Package 'gpbiometrics'

July 4, 2026

Type Package

Title Process and Report Gazepoint Biometrics Data

Version 0.1.0

Description Imports, validates, quality-checks, preprocesses, summarises, synchronises, models, plots, and reports Gazepoint Biometrics exports. The package focuses on Gazepoint-specific biometric channels such as GSR/EDA, heart rate, interbeat intervals, pulse signals, engagement dial, TTL markers, and synchronisation fields that can be combined with Gazepoint GP3 and GP3 HD eye-tracking workflows.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

VignetteBuilder knitr

Suggests ggplot2, knitr, reticulate, rmarkdown, shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Stefanos Balaskas [aut, cre]

Maintainer Stefanos Balaskas <s.balaskas@ac.upatras.gr>

Repository CRAN

Date/Publication 2026-07-04 07:10:15 UTC

Contents

`align_gazepoint_biometrics_to_ttl` *Align Gazepoint biometric samples to TTL events*

Description

Aligns biometric rows to TTL/event markers and returns event-relative time and sample indices. The helper is conservative: TTL events are detected from rising edges by default, validity flags are used when available, and no physiological interpretation is added.

Usage

```
align_gazepoint_biometrics_to_ttl(
  data,
  ttl_cols = NULL,
  event_col = NULL,
  ttl_valid_col = NULL,
  time_col = NULL,
  sample_col = NULL,
  group_cols = NULL,
  participant_col = NULL,
  stimulus_col = NULL,
  trial_col = NULL,
  event_value = NULL,
  valid_values = c(TRUE, 1, "1"),
  event_edge = c("rising", "change", "active"),
  pre_window_ms = 1000,
  post_window_ms = 5000,
  pre_window_samples = NULL,
  post_window_samples = NULL,
  collapse_nearby_ms = 0,
  require_valid_ttl = TRUE
)
```

Arguments

`data`	A data frame containing biometric samples.
`ttl_cols`	Optional TTL marker columns. If `NULL`, the function first looks for `ttl_marker`, then raw `TTL0`-`TTL6` columns.

`event_col` Optional single user-specified event column. If supplied, it is used instead of automatic TTL-column detection.

`ttl_valid_col` Optional TTL validity column. If `NULL`, the function looks for `ttl_validity_flag` or `TTLV`.

`time_col` Optional time column. If `NULL`, common time-column names are detected automatically when present.

`sample_col` Optional sample/counter column. If `NULL`, `CNT`/`cnt` is used when present; otherwise row order is used.

`group_cols` Optional grouping columns. If `NULL`, the function uses available participant/stimulus/trial-like columns when present.

`participant_col`, `stimulus_col`, `trial_col`
Optional explicit grouping columns to add to `group_cols`.

`event_value` Optional value(s) that define an active event. If `NULL`, non-zero numeric/logical values and non-empty character values are treated as active.

`valid_values` Values treated as valid in the TTL validity column.

`event_edge` Event-detection rule. `"rising"` keeps inactive-to-active transitions, `"change"` keeps changes among active event values, and `"active"` keeps every active sample.

`pre_window_ms`, `post_window_ms`
Event window in milliseconds when a usable time column is available.

`pre_window_samples`, `post_window_samples`
Event window in samples when no usable time column is available. If omitted, only the event sample is kept.

`collapse_nearby_ms`
Optional minimum distance between retained events within a group, in milliseconds.

`require_valid_ttl`
If `TRUE`, a detected TTL validity column must be active for a row to count as an event.

Value

A list with `overview`, `events`, `aligned_data`, and `settings`.

`analyze_gazepoint_ac_susceptance`
Analyse AC EDA admittance and susceptance recordings

Description

Computes summaries for specialised alternating-current EDA recordings. This function is for true AC admittance/susceptance data, not ordinary DC skin-conductance columns such as `GSR_US`.

Usage

```
analyze_gazepoint_ac_susceptance(
  dat,
  conductance_col = NULL,
  susceptance_col = NULL,
  admittance_col = NULL,
  phase_col = NULL,
  frequency_col = NULL,
  time_col = NULL,
  group_cols = NULL
)
```

Arguments

dat	A data frame.
conductance_col	Optional real conductance component column.
susceptance_col	Optional imaginary susceptance component column.
admittance_col	Optional admittance magnitude column.
phase_col	Optional phase angle column.
frequency_col	Optional AC frequency column.
time_col	Optional time column.
group_cols	Optional grouping columns.

Value

A list with overview, timeseries, summary, and settings.

---

analyze_gazepoint_cardiorespiratory_causality

Analyse cardiorespiratory Granger-style directionality

Description

Computes dependency-light linear Granger-style directionality tests between a respiration proxy and heart-rate or IBI/RR signal. This estimates predictive directionality in a VAR-style model. It does not prove physiological causality from observational data by itself.

Usage

```
analyze_gazepoint_cardiorespiratory_causality(
  dat,
  respiration_col,
  cardiac_col,
  time_col = NULL,
  group_cols = NULL,
  lag_order = 3,
  min_rows = 30,
  standardise = TRUE
)
```

Arguments

dat	A data frame.
respiration_col	Numeric respiration proxy column.
cardiac_col	Numeric cardiac column, such as HR, IBI, or RR.
time_col	Optional time column for ordering.
group_cols	Optional grouping columns.
lag_order	VAR lag order.
min_rows	Minimum complete rows per group.
standardise	Logical. If TRUE, z-standardise both series per group.

Value

A list with overview, causality_summary, and settings.

---

analyze_gazepoint_skin_potential

Analyse endosomatic skin-potential recordings

Description

Computes skin-potential level and skin-potential response descriptors from a voltage-like skin-potential column. This is for endosomatic skin-potential recordings, not standard exosomatic skin conductance.

Usage

```
analyze_gazepoint_skin_potential(
  dat,
  sp_col,
  time_col,
  group_cols = NULL,
```

```
  response_direction = c("both", "positive", "negative"),
  response_threshold = NULL,
  min_response_distance_s = 1
)
```

Arguments

dat	A data frame.
sp_col	Numeric skin-potential column, usually in millivolts.
time_col	Numeric time column.
group_cols	Optional grouping columns.
response_direction	"both", "positive", or "negative".
response_threshold	Optional absolute threshold for response detection. If NULL, a MAD-based derivative threshold is used.
min_response_distance_s	Minimum distance between detected responses.

Value

A list with overview, level_summary, response_table, timeseries, and settings.

---

assess_gazepoint_hrp_waveform_quality

Assess Gazepoint HRP waveform quality

Description

Computes descriptive quality-control summaries for a Gazepoint HRP/PPG waveform column. The output is intended for waveform availability, missingness, flatness, and timing-gap review. It does not infer diagnosis, emotion, valence, cognition, preference, or true physiological state.

Usage

```
assess_gazepoint_hrp_waveform_quality(
  data,
  hrp_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  time_unit = c("auto", "ms", "seconds", "samples"),
  min_rows = 20,
  min_finite_prop = 0.8,
  max_flat_prop = 0.95,
  flat_tolerance = 1e-08,
  max_gap_multiplier = 3
)
```

Arguments

data	A Gazepoint biometric data frame or a list containing one.
hrp_col	Optional HRP/PPG waveform column. If NULL, common column names are detected.
time_col	Optional time, timestamp, or sample-counter column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz.
time_unit	Unit of time_col: "auto", "ms", "seconds", or "samples".
min_rows	Minimum rows required per group.
min_finite_prop	Minimum finite waveform proportion required per group.
max_flat_prop	Maximum allowed proportion of near-zero consecutive differences among finite waveform values.
flat_tolerance	Absolute difference threshold used to identify near-flat consecutive waveform changes.
max_gap_multiplier	Time gaps larger than this multiple of the median positive time step are flagged.

Value

A list with overview, group_quality, row_flags, and settings.

---

audit_gazepoint_biometric_missingness

Audit missingness in Gazepoint biometric channels

Description

Summarises missingness and zero values for Gazepoint biometric columns. This is useful because Gazepoint exports may contain biometric columns even when a channel was inactive or invalid during recording.

Usage

```
audit_gazepoint_biometric_missingness(data, columns = NULL)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
columns	Optional character vector of columns to audit. If NULL, known present Gazepoint biometric, TTL, and validity columns are audited.

Value

A data frame with one row per audited column.

audit_gazepoint_biometric_sampling *Audit Gazepoint biometric sampling and timing*

Description

Audits timing or row-order information in Gazepoint Biometrics exports. The function checks monotonicity, duplicate timestamps, nonpositive intervals, and estimated sampling rate when the selected time column has a real time unit. If only CNT is available, the function can still check ordering but does not estimate a sampling rate unless time_unit is explicitly meaningful.

Usage

```
audit_gazepoint_biometric_sampling(
  data,
  group_columns = NULL,
  time_column = NULL,
  time_unit = c("seconds", "milliseconds", "microseconds", "samples"),
  expected_rate_hz = 60,
  tolerance_hz = 5
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
group_columns	Optional grouping columns within which timing should be audited, such as c("source_participant", "MEDIA_ID").
time_column	Optional time/order column. If NULL, the function uses the first available column among TIME, TIME_TICK, and CNT.
time_unit	Unit of the selected time column. Use "seconds", "milliseconds", "microseconds", or "samples". When "samples" is used, sampling-rate estimates are returned as NA.
expected_rate_hz	Optional expected sampling rate in Hz.
tolerance_hz	Acceptable absolute deviation from expected_rate_hz.

Value

A data frame with one row per group.

`audit_gazepoint_biometric_sync_drift` *Audit Gazepoint biometric synchronization drift*

Description

Combines time-order/reset diagnostics with conservative signal-lag summaries across signal pairs and groups. The helper is intended for quality control and synchronization review. It does not infer emotional valence, cognitive states, causal timing, or true physiological latency.

Usage

```
audit_gazepoint_biometric_sync_drift(
  data,
  time_col = NULL,
  group_cols = NULL,
  signal_pairs = NULL,
  signal_cols = NULL,
  reference_signal_col = NULL,
  max_lag = 1000,
  lag_step = NULL,
  drift_tolerance = NULL,
  method = c("pearson", "spearman"),
  min_complete_pairs = 20,
  use_first_difference = FALSE,
  include_reset_segments = TRUE
)
```

Arguments

`data`	A Gazepoint biometric data frame.
`time_col`	Optional time or counter column.
`group_cols`	Optional grouping columns.
`signal_pairs`	Optional two-column data frame, matrix, or list defining signal pairs. If `NULL`, pairs are formed between a reference signal and other detected biometric signals.
`signal_cols`	Optional candidate signal columns used when `signal_pairs` is `NULL`.
`reference_signal_col`	Optional reference signal used when `signal_pairs` is `NULL`.
`max_lag`	Maximum absolute lag to evaluate, in the same units as `time_col`.
`lag_step`	Step size between candidate lags. If `NULL`, the median positive time step is used.
`drift_tolerance`	Optional threshold for the range of estimated lags across groups. If `NULL`, drift is summarized but not threshold-classified.
`method`	Correlation method passed to `stats::cor()`.

`min_complete_pairs`
: Minimum complete aligned observations required for each candidate lag.

`use_first_difference`
: If `TRUE`, lag diagnostics use first differences.

`include_reset_segments`
: If `TRUE`, reset segments from `audit_gazepoint_time_resets()` are added to grouping when available.

Value

A list with `overview`, `checks`, `time_reset_audit`, `lag_by_group`, `lag_profile`, `drift_summary`, and `settings`.

`audit_gazepoint_distributional_drift` *Audit distributional drift across sessions or ordered blocks*

Description

Compares signal distributions across sessions/blocks using baseline-vs-current differences, Kolmogorov-Smirnov tests, and Population Stability Index (PSI).

Usage

```
audit_gazepoint_distributional_drift(
  dat,
  signal_cols,
  session_col = "session",
  participant_col = NULL,
  reference_session = NULL,
  bins = 10,
  psi_warn = 0.1,
  psi_fail = 0.25
)
```

Arguments

`dat` A data frame containing longitudinal biometric data.

`signal_cols` Numeric signal columns to audit.

`session_col` Session/block/timepoint column.

`participant_col`
: Optional participant column.

`reference_session`
: Optional reference session. If `NULL`, the first ordered session is used within each participant/global group.

`bins` Number of bins for PSI.

`psi_warn` PSI threshold for warning.

`psi_fail` PSI threshold for failure.

Value

A list with `overview`, `drift_summary`, and `settings`.

`audit_gazepoint_eda_artifacts` *Audit Gazepoint EDA/GSR artifacts*

Description

Flags row-level artifacts in Gazepoint electrodermal activity signals, preferring `GSR_US` conductance when available. The helper detects abrupt jumps, abrupt slopes, flatline runs, zero runs, negative conductance values, and optional out-of-range values. It is a conservative preprocessing/QC helper and does not interpret EDA as emotional valence.

Usage

```
audit_gazepoint_eda_artifacts(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  prefer_gsr_us = TRUE,
  jump_threshold_sd = 6,
  slope_threshold_sd = 6,
  flat_run_length = 20,
  zero_run_length = 20,
  saturation_min = NULL,
  saturation_max = NULL,
  negative_allowed = NULL
)
```

Arguments

`data`	A data frame containing Gazepoint biometric rows.
`signal_col`	Optional EDA/GSR signal column. If `NULL`, the function prefers `GSR_US` and then falls back to common Gazepoint EDA columns.
`time_col`	Optional time/counter column. If `NULL`, common Gazepoint time columns are detected automatically.
`group_cols`	Optional grouping columns. If `NULL`, available source/participant/media/trial-like columns are used.
`prefer_gsr_us`	Logical. If `TRUE`, prefer `GSR_US` when `signal_col` is not supplied.
`jump_threshold_sd`	Robust z threshold for absolute signal jumps.
`slope_threshold_sd`	Robust z threshold for absolute signal slopes.

`flat_run_length`
: Minimum repeated-value run length flagged as flatline.

`zero_run_length`
: Minimum zero-value run length flagged as zero run.

`saturation_min`
: Optional lower bound for acceptable signal values.

`saturation_max`
: Optional upper bound for acceptable signal values.

`negative_allowed`
: Optional logical. If `NULL`, negative values are allowed for phasic component columns but not for conductance-like columns.

Value

A list with `overview`, `row_flags`, `artifact_runs`, `group_summary`, and `settings`.

---

audit_gazepoint_engagement_dial

Audit Gazepoint engagement-dial signal quality

Description

Audits Gazepoint engagement-dial values for missingness, inactive rows, validity flags, plausible range, flatlining, and usable sample coverage.

Usage

```
audit_gazepoint_engagement_dial(
  data,
  value_column = "DIAL",
  validity_column = "DIALV",
  min_value = 0,
  max_value = 1,
  jump_threshold = NULL
)
```

Arguments

`data`
: A data frame or a path to a Gazepoint CSV export.

`value_column`
: Engagement-dial value column. Defaults to `"DIAL"`.

`validity_column`
: Engagement-dial validity column. Defaults to `"DIALV"`.

`min_value`
: Minimum plausible dial value.

`max_value`
: Maximum plausible dial value.

`jump_threshold`
: Optional threshold for detecting large sample-to-sample jumps.

Value

A one-row data frame summarising signal quality.

audit_gazepoint_gsr_quality *Audit Gazepoint GSR/EDA signal quality*

Description

Audits Gazepoint GSR/EDA columns for missingness, inactive zero rows, validity flags, plausible value ranges, flatlining, and usable sample coverage. When available, GSR_US is used by default because it represents skin conductance in microsiemens in Gazepoint exports.

Usage

```
audit_gazepoint_gsr_quality(
  data,
  value_column = NULL,
  validity_column = "GSRV",
  min_value = 0,
  max_value = 100,
  jump_threshold = NULL
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
value_column	Optional GSR/EDA value column. If NULL, GSR_US is used when present, otherwise GSR.
validity_column	Optional validity column. Defaults to "GSRV".
min_value	Minimum plausible value.
max_value	Maximum plausible value.
jump_threshold	Optional threshold for detecting large sample-to-sample jumps.

Value

A one-row data frame summarising signal quality.

audit_gazepoint_gsr_units *Audit likely GSR/EDA units*

Description

Checks whether a Gazepoint GSR/EDA column looks more like conductance in microSiemens or resistance/impedance-like values in Ohms. This is a preprocessing safety audit, not a definitive device calibration test.

Usage

```
audit_gazepoint_gsr_units(
  dat,
  gsr_col = "GSR",
  convert = FALSE,
  output_col = NULL,
  resistance_to_us_factor = 1e+06
)
```

Arguments

dat	A data frame.
gsr_col	Name of the GSR/EDA column to audit.
convert	Logical. If TRUE, add a conductance-converted column when the signal is likely resistance/impedance-like.
output_col	Output column used when convert = TRUE.
resistance_to_us_factor	Conversion factor. For Ohms to microSiemens, use 1000000 / resistance.

Value

A list with overview, diagnostics, recommendation, and, when requested, data.

---

audit_gazepoint_hr_quality

Audit Gazepoint heart-rate signal quality

Description

Audits Gazepoint heart-rate values for missingness, inactive zero rows, validity flags, plausible value ranges, sudden jumps, flatlining, and usable sample coverage. HRV is treated as the heart-rate validity flag, not as a heart-rate-variability metric.

Usage

```
audit_gazepoint_hr_quality(
  data,
  value_column = "HR",
  validity_column = "HRV",
  min_value = 30,
  max_value = 220,
  jump_threshold = 25
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
value_column	Heart-rate value column. Defaults to "HR".
validity_column	Heart-rate validity column. Defaults to "HRV".
min_value	Minimum plausible heart rate.
max_value	Maximum plausible heart rate.
jump_threshold	Threshold for detecting large sample-to-sample jumps.

Value

A one-row data frame summarising signal quality.

---

audit_gazepoint_ibi_quality

Audit IBI/RR interval quality

Description

Audits inter-beat interval (IBI) or RR-interval values in Gazepoint Biometrics-style exports. This helper deliberately does not use raw HRV columns as HRV metrics. HRV-style summaries should be derived only from genuine IBI/RR interval columns.

Usage

```
audit_gazepoint_ibi_quality(
  data,
  ibi_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  unit = c("auto", "milliseconds", "seconds"),
  min_ibi_ms = 300,
  max_ibi_ms = 2000,
  max_jump_ms = 500
)
```

Arguments

data	A data frame.
ibi_col	Optional IBI/RR interval column. If NULL, the function detects recognised IBI/RR-style column names.
group_cols	Optional grouping columns, such as participant, trial, stimulus, condition, or window labels.
time_col	Optional time/order column used to order samples before successive-difference checks.

unit	Unit of the IBI values. "auto" treats median values below 10 as seconds and larger values as milliseconds.
min_ibi_ms	Minimum plausible IBI in milliseconds.
max_ibi_ms	Maximum plausible IBI in milliseconds.
max_jump_ms	Maximum plausible absolute change between successive IBI values within a group.

Value

A list with overview, samples, group_summary, and settings.

Examples

```
df <- data.frame(
  USER = rep(c("P1", "P2"), each = 4),
  IBI = c(800, 810, 790, 805, 900, 910, 905, 920)
)
audit_gazepoint_ibi_quality(df, group_cols = "USER")
```

---

audit_gazepoint_signal_activity

Audit Gazepoint biometric signal activity

Description

Screens biometric signal columns for missingness, all-zero channels, constant values, low variation, and active signal presence within groups. This helper is designed to identify inactive files or channels before event-level EDA, HR, IBI, or multimodal analysis.

Usage

```
audit_gazepoint_signal_activity(
  data,
  signal_cols = NULL,
  group_cols = NULL,
  zero_is_inactive = TRUE,
  min_unique_nonzero = 2,
  missing_as_inactive = TRUE
)
```

Arguments

data	A data frame containing Gazepoint biometric rows.
signal_cols	Optional signal columns. If NULL, common Gazepoint biometric columns are detected automatically.

group_cols — Optional grouping columns. If NULL, available source, participant, media, or trial columns are used.

zero_is_inactive — Logical. If TRUE, all-zero signals are labelled as inactive.

min_unique_nonzero — Minimum number of distinct non-zero finite values required for an "active" status.

missing_as_inactive — Logical. If TRUE, all-missing signals are labelled as insufficient/inactive.

Value

A list with overview, signal_by_group, inactive_groups, inactive_signals, and settings.

---

audit_gazepoint_stabilization_period

Audit or trim the EDA electrode stabilization period

Description

Flags or removes the initial stabilization period in each recording/group. This is intended to prevent early skin-electrode drift from being treated as a stable physiological baseline.

Usage

```
audit_gazepoint_stabilization_period(
  dat,
  time_col = "CNT",
  group_cols = NULL,
  stabilization_minutes = 10,
  action = c("flag", "trim"),
  output_col = "in_stabilization_period",
  time_units = c("auto", "seconds", "milliseconds")
)
```

Arguments

dat — A data frame.

time_col — Numeric time column.

group_cols — Optional grouping columns.

stabilization_minutes — Stabilization duration to flag or trim.

action — "flag" or "trim".

output_col — Output logical flag column.

time_units — "auto", "seconds", or "milliseconds".

Value

A data frame with stabilization-period attributes.

`audit_gazepoint_time_resets` *Audit Gazepoint biometric time resets*

Description

Detects negative time steps, duplicate time steps, non-finite time values, and recording segments within grouped Gazepoint biometric exports. This helper is intended for quality control and synchronization inspection. It does not alter raw values unless `return_reindexed_time = TRUE`, in which case an additional segment-relative time column is added.

Usage

```
audit_gazepoint_time_resets(
  data,
  time_col = NULL,
  group_cols = NULL,
  allow_ties = TRUE,
  split_on_negative_step = TRUE,
  return_reindexed_time = FALSE,
  min_segment_rows = 1
)
```

Arguments

`data`	A data frame containing Gazepoint biometric rows.
`time_col`	Optional time/counter column. If `NULL`, common Gazepoint time columns are detected automatically.
`group_cols`	Optional grouping columns. If `NULL`, available source/participant/media/trial-like columns are used.
`allow_ties`	Logical. If `TRUE`, repeated time values are not treated as non-monotonic.
`split_on_negative_step`	Logical. If `TRUE`, negative time steps start a new segment within each group.
`return_reindexed_time`	Logical. If `TRUE`, adds `time_reindexed_within_segment`, starting at zero within each detected segment.
`min_segment_rows`	Minimum rows expected per segment before a segment is flagged as short.

Value

A list with `overview`, `segment_summary`, `row_flags`, `data_with_segments`, and `settings`.

`baseline_correct_gazepoint_gsr` *Baseline-correct Gazepoint GSR/EDA*

Description

Adds a baseline-corrected GSR/EDA column to a Gazepoint Biometrics table. When available, `GSR_US` is used by default because it represents skin conductance in microsiemens in Gazepoint exports. The baseline is estimated from rows selected by `baseline_rows`, optionally within groups.

Usage

```
baseline_correct_gazepoint_gsr(
  data,
  baseline_rows,
  value_column = NULL,
  validity_column = "GSRV",
  group_columns = NULL,
  output_column = NULL,
  summary = c("mean", "median"),
  exclude_zero = TRUE
)
```

Arguments

`data`	A data frame or a path to a Gazepoint CSV export.
`baseline_rows`	Logical vector identifying baseline rows.
`value_column`	Optional GSR/EDA value column. If `NULL`, `GSR_US` is used when present, otherwise `GSR`.
`validity_column`	Optional validity column. Defaults to `"GSRV"`.
`group_columns`	Optional grouping columns. When supplied, baselines are estimated separately within each group.
`output_column`	Name of the corrected output column.
`summary`	Baseline summary, either `"mean"` or `"median"`.
`exclude_zero`	Should zero values be excluded from baseline estimation?

Value

A data frame with the added baseline-corrected column and a baseline-summary attribute named `"baseline_summary"`.

`baseline_correct_gazepoint_hr` *Baseline-correct Gazepoint heart rate*

Description

Adds a baseline-corrected heart-rate column to a Gazepoint Biometrics table. `HRV` is treated as the heart-rate validity flag, not as a heart-rate variability metric.

Usage

```
baseline_correct_gazepoint_hr(
  data,
  baseline_rows,
  value_column = "HR",
  validity_column = "HRV",
  group_columns = NULL,
  output_column = NULL,
  summary = c("mean", "median"),
  exclude_zero = TRUE
)
```

Arguments

`data`	A data frame or a path to a Gazepoint CSV export.
`baseline_rows`	Logical vector identifying baseline rows.
`value_column`	Heart-rate value column. Defaults to `"HR"`.
`validity_column`	Heart-rate validity column. Defaults to `"HRV"`.
`group_columns`	Optional grouping columns. When supplied, baselines are estimated separately within each group.
`output_column`	Name of the corrected output column.
`summary`	Baseline summary, either `"mean"` or `"median"`.
`exclude_zero`	Should zero values be excluded from baseline estimation?

Value

A data frame with the added baseline-corrected column and a baseline-summary attribute named `"baseline_summary"`.

`baseline_correct_gazepoint_pupil` *Baseline-correct Gazepoint pupil size*

Description

Applies subtractive or divisive baseline correction to pupil-size data within trials or other grouping units.

Usage

```
baseline_correct_gazepoint_pupil(
  dat,
  pupil_col = NULL,
  time_col = "CNT",
  stimulus_onset_col = NULL,
  trial_cols = NULL,
  baseline_window = c(-240, -200),
  baseline_function = c("median", "mean"),
  correction = c("subtract", "divide"),
  suffix = "_baseline_corrected",
  min_baseline_rows = 2,
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame containing pupil-size data.
`pupil_col`	Pupil column. If `NULL`, common Gazepoint pupil columns are detected.
`time_col`	Time column.
`stimulus_onset_col`	Optional stimulus-onset column. If supplied, baseline windows are interpreted relative to onset.
`trial_cols`	Trial/grouping columns.
`baseline_window`	Numeric vector of length two defining the baseline window relative to stimulus onset.
`baseline_function`	`"median"` or `"mean"`.
`correction`	`"subtract"` or `"divide"`.
`suffix`	Suffix for corrected output column.
`min_baseline_rows`	Minimum finite baseline rows required.
`overwrite`	Logical. If `FALSE`, existing output columns are protected.

Value

A data frame with a corrected pupil column and baseline attributes.

`calculate_gazepoint_rsa` *Calculate respiration-informed RSA proxy features*

Description

Calculates dependency-light RSA proxy summaries from IBI/RR intervals and, optionally, PPG-derived respiration features from `extract_gazepoint_pdr_signals()`. This provides peak-to-trough and Porges-Bohrer-inspired band-power proxy summaries. These outputs should be interpreted as respiration-informed HRV/RSA features, not direct clinical vagal-tone estimates.

Usage

```
calculate_gazepoint_rsa(
  dat,
  ibi_col = "IBI",
  time_col = "CNT",
  group_cols = NULL,
  pdr = NULL,
  resp_rate_hz = NULL,
  respiration_band = c(0.12, 0.4),
  resample_rate = 4
)
```

Arguments

`dat`	A data frame containing IBI/RR data.
`ibi_col`	IBI/RR interval column.
`time_col`	Time column for the IBI/RR observations.
`group_cols`	Optional grouping columns.
`pdr`	Optional output from `extract_gazepoint_pdr_signals()`.
`resp_rate_hz`	Optional fixed respiration rate in Hz.
`respiration_band`	Default respiration/HF band when no PDR rate is available.
`resample_rate`	Resampling rate for spectral RSA proxy calculation.

Value

A list with `overview`, `rsa_summary`, and `settings`.

check_gazepoint_biometric_columns

Check Gazepoint biometric columns

Description

Checks whether a data frame contains known Gazepoint Biometrics columns. This function does not assume that the channels are active. It only checks whether the expected columns are present.

Usage

```
check_gazepoint_biometric_columns(data)
```

Arguments

data	A data frame imported from a Gazepoint export.

Value

A data frame describing expected columns, their signal family, interpretation, and whether they are present.

check_gazepoint_plot_contract

Check a Gazepoint plot return contract

Description

Checks whether an object follows the package's plotting return convention.

Usage

```
check_gazepoint_plot_contract(
  plot,
  require_plot_data = TRUE,
  require_settings = TRUE
)
```

Arguments

plot	A plot object.
require_plot_data	Logical. If TRUE, plot_data must be present.
require_settings	Logical. If TRUE, settings must be present.

Value

A list with `overview`, `checks`, `plot_data`, and `settings`.

`chunk_gazepoint_biometrics` *Chunk Gazepoint biometric data into fixed analysis episodes*

Description

Adds programmatic fixed-duration chunks/episodes to continuous biometric data. This is useful for baseline segmentation, repeated-measures feature extraction, and analyses that do not rely on external TTL markers.

Usage

```
chunk_gazepoint_biometrics(
  dat,
  time_col = "CNT",
  group_cols = NULL,
  chunk_seconds = 60,
  start_time = NULL,
  chunk_col = "chunk_id",
  episode_col = "episode_id",
  include_partial = FALSE
)
```

Arguments

`dat`	A data frame.
`time_col`	Numeric time column.
`group_cols`	Optional grouping columns.
`chunk_seconds`	Chunk duration in seconds.
`start_time`	Optional fixed start time. If `NULL`, uses group minimum.
`chunk_col`	Output chunk identifier column.
`episode_col`	Output episode label column.
`include_partial`	Logical. If `FALSE`, partial final chunks are marked but not assigned as complete chunks.

Value

A data frame with chunk columns and chunk-summary attributes.

`classify_gazepoint_eda_response_pattern` *Classify descriptive Gazepoint EDA response patterns*

Description

Classifies descriptive EDA response patterns from an EDA/SCR feature column. The labels are intended for quality-control and descriptive reporting only. They do not infer emotion, valence, stress, trust, preference, cognition, or diagnosis.

Usage

```
classify_gazepoint_eda_response_pattern(
  data,
  response_col = NULL,
  group_cols = NULL,
  summary_function = c("max_abs", "mean_abs", "median_abs"),
  no_response_threshold = 0.01,
  low_response_threshold = 0.05,
  moderate_response_threshold = 0.2
)
```

Arguments

`data`	A data frame containing EDA/SCR values.
`response_col`	Optional response feature column. If omitted, the helper searches for common SCR/EDA response columns and then `GSR_US_PHASIC` or `GSR_US`.
`group_cols`	Optional grouping columns.
`summary_function`	Summary used within each group.
`no_response_threshold`	Absolute response threshold for `no_detectable_response`.
`low_response_threshold`	Upper threshold for `low_response`.
`moderate_response_threshold`	Upper threshold for `moderate_response`.

Value

A list with `overview`, `classifications`, and `settings`.

`classify_gazepoint_scr_intervals` *Classify SCRs into FIR, SIR, and TIR latency intervals*

Description

Classifies extracted SCRs into first-, second-, and third-interval response windows based on response latency after stimulus onset.

Usage

```
classify_gazepoint_scr_intervals(
  dat,
  response_time_col = NULL,
  stimulus_onset_col = NULL,
  latency_col = NULL,
  output_col = "scr_interval",
  latency_output_col = "scr_latency_s",
  fir = c(1, 4),
  sir = c(4, 7),
  tir = c(7, 10)
)
```

Arguments

`dat`	A data frame containing SCR events or peaks.
`response_time_col`	Optional response/peak time column.
`stimulus_onset_col`	Optional stimulus-onset column. Required when `latency_col` is not supplied.
`latency_col`	Optional precomputed latency column.
`output_col`	Name of the output interval column.
`latency_output_col`	Name of the latency output column.
`fir`	Numeric vector of length two defining FIR window in seconds.
`sir`	Numeric vector of length two defining SIR window in seconds.
`tir`	Numeric vector of length two defining TIR window in seconds.

Value

A data frame with interval labels and latency metadata.

`compare_gazepoint_hr_ibi_consistency` *Compare Gazepoint HR and IBI-derived heart rate*

Description

Compares recorded HR against HR derived from genuine IBI/RR intervals using `60000 / IBI_ms`.

Usage

```
compare_gazepoint_hr_ibi_consistency(
  data,
  hr_col = "HR",
  ibi_col = "IBI",
  time_col = NULL,
  group_cols = NULL,
  unit = c("auto", "ms", "seconds"),
  max_abs_diff_bpm = 10,
  max_rel_diff_prop = 0.15
)
```

Arguments

`data`	A Gazepoint biometric data frame or `gazepoint_ibi_filter` object.
`hr_col`	Heart-rate column in beats per minute.
`ibi_col`	IBI/RR interval column.
`time_col`	Optional time/counter column.
`group_cols`	Optional grouping columns.
`unit`	Unit of the IBI column: `"auto"`, `"ms"`, or `"seconds"`.
`max_abs_diff_bpm`	Maximum acceptable absolute HR difference in bpm.
`max_rel_diff_prop`	Maximum acceptable relative HR difference.

Value

A list with `overview`, `row_diagnostics`, `group_summary`, and `settings`.

convert_gazepoint_gsr_to_conductance *Convert Gazepoint GSR resistance to conductance*

Description

Converts GSR resistance values to conductance in microSiemens. The helper is intentionally conservative. If a conductance column such as GSR_US is already present, the data are returned unchanged unless overwrite = TRUE. If input_unit = "auto", conversion is performed only when the source column has a resistance-like name, such as GSR_OHMS or resistance_ohms.

Usage

```
convert_gazepoint_gsr_to_conductance(
  data,
  gsr_col = NULL,
  output_col = "GSR_US",
  input_unit = c("auto", "ohms", "kohms", "microsiemens"),
  overwrite = FALSE
)
```

Arguments

data	A data frame.
gsr_col	Optional source GSR column. If NULL, a resistance-like column is detected when possible.
output_col	Name of the output conductance column.
input_unit	Source unit. "auto" converts only resistance-like columns; "ohms" converts ohms to microSiemens; "kohms" converts kilo-ohms to microSiemens; "microsiemens" copies values to output_col.
overwrite	Logical. If FALSE, an existing output_col is not overwritten.

Details

Generic GSR columns are not automatically assumed to be resistance because Gazepoint exports and workflows may represent GSR/EDA differently. For a generic GSR column, use input_unit = "ohms" or input_unit = "kohms" only when the study documentation confirms resistance units.

Value

The input data frame with a conductance column when conversion is possible. A structured conversion summary is stored in the gsr_conversion_summary attribute.

Examples

```
df <- data.frame(GSR_OHMS = c(1000000, 500000, NA))
convert_gazepoint_gsr_to_conductance(df)
```

`correct_gazepoint_eda_temperature` *Correct EDA for ambient or body temperature*

Description

Regresses an EDA/conductance signal on one or more continuous temperature covariates and returns a temperature-adjusted residual series. The adjusted signal is temperature-adjusted EDA, not "pure" cognitive or emotional EDA.

Usage

```
correct_gazepoint_eda_temperature(
  dat,
  eda_col = "GSR_US",
  temperature_cols,
  group_cols = NULL,
  time_col = NULL,
  output_col = "eda_temperature_adjusted",
  fitted_col = "eda_temperature_fitted",
  model_by_group = TRUE,
  add_intercept_mean = TRUE
)
```

Arguments

`dat`	A data frame.
`eda_col`	Numeric EDA/conductance column.
`temperature_cols`	One or more numeric temperature columns.
`group_cols`	Optional grouping columns.
`time_col`	Optional time column retained in summaries.
`output_col`	Output residual-adjusted EDA column.
`fitted_col`	Output fitted temperature component column.
`model_by_group`	Logical. If `TRUE`, fit one model per group.
`add_intercept_mean`	Logical. If `TRUE`, add the group mean EDA back to residuals so the adjusted signal remains on the original scale.

Value

A data frame with adjusted EDA columns and model-summary attributes.

create_gazepoint_biometrics_checklist

Create a Gazepoint Biometrics reporting checklist

Description

Creates a compact reporting checklist for Gazepoint Biometrics exports. The checklist summarises detected biometric channels, validation issues, signal quality, missingness, available workflow domains, and interpretation cautions. It is intended to support transparent manuscript reporting and reviewer-facing methods documentation.

Usage

```
create_gazepoint_biometrics_checklist(data, require_active_signal = TRUE)
```

Arguments

`data`	A data frame or a path to a Gazepoint CSV export.
`require_active_signal`	Logical. Should inactive biometric channels be flagged in the validation output?

Value

A list with `overview`, `channels`, `quality`, `missingness`, `validation_issues`, `workflow_capabilities`, `feature_inventory`, `reporting_guidance`, and `interpretation_cautions`.

create_gazepoint_biometrics_feature_inventory

Create a gpbiometrics feature inventory

Description

Creates a structured inventory of implemented gpbiometrics helper functions. This is useful for reporting, readiness checks, documentation, and package development audits.

Usage

```
create_gazepoint_biometrics_feature_inventory(include_internal = FALSE)
```

Arguments

`include_internal`	Logical. If `TRUE`, also checks for non-exported internal helper names when they are included in the inventory.

Value

A list with `overview`, `inventory`, `domain_summary`, `missing_expected`, and `settings`.

create_gazepoint_biometrics_methods_text

Create Gazepoint Biometrics methods text

Description

Creates a compact draft methods paragraph describing Gazepoint Biometrics data processing. The text is intentionally cautious and avoids making emotional or cognitive claims from physiological or eye-tracking measures alone.

Usage

```
create_gazepoint_biometrics_methods_text(
  checklist = NULL,
  data = NULL,
  include_cautions = TRUE
)
```

Arguments

checklist	A checklist produced by `create_gazepoint_biometrics_checklist()`. If `NULL`, `data` must be supplied.
data	Optional data frame or path to a Gazepoint CSV export used to create the checklist when `checklist = NULL`.
include_cautions	Logical. Should interpretation cautions be appended?

Value

A character string containing draft methods text.

create_gazepoint_biometrics_report

Create a Gazepoint Biometrics report

Description

Creates a structured, manuscript-oriented report object for Gazepoint Biometrics data, workflow outputs, quality checks, report tables, methods text, and reporting checklists. The report is intentionally conservative: GSR/EDA is described as electrodermal activity/arousal-related signal rather than emotional valence; heart-rate interpretation is tied to baseline/task context; eye-tracking is described as visual attention rather than direct cognition; and raw `HRV` columns are not treated as HRV metrics.

Usage

```
create_gazepoint_biometrics_report(
  data = NULL,
  workflow = NULL,
  validation = NULL,
  quality = NULL,
  sampling = NULL,
  missingness = NULL,
  exclusions = NULL,
  report_tables = NULL,
  methods_text = NULL,
  checklist = NULL,
  title = "Gazepoint Biometrics report",
  subtitle = NULL,
  output_file = NULL,
  format = c("markdown", "html"),
  include_timestamp = FALSE,
  overwrite = FALSE,
  max_table_rows = 20L
)
```

Arguments

data	Optional biometric data frame.
workflow	Optional workflow object or workflow summary list.
validation	Optional validation object or data frame.
quality	Optional quality-audit object or data frame.
sampling	Optional sampling-audit object or data frame.
missingness	Optional missingness-audit object or data frame.
exclusions	Optional exclusion-recommendation object or data frame.
report_tables	Optional report-table object, data frame, or named list of data frames.
methods_text	Optional methods text, character vector, or object returned by create_gazepoint_biometrics_methods
checklist	Optional checklist object or data frame.
title	Report title.
subtitle	Optional report subtitle.
output_file	Optional path to write a report file.
format	Output format when output_file is supplied. Supported values are "markdown" and "html".
include_timestamp	Logical. Should a creation timestamp be included?
overwrite	Logical. Should an existing output_file be overwritten?
max_table_rows	Maximum number of rows shown per table in the written report.

Details

The function can also write a lightweight Markdown or HTML file without adding heavy reporting dependencies.

Value

A list of class "gazepoint_biometrics_report" with overview, sections, tables, objects, output_file, and settings.

Examples

```
df <- data.frame(
  CNT = 1:5,
  GSR = c(1, 1.1, 1.2, 1.1, 1),
  HR = c(70, 71, 72, 71, 70),
  DIAL = c(40, 42, 44, 43, 41)
)
report <- create_gazepoint_biometrics_report(df)
names(report)
```

---

create_gazepoint_biometrics_report_tables

Create Gazepoint Biometrics report tables

Description

Creates compact report-ready tables from a Gazepoint Biometrics workflow object or from separate workflow components. The function does not write files. It returns cleaned tables that can be printed, exported, or inserted into reports and supplementary materials.

Usage

```
create_gazepoint_biometrics_report_tables(
  workflow = NULL,
  validation = NULL,
  quality = NULL,
  sampling = NULL,
  diagnostics = NULL,
  exclusion_recommendations = NULL,
  ttl_events = NULL,
  max_ttl_events = 20
)
```

Arguments

workflow	Optional workflow object produced by run_gazepoint_biometrics_workflow().
validation	Optional validation object produced by validate_gazepoint_biometrics().
quality	Optional quality-audit table.
sampling	Optional sampling/timing audit table produced by audit_gazepoint_biometric_sampling().
diagnostics	Optional diagnostic table produced by diagnose_gazepoint_biometrics_workflow().
exclusion_recommendations	Optional object produced by recommend_gazepoint_biometric_exclusions().
ttl_events	Optional TTL event table produced by extract_gazepoint_ttl_events().
max_ttl_events	Maximum number of TTL events to include in the compact TTL event table.

Value

A list of report-ready tables.

---

create_gazepoint_eda_analysis_pipeline

Create a Gazepoint EDA analysis pipeline guide

Description

Creates a structured six-phase analysis pipeline for Gazepoint Biometrics EDA/GSR workflows. The helper maps each phase to native gpbiometrics functions, optional external-method bridges, and optional downstream model templates for brms and lme4.

Usage

```
create_gazepoint_eda_analysis_pipeline(
  include_external_bridges = TRUE,
  include_model_templates = TRUE,
  include_reporting_guidance = TRUE,
  style = c("compact", "detailed")
)
```

Arguments

include_external_bridges	Logical. If TRUE, include NeuroKit2, Ledalab-style, PsPM-style, cvxEDA-style, RHRV, and pyPPG bridge helpers.
include_model_templates	Logical. If TRUE, include text templates for downstream brms hurdle models and lme4 mixed-effects models.
include_reporting_guidance	Logical. If TRUE, include reporting and interpretation guardrails.
style	Output style. "compact" returns concise phase descriptions; "detailed" returns fuller phase notes.

Details

This function does not fit statistical models, does not run external software, and does not infer emotion, valence, stress, trust, preference, cognition, or diagnosis. It is a reproducible planning and reporting aid.

Value

A list with `overview`, `phases`, `function_map`, `model_templates`, `reporting_guidance`, `interpretation_guardrails`, and `settings`.

`create_gazepoint_preregistration_template` *Create a Gazepoint biometrics preregistration template*

Description

Creates a cautious preregistration template for Gazepoint Biometrics EDA/GSR workflows.

Usage

```
create_gazepoint_preregistration_template(
  study_title = "Gazepoint biometrics study",
  signal_standardization = c("within_participant_z", "range_correction", "none"),
  artifact_rules = c("kleckner_style", "custom", "none"),
  eda_min_us = 0.01,
  eda_max_us = 100,
  rapid_change_threshold = 20,
  output_file = NULL
)
```

Arguments

`study_title`	Study title.
`signal_standardization`	Standardization plan.
`artifact_rules`	Artifact-rule description.
`eda_min_us`	Minimum conductance threshold.
`eda_max_us`	Maximum conductance threshold.
`rapid_change_threshold`	Maximum absolute percent change per second.
`output_file`	Optional path to write the template as a text file.

Value

A character string.

`decompose_gazepoint_eda` *Decompose Gazepoint GSR/EDA into tonic and phasic components*

Description

Creates descriptive tonic and phasic EDA columns from Gazepoint GSR/EDA data. If vendor-provided tonic/phasic columns such as `GSR_US_TONIC` and `GSR_US_PHASIC` are available, they are used by default. Otherwise, a simple rolling-median tonic estimate is used and the phasic component is calculated as signal minus tonic. This helper is intentionally conservative and does not replace specialised biosignal-processing software.

Usage

```
decompose_gazepoint_eda(
  data,
  signal_col = NULL,
  tonic_col = NULL,
  phasic_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  window_size = 31L,
  output_prefix = "eda",
  overwrite = FALSE
)
```

Arguments

`data`	A data frame.
`signal_col`	Optional GSR/EDA signal column. If `NULL`, a likely conductance-like column is detected.
`tonic_col`	Optional existing tonic column.
`phasic_col`	Optional existing phasic column.
`time_col`	Optional time/order column.
`group_cols`	Optional grouping columns.
`window_size`	Rolling-median window size used when existing tonic/phasic columns are not available. Even values are increased by one.
`output_prefix`	Prefix for output columns.
`overwrite`	Logical. Should existing output columns be overwritten?

Value

A data frame with added tonic, phasic, and method columns. Attributes include `overview` and `settings`.

Examples

```
df <- data.frame(CNT = 1:10, GSR_US = seq(1, 2, length.out = 10))
out <- decompose_gazepoint_eda(df, signal_col = "GSR_US", window_size = 3)
names(out)
```

`denoise_gazepoint_eda_autoencoder` *Denoise EDA using a user-supplied autoencoder reconstruction model*

Description

Applies a user-supplied reconstruction function or model to fixed-length EDA windows. No pre-trained neural network is bundled. This function is an interoperability bridge for validated user-supplied autoencoders.

Usage

```
denoise_gazepoint_eda_autoencoder(
  dat,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  model = NULL,
  window_samples = 128,
  output_col = NULL,
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame.
`eda_col`	Numeric EDA column.
`time_col`	Optional time column.
`group_cols`	Optional grouping columns.
`model`	A user-supplied function or model. If `NULL`, the original signal is copied and status records that no model was supplied.
`window_samples`	Window length in samples.
`output_col`	Optional output column.
`overwrite`	Logical. If `FALSE`, existing output column is protected.

Value

A data frame with reconstructed signal and denoising attributes.

`denoise_gazepoint_eda_wavelet` *Denoise EDA using dependency-light Haar wavelet shrinkage*

Description

Applies simple Haar wavelet soft-threshold denoising to EDA signals within optional groups. This is a dependency-light wavelet denoising helper and should not be described as an exact reproduction of stationary-wavelet artifact-removal algorithms.

Usage

```
denoise_gazepoint_eda_wavelet(
  dat,
  eda_col = "GSR_US",
  group_cols = NULL,
  output_col = NULL,
  levels = 3,
  threshold_multiplier = 1,
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame containing EDA data.
`eda_col`	EDA/conductance column.
`group_cols`	Optional grouping columns.
`output_col`	Optional output column.
`levels`	Number of Haar decomposition levels.
`threshold_multiplier`	Multiplier applied to the robust noise estimate.
`overwrite`	Logical. If `FALSE`, protect existing output columns.

Value

A data frame with denoised EDA and denoising attributes.

`denoise_gazepoint_ppg_autoencoder` *Denoise PPG using a user-supplied autoencoder reconstruction model*

Description

Applies a user-supplied reconstruction function or model to fixed-length PPG windows. No pre-trained neural network is bundled. This function is an interoperability bridge for validated user-supplied autoencoders.

Usage

```
denoise_gazepoint_ppg_autoencoder(
  dat,
  ppg_col = "HRP",
  time_col = NULL,
  group_cols = NULL,
  model = NULL,
  window_samples = 128,
  output_col = NULL,
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame.
`ppg_col`	Numeric PPG/pulse column.
`time_col`	Optional time column.
`group_cols`	Optional grouping columns.
`model`	A user-supplied function or model. If `NULL`, the original signal is copied and status records that no model was supplied.
`window_samples`	Window length in samples.
`output_col`	Optional output column.
`overwrite`	Logical. If `FALSE`, existing output column is protected.

Value

A data frame with reconstructed signal and denoising attributes.

`denoise_gazepoint_quantization_noise` *Add small uniform noise to reduce quantization overlap*

Description

Adds uniform white noise with magnitude tied to hardware resolution. This is intended only for nonlinear phase-space methods that are sensitive to exact repeated values caused by coarse interval quantization.

Usage

```
denoise_gazepoint_quantization_noise(
  dat,
  signal_cols,
  resolution,
  group_cols = NULL,
  output_suffix = "_quantization_jittered",
  seed = NULL,
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame.
`signal_cols`	Numeric signal columns to jitter.
`resolution`	Numeric scalar or named numeric vector giving measurement resolution for each column.
`group_cols`	Optional grouping columns, retained in settings.
`output_suffix`	Suffix for jittered columns.
`seed`	Optional random seed.
`overwrite`	Logical. If `FALSE`, existing output columns are protected.

Value

A data frame with jittered columns and quantization-noise attributes.

detect_active_biometric_channels

Detect active Gazepoint biometric channels

Description

Detects whether GSR/EDA, heart-rate, engagement-dial, and TTL channels are present and whether they appear active. A channel can be present but inactive when validity flags are zero or the signal contains only zeros or missing values. For each signal family, `summary_column` identifies the primary column used for the reported minimum and maximum values.

Usage

```
detect_active_biometric_channels(data)
```

Arguments

`data`	A data frame imported from a Gazepoint export.

Value

A data frame with one row per signal family.

detect_gazepoint_biometric_schema

Detect the schema of Gazepoint biometric data

Description

Detects likely biometric, timing, marker, and identifying columns in a Gazepoint Biometrics export. The function is deliberately descriptive. It reports what appears to be present and active, but it does not infer emotion, valence, or HRV from ambiguous raw columns.

Usage

```
detect_gazepoint_biometric_schema(data)
```

Arguments

`data`	A data frame.

Value

A list with `overview`, `columns`, `time_columns`, `timebase`, `name_map`, and `notes`.

Examples

```
df <- data.frame(
  CNT = 1:5,
  TIME = seq(0, by = 1 / 60, length.out = 5),
  GSR = c(100, 101, 102, 101, 100),
  HR = c(70, 71, 72, 71, 70),
  HRV = c(1, 1, 1, 1, 1)
)
detect_gazepoint_biometric_schema(df)
```

---

detect_gazepoint_biometric_timebase

Detect the likely timebase of Gazepoint biometric data

Description

Inspects timing and counter columns and returns a conservative summary of the likely primary timebase. Sampling rate is estimated only when numeric timing information is available and intervals are positive.

Usage

```
detect_gazepoint_biometric_timebase(data, time_col = NULL, counter_col = NULL)
```

Arguments

`data`	A data frame.
`time_col`	Optional explicit timing column.
`counter_col`	Optional explicit counter column.

Value

A list with `overview`, `time_columns`, `interval_summary`, and `warnings`.

Examples

```
df <- data.frame(CNT = 1:5, TIME = seq(0, by = 1 / 60, length.out = 5))
detect_gazepoint_biometric_timebase(df)
```

`detect_gazepoint_doubly_stochastic_changepoints` *Detect stochastic change points in noisy biometric signals*

Description

Detects abrupt changes in noisy biological time series using a dependency-light stochastic rolling-window approximation. The score combines adjacent-window changes in mean and variance with a robust adaptive threshold.

Usage

```
detect_gazepoint_doubly_stochastic_changepoints(
  dat,
  signal_col,
  time_col = "CNT",
  group_cols = NULL,
  window_seconds = 10,
  step_seconds = 2,
  threshold_mad_multiplier = 6,
  min_distance_s = 5
)
```

Arguments

`dat`	A data frame.
`signal_col`	Numeric signal column.
`time_col`	Numeric time column.
`group_cols`	Optional grouping columns.
`window_seconds`	Window length in seconds.
`step_seconds`	Step size in seconds.
`threshold_mad_multiplier`	Robust threshold multiplier.
`min_distance_s`	Minimum distance between detected change points.

Details

This is not a full reproduction of any specific doubly stochastic model. It is a transparent approximation for QC and exploratory segmentation.

Value

A list with `overview`, `score_table`, `changepoints`, and `settings`.

`detect_gazepoint_scr_events` *Detect SCR-like events in Gazepoint GSR/EDA data*

Description

Detects simple SCR-like peaks from a phasic EDA signal. If a phasic column is not supplied, the function first creates a descriptive phasic component using `decompose_gazepoint_eda()`. This helper is intended for exploratory quality control and descriptive summaries, not as a replacement for specialised SCR detection pipelines.

Usage

```
detect_gazepoint_scr_events(
  data,
  phasic_col = NULL,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  threshold = NULL,
  min_peak_distance = 10L,
  window_size = 31L
)
```

Arguments

`data`	A data frame.
`phasic_col`	Optional phasic EDA column.
`signal_col`	Optional raw/conductance EDA column used when `phasic_col` is not supplied.
`time_col`	Optional time/order column.
`group_cols`	Optional grouping columns.
`threshold`	Optional numeric detection threshold. If `NULL`, a robust group-specific threshold is estimated as median plus three MADs, bounded below by zero.
`min_peak_distance`	Minimum distance between retained peaks in samples.
`window_size`	Rolling-median window size used if decomposition is needed.

Value

A list with `overview`, `events`, `group_summary`, and `settings`.

Examples

```
df <- data.frame(
  CNT = 1:20,
  GSR_US_PHASIC = c(rep(0, 5), 0.2, 0.8, 0.2, rep(0, 12))
)
detect_gazepoint_scr_events(df, phasic_col = "GSR_US_PHASIC", time_col = "CNT")
```

---

detect_gazepoint_scr_peaks

Detect Gazepoint SCR peaks

Description

Detects candidate skin conductance responses (SCRs) from Gazepoint EDA/GSR signals. The helper prefers a phasic channel such as `GSR_US_PHASIC` when available, and otherwise falls back to a conductance-like signal such as `GSR_US`. It returns explicit onset, peak, amplitude, rise-time, and recovery-time fields for downstream event-window summaries and statistical modelling.

Usage

```
detect_gazepoint_scr_peaks(
  data,
  signal_col = NULL,
  phasic_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  prefer_vendor_phasic = TRUE,
  amplitude_min = 0.01,
  recovery_fraction = 0.5,
  smooth_width = 1,
  min_peak_distance = 1
)
```

Arguments

`data`	A data frame containing Gazepoint biometric rows.
`signal_col`	Optional conductance-like signal column, typically `GSR_US`. Used when `phasic_col` is absent or unavailable.
`phasic_col`	Optional phasic EDA signal column, typically `GSR_US_PHASIC`.
`time_col`	Optional time/counter column. If `NULL`, common Gazepoint time columns are detected automatically.
`group_cols`	Optional grouping columns. If `NULL`, available source/participant/media/trial-like columns are used.
`prefer_vendor_phasic`	Logical. If `TRUE`, prefer `GSR_US_PHASIC` when available.

`amplitude_min` Minimum trough-to-peak amplitude required for a detected SCR.

`recovery_fraction`
Fraction of the peak amplitude used to define recovery. The default `.5` estimates half-recovery.

`smooth_width` Optional odd integer moving-average width. Use `1` for no smoothing.

`min_peak_distance`
Minimum distance, in rows, allowed between retained candidate peaks within each group. The default `1` preserves all local maxima. Larger values reduce repeated detection of closely spaced local maxima within a sustained SCR-like response.

Details

This is a conservative R-native peak detector. It is not a replacement for full model-based EDA decomposition tools such as Ledalab, PsPM, or cvxEDA.

Value

A list with `overview`, `peaks`, `group_summary`, `signal_summary`, and `settings`.

`detect_gazepoint_time_columns`
Detect Gazepoint biometric time columns

Description

Detects likely timing and counter columns in Gazepoint Biometrics exports. The function reports candidate timing columns rather than assuming that any single time variable is always present or always measured in the same unit.

Usage

```
detect_gazepoint_time_columns(data)
```

Arguments

`data` A data frame or a character vector of column names.

Value

A data frame with one row per detected time-related column.

Examples

```
detect_gazepoint_time_columns(c("CNT", "TIME_MS", "GSR", "HR"))
```

`diagnose_gazepoint_biometrics_workflow` *Diagnose a Gazepoint Biometrics workflow*

Description

Creates a compact diagnostic readiness summary from a Gazepoint Biometrics workflow object. The function does not change or remove data. It returns a one-row decision table with pass/review/fail status and concise reasons.

Usage

```
diagnose_gazepoint_biometrics_workflow(
  workflow,
  require_gsr = TRUE,
  require_hr = TRUE,
  require_dial = FALSE,
  max_exclude_window_pct = 25,
  max_review_window_pct = 25
)
```

Arguments

`workflow`	A workflow object produced by `run_gazepoint_biometrics_workflow()`.
`require_gsr`	Should GSR/EDA be required for a pass status?
`require_hr`	Should heart rate be required for a pass status?
`require_dial`	Should engagement dial be required for a pass status?
`max_exclude_window_pct`	Maximum acceptable percentage of excluded windows before the workflow is marked as fail.
`max_review_window_pct`	Maximum acceptable percentage of review windows before the workflow is marked as review.

Value

A one-row data frame with diagnostic status and reasons.

`estimate_gazepoint_signal_lag` *Estimate lag between two Gazepoint biometric signals*

Description

Estimates the time shift that maximizes the association between two recorded biometric signals within each group. This is a conservative synchronization diagnostic for inspecting whether two recorded traces show similar temporal structure at different shifts. It should not be interpreted as causal timing or true physiological latency unless the design includes appropriate event markers and independently justified signal-processing assumptions.

Usage

```
estimate_gazepoint_signal_lag(
  data,
  signal_x_col,
  signal_y_col,
  time_col = NULL,
  group_cols = NULL,
  max_lag = 1000,
  lag_step = NULL,
  method = c("pearson", "spearman"),
  min_complete_pairs = 20,
  use_first_difference = FALSE
)
```

Arguments

`data`	A Gazepoint biometric data frame.
`signal_x_col`	Name of the first signal column.
`signal_y_col`	Name of the second signal column.
`time_col`	Optional time or counter column. If `NULL`, a common Gazepoint time/counter column is detected.
`group_cols`	Optional grouping columns, such as participant, stimulus, trial, or source file.
`max_lag`	Maximum absolute lag to evaluate, in the same units as `time_col`.
`lag_step`	Step size between candidate lags, in the same units as `time_col`. If `NULL`, the median positive time step is used.
`method`	Correlation method passed to `stats::cor()`.
`min_complete_pairs`	Minimum complete aligned observations required for a candidate lag.
`use_first_difference`	If `TRUE`, correlations are estimated on first differences rather than raw signal levels.

Value

A list with `overview`, `lag_by_group`, `lag_profile`, and `settings`.

`export_gazepoint_biometrics_report_bundle`
Export a Gazepoint biometrics report bundle

Description

Exports selected report tables, text outputs, optional plot objects, and a manifest to a local output directory. This helper is intended for reproducible reporting. It does not commit files and should not be used to export private real-data outputs into a package repository.

Usage

```
export_gazepoint_biometrics_report_bundle(
  bundle = NULL,
  output_dir,
  prefix = "gpbiometrics_report",
  tables = NULL,
  text = NULL,
  plots = NULL,
  include_readme = TRUE,
  include_session_info = TRUE,
  overwrite = FALSE
)
```

Arguments

`bundle`	Optional list-like object containing data frames, text, or plots.
`output_dir`	Output directory.
`prefix`	File prefix.
`tables`	Optional named list of data frames to export as CSV files.
`text`	Optional named list or character vector of text outputs to export as TXT files.
`plots`	Optional named list of ggplot objects to export as PNG files.
`include_readme`	Logical. Should a README text file be written?
`include_session_info`	Logical. Should session information be written?
`overwrite`	Logical. Should existing files be overwritten?

Value

A list with `overview`, `manifest`, `output_dir`, and `settings`.

export_gazepoint_rhrv_input *Export Gazepoint IBI data for RHRV-style workflows*

Description

Prepares cleaned inter-beat interval data for external HRV workflows. The helper writes simple beat tables with cumulative beat time and IBI/RR interval columns. It does not use the Gazepoint HRV column.

Usage

```
export_gazepoint_rhrv_input(
  data,
  ibi_col = "IBI_clean_ms",
  group_cols = NULL,
  unit = c("auto", "ms", "seconds"),
  collapse_repeated_intervals = TRUE,
  repeated_tolerance_ms = 1e-08,
  min_ibi_ms = 300,
  max_ibi_ms = 2000,
  output_dir = NULL,
  prefix = "gazepoint_rhrv"
)
```

Arguments

data	A Gazepoint biometric data frame or gazepoint_ibi_filter object.
ibi_col	IBI/RR interval column.
group_cols	Optional grouping columns.
unit	Unit of the IBI column: "auto", "ms", or "seconds".
collapse_repeated_intervals	Logical. If TRUE, consecutive repeated IBI values are collapsed before export.
repeated_tolerance_ms	Numeric tolerance used when identifying repeated consecutive IBI values.
min_ibi_ms	Minimum plausible IBI in milliseconds retained for export.
max_ibi_ms	Maximum plausible IBI in milliseconds retained for export.
output_dir	Optional directory where per-group CSV files are written.
prefix	File prefix used when output_dir is supplied.

Value

A list with overview, beat_table, group_summary, manifest, and settings.

`extract_gazepoint_beats_kmeans` *Extract heartbeat candidates from Gazepoint pulse using k-means*

Description

Uses k-means clustering on the raw pulse waveform to classify likely heartbeat regions and then selects local extrema as beat candidates. This is a Gazepoint Biometrics-oriented fallback for difficult pulse waveforms, not an ECG-equivalent R-peak detector.

Usage

```
extract_gazepoint_beats_kmeans(
  dat,
  pulse_col = "HRP",
  time_col = "CNT",
  group_cols = NULL,
  k = 2,
  peak_polarity = c("positive", "negative"),
  min_distance_s = 0.3,
  sampling_rate = NULL,
  seed = NULL
)
```

Arguments

`dat`	A data frame.
`pulse_col`	Numeric pulse/PPG column.
`time_col`	Numeric time column.
`group_cols`	Optional grouping columns.
`k`	Number of k-means clusters.
`peak_polarity`	`"positive"` or `"negative"`.
`min_distance_s`	Minimum time between selected beats.
`sampling_rate`	Optional sampling rate in Hz.
`seed`	Optional random seed.

Value

A list with `overview`, `beat_table`, `interval_table`, `timeseries`, and `settings`.

`extract_gazepoint_bilateral_eda_asymmetry` *Extract bilateral EDA asymmetry features*

Description

Computes left-right electrodermal activity asymmetry descriptors from two simultaneously recorded EDA channels. The function returns row-level asymmetry time series and group-level summaries.

Usage

```
extract_gazepoint_bilateral_eda_asymmetry(
  dat,
  left_col,
  right_col,
  time_col = NULL,
  group_cols = NULL,
  output_prefix = "beda"
)
```

Arguments

`dat`	A data frame.
`left_col`	Numeric left-side EDA column.
`right_col`	Numeric right-side EDA column.
`time_col`	Optional numeric time column for ordering and gradient calculation.
`group_cols`	Optional grouping columns.
`output_prefix`	Prefix used for row-level output columns.

Details

These descriptors quantify bilateral EDA differences only. They do not infer hemisphere activation, amygdala activity, psychopathology, emotion, stress, cognition, health status, or diagnosis.

Value

A list with `overview`, `asymmetry_timeseries`, `summary`, and `settings`.

extract_gazepoint_eda_complexity

Extract EDA complexity features

Description

Computes dependency-light EDA complexity descriptors, including sample entropy and detrended fluctuation analysis alpha.

Usage

```
extract_gazepoint_eda_complexity(
  dat,
  eda_col = "GSR_US",
  group_cols = NULL,
  min_samples = 32,
  sampen_m = 2,
  sampen_r_multiplier = 0.2
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	EDA/conductance column.
group_cols	Optional grouping columns.
min_samples	Minimum finite samples per group.
sampen_m	Embedding dimension for sample entropy.
sampen_r_multiplier	Tolerance multiplier applied to within-group SD.

Value

A list with `overview`, `features`, and `settings`.

extract_gazepoint_eda_spectral_power

Extract frequency-domain EDA spectral power

Description

Computes power spectral density summaries for an EDA signal, including spectral power in the EDASymp-inspired 0.045–0.25 Hz band. This is a descriptive spectral feature and should not be interpreted as direct stress, emotion, valence, cognition, trust, preference, or diagnosis.

Usage

```
extract_gazepoint_eda_spectral_power(
  dat,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  band = c(0.045, 0.25),
  min_samples = 32,
  detrend = TRUE
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	EDA/conductance column.
time_col	Optional time column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz. Required if time_col does not allow sampling-rate estimation.
band	Numeric vector of length two defining the frequency band in Hz.
min_samples	Minimum finite samples per group.
detrend	Logical. If TRUE, remove a linear trend before spectral analysis.

Value

A list with overview, spectral_summary, settings, and interpretation text.

---

extract_gazepoint_eda_tvsymp

Extract time-varying spectral EDA features

Description

Computes a dependency-light approximation of TVSymp-style time-varying spectral EDA power using sliding-window spectral analysis. The default band is 0.08–0.24 Hz, following the TVSymp literature. This function does not claim exact VFCDM reproduction.

Usage

```
extract_gazepoint_eda_tvsymp(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  group_cols = NULL,
```

```
  sampling_rate = NULL,
  band = c(0.08, 0.24),
  window_seconds = 60,
  step_seconds = 5,
  min_valid_fraction = 0.7,
  normalise = TRUE
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	Numeric EDA/conductance column.
time_col	Numeric time column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz. If NULL, estimated from time_col.
band	Frequency band in Hz used for TVSymp-style power.
window_seconds	Sliding-window length in seconds.
step_seconds	Sliding-window step in seconds.
min_valid_fraction	Minimum valid fraction per window.
normalise	Logical. If TRUE, compute EDASympn-style relative band power normalised by total positive-frequency power.

Value

A list with overview, tvsymp_timeseries, summary, and settings.

---

extract_gazepoint_edr_pca

Extract ECG-derived respiration using PCA

Description

Extracts an ECG-derived respiration proxy from beat-level ECG morphology features using principal component analysis. This function requires ECG-derived morphology columns, such as QRS amplitudes, widths, or sampled beat-shape features. It is not intended for HR, IBI, or PPG-only data.

Usage

```
extract_gazepoint_edr_pca(
  dat,
  ecg_cols,
  time_col = NULL,
  group_cols = NULL,
```

```
  n_components = 1,
  scale = TRUE,
  output_prefix = "edr_pca"
)
```

Arguments

dat	A data frame.
ecg_cols	Numeric ECG morphology columns.
time_col	Optional time column.
group_cols	Optional grouping columns.
n_components	Number of PCA components to retain.
scale	Logical. If TRUE, scale ECG morphology columns before PCA.
output_prefix	Prefix for PCA output columns.

Value

A list with overview, edr_timeseries, component_summary, and settings.

---

extract_gazepoint_hrv_asymmetry

Extract heart-rate asymmetry features

Description

Computes dependency-light heart-rate asymmetry descriptors from IBI/RR intervals, including acceleration/deceleration proportions, signed run summaries, and Guzik-style squared-difference asymmetry.

Usage

```
extract_gazepoint_hrv_asymmetry(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  zero_tolerance = 0
)
```

Arguments

dat	A data frame containing IBI/RR intervals.
ibi_col	Numeric IBI/RR interval column.
group_cols	Optional grouping columns.
zero_tolerance	Absolute change below which interval differences are treated as zero.

Details

Positive IBI/RR differences are treated as decelerations because the heart period lengthens. Negative IBI/RR differences are treated as accelerations.

Value

A list with `overview`, `features`, `run_table`, and `settings`.

`extract_gazepoint_hrv_features` *Extract time-domain HRV features from Gazepoint IBI intervals*

Description

Computes simple time-domain HRV features from genuine IBI/RR intervals. This helper does not use the Gazepoint `HRV` column as an HRV outcome.

Usage

```
extract_gazepoint_hrv_features(
  data,
  ibi_col = "IBI_clean_ms",
  group_cols = NULL,
  unit = c("auto", "ms", "seconds"),
  min_intervals = 3,
  min_duration_s = 30,
  diff_threshold_ms = 50,
  collapse_repeated_intervals = TRUE,
  repeated_tolerance_ms = 1e-08
)
```

Arguments

`data`	A Gazepoint biometric data frame or `gazepoint_ibi_filter` object.
`ibi_col`	IBI/RR interval column.
`group_cols`	Optional grouping columns.
`unit`	Unit of the IBI column: `"auto"`, `"ms"`, or `"seconds"`.
`min_intervals`	Minimum clean intervals required per group.
`min_duration_s`	Minimum IBI-sequence duration in seconds required before computed HRV features are treated as fully reportable. Groups below this duration still return features but receive `warn_short_hrv_duration`.
`diff_threshold_ms`	Threshold for NN50/pNN50.

collapse_repeated_intervals
: Logical. If TRUE, consecutive repeated IBI values are collapsed before HRV features are computed. This is useful for Gazepoint exports where the same IBI value may be repeated across multiple gaze-sampling rows until a new interval is available.

repeated_tolerance_ms
: Numeric tolerance used when identifying repeated consecutive IBI values.

Value

A list with overview, features, settings.

---

extract_gazepoint_hrv_fragmentation

Extract heart-rate fragmentation features

Description

Computes dependency-light heart-rate fragmentation descriptors from IBI/RR intervals. Metrics include percentage of inflection points (PIP), inverse average segment length (IALS), percentage of short segments (PSS), percentage of alternation segments (PAS), and long/short segment summaries.

Usage

```
extract_gazepoint_hrv_fragmentation(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  zero_tolerance = 0,
  short_segment_length = 3
)
```

Arguments

dat	A data frame containing IBI/RR intervals.
ibi_col	Numeric IBI/RR interval column.
group_cols	Optional grouping columns.
zero_tolerance	Absolute change below which interval differences are treated as zero.
short_segment_length	Maximum segment length counted as short.

Details

These are fragmentation descriptors of interbeat interval dynamics. They should not be interpreted as clinical diagnoses or direct autonomic-state labels by themselves.

Value

A list with `overview`, `features`, and `settings`.

`extract_gazepoint_hrv_fuzzy_csi` *Extract FuzzyEn and Lorenz-plot CSI HRV features*

Description

Computes fuzzy entropy and Lorenz/Poincare-derived cardiac sympathetic index style descriptors from IBI/RR intervals.

Usage

```
extract_gazepoint_hrv_fuzzy_csi(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  m = 2,
  r_multiplier = 0.2,
  fuzzy_power = 2,
  min_intervals = 10
)
```

Arguments

`dat`	A data frame.
`ibi_col`	Numeric IBI/RR interval column.
`group_cols`	Optional grouping columns.
`m`	Embedding dimension.
`r_multiplier`	Tolerance multiplier applied to within-group SD.
`fuzzy_power`	Fuzzy exponential power.
`min_intervals`	Minimum intervals per group.

Details

These outputs are nonlinear/geometric HRV descriptors. They do not infer seizure status, diagnosis, health status, emotion, stress, or cognition.

Value

A list with `overview`, `features`, and `settings`.

extract_gazepoint_hrv_geometric

Extract geometric HRV features

Description

Computes dependency-light geometric HRV descriptors, including the HRV triangular index and an approximate TINN-style triangular interpolation width.

Usage

```
extract_gazepoint_hrv_geometric(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  bin_width = NULL
)
```

Arguments

dat	A data frame.
ibi_col	Numeric IBI/RR interval column.
group_cols	Optional grouping columns.
bin_width	Histogram bin width in the same units as ibi_col.

Value

A list with overview, features, and settings.

extract_gazepoint_hrv_nonlinear

Extract nonlinear HRV features from IBI/RR intervals

Description

Computes dependency-light nonlinear HRV descriptors from IBI/RR intervals, including Poincare SD1/SD2, sample entropy, approximate entropy, multiscale entropy, and detrended fluctuation analysis.

Usage

```
extract_gazepoint_hrv_nonlinear(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  min_intervals = 10,
  sampen_m = 2,
  sampen_r_multiplier = 0.2,
  mse_scales = 1:5
)
```

Arguments

`dat`	A data frame containing IBI/RR intervals.
`ibi_col`	IBI/RR interval column.
`group_cols`	Optional grouping columns.
`min_intervals`	Minimum finite intervals per group.
`sampen_m`	Embedding dimension for sample entropy.
`sampen_r_multiplier`	Tolerance multiplier applied to the within-group SD.
`mse_scales`	Integer scales used for multiscale entropy.

Details

These are variability and complexity descriptors. They should not be interpreted as direct emotion, cognitive-load, health-status, or diagnostic labels by themselves.

Value

A list with `overview`, `features`, and `settings`.

`extract_gazepoint_hrv_rcmse` *Extract refined composite multiscale entropy from HRV intervals*

Description

Computes refined composite multiscale entropy (RCMSE) from IBI/RR intervals. RCMSE pools template-match counts across all coarse-grained offsets at each scale, making it more stable than ordinary MSE for shorter physiological time series.

Usage

```
extract_gazepoint_hrv_rcmse(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  scales = 1:10,
  m = 2,
  r_multiplier = 0.2,
  min_intervals = 20
)
```

Arguments

dat	A data frame containing IBI/RR intervals.
ibi_col	Numeric IBI/RR interval column.
group_cols	Optional grouping columns.
scales	Positive integer scales.
m	Embedding dimension.
r_multiplier	Tolerance multiplier applied to SD.
min_intervals	Minimum intervals per group.

Value

A list with overview, rcmse_by_scale, summary, and settings.

---

extract_gazepoint_hrv_rqa

Extract HRV recurrence quantification analysis features

Description

Computes dependency-light recurrence quantification analysis (RQA) features from IBI/RR intervals. This is intended as a compact nonlinear HRV summary and not as a clinical diagnostic tool.

Usage

```
extract_gazepoint_hrv_rqa(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  embedding_dimension = 2,
  delay = 1,
  radius = NULL,
  radius_multiplier = 0.2,
  min_line_length = 2
)
```

Arguments

dat	A data frame.
ibi_col	Numeric IBI/RR interval column.
group_cols	Optional grouping columns.
embedding_dimension	Embedding dimension for phase-space reconstruction.
delay	Delay used in embedding.
radius	Radius for recurrence threshold. If NULL, uses radius_multiplier * SD.
radius_multiplier	Multiplier used when radius = NULL.
min_line_length	Minimum diagonal/vertical line length.

Value

A list with overview, features, and settings.

---

extract_gazepoint_pdr_signals

Extract PPG-derived respiration proxy signals

Description

Extracts dependency-light PPG-derived respiration proxy features from a Gazepoint pulse/PPG waveform. The function estimates respiration-modulated pulse features such as respiratory-induced intensity variability (RIIV), pulse amplitude variability (PAV), pulse width variability (PWV), and pulse-rate variability (PRV).

Usage

```
extract_gazepoint_pdr_signals(
  dat,
  ppg_col = "HRP",
  time_col = "CNT",
  group_cols = NULL,
  sampling_rate = NULL,
  min_peak_distance_s = 0.3,
  smooth_window = 5,
  respiration_band = c(0.1, 0.6),
  pdr_resample_rate = 4
)
```

Arguments

dat	A data frame containing a PPG/pulse waveform.
ppg_col	Numeric PPG/pulse waveform column, often HRP.
time_col	Numeric time column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz. If NULL, estimated from time_col.
min_peak_distance_s	Minimum plausible distance between pulse peaks.
smooth_window	Number of samples used for simple moving-average smoothing before peak detection.
respiration_band	Expected respiration-frequency band in Hz.
pdr_resample_rate	Resampling rate used for spectral estimation of PDR proxy signals.

Details

These are proxy respiratory features. They should not be treated as a replacement for a respiration belt or clinical respiratory measurement.

Value

A list with overview, pulse_features, pdr_timeseries, pdr_summary, and settings.

---

extract_gazepoint_respiration_ceemdan

Extract respiration proxy using a CEEMDAN-style bridge

Description

Extracts respiration-like components from PPG or ECG-derived respiratory proxy signals. If external_fun is supplied, it is used as the CEEMDAN backend. Otherwise, the function uses a dependency-light multiscale decomposition fallback and labels the result accordingly.

Usage

```
extract_gazepoint_respiration_ceemdan(
  dat,
  signal_col,
  time_col = "CNT",
  group_cols = NULL,
  sampling_rate = NULL,
  respiration_band = c(0.1, 0.6),
  scales = c(5, 15, 30, 60, 120),
  external_fun = NULL
)
```

Arguments

dat	A data frame.
signal_col	Numeric PPG/ECG-derived signal column.
time_col	Numeric time column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz.
respiration_band	Frequency band in Hz used to select respiration-like components.
scales	Moving-average scales used by the fallback decomposition.
external_fun	Optional function with arguments x, time, and sampling_rate, returning either a numeric vector or a list/data frame of components.

Details

This function does not claim to reproduce full CEEMDAN unless a validated external CEEMDAN function is supplied.

Value

A list with overview, component_table, respiration_timeseries, summary, and settings.

---

extract_gazepoint_scr_recovery_times

Extract SCR recovery times

Description

Extracts 50 percent half-recovery time (rec.t2) and 63 percent recovery time (rec.tc) for skin conductance responses from an EDA waveform and event onsets.

Usage

```
extract_gazepoint_scr_recovery_times(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  event_onset_col = NULL,
  group_cols = NULL,
  pre_onset_baseline_s = 2,
  peak_window_s = 5,
  recovery_window_s = 20
)
```

Arguments

dat	A data frame.
eda_col	Numeric EDA/conductance column.
time_col	Numeric time column.
event_onset_col	Optional event onset column. Finite values are treated as event onsets.
group_cols	Optional grouping columns.
pre_onset_baseline_s	Baseline window before event onset.
peak_window_s	Window after onset used to find the response peak.
recovery_window_s	Window after peak used to find recovery.

Value

A list with overview, recovery_table, and settings.

---

extract_gazepoint_ttl_events

Extract Gazepoint TTL marker events

Description

Extracts TTL marker events from Gazepoint Biometrics exports. The function can return either rows where TTL marker values change or all nonzero TTL rows. By default, rows are retained only when the TTL validity column is present and greater than zero. This avoids treating invalid placeholder TTL values as experimental events.

Usage

```
extract_gazepoint_ttl_events(
  data,
  ttl_columns = NULL,
  group_columns = NULL,
  validity_column = "TTLV",
  require_validity = TRUE,
  mode = c("changes", "nonzero"),
  include_initial = TRUE
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
ttl_columns	TTL marker columns. If NULL, the function uses all available columns from TTL0 to TTL6.
group_columns	Optional grouping columns within which TTL changes are detected, such as source_participant, USER, or MEDIA_ID.
validity_column	Optional TTL validity column. Defaults to "TTLV".
require_validity	Logical. Should rows be retained only when validity_column is present and greater than zero? Defaults to TRUE.
mode	Event extraction mode. "changes" returns rows where TTL values change. "nonzero" returns all rows with nonzero TTL values.
include_initial	Should the first valid, non-missing TTL value within each group be treated as an event when mode = "changes"?

Value

A data frame of TTL events.

---

filter_gazepoint_ibi_implausible

Filter implausible Gazepoint IBI values

Description

Flags and optionally cleans implausible inter-beat interval values. The helper is conservative and does not remove rows; instead, it returns row-level flags and a cleaned IBI column with implausible values set to NA.

Usage

```
filter_gazepoint_ibi_implausible(
  data,
  ibi_col = "IBI",
  time_col = NULL,
  group_cols = NULL,
  validity_col = NULL,
  unit = c("auto", "ms", "seconds"),
  min_ibi_ms = 300,
  max_ibi_ms = 2000,
  max_change_ms = 400,
  max_change_prop = 0.3,
  output_col = "IBI_clean_ms"
)
```

Arguments

data	A Gazepoint biometric data frame.
ibi_col	IBI/RR interval column.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
validity_col	Optional validity column.
unit	Unit of the IBI column: "auto", "ms", or "seconds".
min_ibi_ms	Minimum plausible IBI in milliseconds.
max_ibi_ms	Maximum plausible IBI in milliseconds.
max_change_ms	Maximum plausible absolute adjacent IBI change within group, in milliseconds.
max_change_prop	Maximum plausible proportional adjacent IBI change within group.
output_col	Name of the cleaned IBI output column.

Value

A list with overview, data, row_flags, group_summary, and settings.

---

flag_gazepoint_artifacts_svm

Flag EDA artifacts with a user-supplied SVM-style model

Description

Applies a user-supplied model or prediction function to segment-level artifact features. No pre-trained model is bundled, so this function avoids pretending to reproduce any proprietary or externally trained classifier.

Usage

```
flag_gazepoint_artifacts_svm(
  x,
  model = NULL,
  feature_cols = NULL,
  probability_threshold = 0.5,
  ...
)
```

Arguments

x	Either raw EDA data or output from prepare_gazepoint_artifact_svm_features().
model	Optional model object or function. If NULL, features are returned with missing artifact predictions.
feature_cols	Optional feature columns used by the model.
probability_threshold	Threshold for artifact probability.
...	Passed to prepare_gazepoint_artifact_svm_features() when x is raw data.

Value

A data frame with artifact probabilities/classes where available.

---

flag_gazepoint_biometric_dropouts

Flag biometric dropouts and flatline periods

Description

Flags missing-value runs and sustained flatline runs in Gazepoint biometric signal columns. Missing dropouts are defined as consecutive missing or non-finite numeric samples. Flatline dropouts are defined as consecutive finite numeric samples that remain unchanged within a tolerance.

Usage

```
flag_gazepoint_biometric_dropouts(
  data,
  signal_cols = NULL,
  group_cols = NULL,
  time_col = NULL,
  min_missing_run = 5L,
  min_flatline_run = 10L,
  constant_tolerance = 0,
  prefix = "biometric_dropout"
)
```

Arguments

data	A data frame.
signal_cols	Optional character vector of biometric signal columns. If NULL, common Gazepoint biometric signal columns are detected.
group_cols	Optional grouping columns. Runs are computed separately within each group.
time_col	Optional time column used to order rows within each group before run detection. If NULL, the current row order is used.

`min_missing_run`
: Minimum consecutive missing/non-finite samples required to flag a missing dropout.

`min_flatline_run`
: Minimum consecutive unchanged finite samples required to flag a flatline dropout.

`constant_tolerance`
: Numeric tolerance used when detecting unchanged values for flatline runs.

`prefix`
: Prefix for generated dropout columns.

Details

The function adds row-level flags and stores a dropout summary in the returned data frame attributes. It does not remove rows.

Value

The input data frame with added logical dropout columns. The attributes `dropout_summary` and `dropout_settings` contain structured summaries.

Examples

```
df <- data.frame(
  CNT = 1:8,
  GSR = c(1, NA, NA, NA, 2, 2, 2, 3),
  HR = c(70, 71, 72, 73, 74, 75, 76, 77)
)
flag_gazepoint_biometric_dropouts(df, min_missing_run = 3, min_flatline_run = 3)
```

---

flag_gazepoint_mad_artifacts

Flag MAD-based EDA wearable artifacts

Description

Flags dependency-light, subject-specific EDA artifact categories using robust median absolute deviation (MAD) logic. The categories are heuristic QC labels: step artifacts, needle artifacts, flatline artifacts, and wall artifacts.

Usage

```
flag_gazepoint_mad_artifacts(
  dat,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  mad_multiplier = 8,
  flatline_tolerance = 1e-06,
```

```
  flatline_min_run = 5,
  wall_abs_change = NULL,
  output_prefix = "mad"
)
```

Arguments

dat	A data frame.
eda_col	Numeric EDA/conductance column.
time_col	Optional time column for ordering within group.
group_cols	Optional grouping columns.
mad_multiplier	MAD multiplier used for robust thresholding.
flatline_tolerance	Maximum absolute sample-to-sample change treated as flatline.
flatline_min_run	Minimum consecutive flatline samples.
wall_abs_change	Optional absolute change threshold for wall artifacts.
output_prefix	Prefix for output columns.

Value

A data frame with artifact flags and artifact-summary attributes.

---

flag_kleckner_eda_artifacts

Flag EDA artifacts using transparent Kleckner-style heuristics

---

Description

Applies simple transparent EDA artifact flags: non-finite values, physiological range violations, rapid percent change per second, and transitional padding around flagged samples.

Usage

```
flag_kleckner_eda_artifacts(
  dat,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  min_us = 0.01,
  max_us = 100,
  max_abs_percent_change_per_second = 20,
  transition_padding = 1,
  output_prefix = "kleckner"
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	Conductance column in microsiemens.
time_col	Optional time column.
group_cols	Optional grouping columns.
min_us	Minimum plausible conductance.
max_us	Maximum plausible conductance.
max_abs_percent_change_per_second	Maximum absolute percent change per second before flagging.
transition_padding	Number of neighbouring rows to flag around bad samples within each group.
output_prefix	Prefix for output columns.

Details

This helper is Kleckner-style rather than a claim of exact reproduction of every rule in a specific external implementation.

Value

A data frame with artifact flag columns and summary attributes.

---

format_gazepoint_biometrics_feature_inventory *Format the gpbiometrics feature inventory for users*

Description

Adds user-facing labels and interpretation metadata to the package feature inventory. This helper is intentionally non-breaking: it does not replace the core inventory object returned by create_gazepoint_biometrics_featu

Usage

```
format_gazepoint_biometrics_feature_inventory(
  inventory = NULL,
  include_internal = FALSE,
  sort = TRUE
)
```

Arguments

inventory	Optional inventory object returned by create_gazepoint_biometrics_feature_inventory(). If NULL, a fresh inventory is created.
include_internal	Logical. Passed to create_gazepoint_biometrics_feature_inventory() when inventory = NULL.
sort	Logical. If TRUE, sort by domain, user level, and function name.

Value

A data frame with polished user-facing inventory columns.

`fuse_gazepoint_respiration_kalman` *Fuse respiration proxies using a Kalman filter*

Description

Fuses two respiration proxy streams, such as PPG-derived respiration and ECG-derived respiration, using a transparent one-dimensional Kalman filter. This is a linear Kalman fusion helper. It is not an extended Kalman filter unless the user supplies nonlinear state/measurement logic externally.

Usage

```
fuse_gazepoint_respiration_kalman(
  dat,
  primary_col,
  secondary_col,
  time_col = NULL,
  group_cols = NULL,
  process_var = 0.01,
  primary_var = 0.05,
  secondary_var = 0.05,
  output_col = "respiration_kalman_fused"
)
```

Arguments

`dat`	A data frame.
`primary_col`	First respiration proxy column.
`secondary_col`	Second respiration proxy column.
`time_col`	Optional time column.
`group_cols`	Optional grouping columns.
`process_var`	Process variance.
`primary_var`	Measurement variance for `primary_col`.
`secondary_var`	Measurement variance for `secondary_col`.
`output_col`	Output fused respiration column.

Value

A data frame with fused respiration output and Kalman attributes.

get_gazepoint_plot_data *Extract stored plot data*

Description

Extract stored plot data

Usage

```
get_gazepoint_plot_data(plot)
```

Arguments

plot	A plot object returned by a gpbiometrics plotting helper.

Value

A data frame.

get_gazepoint_plot_settings *Extract stored plot settings*

Description

Extract stored plot settings

Usage

```
get_gazepoint_plot_settings(plot)
```

Arguments

plot	A plot object returned by a gpbiometrics plotting helper.

Value

A list.

import_gazepoint_biometrics

Import a Gazepoint Biometrics export

Description

Reads a Gazepoint CSV export containing biometric columns such as GSR, heart rate, interbeat interval, pulse signal, engagement dial, and TTL synchronization fields. The function is conservative: it preserves original column names, removes only empty trailing columns, and attaches a basic biometric-column summary as an attribute.

Usage

```
import_gazepoint_biometrics(file, na = c("", "NA", "NaN"))
```

Arguments

file	Path to a Gazepoint CSV export.
na	Values that should be treated as missing.

Value

A data frame with Gazepoint export columns preserved. The returned object has class "gazepoint_biometrics" and an attribute named "biometric_columns".

import_gazepoint_biometric_folder

Import a folder of Gazepoint Biometrics exports

Description

Reads rectangular Gazepoint CSV exports from a folder and combines files that contain at least one known Gazepoint Biometrics column. This function is designed for all-gaze and fixation-style exports. Multi-section Gazepoint Data Summary files should be parsed separately.

Usage

```
import_gazepoint_biometric_folder(
  path,
  pattern = "\\.csv$",
  recursive = FALSE,
  include_fixations = TRUE,
  include_all_gaze = TRUE,
  include_other_csv = FALSE,
  na = c("", "NA", "NaN")
)
```

Arguments

path	Folder containing Gazepoint CSV exports.
pattern	Regular expression used to identify candidate CSV files.
recursive	Should subfolders be searched?
include_fixations	Should files with "fixation" in the file name be included?
include_all_gaze	Should files with "all_gaze" in the file name be included?
include_other_csv	Should other CSV files be attempted? The default is FALSE to avoid accidentally trying to parse multi-section Data_Summary_export files as rectangular data.
na	Values that should be treated as missing.

Value

A data frame with all imported rows combined. The output includes a source_file column and has class "gazepoint_biometrics_folder".

---

import_gazepoint_data_summary

Import a Gazepoint Data Summary export

Description

Reads a multi-section Data_Summary_export_*.csv file produced by Gazepoint Analysis. These files are not ordinary rectangular CSV files. They contain metadata followed by sections such as AOI Summary and AOI Statistics (for each user). The latter may include AOI-level biometric summaries such as average dial value, average GSR, average heart rate, average interbeat interval, and pupil diameter.

Usage

```
import_gazepoint_data_summary(file)
```

Arguments

file	Path to a Gazepoint Data_Summary_export_*.csv file.

Value

A list with metadata, aoi_summary, and aoi_statistics data frames. The returned object has class "gazepoint_data_summary".

import_gazepoint_lsl_xdf

Import Gazepoint-related streams from an LSL/XDF file

Description

Reads an XDF file through the optional Python `pyxdf` package via `reticulate`. This supports high-end LSL workflows without making Python a hard dependency of gpbiometrics.

Usage

```
import_gazepoint_lsl_xdf(
  path,
  stream_name_pattern = "Gazepoint|GP3|GSR|EDA|Biometric|TTL|Pupil|Gaze",
  include_all_streams = FALSE,
  flatten = TRUE,
  pyxdf_module = "pyxdf"
)
```

Arguments

`path`	Path to an `.xdf` file.
`stream_name_pattern`	Regular expression used to identify Gazepoint-like streams when `include_all_streams = FALSE`.
`include_all_streams`	Logical. If `TRUE`, return all streams.
`flatten`	Logical. If `TRUE`, convert streams to data frames where possible.
`pyxdf_module`	Python module name, usually `"pyxdf"`.

Value

A list with `overview`, `streams`, `header`, and `settings`.

join_gazepoint_biometrics_to_gp3tools

Join Gazepoint Biometrics data to gp3tools-style eye-tracking data

Description

Compatibility wrapper for `join_gazepoint_biometrics_to_master()`. This alias is provided for users who work with a gp3tools master table and want an explicit gp3tools-facing function name. The implementation delegates to the canonical biometric-to-master join helper.

Usage

```
join_gazepoint_biometrics_to_gp3tools(biometrics, gp3tools_master, ...)
```

Arguments

biometrics	A data frame containing Gazepoint Biometrics samples or summaries.
gp3tools_master	A gp3tools-style master eye-tracking data frame.
...	Additional arguments passed to join_gazepoint_biometrics_to_master(), including the required by argument when the underlying join helper requires explicit join columns.

Value

The output of join_gazepoint_biometrics_to_master().

Examples

```
biometrics <- data.frame(USER = rep("P1", 3), CNT = 1:3, HR = c(70, 71, 72))
master <- data.frame(USER = rep("P1", 3), CNT = 1:3, AOI = c("A", "B", "A"))
join_gazepoint_biometrics_to_gp3tools(
  biometrics,
  master,
  by = c("USER", "CNT")
)
```

---

join_gazepoint_biometrics_to_master

Join Gazepoint Biometrics to a master table

Description

Convenience wrapper around sync_gazepoint_biometrics_with_gaze() for joining biometric data to a gp3tools-style master table or any other analysis-ready gaze table.

Usage

```
join_gazepoint_biometrics_to_master(master, biometrics, by, all_x = TRUE)
```

Arguments

master	A master gaze or analysis table.
biometrics	A Gazepoint Biometrics data frame.
by	Character vector of key columns used for joining.
all_x	Logical. Should all rows from master be retained?

Value

A data frame with biometric columns joined to the master table.

`model_gazepoint_eda_point_process`
Model EDA events as a dependency-light point process

Description

Creates event-time, inter-event interval, and inverse-Gaussian-style summary tables for EDA/SCR events. Events can be supplied directly through an event column or derived from positive EDA-derivative bursts.

Usage

```
model_gazepoint_eda_point_process(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  group_cols = NULL,
  event_time_col = NULL,
  event_indicator_col = NULL,
  derivative_mad_multiplier = 6,
  min_event_distance_s = 1
)
```

Arguments

`dat`	A data frame.
`eda_col`	Numeric EDA column.
`time_col`	Numeric time column.
`group_cols`	Optional grouping columns.
`event_time_col`	Optional column of event onset times.
`event_indicator_col`	Optional binary event indicator column.
`derivative_mad_multiplier`	MAD multiplier for derivative-derived events.
`min_event_distance_s`	Minimum distance between derived events.

Details

This function is a compact point-process summary/model-preparation helper. It does not fit a full latent sympathetic state-space model.

Value

A list with `overview`, `event_table`, `interval_table`, `process_summary`, and `settings`.

`model_gazepoint_hrv_ipfm` *Model heartbeat timing using an IPFM-style impulse train*

Description

Builds an impulse-train representation of heartbeat timing from IBI/RR intervals or beat times and computes a simple spectrum of the resulting impulse train. This is an IPFM-style model-preparation helper, not a perfect reconstruction of sinoatrial-node physiology.

Usage

```
model_gazepoint_hrv_ipfm(
  dat,
  ibi_col = "IBI",
  beat_time_col = NULL,
  group_cols = NULL,
  ibi_units = c("auto", "seconds", "milliseconds"),
  output_sampling_rate = 4,
  max_frequency = 0.5
)
```

Arguments

`dat`	A data frame.
`ibi_col`	Optional numeric IBI/RR interval column.
`beat_time_col`	Optional explicit beat-time column.
`group_cols`	Optional grouping columns.
`ibi_units`	`"auto"`, `"seconds"`, or `"milliseconds"`.
`output_sampling_rate`	Sampling rate for regular impulse train in Hz.
`max_frequency`	Maximum frequency returned in spectrum.

Value

A list with `overview`, `beat_table`, `impulse_table`, `spectrum_table`, `summary`, and `settings`.

model_gazepoint_hr_point_process

Model heartbeats as a dependency-light point process

Description

Creates beat-time, interbeat interval, and inverse-Gaussian-style summary tables from IBI/RR intervals. This is a compact point-process model-preparation helper, not a full adaptive Bayesian heartbeat filter.

Usage

```
model_gazepoint_hr_point_process(
  dat,
  ibi_col = "IBI",
  time_col = NULL,
  beat_time_col = NULL,
  group_cols = NULL,
  ibi_units = c("auto", "seconds", "milliseconds")
)
```

Arguments

dat	A data frame.
ibi_col	Numeric IBI/RR interval column.
time_col	Optional time column.
beat_time_col	Optional explicit beat-time column.
group_cols	Optional grouping columns.
ibi_units	"auto", "seconds", or "milliseconds".

Value

A list with overview, beat_table, interval_table, process_summary, and settings.

optimize_gazepoint_cvxeda_tau

Optimise subject-specific cvxEDA slow time constant

Description

Performs a dependency-light grid search over the slow Bateman impulse-response time constant used in cvxEDA-style EDA decomposition workflows. The default fast time constant is fixed at 0.7 seconds and the slow time constant is searched between 2 and 4 seconds. This function does not run the original cvxEDA optimisation; it provides a subject-specific tau-selection bridge for downstream cvxEDA-style workflows.

Usage

```
optimize_gazepoint_cvxeda_tau(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  group_cols = NULL,
  tau0_grid = seq(2, 4, by = 0.25),
  tau1 = 0.7,
  sampling_rate = NULL,
  ridge_lambda = 0.01,
  max_irf_seconds = 20
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	Numeric EDA/conductance column.
time_col	Numeric time column.
group_cols	Optional grouping columns, usually participant/session.
tau0_grid	Candidate slow time constants.
tau1	Fixed fast time constant.
sampling_rate	Optional sampling rate in Hz. If NULL, estimated from time_col.
ridge_lambda	Small ridge penalty used in frequency-domain deconvolution.
max_irf_seconds	Maximum impulse-response duration.

Value

A list with overview, best_tau, optimization_table, and settings.

---

plot_gazepoint_aoi_biometrics *Plot AOI-linked biometric summaries*

Description

Plots AOI-biometric summary values as a ggplot object.

Usage

```
plot_gazepoint_aoi_biometrics(
  x,
  value_col = "mean_value",
  aoi_col = "aoi_label",
```

```
  signal_col = "signal",
  group_col = NULL,
  plot_type = c("boxplot", "point", "line"),
  title = NULL
)
```

Arguments

x	A gazepoint_aoi_biometrics_summary, gazepoint_aoi_biometrics_model_data, or data frame.
value_col	Value column to plot.
aoi_col	AOI label column.
signal_col	Signal label column.
group_col	Optional grouping column.
plot_type	"boxplot", "point", or "line".
title	Optional plot title.

Value

A ggplot object with plot data stored in attributes.

---

plot_gazepoint_biometric_quality

Plot Gazepoint biometric quality indicators

Description

Plots and summarises biometric quality indicators such as dropout flags, validity flags, missingness flags, and quality/audit flags. When no explicit quality columns are available, the function can derive missingness indicators from detected biometric signal columns.

Usage

```
plot_gazepoint_biometric_quality(
  data,
  quality_cols = NULL,
  signal_cols = NULL,
  time_col = NULL,
  group_col = NULL,
  dropout_prefix = "biometric_dropout",
  max_points = 5000L,
  main = NULL,
  plot = TRUE,
  ...
)
```

Arguments

data	A data frame.
quality_cols	Optional quality/flag columns. If NULL, likely quality columns are detected from names and types.
signal_cols	Optional signal columns used to derive missingness flags when no quality columns are detected.
time_col	Optional time/order column recorded in the returned settings.
group_col	Optional grouping column for group-level quality summaries.
dropout_prefix	Prefix used by dropout columns created by flag_gazepoint_biometric_dropouts().
max_points	Maximum number of rows used for row-level returned plot data.
main	Optional plot title.
plot	Logical. If FALSE, no plot is drawn.
...	Additional arguments passed to barplot().

Value

A list with overview, quality_summary, group_summary, plot_data, and settings.

Examples

```
df <- data.frame(
  CNT = 1:5,
  GSR = c(1, NA, 1.2, 1.1, NA),
  HR_valid = c(1, 1, 0, 1, 1)
)
plot_gazepoint_biometric_quality(df, signal_cols = "GSR", plot = FALSE)
```

---

plot_gazepoint_biometric_report_dashboard

Create a lightweight Gazepoint biometric QC plot dashboard

Description

Creates a lightweight dashboard object containing QC plots for signal activity and time-reset diagnostics. This is a structured list of ggplot objects, not a Shiny application. The dashboard is intended for report preparation and manual QC review.

Usage

```
plot_gazepoint_biometric_report_dashboard(
  data = NULL,
  signal_activity = NULL,
  time_resets = NULL,
  signal_cols = NULL,
  group_cols = NULL,
  time_col = NULL,
  include_signal_activity = TRUE,
  include_time_resets = TRUE,
  max_groups = 30,
  continue_on_error = TRUE,
  title_prefix = "Gazepoint biometric QC"
)
```

Arguments

data	Optional Gazepoint biometric data frame.
signal_activity	Optional result from audit_gazepoint_signal_activity().
time_resets	Optional result from audit_gazepoint_time_resets().
signal_cols	Optional signal columns used when computing signal activity from data.
group_cols	Optional grouping columns.
time_col	Optional time or counter column.
include_signal_activity	If TRUE, include a signal-activity plot.
include_time_resets	If TRUE, include a time-reset plot.
max_groups	Maximum number of groups to display in each plot.
continue_on_error	If TRUE, plot failures are recorded in errors rather than stopping the dashboard.
title_prefix	Optional title prefix.

Value

A list with overview, plots, errors, inputs, and settings.

---

plot_gazepoint_biometric_signals

Plot Gazepoint biometric signal time series

Description

Plots one or more Gazepoint biometric signals using base R graphics and returns the plotted data and signal summary. The helper is intentionally descriptive and does not infer emotional valence, cognition, or HRV from raw biometric columns.

Usage

```
plot_gazepoint_biometric_signals(
  data,
  signal_cols = NULL,
  time_col = NULL,
  group_col = NULL,
  max_points = 5000L,
  standardize = FALSE,
  type = c("line", "points", "both"),
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  legend = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

data	A data frame.
signal_cols	Optional character vector of signal columns. If NULL, common Gazepoint biometric signal columns are detected.
time_col	Optional time/order column for the x-axis. If NULL, row number is used.
group_col	Optional grouping column recorded in the returned overview. The current plotting implementation overlays the selected rows rather than faceting by group.
max_points	Maximum number of rows to plot. Large data are evenly downsampled for display only; returned summaries still describe the input signal columns.
standardize	Logical. Should each signal be z-standardised before plotting? This is useful when signals are on different scales.
type	Plot type: "line", "points", or "both".
main	Optional plot title.
xlab	Optional x-axis label.
ylab	Optional y-axis label.
legend	Logical. Should a legend be drawn when more than one signal is plotted?
plot	Logical. If FALSE, no plot is drawn and only the plot object is returned.
...	Additional arguments passed to matplot().

Value

A list with overview, plot_data, signal_summary, and settings.

Examples

```
df <- data.frame(
  CNT = 1:5,
  GSR = c(1, 1.1, 1.2, 1.1, 1),
  HR = c(70, 71, 72, 71, 70)
)
plot_gazepoint_biometric_signals(df, time_col = "CNT", plot = FALSE)
```

---

plot_gazepoint_eda_decomposition

Plot Gazepoint EDA decomposition channels

Description

Plots available EDA/GSR decomposition channels, typically `GSR_US`, `GSR_US_TONIC`, and `GSR_US_PHASIC`, as a ggplot object.

Usage

```
plot_gazepoint_eda_decomposition(
  data,
  time_col = NULL,
  signal_cols = NULL,
  group_cols = NULL,
  standardise = FALSE,
  max_points = 5000,
  title = NULL
)
```

Arguments

`data`	A Gazepoint biometric data frame or list containing a data frame.
`time_col`	Optional time/counter column.
`signal_cols`	Optional signal columns to plot.
`group_cols`	Optional grouping columns used for facets.
`standardise`	Logical. If `TRUE`, standardise each signal to z-scores.
`max_points`	Maximum number of rows retained after simple downsampling.
`title`	Optional plot title.

Value

A ggplot object with plot data stored in attributes.

plot_gazepoint_eda_gram *Plot an EDA-gram-style time-frequency representation*

Description

Creates a dependency-light EDA-gram-style representation using sliding-window spectral power. This is inspired by EDA-gram visualisations, but it does not implement a full sparse dictionary decomposition unless such a model is supplied externally.

Usage

```
plot_gazepoint_eda_gram(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  group_cols = NULL,
  group_id_to_plot = NULL,
  sampling_rate = NULL,
  window_seconds = 30,
  step_seconds = 5,
  frequency_range = c(0.01, 0.5),
  frequency_bins = 64,
  log_power = TRUE,
  plot = TRUE,
  main = "EDA-gram"
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	Numeric EDA/conductance column.
time_col	Numeric time column.
group_cols	Optional grouping columns.
group_id_to_plot	Optional group ID to plot. If NULL, plots the first available group.
sampling_rate	Optional sampling rate in Hz. If NULL, estimated from time_col.
window_seconds	Sliding-window length.
step_seconds	Sliding-window step.
frequency_range	Frequency range shown in Hz.
frequency_bins	Number of frequency bins.
log_power	Logical. If TRUE, plot log1p power.
plot	Logical. If TRUE, draw the EDA-gram.
main	Plot title.

Value

Invisibly returns a list with `overview`, `gram_table`, `plot_matrix`, and `settings`.

`plot_gazepoint_multimodal_timeline` *Plot multimodal Gazepoint biometric timelines*

Description

Creates a conservative timeline plot for one or more biometric channels. The plot is intended for inspection, synchronization checks, and reporting support. It does not interpret electrodermal activity as emotional valence.

Usage

```
plot_gazepoint_multimodal_timeline(
  data,
  time_col = NULL,
  signal_cols = NULL,
  group_cols = NULL,
  participant_col = NULL,
  stimulus_col = NULL,
  trial_col = NULL,
  event_time_col = NULL,
  event_col = NULL,
  standardise = TRUE,
  show_event_markers = TRUE,
  title = NULL
)
```

Arguments

`data`	A data frame containing biometric samples or aligned biometric rows.
`time_col`	Optional time column. If `NULL`, common time columns are detected automatically.
`signal_cols`	Optional biometric signal columns. If `NULL`, common Gazepoint biometric columns are detected automatically.
`group_cols`	Optional grouping columns used to separate trajectories.
`participant_col`, `stimulus_col`, `trial_col`	Optional common grouping columns to add to `group_cols`.
`event_time_col`	Optional column containing event times for vertical markers.
`event_col`	Optional event/TTL indicator column used for vertical markers.
`standardise`	Logical. If `TRUE`, signals are z-scored within channel for visual comparison.
`show_event_markers`	Logical. Should event markers be drawn when available?
`title`	Optional plot title.

Value

A ggplot object with the long plotting data stored in the `plot_data` attribute and settings stored in the `settings` attribute.

`plot_gazepoint_saccade_main_sequence`
Plot Gazepoint saccade main-sequence diagnostics

Description

Plots saccade amplitude against peak velocity. The function expects saccade-level amplitude and peak-velocity columns. If raw sample-level data are supplied, users should first derive saccade-level kinematics with a validated fixation/saccade detector.

Usage

```
plot_gazepoint_saccade_main_sequence(
  dat,
  amplitude_col = NULL,
  peak_velocity_col = NULL,
  group_col = NULL,
  log_axes = TRUE,
  add_smoother = TRUE,
  main = "Gazepoint saccade main-sequence diagnostic"
)
```

Arguments

`dat`	A saccade-level data frame.
`amplitude_col`	Saccade amplitude column.
`peak_velocity_col`	Peak velocity column.
`group_col`	Optional grouping column.
`log_axes`	Logical. If `TRUE`, use log10 axes.
`add_smoother`	Logical. If `TRUE`, add a lowess curve.
`main`	Plot title.

Value

Invisibly returns the plotted data and settings.

plot_gazepoint_scr_events *Plot Gazepoint SCR events on an EDA signal*

Description

Plots an EDA/GSR signal with detected SCR peak markers and optional event onsets from SCR event-window summaries or event tables.

Usage

```
plot_gazepoint_scr_events(
  data,
  scr_peaks,
  event_windows = NULL,
  events = NULL,
  time_col = NULL,
  signal_col = NULL,
  phasic_col = NULL,
  group_cols = NULL,
  show_events = TRUE,
  max_points = 5000,
  title = NULL
)
```

Arguments

data	Gazepoint biometric data frame.
scr_peaks	A gazepoint_scr_peak_detection object or peak data frame.
event_windows	Optional gazepoint_scr_event_window_summary object or event-window data frame.
events	Optional event table used when event_windows is not supplied.
time_col	Optional time/counter column.
signal_col	Optional signal column to plot.
phasic_col	Optional preferred phasic signal column.
group_cols	Optional grouping columns used for facets and matching.
show_events	Logical. If TRUE, show event onsets when available.
max_points	Maximum number of signal rows retained after downsampling.
title	Optional plot title.

Value

A ggplot object with plot data stored in attributes.

plot_gazepoint_scr_specification_curve

Plot an SCR specification curve

Description

Plots a specification-curve style display from the output of `run_gazepoint_scr_multiverse()` or from a compatible data frame.

Usage

```
plot_gazepoint_scr_specification_curve(
  x,
  estimate_col = NULL,
  specification_col = "specification_id",
  add_zero_line = TRUE,
  main = "SCR specification curve"
)
```

Arguments

`x`	Output from `run_gazepoint_scr_multiverse()` or a data frame.
`estimate_col`	Column to rank and plot. Defaults to `"mean_response_amplitude"` when available, otherwise `"response_rate"`.
`specification_col`	Specification identifier column.
`add_zero_line`	Logical. If `TRUE`, draw a horizontal zero line.
`main`	Plot title.

Value

Invisibly returns a list with plot data and settings.

plot_gazepoint_signal_activity

Plot Gazepoint biometric signal activity

Description

Plots signal-activity summaries produced by `audit_gazepoint_signal_activity()`, or computes them from a biometric data frame. The plot is intended for quality-control review of signal availability, missingness, zero activity, and basic activity status. It does not infer emotion, valence, cognition, preference, trust, or physiological diagnosis.

Usage

```
plot_gazepoint_signal_activity(
  data,
  signal_cols = NULL,
  group_cols = NULL,
  metric = c("active_signal", "nonzero_prop", "missing_prop", "n_unique_finite"),
  max_groups = 30,
  title = NULL
)
```

Arguments

data	A Gazepoint biometric data frame, or an audit_gazepoint_signal_activity() result.
signal_cols	Optional signal columns used when data is a raw data frame.
group_cols	Optional grouping columns used when data is a raw data frame.
metric	Summary metric to plot.
max_groups	Maximum number of groups to display.
title	Optional plot title.

Value

A ggplot object with the package plot contract attached.

---

plot_gazepoint_time_resets

Plot Gazepoint time resets and time-order flags

Description

Plots row-level time/counter progression and flags from audit_gazepoint_time_resets(), or computes them from a biometric data frame. The plot is intended for synchronization and file-structure QC. It does not establish causal timing or true physiological latency.

Usage

```
plot_gazepoint_time_resets(
  data,
  time_col = NULL,
  group_cols = NULL,
  max_groups = 30,
  title = NULL
)
```

Arguments

data	A Gazepoint biometric data frame, or an audit_gazepoint_time_resets() result.
time_col	Optional time or counter column used when data is a raw data frame.
group_cols	Optional grouping columns used when data is a raw data frame.
max_groups	Maximum number of groups to display.
title	Optional plot title.

Value

A ggplot object with the package plot contract attached.

---

prepare_gazepoint_aoi_biometrics_model_data

Prepare AOI-biometric model data

Description

Converts AOI-biometric summaries into a modelling-ready table for GLM/LMM/GLMM workflows.

Usage

```
prepare_gazepoint_aoi_biometrics_model_data(
  x,
  outcome_col = "mean_value",
  predictor_cols = c("aoi_label", "signal"),
  factor_cols = c("aoi_label", "signal"),
  numeric_cols = NULL,
  group_cols = NULL,
  drop_missing_outcome = TRUE,
  min_rows = NULL,
  standardise_outcome = FALSE,
  standardise_within = c("signal", "all")
)
```

Arguments

x	A gazepoint_aoi_biometrics_summary object or summary data frame.
outcome_col	Outcome column to model.
predictor_cols	Optional predictor columns to retain.
factor_cols	Optional columns converted to factors.
numeric_cols	Optional columns converted to numeric.
group_cols	Optional grouping columns for random-effect formulas.

`drop_missing_outcome`
: Logical. If `TRUE`, rows with missing outcomes are removed.

`min_rows`
: Optional minimum contributing rows required.

`standardise_outcome`
: Logical. If `TRUE`, add a z-scored outcome column.

`standardise_within`
: Standardization scope used when `standardise_outcome = TRUE`. Use `"signal"` to z-score within each biometric signal or `"all"` to z-score across all rows.

Value

A list with `overview`, `model_data`, `variable_summary`, `model_formulas`, and `settings`.

---

prepare_gazepoint_artifact_svm_features

Prepare EDA artifact-classifier segment features

Description

Creates segment-level features that can be passed to a user-supplied artifact classifier such as an SVM. No pretrained classifier is bundled.

Usage

```
prepare_gazepoint_artifact_svm_features(
  dat,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  segment_seconds = 5,
  samples_per_segment = NULL,
  sampling_rate = NULL
)
```

Arguments

`dat`
: A data frame.

`eda_col`
: EDA/conductance column.

`time_col`
: Optional time column.

`group_cols`
: Optional grouping columns.

`segment_seconds`
: Segment length in seconds when `time_col` is supplied.

`samples_per_segment`
: Segment length in samples when no usable time column or sampling rate is available.

`sampling_rate`
: Optional sampling rate in Hz.

Value

A segment-level feature data frame.

`prepare_gazepoint_biometrics_lme_data` *Prepare Gazepoint biometric summaries for mixed-model analysis*

Description

Prepares biometric window-level or event-level summaries for downstream mixed-model analysis. This helper does not fit a model. It checks variables, optionally baseline-corrects the selected outcome, optionally scales numeric predictors, converts grouping/factor variables, flags complete cases, and returns a conservative model formula.

Usage

```
prepare_gazepoint_biometrics_lme_data(
  data,
  outcome_col,
  fixed_effect_cols = NULL,
  condition_cols = NULL,
  covariate_cols = NULL,
  random_effect_cols = NULL,
  participant_col = NULL,
  stimulus_col = NULL,
  trial_col = NULL,
  window_col = NULL,
  baseline_col = NULL,
  baseline_correct = FALSE,
  factor_cols = NULL,
  continuous_cols = NULL,
  scale_continuous = FALSE,
  include_window = TRUE,
  drop_missing = TRUE,
  min_rows = 10
)
```

Arguments

`data`	A data frame containing biometric summary rows.
`outcome_col`	Name of the outcome column to analyse.
`fixed_effect_cols`	Optional fixed-effect predictor columns.
`condition_cols`	Optional condition/design columns to include as fixed effects.
`covariate_cols`	Optional covariate columns to include as fixed effects.

`random_effect_cols`
: Optional grouping columns for random intercepts.

`participant_col`, `stimulus_col`, `trial_col`
: Optional common grouping columns.

`window_col`
: Optional analysis-window column. Included as a fixed effect when `include_window = TRUE`.

`baseline_col`
: Optional baseline column.

`baseline_correct`
: Logical. If `TRUE`, creates an outcome column equal to `outcome_col - baseline_col`.

`factor_cols`
: Optional columns to convert to factors.

`continuous_cols`
: Optional numeric predictor columns to scale when `scale_continuous = TRUE`.

`scale_continuous`
: Logical. If `TRUE`, creates z-scored versions of numeric continuous predictors and uses those in the formula.

`include_window`
: Logical. Should `window_col` be included as a fixed effect?

`drop_missing`
: Logical. Should incomplete model rows be removed from `model_data`?

`min_rows`
: Minimum number of complete rows required for a `"ready"` status.

Value

A list with `overview`, `data`, `model_data`, `model_formula`, `variable_summary`, and `settings`.

---

prepare_gazepoint_ctsi_input

Prepare Gazepoint EDA data for CTSI sparse deconvolution workflows

---

Description

Prepares signal, event, and configuration tables for downstream continuous-time system identification (CTSI) sparse EDA deconvolution workflows. This function does not implement the full Amin-Faghih CTSI solver in R. It creates reproducible input objects and optional CSV files for external CTSI implementations.

Usage

```
prepare_gazepoint_ctsi_input(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  group_cols = NULL,
  event_onset_col = NULL,
  event_name_col = NULL,
  sampling_rate = NULL,
```

```
  tau0_range = c(2, 4),
  tau1_range = c(0.5, 1),
  sparsity_grid = c(0.001, 0.01, 0.1, 1),
  output_dir = NULL,
  prefix = "gazepoint_ctsi"
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	Numeric EDA/conductance column.
time_col	Numeric time column.
group_cols	Optional grouping columns, such as participant/session.
event_onset_col	Optional event onset column.
event_name_col	Optional event/condition column.
sampling_rate	Optional sampling rate in Hz.
tau0_range	Candidate slow time-constant range.
tau1_range	Candidate fast time-constant range.
sparsity_grid	Candidate sparsity penalties.
output_dir	Optional directory for CSV export.
prefix	Output file prefix.

Value

A list with overview, signal_table, event_table, ctsi_config, ctsi_notes, written_files, and settings.

---

prepare_gazepoint_cvxeda_input

Prepare Gazepoint EDA input for external cvxEDA-style workflows

---

Description

Prepares a clean Gazepoint EDA/conductance time-series table that can be exported for external cvxEDA-style workflows. This function does not run a native cvxEDA solver and does not attempt to reproduce cvxEDA internally.

Usage

```
prepare_gazepoint_cvxeda_input(
  data,
  eda_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  time_unit = c("auto", "ms", "seconds", "samples"),
  convert_resistance_to_us = FALSE,
  min_finite_prop = 0.5,
  output_dir = NULL,
  prefix = "gazepoint_cvxeda"
)
```

Arguments

data	A Gazepoint biometric data frame, or a list containing one.
eda_col	Optional EDA/conductance column. If omitted, the function prefers GSR_US when available.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz, used when the time column is a sample counter.
time_unit	Unit of time_col.
convert_resistance_to_us	If TRUE, convert a selected resistance-like GSR column to microsiemens as 1,000,000 / GSR. The default is FALSE because this conversion should be used only when the user has verified that GSR is resistance-like and GSR_US is unavailable.
min_finite_prop	Minimum finite proportion required for a group to be labelled ready.
output_dir	Optional folder where CSV files should be written.
prefix	File prefix used when output_dir is supplied.

Value

A list with overview, signal_table, group_summary, manifest, and settings.

---

prepare_gazepoint_ledalab_input

Prepare Gazepoint EDA input for external Ledalab-style workflows

---

Description

Prepares a clean Gazepoint EDA/conductance time-series table that can be exported for external Ledalab-style workflows. This function does not run Ledalab and does not attempt to reproduce Ledalab internally.

Usage

```
prepare_gazepoint_ledalab_input(
  data,
  eda_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  time_unit = c("auto", "ms", "seconds", "samples"),
  convert_resistance_to_us = FALSE,
  min_finite_prop = 0.5,
  output_dir = NULL,
  prefix = "gazepoint_ledalab"
)
```

Arguments

data	A Gazepoint biometric data frame, or a list containing one.
eda_col	Optional EDA/conductance column. If omitted, the function prefers GSR_US when available.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz, used when the time column is a sample counter.
time_unit	Unit of time_col.
convert_resistance_to_us	If TRUE, convert a selected resistance-like GSR column to microsiemens as 1,000,000 / GSR. The default is FALSE because this conversion should be used only when the user has verified that GSR is resistance-like and GSR_US is unavailable.
min_finite_prop	Minimum finite proportion required for a group to be labelled ready.
output_dir	Optional folder where CSV files should be written.
prefix	File prefix used when output_dir is supplied.

Value

A list with overview, signal_table, group_summary, manifest, and settings.

---

prepare_gazepoint_multimodal_model_data

Prepare Gazepoint multimodal model data

Description

Creates a model-ready table from Gazepoint biometric window summaries and, optionally, eye-tracking summaries produced by gp3tools or another workflow. The function is intentionally conservative: it does not fit a model, impute missing values, or remove rows automatically.

Usage

```
prepare_gazepoint_multimodal_model_data(
  biometrics,
  eye_tracking = NULL,
  group_columns = NULL,
  biometric_is_summarised = FALSE,
  by = NULL,
  all = FALSE
)
```

Arguments

biometrics	A data frame containing row-level Gazepoint Biometrics data or an already summarised biometric window table.
eye_tracking	Optional eye-tracking summary table to merge with the biometric summaries.
group_columns	Columns defining the analysis unit, such as c("USER", "MEDIA_ID").
biometric_is_summarised	Logical. If FALSE, biometric window summaries are created using summarise_gazepoint_multimodal_v If TRUE, biometrics is treated as already summarised.
by	Optional merge keys. If NULL, group_columns are used.
all	Should a full outer join be used when eye-tracking data are supplied? Defaults to FALSE, giving an inner join.

Value

A data frame with class "gazepoint_multimodal_model_data" and a "model_data_summary" attribute.

---

prepare_gazepoint_neurokit_eda_input

Prepare Gazepoint EDA input for NeuroKit2-style workflows

Description

Prepares EDA/GSR signal tables for optional external NeuroKit2 processing. This helper does not require Python or NeuroKit2.

Usage

```
prepare_gazepoint_neurokit_eda_input(
  data,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
```

```
  output_dir = NULL,
  prefix = "gazepoint_neurokit_eda"
)
```

Arguments

data	A Gazepoint biometric data frame.
eda_col	EDA/GSR signal column.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz.
output_dir	Optional directory where per-group CSV files are written.
prefix	File prefix used when output_dir is supplied.

Value

A list with overview, eda_table, group_summary, manifest, and settings.

---

prepare_gazepoint_pspm_dcm_input

Prepare Gazepoint EDA data for PsPM DCM workflows

Description

Prepares Gazepoint EDA data and event metadata for downstream PsPM dynamic causal modelling workflows. This function does not run PsPM or invert a DCM model in R. It creates structured input tables and notes for MATLAB/PsPM.

Usage

```
prepare_gazepoint_pspm_dcm_input(
  dat,
  eda_col = "GSR_US",
  time_col = "CNT",
  event_onset_col = NULL,
  event_duration_col = NULL,
  event_name_col = NULL,
  participant_col = NULL,
  session_col = NULL,
  sampling_rate = NULL,
  output_dir = NULL,
  prefix = "gazepoint_pspm_dcm"
)
```

Arguments

dat	A data frame containing EDA data.
eda_col	Numeric EDA/conductance column.
time_col	Numeric time column.
event_onset_col	Optional event onset column.
event_duration_col	Optional event duration column.
event_name_col	Optional event name/condition column.
participant_col	Optional participant column.
session_col	Optional session column.
sampling_rate	Optional sampling rate in Hz.
output_dir	Optional directory for CSV export.
prefix	File prefix when output_dir is supplied.

Value

A list with overview, signal_table, event_table, pspm_notes, written_files, and settings.

---

prepare_gazepoint_pspm_input

Prepare Gazepoint EDA input for external PsPM-style workflows

---

Description

Prepares a clean Gazepoint EDA/conductance time-series table that can be exported for external PsPM-style workflows. This function does not run PsPM and does not attempt to reproduce PsPM internally.

Usage

```
prepare_gazepoint_pspm_input(
  data,
  eda_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  time_unit = c("auto", "ms", "seconds", "samples"),
  convert_resistance_to_us = FALSE,
  min_finite_prop = 0.5,
  output_dir = NULL,
  prefix = "gazepoint_pspm"
)
```

Arguments

data	A Gazepoint biometric data frame, or a list containing one.
eda_col	Optional EDA/conductance column. If omitted, the function prefers GSR_US when available.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz, used when the time column is a sample counter.
time_unit	Unit of time_col.
convert_resistance_to_us	If TRUE, convert a selected resistance-like GSR column to microsiemens as 1,000,000 / GSR. The default is FALSE because this conversion should be used only when the user has verified that GSR is resistance-like and GSR_US is unavailable.
min_finite_prop	Minimum finite proportion required for a group to be labelled ready.
output_dir	Optional folder where CSV files should be written.
prefix	File prefix used when output_dir is supplied.

Value

A list with overview, signal_table, group_summary, manifest, and settings.

---

prepare_gazepoint_pyppg_input

Prepare Gazepoint HRP/PPG waveform input for pyPPG

Description

Prepares a Gazepoint heart-rate pulse waveform column, usually HRP, as a lightweight input table for optional pyPPG workflows. This helper does not call Python, does not require pyPPG, and does not derive HRV features. It only prepares waveform values, timing information when available, and conservative group-level summaries for interoperability review.

Usage

```
prepare_gazepoint_pyppg_input(
  data,
  ppg_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  time_unit = c("auto", "ms", "seconds", "samples"),
  min_finite_prop = 0.5,
  output_dir = NULL,
  prefix = "gazepoint_pyppg"
)
```

Arguments

data	A Gazepoint biometric data frame or a list containing one.
ppg_col	Optional HRP/PPG waveform column. If NULL, common Gazepoint HRP/PPG column names are detected.
time_col	Optional time, timestamp, or sample-counter column.
group_cols	Optional grouping columns.
sampling_rate	Optional sampling rate in Hz. Useful when time_col is a sample counter such as CNT.
time_unit	Unit of time_col: "auto", "ms", "seconds", or "samples".
min_finite_prop	Minimum finite waveform proportion used for group readiness summaries.
output_dir	Optional directory for CSV export. If NULL, no files are written.
prefix	File prefix used when output_dir is supplied.

Value

A list with overview, waveform_table, group_summary, manifest, and settings.

---

prepare_gazepoint_rhrv_input

Prepare Gazepoint IBI/RR data for RHRV

Description

Backward-compatible alias for export_gazepoint_rhrv_input(). This helper prepares analysis-ready inter-beat interval input for optional RHRV workflows. It does not call RHRV and does not derive HRV from Gazepoint's raw HRV field. HRV features should be derived from genuine IBI/RR intervals.

Usage

```
prepare_gazepoint_rhrv_input(
  data,
  ibi_col = "IBI_clean_ms",
  group_cols = NULL,
  unit = c("auto", "ms", "seconds"),
  collapse_repeated_intervals = TRUE,
  repeated_tolerance_ms = 1e-08,
  min_ibi_ms = 300,
  max_ibi_ms = 2000,
  output_dir = NULL,
  prefix = "gazepoint_rhrv"
)
```

Arguments

data	A Gazepoint biometric data frame or gazepoint_ibi_filter object.
ibi_col	IBI/RR interval column.
group_cols	Optional grouping columns.
unit	Unit of the IBI column: "auto", "ms", or "seconds".
collapse_repeated_intervals	Logical. If TRUE, consecutive repeated IBI values are collapsed before export.
repeated_tolerance_ms	Numeric tolerance used when identifying repeated consecutive IBI values.
min_ibi_ms	Minimum plausible IBI in milliseconds retained for export.
max_ibi_ms	Maximum plausible IBI in milliseconds retained for export.
output_dir	Optional directory where per-group CSV files are written.
prefix	File prefix used when output_dir is supplied.

Value

A list returned by export_gazepoint_rhrv_input().

---

prepare_gazepoint_scr_hurdle_model_data
Prepare Gazepoint SCR hurdle-model data

Description

Converts SCR event-window summaries into model-ready data for two-part electrodermal-response analyses: a binary response/no-response component and a positive-amplitude component among response events.

Usage

```
prepare_gazepoint_scr_hurdle_model_data(
  scr_event_windows,
  response_col = "response_flag",
  amplitude_col = "scr_amplitude",
  latency_col = "scr_latency",
  rise_time_col = "scr_rise_time",
  recovery_time_col = "scr_recovery_time",
  predictor_cols = NULL,
  factor_cols = NULL,
  numeric_cols = NULL,
  group_cols = NULL,
  event_id_col = "event_id",
  amplitude_transform = c("none", "log", "log1p"),
  amplitude_offset = 1e-06,
  drop_missing_predictors = TRUE
)
```

Arguments

scr_event_windows	A gazepoint_scr_event_window_summary object returned by summarise_gazepoint_scr_event_win or a data frame containing event-window SCR summaries.
response_col	Column containing the binary SCR response flag.
amplitude_col	Column containing SCR amplitude.
latency_col	Optional column containing SCR latency.
rise_time_col	Optional column containing SCR rise time.
recovery_time_col	Optional column containing SCR recovery time.
predictor_cols	Optional fixed-effect predictor columns to include in generated formulas and complete-case checks.
factor_cols	Optional columns to coerce to factors.
numeric_cols	Optional columns to coerce to numeric.
group_cols	Optional grouping columns retained for random effects or clustered summaries.
event_id_col	Optional event identifier column.
amplitude_transform	Transformation for the positive-amplitude outcome: "none", "log", or "log1p".
amplitude_offset	Small positive offset used when amplitude_transform = "log".
drop_missing_predictors	Logical. If TRUE, model datasets are restricted to rows complete on outcome and predictor columns.

Value

A list with overview, response_model_data, amplitude_model_data, variable_summary, model_formulas, and settings.

---

recommend_gazepoint_biometric_exclusions

Recommend Gazepoint biometric exclusions

Description

Creates window-level and participant-level exclusion recommendations from Gazepoint biometric usable-sample summaries. The function does not remove data. It only creates transparent keep, review, or exclude recommendations.

Usage

```
recommend_gazepoint_biometric_exclusions(
  data,
  group_columns = NULL,
  data_is_window_summary = FALSE,
  participant_column = NULL,
  gsr_min_usable_pct = 50,
  hr_min_usable_pct = 50,
  dial_min_usable_pct = 50,
  require_gsr = TRUE,
  require_hr = TRUE,
  require_dial = FALSE
)
```

Arguments

data	A row-level Gazepoint Biometrics data frame, a path to a Gazepoint CSV export, or an already summarised multimodal window table.
group_columns	Columns defining analysis windows when data is row-level, such as c("source_participant", "MEDIA_ID").
data_is_window_summary	Logical. If TRUE, data is treated as an already summarised window table.
participant_column	Optional participant identifier column. If NULL, the function tries source_participant, USER, and USERID.
gsr_min_usable_pct	Minimum acceptable usable percentage for GSR/EDA.
hr_min_usable_pct	Minimum acceptable usable percentage for heart rate.
dial_min_usable_pct	Minimum acceptable usable percentage for engagement dial.
require_gsr	Should low GSR/EDA coverage cause exclusion rather than review?
require_hr	Should low heart-rate coverage cause exclusion rather than review?
require_dial	Should low engagement-dial coverage cause exclusion rather than review?

Value

A list with overview, window_recommendations, participant_recommendations, and settings.

`regress_gazepoint_pupil_luminance` *Regress stimulus luminance from pupil diameter*

Description

Regresses continuous pupil diameter on frame-wise or sample-wise stimulus luminance and returns a luminance-adjusted pupil series. This controls a major visual confound but does not prove that residual pupil changes are cognitive-load-only effects.

Usage

```
regress_gazepoint_pupil_luminance(
  dat,
  pupil_col,
  luminance_col,
  group_cols = NULL,
  time_col = NULL,
  output_col = "pupil_luminance_adjusted",
  fitted_col = "pupil_luminance_fitted",
  include_quadratic = TRUE,
  model_by_group = TRUE,
  add_intercept_mean = TRUE
)
```

Arguments

`dat`	A data frame.
`pupil_col`	Numeric pupil column.
`luminance_col`	Numeric luminance/brightness column.
`group_cols`	Optional grouping columns.
`time_col`	Optional time column.
`output_col`	Output luminance-adjusted pupil column.
`fitted_col`	Output fitted luminance component column.
`include_quadratic`	Logical. If `TRUE`, include luminance squared.
`model_by_group`	Logical. If `TRUE`, fit models per group.
`add_intercept_mean`	Logical. If `TRUE`, add mean pupil size back to residuals.

Value

A data frame with luminance-adjusted pupil columns and attributes.

`run_gazepoint_automated_statistics` *Run automated exploratory statistics for Gazepoint feature tables*

Description

Runs simple exploratory group comparisons for numeric feature columns. The function selects one-way ANOVA when all groups pass Shapiro-Wilk checks and Kruskal-Wallis otherwise. It also performs pairwise post-hoc tests with multiplicity correction.

Usage

```
run_gazepoint_automated_statistics(
  dat,
  outcome_cols,
  group_col,
  alpha = 0.05,
  p_adjust_method = "holm",
  normality_alpha = 0.05,
  min_group_n = 3
)
```

Arguments

`dat`	A data frame.
`outcome_cols`	Numeric outcome columns.
`group_col`	Grouping/condition column.
`alpha`	Significance level.
`p_adjust_method`	P-value adjustment method.
`normality_alpha`	Alpha used for Shapiro-Wilk normality screening.
`min_group_n`	Minimum observations per group.

Details

This is an exploratory reporting helper. It is not a substitute for a preregistered statistical model or expert review of study design.

Value

A list with `overview`, `test_table`, `posthoc_table`, `normality_table`, and `settings`.

`run_gazepoint_biometrics_real_data_readiness` *Run a final real-data readiness gate for Gazepoint biometrics data*

Description

Provides conservative pass/warn/fail checks before using real Gazepoint Biometrics exports for analysis or reporting. The function checks basic row count, signal availability, missingness, time ordering, TTL availability, and HRV/IBI caution status. It does not certify data quality or infer emotional states.

Usage

```
run_gazepoint_biometrics_real_data_readiness(
  data = NULL,
  workflow_result = NULL,
  min_rows = 100,
  min_active_signal_count = 1,
  max_missing_prop = 0.5,
  required_signal_cols = NULL,
  require_gsr_us_preferred = TRUE,
  require_ibi_for_hrv = FALSE,
  time_col = NULL,
  ttl_cols = NULL
)
```

Arguments

`data`	A biometric data frame. If `NULL`, the function tries to extract a data frame from `workflow_result`.
`workflow_result`	Optional workflow/list object containing biometric data.
`min_rows`	Minimum number of rows expected for a usable real-data check.
`min_active_signal_count`	Minimum number of biometric signal columns with at least one non-missing/non-zero value.
`max_missing_prop`	Maximum acceptable missing proportion for detected signal columns before a warning is raised.
`required_signal_cols`	Optional signal columns that must be present.
`require_gsr_us_preferred`	Logical. If `TRUE`, warns when `GSR_US` is absent but `GSR` is present.
`require_ibi_for_hrv`	Logical. If `TRUE`, fails when HRV is present but IBI is absent. If `FALSE`, this condition is reported as a warning.

time_col	Optional time column. If NULL, common time columns are detected automatically.
ttl_cols	Optional TTL marker columns. If NULL, ttl_marker or TTL0-TTL6 are detected automatically.

Value

A list with overview, checks, signal_summary, and settings.

`run_gazepoint_biometrics_workflow`
Run a Gazepoint Biometrics workflow

Description

Runs a compact end-to-end workflow for Gazepoint Biometrics exports. The workflow imports rectangular all-gaze/fixation-style CSV exports from a folder, validates biometric columns, detects active channels, audits missingness, signal quality, sampling/timing, optionally creates window-level summaries, creates optional biometric exclusion recommendations, extracts optional TTL marker events, and produces checklist and methods-text outputs.

Usage

```
run_gazepoint_biometrics_workflow(
  path,
  group_columns = NULL,
  recursive = FALSE,
  include_fixations = FALSE,
  include_all_gaze = TRUE,
  include_other_csv = FALSE,
  require_active_signal = TRUE,
  create_exclusion_recommendations = TRUE,
  gsr_min_usable_pct = 50,
  hr_min_usable_pct = 50,
  dial_min_usable_pct = 50,
  extract_ttl_events = TRUE,
  ttl_event_mode = c("changes", "nonzero"),
  audit_sampling = TRUE,
  sampling_group_columns = NULL,
  sampling_time_column = NULL,
  sampling_time_unit = c("samples", "seconds", "milliseconds", "microseconds"),
  expected_sampling_rate_hz = 60
)
```

Arguments

path	Folder containing Gazepoint CSV exports.
group_columns	Optional columns used to create multimodal window summaries, such as c("source_participant", "MEDIA_ID").
recursive	Should subfolders be searched?
include_fixations	Should fixation files be imported? Defaults to FALSE because continuous biometric summaries should usually be computed from all-gaze sample-level exports rather than fixation-level exports.
include_all_gaze	Should all-gaze files be imported?
include_other_csv	Should other non-Data-Summary CSV files be attempted?
require_active_signal	Logical. Should inactive biometric signals be flagged in validation/checklist outputs?
create_exclusion_recommendations	Logical. Should window-level and participant-level keep/review/exclude recommendations be created when group_columns are supplied?
gsr_min_usable_pct	Minimum acceptable usable percentage for GSR/EDA windows.
hr_min_usable_pct	Minimum acceptable usable percentage for heart-rate windows.
dial_min_usable_pct	Minimum acceptable usable percentage for engagement-dial windows.
extract_ttl_events	Logical. Should TTL marker events be extracted?
ttl_event_mode	TTL event extraction mode passed to extract_gazepoint_ttl_events(). Use "changes" or "nonzero".
audit_sampling	Logical. Should sampling/timing information be audited?
sampling_group_columns	Optional columns for the sampling audit. If NULL, the workflow uses available file/participant/media columns.
sampling_time_column	Optional time/order column for the sampling audit.
sampling_time_unit	Unit of the selected time/order column. Use "seconds", "milliseconds", "microseconds", or "samples".
expected_sampling_rate_hz	Optional expected sampling rate in Hz.

Value

A list with imported data, validation outputs, missingness summaries, quality audits, sampling/timing audits, optional window summaries, optional exclusion recommendations, optional TTL events, checklist, and methods text. The object has class "gazepoint_biometrics_workflow".

`run_gazepoint_eda_analysis_pipeline` *Run a six-phase Gazepoint EDA/GSR analysis pipeline*

Description

Runs a conservative six-phase Gazepoint EDA/GSR workflow using native `gpbiometrics` helpers where possible. The function imports or accepts data, audits signal quality, prepares preprocessing outputs, creates optional external-method bridge inputs, prepares synchronization/model-formatting outputs, attaches model templates, and generates reporting outputs.

Usage

```
run_gazepoint_eda_analysis_pipeline(
  data = NULL,
  path = NULL,
  eda_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  signal_cols = NULL,
  sampling_rate = NULL,
  baseline_window = NULL,
  event_windows = NULL,
  event_data = NULL,
  lag_signal_pair = NULL,
  convert_resistance_to_us = FALSE,
  prepare_external_bridges = TRUE,
  bridge_methods = c("neurokit", "cvxeda", "ledalab", "pspm"),
  prepare_model_data = TRUE,
  create_reports = TRUE,
  output_dir = NULL,
  prefix = "gazepoint_eda_pipeline",
  continue_on_error = TRUE
)
```

Arguments

`data`	Optional Gazepoint biometric data frame or imported object.
`path`	Optional file or folder path. Used only when `data` is `NULL`.
`eda_col`	Optional EDA/conductance column. If omitted, the runner prefers `GSR_US` when available.
`time_col`	Optional time/counter column.
`group_cols`	Optional grouping columns.
`signal_cols`	Optional biometric signal columns for activity plots and signal audits.
`sampling_rate`	Optional sampling rate in Hz, used when `time_col` is a sample counter.

`baseline_window`
: Optional baseline window object passed to baseline correction helpers when supported.

`event_windows`
: Optional event-window table used for SCR event-window summaries when available.

`event_data`
: Optional event table used for TTL/event alignment when available.

`lag_signal_pair`
: Optional character vector of length two giving signals for lag estimation.

`convert_resistance_to_us`
: Logical. If `TRUE`, allow conservative resistance-to-conductance conversion when the selected EDA column is `GSR`.

`prepare_external_bridges`
: Logical. If `TRUE`, prepare selected external EDA bridge inputs.

`bridge_methods`
: Character vector containing any of `"neurokit"`, `"cvxeda"`, `"ledalab"`, and `"pspm"`.

`prepare_model_data`
: Logical. If `TRUE`, attempt to create SCR hurdle and biometric LME-ready data objects.

`create_reports`
: Logical. If `TRUE`, attempt to create report outputs.

`output_dir`
: Optional output directory for report bundles or bridge files where supported.

`prefix`
: File prefix used by output-producing helpers where supported.

`continue_on_error`
: Logical. If `TRUE`, failed steps are stored in `errors` and the pipeline continues.

Details

The function does not fit `brms` or `lme4` models, does not run external software, and does not infer emotion, valence, stress, trust, preference, cognition, or diagnosis.

Value

A list with `overview`, `phases`, `errors`, `pipeline_guide`, `model_templates`, `reporting_guidance`, `interpretation_guardrails`, and `settings`.

`run_gazepoint_neurokit_eda_crosscheck` *Optionally run a NeuroKit2 EDA cross-check*

Description

Optionally calls Python/NeuroKit2 on prepared EDA input. By default, `execute = FALSE`, so no external dependency is required.

Usage

```
run_gazepoint_neurokit_eda_crosscheck(
  data,
  eda_col = "GSR_US",
  time_col = NULL,
  group_cols = NULL,
  sampling_rate = NULL,
  execute = FALSE,
  python = "python",
  output_dir = tempdir(),
  prefix = "gazepoint_neurokit_crosscheck",
  keep_files = FALSE
)
```

Arguments

data	A Gazepoint biometric data frame or gazepoint_neurokit_eda_input object.
eda_col	EDA/GSR signal column, used when data is a data frame.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
sampling_rate	Sampling rate in Hz required for NeuroKit2 execution.
execute	Logical. If FALSE, only prepare input and return skipped status.
python	Python executable.
output_dir	Directory for temporary/input/output files.
prefix	File prefix.
keep_files	Logical. If FALSE, temporary files produced during execution may be removed.

Value

A list with overview, prepared_input, crosscheck_summary, manifest, and settings.

---

run_gazepoint_online_design_optimization

Run blockwise online design optimization decision support

Description

Provides a safe, dependency-light decision-support/simulation helper for online design optimization. The function recommends the next condition by combining expected model-discrimination utility with optional exploration and balancing penalties.

Usage

```
run_gazepoint_online_design_optimization(
  candidate_table,
  condition_col = "condition",
  utility_col = "expected_utility",
  block_col = NULL,
  cost_col = NULL,
  previous_assignments = NULL,
  exploration_weight = 0.1,
  balance_weight = 0.1,
  maximise = TRUE
)
```

Arguments

`candidate_table`	A data frame containing candidate conditions.
`condition_col`	Candidate condition column.
`utility_col`	Expected utility/model-discrimination column.
`block_col`	Optional block column.
`cost_col`	Optional cost or burden column subtracted from utility.
`previous_assignments`	Optional previous condition assignments.
`exploration_weight`	Weight for favouring under-sampled conditions.
`balance_weight`	Weight for penalising over-sampled conditions.
`maximise`	Logical. If `TRUE`, select highest score.

Details

This function does not control stimulus presentation software and should not be used as autonomous real-time experiment control without separate ethical, preregistration, and software-integration review.

Value

A list with `overview`, `ranked_candidates`, `recommendation`, `assignment_summary`, and `settings`.

`run_gazepoint_scr_multiverse` *Run a multiverse of SCR scoring specifications*

Description

Scores SCR amplitudes across multiple latency windows, thresholds, baseline methods, and response metrics. Optionally applies a user-supplied model function to each specification. This supports transparent sensitivity analysis and specification-curve style reporting.

Usage

```
run_gazepoint_scr_multiverse(
  dat,
  signal_col = "GSR_US",
  time_col = "time",
  trial_cols = NULL,
  condition_col = NULL,
  participant_col = NULL,
  event_time_col = NULL,
  latency_windows = list(c(1, 3), c(1, 4), c(1, 5)),
  thresholds = c(0.01, 0.05),
  baseline_methods = c("median", "mean"),
  baseline_window = c(-1, 0),
  response_metrics = c("max_minus_baseline"),
  model_function = NULL
)
```

Arguments

`dat`	Sample-level EDA data.
`signal_col`	Conductance/EDA signal column.
`time_col`	Time column, preferably relative to stimulus onset. If `event_time_col` is supplied, relative time is computed as `time_col - event_time_col`.
`trial_cols`	Columns identifying trials.
`condition_col`	Optional experimental condition column.
`participant_col`	Optional participant column.
`event_time_col`	Optional event/stimulus onset time column.
`latency_windows`	List of response windows in seconds.
`thresholds`	SCR response thresholds.
`baseline_methods`	Baseline methods: `"median"`, `"mean"`, or `"none"`.

`baseline_window`
: Baseline window in relative seconds.

`response_metrics`
: Response metrics: `"max_minus_baseline"` or `"peak_to_peak"`.

`model_function`
: Optional function applied to each specification-level trial summary.

Value

A list with specification grid, scored trials, optional model results, and robustness overview.

`run_gazepoint_scr_threshold_sensitivity`
Run Gazepoint SCR threshold sensitivity checks

Description

Re-runs SCR peak detection across combinations of amplitude thresholds and minimum peak-distance settings. Optionally, it also carries each peak-detection result through SCR event-window summaries so users can see how preprocessing choices affect event-level response rates.

Usage

```
run_gazepoint_scr_threshold_sensitivity(
  data,
  phasic_col = NULL,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  amplitude_min_values = c(0.005, 0.01, 0.02, 0.03),
  min_peak_distance_values = c(1, 5, 10, 20, 30),
  recovery_fraction = 0.5,
  smooth_width = 1,
  events = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  event_label_col = NULL,
  ttl_cols = NULL,
  ttl_valid_col = NULL,
  event_detection = c("rising", "active"),
  analysis_window = c(0, 6),
  response_window = c(1, 4),
  peak_selection = c("largest_amplitude", "first_peak"),
  collapse_simultaneous_events = FALSE,
  include_event_windows = TRUE,
  keep_objects = FALSE
)
```

Arguments

data	A Gazepoint biometric data frame.
phasic_col	Optional phasic EDA signal column, typically GSR_US_PHASIC.
signal_col	Optional conductance-like fallback signal column, typically GSR_US.
time_col	Optional time/counter column.
group_cols	Optional grouping columns.
amplitude_min_values	Numeric vector of SCR amplitude thresholds.
min_peak_distance_values	Numeric vector of minimum peak distances.
recovery_fraction	Fraction used by detect_gazepoint_scr_peaks() to define recovery.
smooth_width	Optional moving-average width passed to detect_gazepoint_scr_peaks().
events	Optional event table passed to summarise_gazepoint_scr_event_windows().
event_time_col	Optional event-time column when events is supplied.
event_id_col	Optional event identifier column when events is supplied.
event_label_col	Optional event label column when events is supplied.
ttl_cols	Optional TTL columns for event derivation when events is NULL.
ttl_valid_col	Optional TTL validity column.
event_detection	Event-detection rule for TTL columns.
analysis_window	Event-relative analysis window.
response_window	Event-relative response window.
peak_selection	Peak-selection rule for event-window summaries.
collapse_simultaneous_events	Logical. Passed to summarise_gazepoint_scr_event_windows().
include_event_windows	Logical. If TRUE, compute event-window summaries for each sensitivity setting.
keep_objects	Logical. If TRUE, retain peak-detection and event-window objects in list columns.

Value

A list with overview, sensitivity_grid, peak_group_summary, event_window_summary, optional objects, and settings.

run_gpbiometrics_shiny

Launch a lightweight gpbiometrics Shiny dashboard

Description

Launches an optional local Shiny interface for inspecting a Gazepoint Biometrics CSV file. Shiny is optional and is not required to use the package.

Usage

```
run_gpbiometrics_shiny()
```

Value

A Shiny application object, launched for interactive use.

run_gpbiometrics_shiny_annotator

Launch a lightweight gpbiometrics Shiny peak/artifact annotator

Description

Launches a local Shiny app for manual EDA peak/artifact annotation. The app lets users upload a Gazepoint CSV, inspect a selected signal, enter manual peak/artifact intervals, and download annotation CSVs.

Usage

```
run_gpbiometrics_shiny_annotator()
```

Details

This is an optional local GUI helper. It does not replace automated scoring or infer emotion, stress, cognition, trust, preference, or diagnosis.

Value

A Shiny application object, launched for interactive use.

screen_gazepoint_eda_nonresponders
Screen Gazepoint EDA/SCR nonresponders

Description

Screens groups, usually participants, for low or absent SCR responding using either SCR event-window summaries or SCR peak-detection outputs. The helper reports candidate nonresponders under explicit, user-controlled criteria.

Usage

```
screen_gazepoint_eda_nonresponders(
  x,
  group_cols = NULL,
  response_col = "response_flag",
  amplitude_col = "scr_amplitude",
  min_events = 1,
  min_response_events = 1,
  min_response_rate = 0.05,
  min_detected_peaks = 1
)
```

Arguments

x	A gazepoint_scr_event_window_summary, gazepoint_scr_peak_detection, or data frame.
group_cols	Optional grouping columns, usually participant columns.
response_col	Binary response column for event-window data.
amplitude_col	SCR amplitude column.
min_events	Minimum number of events required before event-window nonresponder classification is made.
min_response_events	Minimum number of response events required.
min_response_rate	Minimum response rate required.
min_detected_peaks	Minimum detected peaks required when only peak data are available.

Value

A list with overview, group_summary, candidate_nonresponders, and settings.

`simulate_gazepoint_biometrics` *Simulate Gazepoint-style biometric signals*

Description

Generates synthetic Gazepoint-style EDA, PPG, HR, IBI, and TTL-like signals with known ground-truth events. This is intended for teaching, examples, unit tests, and model-validation workflows. It does not generate real participant physiology.

Usage

```
simulate_gazepoint_biometrics(
  n_seconds = 120,
  sampling_rate = 60,
  participant_id = "sim_p1",
  scr_onsets = NULL,
  scr_rate_per_min = 4,
  pulse_rate_bpm = 72,
  respiration_rate_bpm = 15,
  eda_noise_sd = 0.01,
  ppg_noise_sd = 0.02,
  include_ttl = TRUE,
  seed = NULL
)
```

Arguments

`n_seconds`	Duration in seconds.
`sampling_rate`	Sampling rate in Hz.
`participant_id`	Participant identifier.
`scr_onsets`	Optional SCR onset times in seconds.
`scr_rate_per_min`	Expected SCR events per minute when `scr_onsets` is `NULL`.
`pulse_rate_bpm`	Mean pulse rate.
`respiration_rate_bpm`	Respiration-modulation rate.
`eda_noise_sd`	EDA noise SD.
`ppg_noise_sd`	PPG noise SD.
`include_ttl`	Logical. If `TRUE`, adds TTL0 pulses at SCR onsets.
`seed`	Optional random seed.

Value

A list with `overview`, `data`, `ground_truth`, and `settings`.

smooth_gazepoint_biometrics

Smooth a Gazepoint biometric signal

Description

Adds a simple centered moving-average smoothing column to a Gazepoint Biometrics table. This is intentionally conservative and dependency-free. It does not replace specialised biosignal preprocessing libraries.

Usage

```
smooth_gazepoint_biometrics(
  data,
  value_column,
  window = 5L,
  output_column = NULL,
  na_rm = TRUE
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
value_column	Column to smooth.
window	Number of samples in the moving window. Must be a positive odd integer.
output_column	Name of the smoothed output column.
na_rm	Should missing values be ignored within the moving window?

Value

A data frame with the added smoothed column.

standardise_gazepoint_adaptive_ema

Adaptive EMA normalization for non-stationary EDA

Description

Applies dependency-light adaptive normalization using an exponential moving average center and robust local scale after IQR-based outlier screening. This preserves local dynamics more than whole-session z-scoring, but it is still a preprocessing transformation and not an emotion/stress classifier.

Usage

```
standardise_gazepoint_adaptive_ema(
  dat,
  signal_col = "GSR_US",
  group_cols = NULL,
  time_col = NULL,
  alpha = 0.05,
  iqr_multiplier = 1.5,
  suffix = "_adaptive_ema",
  center_suffix = "_ema_center",
  scale_suffix = "_ema_scale",
  min_scale = 1e-08,
  overwrite = FALSE
)

standardize_gazepoint_adaptive_ema(
  dat,
  signal_col = "GSR_US",
  group_cols = NULL,
  time_col = NULL,
  alpha = 0.05,
  iqr_multiplier = 1.5,
  suffix = "_adaptive_ema",
  center_suffix = "_ema_center",
  scale_suffix = "_ema_scale",
  min_scale = 1e-08,
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame.
`signal_col`	Numeric signal column.
`group_cols`	Optional grouping columns.
`time_col`	Optional time column used to order rows within group.
`alpha`	EMA smoothing parameter in `(0, 1]`.
`iqr_multiplier`	IQR multiplier for outlier screening.
`suffix`	Suffix for the normalized output column.
`center_suffix`	Suffix for the EMA center column.
`scale_suffix`	Suffix for the EMA scale column.
`min_scale`	Minimum scale used to avoid division by zero.
`overwrite`	Logical. If `FALSE`, existing output columns are protected.

Value

A data frame with adaptive normalized signal columns and attributes.

`standardise_gazepoint_biometric_names` *Standardise Gazepoint biometric column names*

Description

Standardises common Gazepoint Biometrics column-name variants to stable canonical names. The helper is intentionally conservative: it recognises common biometric, timing, marker, participant, and stimulus columns, but it leaves unknown columns unchanged apart from optional snake-case cleaning.

Usage

```
standardise_gazepoint_biometric_names(
  data,
  style = c("canonical", "snake"),
  rename = TRUE
)
```

Arguments

`data`	A data frame or a character vector of column names.
`style`	Naming style to return. `"canonical"` returns uppercase Gazepoint-style names for recognised columns. `"snake"` returns lowercase snake-case names.
`rename`	Logical. If `data` is a data frame, should the returned data frame have standardised names? If `FALSE`, a name-mapping table is returned.

Value

If `data` is a character vector, a character vector of standardised names. If `data` is a data frame and `rename = TRUE`, the data frame with standardised names. If `rename = FALSE`, a data frame mapping original names to standardised names.

Examples

```
standardise_gazepoint_biometric_names(c("time ms", "heart rate", "eda uS"))

df <- data.frame(`time ms` = 1:3, `heart rate` = c(70, 72, 71))
names(standardise_gazepoint_biometric_names(df))
```

standardise_gazepoint_plot_contract

Standardise a Gazepoint plot return contract

Description

Adds consistent attributes to a ggplot object so plotting helpers can be tested and reused in automated reports.

Usage

```
standardise_gazepoint_plot_contract(
  plot,
  plot_data = NULL,
  settings = list(),
  interpretation_notes = NULL,
  plot_type = NULL
)
```

Arguments

plot	A ggplot object.
plot_data	Optional data frame used to create the plot.
settings	Optional list of plot settings.
interpretation_notes	Optional character vector of interpretation notes.
plot_type	Optional short plot-type label.

Value

A ggplot object with standardized attributes.

standardise_gazepoint_range_correction

Standardise SCR or SCL using within-participant range correction

Description

Adds a range-corrected signal column using `(x - min) / (max - min)` within participant or another grouping unit. This expresses each value as a proportion of the observed within-unit range.

Usage

```
standardise_gazepoint_range_correction(
  dat,
  signal_col,
  group_col = "source_participant",
  suffix = "_Range_Corrected",
  min_valid = 2,
  zero_range_action = c("NA", "zero"),
  overwrite = FALSE
)

standardize_gazepoint_range_correction(
  dat,
  signal_col,
  group_col = "source_participant",
  suffix = "_Range_Corrected",
  min_valid = 2,
  zero_range_action = c("NA", "zero"),
  overwrite = FALSE
)
```

Arguments

`dat`	A data frame containing SCR, SCL, or another biometric signal.
`signal_col`	Signal column to range-correct.
`group_col`	Participant or unit column.
`suffix`	Suffix for the output column.
`min_valid`	Minimum finite observations required within each group.
`zero_range_action`	What to do when max equals min: `"NA"` or `"zero"`.
`overwrite`	Logical. If `FALSE`, existing output columns are protected.

Value

A data frame with an added range-corrected column.

---

standardise_gazepoint_zscore

Standardise SCR or SCL using intra-individual z-scoring

Description

Adds a within-participant z-scored version of a signal column. This is a lightweight compatibility wrapper around the package's more general within-unit standardisation helper.

Usage

```
standardise_gazepoint_zscore(
  dat,
  signal_col = "SCR_Amplitude",
  group_col = "source_participant",
  suffix = "_Z",
  min_valid = 2,
  overwrite = FALSE
)

standardize_gazepoint_zscore(
  dat,
  signal_col = "SCR_Amplitude",
  group_col = "source_participant",
  suffix = "_Z",
  min_valid = 2,
  overwrite = FALSE
)
```

Arguments

dat	A data frame containing SCR, SCL, or another biometric signal.
signal_col	Signal column to standardise.
group_col	Participant or unit column.
suffix	Suffix for the output column.
min_valid	Minimum finite observations required within each group.
overwrite	Logical. If FALSE, existing output columns are protected.

Value

A data frame with an added z-scored column.

---

standardize_gazepoint_biometrics_within_unit

Standardize biometric signals within participant or other analysis units

---

Description

Adds within-unit standardized biometric columns, usually within participant or participant-by-session/stimulus groups. This is useful when the analysis focuses on relative within-person signal change rather than absolute between-person level differences.

Usage

```
standardize_gazepoint_biometrics_within_unit(
  data,
  signal_cols = NULL,
  unit_cols = NULL,
  reference_col = NULL,
  reference_value = TRUE,
  suffix = "_z_within",
  center = TRUE,
  scale = TRUE,
  min_valid = 2,
  zero_sd_action = c("NA", "zero"),
  overwrite = FALSE
)

standardise_gazepoint_biometrics_within_unit(
  data,
  signal_cols = NULL,
  unit_cols = NULL,
  reference_col = NULL,
  reference_value = TRUE,
  suffix = "_z_within",
  center = TRUE,
  scale = TRUE,
  min_valid = 2,
  zero_sd_action = c("NA", "zero"),
  overwrite = FALSE
)
```

Arguments

data	A data frame containing Gazepoint biometric data.
signal_cols	Character vector of biometric signal columns to standardize. If NULL, common numeric biometric columns are detected.
unit_cols	Character vector defining the unit within which means and standard deviations are computed. If NULL, common participant/session columns are detected. If no columns are detected, the whole data frame is treated as one unit.
reference_col	Optional logical or categorical column identifying rows used to estimate the reference mean and standard deviation. For example, this can be a baseline-window flag. The resulting parameters are then applied to all rows in the same unit.
reference_value	Value in reference_col that marks reference rows. Defaults to TRUE.
suffix	Suffix for standardized output columns.
center	Logical. If TRUE, subtract the within-unit reference mean.
scale	Logical. If TRUE, divide by the within-unit reference standard deviation.

min_valid	Minimum number of finite reference observations required per unit and signal.
zero_sd_action	What to do when the within-unit standard deviation is zero or unavailable. "NA" returns NA; "zero" returns zero for finite centered values.
overwrite	Logical. If FALSE, existing output columns are protected.

Details

The helper is intentionally conservative. It does not run automatically in the main workflow and does not infer emotion, valence, stress, trust, preference, cognition, or diagnosis. Within-unit z-scoring removes between-unit level and scale differences and should therefore be reported explicitly.

Value

A data frame with added standardized columns. Attributes include standardization_summary, standardization_parameters, and settings.

---

standardize_gazepoint_plot_contracts

Standardize Gazepoint plot return contracts

Description

US-spelling compatibility wrapper around standardise_gazepoint_plot_contract(). The wrapper accepts either a single ggplot object or a list of ggplot objects. For a list of plots, plot_data, settings, interpretation_notes, and plot_type may be supplied either as single values applied to all plots or as same-length lists/vectors applied elementwise.

Usage

```
standardize_gazepoint_plot_contracts(
  plot,
  plot_data = NULL,
  settings = list(),
  interpretation_notes = NULL,
  plot_type = NULL
)
```

Arguments

plot	A ggplot object, or a list of ggplot objects.
plot_data	Optional data frame, or a list of data frames when plot is a list.
settings	Optional settings list, or a list of settings lists when plot is a list.
interpretation_notes	Optional character vector, or a list/character vector of notes when plot is a list.
plot_type	Optional plot-type label, or a character vector/list of labels when plot is a list.

Value

A standardized ggplot object, or a list of standardized ggplot objects.

`summarise_gazepoint_aoi_biometrics`
Summarise biometric signals by AOI

Description

Summarises Gazepoint biometric channels within area-of-interest (AOI) rows. The helper is intended for AOI-linked physiological descriptions, not for inferring emotional valence.

Usage

```
summarise_gazepoint_aoi_biometrics(
  data,
  aoi_col = "AOI",
  signal_cols = NULL,
  group_cols = NULL,
  time_col = NULL,
  valid_aoi_values = NULL,
  drop_missing_aoi = TRUE,
  min_rows = 1
)
```

Arguments

`data`	A Gazepoint data frame containing AOI labels and biometric signals.
`aoi_col`	AOI label column.
`signal_cols`	Biometric signal columns to summarise.
`group_cols`	Optional grouping columns, for example participant/media.
`time_col`	Optional time/counter column.
`valid_aoi_values`	Optional AOI labels to retain.
`drop_missing_aoi`	Logical. If `TRUE`, rows with missing/blank AOI labels are excluded.
`min_rows`	Minimum rows required for a group/AOI/signal summary to be marked as usable.

Value

A list with `overview`, `summary`, `signal_summary`, `aoi_summary`, `data`, and `settings`.

`summarise_gazepoint_biometrics_feature_inventory`
Summarise the formatted gpbiometrics feature inventory

Description

Summarise the formatted gpbiometrics feature inventory

Usage

```
summarise_gazepoint_biometrics_feature_inventory(formatted_inventory = NULL)
```

Arguments

`formatted_inventory`
Optional table returned by `format_gazepoint_biometrics_feature_inventory()`.

Value

A list with domain, method-family, and user-level summaries.

`summarise_gazepoint_biometrics_workflow`
Summarise a Gazepoint Biometrics workflow object

Description

Creates a compact summary table from an object returned by `run_gazepoint_biometrics_workflow()`.

Usage

```
summarise_gazepoint_biometrics_workflow(workflow)
```

Arguments

`workflow` A workflow object returned by `run_gazepoint_biometrics_workflow()`.

Value

A one-row data frame summarising the workflow.

summarise_gazepoint_biometric_validity
Summarise validity and availability of Gazepoint biometric signals

Description

Summarises missingness, finite numeric availability, variability, and optional validity-flag columns for Gazepoint Biometrics data. The helper is descriptive: it reports whether biometric signals appear available and usable, but it does not infer emotion, valence, or HRV from ambiguous raw columns. In particular, raw HRV columns are treated as validity/vendor flags unless the user has independent documentation proving otherwise.

Usage

```
summarise_gazepoint_biometric_validity(
  data,
  signal_cols = NULL,
  validity_cols = NULL,
  group_cols = NULL,
  active_min_unique = 2L
)
```

Arguments

data	A data frame.
signal_cols	Optional character vector of biometric signal columns to summarise. If NULL, common Gazepoint biometric signal columns are detected from the column names.
validity_cols	Optional character vector of validity-flag columns to summarise. If NULL, common validity-like columns are detected, including HRV when present.
group_cols	Optional character vector of grouping columns, such as participant, stimulus, trial, or condition columns.
active_min_unique	Minimum number of unique finite values required for a numeric signal to be treated as active.

Value

A list with overview, signals, validity_flags, group_summary, and settings.

Examples

```
df <- data.frame(
  USER = rep(c("P1", "P2"), each = 4),
  GSR = c(1, 2, NA, 4, 2, 2, 2, 2),
  HR = c(70, 71, 72, NA, 80, 81, 82, 83),
  HRV = c(1, 1, 0, 1, 1, 1, 1, 1)
```

```
)
summarise_gazepoint_biometric_validity(df, group_cols = "USER")
```

`summarise_gazepoint_dial_windows` *Summarise Gazepoint engagement-dial windows*

Description

Compatibility wrapper for `summarise_gazepoint_engagement_windows()`. This helper uses the term "dial" for users who refer to Gazepoint engagement-dial or self-reported engagement streams, while delegating the calculation to the canonical engagement-window summariser.

Usage

```
summarise_gazepoint_dial_windows(data, ..., dial_col = NULL)
```

Arguments

`data`	A data frame containing Gazepoint Biometrics engagement/dial data.
`...`	Additional arguments passed to `summarise_gazepoint_engagement_windows()`.
`dial_col`	Optional dial/engagement column. When supplied, it is mapped to the corresponding value-column argument of the underlying helper.

Value

The output of `summarise_gazepoint_engagement_windows()`.

Examples

```
df <- data.frame(
  USER = rep(c("P1", "P2"), each = 3),
  DIAL = c(40, 45, 50, 55, 60, 65)
)
summarise_gazepoint_dial_windows(df)
```

summarise_gazepoint_engagement_windows
Summarise Gazepoint engagement-dial windows

Description

Summarises Gazepoint engagement-dial values within participant, trial, stimulus, AOI, or other user-defined windows.

Usage

```
summarise_gazepoint_engagement_windows(
  data,
  group_columns = NULL,
  value_column = "DIAL",
  validity_column = "DIALV",
  exclude_zero = FALSE
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
group_columns	Optional grouping columns defining windows, such as c("USER", "MEDIA_ID").
value_column	Engagement-dial value column. Defaults to "DIAL".
validity_column	Engagement-dial validity column. Defaults to "DIALV".
exclude_zero	Should zero values be excluded from usable summaries?

Value

A data frame with one row per window.

summarise_gazepoint_full_biometric_windows
Summarise full Gazepoint biometric windows

Description

Creates a combined window-level summary table containing GSR/EDA, heart-rate, engagement-dial, and IBI-derived HRV summaries. This function is intended for biometric analyses where both continuous physiological values and interbeat-interval variability features are needed.

Usage

```
summarise_gazepoint_full_biometric_windows(
  data,
  group_columns,
  include_ibi_hrv = TRUE
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
group_columns	Columns defining analysis windows, such as c("source_participant", "MEDIA_ID").
include_ibi_hrv	Logical. Should IBI-derived HRV summaries be included?

Value

A data frame with one row per window and prefixed biometric summary columns.

---

summarise_gazepoint_gsr_tonic_phasic

Summarise tonic and phasic GSR/EDA components

Description

Creates a simple descriptive tonic/phasic decomposition of a GSR/EDA signal. The tonic component is estimated with a rolling median, and the phasic component is the observed signal minus the rolling-median tonic estimate.

Usage

```
summarise_gazepoint_gsr_tonic_phasic(
  data,
  gsr_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  window_n = 15L,
  peak_threshold = NULL,
  output_prefix = "gsr"
)
```

Arguments

data	A data frame.
gsr_col	Optional GSR/EDA column. If NULL, the function prefers GSR_US, then GSR, then other recognised GSR/EDA columns.

group_cols	Optional grouping columns. Tonic/phasic values are computed separately within each group.
time_col	Optional time column used to order rows within groups.
window_n	Rolling-median window size in samples.
peak_threshold	Optional phasic peak threshold. If NULL, a robust data-driven threshold is computed within each group as median(phasic) + 2 * MAD(phasic).
output_prefix	Prefix for generated columns.

Details

This is a lightweight descriptive helper, not a full skin-conductance-response deconvolution model. It should be used for quality checks, window summaries, and exploratory reporting unless a study requires a specialised EDA model.

Value

A list with data, summary, and settings.

Examples

```
df <- data.frame(
  CNT = 1:10,
  GSR_US = c(1, 1.1, 1.0, 1.2, 2.0, 1.3, 1.2, 1.1, 1.0, 1.1)
)
summarise_gazepoint_gsr_tonic_phasic(df, window_n = 3)
```

---

summarise_gazepoint_gsr_windows

Summarise Gazepoint GSR/EDA windows

Description

Summarises Gazepoint GSR/EDA values within participant, trial, stimulus, AOI, or other user-defined windows. When available, GSR_US is used by default because it represents skin conductance in microsiemens in Gazepoint exports.

Usage

```
summarise_gazepoint_gsr_windows(
  data,
  group_columns = NULL,
  value_column = NULL,
  validity_column = "GSRV",
  exclude_zero = TRUE
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
group_columns	Optional grouping columns defining windows, such as c("USER", "MEDIA_ID").
value_column	Optional GSR/EDA value column. If NULL, GSR_US is used when present, otherwise GSR.
validity_column	Optional validity column. Defaults to "GSRV".
exclude_zero	Should zero values be excluded from usable summaries?

Value

A data frame with one row per window.

---

summarise_gazepoint_hrv_features

Summarise time-domain HRV features from Gazepoint IBI/RR intervals

---

Description

Computes conservative time-domain HRV-style features from genuine interbeat interval or RR interval columns. The raw Gazepoint HRV column is not used as an HRV metric because it is treated as a validity/vendor flag unless independently documented otherwise.

Usage

```
summarise_gazepoint_hrv_features(
  data,
  ibi_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  ibi_unit = c("auto", "seconds", "milliseconds"),
  min_ibi_ms = 300,
  max_ibi_ms = 2000,
  min_valid_ibi = 3L
)
```

Arguments

data	A data frame.
ibi_col	Optional IBI/RR interval column. If NULL, a likely IBI/RR column is detected. The raw HRV column is never selected automatically.
group_cols	Optional grouping columns, such as participant, stimulus, trial, or window.
time_col	Optional time/order column used to order IBI values within each group before calculating successive-difference features.

ibi_unit	Unit of the IBI/RR column. Use "auto", "seconds", or "milliseconds".
min_ibi_ms	Minimum plausible IBI in milliseconds.
max_ibi_ms	Maximum plausible IBI in milliseconds.
min_valid_ibi	Minimum number of valid IBI values required before a group is marked as having computed HRV features.

Details

The helper computes descriptive features including mean IBI, mean heart rate derived from IBI, SDNN, RMSSD, and pNN50. It does not compute frequency-domain HRV and does not replace specialised ECG/PPG HRV software.

Value

A list with overview, features, and settings.

Examples

```
df <- data.frame(
  participant = "P1",
  IBI = c(0.9, 1.0, 1.1, 1.0, 0.95)
)
summarise_gazepoint_hrv_features(df, group_cols = "participant")
```

---

summarise_gazepoint_hr_windows

Summarise Gazepoint heart-rate windows

Description

Summarises Gazepoint heart-rate values within participant, trial, stimulus, AOI, or other user-defined windows. HRV is treated as a validity flag, not as a heart-rate-variability metric.

Usage

```
summarise_gazepoint_hr_windows(
  data,
  group_columns = NULL,
  value_column = "HR",
  validity_column = "HRV",
  exclude_zero = TRUE
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
group_columns	Optional grouping columns defining windows, such as c("USER", "MEDIA_ID").
value_column	Heart-rate value column. Defaults to "HR".
validity_column	Heart-rate validity column. Defaults to "HRV".
exclude_zero	Should zero values be excluded from usable summaries?

Value

A data frame with one row per window.

---

summarise_gazepoint_ibi_hrv_windows

Summarise Gazepoint IBI-derived HRV windows

Description

Summarises interbeat-interval (IBI) values within participant, stimulus, trial, AOI, or other user-defined windows. This function derives simple time-domain variability features from IBI. It does not use the Gazepoint HRV column as a heart-rate-variability metric, because HRV is treated as the Gazepoint heart-rate validity flag.

Usage

```
summarise_gazepoint_ibi_hrv_windows(
  data,
  group_columns,
  ibi_column = "IBI",
  validity_column = "HRV",
  min_ibi = 0.3,
  max_ibi = 2
)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
group_columns	Columns defining analysis windows, such as c("source_participant", "MEDIA_ID").
ibi_column	Interbeat-interval column. Defaults to "IBI".
validity_column	Optional validity column. Defaults to "HRV".
min_ibi	Minimum plausible IBI in seconds.
max_ibi	Maximum plausible IBI in seconds.

Value

A data frame with one row per window and IBI-derived HRV summaries.

summarise_gazepoint_ibi_windows
Summarise IBI/RR windows

Description

Computes descriptive IBI/RR interval and simple HRV-style window summaries from genuine inter-beat interval data. The function does not use raw HRV columns as HRV metrics. It calculates metrics such as mean IBI, mean instantaneous heart rate, SDNN, RMSSD, pNN20, and pNN50 only from valid IBI/RR intervals.

Usage

```
summarise_gazepoint_ibi_windows(
  data,
  ibi_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  unit = c("auto", "milliseconds", "seconds"),
  min_ibi_ms = 300,
  max_ibi_ms = 2000,
  max_jump_ms = 500,
  exclude_large_jumps = TRUE,
  min_valid_ibi = 2L
)
```

Arguments

data	A data frame.
ibi_col	Optional IBI/RR interval column. If NULL, the function detects recognised IBI/RR-style column names.
group_cols	Optional grouping columns defining windows, such as participant, trial, stimulus, condition, or window labels.
time_col	Optional time/order column used to order IBI values before successive-difference metrics are computed.
unit	Unit of the IBI values. "auto" treats median values below 10 as seconds and larger values as milliseconds.
min_ibi_ms	Minimum plausible IBI in milliseconds.
max_ibi_ms	Maximum plausible IBI in milliseconds.
max_jump_ms	Maximum plausible absolute change between successive IBI values within a group.
exclude_large_jumps	Logical. Should intervals flagged as large jumps be excluded from the window summaries?
min_valid_ibi	Minimum valid IBI count required for a window to be marked as sufficient.

Value

A list with `overview`, `windows`, `samples`, and `settings`.

Examples

```
df <- data.frame(
  USER = rep(c("P1", "P2"), each = 4),
  IBI = c(800, 810, 790, 805, 900, 910, 905, 920)
)
summarise_gazepoint_ibi_windows(df, group_cols = "USER")
```

---

summarise_gazepoint_multimodal_windows

Summarise Gazepoint multimodal biometric windows

Description

Creates a combined window-level summary table for GSR/EDA, heart rate, and engagement dial. The output is suitable for later merging with eye-tracking summaries from `gp3tools`.

Usage

```
summarise_gazepoint_multimodal_windows(
  data,
  group_columns = NULL,
  exclude_zero = TRUE
)
```

Arguments

`data`	A data frame or a path to a Gazepoint CSV export.
`group_columns`	Optional grouping columns defining windows, such as `c("USER", "MEDIA_ID")`.
`exclude_zero`	Should zero values be excluded from GSR and heart-rate summaries?

Value

A data frame with one row per window and prefixed biometric summary columns.

summarise_gazepoint_scr_event_windows
Summarise Gazepoint SCR responses in event windows

Description

Creates one row per event or TTL marker and links detected SCR peaks to event-relative analysis and response windows. The helper is intended to produce transparent event-level EDA/SCR features for downstream mixed models, hurdle models, and reporting.

Usage

```
summarise_gazepoint_scr_event_windows(
  data = NULL,
  scr_peaks,
  events = NULL,
  time_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  event_label_col = NULL,
  group_cols = NULL,
  ttl_cols = NULL,
  ttl_valid_col = NULL,
  event_detection = c("rising", "active"),
  analysis_window = c(0, 6),
  response_window = c(1, 4),
  amplitude_col = "amplitude",
  peak_time_col = "peak_time",
  onset_time_col = "onset_time",
  rise_time_col = "rise_time",
  recovery_time_col = "recovery_time_after_peak",
  peak_status_col = "status",
  peak_selection = c("largest_amplitude", "first_peak"),
  collapse_simultaneous_events = FALSE
)
```

Arguments

data	Optional Gazepoint biometric data frame. Required when events is NULL and events should be derived from TTL columns.
scr_peaks	A gazepoint_scr_peak_detection object returned by detect_gazepoint_scr_peaks(), or a data frame of detected peaks.
events	Optional event data frame. If NULL, events are derived from TTL columns in data.
time_col	Optional time/counter column in data. Used for TTL-derived events.

`event_time_col` Optional event-time column in `events`. If `NULL`, common event-time column names are detected.

`event_id_col` Optional event identifier column in `events`.

`event_label_col`
Optional event label/condition column in `events`.

`group_cols` Optional grouping columns used to match events and peaks.

`ttl_cols` Optional TTL marker columns used when `events = NULL`.

`ttl_valid_col` Optional TTL validity column. If supplied, TTL-derived events require this column to be non-zero.

`event_detection`
Event-detection rule for TTL columns. `"rising"` detects rising edges; `"active"` treats every active TTL row as an event.

`analysis_window`
Numeric length-two vector giving the event-relative analysis window in the same units as `time_col` or `event_time_col`.

`response_window`
Numeric length-two vector giving the event-relative response window used for the binary SCR response flag.

`amplitude_col` Column in the peak table containing SCR amplitude.

`peak_time_col` Column in the peak table containing peak time.

`onset_time_col` Column in the peak table containing onset time.

`rise_time_col` Column in the peak table containing SCR rise time.

`recovery_time_col`
Column in the peak table containing recovery time after peak.

`peak_status_col`
Column in the peak table containing peak status.

`peak_selection` How to choose one peak when several peaks fall in the response window. `"largest_amplitude"` selects the largest response; `"first_peak"` selects the earliest peak.

`collapse_simultaneous_events`
Logical. If `TRUE`, events with the same group and event time are collapsed into one row before matching peaks. This is useful when Gazepoint TTL0–TTL6 channels mark the same event simultaneously.

Value

A list with `overview`, `event_table`, `window_qc`, `events`, `peaks`, and `settings`.

sync_gazepoint_biometrics_with_gaze

Synchronise Gazepoint Biometrics with gaze data

Description

Joins Gazepoint Biometrics data to gaze or fixation data using exact key columns. This function is intentionally conservative: it does not perform interpolation, nearest-neighbour matching, or automatic time shifting. For a first reproducible workflow, exact joins by participant, media, trial, sample counter, or fixation identifier are preferred when those fields are available.

Usage

```
sync_gazepoint_biometrics_with_gaze(
  biometrics,
  gaze,
  by,
  all_x = TRUE,
  suffixes = c(".gaze", ".bio")
)
```

Arguments

`biometrics`	A Gazepoint Biometrics data frame.
`gaze`	A gaze, fixation, AOI, or `gp3tools`-style summary data frame.
`by`	Character vector of key columns used for joining.
`all_x`	Logical. Should all rows from `gaze` be retained?
`suffixes`	Character vector of length two used for duplicate non-key column names.

Value

A data frame with gaze rows joined to biometric columns. The returned object has class `"gazepoint_biometrics_sync"` and a `"sync_summary"` attribute.

test_gazepoint_hrv_nonlinearity

Test HRV nonlinearity using surrogate data

Description

Tests whether a nonlinear HRV statistic differs from surrogate RR/IBI sequences. This is a screening tool for evidence inconsistent with a simple linear stochastic null process. It does not prove deterministic chaos or diagnose any condition.

Usage

```
test_gazepoint_hrv_nonlinearity(
  dat,
  ibi_col = "IBI",
  group_cols = NULL,
  metric = c("sample_entropy", "approximate_entropy", "sd1_sd2_ratio"),
  n_surrogates = 99,
  surrogate_method = c("phase_randomized", "shuffle"),
  m = 2,
  r_multiplier = 0.2,
  statistic_fun = NULL,
  seed = NULL
)
```

Arguments

dat	A data frame containing IBI/RR intervals.
ibi_col	Numeric IBI/RR interval column.
group_cols	Optional grouping columns.
metric	Nonlinear statistic to test.
n_surrogates	Number of surrogate series per group.
surrogate_method	"phase_randomized" or "shuffle".
m	Embedding dimension for entropy metrics.
r_multiplier	Tolerance multiplier for entropy metrics.
statistic_fun	Optional custom statistic function accepting numeric x.
seed	Optional random seed.

Value

A list with overview, results, surrogate_statistics, and settings.

---

validate_gazepoint_biometrics

Validate a Gazepoint Biometrics export

Description

Performs a conservative validation of a Gazepoint Biometrics table or CSV file. The function checks whether known biometric columns are present, whether biometric channels appear active, whether common time/synchronisation columns are available, and whether obvious structural issues are present.

Usage

```
validate_gazepoint_biometrics(data, require_active_signal = FALSE)
```

Arguments

data	A data frame or a path to a Gazepoint CSV export.
require_active_signal	Logical. If TRUE, validation reports a warning when no active GSR/EDA, heart-rate, or engagement-dial channel is detected.

Value

A list with overview, columns, active_channels, and issues. The returned object has class "gazepoint_biometrics_validation".

---

write_gazepoint_biometrics_report_tables

Write Gazepoint Biometrics report tables

Description

Writes report-ready Gazepoint Biometrics tables to CSV files. The input can be a workflow object produced by run_gazepoint_biometrics_workflow(), a report-table object produced by create_gazepoint_biometrics_report_tables(), or a named list of data frames.

Usage

```
write_gazepoint_biometrics_report_tables(
  tables,
  output_dir,
  prefix = "gazepoint_biometrics",
  overwrite = TRUE,
  include_empty_message_tables = FALSE
)
```

Arguments

tables	A Gazepoint Biometrics workflow object, report-table object, or named list of data frames.
output_dir	Output directory for CSV files.
prefix	Filename prefix.
overwrite	Should existing files be overwritten?
include_empty_message_tables	Should placeholder tables containing only a message column be written?

Value

A data frame indexing written and skipped files.

Index