mvarOrderSelect | <i>MVAR model-order selection by AIC / BIC</i> |
|-----------------|------------------------------------------------|

---

**Description**

Fits MVAR models over a range of orders and returns the order minimising the Akaike (AIC) or Bayesian (BIC) information criterion. For an m-channel model of order p with n effective samples the criteria are  $n \log \det(\Sigma_p) + 2 m^2 p$  (AIC) and  $n \log \det(\Sigma_p) + \log(n) m^2 p$  (BIC).

**Usage**

```
mvarOrderSelect(
  X,
  max_order = 20L,
  criterion = c("bic", "aic"),
  method = c("ols", "yulewalker", "nuttall-strand")
)
```

**Arguments**

|           |                                                               |
|-----------|---------------------------------------------------------------|
| X         | A numeric matrix (time x channels).                           |
| max_order | Maximum order to consider (default: 20).                      |
| criterion | "bic" (default) or "aic".                                     |
| method    | Estimator passed to <a href="#">.fitMVAR</a> (default "ols"). |

**Value**

A list with the selected order, the chosen criterion, and a data.frame criteria of order, aic, bic.

**See Also**

[.fitMVAR](#), [mvarFit](#)

---

|              |                                                         |
|--------------|---------------------------------------------------------|
| mvarTransfer | <i>MVAR transfer function and spectra (user-facing)</i> |
|--------------|---------------------------------------------------------|

---

**Description**

Convenience wrapper around [.mvarSpectral](#) for a fitted "mvar" model.

**Usage**

```
mvarTransfer(fit, freqs, sr)
```

**Arguments**

`fit` An "mvar" object from [mvarFit](#).  
`freqs` Numeric vector of frequencies in Hz.  
`sr` Sampling rate in Hz.

**Value**

A list as returned by [.mvarSpectral](#) (H, S, A, frequencies).

**See Also**

[mvarFit](#)

**Examples**

```
set.seed(1)
X <- matrix(rnorm(600), ncol = 3)
fit <- mvarFit(X, order = 2)
sp <- mvarTransfer(fit, freqs = seq(1, 40, by = 1), sr = 100)
dim(sp$H)
```

---

naSummary

*Get NA summary for all assays*

---

**Description**

Returns a summary of NA values across all assays.

**Usage**

```
naSummary(x)
```

**Arguments**

`x` A `PhysioExperiment` object.

**Value**

A `data.frame` with NA statistics for each assay.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[hasNA](#) for quick NA check, [checkNA](#) for detailed NA validation, [replaceNA](#) for handling NA values

**Examples**

```
pe <- PhysioExperiment(
  assays = list(
    raw = matrix(c(1, NA, 3, 4), nrow = 2),
    filtered = matrix(1:4, nrow = 2)
  ),
  samplingRate = 100
)
naSummary(pe)
```

---

|           |                               |
|-----------|-------------------------------|
| nChannels | <i>Get number of channels</i> |
|-----------|-------------------------------|

---

**Description**

Get number of channels

**Usage**

```
nChannels(x)
```

**Arguments**

x                      A PhysioExperiment object.

**Value**

Integer number of channels.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[channelNames](#) for channel labels, [channelInfo](#) for full channel metadata, [pickChannels](#) for sub-setting channels

Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)
nChannels(pe) # 4
```

---

|         |                             |
|---------|-----------------------------|
| nEvents | <i>Get number of events</i> |
|---------|-----------------------------|

---

Description

Get number of events

Usage

```
nEvents(x)
```

Arguments

x                    A PhysioEvents object.

Value

Integer count of events.

References

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

See Also

[getEvents](#) for retrieving events, [PhysioEvents](#) for the event constructor, [addEvents](#) for appending events

Examples

```
events <- PhysioEvents(onset = c(1, 2, 3), type = "stimulus")
nEvents(events)
```

---

|                 |                                                 |
|-----------------|-------------------------------------------------|
| normativeLookup | <i>Look up normative values for a biomarker</i> |
|-----------------|-------------------------------------------------|

---

## Description

Compares an observed marker to seeded published normative reference values (read from `inst/extdata/normative/*.csv`) returning a z-score and percentile. This is a scaffold seeded with a few quantitative-EEG markers (delta-alpha ratio, brain symmetry index); extend the CSVs to add markers. When the marker, age band, or montage is unknown the z-score and percentile are returned as NA without raising an error.

## Usage

```
normativeLookup(marker, age = NULL, montage = NULL)
```

## Arguments

|         |                                                                                                                                                                        |
|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| marker  | A <a href="#">PhysioBiomarker</a> (its name and value are used), or a character marker name (then only reference values are returned, with the z-score/percentile NA). |
| age     | Optional numeric age in years used to select an age-stratified reference row.                                                                                          |
| montage | Optional character montage used to select a montage-specific reference row.                                                                                            |

## Value

A named list: marker, value, mean, sd, unit, source, z\_score, percentile, and matched (logical). Unknown lookups return NA fields with matched = FALSE.

## See Also

[physioBiomarker\(\)](#), [NormativeReference\(\)](#), [zScore\(\)](#)

## Examples

```
bm <- physioBiomarker(2.35, "DAR")
normativeLookup(bm, age = 40)
normativeLookup("UNKNOWN_MARKER") # returns NA fields, no error
```

---

|                    |                                       |
|--------------------|---------------------------------------|
| NormativeReference | <i>Construct a NormativeReference</i> |
|--------------------|---------------------------------------|

---

## Description

Construct a NormativeReference

**Usage**

```
NormativeReference(
  metric,
  strata,
  source = NA_character_,
  version = NA_character_,
  unit = NA_character_
)
```

**Arguments**

|                      |                                                                                                                                                               |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>metric</code>  | Character metric name.                                                                                                                                        |
| <code>strata</code>  | <code>data.frame</code> with one row per stratum; must include numeric mean and sd columns (optionally <code>n</code> ); remaining columns are matching keys. |
| <code>source</code>  | Character citation (default NA).                                                                                                                              |
| <code>version</code> | Character version tag (default NA).                                                                                                                           |
| <code>unit</code>    | Character unit (default NA).                                                                                                                                  |

**Value**

A NormativeReference object.

**Examples**

```
ref <- NormativeReference(
  "gait_speed",
  strata = data.frame(
    sex = c("M", "F"), mean = c(1.34, 1.24), sd = c(0.20, 0.19), n = c(50, 50)
  ),
  source = "Bohannon 2011", unit = "m/s"
)
zScore(ref, 1.10, by = list(sex = "M"))
```

---

NormativeReference-class

*NormativeReference: stratified normative values for a metric*

---

**Description**

Holds published reference (normative) values for a single metric, stratified by one or more grouping variables (e.g. age band, sex). Downstream code turns an observed value into a z-score or a percent-of-predicted against the matching stratum, which is the basis for most rehabilitation cut-offs (6MWT, gait speed, grip strength, spirometry, ...).

**Usage**

```
## S4 method for signature 'NormativeReference'
show(object)
```

**Arguments**

object            A NormativeReference to display.

**Value**

The show method returns NULL invisibly and is called for the side effect of printing a compact summary.

**Slots**

metric Character name of the metric (e.g. "gait\_speed").  
 strata data.frame with one row per stratum. Must contain numeric mean and sd columns; may contain n; any remaining columns are the stratification keys used for matching.  
 source Character citation for the reference values.  
 version Character version tag for the reference set.  
 unit Character measurement unit.

**See Also**

[zScore](#), [percentPredicted](#)

---

NormativeReference-compare

*Normative comparisons for an observed value*

---

**Description**

Normative comparisons for an observed value

**Usage**

```
zScore(ref, value, by = NULL)

## S4 method for signature 'NormativeReference'
zScore(ref, value, by = NULL)

percentPredicted(ref, value, by = NULL)

## S4 method for signature 'NormativeReference'
percentPredicted(ref, value, by = NULL)
```

**Arguments**

ref            A NormativeReference.  
 value        Numeric observed value(s).  
 by            Named list or vector selecting the stratum (e.g. list(sex = "M", age = "60-69"));  
               NULL when the reference has a single stratum.

Value

zScore() the standard score  $(value - mean)/sd$ ; percentPredicted()  $100 \times value/mean$ .

Examples

```
ref <- NormativeReference(
  "gait_speed",
  strata = data.frame(
    sex = c("M", "F"), mean = c(1.34, 1.24), sd = c(0.20, 0.19)
  ),
  source = "Bohannon 2011", unit = "m/s"
)
zScore(ref, 1.10, by = list(sex = "M"))
percentPredicted(ref, 1.10, by = list(sex = "M"))
```

---

|          |                   |
|----------|-------------------|
| nStreams | Number of streams |
|----------|-------------------|

---

Description

Number of streams

Usage

```
nStreams(x)
```

Arguments

x                    A MultiPhysioExperiment.

Value

Integer count of streams.

---

|                 |                                                 |
|-----------------|-------------------------------------------------|
| physioBiomarker | Construct a reliability-characterised biomarker |
|-----------------|-------------------------------------------------|

---

Description

Builds a [PhysioBiomarker](#) carrying not just the value but the metadata a clinical reader needs: a confidence interval, a published reference range, reliability indices (ICC / SEM / MDC, per COSMIN reporting), and the measurement provenance (assay, band, method, software version). The value prints with its CI and, when a reference range is supplied, its normative percentile; [BiocGenerics::as.data.frame\(\)](#) flattens every field for tabular reporting.

## Usage

```
physioBiomarker(
  value,
  name,
  unit = NA_character_,
  ci = NULL,
  reference_range = NULL,
  reliability = list(),
  provenance = list(),
  interpretation = NA_character_
)
```

## Arguments

|                 |                                                                                                                                                                                       |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| value           | Numeric scalar value.                                                                                                                                                                 |
| name            | Character biomarker name (e.g. "DAR").                                                                                                                                                |
| unit            | Character measurement unit (default NA).                                                                                                                                              |
| ci              | Numeric length-2 confidence interval, or NULL for none.                                                                                                                               |
| reference_range | Numeric length-2 published reference (normal) range, or NULL for none. Interpreted as a 95 percent reference interval when computing the printed normative percentile.                |
| reliability     | Named list of reliability indices, typically <code>icc</code> , <code>sem</code> , and <code>mdc</code> (see <a href="#">icc()</a> , <a href="#">sem()</a> , <a href="#">mdc()</a> ). |
| provenance      | Named list describing how the value was computed, typically <code>assay</code> , <code>band</code> , <code>method</code> , and <code>software_version</code> .                        |
| interpretation  | Optional character interpretation label.                                                                                                                                              |

## Value

A [PhysioBiomarker](#) object.

## See Also

[PhysioBiomarker\(\)](#), [normativeLookup\(\)](#), [is.PhysioBiomarker\(\)](#), [icc\(\)](#), [sem\(\)](#), [mdc\(\)](#)

## Examples

```
bm <- physioBiomarker(
  value = 2.35, name = "DAR", unit = "ratio",
  ci = c(1.90, 2.80), reference_range = c(0.5, 1.2),
  reliability = list(icc = 0.82, sem = 0.15, mdc = 0.42),
  provenance = list(assay = "psd", band = "delta/alpha",
                    method = "welch", software_version = "1.0.0")
)
bm
as.data.frame(bm)
```

---

|                 |                                    |
|-----------------|------------------------------------|
| PhysioBiomarker | <i>Construct a PhysioBiomarker</i> |
|-----------------|------------------------------------|

---

**Description**

Construct a PhysioBiomarker

**Usage**

```
PhysioBiomarker(  
  name,  
  value,  
  unit = NA_character_,  
  ci = numeric(0),  
  interpretation = NA_character_,  
  parameters = list()  
)
```

**Arguments**

- name            Character biomarker name.
- value          Numeric scalar value.
- unit           Character unit (default NA).
- ci             Numeric length-2 confidence interval (default none).
- interpretation Optional character interpretation.
- parameters    Named list of parameters used.

**Value**

A PhysioBiomarker object.

**Examples**

```
PhysioBiomarker("SDNN", 42, unit = "ms", ci = c(38, 46))
```

---

|                       |                                                                   |
|-----------------------|-------------------------------------------------------------------|
| PhysioBiomarker-class | <i>PhysioBiomarker: a single computed biomarker with metadata</i> |
|-----------------------|-------------------------------------------------------------------|

---

**Description**

Extends [AnalysisResult](#) with a scalar value, unit, optional confidence interval, and interpretation  
- the unit reporting consumes.

**Usage**

```
## S4 method for signature 'PhysioBiomarker'
show(object)
```

**Arguments**

`object`                    An object to display.

**Value**

The `show` method for `PhysioBiomarker` returns `NULL` invisibly and is called for the side effect of printing a compact summary.

**Slots**

`name` Character biomarker name.  
`value` Numeric scalar value.  
`unit` Character measurement unit.  
`ci` Numeric length-2 confidence interval, or length-0 if none.  
`interpretation` Optional character interpretation label.  
`reference_range` Numeric length-2 published reference (normal) range, or length-0 if none.  
`reliability` Named list of reliability indices, typically `icc`, `sem`, and `mdc` (see [icc](#), [sem](#), [mdc](#)).  
`provenance_info` Named list describing how the value was computed, typically `assay`, `band`, `method`, and `software_version`.

---

|              |                                 |
|--------------|---------------------------------|
| PhysioCohort | <i>Construct a PhysioCohort</i> |
|--------------|---------------------------------|

---

**Description**

Construct a `PhysioCohort`

**Usage**

```
PhysioCohort(
  subjects = list(),
  ...,
  colData = NULL,
  group = NULL,
  metadata = list()
)
```

**Arguments**

|          |                                                                                                                                                                                                                                                                                                                                                           |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| subjects | A named list of subjects - each a <a href="#">PhysioLongitudinal</a> or a bare <code>PhysioExperiment</code> / <code>MultiRatePhysioExperiment</code> (auto-wrapped). Names become the <code>subject_ids</code> . Subjects may also be passed as named ....                                                                                               |
| ...      | Additional named subjects.                                                                                                                                                                                                                                                                                                                                |
| colData  | Optional <code>DataFrame</code> of subject-level metadata; must have one row per subject. If it has a <code>subject_id</code> column it is reordered to match the subjects, otherwise <code>subject_id</code> is taken from the names. If <code>NULL</code> , a minimal table is built from the names and each subject's own <code>subjectData()</code> . |
| group    | Optional vector (length = number of subjects) giving each subject's group / trial arm; stored as <code>colData\$group</code> .                                                                                                                                                                                                                            |
| metadata | Optional cohort-level metadata list.                                                                                                                                                                                                                                                                                                                      |

**Value**

A `PhysioCohort`.

**See Also**

[cohortDesign](#), [subjects](#), [addSubject](#)

**Examples**

```
mk <- function() PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(200), 100, 2)), samplingRate = 250)
coh <- PhysioCohort(
  "sub-01" = PhysioLongitudinal(baseline = mk(), discharge = mk(),
    design = S4Vectors::DataFrame(session_id = c("baseline", "discharge"),
      visit_label = c("baseline", "discharge"), days_from_baseline = c(0, 42))),
  "sub-02" = mk(),
  group = c("treatment", "control"))
coh
cohortDesign(coh)
```

---

|                    |                                               |
|--------------------|-----------------------------------------------|
| PhysioCohort-class | <i>Multi-subject cohort / study container</i> |
|--------------------|-----------------------------------------------|

---

**Description**

`PhysioCohort` is the multi-subject layer above [PhysioLongitudinal](#): it holds many participants, each represented by their own longitudinal timeline of sessions, alongside a subject-level `colData` table (`id`, `group/arm`, `diagnosis`, `age`, `sex`, `side`, ...). This is the container a rehabilitation study needs - a cohort or the arms of a trial, as subjects x sessions - and the shape downstream pipelines, trial analyses and prognostic models iterate over (see [cohortDesign](#)).

**Details**

A bare [PhysioExperiment](#) / [MultiRatePhysioExperiment](#) passed as a subject is auto-wrapped into a single-session [PhysioLongitudinal](#) (visit label "session"), so cross-sectional and longitudinal cohorts share one API.

**Slots**

`subjects` A [SimpleList](#) of named [PhysioLongitudinal](#) subjects.

`colData` A [DataFrame](#) with one row per subject; a `subject_id` column aligns to (and is ordered as) `names(subjects)`.

`metadata` A list of cohort-level metadata.

**See Also**

[PhysioCohort](#) (constructor), [cohortDesign](#), [subjects](#), [PhysioLongitudinal](#)

---

`PhysioCohort-methods`    *Size, subsetting and display for PhysioCohort*

---

**Description**

Size, subsetting and display for [PhysioCohort](#)

**Usage**

```
## S4 method for signature 'PhysioCohort'
length(x)

## S4 method for signature 'PhysioCohort,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'PhysioCohort'
show(object)
```

**Arguments**

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>x</code>       | A <a href="#">PhysioCohort</a> .                |
| <code>i</code>       | Subject index / name / logical (for []).        |
| <code>j, drop</code> | Ignored (present for [] generic compatibility). |
| <code>...</code>     | Ignored.                                        |
| <code>object</code>  | A <a href="#">PhysioCohort</a> (for show).      |

**Value**

`length()` the number of subjects; `[]` a subset [PhysioCohort](#); `show()` is called for its side effect.

---

PhysioEvents

Create a *PhysioEvents* object

---

## Description

Create a PhysioEvents object

## Usage

```
PhysioEvents(
  onset = numeric(),
  duration = numeric(),
  type = character(),
  value = character()
)
```

## Arguments

|          |                                                                 |
|----------|-----------------------------------------------------------------|
| onset    | Numeric vector of event onset times in seconds.                 |
| duration | Numeric vector of event durations in seconds.                   |
| type     | Character vector of event types (e.g., "stimulus", "response"). |
| value    | Character vector of event values/labels.                        |

## Value

A PhysioEvents object.

## References

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[setEvents](#) for attaching events to a PhysioExperiment, [getEvents](#) for retrieving events, [nEvents](#) for event count

## Examples

```
# Create events for a simple experiment
events <- PhysioEvents(
  onset = c(1.0, 2.5, 4.0, 5.5),
  duration = c(0.5, 0.5, 0.5, 0.5),
  type = c("stimulus", "response", "stimulus", "response"),
```

```

    value = c("target", "hit", "distractor", "false_alarm")
  )
  events

# Create events with single type
stim_events <- PhysioEvents(
  onset = c(1, 2, 3, 4, 5),
  type = "stimulus"
)

```

---

|                    |                                              |
|--------------------|----------------------------------------------|
| PhysioEvents-class | <i>Event management for PhysioExperiment</i> |
|--------------------|----------------------------------------------|

---

### Description

Functions for managing experimental events (triggers, markers, annotations) within PhysioExperiment objects. PhysioEvents class

### Details

A simple S4 class to store event information as a DataFrame.

### Slots

**events** A DataFrame containing event information with columns: onset (numeric), duration (numeric), type (character), value (character).

---

|                  |                                            |
|------------------|--------------------------------------------|
| PhysioExperiment | <i>Construct a PhysioExperiment object</i> |
|------------------|--------------------------------------------|

---

### Description

Creates a new PhysioExperiment instance, which extends SummarizedExperiment with a samplingRate slot for physiological signal data.

### Usage

```

PhysioExperiment(
  assays = S4Vectors::SimpleList(),
  rowData = NULL,
  colData = NULL,
  metadata = list(),
  samplingRate = as.numeric(NA),
  provenance = NULL
)

```

**Arguments**

|                           |                                                                                                                                                                                                                                                                                                                                              |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>assays</code>       | A SimpleList (or coercible object) of assay arrays.                                                                                                                                                                                                                                                                                          |
| <code>rowData</code>      | Feature-level metadata as a DataFrame.                                                                                                                                                                                                                                                                                                       |
| <code>colData</code>      | Sample-level metadata as a DataFrame.                                                                                                                                                                                                                                                                                                        |
| <code>metadata</code>     | Optional experiment-level metadata list.                                                                                                                                                                                                                                                                                                     |
| <code>samplingRate</code> | Numeric scalar sampling rate in Hz.                                                                                                                                                                                                                                                                                                          |
| <code>provenance</code>   | Optional. Either a character source identifier (e.g. a file path or dataset id) - in which case an initial PROV "import" activity recording wasDerivedFrom that source is seeded - or a pre-built provenance log (a list of entries) to attach. NULL (default) leaves the object with an empty audit trail. See <a href="#">provenance</a> . |

**Value**

A PhysioExperiment object containing the supplied assays, row/column metadata, and sampling rate.

**References**

- Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:[10.1038/nmeth.3252](https://doi.org/10.1038/nmeth.3252)
- Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

[samplingRate](#) for accessing the sampling rate, [defaultAssay](#) for retrieving the first assay name, [channelInfo](#) for channel metadata, [setEvents](#) for attaching event information

**Examples**

```
# Create a simple PhysioExperiment with random EEG-like data
# 1000 time points, 4 channels
eeg_data <- matrix(rnorm(1000 * 4), nrow = 1000, ncol = 4)
colnames(eeg_data) <- c("Fz", "Cz", "Pz", "Oz")

pe <- PhysioExperiment(
  assays = list(raw = eeg_data),
  colData = S4Vectors::DataFrame(
    label = c("Fz", "Cz", "Pz", "Oz"),
    type = rep("EEG", 4)
  ),
  samplingRate = 250
)
pe

# Access sampling rate
samplingRate(pe)

# Create with multiple assays
```

```
pe2 <- PhysioExperiment(
  assays = list(raw = eeg_data, filtered = eeg_data * 0.5),
  samplingRate = 500
)
```

---

 PhysioExperiment-class

*PhysioExperiment class definition*


---

## Description

The `PhysioExperiment` class extends `SummarizedExperiment` to store multi-modal physiological signal data alongside metadata such as sampling rate. This file defines the class, its validity checks, and the user-facing constructor.

## Slots

`samplingRate` Numeric scalar describing the acquisition frequency in Hz.

## References

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:10.1038/nmeth.3252

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

## See Also

[PhysioExperiment](#) for the constructor, [samplingRate](#) for accessing the sampling rate, [channelInfo](#) for channel metadata

---

 PhysioLongitudinal

*Construct a PhysioLongitudinal container*


---

## Description

Construct a `PhysioLongitudinal` container

## Usage

```
PhysioLongitudinal(sessions = list(), ..., design = NULL, subject = NULL)
```

**Arguments**

|          |                                                                                                                                                                                                                                                                                        |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sessions | A named list of <code>PhysioExperiment</code> / <code>MultiRatePhysioExperiment</code> sessions (or pass them as named ... arguments). Names become the <code>session_ids</code> .                                                                                                     |
| ...      | Additional named sessions.                                                                                                                                                                                                                                                             |
| design   | Optional <code>DataFrame</code> with columns <code>session_id</code> , <code>visit_label</code> , <code>days_from_baseline</code> and (optionally) <code>condition</code> . If <code>NULL</code> , a default is built from the session names (visit label = name, days = input order). |
| subject  | Optional one-row <code>DataFrame</code> of subject metadata (id, dx, side).                                                                                                                                                                                                            |

**Value**

A `PhysioLongitudinal` with sessions in chronological order.

**See Also**

[sessions](#), [addSession](#), [design](#)

**Examples**

```
mk <- function(sr = 250) PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(100 * 2), 100, 2)), samplingRate = sr)
pl <- PhysioLongitudinal(
  baseline = mk(), discharge = mk(),
  design = S4Vectors::DataFrame(
    session_id = c("baseline", "discharge"),
    visit_label = c("baseline", "discharge"),
    days_from_baseline = c(0, 42)),
  subject = S4Vectors::DataFrame(id = "sub-01", dx = "stroke", side = "L"))
design(pl)
```

---

PhysioLongitudinal-class

*Longitudinal subject/session container*

---

**Description**

`PhysioLongitudinal` is a subject-level container that links several recording sessions of the same participant across a rehabilitation timeline - typically baseline, mid, discharge and followup visits. Each session is a [PhysioExperiment](#) (or [MultiRatePhysioExperiment](#)); a design-schema `DataFrame` records the visit label, days from baseline and condition of each session, and a one-row subject `DataFrame` carries participant-level metadata (id, diagnosis, affected side). Sessions are always kept in chronological order (ascending `days_from_baseline`). It mirrors the **MultiAssay-Experiment** pattern, BIDS ses- entities and the OMOP visit-occurrence model.

**Slots**

`sessions` A SimpleList of named PhysioExperiment / MultiRatePhysioExperiment sessions.  
`design` A DataFrame with one row per session (`session_id`, `visit_label`, `days_from_baseline`, `condition`).  
`subject` A one-row DataFrame of subject metadata (`id`, `dx`, `side`).

**References**

Ramos, M., et al. (2017). Software for the integration of multiomics experiments in Bioconductor. *Cancer Research*, 77(21).

**See Also**

[PhysioLongitudinal](#) for the constructor, [addSession](#), [session](#)

---

 PhysioLongitudinal-methods

*Display, size and subsetting for PhysioLongitudinal*

---

**Description**

Display, size and subsetting for PhysioLongitudinal

**Usage**

```
## S4 method for signature 'PhysioLongitudinal'
length(x)

## S4 method for signature 'PhysioLongitudinal,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'PhysioLongitudinal'
show(object)
```

**Arguments**

|                      |                                                 |
|----------------------|-------------------------------------------------|
| <code>x</code>       | A PhysioLongitudinal.                           |
| <code>i</code>       | Session index/name/logical (for []).            |
| <code>j, drop</code> | Ignored (present for [] generic compatibility). |
| <code>...</code>     | Ignored.                                        |
| <code>object</code>  | A PhysioLongitudinal (for show).                |

**Value**

`length()` the number of sessions; [] a subset PhysioLongitudinal; `show()` is called for its side effect.

---

|               |                                                                      |
|---------------|----------------------------------------------------------------------|
| physioPalette | <i>Colorblind-safe palettes (qualitative, sequential, diverging)</i> |
|---------------|----------------------------------------------------------------------|

---

## Description

Returns  $n$  colors from a colour-vision-deficiency-safe palette. The "qualitative" palette is the Okabe & Ito set (distinguishable under deuteranopia / protanopia); "sequential" is viridis; and "diverging" is a CVD-safe blue-red diverging ramp. Shared across all ecosystem visualizations so figures are consistent and accessible.

## Usage

```
physioPalette(n = 8L, type = c("qualitative", "sequential", "diverging"))
```

## Arguments

|                   |                                                                       |
|-------------------|-----------------------------------------------------------------------|
| <code>n</code>    | Number of colors to return.                                           |
| <code>type</code> | Palette family: "qualitative" (default), "sequential" or "diverging". |

## Value

Character vector of  $n$  hex colors. The qualitative palette has only 8 colour-vision-safe entries; requesting  $n > 8$  qualitative colours emits a warning and returns a best-effort interpolation that is no longer reliably distinguishable under colour-vision deficiency (no qualitative palette is). Prefer faceting or a sequential encoding for more than 8 categories.

## References

Okabe, M. & Ito, K. (2008). Color Universal Design. Garnier et al. (viridis). Zeileis et al. (2020). colorspace, JSS 96(1).

## Examples

```
physioPalette(4)
physioPalette(7, "sequential")
physioPalette(5, "diverging")
```

---

|              |                               |
|--------------|-------------------------------|
| pickChannels | <i>Pick specific channels</i> |
|--------------|-------------------------------|

---

**Description**

Creates a new `PhysioExperiment` with only selected channels.

**Usage**

```
pickChannels(x, channels)
```

**Arguments**

|          |                                                         |
|----------|---------------------------------------------------------|
| x        | A <code>PhysioExperiment</code> object.                 |
| channels | Integer indices or character names of channels to keep. |

**Value**

A new `PhysioExperiment` with selected channels.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[dropChannels](#) for removing channels, [getChannelsByType](#) for finding channels by type, [channelNames](#) for available channel labels

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)

# Pick by index
pe_subset <- pickChannels(pe, c(1, 3))
nChannels(pe_subset) # 2

# Pick by name
pe_frontal <- pickChannels(pe, c("Fz", "Cz"))
```

---

```
print.fpca_result      Print fPCA result
```

---

### Description

Print fPCA result

### Usage

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

### Arguments

|     |                               |
|-----|-------------------------------|
| x   | An fpca_result object         |
| ... | Additional arguments (unused) |

### References

Ramsay JO, Silverman BW (2005). "Functional Data Analysis." 2nd ed. Springer.

### See Also

[fPCA\(\)](#) for performing functional principal component analysis, [plotFPCA\(\)](#) (in *PhysioMoCap*) for visualization of fPCA results, [reconstructFPCA\(\)](#) for waveform reconstruction from fPCA scores.

---

```
provenance      Provenance / audit trail for PhysioExperiment objects
```

---

### Description

Every analysis operation that returns a modified *PhysioExperiment* should record an append-only, timestamped, agent-attributed provenance entry following the W3C PROV data model (Lebo et al. 2013). Each entry captures a PROV *activity* (the operation), the *entity* it generated, the inputs it *used*, the responsible *agent*, its start/end times, and the parameters (also serialized as JSON for downstream export).

### Usage

```
provenance(x)

## S4 method for signature 'PhysioExperiment'
provenance(x)

provenance(x) <- value
```

```
## S4 replacement method for signature 'PhysioExperiment'
provenance(x) <- value

## S4 method for signature 'MultiPhysioExperiment'
provenance(x)

## S4 method for signature 'PhysioLongitudinal'
provenance(x)

## S4 method for signature 'PhysioCohort'
provenance(x)
```

### Arguments

|       |                                               |
|-------|-----------------------------------------------|
| x     | A PhysioExperiment object.                    |
| value | A provenance entry list (used by the setter). |

### Details

The log is stored in the object's `metadata()` under the key "provenance". Storing it in metadata (rather than a dedicated S4 slot) is deliberate: objects serialized before provenance existed, or by any other SummarizedExperiment tool, deserialize cleanly and simply report an empty log, with no need for a class-version `updateObject` migration.

### Value

`provenance()` returns a `data.frame` with one row per recorded PROV activity. Columns include the PROV-O fields `activity`, `entity`, `used`, `generated`, `agent`, `startedAtTime`, `endedAtTime`, `params_json`, plus the back-compatible `step`, `timestamp`, `user`, `package`, `version`, and `params`. Empty if none recorded.

### References

Lebo, T., Sahoo, S., & McGuinness, D. (2013). PROV-O: The PROV Ontology. W3C Recommendation.

### See Also

[logStep](#), [withProvenance](#), [recordActivity](#)

### Examples

```
pe <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
pe <- logStep(pe, "filterSignals", params = list(low = 1, high = 40))
provenance(pe)
```

---

|                |                                                         |
|----------------|---------------------------------------------------------|
| provenanceHash | <i>Deterministic hash of the provenance audit trail</i> |
|----------------|---------------------------------------------------------|

---

### Description

Returns a stable hash of the *semantic* content of the provenance log - the ordered sequence of processing steps and their parameters - deliberately excluding wall-clock timestamps and the machine-specific agent/user. The same pipeline (same steps, same parameters) therefore always hashes to the same value, while any change to a step or its parameters changes the hash. Useful as a reproducibility fingerprint for a processed object.

### Usage

```
provenanceHash(x, algo = "xxhash64")
```

### Arguments

|      |                                                                                           |
|------|-------------------------------------------------------------------------------------------|
| x    | A <code>PhysioExperiment</code> (or any object with a <a href="#">provenance</a> method). |
| algo | Hash algorithm passed to <code>digest::digest</code> (default "xxhash64").                |

### Value

A single character hash string.

### See Also

[provenance](#), [logStep](#)

### Examples

```
pe <- PhysioExperiment(assays = list(raw = matrix(rnorm(40), 10, 4)),
                        samplingRate = 100)
pe <- logStep(pe, "filterSignals", params = list(low = 1, high = 40))
provenanceHash(pe)
```

---

|              |                                                |
|--------------|------------------------------------------------|
| rayleighTest | <i>Rayleigh test for a preferred direction</i> |
|--------------|------------------------------------------------|

---

### Description

Tests the null hypothesis that the angles are uniformly distributed on the circle (no preferred direction) against a unimodal concentration. A small p-value means the angles cluster around a mean direction – e.g. that a coupling angle is consistently in-phase rather than random.

### Usage

```
rayleighTest(angles, units = c("degrees", "radians"))
```

**Arguments**

angles            Numeric vector of angles.  
 units            "degrees" (default) or "radians".

**Value**

an htest: Rayleigh  $Z = n * \bar{r}^2$ , p.value (Zar 1999 approximation), and the estimated mean direction.

**References**

Zar JH (1999) eq. 27.2-27.4; Fisher NI (1993).

**See Also**

[circularSummary\(\)](#), [watsonWilliamsTest\(\)](#)

**Examples**

```
set.seed(1)
rayleighTest(rnorm(40, mean = 30, sd = 12))$p.value    # clustered -> small
```

---

|             |                                                 |
|-------------|-------------------------------------------------|
| rbindPhysio | <i>Combine PhysioExperiment objects by time</i> |
|-------------|-------------------------------------------------|

---

**Description**

Concatenates two PhysioExperiment objects along the time (row) axis. Both objects must have the same number of channels and matching sampling rates. Event onsets in y are offset by the duration of x.

**Usage**

```
rbindPhysio(x, y)
```

**Arguments**

x            A PhysioExperiment object.  
 y            A PhysioExperiment object to concatenate.

**Value**

A PhysioExperiment object with time points from both x and y concatenated, combined rowData, and merged events.

## References

- Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:10.1038/nmeth.3252
- Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

## See Also

[cbindPhysio](#) for combining along the channel axis, [extractWindow](#) for extracting a time window, [\[](#) for general subsetting

## Examples

```
pe1 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)
pe2 <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  samplingRate = 100
)

# Concatenate in time
pe_concat <- rbindPhysio(pe1, pe2)
length(pe_concat) # 200
```

---

readPhysioRDS

---

Read a saved object, migrating a pre-unification container

---

## Description

`readRDS()` performs no validity check, so a container saved before the multimodal classes were unified comes back structurally invalid and says nothing about it; the failure only surfaces when a slot is first touched. This reader detects that layout and converts it, reporting what it did.

## Usage

```
readPhysioRDS(file, migrate = TRUE, verbose = TRUE)
```

## Arguments

|                      |                                                                                                                                                                                                        |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>file</code>    | Path to an <code>.rds</code> file.                                                                                                                                                                     |
| <code>migrate</code> | Convert a legacy container (default). When <code>FALSE</code> the object is returned exactly as stored, and a legacy container is reported with a warning rather than silently handed over as invalid. |
| <code>verbose</code> | Describe the migration.                                                                                                                                                                                |

**Details**

The file is only read - it is never rewritten. Save the returned object somewhere new if the converted form should persist.

**Value**

The stored object, migrated when applicable.

**See Also**

[migrateContainer](#)

---

reconstructFPCA

*Reconstruct waveforms from fPCA*

---

**Description**

Reconstructs individual waveforms using a subset of principal components.

**Usage**

```
reconstructFPCA(fpca_result, n_components = NULL, observation = NULL)
```

**Arguments**

|                           |                                                                     |
|---------------------------|---------------------------------------------------------------------|
| <code>fpca_result</code>  | An <code>fpca_result</code> object from <code>fPCA()</code> .       |
| <code>n_components</code> | Number of components to use for reconstruction.                     |
| <code>observation</code>  | Indices of observations to reconstruct. If <code>NULL</code> , all. |

**Value**

Matrix of reconstructed waveforms (time x observations).

**References**

Ramsay JO, Silverman BW (2005). "Functional Data Analysis." 2nd ed. Springer.

**See Also**

[fPCA\(\)](#) for performing the decomposition, [plotFPCA\(\)](#) (in `PhysioMoCap`) for visualizing `fPCA` results.

## Examples

```
# Create sample data and run fPCA first
set.seed(123)
t <- seq(0, 100, length.out = 100)
base_curve <- sin(2 * pi * t / 100) * 30
data <- sapply(1:20, function(i) base_curve * rnorm(1, 1, 0.2) + rnorm(100, 0, 2))
fpca_result <- fPCA(data, n_components = 4)

# Reconstruct using only first 2 PCs
reconstructed <- reconstructFPCA(fpca_result, n_components = 2)
```

---

|                |                                                   |
|----------------|---------------------------------------------------|
| recordActivity | <i>Record a PROV activity around an operation</i> |
|----------------|---------------------------------------------------|

---

## Description

Evaluates `expr` (an operation that transforms `x` and returns a `PhysioExperiment`), timing it to fill the PROV startedAtTime / endedAtTime, and appends one activity to the result. The input's provenance log is carried onto the result if the operation dropped it.

## Usage

```
recordActivity(
  x,
  activity,
  expr,
  params = list(),
  input_assay = NA_character_,
  output_assay = NA_character_,
  agent = NA_character_,
  software_version = NA_character_,
  package = NA_character_
)
```

## Arguments

|                                                                                                                                  |                                                                                                                   |
|----------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>x</code>                                                                                                                   | The input <code>PhysioExperiment</code> .                                                                         |
| <code>activity</code>                                                                                                            | Character scalar naming the PROV activity.                                                                        |
| <code>expr</code>                                                                                                                | An expression, evaluated lazily, that returns the modified object (e.g. <code>filterSignals(x, low = 1)</code> ). |
| <code>params</code>                                                                                                              | Named list of parameters to record.                                                                               |
| <code>input_assay</code> , <code>output_assay</code> , <code>agent</code> , <code>software_version</code> , <code>package</code> | As in <a href="#">appendProvenance</a> .                                                                          |

## Value

The object returned by `expr`, with the timed activity appended.

**See Also**[withProvenance](#), [logStep](#)**Examples**

```
pe <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
scale2 <- function(p) { SummarizedExperiment::assay(p, "raw") <-
  SummarizedExperiment::assay(p, "raw") * 2; p }
pe <- recordActivity(pe, "rescale", scale2(pe), params = list(gain = 2))
provenance(pe)$activity
```

recordingMap

*How trials map onto recordings***Description**

One trial can be captured by several devices, so this table allows more than one row per trial and is unique on (subject\_id, session\_id, trial\_id, recording\_id).

**Usage**

```
recordingMap(x)
```

```
recordingMap(x) <- value
```

**Arguments**

|       |                                                                                      |
|-------|--------------------------------------------------------------------------------------|
| x     | A MultiPhysioExperiment.                                                             |
| value | A data frame with the four key columns, time_correspondence, and reference / offset. |

**Details**

time\_correspondence states what is known about placing the recordings relative to one another:

"measured" The relative times were measured - a shared trigger, say. reference names what they are measured against (a common clock or a reference recording\_id) and offset gives the value in seconds. Elapsed time between trials may be computed.

"order\_only" Only the order is known. Elapsed time may not be computed - row order is not a duration.

"unknown" Nothing relates the recordings.

A row may declare "measured" while leaving reference or offset missing: the table records what was declared, and [elapsedBetweenTrials](#) refuses to compute from it. The state name alone never authorises a duration.

**Value**

The recording map, or the modified container.

**See Also**

[elapsedBetweenTrials](#), [trials](#)

---

|            |                                           |
|------------|-------------------------------------------|
| recordings | <i>The recordings held by a trial set</i> |
|------------|-------------------------------------------|

---

**Description**

The recordings held by a trial set

**Usage**

```
recordings(x)
```

**Arguments**

x                   A physio\_trial\_set.

**Value**

The named list of MultiPhysioExperiment recordings.

**See Also**

[trialSet](#)

---

|                |                                          |
|----------------|------------------------------------------|
| registerCurves | <i>Curve registration (time warping)</i> |
|----------------|------------------------------------------|

---

**Description**

Aligns waveforms by estimating and removing phase variation. Uses landmark registration or continuous registration.

**Usage**

```
registerCurves(  
  x,  
  method = c("continuous", "landmark"),  
  landmarks = NULL,  
  template = NULL  
)
```

**Arguments**

|           |                                                                         |
|-----------|-------------------------------------------------------------------------|
| x         | A <code>PhysioExperiment</code> object or matrix (time x observations). |
| method    | Registration method: "landmark" or "continuous".                        |
| landmarks | For landmark method, matrix of landmark times (landmarks x obs).        |
| template  | Template curve to align to. If <code>NULL</code> , uses mean.           |

**Details**

Phase variation (timing differences) can obscure amplitude differences in biomechanical data. Registration separates phase and amplitude variation.

**Value**

A list containing:

|            |                                |
|------------|--------------------------------|
| registered | Registered waveforms           |
| warping    | Warping functions              |
| template   | Template used for registration |

**References**

Ramsay JO, Silverman BW (2005). "Functional Data Analysis." 2nd ed. Springer.

**See Also**

[fPCA\(\)](#) for functional PCA after registration, `plotGaitCycle()` (in `PhysioMoCap`) for plotting registered gait waveforms.

**Examples**

```
# Simulate data with phase variation
set.seed(123)
t <- seq(0, 100, length.out = 100)

data <- sapply(1:20, function(i) {
  phase_shift <- rnorm(1, 0, 10)
  t_shifted <- t + phase_shift
  sin(2 * pi * t_shifted / 100) * 30 + rnorm(100, 0, 2)
})

pe <- PhysioExperiment(assays = list(values = data), samplingRate = 100)
reg_result <- registerCurves(pe, method = "continuous")
```

---

|                   |                                               |
|-------------------|-----------------------------------------------|
| registerOperation | <i>Register and dispatch named operations</i> |
|-------------------|-----------------------------------------------|

---

## Description

Register and dispatch named operations

## Usage

```
registerOperation(name, fn, modality = NULL, overwrite = FALSE)
```

```
getOperation(name)
```

```
availableOperations(modality = NULL)
```

```
unregisterOperation(name)
```

## Arguments

|           |                                                                                                                                                                                                                                                                 |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name      | Case-insensitive operation key.                                                                                                                                                                                                                                 |
| fn        | The operation function.                                                                                                                                                                                                                                         |
| modality  | For registerOperation(), an optional modality tag (e.g. "eeg", "ecg"; a character vector is allowed for multi-modality ops). For availableOperations(), an optional modality to filter the table to operations tagged with it (NULL, the default, returns all). |
| overwrite | If FALSE (default), registering an existing name errors; TRUE replaces it.                                                                                                                                                                                      |

## Value

registerOperation() invisibly returns fn; getOperation() returns the registered function; availableOperations() returns a data.frame of name/modality (the name column holds normalized, lower-cased keys); unregisterOperation() invisibly returns TRUE if an operation was removed.

## See Also

Other plugin-api: [physio-registry](#), [registerReader\(\)](#), [registerWriter\(\)](#)

## Examples

```
registerOperation("demo_detrend", function(x) x - mean(x),
                  modality = "generic", overwrite = TRUE)
availableOperations()
availableOperations(modality = "generic")
op <- getOperation("demo_detrend")
unregisterOperation("demo_detrend")
```

---

|                |                                           |
|----------------|-------------------------------------------|
| registerReader | <i>Register and dispatch file readers</i> |
|----------------|-------------------------------------------|

---

## Description

Register and dispatch file readers

## Usage

```
registerReader(format, fn, ext = NULL, overwrite = FALSE)

getReader(format)

availableReaders()

unregisterReader(format)
```

## Arguments

|           |                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------|
| format    | Case-insensitive format key (e.g. "brainvision").                                                                          |
| fn        | The reader function; its first argument should be a file path.                                                             |
| ext       | Optional character vector of file extensions handled (no dot).                                                             |
| overwrite | If FALSE (default), registering an existing format errors; TRUE replaces it (use this in a package <code>.onLoad</code> ). |

## Value

`registerReader()` invisibly returns `fn`; `getReader()` returns the registered function; `availableReaders()` returns a `data.frame` of `format/ext` (the `format` column holds normalized, lower-cased keys, and the frame has zero rows when nothing is registered); `unregisterReader()` invisibly returns TRUE if a reader was removed.

## See Also

Other plugin-api: [physio-registry](#), [registerOperation\(\)](#), [registerWriter\(\)](#)

## Examples

```
registerReader("demo_csv", function(file, ...) read.csv(file, ...),
              ext = "csv", overwrite = TRUE)
availableReaders()
r <- getReader("DEMO_CSV") # case-insensitive
unregisterReader("demo_csv")
```

---

|                |                                           |
|----------------|-------------------------------------------|
| registerWriter | <i>Register and dispatch file writers</i> |
|----------------|-------------------------------------------|

---

## Description

Register and dispatch file writers

## Usage

```
registerWriter(format, fn, ext = NULL, overwrite = FALSE)

getWriter(format)

availableWriters()

unregisterWriter(format)
```

## Arguments

|           |                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------|
| format    | Case-insensitive format key (e.g. "brainvision").                                                                          |
| fn        | The writer function; its first arguments should be the object to write and a file path.                                    |
| ext       | Optional character vector of file extensions handled (no dot).                                                             |
| overwrite | If FALSE (default), registering an existing format errors; TRUE replaces it (use this in a package <code>.onLoad</code> ). |

## Value

`registerWriter()` invisibly returns `fn`; `getWriter()` returns the registered function; `availableWriters()` returns a `data.frame` of `format/ext` (the `format` column holds normalized, lower-cased keys, and the frame has zero rows when nothing is registered); `unregisterWriter()` invisibly returns `TRUE` if a writer was removed.

## See Also

Other plugin-api: [physio-registry](#), [registerOperation\(\)](#), [registerReader\(\)](#)

## Examples

```
registerWriter("demo_csv", function(x, file, ...) write.csv(x, file, ...),
              ext = "csv", overwrite = TRUE)
availableWriters()
unregisterWriter("demo_csv")
```

---

|              |                                                     |
|--------------|-----------------------------------------------------|
| removeEvents | <i>Remove events from a PhysioExperiment object</i> |
|--------------|-----------------------------------------------------|

---

## Description

Remove events from a PhysioExperiment object

## Usage

```
removeEvents(x, type = NULL, indices = NULL)
```

## Arguments

|         |                                                              |
|---------|--------------------------------------------------------------|
| x       | A PhysioExperiment object.                                   |
| type    | Optional event types to remove. If NULL, removes all events. |
| indices | Optional integer indices of events to remove.                |

## Value

The modified PhysioExperiment object.

## References

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[getEvents](#) for retrieving events, [addEvents](#) for appending events, [setEvents](#) for replacing all events

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 100)),
  samplingRate = 100
)
pe <- addEvents(pe, onset = c(1, 2, 3),
  type = c("stimulus", "response", "stimulus"))
pe <- removeEvents(pe, type = "response")
nEvents(pe)
```

---

|                |                        |
|----------------|------------------------|
| renameChannels | <i>Rename channels</i> |
|----------------|------------------------|

---

## Description

Rename channels

## Usage

```
renameChannels(x, old_names, new_names)
```

## Arguments

|           |                                         |
|-----------|-----------------------------------------|
| x         | A <code>PhysioExperiment</code> object. |
| old_names | Character vector of current names.      |
| new_names | Character vector of new names.          |

## Value

Modified `PhysioExperiment` object.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[channelNames](#) for reading channel labels, [channelInfo](#) for full channel metadata, [pickChannels](#) for subsetting channels

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)

# Rename channels
pe <- renameChannels(pe, c("Fz", "Cz"), c("F3", "C3"))
channelNames(pe)
```

---

|           |                                   |
|-----------|-----------------------------------|
| replaceNA | <i>Replace NA values in assay</i> |
|-----------|-----------------------------------|

---

**Description**

Creates a new assay with NA values handled according to the specified method.

**Usage**

```
replaceNA(  
  x,  
  method = "interpolate",  
  input_assay = NULL,  
  output_assay = "na_handled"  
)
```

**Arguments**

|              |                                                              |
|--------------|--------------------------------------------------------------|
| x            | A <code>PhysioExperiment</code> object.                      |
| method       | Method for handling NA (see <code>handleNA</code> ).         |
| input_assay  | Input assay name. If <code>NULL</code> , uses default assay. |
| output_assay | Output assay name. Default is "na_handled".                  |

**Value**

Modified `PhysioExperiment` with new assay.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[handleNA](#) for the underlying NA handling strategies, [checkNA](#) for NA validation, [hasNA](#) for quick NA presence check

**Examples**

```
pe <- PhysioExperiment(  
  assays = list(raw = matrix(c(1, NA, 3, NA, 5, 6), nrow = 3)),  
  samplingRate = 100  
)  
  
# Interpolate NA values  
pe <- replaceNA(pe, method = "interpolate")
```

---

|                  |                                                            |
|------------------|------------------------------------------------------------|
| resampleToCommon | <i>Resample all streams onto a single common-rate view</i> |
|------------------|------------------------------------------------------------|

---

### Description

Interpolates every stream onto a shared time grid at rate, honouring each stream's start offset on the shared clock, and returns a single `PhysioExperiment` whose channels are the union of all streams' channels (prefixed with the stream name). Grid positions before/after a stream's coverage are NA.

### Usage

```
resampleToCommon(x, rate = NULL)
```

### Arguments

|      |                                                                 |
|------|-----------------------------------------------------------------|
| x    | A <code>MultiPhysioExperiment</code> .                          |
| rate | Target sampling rate in Hz (default: the clock reference rate). |

### Value

A single-rate `PhysioExperiment` with an "aligned" assay.

### See Also

[alignStreams](#), [streamTimeIndex](#)

### Examples

```
kin <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(100 * 2), 100, 2)), samplingRate = 100)
emg <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(1000 * 2), 1000, 2)), samplingRate = 1000)
mpe <- MultiPhysioExperiment(kin = kin, emg = emg)
aligned <- resampleToCommon(mpe, 1000)
dim(SummarizedExperiment::assay(aligned, "aligned"))
```

---

|              |                                                     |
|--------------|-----------------------------------------------------|
| resolveQuery | <i>Resolve an EventQuery to get filtered events</i> |
|--------------|-----------------------------------------------------|

---

### Description

Resolve an `EventQuery` to get filtered events

### Usage

```
resolveQuery(q)
```

**Arguments**

q                      An EventQuery object

**Value**

Data frame of filtered events

**Examples**

```
pe <- PhysioExperiment(assays = list(raw = matrix(rnorm(400), 100, 4)),
                           samplingRate = 100)
pe <- addEvents(pe, onset = c(1, 2, 3), type = "stimulus")
resolveQuery(eventQuery(pe))
```

---

|               |                                        |
|---------------|----------------------------------------|
| samplesToTime | <i>Convert sample indices to times</i> |
|---------------|----------------------------------------|

---

**Description**

Convert sample indices to times

**Usage**

```
samplesToTime(x, samples)
```

**Arguments**

x                      A PhysioExperiment object.  
samples                Integer vector of sample indices.

**Value**

Numeric vector of times in seconds.

**References**

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

**See Also**

[timeToSamples](#) for the inverse conversion, [samplingRate](#) for the sampling rate, [timeIndex](#) for the full time vector

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 100)),
  samplingRate = 100
)
samplesToTime(pe, c(1L, 51L, 101L))
```

samplingRate

*Accessors for PhysioExperiment***Description**

These helper functions expose common slots and derived quantities for `PhysioExperiment` objects. Get or set sampling rate

**Usage**

```
samplingRate(x)

## S4 method for signature 'PhysioExperiment'
samplingRate(x)

samplingRate(x) <- value

## S4 replacement method for signature 'PhysioExperiment'
samplingRate(x) <- value
```

**Arguments**

`x`                    A `PhysioExperiment` object.

`value`                Numeric scalar for the new sampling rate in Hz.

**Value**

For `samplingRate(x)`: a numeric scalar giving the sampling rate in Hz. For `samplingRate(x) <- value`: the modified `PhysioExperiment` object (returned invisibly).

**References**

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:10.1038/nmeth.3252

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

`PhysioExperiment` for the constructor, `defaultAssay` for the default assay name, `duration` for signal duration, `timeIndex` for time point vector

### Examples

```
# Create example data
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(100), nrow = 10)),
  samplingRate = 250
)

# Get sampling rate
samplingRate(pe)

# Set sampling rate
samplingRate(pe) <- 500
samplingRate(pe)
```

---

|                    |                                                                  |
|--------------------|------------------------------------------------------------------|
| scale_color_physio | <i>Colorblind-safe discrete colour / fill scales for ggplot2</i> |
|--------------------|------------------------------------------------------------------|

---

### Description

Colorblind-safe discrete colour / fill scales for ggplot2

### Usage

```
scale_color_physio(...)

scale_colour_physio(...)

scale_fill_physio(...)
```

### Arguments

... Passed to `ggplot2::discrete_scale()`.

### Value

A ggplot2 discrete scale using [physioPalette](#).

### Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  p <- ggplot2::ggplot(
    iris,
    ggplot2::aes(Sepal.Length, Sepal.Width, color = Species)) +
    ggplot2::geom_point() + scale_color_physio() + theme_physio()
}
```

sem

*Standard Error of Measurement (SEM)***Description**

Computes the standard error of measurement from the SD of scores and a reliability coefficient (ICC or test-retest correlation).

**Usage**

```
sem(x, icc_value = NULL, reliability = NULL)
```

**Arguments**

|                          |                                                                                                                                                                   |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>           | Numeric vector of observed scores. Used to compute SD.                                                                                                            |
| <code>icc_value</code>   | Numeric; ICC value to use as the reliability coefficient.                                                                                                         |
| <code>reliability</code> | Numeric; alternative reliability coefficient (e.g., test-retest correlation). Exactly one of <code>icc_value</code> or <code>reliability</code> must be provided. |

**Details**

SEM is computed as:

$$SEM = SD \times \sqrt{1 - r}$$

where  $r$  is the reliability coefficient (ICC or correlation).

**Value**

Numeric SEM value.

**References**

Shrout PE, Fleiss JL (1979). "Intraclass Correlations: Uses in Assessing Rater Reliability." *Psychological Bulletin*, 86(2), 420-428.

**See Also**

[icc\(\)](#) for computing intraclass correlation coefficients, [mdc\(\)](#) for minimal detectable change based on SEM.

**Examples**

```
scores <- c(10, 12, 15, 11, 13, 14, 9, 16, 12, 11)
sem(scores, icc_value = 0.90)
```

---

|         |                                                       |
|---------|-------------------------------------------------------|
| session | <i>Retrieve a single session by visit label or id</i> |
|---------|-------------------------------------------------------|

---

**Description**

Retrieve a single session by visit label or id

**Usage**

```
session(x, label)
```

**Arguments**

|       |                                                                              |
|-------|------------------------------------------------------------------------------|
| x     | A <code>PhysioLongitudinal</code> .                                          |
| label | A <code>visit_label</code> (e.g. "discharge") or a <code>session_id</code> . |

**Value**

The matching session, or an error if none/ambiguous.

**See Also**

[sessions](#)

---

|          |                                                                 |
|----------|-----------------------------------------------------------------|
| sessions | <i>Access the sessions of a <code>PhysioLongitudinal</code></i> |
|----------|-----------------------------------------------------------------|

---

**Description**

Access the sessions of a `PhysioLongitudinal`

**Usage**

```
sessions(x)

## S4 method for signature 'PhysioLongitudinal'
sessions(x)

sessions(x) <- value

## S4 replacement method for signature 'PhysioLongitudinal'
sessions(x) <- value
```

**Arguments**

`x`                    A `PhysioLongitudinal`.  
`value`                A named list / `SimpleList` of sessions (setter).

**Value**

`sessions()` a `SimpleList`; setter returns the object.

---

`setAssaySamplingRate`    *Set the sampling rate of a specific assay*

---

**Description**

Records a per-assay sampling rate in the object's metadata; useful after an assay has been resampled to a rate different from the main one.

**Usage**

```
setAssaySamplingRate(x, assay_name, rate)
```

**Arguments**

`x`                    A `PhysioExperiment` object.  
`assay_name`        Name of the assay.  
`rate`                Sampling rate for the assay in Hz.

**Value**

The `PhysioExperiment` with updated per-assay rate metadata.

**See Also**

[assaySamplingRates](#), [MultiRatePhysioExperiment](#)

**Examples**

```
pe <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2),
    decimated = matrix(rnorm(20), 10, 2)),
  samplingRate = 100)
pe <- setAssaySamplingRate(pe, "decimated", 50)
assaySamplingRates(pe)
```

---

|                 |                          |
|-----------------|--------------------------|
| setChannelTypes | <i>Set channel types</i> |
|-----------------|--------------------------|

---

**Description**

Assigns types (EEG, EMG, EOG, etc.) to channels.

**Usage**

```
setChannelTypes(x, types)
```

**Arguments**

|       |                                                                                                            |
|-------|------------------------------------------------------------------------------------------------------------|
| x     | A <code>PhysioExperiment</code> object.                                                                    |
| types | Named character vector or list mapping channel names/indices to types. If unnamed, applies types in order. |

**Value**

Modified `PhysioExperiment` object.

**References**

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

**See Also**

[getChannelsByType](#) for querying channels by type, [channelInfo](#) for full channel metadata, [setChannelUnits](#) for assigning physical units

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "EOG1", "EMG1", "Oz")),
  samplingRate = 100
)

# Set all channels to same type
pe <- setChannelTypes(pe, "EEG")

# Set specific channel types by name
pe <- setChannelTypes(pe, c(EOG1 = "EOG", EMG1 = "EMG"))
```

---

|                 |                          |
|-----------------|--------------------------|
| setChannelUnits | <i>Set channel units</i> |
|-----------------|--------------------------|

---

### Description

Assigns physical units to channels.

### Usage

```
setChannelUnits(x, units)
```

### Arguments

|       |                                          |
|-------|------------------------------------------|
| x     | A <code>PhysioExperiment</code> object.  |
| units | Character vector or named list of units. |

### Value

Modified `PhysioExperiment` object.

### References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

### See Also

[setChannelTypes](#) for assigning channel types, [channelInfo](#) for full channel metadata

### Examples

```
pe <- PhysioExperiment(  
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),  
  samplingRate = 100  
)  
  
# Set all channels to same unit  
pe <- setChannelUnits(pe, "uV")
```

---

setElectrodePositions *Set electrode positions*


---

### Description

Assigns 3D electrode positions to channels.

### Usage

```
setElectrodePositions(x, positions)
```

### Arguments

|           |                                                                                                                            |
|-----------|----------------------------------------------------------------------------------------------------------------------------|
| x         | A <code>PhysioExperiment</code> object.                                                                                    |
| positions | A <code>data.frame</code> or matrix with columns x, y, z and rows matching channels. Row names should match channel names. |

### Value

Modified `PhysioExperiment` object.

### References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

### See Also

[getElectrodePositions](#) for reading positions, [applyMontage](#) for standard electrode layouts, [channelInfo](#) for full channel metadata

### Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(300), nrow = 100, ncol = 3)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz")),
  samplingRate = 100
)

# Set electrode positions
positions <- data.frame(
  x = c(0, 0, 0),
  y = c(0.71, 0, -0.71),
  z = c(0.71, 1, 0.71)
)
pe <- setElectrodePositions(pe, positions)
```

---

|           |                                                 |
|-----------|-------------------------------------------------|
| setEvents | <i>Set events for a PhysioExperiment object</i> |
|-----------|-------------------------------------------------|

---

### Description

Set events for a PhysioExperiment object

### Usage

```
setEvents(x, events)
```

### Arguments

|        |                                                                                   |
|--------|-----------------------------------------------------------------------------------|
| x      | A PhysioExperiment object.                                                        |
| events | A PhysioEvents object or a data.frame with columns: onset, duration, type, value. |

### Value

The modified PhysioExperiment object.

### References

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

### See Also

[getEvents](#) for retrieving events, [addEvents](#) for appending events, [PhysioEvents](#) for the event constructor

### Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(1000), nrow = 100)),
  samplingRate = 100
)

# Set events using PhysioEvents object
events <- PhysioEvents(onset = c(1, 2, 3), type = "stimulus")
pe <- setEvents(pe, events)

# Set events using data.frame
pe <- setEvents(pe, data.frame(onset = c(1, 2), type = "response"))
```

---

|              |                                |
|--------------|--------------------------------|
| setReference | <i>Set reference electrode</i> |
|--------------|--------------------------------|

---

## Description

Records the reference electrode used for the recording.

## Usage

```
setReference(x, reference)
```

## Arguments

|           |                                                  |
|-----------|--------------------------------------------------|
| x         | A <code>PhysioExperiment</code> object.          |
| reference | Character string naming the reference electrode. |

## Value

Modified `PhysioExperiment` object.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[getReference](#) for reading the reference, [channelInfo](#) for full channel metadata, [applyMontage](#) for applying standard montages

## Examples

```
pe <- PhysioExperiment(  
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),  
  samplingRate = 100  
)  
  
# Set reference electrode  
pe <- setReference(pe, "average")  
getReference(pe) # "average"
```

---

show,PhysioEvents-method

*Show method for PhysioEvents*

---

### Description

Show method for PhysioEvents

### Usage

```
## S4 method for signature 'PhysioEvents'  
show(object)
```

### Arguments

object                    A PhysioEvents object.

### Value

Returns NULL invisibly; called for the side effect of printing a compact summary of the events to the console.

### See Also

[PhysioEvents](#) for the constructor, [nEvents](#) for event count

---

show,PhysioExperiment-method

*S4 Methods for PhysioExperiment*

---

### Description

Standard S4 methods for PhysioExperiment objects including show, subsetting, and combining.  
Show method for PhysioExperiment

### Usage

```
## S4 method for signature 'PhysioExperiment'  
show(object)
```

### Arguments

object                    A PhysioExperiment object.

### Details

Displays a summary of the PhysioExperiment object.

**Value**

Invisibly returns NULL. Called for its side effect of printing a human-readable summary to the console.

**References**

Huber, W., et al. (2015). "Orchestrating high-throughput genomic analysis with Bioconductor." *Nature Methods*, 12(2), 115-121. doi:10.1038/nmeth.3252

Morgan, M., et al. (2022). "S4Vectors: Foundation of vector-like and list-like containers in Bioconductor." R package.

**See Also**

[PhysioExperiment](#) for the constructor, [summary,PhysioExperiment-method](#) for channel-level statistics, [as.data.frame,PhysioExperiment-method](#) for conversion to data.frame

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)
pe # Displays summary
```

---

streamCoverage

*Coverage of a time span by a stream's samples*


---

**Description**

Reports how much of [start, end] a stream actually covers. The distinction the return value keeps is between what was **observed** and what was **derived**: n\_present is a count of real samples, while n\_expected and coverage depend on an assumed sampling scheme and are NA when that scheme cannot decide them.

**Usage**

```
streamCoverage(x, stream, start, end, rule = "nominal", gap_factor = 1.5)
```

**Arguments**

|            |                                                                                                                         |
|------------|-------------------------------------------------------------------------------------------------------------------------|
| x          | A MultiPhysioExperiment.                                                                                                |
| stream     | Stream name.                                                                                                            |
| start, end | Bounds in seconds on the shared clock (closed interval, the same meaning and tolerance as <a href="#">timeWindow</a> ). |

|            |                                                                                                                                                                                                                                                                   |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rule       | Name of the rule for n_expected. "nominal" counts the grid points of step 1/rate inside the interval, anchored on the stream's first sample time and using the same tolerance as <code>timeWindow</code> . Any other value reports n_expected and coverage as NA. |
| gap_factor | Multiple of the nominal period above which a step counts as a gap.                                                                                                                                                                                                |

### Details

Gaps are reported as intervals, not as a shortfall in a count, so a caller can see *where* the recording stops. A step between measured sample times larger than gap\_factor times the nominal period counts as a gap; that factor is an adopted criterion shared with **PhysioCrossModal**'s interpolation guard, and it separates acquisition jitter from a dropped sample rather than proving that no gap exists.

For a stream with no measured times the sample times are reconstructed from the start offset and the rate, so no gap can be found. That is a consequence of the reconstruction, not a measurement, and rule\$basis says so.

n\_expected counts the points of a reference grid that fall inside the interval. It is not (end - start) \* rate, which measures a span rather than a grid and overstates the count whenever the bounds do not land on sample times – a whole, evenly sampled stream would then read as incomplete for being asked about off its own sample times. The grid has step 1/rate and is anchored on the stream's first sample time; for a measured stream that anchor is an assumption, and rule\$anchor, rule\$step and rule\$basis report which grid was used.

### Value

A list with n\_present, n\_expected, coverage, gaps (a data frame of start/end) and rule (its name, gap\_factor, the grid anchor and step, and the basis the expected count rests on).

### See Also

`timeWindow`, `hasMeasuredTimes`

### Examples

```
pe <- PhysioExperiment(assays = list(raw = matrix(1:5, ncol = 1)),
  samplingRate = 10,
  rowData = S4Vectors::DataFrame(
    time_from_t0 = c(0, 0.1, 0.2, 1.2, 1.3)))
streamCoverage(MultiPhysioExperiment(a = pe), "a", 0, 1.3)
```

---

|             |                             |
|-------------|-----------------------------|
| streamNames | <i>Names of the streams</i> |
|-------------|-----------------------------|

---

### Description

Names of the streams

**Usage**

streamNames(x)

**Arguments**

x                      A MultiPhysioExperiment.

**Value**

Character vector of stream names.

---

|             |                                  |
|-------------|----------------------------------|
| streamRates | <i>Per-stream sampling rates</i> |
|-------------|----------------------------------|

---

**Description**

Per-stream sampling rates

**Usage**

streamRates(x)

**Arguments**

x                      A MultiPhysioExperiment.

**Value**

Named numeric of sampling rates (Hz), one per stream.

---

|         |                                                      |
|---------|------------------------------------------------------|
| streams | <i>Access the streams of a MultiPhysioExperiment</i> |
|---------|------------------------------------------------------|

---

**Description**

Access the streams of a MultiPhysioExperiment

**Usage**

```
streams(x)

## S4 method for signature 'MultiPhysioExperiment'
streams(x)

streams(x) <- value

## S4 replacement method for signature 'MultiPhysioExperiment'
streams(x) <- value
```

**Arguments**

|       |                                                        |
|-------|--------------------------------------------------------|
| x     | A MultiPhysioExperiment.                               |
| value | A named list / SimpleList of PhysioExperiment streams. |

**Value**

streams() a SimpleList; setter returns the updated object.

---

|                 |                                                     |
|-----------------|-----------------------------------------------------|
| streamTimeIndex | <i>Sample times of a stream on the shared clock</i> |
|-----------------|-----------------------------------------------------|

---

**Description**

Returns  $\text{offset} + (i - 1) / \text{samplingRate}$  for each sample, i.e. seconds *relative to* clock\$t0. It deliberately does not add t0; use commonClock(x)\$t0 when an absolute time is needed.

**Usage**

```
streamTimeIndex(x, stream = NULL)
```

**Arguments**

|        |                                                        |
|--------|--------------------------------------------------------|
| x      | A MultiPhysioExperiment.                               |
| stream | Stream name (or NULL for a named list of all streams). |

**Value**

Numeric time vector (seconds from t0) for the stream, or a named list of such vectors when stream is NULL.

**See Also**

[timeWindow](#), [resampleToCommon](#)

---

|         |                                                              |
|---------|--------------------------------------------------------------|
| subject | <i>Retrieve a single subject's longitudinal record by id</i> |
|---------|--------------------------------------------------------------|

---

**Description**

Retrieve a single subject's longitudinal record by id

**Usage**

```
subject(x, id)
```

**Arguments**

|    |                      |
|----|----------------------|
| x  | A PhysioCohort.      |
| id | A subject id (name). |

**Value**

The subject's PhysioLongitudinal.

**See Also**

[subjects](#)

---

|             |                               |
|-------------|-------------------------------|
| subjectData | <i>Subject-level metadata</i> |
|-------------|-------------------------------|

---

**Description**

Subject-level metadata

**Usage**

```
subjectData(x)

## S4 method for signature 'PhysioLongitudinal'
subjectData(x)

subjectData(x) <- value

## S4 replacement method for signature 'PhysioLongitudinal'
subjectData(x) <- value
```

**Arguments**

- x                    A `PhysioLongitudinal`.
- value               A one-row subject `DataFrame` (setter).

**Value**

subjectData() the subject `DataFrame`.

---

|            |                                                                                   |
|------------|-----------------------------------------------------------------------------------|
| subjectIds | <i>Subject ids, subject count and group labels of a <code>PhysioCohort</code></i> |
|------------|-----------------------------------------------------------------------------------|

---

**Description**

Subject ids, subject count and group labels of a `PhysioCohort`

**Usage**

- subjectIds(x)
- nSubjects(x)
- groups(x)

**Arguments**

- x                    A `PhysioCohort`.

**Value**

subjectIds() a character vector; nSubjects() an integer; groups() the `colData$group` vector (or `NULL`).

---

|          |                                                |
|----------|------------------------------------------------|
| subjects | <i>Subjects of a <code>PhysioCohort</code></i> |
|----------|------------------------------------------------|

---

**Description**

Subjects of a `PhysioCohort`

**Usage**

```

subjects(x)

## S4 method for signature 'PhysioCohort'
subjects(x)

subjects(x) <- value

## S4 replacement method for signature 'PhysioCohort'
subjects(x) <- value

```

**Arguments**

x                    A PhysioCohort.  
 value                A named list / SimpleList of PhysioLongitudinal (setter).

**Value**

subjects() a SimpleList of PhysioLongitudinal.

---

|               |                                                    |
|---------------|----------------------------------------------------|
| subjectTrials | <i>Every trial of one subject, across sessions</i> |
|---------------|----------------------------------------------------|

---

**Description**

Lists the trial tables of all sessions in a PhysioLongitudinal, carrying the session each row came from. Without that column the rows would be ambiguous: the same trial\_id legitimately recurs in different sessions.

**Usage**

```
subjectTrials(x)
```

**Arguments**

x                    A PhysioLongitudinal.

**Details**

A hierarchy with no trials returns an empty table with the same columns, not an error – "this subject has no trials recorded" is an answer.

**Value**

A data frame of trials with a session column.

**See Also**

[cohortTrials](#), [trials](#)

---

|              |                                                                   |
|--------------|-------------------------------------------------------------------|
| subsetCohort | <i>Subset a PhysioCohort by group (or a predicate on colData)</i> |
|--------------|-------------------------------------------------------------------|

---

**Description**

Subset a PhysioCohort by group (or a predicate on colData)

**Usage**

```
subsetCohort(x, group = NULL, subset = NULL)
```

**Arguments**

|        |                                                                                                                               |
|--------|-------------------------------------------------------------------------------------------------------------------------------|
| x      | A PhysioCohort.                                                                                                               |
| group  | Optional character vector of group label(s) to keep.                                                                          |
| subset | Optional logical vector (length = number of subjects) or an expression evaluated in cohortData(x) selecting subjects to keep. |

**Value**

A PhysioCohort with the selected subjects.

**See Also**

[PhysioCohort](#)

---

|                                 |                                                |
|---------------------------------|------------------------------------------------|
| summary,PhysioExperiment-method | <i>Summary statistics for PhysioExperiment</i> |
|---------------------------------|------------------------------------------------|

---

**Description**

Computes per-channel summary statistics (min, max, mean, sd, median) for the default assay. For 3D arrays, values are first averaged across the third dimension.

**Usage**

```
## S4 method for signature 'PhysioExperiment'
summary(object, ...)
```

**Arguments**

|        |                                  |
|--------|----------------------------------|
| object | A PhysioExperiment object.       |
| ...    | Additional arguments (not used). |

**Value**

A `data.frame` with columns `channel`, `min`, `max`, `mean`, `sd`, and `median`, with one row per channel. Returns an empty `data.frame` if no assays are present.

**References**

R Core Team (2024). "R: A Language and Environment for Statistical Computing." R Foundation for Statistical Computing, Vienna, Austria.

**See Also**

[PhysioExperiment](#) for the constructor, [as.data.frame,PhysioExperiment-method](#) for full data export, [channelNames](#) for channel labels

**Examples**

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(400), nrow = 100, ncol = 4)),
  colData = S4Vectors::DataFrame(label = c("Fz", "Cz", "Pz", "Oz")),
  samplingRate = 100
)
summary(pe)
```

---

 theme\_physio

---

*A clean, accessible ggplot2 theme for x-biosignal figures*


---

**Description**

A clean, accessible ggplot2 theme for x-biosignal figures

**Usage**

```
theme_physio(base_size = 11, base_family = "")
```

**Arguments**

`base_size`, `base_family`  
 Passed to the underlying base theme.

**Value**

A ggplot2 theme object.

**Examples**

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
    ggplot2::geom_point() + theme_physio()
}
```

---

|           |                          |
|-----------|--------------------------|
| timeIndex | <i>Time index helper</i> |
|-----------|--------------------------|

---

## Description

Computes a time vector for the default assay using the object's sampling rate.

## Usage

```
timeIndex(x)
```

## Arguments

`x` A `PhysioExperiment` instance.

## Value

Numeric vector of time points in seconds.

## References

Oostenveld R, Fries P, Maris E, Schoffelen JM (2011). "FieldTrip: Open source software for advanced analysis of MEG, EEG, and invasive electrophysiological data." *Computational Intelligence and Neuroscience*, 2011, 156869.

## See Also

[samplingRate](#) for the sampling rate, [duration](#) for signal duration, [timeToSamples](#) for converting times to sample indices

## Examples

```
pe <- PhysioExperiment(
  assays = list(raw = matrix(rnorm(40), nrow = 10, ncol = 4)),
  samplingRate = 100
)
head(timeIndex(pe))
```

---

|               |                                              |
|---------------|----------------------------------------------|
| timeToSamples | <i>Convert event times to sample indices</i> |
|---------------|----------------------------------------------|

---

## Description

Convert event times to sample indices

## Usage

```
timeToSamples(x, times)
```

## Arguments

|       |                                         |
|-------|-----------------------------------------|
| x     | A <code>PhysioExperiment</code> object. |
| times | Numeric vector of times in seconds.     |

## Value

Integer vector of sample indices.

## References

Delorme A, Makeig S (2004). "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics." *Journal of Neuroscience Methods*, 134(1), 9-21.

## See Also

[samplesToTime](#) for the inverse conversion, [samplingRate](#) for the sampling rate, [timeIndex](#) for the full time vector

## Examples

```
pe <- PhysioExperiment(  
  assays = list(raw = matrix(rnorm(1000), nrow = 100)),  
  samplingRate = 100  
)  
timeToSamples(pe, c(0, 0.5, 1))
```

---

|            |                                         |
|------------|-----------------------------------------|
| timeWindow | Select a time window across all streams |
|------------|-----------------------------------------|

---

### Description

Selects, from every stream, the samples whose time on the shared clock falls in the closed interval `[start, end]`, expressed in seconds relative to `clock$t0`. Selection uses each sample's actual time, so a stream's offset is honoured rather than ignored.

### Usage

```
timeWindow(
  x,
  start,
  end,
  streams = NULL,
  drop_empty = FALSE,
  complete_only = FALSE
)
```

### Arguments

|                            |                                                                                                                                                                                                        |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>x</code>             | A <code>MultiPhysioExperiment</code> .                                                                                                                                                                 |
| <code>start, end</code>    | Window bounds in seconds relative to <code>clock\$t0</code> (closed interval, <code>start &lt;= end</code> ).                                                                                          |
| <code>streams</code>       | Optional character vector of stream names to keep.                                                                                                                                                     |
| <code>drop_empty</code>    | Exclude non-overlapping streams instead of failing.                                                                                                                                                    |
| <code>complete_only</code> | Keep only trials lying wholly inside the window. By default a trial overlapping the window is kept, truncated to it and marked, because the trial happened; the excluded ones are recorded either way. |

### Details

`t0` is preserved and each stream's offset is updated to the time of its first retained sample, so the result sits on the same clock as the input. The input object is not modified.

A stream that does not overlap the window is an error by default, naming the streams concerned: silently returning a container with fewer streams than asked for has been a source of wrong answers. Pass `drop_empty = TRUE` to exclude them deliberately; the exclusion and its reason are then recorded in the result's provenance.

### Value

A `MultiPhysioExperiment` carrying the selected samples.

### See Also

[streamTimeIndex](#), [resampleToCommon](#)

**Examples**

```

a <- PhysioExperiment(S4Vectors::SimpleList(raw = matrix(rnorm(200), 100, 2)),
  samplingRate = 100)
b <- PhysioExperiment(S4Vectors::SimpleList(raw = matrix(rnorm(500), 250, 2)),
  samplingRate = 250)
mpe <- MultiPhysioExperiment(streams = list(a = a, b = b),
  offsets = c(a = 0, b = 0.2))
w <- timeWindow(mpe, 0.3, 0.5)
vapply(streamTimeIndex(w), function(t) t[1], numeric(1))

```

---

|       |                          |
|-------|--------------------------|
| trial | <i>Extract one trial</i> |
|-------|--------------------------|

---

**Description**

Selects the trial's span with [timeWindow](#), so the selection, the boundary tolerance, the clock and the provenance all follow the same rules as any other window.

**Usage**

```
trial(x, id, streams = NULL, ...)
```

**Arguments**

|         |                                        |
|---------|----------------------------------------|
| x       | A MultiPhysioExperiment.               |
| id      | Trial identifier.                      |
| streams | Optional stream names to keep.         |
| ...     | Passed to <a href="#">timeWindow</a> . |

**Value**

A MultiPhysioExperiment covering the trial.

**See Also**

[trialIntervals](#), [timeWindow](#)

---

|               |                                                           |
|---------------|-----------------------------------------------------------|
| trialCoverage | <i>Coverage of one trial across every stream it names</i> |
|---------------|-----------------------------------------------------------|

---

### Description

A trial is usually captured by more than one stream, and they need not cover it equally: one may have a dropout the others do not. This reports [streamCoverage](#) for each stream the trial's intervals name, so a caller can see whether a cross-stream comparison over that trial is defensible before making one.

### Usage

```
trialCoverage(x, id, rule = "nominal", gap_factor = 1.5, tolerance = 1e-06)
```

### Arguments

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| x                | A MultiPhysioExperiment carrying trial intervals.              |
| id               | Trial identifier.                                              |
| rule, gap_factor | Passed to <a href="#">streamCoverage</a> .                     |
| tolerance        | Coverage difference below which the streams count as agreeing. |

### Details

The returned table carries a consistent attribute: FALSE when the streams do not agree about how much of the trial they cover. Disagreement is surfaced rather than resolved – which stream to trust is not something this function can decide.

### Value

A data frame with one row per stream: stream, start, end, n\_present, n\_expected, coverage, n\_gaps and n\_unusable\_carried\_gaps – dropouts an earlier window recorded that this query cannot re-derive, and which therefore leave completeness undecided rather than confirmed.

### See Also

[streamCoverage](#), [trialIntervals](#)

---

|                |                                    |
|----------------|------------------------------------|
| trialIntervals | <i>Trial intervals on a stream</i> |
|----------------|------------------------------------|

---

### Description

The time span each trial occupies, per recording and per stream. Because the same trial can be captured by several devices with different spans, the interval table is keyed on (subject\_id, session\_id, trial\_id, recording\_id, stream).

### Usage

```
trialIntervals(x)

trialIntervals(x, allow_overlap = FALSE) <- value
```

### Arguments

|               |                                                                                                                                                                                                                                                                                                                            |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| x             | A MultiPhysioExperiment.                                                                                                                                                                                                                                                                                                   |
| allow_overlap | Overlapping intervals on the same recording and stream are refused by default, because two trials claiming the same samples is usually a mistake in the table rather than a property of the experiment. Pass a non-empty character reason to allow them; the reason is kept on the table so the decision is visible later. |
| value         | A data frame with the five key columns plus start and end in seconds on the shared clock.                                                                                                                                                                                                                                  |

### Details

An interval that contains a gap **is stored**: the trial happened, and refusing it would erase that. Whether the span can be analysed is a separate question, answered by [streamCoverage](#) at the point of use. Invalid input is refused: reversed or non-finite bounds, an unknown stream, a duplicate key, or an interval lying wholly outside the recording.

### Value

The interval table, or the modified container.

### See Also

[trials](#), [streamCoverage](#)

---

|        |                                       |
|--------|---------------------------------------|
| trials | <i>Trials recorded in a container</i> |
|--------|---------------------------------------|

---

### Description

The trial table records that a trial happened. It is keyed on (subject\_id, session\_id, trial\_id) and is unique on that composite: trial\_id alone is not assumed unique across subjects.

### Usage

```
trials(x)
```

```
trials(x) <- value
```

### Arguments

|       |                                                                                           |
|-------|-------------------------------------------------------------------------------------------|
| x     | A MultiPhysioExperiment.                                                                  |
| value | A data frame with subject_id, session_id, trial_id and optionally condition and observed. |

### Details

A trial that took place outside the recording is carried here with observed = FALSE rather than as an interval with out-of-range times.

### Value

The trial table, or the modified container.

### See Also

[trialIntervals](#), [trial](#)

---

|          |                                                            |
|----------|------------------------------------------------------------|
| trialSet | <i>Hold several independently recorded trials together</i> |
|----------|------------------------------------------------------------|

---

### Description

When each trial is its own recording, the trials belong to one session but the signals live in separate containers. A trial set holds those containers alongside the map that says which trial each one carries and how the recordings are placed relative to one another.

### Usage

```
trialSet(recordings = list(), map = NULL)
```

**Arguments**

|            |                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------|
| recordings | A named list of MultiPhysioExperiment objects, named by recording_id.                                                            |
| map        | A data frame with subject_id, session_id, trial_id, recording_id, time_correspondence and, for "measured", reference and offset. |

**Details**

**No common time axis is fabricated.** Each recording keeps its own clock, and the second does not start where the first left off. Whether elapsed time between trials can be computed is decided by time\_correspondence in the map, exactly as for [recordingMap](#) on a single container.

Every recording\_id in the map must name a recording that is actually held; a map referring to a recording kept elsewhere is the situation this structure exists to end.

**Value**

An object of class physio\_trial\_set.

**See Also**

[recordings](#), [recordingMap](#), [elapsedBetweenTrials](#)

**Examples**

```
rec <- function() MultiPhysioExperiment(emg = PhysioExperiment(
  assays = list(raw = matrix(rnorm(100), 50, 2)), samplingRate = 50))
ts <- trialSet(
  recordings = list(R1 = rec(), R2 = rec()),
  map = data.frame(subject_id = "P1", session_id = "S1",
    trial_id = c("T1", "T2"), recording_id = c("R1", "R2"),
    time_correspondence = "measured", reference = "trigger",
    offset = c(0, 12.5)))
elapsedBetweenTrials(ts, "T1", "T2")
```

---

|                    |                                                       |
|--------------------|-------------------------------------------------------|
| watsonWilliamsTest | <i>Watson-Williams test for equal mean directions</i> |
|--------------------|-------------------------------------------------------|

---

**Description**

The circular analogue of a one-way ANOVA: tests whether two or more groups of angles share a common mean direction (assuming von Mises data of similar concentration). Use it to compare, e.g., coupling angles between limbs, speeds or groups.

**Usage**

```
watsonWilliamsTest(angles, group = NULL, units = c("degrees", "radians"))
```

Arguments

|        |                                                                                                                  |
|--------|------------------------------------------------------------------------------------------------------------------|
| angles | Numeric vector of angles, or a list of per-group angle vectors.                                                  |
| group  | Grouping factor (length length(angles)); required when angles is a single vector, ignored when angles is a list. |
| units  | "degrees" (default) or "radians".                                                                                |

Value

an htest: F statistic (concentration-corrected), df and p.value. Warns when the pooled concentration is low ( $\bar{r} < 0.45$ ), where the test is unreliable.

References

Watson GS, Williams EJ (1956); Zar JH (1999) eq. 27.9.

See Also

[rayleighTest\(\)](#)

Examples

```
set.seed(2)
g1 <- rnorm(25, 20, 10); g2 <- rnorm(25, 60, 10)
watsonWilliamsTest(list(g1, g2))$p.value           # differ -> small
```

---

|             |                                                                                |
|-------------|--------------------------------------------------------------------------------|
| waveformCMC | <i>Coefficient of multiple correlation for repeated waveforms (Kadaba CMC)</i> |
|-------------|--------------------------------------------------------------------------------|

---

Description

The waveform analogue of a reliability coefficient (Kadaba et al. 1989): how similar a set of repeated curves are, relative to the curve’s own variation. With groups = NULL the rows of x are repeated trials and the result is the **within-day** repeatability; pass groups (e.g. a session/day label per row) for the **between-day** CMC of the session-mean waveforms.

Usage

```
waveformCMC(x, groups = NULL)
```

Arguments

|        |                                                                                                                                        |
|--------|----------------------------------------------------------------------------------------------------------------------------------------|
| x      | A trials x frames numeric matrix: each row one waveform (e.g. a joint angle over the time-normalised cycle), columns the cycle points. |
| groups | Optional grouping (length nrow(x)); when given, the CMC is computed between the group-mean waveforms (between-day/-condition).         |

## Value

The CMC in [0, 1] (higher = more repeatable), or NA with a warning when the radicand is negative (waveform variation too small for a defined CMC – a known limitation for low-excursion curves).

## References

Kadaba MP, et al. (1989) J Orthop Res 7:849-860.

## See Also

[waveformICC\(\)](#), [waveformReliability\(\)](#)

## Examples

```
base <- sin(seq(0, 2 * pi, length.out = 101))
trials <- t(sapply(1:5, function(i) base + rnorm(101, 0, 0.02)))
waveformCMC(trials) # ~ 1 (repeatable)
```

---

waveformICC

*Pointwise intraclass correlation across the movement cycle*

---

## Description

The reliability *curve*: the between-subject ICC computed frame-by-frame from repeated trials, using the ecosystem's [icc\(\)](#) at each cycle point. It shows where along the movement a measurement is reliable.

## Usage

```
waveformICC(x, model = "twoway", type = "agreement", unit = "single")
```

## Arguments

**x** A [subject, trial, frame] array, or a list of trials x frames matrices (one per subject). Every subject must have the same trials and frames.

**model, type, unit** Passed to [icc\(\)](#) (default two-way, absolute agreement, single measure – ICC(2,1)).

## Value

a waveform\_icc list: icc, ci\_lower, ci\_upper (per frame), mean\_icc, min\_icc.

## See Also

[waveformReliability\(\)](#), [waveformCMC\(\)](#)

## Examples

```
set.seed(1)
subs <- lapply(1:8, function(s) {
  base <- s + sin(seq(0, 2 * pi, length.out = 101)) # subject offset = signal
  t(sapply(1:3, function(k) base + rnorm(101, 0, 0.1)))
})
waveformICC(subs)$mean_icc # high (subjects separable)
```

---

waveformReliability      *Waveform reliability report: CMC, ICC / SEM / MDC curves*

---

## Description

The integrated movement-waveform reliability summary. It combines the within-subject repeatability (per-subject Kadaba [waveformCMC\(\)](#)) with the between-subject pointwise ICC curve and the frame-by-frame standard error of measurement and minimal detectable change (from [sem\(\)](#) / [mdc\(\)](#), the same definitions as the scalar clinimetrics).

## Usage

```
waveformReliability(x, model = "twoway", type = "agreement", confidence = 0.95)
```

## Arguments

|             |                                                                                           |
|-------------|-------------------------------------------------------------------------------------------|
| x           | A [subject, trial, frame] array, or a list of trials x frames matrices (one per subject). |
| model, type | Passed to <a href="#">icc()</a> for the pointwise ICC.                                    |
| confidence  | Confidence level for the MDC (default 0.95).                                              |

## Value

a waveform\_reliability object with per-subject cmc, the icc, sem, mdc curves (+ ICC CIs), and scalar summaries (mean\_cmc, mean\_icc, peak\_mdc, ...). The SEM curve is the between-subject SD at each frame times  $\sqrt{1 - \text{ICC}}$ ; the MDC curve is the 95% minimal detectable change in the measurement's own units.

## References

Kadaba MP, et al. (1989) J Orthop Res 7:849-860; Weir JP (2005) J Strength Cond Res 19:231-240 (SEM/MDC).

## See Also

[waveformCMC\(\)](#), [waveformICC\(\)](#)

## Examples

```
set.seed(1)
subs <- lapply(1:10, function(s) {
  base <- s + 5 * sin(seq(0, 2 * pi, length.out = 101))
  t(sapply(1:3, function(k) base + rnorm(101, 0, 0.3)))
})
r <- waveformReliability(subs)
r$mean_cmc; r$mean_icc; r$peak_mdc
```

---

|                |                                                                |
|----------------|----------------------------------------------------------------|
| withProvenance | Carry a provenance log onto a derived object and record a step |
|----------------|----------------------------------------------------------------|

---

## Description

When an operation constructs a NEW `PhysioExperiment` from an input (rather than modifying it in place), use this to copy the input's provenance onto the result and append the current step in one call.

## Usage

```
withProvenance(
  from,
  to,
  step,
  params = list(),
  package = NA_character_,
  version = NA_character_
)
```

## Arguments

|                                |                                                               |
|--------------------------------|---------------------------------------------------------------|
| from                           | The source object whose provenance should be carried forward. |
| to                             | The newly derived object.                                     |
| step, params, package, version | As in <a href="#">logStep</a> .                               |

## Value

to carrying from's provenance plus the new step.

## See Also

[recordActivity](#) for the expression-wrapping form.

## Examples

```
from <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
from <- logStep(from, "import")
to <- PhysioExperiment(
  S4Vectors::SimpleList(raw = matrix(rnorm(20), 10, 2)), samplingRate = 100)
to <- withProvenance(from, to, "resample", params = list(to = 50))
provenance(to)
```

---

wpliEstimate

*Weighted phase-lag index estimator (raw and debiased)*


---

## Description

Computes the weighted phase-lag index (wPLI) and its unbiased (debiased) estimator from the imaginary part of a cross-spectrum, following Vinck et al. (2011, Eq. 6). This is the single source of the wPLI debiasing math shared by **PhysioEEG** (eegWPLI) and **PhysioCrossModal** (weightedPLI), so the two agree exactly on identical input.

## Usage

```
wpliEstimate(imag, debiased = TRUE)
```

## Arguments

|          |                                                              |
|----------|--------------------------------------------------------------|
| imag     | Numeric vector of imaginary cross-spectrum values.           |
| debiased | Logical; also compute the debiased estimator (default TRUE). |

## Details

The estimators, over the imaginary cross-spectrum values  $X_j$  (one per window, taper, or time sample), are

$$\text{wPLI} = \frac{|\sum_j X_j|}{\sum_j |X_j|}$$

$$\text{debiased wPLI} = \frac{(\sum_j X_j)^2 - \sum_j X_j^2}{(\sum_j |X_j|)^2 - \sum_j X_j^2}.$$

The debiased numerator and denominator subtract the diagonal self-terms, so the estimator is unbiased: for independent signals it is distributed around 0 rather than being inflated toward positive values.

## Value

A list with `wpli` (raw wPLI, from 0 to 1), `wpli_debiased` (the debiased estimator, or NA if debiased = FALSE or fewer than two values), and `n` (the number of values used).

**References**

Vinck, M., Oostenveld, R., van Wingerden, M., Battaglia, F., & Pennartz, C.M.A. (2011). "An improved index of phase-synchronization for electrophysiological data in the presence of volume-conduction, noise and sample-size bias." *NeuroImage*, 55(4), 1548-1565. doi:[10.1016/j.neuroimage.2011.01.055](https://doi.org/10.1016/j.neuroimage.2011.01.055)

**Examples**

```
set.seed(1)
wpliEstimate(rnorm(500))
```

# Index

- \* **lifecycle**
  - deprecate\_physio, 41
- \* **plugin-api**
  - registerOperation, 110
  - registerReader, 111
  - registerWriter, 112
- .fitMVAR, 78, 79
- .mvarSpectral, 79, 80
- [,MultiPhysioExperiment,ANY,ANY,ANY-method, 5
- [,PhysioCohort,ANY,ANY,ANY-method (PhysioCohort-methods), 91
- [,PhysioExperiment,ANY,ANY,ANY-method, 6
- [,PhysioLongitudinal,ANY,ANY,ANY-method (PhysioLongitudinal-methods), 97
- [[,MultiPhysioExperiment,ANY,ANY-method (MultiPhysioExperiment-methods), 77
- addEvents, 7, 59, 82, 113, 126
- addProvenance, 8
- addSession, 9, 96, 97
- addSubject, 10, 90
- aggregatePhysioFeatures, 11, 13–15, 51
- aggregateTrials, 13, 15, 16, 51
- aggregateTrialWaveforms, 14, 51
- alignment, 76
- alignment
  - (multiphysio-legacy-accessors), 73
- alignment,MultiPhysioExperiment-method (multiphysio-legacy-accessors), 73
- alignment<-
  - (multiphysio-legacy-accessors), 73
- alignment<-,MultiPhysioExperiment-method (multiphysio-legacy-accessors), 73
- alignStreams, 16, 76, 116
- AnalysisResult, 17, 88
- AnalysisResult-accessors, 18
- AnalysisResult-carrier, 19
- AnalysisResult-class, 20
- appendProvenance, 21, 106
- applyMontage, 22, 58, 64, 125, 127
- as.data.frame,PhysioExperiment-method, 23
- as.data.frame.PhysioBiomarker, 24
- asMSKTracker, 24, 30
- asMultiAssayExperiment, 25
- assaySamplingRates, 26, 122
- availableOperations
  - (registerOperation), 110
- availableReaders (registerReader), 111
- availableWriters (registerWriter), 112
- base::data.frame(), 24
- BiocGenerics::as.data.frame(), 86
- biomarkerValue
  - (AnalysisResult-accessors), 18
- biomarkerValue,PhysioBiomarker-method (AnalysisResult-accessors), 18
- blandAltman, 27
- blandAltman(), 68
- cbindPhysio, 28, 104
- changeScores, 24, 25, 29
- channelInfo, 30, 32–34, 60, 81, 94, 95, 114, 123–125, 127
- channelInfo<-, 31
- channelNames, 22, 23, 31, 32, 32, 34, 58, 81, 99, 114, 137
- channelNames<-, 33
- checkNA, 34, 61, 66, 81, 115
- circularLinearCorrelation, 35
- circularLinearCorrelation(), 36
- circularSummary, 36

- circularSummary(), [35](#), [103](#)
- cohensD, [36](#)
- cohensD(), [48](#)
- cohortData, [38](#), [39](#)
- cohortData, [PhysioCohort-method](#)  
(cohortData), [38](#)
- cohortData<- (cohortData), [38](#)
- cohortData<- , [PhysioCohort-method](#)  
(cohortData), [38](#)
- cohortDesign, [38](#), [90](#), [91](#)
- cohortTrials, [39](#), [135](#)
- commonClock, [40](#)
  
- defaultAssay, [40](#), [44](#), [94](#), [118](#)
- deprecate\_physio, [41](#)
- design, [96](#)
- design, [PhysioLongitudinal-method](#), [42](#)
- design<- , [PhysioLongitudinal-method](#)  
(design, [PhysioLongitudinal-method](#)),  
[42](#)
- detectGaps, [43](#)
- dim, [MultiPhysioExperiment-method](#)  
([MultiPhysioExperiment-methods](#)),  
[77](#)
- dim, [PhysioExperiment-method](#), [44](#)
- dropChannels, [7](#), [28](#), [45](#), [57](#), [99](#)
- duration, [46](#), [52](#), [70](#), [118](#), [138](#)
  
- elapsedBetweenTrials, [47](#), [107](#), [108](#), [145](#)
- estimandOf (AnalysisResult-carrier), [19](#)
- estimandOf, [AnalysisResult-method](#)  
(AnalysisResult-carrier), [19](#)
- estimateOf (AnalysisResult-carrier), [19](#)
- estimateOf, [AnalysisResult-method](#)  
(AnalysisResult-carrier), [19](#)
- etaSquared, [47](#)
- etaSquared(), [37](#)
- eventQuery, [49](#)
- EventQuery-class, [49](#)
- experiments  
(multiphysio-legacy-accessors),  
[73](#)
- experiments, [MultiPhysioExperiment-method](#)  
(multiphysio-legacy-accessors),  
[73](#)
- experiments<-  
(multiphysio-legacy-accessors),  
[73](#)
- experiments<- , [MultiPhysioExperiment-method](#)  
(multiphysio-legacy-accessors),  
[73](#)
- extractTrialFeatures, [50](#)
- extractTrialFeatures(), [11](#)
- extractWindow, [7](#), [46](#), [51](#), [104](#)
  
- fillEdgeNA, [52](#), [61](#)
- filterType, [53](#)
- filterValue, [54](#)
- format.PhysioBiomarker, [55](#)
- fPCA, [55](#)
- fPCA(), [100](#), [105](#), [109](#)
  
- getChannelsByType, [45](#), [57](#), [99](#), [123](#)
- getElectrodePositions, [22](#), [58](#), [125](#)
- getEvents, [8](#), [59](#), [82](#), [92](#), [113](#), [126](#)
- getOperation (registerOperation), [110](#)
- getReader (registerReader), [111](#)
- getReference, [60](#), [64](#), [127](#)
- getWriter (registerWriter), [112](#)
- groups (subjectIds), [134](#)
  
- handleNA, [34](#), [53](#), [61](#), [115](#)
- harmonize, [62](#), [63](#), [64](#)
- harmonizeChannels, [62](#), [63](#), [65](#)
- harmonizeMontage, [62](#), [64](#)
- harmonizeReference, [62](#), [64](#)
- harmonizeReport, [62](#), [63](#), [65](#)
- hasMeasuredTimes, [65](#), [130](#)
- hasNA, [34](#), [66](#), [81](#), [115](#)
  
- icc, [67](#), [89](#)
- icc(), [27](#), [71](#), [87](#), [120](#), [147](#), [148](#)
- is.PhysioBiomarker, [68](#)
- is.PhysioBiomarker(), [87](#)
  
- length, [MultiPhysioExperiment-method](#)  
([MultiPhysioExperiment-methods](#)),  
[77](#)
- length, [PhysioCohort-method](#)  
([PhysioCohort-methods](#)), [91](#)
- length, [PhysioExperiment-method](#), [69](#)
- length, [PhysioLongitudinal-method](#)  
([PhysioLongitudinal-methods](#)),  
[97](#)
- lifecycle::deprecate\_stop(), [41](#), [42](#)
- lifecycle::deprecate\_warn(), [41](#), [42](#)
- logStep, [8](#), [9](#), [70](#), [101](#), [102](#), [107](#), [149](#)

- mdc, [71, 89](#)
- mdc(), [68, 87, 120, 148](#)
- migrateContainer, [72, 105](#)
- modalities
  - (multiphysio-legacy-accessors), [73](#)
- modalities, MultiPhysioExperiment-method
  - (multiphysio-legacy-accessors), [73](#)
- multiphysio-legacy-accessors, [73](#)
- MultiPhysioExperiment, [72, 74, 76, 78](#)
- MultiPhysioExperiment-class, [76](#)
- MultiPhysioExperiment-methods, [77](#)
- MultiRatePhysioExperiment, [26, 96, 122](#)
- MultiRatePhysioExperiment
  - (MultiPhysioExperiment), [74](#)
- MultiRatePhysioExperiment-class, [77](#)
- mvarFit, [78, 79, 80](#)
- mvarOrderSelect, [78, 79](#)
- mvarTransfer, [78, 79](#)
- names, MultiPhysioExperiment-method
  - (MultiPhysioExperiment-methods), [77](#)
- naSummary, [34, 66, 80](#)
- nChannels, [31, 33, 44, 45, 70, 81](#)
- nEvents, [8, 59, 82, 92, 128](#)
- nModalities
  - (multiphysio-legacy-accessors), [73](#)
- nModalities, MultiPhysioExperiment-method
  - (multiphysio-legacy-accessors), [73](#)
- normativeLookup, [83](#)
- normativeLookup(), [87](#)
- NormativeReference, [83](#)
- NormativeReference(), [83](#)
- NormativeReference-class, [84](#)
- NormativeReference-compare, [85](#)
- nStreams, [86](#)
- nSubjects (subjectIds), [134](#)
- percentPredicted, [85](#)
- percentPredicted
  - (NormativeReference-compare), [85](#)
- percentPredicted, NormativeReference-method
  - (NormativeReference-compare), [85](#)
- PhysioBiomarker, [20, 24, 55, 69, 83, 86, 87, 88](#)
- physioBiomarker, [86](#)
- PhysioBiomarker(), [87](#)
- physioBiomarker(), [69, 83](#)
- PhysioBiomarker-class, [88](#)
- PhysioCohort, [10, 39, 89, 91, 136](#)
- PhysioCohort-class, [90](#)
- PhysioCohort-methods, [91](#)
- PhysioEvents, [82, 92, 126, 128](#)
- PhysioEvents-class, [93](#)
- PhysioExperiment, [41, 91, 93, 95, 96, 118, 129, 137](#)
- PhysioExperiment-class, [95](#)
- PhysioLongitudinal, [10, 25, 29, 30, 90, 91, 95, 97](#)
- PhysioLongitudinal-class, [96](#)
- PhysioLongitudinal-methods, [97](#)
- physioPalette, [98, 119](#)
- pickChannels, [7, 28, 45, 57, 63, 81, 99, 114](#)
- print.fPCA\_result, [100](#)
- provenance, [9, 20, 94, 100, 102](#)
- provenance, MultiPhysioExperiment-method
  - (provenance), [100](#)
- provenance, PhysioCohort-method
  - (provenance), [100](#)
- provenance, PhysioExperiment-method
  - (provenance), [100](#)
- provenance, PhysioLongitudinal-method
  - (provenance), [100](#)
- provenance<- (provenance), [100](#)
- provenance<- , PhysioExperiment-method
  - (provenance), [100](#)
- provenanceHash, [102](#)
- provenanceOf (AnalysisResult-carrier), [19](#)
- provenanceOf, AnalysisResult-method
  - (AnalysisResult-carrier), [19](#)
- rayleighTest, [102](#)
- rayleighTest(), [36, 146](#)
- rbindPhysio, [28, 103](#)
- readPhysioRDS, [72, 104](#)
- reconstructFPCA, [105](#)
- reconstructFPCA(), [56, 100](#)
- recordActivity, [101, 106, 149](#)
- recordingMap, [47, 107, 145](#)
- recordingMap<- (recordingMap), [107](#)
- recordings, [108, 145](#)

- registerCurves, 108
- registerCurves(), 56
- registerOperation, 110, 111, 112
- registerReader, 110, 111, 112
- registerWriter, 110, 111, 112
- removeEvents, 8, 59, 113
- renameChannels, 33, 34, 114
- replaceNA, 53, 61, 66, 81, 115
- resampleToCommon, 16, 75, 76, 116, 132, 140
- resolveQuery, 116
- resultType (AnalysisResult-accessors), 18
- resultType, AnalysisResult-method (AnalysisResult-accessors), 18
- resultValue (AnalysisResult-accessors), 18
- resultValue, AnalysisResult-method (AnalysisResult-accessors), 18
- resultValue, PhysioBiomarker-method (AnalysisResult-accessors), 18
- samplesToTime, 117, 139
- samplingRate, 26, 41, 46, 94, 95, 117, 118, 138, 139
- samplingRate, PhysioExperiment-method (samplingRate), 118
- samplingRate<- (samplingRate), 118
- samplingRate<-, PhysioExperiment-method (samplingRate), 118
- samplingRates (multiphysio-legacy-accessors), 73
- samplingRates, MultiPhysioExperiment-method (multiphysio-legacy-accessors), 73
- scale\_color\_physio, 119
- scale\_colour\_physio (scale\_color\_physio), 119
- scale\_fill\_physio (scale\_color\_physio), 119
- sem, 89, 120
- sem(), 68, 71, 87, 148
- session, 97, 121
- sessions, 96, 121, 121
- sessions, PhysioLongitudinal-method (sessions), 121
- sessions<- (sessions), 121
- sessions<-, PhysioLongitudinal-method (sessions), 121
- setAssaySamplingRate, 26, 122
- setChannelTypes, 31, 32, 57, 123, 124
- setChannelUnits, 123, 124
- setElectrodePositions, 22, 58, 125
- setEvents, 8, 59, 92, 94, 113, 126
- setReference, 22, 60, 64, 127
- show, AnalysisResult-method (AnalysisResult-class), 20
- show, MultiPhysioExperiment-method (MultiPhysioExperiment-methods), 77
- show, NormativeReference-method (NormativeReference-class), 84
- show, PhysioBiomarker-method (PhysioBiomarker-class), 88
- show, PhysioCohort-method (PhysioCohort-methods), 91
- show, PhysioEvents-method, 128
- show, PhysioExperiment-method, 128
- show, PhysioLongitudinal-method (PhysioLongitudinal-methods), 97
- streamCoverage, 13–15, 43, 129, 142, 143
- streamNames, 130
- streamRates, 75, 131
- streams, 131
- streams, MultiPhysioExperiment-method (streams), 131
- streams<- (streams), 131
- streams<-, MultiPhysioExperiment-method (streams), 131
- streamTimeIndex, 16, 65, 76, 116, 132, 140
- subject, 133
- subjectData, 133
- subjectData, PhysioLongitudinal-method (subjectData), 133
- subjectData<- (subjectData), 133
- subjectData<-, PhysioLongitudinal-method (subjectData), 133
- subjectIds, 134
- subjects, 90, 91, 133, 134
- subjects, PhysioCohort-method (subjects), 134
- subjects<- (subjects), 134
- subjects<-, PhysioCohort-method (subjects), 134
- subjectTrials, 39, 135
- subsetCohort, 136

summary, *PhysioExperiment*-method, 136

theme\_physio, 137

timeIndex, 23, 41, 46, 52, 117, 118, 138, 139

timeToSamples, 52, 117, 138, 139

timeWindow, 5, 43, 75, 76, 129, 130, 132, 140, 141

trial, 141, 144

trialCoverage, 142

trialIntervals, 14, 43, 141, 142, 143, 144

trialIntervals<- (trialIntervals), 143

trials, 108, 135, 143, 144

trials<- (trials), 144

trialSet, 108, 144

uncertaintyOf (*AnalysisResult*-carrier), 19

uncertaintyOf, *AnalysisResult*-method  
(*AnalysisResult*-carrier), 19

unregisterOperation  
(registerOperation), 110

unregisterReader (registerReader), 111

unregisterWriter (registerWriter), 112

updateObject, *MultiPhysioExperiment*-method  
(migrateContainer), 72

watsonWilliamsTest, 145

watsonWilliamsTest(), 103

waveformCMC, 146

waveformCMC(), 147, 148

waveformICC, 147

waveformICC(), 147, 148

waveformReliability, 148

waveformReliability(), 147

withProvenance, 101, 107, 149

wpliEstimate, 150

zScore, 85

zScore (*NormativeReference*-compare), 85

zScore(), 83

zScore, *NormativeReference*-method  
(*NormativeReference*-compare), 85