Package: gpbiometrics (via r-universe)

October 9, 2026

Type Package

Title Process and Report Gazepoint Biometrics Data

Version 2.1.0

Description Gazepoint-native tools for reproducible analysis and reporting of Gazepoint Biometrics and GP3-derived physiological, gaze, pupil, fixation, AOI, marker, and event-level exports. The package supports import, validation, preprocessing, missing-value imputation, EDA/GSR, SCR epoching, IBI/RR/HRV, PPG/pulse workflows, pupil blink detection and cleaning, gaze filtering, fixation summaries, engagement-dial indices, multimodal trial regressors, quality reports, exclusion and QC recommendations, synthetic eye-data generation, and interoperability or workflow-inspired helpers for common psychophysiology toolchains. The package is designed for transparent, CRAN-safe, scriptable analysis rather than as a full clone of external commercial or open-source toolboxes.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

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

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

VignetteBuilder knitr

URL https://github.com/stefanosbalaskas/gpbiometrics, https://stefanosbalaskas.github.io/gpbiometrics/

BugReports https://github.com/stefanosbalaskas/gpbiometrics/issues

Repository https://stefanosbalaskas.r-universe.dev

Date/Publication 2026-10-09 20:31:54 UTC

RemoteUrl https://github.com/stefanosbalaskas/gpbiometrics

RemoteRef HEAD

RemoteSha 7794ebd02e73f9cc167061f6ff9722c7127dcc40

Contents

add_gazepoint_decision . . . 10
align_gazepoint_biometrics_to_ttl . . . 11
align_gazepoint_streams_by_events . . . 13
align_gazepoint_to_eeg . . . 14
analyze_gazepoint_ac_susceptance . . . 16
analyze_gazepoint_cardiorespiratory_causality . . . 17
analyze_gazepoint_skin_potential . . . 18
anonymize_gazepoint_data . . . 19
assert_gazepoint_columns . . . 19
assess_gazepoint_hrp_waveform_quality . . . 20
assess_gazepoint_sampling_irregularity . . . 21
assign_gazepoint_aoi . . . 22
audit_gazepoint_beats . . . 25
audit_gazepoint_biometric_missingness . . . 26
audit_gazepoint_biometric_sampling . . . 26
audit_gazepoint_biometric_sync_drift . . . 27
audit_gazepoint_biometrics_file . . . 28
audit_gazepoint_condition_balance . . . 29
audit_gazepoint_dataset_structure . . . 30
audit_gazepoint_distributional_drift . . . 31
audit_gazepoint_eda_artifacts . . . 32
audit_gazepoint_engagement_dial . . . 33
audit_gazepoint_event_coverage . . . 34
audit_gazepoint_experiment_design . . . 35
audit_gazepoint_export_schema . . . 36
audit_gazepoint_gsr_quality . . . 37
audit_gazepoint_gsr_units . . . 37
audit_gazepoint_hr_quality . . . 38
audit_gazepoint_ibi_quality . . . 39
audit_gazepoint_interoperability_versions . . . 40
audit_gazepoint_pipeline_steps . . . 41
audit_gazepoint_preregistration_consistency . . . 42
audit_gazepoint_pupil_luminance . . . 43
audit_gazepoint_release_readiness . . . 43
audit_gazepoint_session_comparability . . . 44
audit_gazepoint_signal_activity . . . 45
audit_gazepoint_smoke_privacy . . . 46
audit_gazepoint_stabilization_period . . . 46
audit_gazepoint_time_resets . . . 47
audit_gazepoint_timecourse_grid . . . 48
baseline_correct_gazepoint_gsr . . . 49

baseline_correct_gazepoint_hr . . . 50
baseline_correct_gazepoint_pupil . . . 51
build_gazepoint_aoi_timecourse . . . 52
calculate_gazepoint_rsa . . . 53
check_gazepoint_bids . . . 54
check_gazepoint_biometric_columns . . . 55
check_gazepoint_plot_contract . . . 55
check_gazepoint_ppg_binary_quality . . . 56
check_gazepoint_pyhrv_interval . . . 57
chunk_gazepoint_biometrics . . . 57
classify_gazepoint_eda_response_pattern . . . 58
classify_gazepoint_scr_intervals . . . 59
classify_gazepoint_signal_quality . . . 60
clean_gazepoint_pupil . . . 60
clean_gazepoint_pupil_signal . . . 61
clean_gazepoint_rr_intervals . . . 63
combine_gazepoint_marker_channels_pspm_style . . . 63
compare_gazepoint_conditions_bootstrap . . . 64
compare_gazepoint_export_profiles . . . 65
compare_gazepoint_hr_ibi_consistency . . . 66
compare_gazepoint_pyhrv_psd_methods . . . 67
compare_hrv_prv_devices . . . 67
compute_gazepoint_engagement_index . . . 68
compute_gazepoint_hrv_wavelet_psd . . . 69
compute_gazepoint_ppg_frequency_measures . . . 70
compute_gazepoint_ppg_measures . . . 71
compute_gazepoint_ppg_template_similarity . . . 71
compute_gazepoint_pyhrv_ar_psd . . . 72
compute_gazepoint_pyhrv_dfa . . . 73
compute_gazepoint_pyhrv_frequency_domain . . . 73
compute_gazepoint_pyhrv_heart_rate . . . 74
compute_gazepoint_pyhrv_hr_parameters . . . 74
compute_gazepoint_pyhrv_lomb_psd . . . 75
compute_gazepoint_pyhrv_nn_diff . . . 75
compute_gazepoint_pyhrv_nn20 . . . 76
compute_gazepoint_pyhrv_nn50 . . . 76
compute_gazepoint_pyhrv_nni_differences_parameters . . . 77
compute_gazepoint_pyhrv_nni_parameters . . . 77
compute_gazepoint_pyhrv_nnxx . . . 78
compute_gazepoint_pyhrv_nonlinear . . . 78
compute_gazepoint_pyhrv_poincare . . . 79
compute_gazepoint_pyhrv_psd_waterfall . . . 79
compute_gazepoint_pyhrv_rmssd . . . 80
compute_gazepoint_pyhrv_sample_entropy . . . 80
compute_gazepoint_pyhrv_sdann . . . 81
compute_gazepoint_pyhrv_sdnn . . . 81
compute_gazepoint_pyhrv_sdnn_index . . . 82
compute_gazepoint_pyhrv_sdsd . . . 82

compute_gazepoint_pyhrv_time_domain . . . 83
compute_gazepoint_pyhrv_tinn . . . 83
compute_gazepoint_pyhrv_triangular_index . . . 84
compute_gazepoint_pyhrv_welch_psd . . . 84
compute_gazepoint_quality_index . . . 85
compute_gazepoint_scr_habituation . . . 86
compute_gazepoint_scr_latency . . . 86
compute_gazepoint_signal_band_power . . . 87
compute_gazepoint_signal_correlation . . . 88
compute_gazepoint_signal_lag_matrix . . . 89
compute_gazepoint_signal_phase_locking . . . 90
compute_gazepoint_signal_power_spectrum . . . 90
compute_gazepoint_signal_quality . . . 91
convert_gazepoint_gsr_to_conductance . . . 92
correct_gazepoint_beats . . . 93
correct_gazepoint_eda_temperature . . . 94
correct_gazepoint_ppg_hampel . . . 95
correct_gazepoint_rri_artifacts_local . . . 95
create_gazepoint_analysis_decision_log . . . 96
create_gazepoint_analysis_manifest . . . 97
create_gazepoint_audit_index . . . 98
create_gazepoint_audit_report_section . . . 99
create_gazepoint_biometrics_checklist . . . 99
create_gazepoint_biometrics_feature_inventory . . . 100
create_gazepoint_biometrics_methods_text . . . 101
create_gazepoint_biometrics_report . . . 101
create_gazepoint_biometrics_report_tables . . . 103
create_gazepoint_dictionary . . . 104
create_gazepoint_eda_analysis_pipeline . . . 105
create_gazepoint_eye_methods_text . . . 106
create_gazepoint_heartpy_report . . . 107
create_gazepoint_methods_section . . . 108
create_gazepoint_pipeline_map . . . 109
create_gazepoint_preregistration_checklist . . . 110
create_gazepoint_preregistration_template . . . 110
create_gazepoint_pspm_glm_design . . . 111
create_gazepoint_pyhrv_time_vector . . . 112
create_gazepoint_qc_supplement . . . 113
create_gazepoint_quality_dashboard . . . 113
create_gazepoint_release_checklist . . . 114
create_gazepoint_reproducibility_statement . . . 115
create_gazepoint_sidecar_template . . . 116
create_gazepoint_trial_regressors . . . 116
decompose_gazepoint_eda . . . 118
denoise_gazepoint_eda_autoencoder . . . 119
denoise_gazepoint_eda_wavelet . . . 120
denoise_gazepoint_ppg_autoencoder . . . 121
denoise_gazepoint_quantization_noise . . . 122

detect_active_biometric_channels . . . 123
detect_gazepoint_biometric_schema . . . 123
detect_gazepoint_biometric_timebase . . . 124
detect_gazepoint_blinks . . . 125
detect_gazepoint_doubly_stochastic_changepoints . . . 126
detect_gazepoint_fixations . . . 127
detect_gazepoint_nonwear . . . 129
detect_gazepoint_ppg_onsets . . . 130
detect_gazepoint_ppg_peaks . . . 131
detect_gazepoint_pupil_blinks . . . 132
detect_gazepoint_saccades . . . 134
detect_gazepoint_scr_events . . . 135
detect_gazepoint_scr_peaks . . . 136
detect_gazepoint_time_columns . . . 137
detrend_gazepoint_rri_window . . . 138
detrend_gazepoint_signal . . . 138
diagnose_gazepoint_biometrics_workflow . . . 139
diagnose_gazepoint_cluster_design . . . 140
diagnose_gazepoint_sync_drift . . . 141
downsample_gazepoint_data . . . 142
enhance_gazepoint_ppg_peaks . . . 143
epoch_gazepoint_scr . . . 144
estimate_gazepoint_breathing_rate_from_ibi . . . 145
estimate_gazepoint_cluster_offset . . . 146
estimate_gazepoint_cluster_onset . . . 146
estimate_gazepoint_eda_recovery_times . . . 147
estimate_gazepoint_lsl_clock_offsets . . . 148
estimate_gazepoint_respiration_from_ppg . . . 149
estimate_gazepoint_samplerate_datetime . . . 149
estimate_gazepoint_samplerate_mstimer . . . 150
estimate_gazepoint_signal_lag . . . 151
export_gazepoint_audit_trail_markdown . . . 152
export_gazepoint_biometrics_report_bundle . . . 153
export_gazepoint_cluster_results . . . 154
export_gazepoint_heartpy_input . . . 154
export_gazepoint_mne_cluster_input . . . 155
export_gazepoint_permuco_cluster_input . . . 156
export_gazepoint_permutes_cluster_input . . . 157
export_gazepoint_pipeline_dot . . . 158
export_gazepoint_pspm_model_estimates . . . 159
export_gazepoint_pyhrv_results . . . 159
export_gazepoint_rhrv_input . . . 160
export_gazepoint_to_bids . . . 161
extract_gazepoint_beats_kmeans . . . 165
extract_gazepoint_bilateral_eda_asymmetry . . . 166
extract_gazepoint_eda_complexity . . . 167
extract_gazepoint_eda_events_biosppy_style . . . 168
extract_gazepoint_eda_spectral_power . . . 169

extract_gazepoint_eda_tvsymp . . . 170
extract_gazepoint_edr_pca . . . 171
extract_gazepoint_hrv_asymmetry . . . 172
extract_gazepoint_hrv_features . . . 172
extract_gazepoint_hrv_fragmentation . . . 173
extract_gazepoint_hrv_fuzzy_csi . . . 174
extract_gazepoint_hrv_geometric . . . 175
extract_gazepoint_hrv_nonlinear . . . 176
extract_gazepoint_hrv_rcmse . . . 177
extract_gazepoint_hrv_rqa . . . 178
extract_gazepoint_markerinfo_pspm_style . . . 179
extract_gazepoint_pdr_signals . . . 180
extract_gazepoint_ppg_morphology . . . 181
extract_gazepoint_ppg_templates . . . 182
extract_gazepoint_pyhrv_nn_intervals . . . 183
extract_gazepoint_respiration_ceemdan . . . 183
extract_gazepoint_scr_recovery_times . . . 184
extract_gazepoint_segments_pspm_style . . . 185
extract_gazepoint_ttl_events . . . 186
filter_gazepoint_gaze . . . 187
filter_gazepoint_ibi_implausible . . . 188
filter_gazepoint_ppg_butterworth . . . 189
filter_gazepoint_ppg_signal . . . 190
filter_gazepoint_signal . . . 191
fit_gazepoint_convolution_glm . . . 192
flag_gazepoint_artifacts_svm . . . 192
flag_gazepoint_biometric_dropouts . . . 193
flag_gazepoint_hrv_segments . . . 194
flag_gazepoint_mad_artifacts . . . 195
flag_gazepoint_ppg_quality . . . 196
flag_gazepoint_rr_outliers . . . 197
flag_kleckner_eda_artifacts . . . 198
flip_gazepoint_ppg_signal . . . 199
format_gazepoint_biometrics_feature_inventory . . . 200
fuse_gazepoint_respiration_kalman . . . 200
gazepoint_interoperability_manifest . . . 201
generate_gazepoint_manifest . . . 202
get_gazepoint_plot_data . . . 203
get_gazepoint_plot_settings . . . 203
gpbiometrics_info . . . 204
import_gazepoint_biometric_folder . . . 204
import_gazepoint_biometrics . . . 205
import_gazepoint_data . . . 206
import_gazepoint_data_summary . . . 207
import_gazepoint_event_log . . . 207
import_gazepoint_lsl_xdf . . . 208
import_gazepoint_pyhrv_results . . . 209
impute_gazepoint_missing . . . 209

interpolate_gazepoint_pupil_blinks . . . 210
join_gazepoint_biometrics_to_gp3tools . . . 211
join_gazepoint_biometrics_to_master . . . 212
match_gazepoint_events_to_biometrics . . . 213
merge_gazepoint_recordings_pspm_style . . . 214
model_gazepoint_eda_point_process . . . 214
model_gazepoint_hr_point_process . . . 215
model_gazepoint_hrv_ipfm . . . 216
normalize_gazepoint_scr . . . 217
optimize_gazepoint_cvxeda_tau . . . 218
physiology_validation_ladder . . . 219
pipeline_comparison_dashboard . . . 219
plot_gazepoint_aoi_biometrics . . . 221
plot_gazepoint_biometric_quality . . . 221
plot_gazepoint_biometric_report_dashboard . . . 223
plot_gazepoint_biometric_signals . . . 224
plot_gazepoint_cluster_null_distribution . . . 225
plot_gazepoint_cluster_permutation . . . 226
plot_gazepoint_design_coverage . . . 226
plot_gazepoint_eda_decomposition . . . 227
plot_gazepoint_eda_gram . . . 228
plot_gazepoint_export_profile . . . 229
plot_gazepoint_missingness . . . 229
plot_gazepoint_multimodal_timeline . . . 230
plot_gazepoint_ppg_breathing . . . 232
plot_gazepoint_ppg_peak_detection . . . 232
plot_gazepoint_ppg_poincare . . . 233
plot_gazepoint_ppg_segmentwise . . . 233
plot_gazepoint_pyhrv_hr_heatplot . . . 234
plot_gazepoint_pyhrv_radar_chart . . . 234
plot_gazepoint_pyhrv_tachogram . . . 235
plot_gazepoint_saccade_main_sequence . . . 235
plot_gazepoint_scr_events . . . 236
plot_gazepoint_scr_specification_curve . . . 237
plot_gazepoint_signal_activity . . . 238
plot_gazepoint_signal_quality . . . 239
plot_gazepoint_time_resets . . . 240
ppg_topology_features . . . 240
prepare_gazepoint_aoi_biometrics_model_data . . . 241
prepare_gazepoint_artifact_svm_features . . . 242
prepare_gazepoint_bids_eye . . . 243
prepare_gazepoint_bids_physio . . . 243
prepare_gazepoint_biometrics_lme_data . . . 244
prepare_gazepoint_biosppy_input . . . 245
prepare_gazepoint_ctsi_input . . . 247
prepare_gazepoint_cvxeda_input . . . 249
prepare_gazepoint_eyetrackingr_input . . . 250
prepare_gazepoint_gazer_input . . . 252

prepare_gazepoint_heartpy_input . . . 255
prepare_gazepoint_ledalab_input . . . 256
prepare_gazepoint_mixed_model_data . . . 257
prepare_gazepoint_mne_events . . . 258
prepare_gazepoint_mne_input . . . 260
prepare_gazepoint_multimodal_model_data . . . 261
prepare_gazepoint_neurokit_eda_input . . . 262
prepare_gazepoint_pspm_dcm_input . . . 262
prepare_gazepoint_pspm_input . . . 263
prepare_gazepoint_pupillometryr_input . . . 264
prepare_gazepoint_pyhrv_input . . . 267
prepare_gazepoint_pyppg_input . . . 269
prepare_gazepoint_rhrv_input . . . 270
prepare_gazepoint_scr_hurdle_model_data . . . 271
prepare_gazepoint_timecourse_test_data . . . 272
preprocess_gazepoint_all . . . 274
preprocess_gazepoint_scr_pspm_style . . . 275
process_gazepoint_ppg_heartpy_style . . . 276
process_gazepoint_ppg_segmentwise . . . 277
profile_gazepoint_export_folder . . . 278
recommend_gazepoint_biometric_exclusions . . . 279
reconstruct_gazepoint_ppg_clipping . . . 280
regress_gazepoint_pupil_luminance . . . 281
reject_gazepoint_ppg_peaks . . . 282
remove_gazepoint_ppg_baseline_wander . . . 282
report_gazepoint_cluster_permutation . . . 283
report_gazepoint_data_quality . . . 284
respiration_from_ppg . . . 285
run_gazepoint_automated_statistics . . . 285
run_gazepoint_biometrics_real_data_readiness . . . 286
run_gazepoint_biometrics_workflow . . . 287
run_gazepoint_biosppy_eda . . . 289
run_gazepoint_biosppy_ppg . . . 290
run_gazepoint_cluster_permutation . . . 291
run_gazepoint_cluster_permutation_anova . . . 292
run_gazepoint_cluster_permutation_covariate_adjusted . . . 293
run_gazepoint_cluster_permutation_lmer . . . 293
run_gazepoint_cluster_permutation_parallel . . . 294
run_gazepoint_cluster_threshold_sensitivity . . . 294
run_gazepoint_eda_analysis_pipeline . . . 295
run_gazepoint_heartpy_crosscheck . . . 297
run_gazepoint_multidimensional_cluster_permutation . . . 298
run_gazepoint_neurokit_eda_crosscheck . . . 298
run_gazepoint_online_design_optimization . . . 299
run_gazepoint_pyhrv_style . . . 300
run_gazepoint_real_data_smoke . . . 301
run_gazepoint_scr_multiverse . . . 303
run_gazepoint_scr_threshold_sensitivity . . . 304

run_gazepoint_tfce . . . 306
run_gpbiometrics_shiny . . . 306
run_gpbiometrics_shiny_annotator . . . 307
scale_gazepoint_ppg_sections . . . 307
scale_gazepoint_ppg_signal . . . 308
scr_responsivity_sensitivity . . . 309
screen_gazepoint_eda_nonresponders . . . 309
segment_gazepoint_pyhrv_nni . . . 310
session_info_gazepoint . . . 311
simulate_gazepoint_artifact . . . 312
simulate_gazepoint_biometrics . . . 313
simulate_gazepoint_cluster_timecourse_data . . . 314
simulate_gazepoint_eye_data . . . 315
simulate_gazepoint_multimodal_data . . . 316
smooth_gazepoint_biometrics . . . 316
smooth_gazepoint_ppg_signal . . . 317
smooth_gazepoint_pupil . . . 318
split_gazepoint_sessions_pspm_style . . . 319
standardise_gazepoint_adaptive_ema . . . 319
standardise_gazepoint_biometric_names . . . 321
standardise_gazepoint_plot_contract . . . 322
standardise_gazepoint_range_correction . . . 322
standardise_gazepoint_zscore . . . 323
standardize_gazepoint_biometrics_within_unit . . . 324
standardize_gazepoint_column_names . . . 326
standardize_gazepoint_columns . . . 327
standardize_gazepoint_plot_contracts . . . 327
summarise_gazepoint_aoi_biometrics . . . 328
summarise_gazepoint_biometric_validity . . . 329
summarise_gazepoint_biometrics_feature_inventory . . . 330
summarise_gazepoint_biometrics_workflow . . . 330
summarise_gazepoint_decision_log . . . 331
summarise_gazepoint_dial_windows . . . 331
summarise_gazepoint_engagement_windows . . . 332
summarise_gazepoint_fixations_by_aoi . . . 333
summarise_gazepoint_full_biometric_windows . . . 334
summarise_gazepoint_gsr_tonic_phasic . . . 335
summarise_gazepoint_gsr_windows . . . 336
summarise_gazepoint_hr_windows . . . 337
summarise_gazepoint_hrv_features . . . 337
summarise_gazepoint_ibi_hrv_windows . . . 339
summarise_gazepoint_ibi_windows . . . 340
summarise_gazepoint_multimodal_windows . . . 341
summarise_gazepoint_scr_event_windows . . . 342
summarize_gazepoint_aoi_dwell . . . 344
summarize_gazepoint_audit_trail . . . 344
summarize_gazepoint_beat_corrections . . . 345
summarize_gazepoint_eventlocked_multimodal . . . 346

summarize_gazepoint_export_inventory . . . 347
summarize_gazepoint_feature_coverage . . . 347
summarize_gazepoint_fixations . . . 348
summarize_gazepoint_fixations_by_aoi . . . 349
summarize_gazepoint_missingness . . . 350
summarize_gazepoint_nonwear . . . 351
summarize_gazepoint_preregistration_readiness . . . 351
summarize_gazepoint_pupil_events . . . 352
summarize_gazepoint_qc_overview . . . 353
summarize_gazepoint_scanpath_metrics . . . 353
summarize_gazepoint_scr_recovery . . . 354
summarize_gazepoint_signal_quality . . . 355
summarize_gazepoint_time_clusters . . . 356
summarize_gazepoint_tracking . . . 356
sync_gazepoint_biometrics_with_gaze . . . 357
sync_gazepoint_signals_via_lsl . . . 358
test_gazepoint_hrv_nonlinearity . . . 359
trim_gazepoint_biometrics_pspm_style . . . 360
upsample_gazepoint_data . . . 360
validate_gazepoint_biometrics . . . 361
validate_gazepoint_format . . . 362
validate_gazepoint_gaze . . . 363
validate_gazepoint_metadata . . . 364
write_gazepoint_biometrics_report_tables . . . 365
write_gazepoint_decision_log . . . 366
write_gazepoint_export_profile . . . 366
write_gazepoint_interoperability_audit . . . 367
write_gazepoint_mne_fif . . . 368
write_gazepoint_real_data_smoke . . . 369

Index **370**

`add_gazepoint_decision` *Add a decision to a Gazepoint analysis decision log*

Description

Add one structured record to a decision log created with `create_gazepoint_analysis_decision_log()`.

Usage

```
add_gazepoint_decision(
  log,
  stage,
  object_type,
  object_id = NA_character_,
  decision,
```

```
  reason = NA_character_,
  function_name = NA_character_,
  parameter = NA_character_,
  value = NA_character_,
  reviewer_note = NA_character_,
  timestamp = Sys.time()
)
```

Arguments

log	A decision log created with create_gazepoint_analysis_decision_log().
stage	Workflow stage, for example "import", "quality_control", "preprocessing", "feature_extraction", "modelling", "reporting", or "cluster_permutation".
object_type	Type of object affected by the decision, for example "file", "participant", "trial", "signal", "channel", "event_window", "model", or "cluster_test".
object_id	Identifier for the affected object.
decision	Compact decision label.
reason	Reason for the decision.
function_name	Optional function associated with the decision.
parameter	Optional parameter name.
value	Optional parameter value. Non-scalar values are collapsed into a compact character string.
reviewer_note	Optional reviewer-facing note.
timestamp	Optional timestamp. Defaults to the current time.

Value

Updated decision log.

---

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.

---

align_gazepoint_streams_by_events

Align two Gazepoint streams using matched event markers

Description

Estimates offset or linear drift between a reference stream and a target stream using matched event times, then adds an aligned reference-clock time column to the target stream.

Usage

```
align_gazepoint_streams_by_events(
  reference,
  target,
  reference_events,
  target_events,
  reference_time_col = NULL,
  target_time_col = NULL,
  reference_event_time_col = NULL,
  target_event_time_col = NULL,
  event_id_col = NULL,
  method = c("linear", "offset"),
  include_streams = TRUE
)
```

Arguments

reference
: Reference data frame.

target
: Target data frame to align to the reference clock.

reference_events
: Reference event table or numeric event times.

`target_events` Target event table or numeric event times.

`reference_time_col`
: Reference stream time column.

`target_time_col`
: Target stream time column.

`reference_event_time_col`
: Optional reference-event time column.

`target_event_time_col`
: Optional target-event time column.

`event_id_col` Optional event identifier column used for matching.

`method` `"linear"` estimates offset and drift; `"offset"` estimates a fixed lag only.

`include_streams`
: If TRUE, include reference and aligned target streams in the returned object.

Value

Object of class `gazepoint_stream_alignment`.

`align_gazepoint_to_eeg` *Align Gazepoint timestamps to an EEG clock*

Description

Estimates a constant offset or linear offset-plus-drift mapping from matched Gazepoint and EEG events, then applies that mapping to a Gazepoint stream.

Usage

```
align_gazepoint_to_eeg(
  gazepoint,
  gazepoint_events,
  eeg_events,
  gazepoint_time_col = NULL,
  gazepoint_event_time_col = NULL,
  eeg_event_time_col = NULL,
  eeg_event_sample_col = NULL,
  gazepoint_event_id_col = NULL,
  eeg_event_id_col = NULL,
  gazepoint_time_unit = c("auto", "seconds", "milliseconds", "samples"),
  eeg_time_unit = c("auto", "seconds", "milliseconds", "samples"),
  eeg_sampling_rate_hz = NULL,
  method = c("offset", "linear"),
  match_by = c("auto", "id", "row"),
  robust = TRUE,
  maximum_residual_s = NULL,
```

```
  residual_action = c("error", "allow"),
  output_col = "time_eeg_s"
)
```

Arguments

gazepoint	Gazepoint sample-level data.
gazepoint_events	Gazepoint event table or numeric times.
eeg_events	EEG event table or numeric times.
gazepoint_time_col	Gazepoint stream time column.
gazepoint_event_time_col	Gazepoint event-time column.
eeg_event_time_col	EEG event-time column.
eeg_event_sample_col	Optional EEG event-sample column.
gazepoint_event_id_col	Optional Gazepoint event identifier.
eeg_event_id_col	Optional EEG event identifier.
gazepoint_time_unit	Gazepoint time unit.
eeg_time_unit	EEG event-time unit.
eeg_sampling_rate_hz	Required when EEG events are sample numbers.
method	"offset" or "linear".
match_by	Match events by identifier or row order.
robust	Use one MAD-based residual refit for linear alignment.
maximum_residual_s	Optional maximum permitted absolute residual.
residual_action	Handling when the residual threshold is exceeded.
output_col	Added aligned-time column.

Value

A "gazepoint_eeg_alignment" object.

`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`.

anonymize_gazepoint_data

Pseudonymize participant or record identifiers

Description

Replaces selected identifier columns with deterministic sequential codes. This function supports de-identification workflows, but it does not by itself guarantee anonymity or regulatory compliance.

Usage

```
anonymize_gazepoint_data(
  data,
  id_cols,
  prefix = "P",
  width = 3,
  keep_mapping = TRUE
)
```

Arguments

`data`	A data frame.
`id_cols`	Character vector of identifier columns to replace.
`prefix`	Prefix used in generated codes.
`width`	Numeric width used for zero-padded codes.
`keep_mapping`	Logical. If `TRUE`, store the mapping table as an attribute named `"id_mapping"`.

Value

A data frame with class `gazepoint_anonymized_data`.

assert_gazepoint_columns

Assert that required Gazepoint columns are present

Description

Validates expected columns and either throws an error, raises a warning, or returns a summary table.

Usage

```
assert_gazepoint_columns(
  data,
  required,
  optional = character(),
  mode = c("error", "warning", "summary"),
  ignore_case = TRUE
)
```

Arguments

data	Data frame.
required	Character vector of required columns.
optional	Character vector of optional columns.
mode	"error", "warning", or "summary".
ignore_case	If TRUE, match columns case-insensitively.

Value

Invisibly TRUE for passing checks, or a summary data frame when mode = "summary".

---

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.

---

assess_gazepoint_sampling_irregularity

Assess sampling irregularity in Gazepoint time series

Description

Summarizes median sample interval, effective sampling rate, jitter, repeated timestamps, negative steps, and large gaps overall or by group.

Usage

```
assess_gazepoint_sampling_irregularity(
  data,
  time_col = NULL,
  group_cols = NULL,
  nominal_rate_hz = NULL,
  large_gap_factor = 3
)
```

Arguments

data	Data frame or numeric time vector.
time_col	Time column for data-frame input.
group_cols	Optional grouping columns.
nominal_rate_hz	Optional expected sampling rate.
large_gap_factor	Gap threshold relative to median interval.

Value

Data frame with one row per group.

---

assign_gazepoint_aoi *Assign Gazepoint coordinates to areas of interest*

Description

Assigns sample-level gaze coordinates or fixation centroids to rectangular or polygonal areas of interest (AOIs). The original rows are preserved and augmented with an AOI label, match count, ambiguity flag, and assignment status.

Usage

```
assign_gazepoint_aoi(
  data,
  aois,
  x_col = NULL,
  y_col = NULL,
  aoi_label_col = "aoi",
  format = c("auto", "rectangle", "polygon"),
  aoi_id_col = NULL,
  data_match_cols = NULL,
  aoi_match_cols = data_match_cols,
  xmin_col = "xmin",
  xmax_col = "xmax",
  ymin_col = "ymin",
  ymax_col = "ymax",
  vertex_x_col = "vertex_x",
  vertex_y_col = "vertex_y",
  priority_col = NULL,
  overlap = c("priority", "first", "smallest", "all", "error"),
  boundary = c("inside", "outside"),
  output_col = "AOI",
  match_count_col = "aoi_match_count",
```

```
  ambiguous_col = "aoi_ambiguous",
  status_col = "aoi_assignment_status",
  all_separator = "|",
  overwrite = FALSE
)
```

Arguments

`data`	A data frame containing gaze coordinates or fixation centroids.
`aois`	A data frame containing rectangular or polygonal AOI definitions.
`x_col`	Numeric horizontal-coordinate column in `data`. If `NULL`, common Gazepoint and fixation-summary names are searched.
`y_col`	Numeric vertical-coordinate column in `data`. If `NULL`, common Gazepoint and fixation-summary names are searched.
`aoi_label_col`	AOI label column in `aois`.
`format`	AOI-definition format: `"auto"`, `"rectangle"`, or `"polygon"`.
`aoi_id_col`	Optional AOI identifier column. For polygon definitions, this distinguishes multiple polygons with the same label.
`data_match_cols`	Optional columns in `data` used to restrict eligible AOI definitions, for example `"trial"` or `"MEDIA_ID"`.
`aoi_match_cols`	Corresponding columns in `aois`. Must have the same length and order as `data_match_cols`.
`xmin_col`, `xmax_col`, `ymin_col`, `ymax_col`	Rectangle-boundary columns.
`vertex_x_col`, `vertex_y_col`	Polygon-vertex columns.
`priority_col`	Optional numeric priority column. Smaller values receive higher priority when `overlap = "priority"`.
`overlap`	Rule for coordinates falling within multiple AOIs: `"priority"`, `"first"`, `"smallest"`, `"all"`, or `"error"`.
`boundary`	Should coordinates on AOI boundaries be treated as `"inside"` or `"outside"`?
`output_col`	Name of the generated AOI-label column.
`match_count_col`	Name of the generated AOI-match-count column.
`ambiguous_col`	Name of the generated logical ambiguity column.
`status_col`	Name of the generated assignment-status column.
`all_separator`	Separator used when `overlap = "all"`.
`overwrite`	Logical. If `FALSE`, existing output columns are protected.

Details

Rectangular AOIs use one row per definition with minimum and maximum x and y coordinates. Polygon AOIs use long-format vertices, with one row per vertex and repeated AOI identifiers or labels.

Optional matching columns can restrict AOIs to particular trials, screens, media items, stimuli, or other recording contexts. Missing or blank values in AOI matching columns act as wildcards.

Supported assignment statuses are:

- `"matched"`: exactly one AOI matched;
- `"ambiguous_resolved"`: multiple AOIs matched and one was selected;
- `"ambiguous_all"`: all matching labels were retained;
- `"unmatched"`: valid coordinates did not fall within an eligible AOI;
- `"invalid_coordinate"`: x or y was missing or non-finite.

Coordinate systems are not transformed. Gaze coordinates and AOI definitions must therefore use the same scale, orientation, and origin.

Value

A data frame with class `"gazepoint_aoi_assignment"`. Attributes `aoi_assignment_log`, `aoi_assignment_settings`, and `aoi_definitions` contain structured provenance information.

See Also

`summarize_gazepoint_fixations()`, `summarize_gazepoint_aoi_dwell()`, `summarise_gazepoint_aoi_biometrics()`

Examples

```
gaze <- data.frame(
  gaze_x = c(0.1, 0.5, 0.9),
  gaze_y = c(0.5, 0.5, 0.5)
)

rectangles <- data.frame(
  aoi = c("left", "right"),
  xmin = c(0, 0.6),
  xmax = c(0.4, 1),
  ymin = c(0, 0),
  ymax = c(1, 1)
)

assign_gazepoint_aoi(
  gaze,
  rectangles,
  x_col = "gaze_x",
  y_col = "gaze_y"
)
```

audit_gazepoint_beats *Audit beat or IBI sequences*

Description

Flags short, long, non-finite, duplicate-time, and abrupt-change intervals in beat or inter-beat-interval data. The function is intended for quality control and audit reporting only. It does not remove beats, modify data, or make physiological, psychological, diagnostic, or clinical claims.

Usage

```
audit_gazepoint_beats(
  data,
  ibi_col = NULL,
  beat_time_col = NULL,
  group_cols = NULL,
  min_ibi = 300,
  max_ibi = 2000,
  duplicate_tolerance = 0,
  max_relative_change = NULL
)
```

Arguments

data	A data frame.
ibi_col	Optional numeric inter-beat-interval column. If omitted, intervals are derived from beat_time_col.
beat_time_col	Optional numeric beat-time column. Required when ibi_col is omitted. When supplied, rows are ordered by beat time within each group and duplicate-time checks are enabled.
group_cols	Optional character vector of grouping columns, such as participant, session, trial, or condition.
min_ibi	Minimum plausible interval, in the same units as the IBI column or beat-time column.
max_ibi	Maximum plausible interval, in the same units as the IBI column or beat-time column.
duplicate_tolerance	Maximum adjacent beat-time difference treated as a duplicate-time flag. Ignored when beat_time_col is NULL.
max_relative_change	Optional relative-change threshold for flagging abrupt adjacent IBI changes. If NULL, this check is skipped.

Value

A list with class gazepoint_beat_audit, containing beat-level flags, group summaries, and parameters.

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_biometrics_file` *Audit a Gazepoint biometrics file before analysis*

Description

Provides a single front-door preflight audit for a Gazepoint-style biometric export. The audit standardizes column names when requested, reports likely modality availability, schema status, timestamp irregularity, missingness, duplicate rows, and reviewer-facing warnings.

Usage

```
audit_gazepoint_biometrics_file(
  path = NULL,
  data = NULL,
 expected_modalities = c("time", "eda", "ppg", "hr", "ibi", "pupil", "gaze", "events"),
  time_col = NULL,
  standardize = TRUE,
  include_data = FALSE,
  long_gap_s = NULL
)

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

## S3 method for class 'gazepoint_biometrics_audit'
summary(object, ...)
```

Arguments

path	Optional CSV/TSV file path.
data	Optional data frame. If supplied, it is used instead of path.
expected_modalities	Modalities expected in the export.
time_col	Optional time column. If omitted, a common Gazepoint time column is guessed after optional standardization.
standardize	If TRUE, apply standardize_gazepoint_column_names() before auditing.
include_data	If TRUE, include the standardized data in the returned audit object.
long_gap_s	Optional missing-gap threshold passed to summarize_gazepoint_missingness().
x	Object of class gazepoint_biometrics_audit for the print method.
...	Additional arguments currently ignored.
object	Object of class gazepoint_biometrics_audit for the summary method.

Value

An object of class gazepoint_biometrics_audit.

---

audit_gazepoint_condition_balance

Audit Gazepoint condition balance

Description

Summarise condition balance across participants and trials before modelling, event-window summaries, or two-condition time-course workflows.

Usage

```
audit_gazepoint_condition_balance(
  data,
  participant_col,
  condition_col,
  trial_col = NULL,
  expected_conditions = NULL
)
```

Arguments

data	A data frame containing trial-, event-, or sample-level records.
participant_col	Name of the participant identifier column.
condition_col	Name of the condition column.
trial_col	Optional trial identifier column.
expected_conditions	Optional character vector of expected conditions.

Value

A list with class "gazepoint_condition_balance_audit".

---

audit_gazepoint_dataset_structure

Audit Gazepoint dataset structure

Description

Checks a dataset or export folder for expected directories, files, filename patterns, duplicate names, empty files, and optional sidecar coverage. This is a lightweight structure audit, not a full BIDS validator or converter.

Usage

```
audit_gazepoint_dataset_structure(
  root,
  expected_dirs = NULL,
  expected_files = NULL,
  expected_patterns = NULL,
  allowed_extensions = c("csv", "tsv", "txt", "json", "xlsx", "rds"),
  require_sidecars = FALSE,
  recursive = TRUE,
  include_hidden = FALSE
)
```

Arguments

`root`	Dataset or export root directory.
`expected_dirs`	Optional character vector of directory paths expected under `root`.
`expected_files`	Optional character vector of file paths expected under `root`.
`expected_patterns`	Optional named or unnamed character vector of regular expressions that should match at least one relative file path.
`allowed_extensions`	Optional character vector of allowed file extensions, without leading dots.
`require_sidecars`	Logical. If `TRUE`, non-sidecar files are flagged when no same-stem `.json` sidecar exists.
`recursive`	Logical. If `TRUE`, files are inventoried recursively.
`include_hidden`	Logical. If `TRUE`, hidden files are included.

Value

A list with class `gazepoint_dataset_structure_audit`.

`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_event_coverage

Audit Gazepoint event coverage

Description

Check whether expected events, TTL markers, AOI events, or task events are present across participant/trial/condition units.

Usage

```
audit_gazepoint_event_coverage(
  data,
  event_col,
  participant_col = NULL,
  trial_col = NULL,
  condition_col = NULL,
  expected_events = NULL,
  unit_cols = NULL
)
```

Arguments

data	A data frame containing event-, trial-, or sample-level records.
event_col	Name of the event/marker column.
participant_col	Optional participant identifier column.
trial_col	Optional trial identifier column.
condition_col	Optional condition column.
expected_events	Optional character vector of expected event labels.
unit_cols	Optional character vector of columns defining event-coverage units. If supplied, this overrides participant_col, trial_col, and condition_col for unit construction.

Value

A list with class "gazepoint_event_coverage_audit".

---

audit_gazepoint_experiment_design

Audit Gazepoint experiment design structure

Description

Check whether an imported or prepared Gazepoint workflow table contains the expected participant, trial, condition, and optional session structure before modelling, event-locking, or time-course analysis.

Usage

```
audit_gazepoint_experiment_design(
  data,
  participant_col,
  trial_col = NULL,
  condition_col = NULL,
  session_col = NULL,
  expected_conditions = NULL,
  min_trials_per_condition = 1L
)
```

Arguments

data	A data frame containing trial-, event-, or sample-level records.
participant_col	Name of the participant identifier column.
trial_col	Optional name of the trial identifier column.
condition_col	Optional name of the condition column.
session_col	Optional name of the session/block column.
expected_conditions	Optional character vector of expected conditions.
min_trials_per_condition	Minimum acceptable number of trials per participant-condition cell when trial_col and condition_col are supplied.

Value

A list with class "gazepoint_experiment_design_audit".

Examples

```
dat <- data.frame(
  participant = rep(paste0("P", 1:4), each = 4),
  trial = rep(1:4, times = 4),
  condition = rep(c("A", "B"), times = 8)
)

audit_gazepoint_experiment_design(
  dat,
  participant_col = "participant",
  trial_col = "trial",
  condition_col = "condition",
  expected_conditions = c("A", "B")
)
```

`audit_gazepoint_export_schema` *Audit a Gazepoint export schema*

Description

Reports whether expected Gazepoint-style roles are present, missing, or ambiguous, using the same alias dictionary as `standardize_gazepoint_column_names()`.

Usage

```
audit_gazepoint_export_schema(
  data,
  expected_roles = NULL,
  dictionary = NULL,
  strict = FALSE
)
```

Arguments

`data`	Data frame or CSV/TSV path.
`expected_roles`	Optional character vector of expected canonical roles.
`dictionary`	Optional alias dictionary.
`strict`	If TRUE, error when required roles are missing.

Value

Data frame with one row per expected role.

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_interoperability_versions

Audit external interoperability versions

Description

Checks whether declared bridge functions are exported and records the installed versions of optional R and Python dependencies. Missing optional dependencies are recorded for review rather than treated as core-package failures.

Usage

```
audit_gazepoint_interoperability_versions(
  manifest = gazepoint_interoperability_manifest(),
  include_python = TRUE,
  strict = FALSE
)
```

Arguments

manifest	Interoperability manifest. Defaults to gazepoint_interoperability_manifest().
include_python	Logical. Inspect the active Python environment through the optional reticulate package.
strict	Logical. Stop when a required contract fails, a declared bridge function is missing, or an installed dependency is below its minimum tested version.

Details

Python package versions are queried through `importlib.metadata`; target modules are not imported merely to determine their versions.

Value

An object of class `"gazepoint_interoperability_audit"` containing:

- `results`: target-level compatibility findings;
- `summary`: aggregate pass and review counts;
- `session`: R, Python, operating-system, package, and timestamp metadata;
- `manifest`: the manifest used for the audit.

Examples

```
audit <- audit_gazepoint_interoperability_versions(
  include_python = FALSE
)
audit$summary
```

---

audit_gazepoint_pipeline_steps

Audit Gazepoint pipeline steps

Description

Checks a pipeline map for missing expected steps, duplicate step identifiers, edge references to missing nodes, and simple ordering violations. The audit is descriptive and does not evaluate whether a scientific analysis is valid.

Usage

```
audit_gazepoint_pipeline_steps(
  pipeline,
  expected_steps = NULL,
  required_order = NULL,
  allow_extra = TRUE
)
```

Arguments

`pipeline`	A `gazepoint_pipeline_map` object or a data frame of pipeline steps.
`expected_steps`	Optional character vector of step identifiers expected in the pipeline. If `NULL`, required steps in the pipeline map are used.
`required_order`	Optional character vector describing the expected order of selected steps. If `NULL`, the order implied by `expected_order` is used when available.
`allow_extra`	Logical. If `FALSE`, steps not listed in `expected_steps` are reported as warnings.

Value

A list with class `gazepoint_pipeline_audit`.

`audit_gazepoint_preregistration_consistency`
Audit preregistration checklist consistency

Description

Checks whether a preregistration checklist has corresponding evidence objects and whether those evidence objects contain expected fields. The audit is a documentation-readiness check only; it does not judge study quality or perform automatic exclusion.

Usage

```
audit_gazepoint_preregistration_consistency(
  checklist = NULL,
  evidence = list(),
  require_required_fields = TRUE
)
```

Arguments

`checklist`	A checklist produced by `create_gazepoint_preregistration_checklist()`, or `NULL` to use the default checklist.
`evidence`	A named list of evidence objects. Each checklist row uses `evidence_key` to look up an object in this list. Evidence objects can be data frames, named lists, character vectors, logical values, or other non-empty objects.
`require_required_fields`	Logical. If `TRUE`, listed `required_fields` must be present in data-frame or named-list evidence.

Value

A list with class `gazepoint_preregistration_audit`.

audit_gazepoint_pupil_luminance

Audit pupil-luminance confounding

Description

Computes correlations between pupil size and luminance/brightness and flags groups where the absolute association exceeds a threshold.

Usage

```
audit_gazepoint_pupil_luminance(
  data,
  pupil_col = NULL,
  luminance_col = NULL,
  group_cols = NULL,
  threshold = 0.3,
  method = c("pearson", "spearman")
)
```

Arguments

data	Data frame containing pupil and luminance columns.
pupil_col	Pupil column.
luminance_col	Luminance/brightness column.
group_cols	Optional grouping columns.
threshold	Absolute correlation threshold for flagging.
method	Correlation method.

Value

Data frame with correlation and flag columns.

audit_gazepoint_release_readiness

Audit gpbiometrics release readiness

Description

Performs conservative package-level checks for release preparation. The audit inspects package files, exported functions, manual pages, tests, pkgdown reference pages, vignettes/articles, and optional roadmap terms. It does not inspect participant-level data and does not make biometric, psychological, clinical, or diagnostic interpretations.

Usage

```
audit_gazepoint_release_readiness(
  path = ".",
  required_files = c("DESCRIPTION", "NAMESPACE", "R", "man", "tests/testthat",
    "_pkgdown.yml"),
  expected_exports = NULL,
  roadmap_terms = NULL,
  require_pkgdown = TRUE
)
```

Arguments

path	Package root.
required_files	Character vector of required package paths.
expected_exports	Optional character vector of exported function names expected to be present in NAMESPACE.
roadmap_terms	Optional character vector of roadmap terms or helper names to check across R files, tests, documentation, and pkgdown output.
require_pkgdown	Logical. If TRUE, missing pkgdown reference pages for exported functions are warnings. If FALSE, they are recorded as not-checked.

Value

A list with checks, overview, exports, feature coverage, release checklist, and settings. The object has class gazepoint_release_readiness_audit.

---

audit_gazepoint_session_comparability

Audit session comparability across QC metrics

Description

Aggregates selected QC metrics by session, participant, trial, or another grouping unit, then flags unusual values using transparent z-score and/or IQR rules. The function is intended for audit reporting only and does not imply automatic exclusion.

Usage

```
audit_gazepoint_session_comparability(
  data,
  metric_cols,
  group_cols = NULL,
  method = c("both", "z", "iqr"),
  z_threshold = 2,
  iqr_multiplier = 1.5
)
```

Arguments

data	A data frame.
metric_cols	Character vector of numeric QC metric columns.
group_cols	Optional character vector defining the session or analysis unit to compare.
method	Outlier rule: "z", "iqr", or "both".
z_threshold	Absolute z-score threshold.
iqr_multiplier	IQR multiplier used for Tukey-style fences.

Value

A list with class gazepoint_session_comparability_audit.

---

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_smoke_privacy`
Audit a smoke-test result for private information

Description

Checks that a smoke-test result contains no private paths, source filenames, participant-level identifier columns, or retained raw workflow objects.

Usage

```
audit_gazepoint_smoke_privacy(x, private_values = NULL)
```

Arguments

`x`	Object returned by `run_gazepoint_real_data_smoke()`.
`private_values`	Optional private path strings that must not occur in the object.

Value

A data frame of privacy checks and pass/fail statuses.

`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`.

---

audit_gazepoint_timecourse_grid

Audit the participant-condition-time grid for cluster-permutation analysis

---

Description

Checks whether a time-course data set has a complete participant by condition by time grid. This is intended as a safety diagnostic before calling `run_gazepoint_cluster_permutation()`.

Usage

```
audit_gazepoint_timecourse_grid(
  data,
  subject,
  condition,
  time,
  value = NULL,
  max_report_cells = 1000L
)
```

Arguments

`data` A data frame.

`subject` Participant identifier column, supplied as a bare column name or a single string.

`condition` Condition column, supplied as a bare column name or a single string.

`time` Time-bin column, supplied as a bare column name or a single string.

`value` Optional numeric outcome column used to count missing outcome values.

`max_report_cells`
Maximum number of missing or duplicate cells stored in the returned object.

Value

An object of class `gazepoint_timecourse_grid_audit`.

`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.

`build_gazepoint_aoi_timecourse` *Build a tidy AOI time course*

Description

Converts gaze or fixation samples into binned AOI proportions by participant, trial, condition, or other grouping columns. AOIs can be supplied as labels or derived from rectangular AOI definitions.

Usage

```
build_gazepoint_aoi_timecourse(
  data,
  time_col = NULL,
  aoi_col = NULL,
  x_col = NULL,
  y_col = NULL,
  aoi_definitions = NULL,
  group_cols = NULL,
  bin_width_s = 0.1,
  valid_col = NULL,
  include_empty = TRUE
)
```

Arguments

`data`	Gaze or fixation data frame.
`time_col`	Time column.
`aoi_col`	Optional AOI label column.
`x_col`	Optional x-coordinate column.
`y_col`	Optional y-coordinate column.
`aoi_definitions`	Optional data frame with `AOI`, `xmin`, `xmax`, `ymin`, and `ymax`.
`group_cols`	Optional grouping columns.
`bin_width_s`	Bin width in seconds.
`valid_col`	Optional validity column.
`include_empty`	If TRUE, include AOI/bin combinations with zero samples.

Value

Tidy AOI time-course data frame.

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_bids` *Check a Gazepoint dataset for a BIDS-like layout*

Description

Performs a conservative, Gazepoint-oriented audit of a dataset folder that follows, or approximately follows, a BIDS-like organization. This function is not a full BIDS validator and does not convert data. It checks only simple, reviewable layout features such as dataset-level metadata files, subject folders, Gazepoint-like export files, and optional JSON sidecars.

Usage

```
check_gazepoint_bids(
  root,
  subject_pattern = "^sub-[A-Za-z0-9]+$",
  recursive = TRUE,
  expected_files = c("dataset_description.json", "participants.tsv"),
 gazepoint_patterns = c("all[_-]?gaze", "fixation", "summary", "biometric", "eda",
    "gsr", "ecg", "ppg", "hr", "ibi")
)
```

Arguments

`root`	Character scalar. Path to the dataset root folder.
`subject_pattern`	Regular expression used to identify subject folders. Defaults to folders such as `sub-001`.
`recursive`	Logical. Whether to search for Gazepoint export files recursively under `root`.
`expected_files`	Character vector of optional dataset-level files to check.
`gazepoint_patterns`	Character vector of case-insensitive filename patterns used to identify Gazepoint-derived exports.

Value

A list with class `gazepoint_bids_layout_audit` containing the root path, check table, discovered files, and summary counts.

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`.

`check_gazepoint_ppg_binary_quality` *Check binary quality of Gazepoint pulse/PPG analysis results*

Description

Check binary quality of Gazepoint pulse/PPG analysis results

Usage

```
check_gazepoint_ppg_binary_quality(
  measures = NULL,
  peaks = NULL,
  min_peaks = 5L,
  bpm_range = c(40, 180),
  max_missing_prop = 0.25
)
```

Arguments

`measures`	Optional measures table.
`peaks`	Optional peaks table.
`min_peaks`	Minimum accepted peaks.
`bpm_range`	Plausible BPM range.
`max_missing_prop`	Maximum missing proportion, if available.

Value

Data frame with binary quality status.

check_gazepoint_pyhrv_interval

Check whether NN intervals are inside a plausible interval

Description

Check whether NN intervals are inside a plausible interval

Usage

```
check_gazepoint_pyhrv_interval(nni_ms, min_ms = 250, max_ms = 2500)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
min_ms	Minimum plausible interval.
max_ms	Maximum plausible interval.

Value

Data frame with interval status.

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.

`classify_gazepoint_signal_quality` *Classify signal-quality rows using transparent threshold rules*

Description

Applies user-visible threshold rules to a signal-quality table. The function adds review labels and failing-rule descriptions. It does not remove data.

Usage

```
classify_gazepoint_signal_quality(quality, rules = NULL)
```

Arguments

quality	A data frame returned by `compute_gazepoint_signal_quality()`.
rules	Optional named list of threshold rules. Missing rules use the conservative defaults. Set a rule to `NULL` to remove it.

Value

A data frame with class `gazepoint_signal_quality_classification`.

`clean_gazepoint_pupil` *Clean Gazepoint pupil data*

Description

Compatibility helper for users searching for a short pupil-cleaning function. By default it applies transparent blink/dropout interpolation using `interpolate_gazepoint_pupil_blinks()`. Set `prefer_existing = TRUE` to delegate to `clean_gazepoint_pupil_signal()` when compatible with the supplied arguments.

Usage

```
clean_gazepoint_pupil(
  data,
  pupil_cols = NULL,
  time_col = NULL,
  blink_col = NULL,
  max_gap_s = NULL,
  method = c("linear", "constant"),
  suffix = "_clean",
  prefer_existing = FALSE,
  ...
)
```

Arguments

data	Data frame containing pupil columns.
pupil_cols	Pupil columns to clean. If omitted, common pupil columns are detected.
time_col	Optional time column.
blink_col	Optional logical/numeric blink mask column.
max_gap_s	Optional maximum interpolated gap duration in seconds.
method	Interpolation method passed to stats::approx().
suffix	Suffix for cleaned pupil columns.
prefer_existing	If TRUE, first try clean_gazepoint_pupil_signal().
...	Additional arguments passed to the preferred existing cleaner when prefer_existing = TRUE.

Value

Data frame with cleaned pupil columns and interpolation flags.

---

clean_gazepoint_pupil_signal

Clean Gazepoint pupil signal

Description

Flags blink samples, non-positive pupil values, and robust outlier spikes, then interpolates short missing segments. The function returns the original data with cleaned pupil columns and provenance flags.

Usage

```
clean_gazepoint_pupil_signal(
  data,
  pupil_cols = NULL,
  time_col = NULL,
  group_cols = NULL,
  validity_cols = NULL,
  method = c("linear", "locf", "nocb", "nearest", "constant"),
  max_gap = Inf,
  spike_mad = 6,
  combine = c("all", "any"),
  min_blink_samples = 1L,
  suffix = "_clean",
  keep_flags = TRUE
)
```

Arguments

data	Eye-tracking data frame.
pupil_cols	Pupil columns. If NULL, common Gazepoint pupil columns are detected automatically.
time_col	Optional time column.
group_cols	Optional grouping columns.
validity_cols	Optional validity columns corresponding to pupil columns.
method	Imputation method passed to impute_gazepoint_missing().
max_gap	Maximum missing run length, in samples, to interpolate.
spike_mad	Robust MAD threshold for pupil outlier spikes.
combine	Blink rule passed to detect_gazepoint_pupil_blinks().
min_blink_samples	Minimum blink run length in samples.
suffix	Suffix for cleaned pupil columns.
keep_flags	If TRUE, add blink/spike/imputation flag columns.

Value

Data frame with cleaned pupil columns and a pupil_cleaning_summary attribute.

Examples

```
dat <- data.frame(time_s = 0:4, LPD = c(3, NA, 3.2, 40, 3.1))
clean_gazepoint_pupil_signal(dat, pupil_cols = "LPD", time_col = "time_s")
```

clean_gazepoint_rr_intervals *Clean RR or IBI intervals using HeartPy-style methods*

Description

Clean RR or IBI intervals using HeartPy-style methods

Usage

```
clean_gazepoint_rr_intervals(
  rr_ms,
  method = c("quotient", "iqr", "modified_z", "zscore", "none"),
  group_col = "group",
  quotient_threshold = 0.2,
  iqr_multiplier = 1.5,
  z_threshold = 3.5
)
```

Arguments

rr_ms	Numeric RR/IBI intervals in milliseconds, or a peak table.
method	Cleaning method: quotient, iqr, modified_z, zscore, or none.
group_col	Group column when rr_ms is a peak table.
quotient_threshold	Maximum allowed ratio between adjacent intervals.
iqr_multiplier	IQR multiplier for IQR cleaning.
z_threshold	Z-score threshold.

Value

Cleaned interval table, or cleaned peak table when a peak table is supplied.

combine_gazepoint_marker_channels_pspm_style *Combine multiple Gazepoint marker channels PsPM-style*

Description

Combine multiple Gazepoint marker channels PsPM-style

Usage

```
combine_gazepoint_marker_channels_pspm_style(
  data,
  marker_cols = NULL,
  time_col = NULL,
  sampling_rate_hz = NULL,
  group_cols = NULL,
  combined_col = "pspm_marker"
)
```

Arguments

data	Gazepoint data frame.
marker_cols	Marker columns to combine.
time_col	Time column.
sampling_rate_hz	Sampling rate if time column is absent.
group_cols	Optional grouping columns.
combined_col	Name of combined marker column.

Value

List with combined data and marker table.

---

compare_gazepoint_conditions_bootstrap

Bootstrap condition comparisons for Gazepoint-derived outcomes

---

Description

Compares two conditions using a transparent percentile bootstrap. The helper is intended for trial-level, participant-level, or event-locked summaries produced by gpbiometrics. When participant_col is supplied, observations are first averaged at the participant-by-condition level to reduce pseudo-replication. When paired = TRUE, only participants with both conditions are retained and resampled as paired units.

Usage

```
compare_gazepoint_conditions_bootstrap(
  data,
  outcome_col,
  condition_col,
  participant_col = NULL,
  condition_levels = NULL,
  paired = FALSE,
```

```
  by_cols = NULL,
 statistic = c("mean_difference", "median_difference", "standardized_mean_difference"),
  n_boot = 2000,
  conf_level = 0.95,
  seed = NULL,
  na_rm = TRUE
)
```

Arguments

data	Data frame containing the outcome and condition columns.
outcome_col	Numeric outcome column.
condition_col	Two-level condition column.
participant_col	Optional participant/unit identifier. If supplied, the bootstrap uses participant-by-condition means.
condition_levels	Optional two-element character vector defining the reference and target condition. The estimate is condition_levels[2] - condition_levels[1].
paired	If TRUE, perform a paired participant-level bootstrap. Requires participant_col.
by_cols	Optional columns used to run separate comparisons by subgroup.
statistic	Statistic to bootstrap: "mean_difference", "median_difference", or "standardized_mean_differer
n_boot	Number of bootstrap resamples.
conf_level	Confidence level for percentile intervals.
seed	Optional random seed.
na_rm	If TRUE, remove rows with missing/non-finite outcomes or missing condition labels.

Value

Object of class gazepoint_bootstrap_condition_comparison, stored as a data frame. Bootstrap samples are stored in the bootstrap_samples attribute.

---

compare_gazepoint_export_profiles

Compare Gazepoint export-folder profiles

Description

Compare two or more objects created by profile_gazepoint_export_folder(). The comparison reports folder-level dimensions and column-role coverage across profiles.

Usage

```
compare_gazepoint_export_profiles(..., labels = NULL)
```

Arguments

...	Profile objects returned by `profile_gazepoint_export_folder()`, or a single list of profile objects.
labels	Optional labels for the profiles.

Value

A list with class "gazepoint_export_profile_comparison".

---

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`.

compare_gazepoint_pyhrv_psd_methods *Compare pyHRV-style PSD methods*

Description

Compare pyHRV-style PSD methods

Usage

```
compare_gazepoint_pyhrv_psd_methods(
  nni_ms,
  time_s = NULL,
  methods = c("welch", "lomb", "ar"),
  plot = FALSE
)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
time_s	Optional time vector in seconds.
methods	Methods to compare.
plot	If TRUE, draw a comparison plot.

Value

List with method outputs and combined measures.

compare_hrv_prv_devices *Compare ECG-HRV and PPG-PRV devices for one derived metric*

Description

Computes ICC(A,1), Lin's concordance correlation coefficient, and Bland-Altman bias/95% limits. Agreement is deliberately scoped to the supplied metric and acquisition configuration; it is not a global ECG/PPG equivalence decision.

Usage

```
compare_hrv_prv_devices(
  reference,
  candidate,
  metric,
  reference_source = "ECG-HRV",
  candidate_source = "PPG-PRV",
  reference_site = NA_character_,
  candidate_site = NA_character_
)
```

Arguments

`reference`, `candidate`	Paired numeric values.
`metric`	Name of the derived metric (for example `"RMSSD"`).
`reference_source`, `candidate_source`	Signal-source labels.
`reference_site`, `candidate_site`	Optional sensor-site labels.

Value

A metric-specific agreement record.

`compute_gazepoint_engagement_index` *Compute engagement-dial summary indices*

Description

Summarizes a continuous engagement-dial signal, typically scaled from 0 to 100, into interpretable behavioral indices such as mean engagement, percent time above threshold, volatility, and area under the curve.

Usage

```
compute_gazepoint_engagement_index(
  dial,
  time = NULL,
  threshold = 50,
  group = NULL,
  return = c("data", "scalar")
)
```

Arguments

dial	Numeric engagement-dial values.
time	Optional time vector.
threshold	Engagement threshold.
group	Optional grouping vector for grouped summaries.
return	"data" for a one-row data frame per group, or "scalar" for percent time above threshold.

Value

Data frame of engagement metrics or a scalar.

Examples

```
compute_gazepoint_engagement_index(c(20, 60, 80), time = 1:3)
```

---

compute_gazepoint_hrv_wavelet_psd

Compute a simple Haar-style HRV wavelet PSD summary

Description

Computes a conservative Haar-style multiscale power summary for RR/NN intervals. This is intended as a lightweight, CRAN-safe exploratory nonstationary-HRV helper, not as a replacement for specialist HRV packages.

Usage

```
compute_gazepoint_hrv_wavelet_psd(
  rr_intervals,
  time = NULL,
  bands = list(vlf = c(0.0033, 0.04), lf = c(0.04, 0.15), hf = c(0.15, 0.4)),
  max_scale = NULL
)
```

Arguments

rr_intervals	Numeric RR/NN intervals in milliseconds or seconds.
time	Optional timestamps for intervals.
bands	Named list of frequency bands in Hz.
max_scale	Optional maximum Haar scale in beats.

Value

List with psd, band_power, and settings.

`compute_gazepoint_ppg_frequency_measures` *Compute Gazepoint pulse/PPG frequency-domain measures*

Description

Compute Gazepoint pulse/PPG frequency-domain measures

Usage

```
compute_gazepoint_ppg_frequency_measures(
  peaks = NULL,
  rr_ms = NULL,
  rr_time_s = NULL,
  group_col = "group",
  method = c("welch", "fft", "periodogram"),
  resample_hz = 4,
  bands = list(lf = c(0.05, 0.15), hf = c(0.15, 0.5)),
  welch_window_seconds = 64,
  welch_overlap = 0.5
)
```

Arguments

`peaks`	Optional peak table.
`rr_ms`	Optional RR/IBI intervals in milliseconds.
`rr_time_s`	Optional interval timestamps.
`group_col`	Group column when peaks are supplied.
`method`	PSD method: fft, periodogram, or welch.
`resample_hz`	RR interpolation frequency.
`bands`	Named list of frequency bands.
`welch_window_seconds`	Welch window length.
`welch_overlap`	Welch overlap proportion.

Value

Data frame of frequency-domain measures.

compute_gazepoint_ppg_measures

Compute HeartPy-style pulse/PPG measures

Description

Compute HeartPy-style pulse/PPG measures

Usage

```
compute_gazepoint_ppg_measures(peaks, group_col = "group")
```

Arguments

peaks	Peak table from detect_gazepoint_ppg_peaks() or reject_gazepoint_ppg_peaks().
group_col	Group column.

Value

Data frame with BPM, IBI, SDNN, SDSD, RMSSD, pNN20, pNN50, MAD, LF, HF, HF/LF, and breathing rate.

compute_gazepoint_ppg_template_similarity

Compute PPG beat-template similarity

Description

Extracts peak-centered PPG windows, builds a median beat template, and computes per-beat correlation with the template as an interpretable quality metric.

Usage

```
compute_gazepoint_ppg_template_similarity(
  data,
  time_col = NULL,
  ppg_col = NULL,
  peaks = NULL,
  window_s = c(-0.25, 0.45),
  sampling_rate_hz = NULL,
  n_grid = 101,
  similarity_threshold = 0.8
)
```

Arguments

data	PPG data frame or numeric PPG vector.
time_col	Time column for data-frame input.
ppg_col	PPG/BVP signal column for data-frame input.
peaks	Optional peak indices or peak times.
window_s	Two-element window around each peak in seconds.
sampling_rate_hz	Sampling rate for vector input or when time is absent.
n_grid	Number of points in the normalized beat template.
similarity_threshold	Correlation threshold for quality_ok.

Value

List with beats, template, summary, and settings.

---

compute_gazepoint_pyhrv_ar_psd

Compute pyHRV-style autoregressive PSD

Description

Compute pyHRV-style autoregressive PSD

Usage

```
compute_gazepoint_pyhrv_ar_psd(
  nni_ms,
  time_s = NULL,
  resample_hz = 4,
  order = NULL
)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
time_s	Optional time vector in seconds.
resample_hz	Resampling frequency.
order	Optional AR order.

Value

List with PSD and measures.

`compute_gazepoint_pyhrv_dfa` *Compute detrended fluctuation analysis*

Description

Compute detrended fluctuation analysis

Usage

```
compute_gazepoint_pyhrv_dfa(
  nni_ms,
  scales = unique(round(exp(seq(log(4), log(64), length.out = 12))))
)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`scales`	Window sizes in beats.

Value

Data frame with DFA alpha estimates.

`compute_gazepoint_pyhrv_frequency_domain` *Compute pyHRV-style frequency-domain summary*

Description

Compute pyHRV-style frequency-domain summary

Usage

```
compute_gazepoint_pyhrv_frequency_domain(
  nni_ms,
  time_s = NULL,
  method = c("welch", "lomb", "ar")
)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`time_s`	Optional time vector in seconds.
`method`	Frequency method: welch, lomb, or ar.

Value

List with PSD and measures.

`compute_gazepoint_pyhrv_heart_rate`
Compute heart rate from NN intervals

Description

Compute heart rate from NN intervals

Usage

```
compute_gazepoint_pyhrv_heart_rate(nni_ms)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.

Value

Numeric heart rate in beats per minute.

`compute_gazepoint_pyhrv_hr_parameters`
Compute pyHRV-style heart-rate parameters

Description

Compute pyHRV-style heart-rate parameters

Usage

```
compute_gazepoint_pyhrv_hr_parameters(nni_ms)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.

Value

Data frame of heart-rate parameters.

`compute_gazepoint_pyhrv_lomb_psd` *Compute pyHRV-style Lomb PSD*

Description

Compute pyHRV-style Lomb PSD

Usage

```
compute_gazepoint_pyhrv_lomb_psd(
  nni_ms,
  time_s = NULL,
  min_hz = 0.003,
  max_hz = 0.4,
  n_freq = 512
)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`time_s`	Optional time vector in seconds.
`min_hz`	Minimum frequency.
`max_hz`	Maximum frequency.
`n_freq`	Number of frequencies.

Value

List with PSD and measures.

`compute_gazepoint_pyhrv_nn_diff` *Compute NN interval differences*

Description

Compute NN interval differences

Usage

```
compute_gazepoint_pyhrv_nn_diff(nni_ms, absolute = FALSE)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
absolute	If TRUE, return absolute differences.

Value

Numeric successive NN differences in milliseconds.

---

compute_gazepoint_pyhrv_nn20

Compute NN20 and pNN20

Description

Compute NN20 and pNN20

Usage

```
compute_gazepoint_pyhrv_nn20(nni_ms)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.

Value

Data frame with NN20 and pNN20.

---

compute_gazepoint_pyhrv_nn50

Compute NN50 and pNN50

Description

Compute NN50 and pNN50

Usage

```
compute_gazepoint_pyhrv_nn50(nni_ms)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.

Value

Data frame with NN50 and pNN50.

compute_gazepoint_pyhrv_nni_differences_parameters

Compute pyHRV-style NNI difference parameters

Description

Compute pyHRV-style NNI difference parameters

Usage

```
compute_gazepoint_pyhrv_nni_differences_parameters(nni_ms)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.

Value

Data frame of successive-difference parameters.

compute_gazepoint_pyhrv_nni_parameters

Compute pyHRV-style NNI parameters

Description

Compute pyHRV-style NNI parameters

Usage

```
compute_gazepoint_pyhrv_nni_parameters(nni_ms)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.

Value

Data frame of NNI parameters.

`compute_gazepoint_pyhrv_nnxx`
Compute NNxx and pNNxx

Description

Compute NNxx and pNNxx

Usage

```
compute_gazepoint_pyhrv_nnxx(nni_ms, threshold_ms = 50)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`threshold_ms`	Threshold in milliseconds.

Value

Data frame with NNxx and pNNxx.

`compute_gazepoint_pyhrv_nonlinear`
Compute pyHRV-style nonlinear summary

Description

Compute pyHRV-style nonlinear summary

Usage

```
compute_gazepoint_pyhrv_nonlinear(nni_ms)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.

Value

Data frame with nonlinear measures.

compute_gazepoint_pyhrv_poincare

Compute pyHRV-style Poincare measures

Description

Compute pyHRV-style Poincare measures

Usage

```
compute_gazepoint_pyhrv_poincare(nni_ms, plot = FALSE)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
plot	If TRUE, draw Poincare plot.

Value

Data frame of Poincare measures.

compute_gazepoint_pyhrv_psd_waterfall

Compute pyHRV-style PSD waterfall over segments

Description

Compute pyHRV-style PSD waterfall over segments

Usage

```
compute_gazepoint_pyhrv_psd_waterfall(
  nni_ms,
  segment_seconds = 300,
  method = c("welch", "lomb", "ar"),
  plot = FALSE
)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
segment_seconds	Segment length.
method	Frequency method.
plot	If TRUE, draw a heatmap-style waterfall.

Value

List with PSD grid and segment measures.

`compute_gazepoint_pyhrv_rmssd` *Compute RMSSD*

Description

Compute RMSSD

Usage

```
compute_gazepoint_pyhrv_rmssd(nni_ms)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.

Value

RMSSD in milliseconds.

`compute_gazepoint_pyhrv_sample_entropy` *Compute sample entropy*

Description

Compute sample entropy

Usage

```
compute_gazepoint_pyhrv_sample_entropy(nni_ms, m = 2L, r = NULL)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`m`	Embedding dimension.
`r`	Tolerance. If NULL, 0.2 * SD is used.

Value

Sample entropy.

compute_gazepoint_pyhrv_sdann *Compute SDANN*

Description

Compute SDANN

Usage

```
compute_gazepoint_pyhrv_sdann(nni_ms, segment_seconds = 300)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
segment_seconds	Segment length.

Value

SDANN in milliseconds.

compute_gazepoint_pyhrv_sdnn *Compute SDNN*

Description

Compute SDNN

Usage

```
compute_gazepoint_pyhrv_sdnn(nni_ms)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.

Value

SDNN in milliseconds.

compute_gazepoint_pyhrv_sdnn_index
Compute SDNN index

Description

Compute SDNN index

Usage

```
compute_gazepoint_pyhrv_sdnn_index(nni_ms, segment_seconds = 300)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
segment_seconds	Segment length.

Value

SDNN index in milliseconds.

compute_gazepoint_pyhrv_sdsd
Compute SDSD

Description

Compute SDSD

Usage

```
compute_gazepoint_pyhrv_sdsd(nni_ms)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.

Value

SDSD in milliseconds.

`compute_gazepoint_pyhrv_time_domain` *Compute pyHRV-style time-domain summary*

Description

Compute pyHRV-style time-domain summary

Usage

```
compute_gazepoint_pyhrv_time_domain(nni_ms, segment_seconds = 300)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`segment_seconds`	Segment length for SDNN index and SDANN.

Value

Data frame with time-domain HRV measures.

`compute_gazepoint_pyhrv_tinn` *Compute TINN*

Description

Compute TINN

Usage

```
compute_gazepoint_pyhrv_tinn(nni_ms, bin_width_ms = 7.8125)
```

Arguments

`nni_ms`	Numeric NN intervals in milliseconds.
`bin_width_ms`	Histogram bin width.

Value

Approximate TINN in milliseconds.

compute_gazepoint_pyhrv_triangular_index *Compute triangular index*

Description

Compute triangular index

Usage

```
compute_gazepoint_pyhrv_triangular_index(nni_ms, bin_width_ms = 7.8125)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
bin_width_ms	Histogram bin width.

Value

HRV triangular index.

compute_gazepoint_pyhrv_welch_psd *Compute pyHRV-style Welch PSD*

Description

Compute pyHRV-style Welch PSD

Usage

```
compute_gazepoint_pyhrv_welch_psd(
  nni_ms,
  time_s = NULL,
  resample_hz = 4,
  window_seconds = 256,
  overlap = 0.5
)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
time_s	Optional time vector in seconds.
resample_hz	Resampling frequency.
window_seconds	Welch window length.
overlap	Window overlap proportion.

Value

List with PSD and measures.

`compute_gazepoint_quality_index`
Compute a heuristic QC quality index

Description

Combines user-selected numeric QC metrics into a transparent row-level heuristic quality index. Metrics are range-scaled by default and combined using user-supplied directions and weights. The result is intended for quality-control summaries only. It does not remove data and does not make physiological, psychological, diagnostic, or clinical claims.

Usage

```
compute_gazepoint_quality_index(
  data,
  metric_cols,
  directions = NULL,
  weights = NULL,
  index_col = "quality_index",
  component_prefix = "quality_component_",
  overwrite = FALSE
)
```

Arguments

`data`	A data frame.
`metric_cols`	Character vector of numeric QC metric columns.
`directions`	Direction for each metric. Use `"higher"` when higher values indicate better quality and `"lower"` when lower values indicate better quality. May be unnamed, named by metric, or length one.
`weights`	Optional non-negative numeric weights. May be unnamed, named by metric, or length one.
`index_col`	Name of the output quality-index column.
`component_prefix`	Prefix used for metric component-score columns.
`overwrite`	Logical. If `FALSE`, existing output columns are not overwritten.

Value

A data frame with class `gazepoint_quality_index`.

compute_gazepoint_scr_habituation

Compute SCR habituation across trials

Description

Estimates habituation of SCR amplitudes across ordered trials, either for a vector or within subjects/groups in a data frame.

Usage

```
compute_gazepoint_scr_habituation(
  data,
  amplitude_col = NULL,
  trial_col = NULL,
  subject_col = NULL,
  method = c("linear", "log_linear", "ratio"),
  min_trials = 3
)
```

Arguments

`data`	Numeric SCR-amplitude vector or data frame.
`amplitude_col`	SCR amplitude column for data-frame input.
`trial_col`	Trial/order column for data-frame input.
`subject_col`	Optional subject/grouping column.
`method`	`"linear"` for raw amplitudes, `"log_linear"` for log1p amplitudes, or `"ratio"` for first/last ratio only.
`min_trials`	Minimum valid trials required for model-based estimates.

Value

Data frame with habituation slope, p value, correlation, and first/last response summaries.

compute_gazepoint_scr_latency

Compute SCR latency metrics from event-locked EDA

Description

Estimates onset latency, peak latency, peak amplitude, AUC, and half-recovery latency for each event using a baseline window and response window.

Usage

```
compute_gazepoint_scr_latency(
  data,
  events,
  time_col = NULL,
  eda_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  group_cols = NULL,
  baseline_window_s = c(-1, 0),
  response_window_s = c(0, 5),
  onset_threshold = 0.01,
  recovery_fraction = 0.5
)
```

Arguments

`data`	Data frame containing EDA/GSR samples.
`events`	Event table or numeric event times.
`time_col`	Time column in `data`.
`eda_col`	EDA/GSR column.
`event_time_col`	Event-time column.
`event_id_col`	Event identifier column.
`group_cols`	Optional grouping columns used to match events to samples.
`baseline_window_s`	Two-element baseline window relative to event.
`response_window_s`	Two-element response window relative to event.
`onset_threshold`	Minimum increase above baseline used for onset.
`recovery_fraction`	Fraction of peak amplitude used for recovery time.

Value

Data frame with one row per event.

`compute_gazepoint_signal_band_power` *Compute signal band power*

Description

Compute signal band power

Usage

```
compute_gazepoint_signal_band_power(
  x,
  sampling_rate_hz = NULL,
 bands = list(very_low = c(0.003, 0.04), low = c(0.04, 0.15), high = c(0.15, 0.4)),
  relative = TRUE
)
```

Arguments

x	Numeric signal or power-spectrum data frame.
sampling_rate_hz	Sampling rate in Hz when x is a signal.
bands	Named list of frequency bands.
relative	If TRUE, include relative band power.

Value

Data frame of band powers.

---

compute_gazepoint_signal_correlation

Compute correlation between two Gazepoint signals

Description

Compute correlation between two Gazepoint signals

Usage

```
compute_gazepoint_signal_correlation(
  x,
  y,
  method = c("pearson", "spearman", "kendall"),
  lag_max = NULL
)
```

Arguments

x	First numeric signal.
y	Second numeric signal.
method	Correlation method.
lag_max	Optional maximum lag in samples for cross-correlation.

Value

Correlation summary.

compute_gazepoint_signal_lag_matrix
Compute a pairwise signal-lag matrix

Description

Computes pairwise lag/correlation summaries across synchronized numeric signals. The output is useful as a compact multimodal lag screen before event-locked or mixed-model analyses.

Usage

```
compute_gazepoint_signal_lag_matrix(
  data,
  signal_cols = NULL,
  time_col = NULL,
  group_cols = NULL,
  max_lag_s = 2,
  lag_step_s = NULL,
  min_overlap = 10
)
```

Arguments

`data`	Data frame with a common time column and numeric signals.
`signal_cols`	Numeric signal columns. If omitted, all numeric columns except `time_col` and `group_cols` are used.
`time_col`	Time column.
`group_cols`	Optional grouping columns.
`max_lag_s`	Maximum lag in seconds.
`lag_step_s`	Optional lag-step size. If omitted, the median sample interval is used.
`min_overlap`	Minimum paired samples required per lag.

Value

Data frame with one row per group and signal pair.

`compute_gazepoint_signal_phase_locking`
Compute phase-locking value between two signals

Description

Compute phase-locking value between two signals

Usage

```
compute_gazepoint_signal_phase_locking(x, y, sampling_rate_hz, band = NULL)
```

Arguments

`x`	First numeric signal.
`y`	Second numeric signal.
`sampling_rate_hz`	Sampling rate in Hz.
`band`	Optional two-element frequency band.

Value

Data frame with phase-locking value and circular phase difference.

`compute_gazepoint_signal_power_spectrum`
Compute signal power spectrum

Description

Compute signal power spectrum

Usage

```
compute_gazepoint_signal_power_spectrum(x, sampling_rate_hz, detrend = TRUE)
```

Arguments

`x`	Numeric signal.
`sampling_rate_hz`	Sampling rate in Hz.
`detrend`	If TRUE, remove the mean before FFT.

Value

Data frame with frequency and power.

compute_gazepoint_signal_quality
Compute transparent signal-quality indicators

Description

Computes conservative, rule-based descriptive quality indicators for one or more Gazepoint biometric signal columns. The function does not interpret the physiological meaning of the signal and does not exclude data.

Usage

```
compute_gazepoint_signal_quality(
  data,
  signal_cols,
  group_cols = NULL,
  flatline_tolerance = 0,
  long_missing_run_threshold = 10,
  long_constant_run_threshold = 10,
  spike_z = 4,
  extreme_z = 4
)
```

Arguments

data	A data frame.
signal_cols	Character vector of numeric signal columns to evaluate.
group_cols	Optional character vector of grouping columns, such as participant, trial, condition, session, window, or segment identifiers.
flatline_tolerance	Numeric tolerance used when detecting adjacent constant values. Defaults to 0.
long_missing_run_threshold	Integer threshold used to count whether a segment contains a long missing run. The maximum run length is always returned regardless of this threshold.
long_constant_run_threshold	Integer threshold used to count whether a segment contains a long constant run. The maximum run length is always returned regardless of this threshold.
spike_z	Numeric z-score threshold for adjacent-change spikes.
extreme_z	Numeric z-score threshold for extreme standardized values.

Value

A data frame with class gazepoint_signal_quality.

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_beats` *Apply conservative rule-based IBI corrections*

Description

Applies a conservative correction action to beats flagged by `audit_gazepoint_beats()`. The default action masks flagged intervals by setting their corrected IBI to `NA`. The local-median action replaces flagged intervals only when an unflagged local or group median is available. Every change is logged. The function does not add or remove beat rows and does not compute or interpret HRV outcomes.

Usage

```
correct_gazepoint_beats(
  audit,
  action = c("mask", "local_median"),
  corrected_col = "ibi_corrected",
  local_window = 5,
  overwrite = FALSE,
  ...
)
```

Arguments

`audit`	A `gazepoint_beat_audit` object, or a data frame that can be passed to `audit_gazepoint_beats()`.
`action`	Correction action. One of `"mask"` or `"local_median"`.
`corrected_col`	Name of the corrected IBI column to create.
`local_window`	Number of rows on each side to inspect for local-median replacement when `action = "local_median"`.
`overwrite`	Logical. If `TRUE`, overwrite an existing corrected column.
`...`	Arguments passed to `audit_gazepoint_beats()` when `audit` is a data frame.

Value

A list with class `gazepoint_beat_correction`, containing corrected beat data, a correction log, a summary table, and parameters.

`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.

correct_gazepoint_ppg_hampel

Apply Hampel-style correction to raw pulse/PPG data

Description

Apply Hampel-style correction to raw pulse/PPG data

Usage

```
correct_gazepoint_ppg_hampel(
  x,
  sampling_rate_hz,
  window_seconds = 1,
  n_sigmas = 3
)
```

Arguments

x	Numeric signal.
sampling_rate_hz	Sampling rate in Hz.
window_seconds	Running median window length.
n_sigmas	Threshold in MAD units.

Value

Corrected numeric signal.

correct_gazepoint_rri_artifacts_local

Correct local RRI/IBI artifacts

Description

Correct local RRI/IBI artifacts

Usage

```
correct_gazepoint_rri_artifacts_local(
  rri_ms,
  method = c("local_median", "quotient", "zscore"),
  window_intervals = 5L,
  threshold = 0.2,
  replacement = c("local_median", "interpolate")
)
```

Arguments

rri_ms	Numeric RRI/IBI intervals in milliseconds.
method	local_median, quotient, or zscore.
window_intervals	Local window in intervals.
threshold	Threshold for artifact detection.
replacement	Replacement method: local_median or interpolate.

Value

Data frame with corrected RRI values and artifact flags.

---

create_gazepoint_analysis_decision_log

Create a Gazepoint analysis decision log

Description

Create an empty structured log for recording preprocessing choices, exclusions, missing channels, quality-control decisions, modelling settings, cluster-permutation settings, and reviewer-facing notes.

Usage

```
create_gazepoint_analysis_decision_log(
  study_id = NA_character_,
  analyst = NA_character_,
  description = NA_character_
)
```

Arguments

study_id	Optional study identifier.
analyst	Optional analyst name or identifier.
description	Optional free-text description of the analysis workflow.

Value

A data frame with class "gazepoint_analysis_decision_log".

Examples

```
log <- create_gazepoint_analysis_decision_log(
  study_id = "demo_study",
  analyst = "analyst"
)

log <- add_gazepoint_decision(
  log,
  stage = "preprocessing",
  object_type = "signal",
  object_id = "GSR",
  decision = "baseline_corrected",
  reason = "Pre-event baseline window available",
  function_name = "baseline_correct_gazepoint_gsr",
  parameter = "baseline_window",
  value = "-1000_to_0_ms"
)

summarise_gazepoint_decision_log(log)
```

`create_gazepoint_analysis_manifest`

Create a Gazepoint analysis manifest

Description

Creates a reproducibility manifest containing package version, files, settings, outputs, exclusions, and optional session information. If `path` is supplied, a plain-text manifest is written for audit trails.

Usage

```
create_gazepoint_analysis_manifest(
  files = NULL,
  settings = list(),
  outputs = NULL,
  exclusions = NULL,
  path = NULL,
  include_session = TRUE
)
```

Arguments

`files`	Optional character vector of input files.
`settings`	Optional named list of analysis settings.
`outputs`	Optional named list or character vector of generated outputs.
`exclusions`	Optional data frame or named list of exclusions.

path Optional output path for a text manifest.

include_session
If TRUE, include sessionInfo() in the return object.

Value

Manifest list.

---

create_gazepoint_audit_index
Create a Gazepoint audit index

Description

Normalizes one or more Gazepoint audit/check objects into a single tabular index. The function is designed for reproducibility and reporting: it records object identifiers, source tables, checks, items, statuses, messages, and optional paths. It does not modify data and does not make scientific or clinical interpretations.

Usage

```
create_gazepoint_audit_index(
  audits,
  audit_ids = NULL,
  include_summary_rows = FALSE
)
```

Arguments

audits A single audit object, a data frame, or a named/unnamed list of audit objects. Recognized list elements include checks, item_results, results, issues, flags, problems, and summary.

audit_ids Optional character vector of audit identifiers. If omitted, names from audits are used where available.

include_summary_rows
Logical. If TRUE, summary rows are added when a supplied audit object contains a summary data frame. If FALSE, summary rows are added only when no detailed audit table is available.

Value

A data frame with class gazepoint_audit_index.

create_gazepoint_audit_report_section

Create a Gazepoint audit-report section

Description

Create a concise integrated report section from available Gazepoint audit objects.

Usage

```
create_gazepoint_audit_report_section(
  export_profile = NULL,
  design_audit = NULL,
  event_audit = NULL,
  condition_audit = NULL,
  decision_log = NULL,
  include_warnings = TRUE
)
```

Arguments

export_profile	Optional object from profile_gazepoint_export_folder().
design_audit	Optional object from audit_gazepoint_experiment_design().
event_audit	Optional object from audit_gazepoint_event_coverage().
condition_audit	Optional object from audit_gazepoint_condition_balance().
decision_log	Optional object from create_gazepoint_analysis_decision_log().
include_warnings	Logical. If TRUE, include warning summaries.

Value

A character vector with class "gazepoint_report_text".

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_dictionary

Create a simple Gazepoint data dictionary

Description

Creates a column-level dictionary from a data frame or from CSV file headers. The output is intended for reporting and reproducibility documentation.

Usage

```
create_gazepoint_dictionary(
  data = NULL,
  file_paths = NULL,
  units = NULL,
  descriptions = NULL,
  required_cols = NULL,
  write_path = NULL
)
```

Arguments

data	Optional data frame.
file_paths	Optional character vector of CSV files to inspect when data is NULL.
units	Optional named character vector or named list mapping columns to units.
descriptions	Optional named character vector or named list mapping columns to descriptions.
required_cols	Optional character vector of columns expected in the data.
write_path	Optional output path. Use .csv for CSV; otherwise a simple Markdown table is written.

Value

A data frame with class `gazepoint_dictionary`.

`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_eye_methods_text` *Create reproducible Gazepoint eye-tracking methods text*

Description

Create reproducible Gazepoint eye-tracking methods text

Usage

```
create_gazepoint_eye_methods_text(
  sampling_rate_hz,
  device_model = "Gazepoint GP3",
  calibration_points = 9L,
  binocular = TRUE,
  software = "Gazepoint Analysis",
  screen_resolution = NULL,
  viewing_distance_cm = NULL,
  coordinate_space = NULL,
  preprocessing = NULL,
  fixation_detection = NULL,
  aoi_definition = NULL,
  synchronization = NULL,
  exclusions = NULL,
  tense = c("past", "future"),
  include_package_version = TRUE
)
```

Arguments

`sampling_rate_hz`	Recording frequency.
`device_model`	Device model.
`calibration_points`	Number of calibration points.
`binocular`	Whether binocular data were recorded.
`software`	Optional acquisition software.
`screen_resolution`	Optional width-height pixel vector.
`viewing_distance_cm`	Optional viewing distance.
`coordinate_space`	Optional coordinate description.
`preprocessing`	Optional preprocessing descriptions.

`fixation_detection`
: Optional fixation-detection description.

`aoi_definition`
: Optional AOI description.

`synchronization`
: Optional synchronization description.

`exclusions`
: Optional exclusion description.

`tense`
: `"past"` for manuscripts or `"future"` for preregistrations.

`include_package_version`
: Include the gpbiometrics version.

Value

A character object of class `"gazepoint_eye_methods_text"`.

---

create_gazepoint_heartpy_report

Create HeartPy-style report tables for Gazepoint pulse/PPG data

---

Description

Create HeartPy-style report tables for Gazepoint pulse/PPG data

Usage

```
create_gazepoint_heartpy_report(
  detection,
  output_dir = NULL,
  prefix = "gazepoint_heartpy"
)
```

Arguments

`detection`
: Detection object returned by detect_gazepoint_ppg_peaks().

`output_dir`
: Optional output directory. If NULL, no files are written.

`prefix`
: File prefix when output_dir is supplied.

Value

A list with peaks, measures, diagnostics, and paths.

`create_gazepoint_methods_section` *Create a compact Gazepoint methods section*

Description

Generate conservative, manuscript-ready methods text from available Gazepoint workflow objects. The function reports workflow structure and audit coverage only; it does not infer emotion, stress, cognition, preference, diagnosis, mechanism, or exact temporal onset from biometric or gaze-linked signals.

Usage

```
create_gazepoint_methods_section(
  export_profile = NULL,
  design_audit = NULL,
  event_audit = NULL,
  condition_audit = NULL,
  decision_log = NULL,
  package_version = as.character(utils::packageVersion("gpbiometrics")),
  validation = NULL,
  include_guardrails = TRUE
)
```

Arguments

`export_profile`	Optional object from `profile_gazepoint_export_folder()`.
`design_audit`	Optional object from `audit_gazepoint_experiment_design()`.
`event_audit`	Optional object from `audit_gazepoint_event_coverage()`.
`condition_audit`	Optional object from `audit_gazepoint_condition_balance()`.
`decision_log`	Optional object from `create_gazepoint_analysis_decision_log()`.
`package_version`	Package version to report. Defaults to the installed gpbiometrics version.
`validation`	Optional named list with validation entries, for example `list(test = "PASS 2322", check = "0 errors, 0 warnings, 0 notes")`.
`include_guardrails`	Logical. If `TRUE`, append conservative interpretation guardrails.

Value

A character vector with class `"gazepoint_report_text"`.

Examples

```
log <- create_gazepoint_analysis_decision_log(study_id = "demo")
create_gazepoint_methods_section(decision_log = log)
```

`create_gazepoint_pipeline_map` *Create a Gazepoint pipeline map*

Description

Creates a lightweight tabular representation of a Gazepoint analysis pipeline. The output contains workflow nodes and directed edges that can be audited or exported as DOT/Graphviz text. This function is intended for documentation, reporting, and reproducibility only.

Usage

```
create_gazepoint_pipeline_map(
  steps = NULL,
  edges = NULL,
  pipeline_id = NULL,
  include_default = TRUE
)
```

Arguments

`steps`	Optional data frame of pipeline steps. Required column: `step_id`. Optional columns include `label`, `domain`, `description`, `expected_order`, `required`, `status`, and `notes`. If `NULL`, a conservative default workflow is used.
`edges`	Optional data frame of directed edges with columns `from` and `to`. Optional columns include `edge_type`, `description`, and `required`. If `NULL`, sequential edges are created from the step order.
`pipeline_id`	Optional pipeline identifier.
`include_default`	Logical. If `TRUE` and `steps = NULL`, the default Gazepoint pipeline steps are used.

Value

A list with class `gazepoint_pipeline_map`.

create_gazepoint_preregistration_checklist

Create a Gazepoint preregistration checklist

Description

Creates a structured checklist for preregistration and reviewer-readiness documentation. The checklist is descriptive only: it helps users document design, preprocessing, QC, exclusions, outcomes, and reporting decisions. It does not judge scientific validity, remove data, or make biometric, psychological, diagnostic, or clinical claims.

Usage

```
create_gazepoint_preregistration_checklist(
  study_id = NULL,
  include_optional = TRUE,
  custom_items = NULL
)
```

Arguments

study_id	Optional study identifier added to the checklist.
include_optional	Logical. If TRUE, optional auditability items such as sensitivity analyses, data dictionaries, and manifests are included.
custom_items	Optional data frame of additional checklist items. It may contain columns domain, item_id, item, required, evidence_key, and required_fields.

Value

A data frame with class gazepoint_preregistration_checklist.

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.

---

create_gazepoint_pspm_glm_design

Create PsPM-style convolution GLM design matrix

Description

Create PsPM-style convolution GLM design matrix

Usage

```
create_gazepoint_pspm_glm_design(
  events,
  time,
  time_col = NULL,
  onset_col = "onset_time_s",
  condition_col = "condition",
  duration_col = NULL,
  response = c("scr", "canonical", "boxcar"),
```

```
  response_length_s = 20,
  include_derivative = FALSE,
  add_intercept = TRUE
)
```

Arguments

events	Event table.
time	Numeric time vector or data frame containing time_col.
time_col	Time column if time is a data frame.
onset_col	Event onset column.
condition_col	Event condition column.
duration_col	Optional duration column.
response	Response kernel: scr, canonical, or boxcar.
response_length_s	Kernel length in seconds.
include_derivative	If TRUE, include temporal derivative regressors.
add_intercept	If TRUE, add intercept column.

Value

Design matrix as data frame.

`create_gazepoint_pyhrv_time_vector`
Create a time vector from NN intervals

Description

Create a time vector from NN intervals

Usage

```
create_gazepoint_pyhrv_time_vector(nni_ms, start_s = 0)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
start_s	Start time in seconds.

Value

Numeric time vector in seconds.

create_gazepoint_qc_supplement

Create a Gazepoint quality-control supplement section

Description

Generate a compact reviewer-facing quality-control supplement from available export-profile, design-audit, event-audit, condition-balance, and decision-log objects.

Usage

```
create_gazepoint_qc_supplement(
  export_profile = NULL,
  design_audit = NULL,
  event_audit = NULL,
  condition_audit = NULL,
  decision_log = NULL,
  title = "Gazepoint workflow quality-control supplement"
)
```

Arguments

export_profile	Optional object from profile_gazepoint_export_folder().
design_audit	Optional object from audit_gazepoint_experiment_design().
event_audit	Optional object from audit_gazepoint_event_coverage().
condition_audit	Optional object from audit_gazepoint_condition_balance().
decision_log	Optional object from create_gazepoint_analysis_decision_log().
title	Section title.

Value

A character vector with class "gazepoint_report_text".

create_gazepoint_quality_dashboard

Create a compact Gazepoint quality dashboard object

Description

Combines audit, missingness, synchronization, and event-locked summary objects into a reviewer-friendly quality dashboard. If output_dir is supplied, core dashboard tables are exported as CSV/text files.

Usage

```
create_gazepoint_quality_dashboard(
  data = NULL,
  audit = NULL,
  missingness = NULL,
  alignment = NULL,
  eventlocked = NULL,
  title = "Gazepoint quality dashboard",
  output_dir = NULL
)
```

Arguments

data	Optional data frame used to compute audit and missingness when explicit objects are not supplied.
audit	Optional object from audit_gazepoint_biometrics_file().
missingness	Optional object from summarize_gazepoint_missingness().
alignment	Optional object from align_gazepoint_streams_by_events().
eventlocked	Optional object from summarize_gazepoint_eventlocked_multimodal().
title	Dashboard title.
output_dir	Optional directory where dashboard tables are written.

Value

Object of class gazepoint_quality_dashboard.

---

create_gazepoint_release_checklist

Create a gpbiometrics release checklist

Description

Creates a conservative release checklist. If an audit table is supplied, each checklist item is linked to the current audit status. The checklist is meant to support human review before tagging or submission.

Usage

```
create_gazepoint_release_checklist(audit = NULL, include_optional = TRUE)
```

Arguments

audit	Optional audit data frame or gazepoint_release_readiness_audit object.
include_optional	Logical. If TRUE, optional release-polish items are included.

Value

A data frame with release checklist items.

`create_gazepoint_reproducibility_statement` *Create a Gazepoint reproducibility statement*

Description

Generate a compact reproducibility statement for manuscripts, supplements, or reviewer responses.

Usage

```
create_gazepoint_reproducibility_statement(
  decision_log = NULL,
  package_version = as.character(utils::packageVersion("gpbiometrics")),
  repository_url = NA_character_,
  validation = NULL,
  data_statement = NA_character_,
  include_guardrails = TRUE
)
```

Arguments

`decision_log`	Optional object from `create_gazepoint_analysis_decision_log()`.
`package_version`	Package version to report.
`repository_url`	Optional repository URL.
`validation`	Optional named list with validation entries.
`data_statement`	Optional text describing data availability or synthetic demonstration status.
`include_guardrails`	Logical. If `TRUE`, include conservative interpretation guardrails.

Value

A character vector with class `"gazepoint_report_text"`.

create_gazepoint_sidecar_template

Create a Gazepoint sidecar metadata template

Description

Creates a simple tabular template for dataset, export, device, timing, and processing metadata. The output can be written by the user as CSV/TSV/JSON outside this function. This function does not perform full BIDS conversion.

Usage

```
create_gazepoint_sidecar_template(
  dataset_id = NULL,
  export_type = NULL,
  include_optional = TRUE,
  custom_fields = NULL
)
```

Arguments

dataset_id	Optional dataset identifier.
export_type	Optional export type label, such as "all_gaze", "fixations", "biometrics", or "summary".
include_optional	Logical. If TRUE, optional auditability fields are included.
custom_fields	Optional data frame with columns field, description, required, value, and notes.

Value

A data frame with class gazepoint_sidecar_template.

create_gazepoint_trial_regressors

Create Gazepoint trial-level regressors for modeling

Description

Prepares a long-format trial table by joining stimulus/event design information with event-window summaries from numeric Gazepoint channels. The output is intended for downstream GLM, LMM, or mixed-model workflows.

Usage

```
create_gazepoint_trial_regressors(
  data,
  design,
  pre = 0,
  post = 5,
  time_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  signal_cols = NULL,
  subject_col = NULL,
  design_subject_col = NULL,
  carry_design_cols = NULL
)
```

Arguments

data	Data frame containing time-series signals, or a list of data frames.
design	Numeric event timestamps, a design data frame, or a list with an events data frame.
pre	Seconds before event onset to summarize.
post	Seconds after event onset to summarize.
time_col	Time column in data.
event_time_col	Event-time column in design.
event_id_col	Trial/event identifier column in design.
signal_cols	Numeric signal columns to summarize. If NULL, all numeric columns except the time column are used.
subject_col	Optional subject/participant column in data.
design_subject_col	Optional subject/participant column in design.
carry_design_cols	Design columns to carry into the output. If NULL, all non-time design columns are carried.

Value

A data frame with one row per trial/event and signal summary regressors.

Examples

```
dat <- data.frame(time_s = seq(0, 10, by = 1), GSR = seq(0, 1, length.out = 11))
design <- data.frame(trial = "T1", onset = 5, condition = "A")
create_gazepoint_trial_regressors(dat, design, pre = 1, post = 2)
```

`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_blinks` *Detect likely blink or pupil-dropout samples*

Description

Detects likely blink or pupil-dropout samples in Gazepoint pupil columns using transparent, rule-based criteria. The function flags non-finite values, values outside a plausible pupil range, and optionally large sample-to-sample changes. It does not infer attention, cognition, arousal, emotion, or visual processing from the pupil signal.

Usage

```
detect_gazepoint_blinks(
  data,
  pupil_cols = NULL,
  id_cols = NULL,
  min_pupil = 0,
  max_pupil = Inf,
  change_threshold = NULL,
  extend_samples = 0L,
  mask = TRUE,
  flag_suffix = "_blink_flag",
  clean_suffix = "_blink_clean"
)
```

Arguments

`data`	A data frame containing Gazepoint pupil data.
`pupil_cols`	Character vector naming pupil columns. If `NULL`, numeric columns whose names contain `"pupil"` are used.
`id_cols`	Optional character vector naming grouping columns. Blink detection based on sample-to-sample change is applied within groups.
`min_pupil`	Minimum plausible pupil value. Values less than or equal to this threshold are flagged. Use `NULL` to disable the lower-bound rule.
`max_pupil`	Maximum plausible pupil value. Values greater than or equal to this threshold are flagged. Use `NULL` to disable the upper-bound rule.
`change_threshold`	Optional maximum plausible absolute sample-to-sample change. Use `NULL` to disable the change rule.
`extend_samples`	Non-negative integer. Number of neighbouring samples on each side of a detected blink/dropout to also flag.
`mask`	Logical. If `TRUE`, add cleaned pupil columns with detected blink/dropout samples set to `NA`.
`flag_suffix`	Suffix appended to pupil-column names for logical flag columns.
`clean_suffix`	Suffix appended to pupil-column names for cleaned columns when `mask = TRUE`.

Value

A list with the processed data, a summary table, rule settings, and warnings. The object has class "gazepoint_blink_audit".

Examples

```
d <- data.frame(
  participant = rep("P01", 8),
  time = seq_len(8),
  pupil_left = c(3.1, 3.2, 0, 3.2, 8.5, 3.1, NA, 3.0)
)
detect_gazepoint_blinks(d, pupil_cols = "pupil_left")
```

---

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_fixations` *Detect fixations and saccades in Gazepoint gaze samples*

Description

Applies a transparent velocity-threshold classification to raw Gazepoint gaze coordinates. Samples are classified as fixation, saccade, unclassified, or invalid, after which fixation- and saccade-level event tables are derived.

Usage

```
detect_gazepoint_fixations(
  data,
  time_col = NULL,
  x_col = NULL,
  y_col = NULL,
  group_cols = NULL,
  valid_col = NULL,
  valid_values = c(1, TRUE),
  time_unit = c("seconds", "milliseconds", "microseconds", "samples"),
  sampling_rate_hz = NULL,
  coordinate_unit = c("native", "normalized", "pixels", "degrees"),
  velocity_threshold,
  min_fixation_duration_ms = 100,
  min_saccade_duration_ms = 10,
  max_gap_ms = 100,
  velocity_col = "gaze_velocity",
  class_col = "gaze_class",
  event_id_col = "gaze_event_id",
  overwrite = FALSE
)
```

Arguments

`data`	A data frame containing sample-level gaze coordinates.
`time_col`	Numeric time column. If `NULL`, common Gazepoint time-column names are searched.

x_col	Numeric horizontal gaze-coordinate column. If NULL, common Gazepoint names such as BPOGX, gaze_x, and FPOGX are searched.
y_col	Numeric vertical gaze-coordinate column. If NULL, common Gazepoint names such as BPOGY, gaze_y, and FPOGY are searched.
group_cols	Optional participant, trial, session, or file columns. Detection is performed independently within each group.
valid_col	Optional gaze-validity column.
valid_values	Values in valid_col treated as valid.
time_unit	Unit of time_col: "seconds", "milliseconds", "microseconds", or "samples".
sampling_rate_hz	Sampling rate required when time_unit = "samples".
coordinate_unit	Descriptive unit of the gaze coordinates: "native", "normalized", "pixels", or "degrees". Coordinates are not transformed automatically.
velocity_threshold	Positive velocity threshold expressed in coordinate units per second. Samples above this threshold are classified as saccadic.
min_fixation_duration_ms	Minimum fixation duration in milliseconds. Shorter fixation runs are labelled "unclassified".
min_saccade_duration_ms	Minimum saccade duration in milliseconds. Shorter saccade runs are labelled "unclassified".
max_gap_ms	Maximum permitted time difference between adjacent samples. Larger gaps split events. Set to NULL to disable this check.
velocity_col	Name of the generated sample-velocity column.
class_col	Name of the generated sample-classification column.
event_id_col	Name of the generated sample event-ID column.
overwrite	Logical. If FALSE, existing generated columns are protected.

Details

This is an I-VT-style detector rather than an exact reproduction of a specific vendor or external-toolbox implementation. The velocity threshold must match the supplied coordinate unit and should be justified, reported, and sensitivity-checked for the recording design.

Value

An object of class "gazepoint_gaze_events" containing:

- samples: original rows with velocity, class, and event ID;
- fixations: fixation-level timing, location, and dispersion summaries;
- saccades: saccade-level timing, amplitude, direction, and velocity;
- summary: group-level sample and event counts;
- settings: complete detector settings.

References

Salvucci, D. D., and Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. Proceedings of the Eye Tracking Research and Applications Symposium.

See Also

detect_gazepoint_saccades(), plot_gazepoint_saccade_main_sequence(), summarize_gazepoint_fixations()

Examples

```
gaze <- data.frame(
  time_s = seq(0, 0.9, by = 0.1),
  gaze_x = c(0, 0.01, 0.02, 0.03, 1, 1.01, 1.02, 1.03, 1.04, 1.05),
  gaze_y = 0
)

events <- detect_gazepoint_fixations(
  gaze,
  time_col = "time_s",
  x_col = "gaze_x",
  y_col = "gaze_y",
  velocity_threshold = 2,
  min_fixation_duration_ms = 100,
  min_saccade_duration_ms = 50
)

events$fixations
events$saccades
```

---

detect_gazepoint_nonwear

Detect long signal-dropout intervals

Description

Flags long missing, zero, constant, or low-variance runs in selected signal columns. The function is intended for quality control and audit reporting only. It does not remove samples and does not make physiological, psychological, diagnostic, or clinical claims.

Usage

```
detect_gazepoint_nonwear(
  data,
  signal_cols,
  group_cols = NULL,
  time_col = NULL,
  min_run_length = 10,
```

```
  zero_tolerance = 0,
  constant_tolerance = 0,
  low_variance_threshold = NULL,
  detect_missing = TRUE,
  detect_zero = TRUE,
  detect_constant = TRUE,
  detect_low_variance = TRUE
)
```

Arguments

data	A data frame.
signal_cols	Character vector of numeric signal columns to inspect.
group_cols	Optional character vector of grouping columns, such as participant, trial, session, or condition.
time_col	Optional numeric time column used to report interval start and end times.
min_run_length	Minimum number of consecutive samples required for a run to be reported.
zero_tolerance	Absolute tolerance used when detecting zero-valued runs.
constant_tolerance	Absolute adjacent-difference tolerance used when detecting constant runs.
low_variance_threshold	Optional standard-deviation threshold for low-variance windows. If NULL, low-variance detection is skipped.
detect_missing	Logical. If TRUE, detect non-finite runs.
detect_zero	Logical. If TRUE, detect near-zero runs.
detect_constant	Logical. If TRUE, detect constant runs.
detect_low_variance	Logical. If TRUE, detect low-variance windows when low_variance_threshold is supplied.

Value

A list with class gazepoint_nonwear_detection, containing interval and summary tables.

---

detect_gazepoint_ppg_onsets

Detect BioSPPy-style PPG pulse onsets

Description

Detect BioSPPy-style PPG pulse onsets

Usage

```
detect_gazepoint_ppg_onsets(
  data,
  signal_col = NULL,
  time_col = NULL,
  peaks = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  search_seconds = 0.6
)
```

Arguments

data	Data frame or numeric PPG/BVP signal.
signal_col	PPG/BVP column.
time_col	Optional time column.
peaks	Optional peak table or peak indices.
group_cols	Optional grouping columns.
sampling_rate_hz	Sampling rate in Hz.
search_seconds	Seconds before peak used to find onset.

Value

Data frame of onsets.

---

detect_gazepoint_ppg_peaks

Detect HeartPy-style pulse/PPG peaks in Gazepoint exports

Description

Detect HeartPy-style pulse/PPG peaks in Gazepoint exports

Usage

```
detect_gazepoint_ppg_peaks(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  bpm_min = 40,
  bpm_max = 180,
  moving_average_seconds = 0.75,
```

```
  threshold_offsets = seq(-0.25, 1.25, by = 0.05),
  reconstruct_clipping = TRUE,
  enhance_peaks = FALSE,
  lowpass_hz = NULL,
  hampel = FALSE,
  high_precision = TRUE
)
```

Arguments

data	Data frame or numeric signal.
signal_col	Signal column when data is a data frame.
time_col	Optional time column in seconds.
group_cols	Optional grouping columns.
sampling_rate_hz	Sampling rate in Hz.
bpm_min	Minimum plausible BPM.
bpm_max	Maximum plausible BPM.
moving_average_seconds	Moving-average half-window scale used for thresholding.
threshold_offsets	Candidate threshold offsets in signal SD units.
reconstruct_clipping	Whether to reconstruct clipped peaks.
enhance_peaks	Whether to run peak enhancement.
lowpass_hz	Optional low-pass cutoff.
hampel	Whether to apply Hampel correction.
high_precision	Whether to refine peak timing by local spline upsampling.

Value

A list with peaks, processed_signal, settings, and diagnostics.

---

detect_gazepoint_pupil_blinks

Detect blink intervals from Gazepoint pupil data

Description

Identifies blink-like intervals as runs of missing, zero, non-finite, or invalid pupil samples. For binocular data, the default treats a blink as a sample where all selected pupil channels are invalid.

Usage

```
detect_gazepoint_pupil_blinks(
  data,
  pupil_cols = NULL,
  time_col = NULL,
  group_cols = NULL,
  validity_cols = NULL,
  invalid_values = c(0),
  nonpositive_is_missing = TRUE,
  combine = c("all", "any"),
  min_blink_samples = 1L,
  return = c("intervals", "onsets", "flags")
)
```

Arguments

data	Eye-tracking data frame.
pupil_cols	Pupil columns. If NULL, common Gazepoint names such as LPD and RPD are detected automatically.
time_col	Time column. If NULL, common Gazepoint time columns are detected automatically.
group_cols	Optional grouping columns, such as participant or trial.
validity_cols	Optional validity columns corresponding to pupil_cols, such as LPV and RPV.
invalid_values	Numeric values treated as missing pupil samples.
nonpositive_is_missing	If TRUE, pupil values less than or equal to zero are treated as invalid.
combine	Whether a blink requires "all" or "any" selected pupil channels to be invalid.
min_blink_samples	Minimum run length in samples.
return	Return "intervals", "onsets", or "flags".

Value

A data frame of blink intervals by default. If return = "onsets", a numeric vector of blink onset times is returned. If return = "flags", a logical vector marking blink samples is returned.

Examples

```
dat <- data.frame(time_s = 0:4, LPD = c(3, NA, NA, 3.1, 3.2))
detect_gazepoint_pupil_blinks(dat, pupil_cols = "LPD", time_col = "time_s")
```

`detect_gazepoint_saccades` *Detect saccades in Gazepoint gaze samples*

Description

Convenience wrapper around `detect_gazepoint_fixations()` that returns only the saccade-level event table. Detector settings and group summaries are retained as attributes.

Usage

```
detect_gazepoint_saccades(...)
```

Arguments

`...`	Additional arguments passed to `detect_gazepoint_fixations()`.

Value

A data frame with class `"gazepoint_detected_saccades"` containing saccade timing, amplitude, direction, and velocity measures.

Examples

```
gaze <- data.frame(
  time_s = seq(0, 0.9, by = 0.1),
  gaze_x = c(0, 0.01, 0.02, 0.03, 1, 1.01, 1.02, 1.03, 1.04, 1.05),
  gaze_y = 0
)

detect_gazepoint_saccades(
  gaze,
  time_col = "time_s",
  x_col = "gaze_x",
  y_col = "gaze_y",
  velocity_threshold = 2,
  min_saccade_duration_ms = 50
)
```

`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"))
```

detrend_gazepoint_rri_window

Detrend RRI/IBI intervals in windows

Description

Detrend RRI/IBI intervals in windows

Usage

```
detrend_gazepoint_rri_window(
  rri_ms,
  time_s = NULL,
  window_seconds = 60,
  method = c("median", "mean", "linear")
)
```

Arguments

rri_ms	Numeric RRI/IBI intervals in milliseconds.
time_s	Optional time vector.
window_seconds	Window length for local trend.
method	mean, median, or linear.

Value

Data frame with original, trend, and detrended RRI.

detrend_gazepoint_signal

Detrend a Gazepoint signal

Description

Applies a lightweight generic detrending step to any numeric Gazepoint signal, either globally or within participant/trial groups. This is useful for slow drift in channels such as EDA, pupil size, PPG baseline, or temperature.

Usage

```
detrend_gazepoint_signal(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  method = c("linear", "mean", "median", "loess", "none"),
  span = 0.3,
  preserve_mean = FALSE,
  suffix = "_detrended"
)
```

Arguments

data	Data frame or numeric vector.
signal_col	Signal column for data-frame input.
time_col	Optional time column. If omitted, sample index is used.
group_cols	Optional grouping columns for within-group detrending.
method	Detrending method: "linear", "mean", "median", "loess", or "none".
span	Span for LOESS detrending.
preserve_mean	If TRUE, add the mean trend back after removing drift.
suffix	Suffix for the detrended signal column.

Value

Data frame with added trend and detrended columns.

---

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.

---

diagnose_gazepoint_cluster_design

Diagnose whether a design is suitable for cluster-permutation analysis

---

Description

Performs high-level checks on the data structure and design assumptions used by the current cluster-permutation prototype.

Usage

```
diagnose_gazepoint_cluster_design(
  data,
  subject,
  condition,
  time,
  value = NULL,
  design = c("within", "between"),
  min_subjects = 10L
)
```

Arguments

data	A data frame.
subject	Participant identifier column.
condition	Condition column.
time	Time-bin column.
value	Optional numeric outcome column.
design	Design to diagnose. The current runner is designed for within-subject two-condition time-course data.
min_subjects	Minimum recommended number of participants.

Value

An object of class gazepoint_cluster_design_diagnostic.

---

diagnose_gazepoint_sync_drift

Diagnose synchronization drift between two time bases

Description

Estimates lag and linear drift between matched timestamps or event markers. A fixed lag has a near-zero drift slope; a changing lag has a non-zero slope.

Usage

```
diagnose_gazepoint_sync_drift(
  reference,
  target = NULL,
  reference_time_col = NULL,
  target_time_col = NULL,
  max_pairs = NULL
)
```

Arguments

reference	Reference timestamps or data frame.
target	Target timestamps or data frame.
reference_time_col	Optional reference time column.
target_time_col	Optional target time column.
max_pairs	Optional maximum number of matched pairs.

Value

List with summary and lag_table.

`downsample_gazepoint_data` *Downsample Gazepoint time-series data*

Description

Aggregates selected numeric gaze or biometric signals into fixed-width time bins. Processing can be performed independently within participant, trial, session, or other user-defined groups. Only occupied bins are returned; the function does not fabricate observations for empty periods.

Usage

```
downsample_gazepoint_data(
  data,
  time_col,
  signal_cols = NULL,
  group_cols = NULL,
  interval,
  method = c("mean", "median", "first", "last"),
  na_rm = TRUE,
  time_value = c("start", "center", "mean"),
  origin = NULL
)
```

Arguments

`data`	A data frame containing a numeric time column and one or more numeric signal columns.
`time_col`	Name of the numeric time column.
`signal_cols`	Optional character vector of numeric columns to aggregate. If `NULL`, all numeric columns except `time_col` and `group_cols` are used.
`group_cols`	Optional character vector of grouping columns. Downsampling is performed independently within each group.
`interval`	Positive width of each output time bin, expressed in the same units as `time_col`.
`method`	Aggregation method applied to each signal within each bin: `"mean"`, `"median"`, `"first"`, or `"last"`.
`na_rm`	Logical. If `TRUE`, missing signal values are removed before aggregation. If `FALSE`, a missing value causes mean or median aggregation for that signal-bin combination to return `NA`.
`time_value`	Value assigned to the output time column: the bin `"start"`, bin `"center"`, or mean observed sample time (`"mean"`).
`origin`	Optional finite numeric origin used to align the bin grid. If `NULL`, the minimum finite time across the complete input is used.

Details

The returned object records the number of contributing source rows for each bin and stores a structured downsampling log and settings as attributes.

Value

A data frame with class "gazepoint_downsampled_data". The output contains grouping columns, the downsampled time column, aggregated signals, and n_source_rows. Attributes downsample_log and downsample_settings provide provenance information.

Examples

```
dat <- data.frame(
  participant = rep(c("P01", "P02"), each = 6),
  time_ms = rep(0:5, 2),
  pupil = c(3.0, 3.1, 3.2, 3.3, 3.4, 3.5,
            2.9, 3.0, 3.1, 3.2, 3.3, 3.4)
)

downsample_gazepoint_data(
  dat,
  time_col = "time_ms",
  signal_cols = "pupil",
  group_cols = "participant",
  interval = 3
)
```

---

enhance_gazepoint_ppg_peaks

Enhance pulse/PPG peaks using repeated local baseline removal

---

Description

Enhance pulse/PPG peaks using repeated local baseline removal

Usage

```
enhance_gazepoint_ppg_peaks(x, sampling_rate_hz, iterations = 2L)
```

Arguments

x	Numeric pulse/PPG signal.
sampling_rate_hz	Sampling rate in Hz.
iterations	Number of enhancement passes.

Value

Enhanced numeric signal.

`epoch_gazepoint_scr` *Epoch Gazepoint SCR/EDA data around events*

Description

Segments an EDA/GSR signal around event timestamps and returns event-level SCR metrics, including SCR count, maximum amplitude, mean amplitude, AUC, and baseline-corrected epoch summaries.

Usage

```
epoch_gazepoint_scr(
  data,
  events,
  pre,
  post,
  time_col = NULL,
  signal_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  event_group_cols = NULL,
  baseline_window = NULL,
  response_window = NULL,
  min_amplitude = 0.01,
  min_distance_s = 1
)
```

Arguments

`data`	Data frame containing time and EDA/GSR columns.
`events`	Numeric vector of event timestamps or a data frame with an event-time column.
`pre`	Seconds before each event to include.
`post`	Seconds after each event to include.
`time_col`	Time column in `data`. If NULL, common names are detected.
`signal_col`	EDA/GSR signal column. If NULL, common names are detected.
`event_time_col`	Event-time column when `events` is a data frame.
`event_id_col`	Optional event identifier column.
`event_group_cols`	Optional event metadata columns to carry into the output.
`baseline_window`	Baseline window relative to event time. Defaults to `c(-pre, 0)`.

response_window	SCR response window relative to event time. Defaults to c(0, post).
min_amplitude	Minimum peak-minus-trough amplitude counted as SCR.
min_distance_s	Minimum time between counted SCR peaks.

Value

Data frame with one row per event and event-level SCR metrics.

Examples

```
eda <- data.frame(time_s = seq(0, 10, by = .1), GSR = sin(seq(0, 10, by = .1)) / 20)
epoch_gazepoint_scr(eda, events = 5, pre = 1, post = 3)
```

---

estimate_gazepoint_breathing_rate_from_ibi

Estimate breathing rate from RR/IBI frequency content

Description

Estimate breathing rate from RR/IBI frequency content

Usage

```
estimate_gazepoint_breathing_rate_from_ibi(
  rr_ms,
  rr_time_s = NULL,
  resample_hz = 4,
  breathing_band = c(0.1, 0.5)
)
```

Arguments

rr_ms	RR or IBI intervals in milliseconds.
rr_time_s	Optional interval time stamps in seconds.
resample_hz	Interpolation frequency.
breathing_band	Frequency band for breathing-rate search.

Value

A list with breathing_rate_hz, frequency, psd, and band.

estimate_gazepoint_cluster_offset

Guardrail against exact cluster-offset estimation

Description

Cluster-permutation timing should be interpreted descriptively, not as a precise offset estimate.

Usage

```
estimate_gazepoint_cluster_offset(...)
```

Arguments

...	Ignored.

Value

This function always errors.

estimate_gazepoint_cluster_onset

Guardrail against exact cluster-onset estimation

Description

Cluster-permutation timing should be interpreted descriptively, not as a precise onset estimate.

Usage

```
estimate_gazepoint_cluster_onset(...)
```

Arguments

...	Ignored.

Value

This function always errors.

`estimate_gazepoint_eda_recovery_times` *Estimate BioSPPy-style EDA recovery times*

Description

Estimate BioSPPy-style EDA recovery times

Usage

```
estimate_gazepoint_eda_recovery_times(
  data,
  events = NULL,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  recovery_prop = 0.5,
  max_recovery_seconds = 10
)
```

Arguments

`data`	Data frame or numeric EDA/GSR signal.
`events`	Optional event table from extract_gazepoint_eda_events_biosppy_style().
`signal_col`	EDA/GSR column.
`time_col`	Optional time column.
`group_cols`	Optional grouping columns.
`sampling_rate_hz`	Sampling rate in Hz.
`recovery_prop`	Proportion of amplitude used for recovery threshold.
`max_recovery_seconds`	Maximum search window after peak.

Value

Event table with recovery-time columns.

`estimate_gazepoint_lsl_clock_offsets` *Estimate live LSL clock offsets through pylsl*

Description

Resolves active LSL streams and repeatedly calls each inlet's `time_correction()` method. The returned correction is ready to pass to `sync_gazepoint_signals_via_lsl()` as a value added to remote timestamps.

Usage

```
estimate_gazepoint_lsl_clock_offsets(
  stream_name = NULL,
  stream_type = NULL,
  source_id = NULL,
  timeout_s = 5,
  n_estimates = 5L,
  pause_s = 0.05,
  python = NULL,
  execute = TRUE
)
```

Arguments

`stream_name`	Optional exact LSL stream name.
`stream_type`	Optional exact LSL stream type.
`source_id`	Optional exact LSL source ID.
`timeout_s`	Resolution and first-correction timeout.
`n_estimates`	Number of offset estimates per stream.
`pause_s`	Pause between estimates.
`python`	Optional Python executable or Windows `py` launcher.
`execute`	If `FALSE`, return a dry-run specification.

Value

A `"gazepoint_lsl_clock_offsets"` object with raw estimates, stream-level summaries, and named median offsets.

`estimate_gazepoint_respiration_from_ppg` *Estimate respiration rate from PPG*

Description

Estimates an exploratory respiration rate from low-frequency modulation in a PPG/BVP signal. This is a lightweight screening helper and should not be interpreted as a replacement for a respiratory sensor.

Usage

```
estimate_gazepoint_respiration_from_ppg(
  data,
  ppg_col = NULL,
  time_col = NULL,
  sampling_rate_hz = NULL,
  respiratory_band_hz = c(0.1, 0.5),
  detrend = TRUE
)
```

Arguments

`data`	Data frame or numeric PPG vector.
`ppg_col`	PPG/BVP signal column for data-frame input.
`time_col`	Optional time column.
`sampling_rate_hz`	Sampling rate for vector input or when no time column is available.
`respiratory_band_hz`	Two-element frequency band used to search for the respiration peak.
`detrend`	If TRUE, remove a linear trend before spectral estimation.

Value

List with `summary`, `spectrum`, and `settings`.

`estimate_gazepoint_samplerate_datetime` *Estimate sampling rate from datetime stamps*

Description

Estimate sampling rate from datetime stamps

Usage

```
estimate_gazepoint_samplerate_datetime(
  datetime,
  format = NULL,
  tz = "UTC",
  robust = TRUE
)
```

Arguments

datetime	POSIXct, POSIXlt, Date, or character timestamps.
format	Optional datetime format for character input.
tz	Time zone used when parsing character input.
robust	If TRUE, use the median interval.

Value

Estimated sampling-rate information.

---

estimate_gazepoint_samplerate_mstimer

Estimate sampling rate from a millisecond timer

Description

Estimate sampling rate from a millisecond timer

Usage

```
estimate_gazepoint_samplerate_mstimer(mstimer, robust = TRUE)
```

Arguments

mstimer	Numeric millisecond timer.
robust	If TRUE, use the median interval.

Value

Estimated sampling-rate information.

`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_audit_trail_markdown`
Export a Gazepoint audit trail as Markdown

Description

Creates a plain Markdown audit-trail summary from a Gazepoint audit index. The output can be copied into supplementary materials, review responses, or project documentation. No rendering dependency is required.

Usage

```
export_gazepoint_audit_trail_markdown(
  audit_index,
  summary = NULL,
  title = "Gazepoint audit trail",
  include_details = TRUE,
  max_details = 50,
  file = NULL
)
```

Arguments

`audit_index`	A `gazepoint_audit_index` object, a data frame, or an object accepted by `create_gazepoint_audit_index()`.
`summary`	Optional summary table. If `NULL`, the summary is created with `summarize_gazepoint_audit_trail()`.
`title`	Markdown title.
`include_details`	Logical. If `TRUE`, detail rows are included.
`max_details`	Maximum number of detail rows to include.
`file`	Optional file path. If supplied, the Markdown text is written to disk.

Value

A single character string containing Markdown text.

`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_cluster_results

Export cluster-permutation result components

Description

Writes the cluster summary, timewise statistics, null distribution, parameter table, and conservative reporting text to disk.

Usage

```
export_gazepoint_cluster_results(
  result,
  path = ".",
  prefix = "gazepoint_cluster",
  overwrite = FALSE
)
```

Arguments

result	Object returned by run_gazepoint_cluster_permutation().
path	Output directory.
prefix	File-name prefix.
overwrite	Should existing files be overwritten?

Value

A data frame listing written files, invisibly.

export_gazepoint_heartpy_input

Export Gazepoint pulse/PPG data for HeartPy-style workflows

Description

Export Gazepoint pulse/PPG data for HeartPy-style workflows

Usage

```
export_gazepoint_heartpy_input(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  output_dir,
  prefix = "gazepoint_heartpy"
)
```

Arguments

data	Data frame containing Gazepoint biometric samples.
signal_col	Pulse/PPG signal column. If NULL, a likely column is inferred.
time_col	Time column in seconds. If NULL, a likely column is inferred or created from sampling_rate_hz.
group_cols	Optional grouping columns such as participant or trial identifiers.
sampling_rate_hz	Sampling rate in Hz. Required if time_col cannot be inferred.
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 returned by prepare_gazepoint_heartpy_input().

---

export_gazepoint_mne_cluster_input *Export time-course data for MNE cluster-permutation workflows*

Description

Prepares participant-level condition differences in a wide matrix format commonly used by MNE-style one-sample cluster-permutation workflows.

Usage

```
export_gazepoint_mne_cluster_input(
  data,
  outcome_col,
  time_col,
  condition_col,
  participant_col,
  condition_a = NULL,
```

```
  condition_b = NULL,
  path = NULL,
  prefix = "gazepoint_mne_cluster",
  overwrite = FALSE,
  aggregate = TRUE
)
```

Arguments

data	A data frame.
outcome_col	Numeric outcome column.
time_col	Time-bin column.
condition_col	Condition column.
participant_col	Participant identifier column.
condition_a	Reference condition. If NULL, the first sorted condition is used.
condition_b	Comparison condition. If NULL, the second sorted condition is used.
path	Optional output directory. If NULL, files are not written.
prefix	File-name prefix used when path is supplied.
overwrite	Should existing files be overwritten?
aggregate	Should repeated participant-condition-time cells be averaged?

Value

A list containing long data, a wide difference matrix, and metadata, invisibly if files are written.

---

export_gazepoint_permuco_cluster_input

Export time-course data for permuco cluster workflows

Description

Prepares conservative long-format participant-level time-course data and metadata for external analysis in packages such as permuco.

Usage

```
export_gazepoint_permuco_cluster_input(
  data,
  outcome_col,
  time_col,
  condition_col,
  participant_col,
  path = NULL,
```

```
  prefix = "gazepoint_permuco_cluster",
  overwrite = FALSE,
  aggregate = TRUE
)
```

Arguments

data	A data frame.
outcome_col	Numeric outcome column.
time_col	Time-bin column.
condition_col	Condition column.
participant_col	Participant identifier column.
path	Optional output directory.
prefix	File-name prefix.
overwrite	Should existing files be overwritten?
aggregate	Should repeated participant-condition-time cells be averaged?

Value

A list containing long data and metadata, or a file table if path is supplied.

---

export_gazepoint_permutes_cluster_input

Export time-course data for permutes cluster workflows

Description

Prepares conservative long-format participant-level time-course data and metadata for external analysis in packages such as permutes.

Usage

```
export_gazepoint_permutes_cluster_input(
  data,
  outcome_col,
  time_col,
  condition_col,
  participant_col,
  path = NULL,
  prefix = "gazepoint_permutes_cluster",
  overwrite = FALSE,
  aggregate = TRUE
)
```

Arguments

data	A data frame.
outcome_col	Numeric outcome column.
time_col	Time-bin column.
condition_col	Condition column.
participant_col	Participant identifier column.
path	Optional output directory.
prefix	File-name prefix.
overwrite	Should existing files be overwritten?
aggregate	Should repeated participant-condition-time cells be averaged?

Value

A list containing long data and metadata, or a file table if path is supplied.

---

export_gazepoint_pipeline_dot

Export a Gazepoint pipeline map as DOT text

Description

Converts a Gazepoint pipeline map to a lightweight DOT/Graphviz character string. The DOT text can be copied into external graph-rendering tools. This function does not require Graphviz, DiagrammeR, or any rendering dependency.

Usage

```
export_gazepoint_pipeline_dot(
  pipeline,
  file = NULL,
  graph_name = "gazepoint_pipeline",
  rankdir = "LR",
  include_descriptions = FALSE
)
```

Arguments

pipeline	A gazepoint_pipeline_map object or a data frame of pipeline steps.
file	Optional file path. If supplied, the DOT text is written to disk.
graph_name	DOT graph name.
rankdir	Graph direction. Common values are "LR" and "TB".
include_descriptions	Logical. If TRUE, node descriptions are included in DOT labels.

Value

A single character string containing DOT text.

`export_gazepoint_pspm_model_estimates`
Export PsPM-style Gazepoint model estimates

Description

Export PsPM-style Gazepoint model estimates

Usage

```
export_gazepoint_pspm_model_estimates(
  model,
  path,
  format = NULL,
  include_predictions = TRUE
)
```

Arguments

`model`	Model object from fit_gazepoint_convolution_glm().
`path`	Output path. Use .csv, .rds, or .json.
`format`	Optional format. If NULL, inferred from path extension.
`include_predictions`	If TRUE, CSV export also writes predictions.

Value

Data frame of written files.

`export_gazepoint_pyhrv_results`
Export pyHRV-style results

Description

Export pyHRV-style results

Usage

```
export_gazepoint_pyhrv_results(results, path)
```

Arguments

results	Result object.
path	Output path. Use .json or .rds.

Value

Output path invisibly.

---

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`.

`export_gazepoint_to_bids`

Export Gazepoint eye-tracking data to BIDS

Description

Exports one Gazepoint eye-tracking recording as a BIDS 1.11.1 physiological-recording pair consisting of a compressed, headerless `_physio.tsv.gz` file and a matching JSON sidecar.

Usage

```
export_gazepoint_to_bids(
  data,
  bids_root,
  subject,
  task,
  dataset_name = NULL,
  recorded_eye = c("cyclopean", "left", "right"),
  recording = "eye1",
 datatype = c("beh", "eeg", "emg", "func", "ieeg", "meg", "motion", "nirs", "pet"),
  session = NULL,
  acquisition = NULL,
  run = NULL,
  timestamp_col = NULL,
  x_col = NULL,
  y_col = NULL,
  include_pupil = TRUE,
  pupil_col = NULL,
  additional_cols = NULL,
  timestamp_units = c("auto", "seconds", "milliseconds", "microseconds"),
 coordinate_units = c("normalized", "pixel", "degree", "radian", "mm", "cm", "m",
    "arbitrary"),
  pupil_units = "arbitrary",
  sample_coordinate_system = c("gaze-on-screen", "eye-in-head", "gaze-in-world",
    "custom"),
  sampling_rate_hz = NULL,
  sampling_tolerance = 0.05,
  start_time_s = 0,
  screen_distance_m = NULL,
  screen_origin = NULL,
  screen_resolution_px = NULL,
  screen_size_m = NULL,
  screen_refresh_rate_hz = NULL,
```

```
  stimulus_software_name = NULL,
  stimulus_software_version = NULL,
  operating_system = NULL,
  vision_correction = NULL,
  manufacturer = "Gazepoint",
  manufacturers_model_name = NULL,
  software_versions = NULL,
  device_serial_number = NULL,
  eye_tracking_method = "P-CR",
  calibration_type = NULL,
  calibration_count = NULL,
  average_calibration_error_deg = NULL,
  maximal_calibration_error_deg = NULL,
  eye_tracker_distance_m = NULL,
  raw_data_filters = NULL,
  timestamp_origin = "Eye-tracker clock",
  custom_coordinate_system_description = NULL,
  column_metadata = list(),
  bids_version = "1.11.1",
  dry_run = FALSE,
  overwrite = FALSE
)
```

Arguments

`data`	Data frame containing regularly sampled Gazepoint eye-tracking samples.
`bids_root`	Root directory of the BIDS dataset.
`subject`	BIDS subject label without the `sub-` prefix.
`task`	BIDS task label without the `task-` prefix.
`dataset_name`	Dataset name used when creating `dataset_description.json`. May be `NULL` when a valid file already exists.
`recorded_eye`	Recorded eye: `"left"`, `"right"`, or `"cyclopean"`.
`recording`	Required BIDS recording label. The recommended labels are `"eye1"`, `"eye2"`, and `"eye3"`.
`datatype`	BIDS datatype directory. Supported values are `"beh"`, `"eeg"`, `"emg"`, `"func"`, `"ieeg"`, `"meg"`, `"motion"`, `"nirs"`, and `"pet"`.
`session`	Optional BIDS session label without the `ses-` prefix.
`acquisition`	Optional BIDS acquisition label without the `acq-` prefix.
`run`	Optional non-negative BIDS run index.
`timestamp_col`	Timestamp column issued by the eye tracker. If `NULL`, common Gazepoint timestamp columns are searched.
`x_col`	Horizontal gaze-coordinate column. If `NULL`, common Gazepoint columns are searched according to `recorded_eye`.
`y_col`	Vertical gaze-coordinate column. If `NULL`, common Gazepoint columns are searched according to `recorded_eye`.

include_pupil Logical. Attempt to include a pupil_size column.

pupil_col Optional pupil-size column. If NULL and include_pupil = TRUE, common Gazepoint pupil columns are searched.

additional_cols
Optional additional numeric or logical columns to append after the prescribed eye-tracking columns.

timestamp_units
Units of the source timestamp: "auto", "seconds", "milliseconds", or "microseconds".

coordinate_units
Units of the gaze coordinates. "normalized" is encoded as the dimensionless BIDS unit "1".

pupil_units Units of pupil size when included.

sample_coordinate_system
Coordinate system: "gaze-on-screen", "eye-in-head", "gaze-in-world", or "custom".

sampling_rate_hz
Optional positive sampling frequency. If omitted, it is inferred from the timestamps.

sampling_tolerance
Maximum relative deviation from the expected sampling interval.

start_time_s BIDS StartTime, in seconds relative to the associated acquisition.

screen_distance_m
Required for "gaze-on-screen" unless already defined in an existing events JSON sidecar.

screen_origin Two strings describing vertical and horizontal screen origin, for example c("top", "left").

screen_resolution_px
Two positive integers giving screen width and height in pixels.

screen_size_m Two positive numbers giving screen width and height in metres.

screen_refresh_rate_hz
Optional screen refresh rate.

stimulus_software_name
Optional stimulus-presentation software name.

stimulus_software_version
Optional stimulus-presentation software version.

operating_system
Optional operating-system description.

vision_correction
Optional vision-correction description.

manufacturer Optional eye-tracker manufacturer.

manufacturers_model_name
Optional eye-tracker model.

software_versions
Optional acquisition-software version.

`device_serial_number`
: Optional device serial number or pseudonym.

`eye_tracking_method`
: Optional eye-tracking method.

`calibration_type`
: Optional calibration type, for example `"HV9"`.

`calibration_count`
: Optional non-negative calibration count.

`average_calibration_error_deg`
: Optional average calibration error.

`maximal_calibration_error_deg`
: Optional maximal calibration error.

`eye_tracker_distance_m`
: Optional eye-to-tracker distance in metres.

`raw_data_filters`
: Optional description of device-side filters.

`timestamp_origin`
: Description of the timestamp origin.

`custom_coordinate_system_description`
: Required when `sample_coordinate_system = "custom"`.

`column_metadata`
: Optional named list of additional or replacement metadata entries for exported columns.

`bids_version`
: BIDS specification version written to a newly created dataset description.

`dry_run`
: Logical. Validate and preview paths without writing files.

`overwrite`
: Logical. Permit replacement of the recording pair and merging of supplied stimulus metadata into an existing events JSON file.

Details

One call exports one eye or one cyclopean recording. Binocular recordings should be exported through separate calls with different `recording` labels and appropriate `RecordedEye` metadata.

The compressed TSV file is headerless. Its first three columns are always `timestamp`, `x_coordinate`, and `y_coordinate`. Missing numeric values are written as `n/a`.

When `sample_coordinate_system = "gaze-on-screen"`, the corresponding events JSON metadata must define screen distance, origin, resolution, and physical size. This function creates or validates that metadata.

This helper does not execute the external BIDS Validator. Exported datasets should still be checked with the current official validator.

Value

An object of class `"gazepoint_bids_export"` containing the prepared table, sidecars, file manifest, audit information, and settings.

See Also

check_gazepoint_bids()

Examples

```
gaze <- data.frame(
  TIME = c(0, 1 / 60, 2 / 60),
  BPOGX = c(0.45, 0.46, 0.47),
  BPOGY = c(0.52, 0.51, 0.50)
)

preview <- export_gazepoint_to_bids(
  gaze,
  bids_root = tempfile("bids-"),
  subject = "01",
  task = "viewing",
  dataset_name = "Gazepoint viewing study",
  recorded_eye = "cyclopean",
  coordinate_units = "normalized",
  screen_distance_m = 0.60,
  screen_origin = c("top", "left"),
  screen_resolution_px = c(1920, 1080),
  screen_size_m = c(0.53, 0.30),
  dry_run = TRUE
)

preview$files
```

---

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_events_biosppy_style`
Extract BioSPPy-style EDA events from Gazepoint GSR/EDA data

Description

Extract BioSPPy-style EDA events from Gazepoint GSR/EDA data

Usage

```
extract_gazepoint_eda_events_biosppy_style(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  smoothing_seconds = 1,
  min_amplitude = NULL,
  min_distance_seconds = 1,
  onset_window_seconds = 4
)
```

Arguments

`data`	Data frame or numeric EDA/GSR signal.
`signal_col`	EDA/GSR column.
`time_col`	Optional time column in seconds.
`group_cols`	Optional grouping columns.
`sampling_rate_hz`	Sampling rate in Hz.
`smoothing_seconds`	Smoothing window for tonic estimate.
`min_amplitude`	Minimum SCR amplitude.
`min_distance_seconds`	Minimum distance between SCR peaks.
`onset_window_seconds`	Window before peak used to find onset.

Value

Data frame of EDA events.

`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_markerinfo_pspm_style` *Extract PsPM-style marker information from Gazepoint biometrics data*

Description

Extract PsPM-style marker information from Gazepoint biometrics data

Usage

```
extract_gazepoint_markerinfo_pspm_style(
  data,
  marker_cols = NULL,
  time_col = NULL,
  sampling_rate_hz = NULL,
  group_cols = NULL,
  edge = c("rising", "change", "nonzero"),
  nonzero_only = TRUE
)
```

Arguments

`data`	Gazepoint data frame.
`marker_cols`	Marker/TTL columns. If NULL, likely marker columns are inferred.
`time_col`	Time column in seconds. If NULL, inferred or created from sampling_rate_hz.
`sampling_rate_hz`	Sampling rate used when no time column is available.
`group_cols`	Optional grouping columns such as participant or trial.
`edge`	Event rule: rising, change, or nonzero.
`nonzero_only`	Whether zero-valued markers should be ignored.

Value

Data frame with marker events.

`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_ppg_morphology`
Extract simple PPG pulse morphology metrics

Description

Extracts peak-centered pulse morphology metrics from a PPG/BVP signal, including rise time, decay time, half-amplitude width, pulse amplitude, and a simple post-peak notch proxy.

Usage

```
extract_gazepoint_ppg_morphology(
  data,
  time_col = NULL,
  ppg_col = NULL,
  peaks = NULL,
  min_peak_distance_s = 0.3
)
```

Arguments

`data`	PPG data frame.
`time_col`	Time column.
`ppg_col`	PPG/BVP signal column.
`peaks`	Optional peak indices or peak times.
`min_peak_distance_s`	Minimum distance between automatically detected peaks.

Value

Data frame with one row per pulse peak.

`extract_gazepoint_ppg_templates` *Extract BioSPPy-style PPG pulse templates*

Description

Extract BioSPPy-style PPG pulse templates

Usage

```
extract_gazepoint_ppg_templates(
  data,
  signal_col = NULL,
  time_col = NULL,
  peaks = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  before_seconds = 0.3,
  after_seconds = 0.6
)
```

Arguments

`data`	Data frame or numeric PPG/BVP signal.
`signal_col`	PPG/BVP column.
`time_col`	Optional time column.
`peaks`	Optional peak table or peak indices.
`group_cols`	Optional grouping columns.
`sampling_rate_hz`	Sampling rate in Hz.
`before_seconds`	Seconds before peak.
`after_seconds`	Seconds after peak.

Value

List with templates, average template, and template time.

extract_gazepoint_pyhrv_nn_intervals

Extract NN intervals from peak timestamps

Description

Extract NN intervals from peak timestamps

Usage

```
extract_gazepoint_pyhrv_nn_intervals(
  peaks,
  peak_time_col = "peak_time_s",
  time_unit = c("seconds", "milliseconds")
)
```

Arguments

peaks	Data frame with peak timestamps, or numeric peak timestamps.
peak_time_col	Peak timestamp column when peaks is a data frame.
time_unit	Unit of peak timestamps: seconds or milliseconds.

Value

Numeric NN intervals in milliseconds.

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_segments_pspm_style

Extract event-centred Gazepoint segments PsPM-style

Description

Extract event-centred Gazepoint segments PsPM-style

Usage

```
extract_gazepoint_segments_pspm_style(
  data,
  events,
  signal_col,
  time_col = NULL,
  event_time_col = "onset_time_s",
  event_id_col = NULL,
  condition_col = NULL,
  pre_s = 1,
  post_s = 5,
  baseline_window = c(-1, 0),
  baseline_correct = TRUE
)
```

Arguments

data	Gazepoint signal data.
events	Event table.
signal_col	Signal column.
time_col	Signal time column.
event_time_col	Event onset-time column.
event_id_col	Optional event-id column.
condition_col	Optional condition column.
pre_s	Seconds before event.
post_s	Seconds after event.
baseline_window	Baseline window relative to event.
baseline_correct	If TRUE, subtract event baseline.

Value

Long-format segment table.

---

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_gaze *Filter implausible Gazepoint gaze samples*

Description

Flags and optionally removes gaze samples outside screen bounds or with implausibly high point-to-point velocity.

Usage

```
filter_gazepoint_gaze(
  data,
  x_col = NULL,
  y_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  screen_bounds = c(0, 1, 0, 1),
  max_velocity = Inf,
  drop_invalid = FALSE,
  suffix = "_filtered"
)
```

Arguments

data	Gaze data frame.
x_col	Gaze x column. If NULL, common Gazepoint columns are detected.
y_col	Gaze y column. If NULL, common Gazepoint columns are detected.
time_col	Optional time column.
group_cols	Optional grouping columns.
screen_bounds	Numeric vector c(x_min, x_max, y_min, y_max).
max_velocity	Maximum allowed gaze velocity in coordinate units per second. Use Inf to disable velocity filtering.
drop_invalid	If TRUE, return only valid rows.
suffix	Suffix for filtered coordinate columns.

Value

Data frame with validity flags, velocity, filter reason, and filtered x/y columns.

Examples

```
gaze <- data.frame(time_s = 1:3, BPOGX = c(.1, .2, 2), BPOGY = c(.2, .3, .4))
filter_gazepoint_gaze(gaze, screen_bounds = c(0, 1, 0, 1))
```

---

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.

---

filter_gazepoint_ppg_butterworth

Apply a second-order Butterworth-style low-pass filter to pulse/PPG data

---

Description

Apply a second-order Butterworth-style low-pass filter to pulse/PPG data

Usage

```
filter_gazepoint_ppg_butterworth(
  x,
  cutoff_hz = 5,
  sampling_rate_hz,
  passes = 1L
)
```

Arguments

x	Numeric signal.
cutoff_hz	Low-pass cutoff frequency in Hz.
sampling_rate_hz	Sampling rate in Hz.
passes	Number of repeated two-pole sections.

Value

Filtered numeric signal.

`filter_gazepoint_ppg_signal` *Apply generic filtering to Gazepoint pulse/PPG data*

Description

Apply generic filtering to Gazepoint pulse/PPG data

Usage

```
filter_gazepoint_ppg_signal(
  x,
  sampling_rate_hz,
  type = c("lowpass", "highpass", "bandpass", "notch"),
  low_hz = NULL,
  high_hz = NULL,
  passes = 1L
)
```

Arguments

`x`	Numeric signal.
`sampling_rate_hz`	Sampling rate in Hz.
`type`	Filter type: lowpass, highpass, bandpass, or notch.
`low_hz`	Low cutoff for highpass, bandpass, or notch.
`high_hz`	High cutoff for lowpass, bandpass, or notch.
`passes`	Repeated filter passes.

Value

Filtered numeric signal.

`filter_gazepoint_signal` *Apply lightweight preprocessing to signal columns*

Description

Applies conservative base-R preprocessing to selected numeric signal columns. Supported methods are moving average, rolling median, and linear detrending. The function returns new columns by default and records a processing log.

Usage

```
filter_gazepoint_signal(
  data,
  signal_cols,
  method = c("moving_average", "rolling_median", "detrend"),
  group_cols = NULL,
  time_col = NULL,
  window = 5,
  suffix = NULL,
  overwrite = FALSE,
  na_rm = FALSE
)
```

Arguments

`data`	A data frame.
`signal_cols`	Character vector of numeric signal columns.
`method`	One of `"moving_average"`, `"rolling_median"`, or `"detrend"`.
`group_cols`	Optional grouping columns.
`time_col`	Optional numeric time column used for ordering within groups.
`window`	Window length for moving-average and rolling-median methods.
`suffix`	Optional suffix for created columns. If `NULL`, a suffix is derived from `method`.
`overwrite`	Logical. If `TRUE`, overwrite input signal columns.
`na_rm`	Logical. If `TRUE`, ignore missing values inside rolling windows.

Value

A data frame with class `gazepoint_filtered_signal`.

fit_gazepoint_convolution_glm

Fit PsPM-style event-related convolution GLM

Description

Fit PsPM-style event-related convolution GLM

Usage

```
fit_gazepoint_convolution_glm(
  data,
  design,
  signal_col,
  time_col = NULL,
  design_time_col = "time_s",
  regressor_cols = NULL
)
```

Arguments

data	Gazepoint signal data.
design	Design matrix from create_gazepoint_pspm_glm_design().
signal_col	Signal column.
time_col	Time column in data.
design_time_col	Time column in design.
regressor_cols	Optional regressor columns. If NULL, inferred.

Value

List with coefficients, fitted values, residuals, and model summary.

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_hrv_segments

Flag HRV/RR segments for quality review

Description

Splits RR/NN intervals into windows and flags segments with too few beats, implausible intervals, excessive beat-to-beat changes, short duration, or high artifact burden. The helper is intended as transparent QC before HRV summaries or mixed-model analysis.

Usage

```
flag_gazepoint_hrv_segments(
  data,
  rr_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  window_s = 60,
  min_beats = 20,
  min_duration_s = 20,
  min_rr_ms = 300,
  max_rr_ms = 2000,
  max_artifact_prop = 0.2,
  max_successive_change_prop = 0.2
)
```

Arguments

data	Data frame or numeric RR/NN interval vector.
rr_col	RR/NN interval column for data-frame input.
time_col	Optional timestamp column. If omitted, cumulative RR time is used.
group_cols	Optional grouping columns such as participant or condition.
window_s	Optional window length in seconds. If NULL, one segment is returned per group.
min_beats	Minimum finite beats required for quality_ok.
min_duration_s	Minimum segment duration in seconds.
min_rr_ms	Minimum plausible RR interval in milliseconds.
max_rr_ms	Maximum plausible RR interval in milliseconds.
max_artifact_prop	Maximum allowed artifact proportion.
max_successive_change_prop	Maximum allowed proportional beat-to-beat change before an interval is flagged.

Value

Data frame with one row per segment.

---

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_gazepoint_ppg_quality

Flag segment-level PPG quality

Description

Computes windowed PPG quality metrics and flags segments with excessive missingness, flatness, or robust outliers.

Usage

```
flag_gazepoint_ppg_quality(
  data,
  time_col = NULL,
  ppg_col = NULL,
  window_s = 10,
  step_s = NULL,
  missing_prop_threshold = 0.2,
  flat_sd_threshold = 1e-06,
  outlier_prop_threshold = 0.1
)
```

Arguments

`data`	PPG data frame.
`time_col`	Time column.
`ppg_col`	PPG/BVP signal column.
`window_s`	Window duration in seconds.
`step_s`	Step size in seconds. Defaults to `window_s`.
`missing_prop_threshold`	Maximum allowed missing proportion.
`flat_sd_threshold`	Minimum allowed standard deviation.
`outlier_prop_threshold`	Maximum allowed robust outlier proportion.

Value

Data frame with one row per segment.

`flag_gazepoint_rr_outliers` *Flag outlying Gazepoint RR/IBI intervals*

Description

Flags implausible RR intervals using absolute physiological limits and either robust MAD-based or z-score-based deviation rules.

Usage

```
flag_gazepoint_rr_outliers(
  rr_intervals,
  method = c("mad", "z", "range"),
  z_threshold = 5,
  mad_threshold = 5,
  min_rr = 300,
  max_rr = 2000,
  return = c("flags", "filtered", "data")
)
```

Arguments

rr_intervals	Numeric RR/IBI vector, usually in milliseconds.
method	"mad", "z", or "range".
z_threshold	Z-score threshold when method = "z".
mad_threshold	Robust MAD threshold when method = "mad".
min_rr	Minimum plausible RR interval in ms.
max_rr	Maximum plausible RR interval in ms.
return	"flags", "filtered", or "data".

Value

Logical vector, filtered RR vector with outliers set to NA, or a data frame with reason columns.

Examples

```
flag_gazepoint_rr_outliers(c(800, 810, 3000, 790))
```

---

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.

---

flip_gazepoint_ppg_signal

Flip a Gazepoint pulse/PPG signal

Description

Flip a Gazepoint pulse/PPG signal

Usage

```
flip_gazepoint_ppg_signal(x, method = c("negative", "max_minus"))
```

Arguments

x	Numeric signal.
method	Flip method: negative or max_minus.

Value

Flipped numeric signal.

`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.

---

gazepoint_interoperability_manifest

Define gpbiometrics interoperability targets

Description

Creates the machine-readable interoperability manifest used by audit_gazepoint_interoperability_versions(). The manifest distinguishes R packages, Python distributions, supporting Python packages, and external standards.

Usage

```
gazepoint_interoperability_manifest(include_support = TRUE)
```

Arguments

include_support
: Logical. Include NumPy and pandas support rows in addition to the direct interoperability targets.

Details

A minimum tested version records the oldest version exercised by the maintained interoperability workflow. It is not a claim that every earlier version is incompatible.

Value

A data frame of class "gazepoint_interoperability_manifest" with one row per target.

Examples

```
manifest <- gazepoint_interoperability_manifest()
manifest[, c("target", "dependency", "minimum_tested_version")]
```

---

generate_gazepoint_manifest

Generate a reproducibility manifest

Description

Creates a conservative analysis manifest containing package/session information, input file metadata, output paths, user-supplied parameters, and notes. The function is designed for auditability and does not inspect private data contents.

Usage

```
generate_gazepoint_manifest(
  input_paths = NULL,
  parameters = list(),
  outputs = NULL,
  notes = NULL,
  write_path = NULL,
  include_session_info = TRUE
)
```

Arguments

input_paths	Optional character vector of input file or folder paths.
parameters	Optional named list of analysis parameters.
outputs	Optional character vector of output paths or object names.
notes	Optional character vector of free-text notes.

`write_path` Optional path. Use `.rds` to save the manifest object; otherwise a plain-text manifest is written.

`include_session_info` Logical. If `TRUE`, include `utils::sessionInfo()`.

Value

A list with class `gazepoint_manifest`.

`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.

`gpbiometrics_info` *Print gpbiometrics reproducibility information*

Description

Returns package, R, platform, and session information useful for manuscripts, reproducibility logs, and bug reports.

Usage

```
gpbiometrics_info(print = TRUE, include_session = FALSE)
```

Arguments

`print`	If TRUE, print a compact summary.
`include_session`	If TRUE, include `utils::sessionInfo()` in the return object.

Value

A list with package and session metadata.

`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_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_data` *Import Gazepoint export files from a session folder*

Description

Loads Gazepoint-style CSV exports from a folder into a named list of data frames. The function is intended as a single entry point for common session folders that contain all-gaze, fixation, summary, biometric, marker, or related exports.

Usage

```
import_gazepoint_data(
  dir,
  session = NULL,
  pattern = "\\.csv$",
  recursive = FALSE,
  session_match = c("prefix", "contains", "regex"),
  file_encoding = "UTF-8-BOM",
  add_file_info = TRUE
)
```

Arguments

`dir`	Folder containing Gazepoint export files.
`session`	Optional session or participant prefix used to select files. Can be a character vector.
`pattern`	File pattern. Defaults to CSV files.
`recursive`	If TRUE, search subfolders.
`session_match`	How `session` should be matched: `"prefix"`, `"contains"`, or `"regex"`.
`file_encoding`	Encoding passed to `read.table()`.
`add_file_info`	If TRUE, add source-file columns to each imported data frame.

Value

A named list of data frames with class `gazepoint_session_data`. File metadata are stored in the `file_index` attribute.

Examples

```
## Not run:
session_data <- import_gazepoint_data("path/to/session", session = "P01")
names(session_data)
attr(session_data, "file_index")

## End(Not run)
```

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_event_log

Import a generic Gazepoint-compatible event log

Description

Reads a CSV/TSV event log and standardizes event identifiers, timestamps, and labels for alignment with Gazepoint biometric or eye-tracking data.

Usage

```
import_gazepoint_event_log(
  path,
  time_col = NULL,
  event_col = NULL,
  id_col = NULL,
  sep = NULL,
  ...
)
```

Arguments

path	Path to a CSV/TSV file, or an existing data frame.
time_col	Event time column.
event_col	Event label column.
id_col	Optional event identifier column.
sep	Optional delimiter. If NULL, it is guessed from the first line.
...	Additional arguments passed to utils::read.table().

Value

Data frame with event_id, event_time, and event_label.

---

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.

import_gazepoint_pyhrv_results

Import pyHRV-style results

Description

Import pyHRV-style results

Usage

```
import_gazepoint_pyhrv_results(path)
```

Arguments

path	Input path. Use .json or .rds.

Value

Imported object.

impute_gazepoint_missing

Impute missing values in Gazepoint signals

Description

Interpolates missing values in numeric Gazepoint time series, such as pupil, GSR/EDA, PPG/BVP, heart-rate, IBI/RRI, or other continuous channels. The function can work on a numeric vector, a time-series object, or selected numeric columns of a data frame.

Usage

```
impute_gazepoint_missing(
  data,
  method = c("linear", "locf", "nocb", "nearest", "constant"),
  cols = NULL,
  time_col = NULL,
  group_cols = NULL,
  max_gap = Inf,
  fill_edges = TRUE,
  constant_value = 0,
  add_flags = TRUE,
  treat_infinite_as_missing = TRUE
)
```

Arguments

data	Numeric vector, time-series object, or data frame.
method	Imputation method: "linear", "locf", "nocb", "nearest", or "constant".
cols	Columns to impute when data is a data frame. If NULL, all numeric columns except time and grouping columns are used.
time_col	Optional time column for interpolation.
group_cols	Optional grouping columns. Imputation is performed within groups.
max_gap	Maximum missing-gap length, in samples, to impute. Longer gaps remain missing. Defaults to Inf.
fill_edges	If TRUE, leading and trailing gaps are filled using the nearest observed value for methods that support it.
constant_value	Value used when method = "constant".
add_flags	If TRUE and data is a data frame, add logical <column>_was_imputed columns.
treat_infinite_as_missing	If TRUE, infinite values are treated as missing before imputation.

Value

Object of the same basic type as data. Data-frame outputs include an imputation_summary attribute.

Examples

```
x <- c(1, NA, 3, 4)
impute_gazepoint_missing(x)

dat <- data.frame(time_s = 1:5, GSR = c(1, NA, 3, NA, 5))
impute_gazepoint_missing(dat, cols = "GSR", time_col = "time_s")
```

---

interpolate_gazepoint_pupil_blinks

Interpolate Gazepoint pupil blink/dropout spans

Description

Interpolates internal blink or dropout spans in pupil columns using transparent, auditable rules. Leading/trailing gaps are not interpolated.

Usage

```
interpolate_gazepoint_pupil_blinks(
  data,
  pupil_cols = NULL,
  time_col = NULL,
  blink_col = NULL,
  max_gap_s = NULL,
  method = c("linear", "constant"),
  suffix = "_interp"
)
```

Arguments

`data`	Data frame containing pupil columns.
`pupil_cols`	Pupil columns to interpolate. If omitted, common pupil columns are detected.
`time_col`	Optional time column.
`blink_col`	Optional logical/numeric blink mask column.
`max_gap_s`	Optional maximum interpolated gap duration in seconds.
`method`	Interpolation method passed to `stats::approx()`.
`suffix`	Suffix for interpolated pupil columns.

Value

Data frame with interpolated pupil columns and audit flags.

`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.

`match_gazepoint_events_to_biometrics` *Match events to biometric windows*

Description

Aligns event logs to Gazepoint biometric or eye-tracking data and returns either sample-level event windows or event-level summary features.

Usage

```
match_gazepoint_events_to_biometrics(
  data,
  events,
  pre = 0,
  post = 5,
  time_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  summary_cols = NULL,
  return = c("windows", "summary")
)
```

Arguments

`data`	Biometric or eye-tracking data frame.
`events`	Event timestamps, event data frame, or event-log path.
`pre`	Seconds before event onset.
`post`	Seconds after event onset.
`time_col`	Time column in `data`.
`event_time_col`	Event-time column in `events`.
`event_id_col`	Event identifier column in `events`.
`summary_cols`	Numeric columns to summarize when `return = "summary"`.
`return`	`"windows"` or `"summary"`.

Value

Data frame of sample-level windows or event-level summaries.

merge_gazepoint_recordings_pspm_style

Merge multiple Gazepoint recordings PsPM-style

Description

Merge multiple Gazepoint recordings PsPM-style

Usage

```
merge_gazepoint_recordings_pspm_style(
  recordings,
  time_col = NULL,
  gap_seconds = 1,
  recording_col = "pspm_recording",
  reset_first_time = TRUE
)
```

Arguments

recordings	List of data frames.
time_col	Time column.
gap_seconds	Gap inserted between recordings.
recording_col	Output recording-id column.
reset_first_time	If TRUE, each input time starts from zero before offsetting.

Value

Merged data frame.

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_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.

---

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.

---

normalize_gazepoint_scr

Normalize Gazepoint SCR amplitudes

Description

Normalizes SCR amplitudes using z-scores, percent of maximum, min-max range, centered scores, log-z scores, or no transformation.

Usage

```
normalize_gazepoint_scr(
  amplitudes,
  method = c("z", "percent_max", "range", "center", "log_z", "none"),
  amplitude_col = NULL,
  group_cols = NULL,
  output_col = "scr_amplitude_normalized",
  na.rm = TRUE
)
```

Arguments

amplitudes	Numeric vector or data frame.
method	Normalization method: "z", "percent_max", "range", "center", "log_z", or "none".
amplitude_col	Column to normalize when amplitudes is a data frame.
group_cols	Optional grouping columns for subject-specific normalization.
output_col	Name of normalized column for data-frame input.
na.rm	If TRUE, ignore missing values when computing normalization constants.

Value

Numeric vector for vector input, or data frame for data-frame input.

Examples

```
normalize_gazepoint_scr(c(0.1, 0.2, 0.3), method = "percent_max")
```

`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`.

`physiology_validation_ladder` *Build a multimodal physiology validation ladder*

Description

Build a multimodal physiology validation ladder

Usage

```
physiology_validation_ladder(
  acquisition_qc = "not_assessed",
  analytical_qc = "not_assessed",
  construct_check = "not_assessed",
  within_person = "not_assessed",
  held_out_person = "not_assessed",
  claim = "descriptive"
)
```

Arguments

`acquisition_qc, analytical_qc, construct_check, within_person, held_out_person`	Stage statuses: `pass`, `warning`, `fail`, or `not_assessed`.
`claim`	Claim type. `generalizable` requires held-out-person evidence.

Value

A structured validation result.

`pipeline_comparison_dashboard` *Create a static Gazepoint pipeline comparison dashboard*

Description

Aggregates participant-, session-, or row-level Gazepoint QC outputs into a compact static dashboard object. The function is intentionally conservative: it summarizes existing QC indicators, missingness rates, signal-quality scores, rule failures, exclusion flags, and audit notes without fitting models, making clinical claims, or launching an interactive dashboard.

Usage

```
pipeline_comparison_dashboard(
  data,
  participant_col = NULL,
  session_col = NULL,
  grouping_cols = NULL,
  missingness_col = NULL,
  quality_col = NULL,
  qc_status_col = NULL,
  failed_rules_col = NULL,
  excluded_col = NULL,
  notes_col = NULL
)
```

Arguments

`data`	A data frame containing QC, quality, missingness, exclusion, or audit-summary columns.
`participant_col`	Optional participant identifier column. If `NULL`, a common participant column is detected when available.
`session_col`	Optional session identifier column. If `NULL`, a common session column is detected when available.
`grouping_cols`	Optional character vector of grouping columns. If supplied, this overrides `participant_col` and `session_col`.
`missingness_col`	Optional numeric missingness-rate column.
`quality_col`	Optional numeric quality-score or SQI column.
`qc_status_col`	Optional QC status column.
`failed_rules_col`	Optional column containing failed rule labels.
`excluded_col`	Optional logical, numeric, or character exclusion flag column.
`notes_col`	Optional audit-note column.

Value

A list with class `gazepoint_pipeline_comparison_dashboard` containing an overall summary, grouped dashboard table, issue table, and detected column mapping.

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_cluster_null_distribution

Plot the cluster-permutation null distribution

Description

Plots the permutation null distribution of maximum cluster masses and, when available, overlays the observed cluster mass for one observed cluster.

Usage

```
plot_gazepoint_cluster_null_distribution(
  result,
  cluster_id = 1L,
  observed_mass = NULL,
  bins = 30L
)
```

Arguments

result	Object returned by run_gazepoint_cluster_permutation().
cluster_id	Observed cluster to overlay.
observed_mass	Optional observed cluster mass. If NULL, the function attempts to extract the mass from result.
bins	Number of histogram bins.

Value

A ggplot object.

`plot_gazepoint_cluster_permutation` *Plot Gazepoint cluster permutation results*

Description

Plots condition-level time courses and highlights clusters that pass the cluster-level alpha threshold. The shaded regions are descriptive aids only; they should not be interpreted as precise effect onset or offset estimates.

Usage

```
plot_gazepoint_cluster_permutation(x, alpha = NULL, show_all_clusters = FALSE)
```

Arguments

`x`	Object returned by `run_gazepoint_cluster_permutation()`.
`alpha`	Optional cluster-level alpha. If `NULL`, the alpha stored in the object is used.
`show_all_clusters`	Logical. If `FALSE`, only clusters with `p <= alpha` are shaded.

Value

A ggplot object.

`plot_gazepoint_design_coverage` *Plot Gazepoint design and event-coverage audits*

Description

Create compact diagnostic plots from design, condition-balance, or event-coverage audit objects.

Usage

```
plot_gazepoint_design_coverage(
  audit,
  type = c("condition_counts", "participant_trials", "event_coverage", "warnings")
)
```

Arguments

audit	Object returned by audit_gazepoint_experiment_design(), audit_gazepoint_event_coverage(), or audit_gazepoint_condition_balance().
type	Plot type. Supported values depend on the audit object: "condition_counts", "participant_trials", "event_coverage", and "warnings".

Value

A ggplot object.

---

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_export_profile`
Plot a Gazepoint export-folder profile

Description

Create compact plots for file extensions, detected column roles, mean missingness, or numeric signal activity.

Usage

```
plot_gazepoint_export_profile(
  profile,
  type = c("files", "roles", "missingness", "activity"),
  top_n = 20
)
```

Arguments

`profile`	Object returned by `profile_gazepoint_export_folder()`.
`type`	Plot type. One of `"files"`, `"roles"`, `"missingness"`, or `"activity"`.
`top_n`	Number of columns or files to show for selected plots.

Value

A `ggplot` object.

`plot_gazepoint_missingness`
Plot missingness across Gazepoint signal columns

Description

Creates a heatmap-style plot showing missing and observed samples across selected Gazepoint signal columns. This is a descriptive quality-control display and should not be interpreted as evidence of psychological or physiological state.

Usage

```
plot_gazepoint_missingness(
  data,
  cols = NULL,
  time_col = NULL,
  id_col = NULL,
  max_points = 5000L
)
```

Arguments

data	A data frame.
cols	Character vector naming columns to inspect. If NULL, numeric columns are used.
time_col	Optional column used for the x-axis. If NULL, row number is used.
id_col	Optional participant/session column used for faceting.
max_points	Maximum number of rows to plot. Larger data sets are evenly down-sampled for display only.

Value

A ggplot object.

Examples

```
d <- data.frame(
  time = 1:5,
  pupil_left = c(3, NA, 3.1, 3.2, NA),
  eda = c(1, 1.1, NA, 1.2, 1.3)
)
plot_gazepoint_missingness(d, cols = c("pupil_left", "eda"), time_col = "time")
```

---

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_ppg_breathing

Plot breathing-rate spectrum from Gazepoint RR/IBI intervals

Description

Plot breathing-rate spectrum from Gazepoint RR/IBI intervals

Usage

```
plot_gazepoint_ppg_breathing(
  rr_ms,
  rr_time_s = NULL,
  resample_hz = 4,
  breathing_band = c(0.1, 0.5)
)
```

Arguments

rr_ms	RR/IBI intervals in milliseconds.
rr_time_s	Optional interval timestamps.
resample_hz	RR interpolation frequency.
breathing_band	Breathing frequency band.

Value

Invisibly returns breathing-rate object.

plot_gazepoint_ppg_peak_detection

Plot Gazepoint pulse/PPG peak detection results

Description

Plot Gazepoint pulse/PPG peak detection results

Usage

```
plot_gazepoint_ppg_peak_detection(
  detection,
  group = NULL,
  accepted_only = FALSE
)
```

Arguments

detection	Detection object returned by detect_gazepoint_ppg_peaks().
group	Optional group to plot.
accepted_only	Whether to show only accepted peaks.

Value

Invisibly returns the plotted data.

---

plot_gazepoint_ppg_poincare

Plot a Poincare plot from Gazepoint pulse/PPG peaks or RR intervals

---

Description

Plot a Poincare plot from Gazepoint pulse/PPG peaks or RR intervals

Usage

```
plot_gazepoint_ppg_poincare(peaks = NULL, rr_ms = NULL, group_col = "group")
```

Arguments

peaks	Optional peak table.
rr_ms	Optional RR/IBI intervals in milliseconds.
group_col	Group column when peaks are supplied.

Value

Invisibly returns plotting data and Poincare summaries.

---

plot_gazepoint_ppg_segmentwise

Plot segmentwise Gazepoint pulse/PPG measures

Description

Plot segmentwise Gazepoint pulse/PPG measures

Usage

```
plot_gazepoint_ppg_segmentwise(segmentwise, measure = "bpm")
```

Arguments

segmentwise	Object returned by process_gazepoint_ppg_segmentwise().
measure	Measure column to plot.

Value

Invisibly returns plotted data.

---

plot_gazepoint_pyhrv_hr_heatplot

Plot pyHRV-style heart-rate heatplot

Description

Plot pyHRV-style heart-rate heatplot

Usage

```
plot_gazepoint_pyhrv_hr_heatplot(nni_ms, time_bins = 20, hr_bins = 20)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
time_bins	Number of time bins.
hr_bins	Number of heart-rate bins.

Value

Invisibly returns heatplot data.

---

plot_gazepoint_pyhrv_radar_chart

Plot pyHRV-style radar chart

Description

Plot pyHRV-style radar chart

Usage

```
plot_gazepoint_pyhrv_radar_chart(
  measures,
  columns = c("sdnn", "rmssd", "sdsd", "pnn50", "lf_norm", "hf_norm", "sd1", "sd2")
)
```

Arguments

measures	Named numeric vector or one-row data frame.
columns	Optional columns to plot.

Value

Invisibly returns plotted values.

---

plot_gazepoint_pyhrv_tachogram

Plot pyHRV-style tachogram

Description

Plot pyHRV-style tachogram

Usage

```
plot_gazepoint_pyhrv_tachogram(nni_ms, time_s = NULL)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
time_s	Optional time vector.

Value

Invisibly returns plotted data.

---

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_signal_quality` *Plot signal-quality diagnostics*

Description

Creates a lightweight ggplot diagnostic for signal-quality metrics or labels. The plot is intended for quality control and audit reporting only.

Usage

```
plot_gazepoint_signal_quality(
  quality,
  metric = "prop_missing",
  x = NULL,
  colour = NULL,
  facet = NULL
)
```

Arguments

`quality`	A data frame returned by `compute_gazepoint_signal_quality()` or `classify_gazepoint_signal_q`
`metric`	Metric to plot. Use `"quality_label"` for label counts.
`x`	Optional x-axis column. If omitted, the function uses the first available column among participant, participant_id, trial, trial_id, segment, segment_id, session, or signal.
`colour`	Optional colour/grouping column.
`facet`	Optional faceting column. Defaults to `"signal"` when available.

Value

A ggplot object.

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.

ppg_topology_features *Experimental topology-aware PPG morphology features*

Description

Uses H0 persistence lifetimes, represented by Euclidean minimum-spanning-tree edge lengths over a delay embedding. The output is explicitly a structural waveform descriptor, not a direct physiological surrogate.

Usage

```
ppg_topology_features(signal, delay = 2L, dimension = 3L, max_points = 300L)
```

Arguments

signal	Numeric PPG signal.
delay	Embedding delay in samples.
dimension	Embedding dimension.
max_points	Maximum embedded points used by the diagnostic.

Value

A named list of topology-aware structural summaries.

---

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_bids_eye` *Prepare an eye-tracking-only BIDS export*

Description

A modality-specific convenience wrapper around `export_gazepoint_to_bids()`. Arguments are forwarded without changing the unified export contract. When the unified exporter exposes modality switches, they are set to eye-tracking mode automatically.

Usage

```
prepare_gazepoint_bids_eye(data, ..., execute = TRUE)
```

Arguments

`data`	Gazepoint eye-tracking data accepted by `export_gazepoint_to_bids()`.
`...`	Additional arguments forwarded to `export_gazepoint_to_bids()`.
`execute`	If `FALSE`, return the resolved call specification without writing files.

Value

The result of `export_gazepoint_to_bids()` or a dry-run specification.

`prepare_gazepoint_bids_physio` *Prepare a physiology-only BIDS export*

Description

A modality-specific convenience wrapper around `export_gazepoint_to_bids()`. Arguments are forwarded without changing the unified export contract. When the unified exporter exposes modality switches, they are set to physiology mode automatically.

Usage

```
prepare_gazepoint_bids_physio(data, ..., execute = TRUE)
```

Arguments

`data`	Gazepoint physiological data accepted by `export_gazepoint_to_bids()`.
`...`	Additional arguments forwarded to `export_gazepoint_to_bids()`.
`execute`	If `FALSE`, return the resolved call specification without writing files.

Value

The result of `export_gazepoint_to_bids()` or a dry-run specification.

`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_biosppy_input

Prepare Gazepoint signals for Python BioSPPy

Description

Converts Gazepoint EDA/GSR or PPG/BVP waveform data into grouped numeric vectors suitable for transfer to Python BioSPPy. The function validates the timebase, records missing-data handling, and can write headerless one-column CSV files.

Usage

```
prepare_gazepoint_biosppy_input(
  data,
  signal_type = c("auto", "eda", "ppg"),
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  missing = c("error", "interpolate", "segments"),
  irregular = c("error", "allow"),
  sampling_tolerance = 0.05,
```

```
  min_segment_samples = 3L,
  signal_units = NULL,
  output_dir = NULL,
  prefix = "gazepoint_biosppy",
  write_manifest = TRUE,
  overwrite = FALSE
)
```

Arguments

data	A numeric signal vector or a data frame containing EDA/GSR or PPG/BVP samples.
signal_type	Signal type: "auto", "eda", or "ppg". Numeric-vector input requires an explicit signal type.
signal_col	Signal column when data is a data frame. If NULL, common Gazepoint and biosignal column names are searched.
time_col	Optional numeric time column expressed in seconds. If absent, sampling_rate_hz must be supplied and a time vector is generated.
group_cols	Optional participant, session, trial, file, or other grouping columns.
sampling_rate_hz	Optional positive sampling frequency in hertz. When omitted, it is inferred separately for each group from time_col.
missing	Missing-signal handling: "error", "interpolate", or "segments".
irregular	Handling of irregular sampling intervals: "error" or "allow".
sampling_tolerance	Maximum relative deviation from the expected sample interval before an interval is marked irregular.
min_segment_samples	Minimum number of samples retained when missing = "segments".
signal_units	Optional descriptive signal unit, such as "microsiemens" or "arbitrary".
output_dir	Optional directory for one-column CSV files and a manifest.
prefix	Filename prefix used for exported files.
write_manifest	Logical. Write a manifest when output_dir is supplied.
overwrite	Logical. Permit replacement of existing output files.

Details

The function prepares data only. It does not invoke Python or BioSPPy and does not perform physiological interpretation.

BioSPPy signal functions assume a regularly sampled one-dimensional signal and a sampling frequency in hertz. When a time column is supplied, each group must be strictly increasing. Sampling irregularity is assessed against the supplied or inferred sampling rate.

With missing = "interpolate", non-finite signal values are replaced by linear interpolation within each group. Edge values use the nearest finite value. At least two finite samples are required.

With `missing = "segments"`, each contiguous finite run is exported as a separate vector. Runs shorter than `min_segment_samples` are retained in the audit table but excluded from the prepared vectors.

Exported signal files contain one numeric value per line without a header, quotation marks, or row names.

Value

An object of class `"gazepoint_biosppy_input"` containing:

- `samples`: row-level audit table;
- `vectors`: named Python-ready numeric signal vectors;
- `sampling_rates_hz`: sampling frequency for each vector;
- `manifest`: vector-level preparation summary;
- `files`: paths written to disk;
- `settings`: preparation settings and Python call templates.

See Also

`run_gazepoint_biosppy_eda()`, `run_gazepoint_biosppy_ppg()`

Examples

```
eda <- data.frame(
  participant = rep("P01", 4),
  time_s = c(0, 0.1, 0.2, 0.3),
  EDA = c(1.0, 1.1, 1.05, 1.2)
)

prepared <- prepare_gazepoint_biosppy_input(
  eda,
  signal_type = "eda",
  group_cols = "participant"
)

prepared$vectors$P01
prepared$sampling_rates_hz
```

`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_eyetrackingr_input
Prepare Gazepoint data for eyetrackingR

Description

Converts sample-level Gazepoint gaze and AOI data into a conservative, audited table compatible with `eyetrackingR::make_eyetrackingr_data()`.

Usage

```
prepare_gazepoint_eyetrackingr_input(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
  sampling_rate_hz = NULL,
  rezero_time = FALSE,
  trackloss_col = NULL,
  validity_col = NULL,
  valid_values = NULL,
  x_col = NULL,
  y_col = NULL,
  aoi_col = NULL,
  aoi_cols = NULL,
  aoi_levels = NULL,
 outside_aoi_values = c("", "none", "no_aoi", "outside", "outside_aoi", "non_aoi",
    "background"),
  allow_aoi_overlap = FALSE,
  item_cols = NULL,
  predictor_cols = NULL,
  treat_non_aoi_looks_as_missing = TRUE,
  sampling_tolerance = 0.05,
  irregular = c("error", "allow"),
  create_object = FALSE
)
```

Arguments

`data`	Sample-level Gazepoint data frame.
`participant_col`	Participant identifier column. If `NULL`, common participant names are searched.
`trial_col`	Trial identifier column. If `NULL`, common trial and stimulus names are searched.
`time_col`	Numeric time or sample-counter column. If `NULL`, common Gazepoint time columns are searched.

`time_unit` Source time unit: `"auto"`, `"seconds"`, `"milliseconds"`, or `"samples"`. Output time is always milliseconds.

`sampling_rate_hz` Sampling rate required when `time_unit = "samples"` or when an automatically detected sample counter is used.

`rezero_time` Logical. Subtract the minimum time separately within each participant-trial group.

`trackloss_col` Optional column where `TRUE` or non-zero means tracking was lost.

`validity_col` Optional gaze-validity column where `TRUE`, `"valid"`, or a positive numeric value means valid tracking.

`valid_values` Optional explicit values treated as valid in `validity_col`.

`x_col, y_col` Optional gaze-coordinate columns. Missing or non-finite coordinates are treated as track loss.

`aoi_col` Optional categorical AOI column.

`aoi_cols` Optional existing binary or logical AOI columns. Supply either `aoi_col` or `aoi_cols`, not both.

`aoi_levels` Optional ordered AOI labels to create from `aoi_col`.

`outside_aoi_values` Values in `aoi_col` treated as valid looks outside all supplied AOIs rather than track loss.

`allow_aoi_overlap` Logical. Permit more than one AOI column to be `TRUE` in a sample.

`item_cols` Optional item identifier columns retained in the output and passed to `eyetrackingR::make_eyetracking`

`predictor_cols` Optional condition or predictor columns retained in the compatibility table.

`treat_non_aoi_looks_as_missing` Logical passed unchanged to `eyetrackingR::make_eyetrackingr_data()` when `create_object = TRUE`.

`sampling_tolerance` Maximum relative deviation from the median within-trial sampling interval.

`irregular` Handling of irregular within-trial sampling: `"error"` or `"allow"`.

`create_object` Logical. If `TRUE`, construct an actual `eyetrackingR_data` object. The optional eyetrackingR package must then be installed.

Details

The returned compatibility table uses the standardized columns `ParticipantName`, `Trial`, `Time_ms`, and `TrackLoss`. `Time_ms` is expressed in milliseconds. AOI columns are logical.

Track loss is derived conservatively from the union of an explicit track-loss flag, an invalid validity flag, and missing or non-finite gaze coordinates. A valid look outside every AOI is not silently reclassified as hardware track loss.

eyetrackingR is intended for relatively raw sample-level data in which rows represent equally spaced time samples. Fixation-level or event-level tables should not be supplied to this helper.

Value

An object of class "gazepoint_eyetrackingr_input" containing:

- data: plain compatibility data frame;
- object: optional output from make_eyetrackingr_data();
- row_audit: row-level derivation and ordering audit;
- sampling: participant-trial sampling audit;
- manifest: column mappings and preparation summary;
- settings: resolved preparation settings.

See Also

assign_gazepoint_aoi()

Examples

```
gaze <- data.frame(
  participant = rep("P01", 4),
  trial = rep("T01", 4),
  time_s = c(0, 0.1, 0.2, 0.3),
  gaze_x = c(0.2, 0.5, 0.8, NA),
  gaze_y = c(0.5, 0.5, 0.5, NA),
  AOI = c("left", "center", "right", NA)
)

prepared <- prepare_gazepoint_eyetrackingr_input(gaze)
prepared$data
prepared$sampling
```

---

prepare_gazepoint_gazer_input

Prepare Gazepoint data for gazeR

Description

Converts long-form sample-level Gazepoint gaze and pupil data into a conservative, audited table compatible with gazer::make_gazer().

Usage

```
prepare_gazepoint_gazer_input(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
```

```
  sampling_rate_hz = NULL,
  rezero_time = FALSE,
  x_col = NULL,
  y_col = NULL,
  x_left_col = NULL,
  y_left_col = NULL,
  x_right_col = NULL,
  y_right_col = NULL,
  pupil_col = NULL,
  pupil_left_col = NULL,
  pupil_right_col = NULL,
  validity_col = NULL,
  validity_left_col = NULL,
  validity_right_col = NULL,
  valid_values = NULL,
  blink_col = NULL,
  blink_left_col = NULL,
  blink_right_col = NULL,
  invalid_coordinate_values = NULL,
  invalid_pupil_values = NULL,
  mask_invalid = FALSE,
  other_cols = NULL,
  sampling_tolerance = 0.05,
  irregular = c("error", "allow"),
  create_object = FALSE
)
```

Arguments

`data`	Sample-level Gazepoint data frame.
`participant_col`	Participant identifier column. If `NULL`, common participant names are searched.
`trial_col`	Trial identifier column. If `NULL`, common trial, media, and stimulus names are searched.
`time_col`	Numeric time or sample-counter column. If `NULL`, common Gazepoint time columns are searched.
`time_unit`	Source time unit: `"auto"`, `"seconds"`, `"milliseconds"`, or `"samples"`. Output time is expressed in milliseconds.
`sampling_rate_hz`	Sampling rate required when time is represented by sample indices.
`rezero_time`	Logical. Subtract the minimum time separately within each participant-trial group.
`x_col`, `y_col`	Optional monocular or already-combined gaze-coordinate columns.
`x_left_col`, `y_left_col`	Optional left-eye gaze-coordinate columns.
`x_right_col`, `y_right_col`	Optional right-eye gaze-coordinate columns.

`pupil_col` Optional monocular, cyclopean, or previously combined pupil column.

`pupil_left_col`, `pupil_right_col`
Optional left- and right-eye pupil columns.

`validity_col` Optional shared gaze/pupil validity column.

`validity_left_col`, `validity_right_col`
Optional per-eye validity columns.

`valid_values` Optional explicit values treated as valid. Without this argument, positive numeric values, `TRUE`, and common textual valid labels are treated as valid.

`blink_col` Optional shared blink column.

`blink_left_col`, `blink_right_col`
Optional per-eye blink columns.

`invalid_coordinate_values`
Optional coordinate values to flag explicitly as invalid. Zero is not treated as invalid by default because it can be a valid screen-edge coordinate.

`invalid_pupil_values`
Optional pupil values to flag explicitly as invalid, for example `c(-1, 0)`.

`mask_invalid` Logical. If `TRUE`, explicitly invalid values, failed validity samples, and blink samples are replaced by `NA` in the prepared gaze and pupil columns. Non-finite values are always represented as `NA`.

`other_cols` Optional item, condition, block, AOI, stimulus, or other metadata columns retained unchanged.

`sampling_tolerance`
Maximum relative deviation from the median within-trial sampling interval.

`irregular` Handling of irregular within-trial sampling: `"error"` or `"allow"`.

`create_object` Logical. If `TRUE`, call `make_gazer()` from a locally installed gazeR package. gazeR is GitHub-hosted and is therefore not a declared gpbiometrics dependency.

Details

The standardized identifier columns are `subject`, `trial`, and `time`. `time` is expressed in milliseconds.

Monocular or combined input uses the canonical columns `x`, `y`, and `pupil`. Binocular input uses `x_left`, `y_left`, `pupil_left`, `x_right`, `y_right`, and `pupil_right` as available. gazeR retains multiple selected eye columns when constructing its compatibility table.

The helper does not assign AOIs, calculate track loss, detect or extend blinks, interpolate data, smooth signals, downsample, upsample, baseline-correct pupil size, or run inferential analyses.

Value

An object of class `"gazepoint_gazer_input"` containing:

- `data`: plain gazeR-compatible long-form data;
- `object`: optional output from `make_gazer()`;
- `row_audit`: row-level availability and invalidity audit;

- `sampling`: participant-trial sampling audit;
- `manifest`: column mappings and preparation summary;
- `settings`: resolved preparation settings.

See Also

`assign_gazepoint_aoi()`, `downsample_gazepoint_data()`, `detect_gazepoint_pupil_blinks()`, `clean_gazepoint_pupil_signal()`

Examples

```
gaze <- data.frame(
  participant = rep("P01", 4),
  trial = rep("T01", 4),
  time_s = c(0, 0.1, 0.2, 0.3),
  gaze_x = c(0.2, 0.4, 0.6, NA),
  gaze_y = c(0.5, 0.5, 0.5, NA),
  pupil_left = c(3.1, 3.2, 3.3, NA),
  pupil_right = c(3.0, 3.1, 3.2, NA)
)

prepared <- prepare_gazepoint_gazer_input(gaze)
prepared$data
prepared$sampling
```

---

prepare_gazepoint_heartpy_input

Prepare Gazepoint pulse/PPG data for HeartPy-style workflows

Description

Creates a compact table with time, signal, and optional grouping columns from Gazepoint Biometrics exports. No files are written unless output_dir is supplied.

Usage

```
prepare_gazepoint_heartpy_input(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  output_dir = NULL,
  prefix = "gazepoint_heartpy"
)
```

Arguments

data	Data frame containing Gazepoint biometric samples.
signal_col	Pulse/PPG signal column. If NULL, a likely column is inferred.
time_col	Time column in seconds. If NULL, a likely column is inferred or created from sampling_rate_hz.
group_cols	Optional grouping columns such as participant or trial identifiers.
sampling_rate_hz	Sampling rate in Hz. Required if time_col cannot be inferred.
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 signal_table, sampling_rate_hz, group_summary, and path.

---

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_mixed_model_data

Prepare Gazepoint data for mixed-model analysis

Description

Lightweight mixed-model preparation helper for Gazepoint-derived trial-level, event-locked, AOI, or physiology summaries. It coerces identifier and condition columns to factors, optionally drops rows with missing outcomes, and adds centered or standardized numeric predictors.

Usage

```
prepare_gazepoint_mixed_model_data(
  data,
  outcome_cols = NULL,
  participant_col = NULL,
  trial_col = NULL,
  condition_cols = NULL,
  factor_cols = NULL,
  numeric_cols = NULL,
  center_numeric = TRUE,
  scale_numeric = FALSE,
  drop_missing_outcomes = TRUE,
  ...
)
```

Arguments

data	Data frame to prepare.
outcome_cols	Outcome columns used for optional missing-row removal.
participant_col	Optional participant identifier column.
trial_col	Optional trial/item identifier column.
condition_cols	Optional condition columns.
factor_cols	Additional columns to coerce to factors.
numeric_cols	Numeric predictors to center or scale. If omitted, all numeric columns except outcomes are considered.
center_numeric	If TRUE, add centered numeric columns using suffix "_c".
scale_numeric	If TRUE, add standardized numeric columns using suffix "_z".
drop_missing_outcomes	If TRUE, remove rows with missing/non-finite values in outcome_cols.
...	Reserved for future extensions.

Value

Data frame of class gazepoint_mixed_model_data.

---

prepare_gazepoint_mne_events

Prepare Gazepoint events for MNE-Python

Description

Creates an integer MNE-style event matrix with columns representing sample number, previous value, and event code. Input may be an event table, numeric event-time vector, or continuous table containing marker columns.

Usage

```
prepare_gazepoint_mne_events(
  events,
  event_time_col = NULL,
  event_label_col = NULL,
  event_code_col = NULL,
  marker_cols = NULL,
  participant_col = NULL,
  trial_col = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
  sampling_rate_hz,
  recording_start_s = 0,
  first_samp = 0L,
```

```
  event_id = NULL,
  previous_value = 0L,
  marker_onset = c("change", "nonzero"),
  duplicate = c("error", "allow"),
  export_csv = NULL
)
```

Arguments

`events`	Numeric event-time vector or data frame.
`event_time_col`	Event-time column for event-table input.
`event_label_col`	Optional event-label column.
`event_code_col`	Optional positive integer event-code column.
`marker_cols`	Optional marker or TTL columns in continuous input.
`participant_col`	Optional participant column retained in the audit.
`trial_col`	Optional trial column retained in the audit.
`time_unit`	Time unit: `"auto"`, `"seconds"`, `"milliseconds"`, or `"samples"`.
`sampling_rate_hz`	Positive sampling frequency used to convert event times to MNE sample numbers.
`recording_start_s`	Recording start in seconds in the same clock domain as the converted event times.
`first_samp`	Integer MNE first-sample offset.
`event_id`	Optional named integer event dictionary or data frame with `event_label` and `event_code`.
`previous_value`	Integer previous-event value placed in the second event-matrix column.
`marker_onset`	Whether marker input creates events only on value changes or for every nonzero row.
`duplicate`	Handling of repeated event sample numbers.
`export_csv`	Optional path for a headerless three-column event file.

Value

A `"gazepoint_mne_events"` object containing the event matrix, event table, event dictionary, audit, and settings.

prepare_gazepoint_mne_input *Prepare Gazepoint channels for an MNE RawArray*

Description

Produces a channel-by-sample matrix and metadata specifications suitable for `mne.create_info()` and `mne.io.RawArray()`. No Python execution is required.

Usage

```
prepare_gazepoint_mne_input(
  data,
  channel_cols = NULL,
  channel_names = NULL,
  channel_types = NULL,
  time_col = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
  sampling_rate_hz = NULL,
  first_samp = 0L,
  scale_factors = NULL,
  missing = c("error", "allow"),
  irregular = c("error", "allow"),
  sampling_tolerance = 0.05
)
```

Arguments

`data`	One continuous sample-level data frame.
`channel_cols`	Numeric signal columns. If omitted, common Gazepoint gaze, pupil, physiology, marker, and temperature columns are detected.
`channel_names`	Optional MNE channel names.
`channel_types`	Optional MNE channel types, either parallel to `channel_cols` or named by source column.
`time_col`	Numeric time column.
`time_unit`	Time unit.
`sampling_rate_hz`	Optional sampling frequency. If omitted it is inferred from the median interval.
`first_samp`	Integer MNE first-sample offset.
`scale_factors`	Optional explicit numeric factors applied to channels.
`missing`	Handling of non-finite signal values.
`irregular`	Handling of irregular sampling.
`sampling_tolerance`	Maximum relative interval deviation.

Value

A "gazepoint_mne_input" object.

`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_w` 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_pupillometryr_input

Prepare Gazepoint pupil data for PupillometryR

Description

Converts sample-level Gazepoint pupil data into a conservative, audited long-form table compatible with PupillometryR::make_pupillometryr_data().

Usage

```
prepare_gazepoint_pupillometryr_input(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  condition_col = NULL,
  pupil_left_col = NULL,
  pupil_right_col = NULL,
  pupil_col = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
  sampling_rate_hz = NULL,
  rezero_time = FALSE,
  invalid_pupil_values = NULL,
  validity_cols = NULL,
  valid_values = NULL,
  blink_cols = NULL,
  mask_invalid = FALSE,
  create_mean_pupil = TRUE,
  other_cols = NULL,
  sampling_tolerance = 0.05,
  irregular = c("error", "allow"),
  create_object = FALSE
)
```

Arguments

data	Sample-level Gazepoint pupil data frame.
participant_col	Participant identifier column. If NULL, common Gazepoint participant names are searched.
trial_col	Trial identifier column. If NULL, common trial, media, and stimulus names are searched.
time_col	Numeric time or sample-counter column. If NULL, common Gazepoint time columns are searched.
condition_col	Experimental-condition column. If NULL, common condition names are searched.
pupil_left_col	Optional left-pupil column.
pupil_right_col	Optional right-pupil column.
pupil_col	Optional single, cyclopean, or previously averaged pupil column. Supply either pupil_col or left/right columns, not both.
time_unit	Source time unit: "auto", "seconds", "milliseconds", or "samples". The prepared Time column is expressed in milliseconds.
sampling_rate_hz	Sampling rate required when time is represented by sample indices.
rezero_time	Logical. Subtract the first time separately within each participant-trial group.

`invalid_pupil_values`
: Optional pupil values to flag explicitly as invalid, for example `c(-1, 0)`.

`validity_cols`
: Optional pupil-validity columns. Supply one column to apply it to all pupil channels or one column per pupil channel.

`valid_values`
: Optional explicit values treated as valid in `validity_cols`. Without this argument, positive numeric values, `TRUE`, and common textual valid labels are treated as valid.

`blink_cols`
: Optional blink columns. Supply one column to apply it to all pupil channels or one column per pupil channel.

`mask_invalid`
: Logical. When `TRUE`, samples flagged by explicit invalid values, failed validity, or blink columns are replaced with `NA` in the prepared pupil columns. Non-finite pupil values are always represented as `NA`.

`create_mean_pupil`
: Logical. Create `Pupil_Mean` when both left and right pupil columns are available.

`other_cols`
: Optional additional condition, item, stimulus, block, or metadata columns retained unchanged.

`sampling_tolerance`
: Maximum relative deviation from the median within-trial sampling interval.

`irregular`
: Handling of irregular sampling: `"error"` or `"allow"`.

`create_object`
: Logical. If `TRUE`, construct an actual PupillometryR object. The optional PupillometryR package must then be installed.

Details

The standardized compatibility columns are `Subject`, `Trial`, `Time`, and `Condition`. `Time` is expressed in milliseconds. Pupil columns are named `Pupil_Left`, `Pupil_Right`, `Pupil`, and, when requested and available, `Pupil_Mean`.

The helper does not detect blinks, interpolate gaps, filter pupil signals, remove trials, baseline-correct pupil size, or run inferential analyses. Existing blink and validity columns are preserved as audit information and are used for masking only when `mask_invalid = TRUE`.

Value

An object of class `"gazepoint_pupillometryr_input"` containing:

- `data`: plain PupillometryR-compatible long-form data;
- `object`: optional PupillometryR object;
- `row_audit`: row-level pupil availability and flag audit;
- `sampling`: participant-trial sampling audit;
- `manifest`: column mappings and preparation summary;
- `settings`: resolved preparation settings.

See Also

`detect_gazepoint_pupil_blinks()`, `clean_gazepoint_pupil_signal()`, `baseline_correct_gazepoint_pupil()`

Examples

```
pupil <- data.frame(
  participant = rep("P01", 4),
  trial = rep("T01", 4),
  condition = rep("target", 4),
  time_s = c(0, 0.1, 0.2, 0.3),
  pupil_left = c(3.1, 3.2, NA, 3.3),
  pupil_right = c(3.0, 3.1, NA, 3.2)
)

prepared <- prepare_gazepoint_pupillometryr_input(pupil)
prepared$data
prepared$sampling
```

---

prepare_gazepoint_pyhrv_input

Prepare Gazepoint intervals for Python pyHRV

Description

Converts Gazepoint inter-beat or RR intervals into millisecond NN-interval vectors suitable for transfer to Python pyHRV workflows. The function does not invoke Python or pyHRV.

Usage

```
prepare_gazepoint_pyhrv_input(
  data,
  ibi_col = NULL,
  group_cols = NULL,
  unit = c("auto", "milliseconds", "seconds"),
  filter = c("none", "plausible"),
  min_nni_ms = 300,
  max_nni_ms = 2000,
  collapse_repeated_intervals = FALSE,
  repeated_tolerance_ms = 1e-08,
  output_dir = NULL,
  prefix = "gazepoint_pyhrv",
  write_manifest = TRUE,
  overwrite = FALSE
)
```

Arguments

data	A numeric interval vector or a data frame containing an IBI, RR, or NN-interval column.
ibi_col	Interval column when data is a data frame. If NULL, common Gazepoint, RR, and NNI column names are searched.

group_cols	Optional participant, session, trial, or file columns. One pyHRV-ready vector is produced per group.
unit	Input interval unit: "auto", "milliseconds", or "seconds".
filter	Which intervals should be included in the pyHRV vectors: "none" retains all finite positive intervals, whereas "plausible" also applies min_nni_ms and max_nni_ms.
min_nni_ms	Minimum plausible NN interval in milliseconds.
max_nni_ms	Maximum plausible NN interval in milliseconds.
collapse_repeated_intervals	Logical. If TRUE, consecutive interval values equal within repeated_tolerance_ms are represented once. This can be useful for sample-level Gazepoint exports in which the current IBI is repeated across multiple rows.
repeated_tolerance_ms	Non-negative tolerance used to identify consecutive repeated intervals.
output_dir	Optional directory in which one-column CSV files and a manifest are written.
prefix	Filename prefix used when output_dir is supplied.
write_manifest	Logical. If TRUE and output_dir is supplied, write a group-level manifest CSV.
overwrite	Logical. If FALSE, existing output files are protected.

Details

Input rows are retained in an auditable interval table. Missing, non-positive, implausible, and repeated intervals are flagged explicitly rather than removed silently.

Automatic unit assessment first examines the interval-column name. Names containing common millisecond markers are interpreted as milliseconds. Otherwise, the median positive interval is used: values no greater than 10 are interpreted as seconds, and values at least 100 as milliseconds. Intermediate values are considered ambiguous and require an explicit unit.

CSV interval files contain one numeric millisecond value per line, without row names, quotation marks, or a header. They can therefore be read into Python as a one-dimensional numeric vector.

Repeated-interval collapsing is optional because physiologically genuine adjacent intervals may occasionally have identical values. The setting should be chosen according to the structure of the source export.

Value

An object of class "gazepoint_pyhrv_input" containing:

- intervals: auditable row-level interval table;
- vectors: named list of pyHRV-ready numeric millisecond vectors;
- manifest: group-level interval and exclusion summary;
- files: paths written when output_dir is supplied;
- settings: complete preparation settings.

See Also

run_gazepoint_pyhrv_style(), check_gazepoint_pyhrv_interval(), prepare_gazepoint_rhrv_input()

Examples

```
ibi <- data.frame(
  participant = c("P01", "P01", "P01"),
  IBI_clean_ms = c(800, 810, 790)
)

prepared <- prepare_gazepoint_pyhrv_input(
  ibi,
  group_cols = "participant"
)

prepared$vectors$P01
prepared$manifest
```

---

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.

---

prepare_gazepoint_timecourse_test_data

Prepare Gazepoint time-course data for cluster permutation testing

---

Description

Prepares a participant-level, condition-level time-course data set for a conservative two-condition cluster-based permutation prototype. The helper aggregates repeated observations to one value per participant, condition, and time bin, then checks that the resulting data form a complete within-subject time grid.

Usage

```
prepare_gazepoint_timecourse_test_data(
  data,
  outcome_col,
  time_col,
  condition_col,
  participant_col,
  condition_a = NULL,
  condition_b = NULL,
  time_bin_width = NULL,
  aggregation = c("mean", "median"),
  require_complete = TRUE
)
```

Arguments

`data`	A data frame.
`outcome_col`	Name of the numeric outcome column.
`time_col`	Name of the numeric time column.
`condition_col`	Name of the condition column.
`participant_col`	Name of the participant identifier column.
`condition_a`	Optional first condition level.
`condition_b`	Optional second condition level.
`time_bin_width`	Optional numeric time-bin width. If supplied, time is binned using `floor(time / time_bin_width) * time_bin_width`.
`aggregation`	Aggregation rule for repeated rows within participant, condition, and time. Currently `"mean"` or `"median"`.
`require_complete`	Logical. If `TRUE`, require a complete participant by condition by time grid.

Details

This function is intended for exploratory time-course inference on already preprocessed Gazepoint-derived signals. It does not perform blink correction, artefact correction, baseline correction, filtering, or physiological interpretation.

Value

A data frame with columns `participant`, `condition`, `time`, and `value`, with class `gazepoint_timecourse_test_data`.

`preprocess_gazepoint_all` *Preprocess all available Gazepoint channels*

Description

Runs a conservative, beginner-friendly preprocessing sequence over a Gazepoint data frame or imported session list. Available channels are detected heuristically. Missing numeric signal gaps can be imputed, pupil blinks cleaned, and gaze samples filtered when the relevant functions and columns are available.

Usage

```
preprocess_gazepoint_all(
  data,
  impute_missing = TRUE,
  clean_pupil = TRUE,
  filter_gaze = TRUE,
  max_gap = 10,
  screen_bounds = c(0, 1, 0, 1),
  max_velocity = Inf,
  verbose = TRUE
)
```

Arguments

`data`	Data frame or list of data frames.
`impute_missing`	If TRUE, impute short missing gaps in numeric columns.
`clean_pupil`	If TRUE, clean detected pupil columns.
`filter_gaze`	If TRUE, filter detected gaze coordinates.
`max_gap`	Maximum gap length in samples for imputation.
`screen_bounds`	Screen bounds for gaze filtering.
`max_velocity`	Maximum gaze velocity for gaze filtering.
`verbose`	If TRUE, print a compact preprocessing log.

Value

A preprocessed object of the same basic structure as `data`, with a `preprocessing_log` attribute.

Examples

```
dat <- data.frame(time_s = 1:5, GSR = c(1, NA, 3, 4, 5))
preprocess_gazepoint_all(dat)
```

`preprocess_gazepoint_scr_pspm_style` *Preprocess Gazepoint SCR/GSR data PsPM-style*

Description

Preprocess Gazepoint SCR/GSR data PsPM-style

Usage

```
preprocess_gazepoint_scr_pspm_style(
  data,
  signal_col = NULL,
  time_col = NULL,
  sampling_rate_hz = NULL,
  range = c(0, 50),
  slope_limit_per_s = 10,
  clipping_tolerance = 1e-05,
  clipping_seconds = 0.5,
  min_valid_island_seconds = 1,
  artifact_epoch_seconds = 0.25,
  smoothing_seconds = 0.25
)
```

Arguments

`data`	Gazepoint data frame or numeric SCR/GSR signal.
`signal_col`	SCR/GSR column.
`time_col`	Time column.
`sampling_rate_hz`	Sampling rate for numeric input or missing time.
`range`	Valid signal range.
`slope_limit_per_s`	Maximum absolute slope per second.
`clipping_tolerance`	Difference threshold for flat clipping detection.
`clipping_seconds`	Minimum flat-run duration.
`min_valid_island_seconds`	Minimum valid island length.
`artifact_epoch_seconds`	Seconds to expand around detected artefacts.
`smoothing_seconds`	Smoothing window after artefact correction.

Value

List with processed signal, artifact table, summary, and settings.

`process_gazepoint_ppg_heartpy_style` *Run a full HeartPy-style Gazepoint pulse/PPG process*

Description

Run a full HeartPy-style Gazepoint pulse/PPG process

Usage

```
process_gazepoint_ppg_heartpy_style(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  clean_rr = TRUE,
  clean_rr_method = c("quotient", "iqr", "modified_z", "zscore", "none"),
  frequency_method = c("welch", "fft", "periodogram"),
  output_dir = NULL,
  ...
)
```

Arguments

`data`	Data frame or numeric pulse/PPG signal.
`signal_col`	Signal column when data is a data frame.
`time_col`	Optional time column.
`group_cols`	Optional grouping columns.
`sampling_rate_hz`	Sampling rate in Hz.
`clean_rr`	Whether to clean RR intervals after peak rejection.
`clean_rr_method`	RR cleaning method.
`frequency_method`	Frequency-domain method.
`output_dir`	Optional output directory. If NULL, no files are written.
`...`	Additional arguments passed to detect_gazepoint_ppg_peaks().

Value

A list with detection, peaks, measures, frequency, quality, report, and settings.

process_gazepoint_ppg_segmentwise *Process Gazepoint pulse/PPG data in overlapping HeartPy-style segments*

Description

Process Gazepoint pulse/PPG data in overlapping HeartPy-style segments

Usage

```
process_gazepoint_ppg_segmentwise(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  window_seconds = 60,
  overlap = 0.5,
  min_segment_seconds = 10,
  clean_rr = TRUE,
  clean_rr_method = c("quotient", "iqr", "modified_z", "zscore", "none"),
  frequency_method = c("welch", "fft", "periodogram"),
  ...
)
```

Arguments

data	Data frame or numeric pulse/PPG signal.
signal_col	Signal column when data is a data frame.
time_col	Time column.
group_cols	Optional grouping columns.
sampling_rate_hz	Sampling rate in Hz.
window_seconds	Segment length.
overlap	Segment overlap proportion.
min_segment_seconds	Minimum segment duration to process.
clean_rr	Whether to clean RR intervals.
clean_rr_method	RR cleaning method.
frequency_method	Frequency-domain method.
...	Additional arguments passed to detect_gazepoint_ppg_peaks().

Value

A list with segment table, peaks, measures, frequency, and settings.

`profile_gazepoint_export_folder` *Profile a Gazepoint export folder before analysis*

Description

Create a compact, auditable profile of a Gazepoint-style export folder before running a full workflow. The profiler inspects files, row and column counts, likely Gazepoint column roles, missingness, numeric signal activity, constant columns, all-zero columns, and read errors.

Usage

```
profile_gazepoint_export_folder(
  path,
  pattern = "\\.csv$",
  recursive = FALSE,
  max_files = Inf,
  max_rows = Inf,
  na.strings = c("", "NA", "NaN")
)
```

Arguments

`path`	Path to a folder containing Gazepoint-style export files.
`pattern`	File-name regular expression. Defaults to CSV files.
`recursive`	Logical. If `TRUE`, search subfolders recursively.
`max_files`	Maximum number of matching files to inspect.
`max_rows`	Maximum number of rows to read per file for profiling. Use `Inf` to read complete files.
`na.strings`	Character vector of strings to treat as missing values.

Value

A list with class `"gazepoint_export_folder_profile"` containing `overview`, `files`, `columns`, `warnings`, and `settings` tables.

Examples

```
demo_dir <- system.file(
  "extdata",
  "gazepoint_biometrics_kiosk_demo_exports",
  package = "gpbiometrics"
)

if (nzchar(demo_dir)) {
  profile <- profile_gazepoint_export_folder(demo_dir, max_files = 2)
  profile
}
```

---

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.

---

reconstruct_gazepoint_ppg_clipping

Detect and reconstruct clipped pulse/PPG samples

Description

Detect and reconstruct clipped pulse/PPG samples

Usage

```
reconstruct_gazepoint_ppg_clipping(
  x,
  near_max_prop = 0.02,
  flat_diff_prop = 0.001,
  min_run = 2L
)
```

Arguments

x
: Numeric pulse/PPG signal.

near_max_prop
: Proportion of the observed range used to define near-maximum samples.

flat_diff_prop
: Proportion of the observed range used to define near-flat differences.

min_run
: Minimum number of consecutive clipped samples.

Value

A list with signal, clipped, and runs.

`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.

`reject_gazepoint_ppg_peaks` *Reject implausible pulse/PPG peaks using RR-interval thresholds*

Description

Reject implausible pulse/PPG peaks using RR-interval thresholds

Usage

```
reject_gazepoint_ppg_peaks(
  peaks,
  group_col = "group",
  rr_tolerance = 0.3,
  min_rr_ms = 300
)
```

Arguments

`peaks`	Peak table returned by detect_gazepoint_ppg_peaks().
`group_col`	Group column.
`rr_tolerance`	Proportional RR tolerance around the group mean.
`min_rr_ms`	Minimum absolute tolerance in milliseconds.

Value

Peak table with accepted and rr_ms columns.

`remove_gazepoint_ppg_baseline_wander` *Remove baseline wander from Gazepoint pulse/PPG data*

Description

Remove baseline wander from Gazepoint pulse/PPG data

Usage

```
remove_gazepoint_ppg_baseline_wander(
  x,
  sampling_rate_hz,
  method = c("median", "mean"),
  window_seconds = 2
)
```

Arguments

x	Numeric signal.
sampling_rate_hz	Sampling rate in Hz.
method	Baseline method: median or mean.
window_seconds	Baseline window length.

Value

Baseline-corrected numeric signal.

---

report_gazepoint_cluster_permutation

Create conservative reporting text for a cluster-permutation result

---

Description

Generates cautious manuscript-ready wording for a cluster-permutation result. The wording avoids precise onset or offset claims and frames cluster timing as descriptive.

Usage

```
report_gazepoint_cluster_permutation(
  result,
  cluster_alpha = 0.05,
  digits = 3L,
  include_assumptions = TRUE
)
```

Arguments

result	Object returned by run_gazepoint_cluster_permutation().
cluster_alpha	Cluster-level alpha used to classify clusters.
digits	Number of digits used for p-values.
include_assumptions	Should an assumptions note be included?

Value

An object of class gazepoint_cluster_report.

`report_gazepoint_data_quality` *Report Gazepoint data quality*

Description

Generates dependency-free data-quality report files for Gazepoint data. The report summarizes missingness, numeric distributions, robust outlier counts, and simple QC plots. HTML and PDF outputs are created using base R only.

Usage

```
report_gazepoint_data_quality(
  data,
  output_dir = tempfile("gazepoint_quality_report_"),
  report_name = "gazepoint_data_quality",
  formats = c("html", "csv"),
  max_plot_columns = 6,
  open = FALSE
)
```

Arguments

`data`	Data frame or list of data frames.
`output_dir`	Output directory.
`report_name`	File prefix for report outputs.
`formats`	Character vector containing `"html"`, `"pdf"`, and/or `"csv"`.
`max_plot_columns`	Maximum number of numeric columns plotted per table.
`open`	If TRUE, open the HTML report interactively.

Value

Invisibly returns a list with output paths and summary tables.

Examples

```
## Not run:
report_gazepoint_data_quality(data.frame(time_s = 1:5, GSR = rnorm(5)))

## End(Not run)
```

respiration_from_ppg *Estimate respiration from PPG*

Description

Compatibility alias for `estimate_gazepoint_respiration_from_ppg()`. The returned value should be interpreted as an exploratory PPG-derived respiration-rate estimate, not as a direct respiratory measurement.

Usage

```
respiration_from_ppg(data, ...)
```

Arguments

`data`	Data frame or numeric PPG vector.
`...`	Additional arguments passed to `estimate_gazepoint_respiration_from_ppg()`.

Value

Output from `estimate_gazepoint_respiration_from_ppg()`.

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_biosppy_eda` *Run BioSPPy-style Gazepoint EDA processing*

Description

Run BioSPPy-style Gazepoint EDA processing

Usage

```
run_gazepoint_biosppy_eda(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  smoothing_seconds = 4
)
```

Arguments

data	Data frame or numeric EDA/GSR signal.
signal_col	EDA/GSR column.
time_col	Optional time column.
group_cols	Optional grouping columns.
sampling_rate_hz	Sampling rate in Hz.
smoothing_seconds	Window used for tonic smoothing.

Value

List with raw table, tonic, phasic, events, and recovery estimates.

---

run_gazepoint_biosppy_ppg *Run BioSPPy-style Gazepoint PPG/BVP processing*

Description

Run BioSPPy-style Gazepoint PPG/BVP processing

Usage

```
run_gazepoint_biosppy_ppg(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL
)
```

Arguments

data	Data frame or numeric PPG/BVP signal.
signal_col	PPG/BVP column.
time_col	Optional time column.
group_cols	Optional grouping columns.
sampling_rate_hz	Sampling rate in Hz.

Value

List with filtered signal, peaks, onsets, templates, and heart rate.

run_gazepoint_cluster_permutation *Run a conservative cluster-based permutation test for Gazepoint time courses*

Description

Runs a narrow within-subject, two-condition, one-dimensional cluster-based permutation test on participant-level Gazepoint-derived time courses.

Usage

```
run_gazepoint_cluster_permutation(
  data,
  outcome_col = "value",
  time_col = "time",
  condition_col = "condition",
  participant_col = "participant",
  design = "within",
  condition_a = NULL,
  condition_b = NULL,
  n_permutations = 1000,
  cluster_forming_alpha = 0.05,
  cluster_alpha = 0.05,
  tail = c("two.sided", "positive", "negative"),
  seed = NULL,
  time_bin_width = NULL,
  aggregation = c("mean", "median")
)
```

Arguments

data	A data frame, preferably returned by prepare_gazepoint_timecourse_test_data().
outcome_col	Name of the numeric outcome column.
time_col	Name of the numeric time column.
condition_col	Name of the condition column.
participant_col	Name of the participant identifier column.
design	Currently only "within" is supported.
condition_a	Optional first condition level. The tested difference is condition_a - condition_b.
condition_b	Optional second condition level.
n_permutations	Number of sign-flip permutations.
cluster_forming_alpha	Per-time-point alpha used only to form clusters.

cluster_alpha	Cluster-level alpha used for the significant flag.
tail	Test tail. Currently "two.sided", "positive", or "negative".
seed	Optional random seed.
time_bin_width	Optional time-bin width passed to prepare_gazepoint_timecourse_test_data() when the input has not already been prepared.
aggregation	Aggregation rule passed to prepare_gazepoint_timecourse_test_data() when needed.

Details

The function computes a paired t-statistic at each time point using participant-level condition differences, forms temporal clusters from adjacent suprathreshold time points, uses summed absolute t-statistics as cluster mass, and compares observed cluster masses with a sign-flip permutation null distribution.

Value

An object of class gazepoint_cluster_permutation.

Caution

This helper tests the global null of no condition difference anywhere in the tested time range. A significant cluster indicates evidence against that global null under the permutation scheme. It does not establish the precise onset, offset, latency, or physiological timing of an effect. Avoid wording such as "the effect starts at X ms". Prefer conservative wording such as "the cluster-based permutation test indicated a condition difference in the tested time course".

---

run_gazepoint_cluster_permutation_anova

Guardrail for cluster-permutation ANOVA

Description

This function is intentionally not implemented as a runnable inferential engine. It exists to make the unsupported scope explicit.

Usage

```
run_gazepoint_cluster_permutation_anova(...)
```

Arguments

...	Ignored.

Value

This function always errors.

run_gazepoint_cluster_permutation_covariate_adjusted

Guardrail for covariate-adjusted cluster permutation

Description

This function is intentionally not implemented as a runnable inferential engine.

Usage

```
run_gazepoint_cluster_permutation_covariate_adjusted(...)
```

Arguments

...	Ignored.

Value

This function always errors.

run_gazepoint_cluster_permutation_lmer

Guardrail for mixed-model cluster permutation

Description

This function is intentionally not implemented as a runnable inferential engine.

Usage

```
run_gazepoint_cluster_permutation_lmer(...)
```

Arguments

...	Ignored.

Value

This function always errors.

run_gazepoint_cluster_permutation_parallel

Guardrail for high-performance cluster permutation

Description

This function is intentionally not implemented as a separate backend.

Usage

```
run_gazepoint_cluster_permutation_parallel(...)
```

Arguments

...	Ignored.

Value

This function always errors.

run_gazepoint_cluster_threshold_sensitivity

Run threshold-sensitivity checks for cluster-permutation analysis

Description

Re-runs `run_gazepoint_cluster_permutation()` across several cluster-forming thresholds and summarizes whether the broad result is stable.

Usage

```
run_gazepoint_cluster_threshold_sensitivity(
  data,
  dv,
  time,
  condition,
  subject,
  thresholds = c(0.01, 0.025, 0.05, 0.1),
  cluster_alpha = 0.05,
  seed = NULL,
  ...
)
```

Arguments

data	A data frame.
dv	Numeric outcome column.
time	Time-bin column.
condition	Condition column.
subject	Participant identifier column.
thresholds	Numeric vector of cluster-forming thresholds.
cluster_alpha	Cluster-level alpha used for counting significant clusters in the summary.
seed	Optional seed. If provided, each threshold receives a deterministic seed offset.
...	Additional arguments passed to run_gazepoint_cluster_permutation().

Value

An object of class gazepoint_cluster_threshold_sensitivity.

---

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_heartpy_crosscheck` *Run a Gazepoint pulse/PPG cross-check against HeartPy when available*

Description

If Python HeartPy is available through reticulate, this function attempts to run heartpy.process() on the first group. Otherwise it returns native Gazepoint HeartPy-style results only.

Usage

```
run_gazepoint_heartpy_crosscheck(
  data,
  signal_col = NULL,
  time_col = NULL,
  group_cols = NULL,
  sampling_rate_hz = NULL,
  ...
)
```

Arguments

data	Data frame containing Gazepoint pulse/PPG samples.
signal_col	Signal column.
time_col	Optional time column.
group_cols	Optional grouping columns.
sampling_rate_hz	Sampling rate in Hz.
...	Additional arguments passed to detect_gazepoint_ppg_peaks().

Value

A list with native, heartpy, and heartpy_available.

run_gazepoint_multidimensional_cluster_permutation

Guardrail for multidimensional cluster permutation

Description

This function is intentionally not implemented as a runnable inferential engine.

Usage

```
run_gazepoint_multidimensional_cluster_permutation(...)
```

Arguments

...	Ignored.

Value

This function always errors.

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_pyhrv_style

Run pyHRV-style Gazepoint HRV analysis

Description

Run pyHRV-style Gazepoint HRV analysis

Usage

```
run_gazepoint_pyhrv_style(
  nni_ms = NULL,
  peaks = NULL,
  peak_time_col = "peak_time_s",
  time_unit = c("seconds", "milliseconds"),
  frequency_method = c("welch", "lomb", "ar")
)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
peaks	Optional peak timestamps/table. Used if nni_ms is missing.
peak_time_col	Peak timestamp column if peaks is a data frame.
time_unit	Unit for peaks.
frequency_method	Frequency method.

Value

List with time-domain, frequency-domain, nonlinear, and intervals.

---

run_gazepoint_real_data_smoke *Run privacy-safe smoke tests on private Gazepoint exports*

Description

Runs the standard gpbiometrics workflow, workflow summary, and workflow diagnostic on one or more private Gazepoint export folders. Raw data, participant-level values, source filenames, and private paths are never retained in the returned object.

Usage

```
run_gazepoint_real_data_smoke(
  data_dir = Sys.getenv("GPBIOMETRICS_SMOKE_DIR", unset = ""),
  output_dir = NULL,
  dataset_mode = c("subdirectories", "root"),
  pattern = "\\.csv$",
  recursive = TRUE,
  workflow_args = list(),
  diagnostic_args = list(),
  workflow_runner = NULL,
  summary_runner = NULL,
  diagnostic_runner = NULL,
  stop_on_error = FALSE,
  write_results = FALSE,
  overwrite = FALSE,
  protect_repository = TRUE
)
```

Arguments

data_dir	Private external data directory. Defaults to Sys.getenv("GPBIOMETRICS_SMOKE_DIR").
output_dir	Optional directory for aggregate smoke-test outputs. This should remain outside the package repository.
dataset_mode	How datasets are identified. "subdirectories" treats each immediate subdirectory containing matching files as one dataset. "root" treats data_dir as one dataset.
pattern	Regular expression used to identify candidate input files.
recursive	Search recursively within each dataset directory?
workflow_args	Named list of additional arguments passed to run_gazepoint_biometrics_workflow().
diagnostic_args	Named list of additional arguments passed to diagnose_gazepoint_biometrics_workflow().
workflow_runner	Optional workflow function. Defaults to run_gazepoint_biometrics_workflow(). This argument primarily supports controlled testing.
summary_runner	Optional summary function. Defaults to summarise_gazepoint_biometrics_workflow().
diagnostic_runner	Optional diagnostic function. Defaults to diagnose_gazepoint_biometrics_workflow().
stop_on_error	Stop after the first failed dataset?
write_results	Write aggregate CSV outputs to output_dir?
overwrite	Permit replacement of existing aggregate output files?
protect_repository	Reject private input or generated output directories located inside the current package repository?

Details

The source directory should normally be outside the package repository and supplied through the GPBIOMETRICS_SMOKE_DIR environment variable.

Value

An object of class "gazepoint_real_data_smoke" containing only privacy-safe aggregate results, condition summaries, runtime metadata, and non-sensitive settings.

Examples

```
private_dir <- Sys.getenv("GPBIOMETRICS_SMOKE_DIR")

if (nzchar(private_dir) && dir.exists(private_dir)) {
  smoke <- run_gazepoint_real_data_smoke(
    data_dir = private_dir
  )

  smoke$results
}
```

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_gazepoint_tfce *Guardrail for threshold-free cluster enhancement*

Description

This function is intentionally not implemented as a runnable inferential engine.

Usage

```
run_gazepoint_tfce(...)
```

Arguments

...	Ignored.

Value

This function always errors.

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.

scale_gazepoint_ppg_sections

Scale Gazepoint pulse/PPG signals within sections

Description

Scale Gazepoint pulse/PPG signals within sections

Usage

```
scale_gazepoint_ppg_sections(
  data,
  signal_col = NULL,
  section_cols = NULL,
  method = c("zscore", "minmax", "robust", "center", "none"),
  output_col = "ppg_scaled",
  range = c(0, 1)
)
```

Arguments

data	Data frame or numeric signal.
signal_col	Signal column when data is a data frame.
section_cols	Optional section/grouping columns.
method	Scaling method.
output_col	Name of the scaled output column.
range	Output range for minmax scaling.

Value

Data frame with an added scaled column, or a scaled vector for numeric input.

---

scale_gazepoint_ppg_signal

Scale a Gazepoint pulse/PPG signal

Description

Scale a Gazepoint pulse/PPG signal

Usage

```
scale_gazepoint_ppg_signal(
  x,
  method = c("zscore", "minmax", "robust", "center", "none"),
  range = c(0, 1)
)
```

Arguments

x	Numeric signal.
method	Scaling method: zscore, minmax, robust, center, or none.
range	Output range for minmax scaling.

Value

Scaled numeric vector.

scr_responsivity_sensitivity

Retention-first SCR responsivity sensitivity

Description

Preserves the conventional threshold/non-responder flag but estimates a Beta-Binomial posterior response probability and retains low-reactive participants for model-based sensitivity analyses.

Usage

```
scr_responsivity_sensitivity(
  participant,
  amplitude,
  threshold = 0.02,
  prior_alpha = 1,
  prior_beta = 1
)
```

Arguments

`participant`	Participant identifiers.
`amplitude`	Trial-level SCR amplitudes.
`threshold`	Conventional response threshold in microsiemens.
`prior_alpha, prior_beta`	Beta prior parameters.

Value

A data frame with conventional and posterior responsivity summaries.

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.

---

segment_gazepoint_pyhrv_nni

Segment NN intervals into time windows

Description

Segment NN intervals into time windows

Usage

```
segment_gazepoint_pyhrv_nni(
  nni_ms,
  segment_seconds = 300,
  overlap = 0,
  min_intervals = 3L
)
```

Arguments

nni_ms	Numeric NN intervals in milliseconds.
segment_seconds	Segment duration.
overlap	Segment overlap proportion.
min_intervals	Minimum intervals per segment.

Value

Data frame with segment membership.

---

session_info_gazepoint

Record gpbiometrics session information

Description

Record gpbiometrics session information

Usage

```
session_info_gazepoint(
  packages = NULL,
  include_loaded = TRUE,
  timestamp = Sys.time()
)
```

Arguments

packages	Optional additional packages to report.
include_loaded	Include loaded namespaces.
timestamp	Timestamp recorded in the output.

Value

A "gazepoint_session_info" object containing system metadata, package versions, and printable session text.

simulate_gazepoint_artifact *Inject simple synthetic artifacts into signal columns*

Description

Adds transparent, rule-based synthetic artifacts to selected numeric signal columns. The function is intended for testing QC pipelines and does not interpret the physiological meaning of any signal.

Usage

```
simulate_gazepoint_artifact(
  data,
  signal_cols,
  artifact = c("missing_run", "flatline", "spike"),
  n_artifacts = 1,
  artifact_length = 5,
  magnitude = NULL,
  seed = NULL,
  suffix = "_artifact",
  overwrite = FALSE
)
```

Arguments

data	A data frame.
signal_cols	Character vector of numeric signal columns.
artifact	Character vector of artifact types. Supported values are "missing_run", "flatline", "spike", "noise", and "drift".
n_artifacts	Number of artifacts of each requested type to add per signal column.
artifact_length	Number of rows affected by each inserted artifact.
magnitude	Optional numeric artifact magnitude. If NULL, a conservative value is derived from the signal standard deviation.
seed	Optional random seed for reproducible artifact placement.
suffix	Suffix used for new artifact-injected columns when overwrite = FALSE.
overwrite	Logical. If TRUE, modify the original signal columns. If FALSE, create new columns.

Value

A list with class gazepoint_artifact_simulation, containing the modified data, an artifact log, and the parameters used.

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.

`simulate_gazepoint_cluster_timecourse_data` *Simulate two-condition time-course data for cluster-permutation examples*

Description

Creates participant-level synthetic time-course data with an optional effect window. The function is intended for examples, tests, teaching, and documentation.

Usage

```
simulate_gazepoint_cluster_timecourse_data(
  n_subjects = 12L,
  n_time = 60L,
  conditions = c("A", "B"),
  effect_start = 25,
  effect_end = 38,
  effect_size = 0.6,
  noise_sd = 0.4,
  subject_sd = 0.25,
  time_start = 1,
  time_step = 1,
  effect_condition = "B",
  seed = NULL
)
```

Arguments

`n_subjects`	Number of participants.
`n_time`	Number of time bins.
`conditions`	Character vector of two condition labels.
`effect_start`	First time value included in the effect window.
`effect_end`	Last time value included in the effect window.
`effect_size`	Additive effect size inside the effect window.
`noise_sd`	Standard deviation of observation-level noise.
`subject_sd`	Standard deviation of participant-level random intercepts.
`time_start`	First time value.
`time_step`	Step between time bins.
`effect_condition`	Condition receiving the additive effect.
`seed`	Optional random seed.

Value

A data frame with columns `subject`, `condition`, `time`, `value`, and `true_effect`.

`simulate_gazepoint_eye_data` *Simulate Gazepoint-style gaze and pupil data*

Description

Generates synthetic Gazepoint-like eye-tracking data with time stamps, gaze coordinates, fixation identifiers, pupil diameter, validity columns, and random blink intervals. The output is intended for teaching, examples, testing, smoke tests, and vignette demonstrations; it is not a physiological ground-truth simulator.

Usage

```
simulate_gazepoint_eye_data(params = list())
```

Arguments

`params`	Named list of simulation parameters. Supported entries include `n`, `duration_s`, `sampling_rate_hz`, `seed`, `participant_id`, `trial_id`, `screen_bounds`, `n_fixations`, `fixation_mean_s`, `fixation_sd_s`, `gaze_noise_sd`, `saccade_samples`, `pupil_mean`, `pupil_sd`, `pupil_drift_sd`, `blink_rate_per_min`, `blink_duration_mean_s`, `blink_duration_sd_s`, `include_invalid_gaze`, and `invalid_gaze_prop`.

Value

A data frame with Gazepoint-style columns, including `time_s`, `MSTIMER`, `BPOGX`, `BPOGY`, `FPOGX`, `FPOGY`, `LPD`, `RPD`, `LPV`, `RPV`, `fixation_id`, `in_blink`, `blink_id`, `participant`, and `trial`.

Examples

```
dat <- simulate_gazepoint_eye_data(list(n = 120, seed = 1))
head(dat)
```

`simulate_gazepoint_multimodal_data` *Simulate a small multimodal Gazepoint-style dataset*

Description

Generates deterministic synthetic biometric, pupil/gaze, event, AOI, and fixation tables for tests, examples, and smoke-test workflows. The output is for software demonstration only and should not be treated as physiological validation data.

Usage

```
simulate_gazepoint_multimodal_data(
  n = NULL,
  duration_s = 20,
  sampling_rate_hz = 50,
  seed = 1,
  participant = "P01",
  n_trials = 4
)
```

Arguments

`n`	Number of samples.
`duration_s`	Recording duration in seconds.
`sampling_rate_hz`	Sampling rate in Hz. If supplied, overrides `n`.
`seed`	Optional random seed.
`participant`	Participant identifier.
`n_trials`	Number of trial/event periods.

Value

Named list with `biometrics`, `eye`, `events`, `fixations`, and `metadata`.

`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.

---

smooth_gazepoint_ppg_signal

Smooth a Gazepoint pulse/PPG signal

Description

Smooth a Gazepoint pulse/PPG signal

Usage

```
smooth_gazepoint_ppg_signal(
  x,
  sampling_rate_hz,
  method = c("mean", "median"),
  window_seconds = 0.1
)
```

Arguments

x	Numeric signal.
sampling_rate_hz	Sampling rate in Hz.
method	Smoothing method: mean or median.
window_seconds	Smoothing window length.

Value

Smoothed numeric signal.

`smooth_gazepoint_pupil` *Smooth Gazepoint pupil columns*

Description

Applies a simple centred moving-average smoother to Gazepoint pupil columns. The function is intended for transparent preprocessing and quality-control workflows. It does not interpolate long missing gaps and does not interpret pupil values as psychological or physiological states.

Usage

```
smooth_gazepoint_pupil(
  data,
  pupil_cols = NULL,
  id_cols = NULL,
  window = 5L,
  suffix = "_smooth",
  min_nonmissing = 1L
)
```

Arguments

`data`	A data frame containing Gazepoint pupil data.
`pupil_cols`	Character vector naming pupil columns. If `NULL`, numeric columns whose names contain `"pupil"` are used.
`id_cols`	Optional character vector naming grouping columns. Smoothing is applied within groups.
`window`	Positive odd integer giving the moving-average window size.
`suffix`	Suffix appended to smoothed pupil-column names.
`min_nonmissing`	Minimum number of non-missing values required inside a window to compute a smoothed value.

Value

A list with the processed data, a summary table, and settings. The object has class `"gazepoint_pupil_smoothing"`.

Examples

```
d <- data.frame(
  participant = rep("P01", 6),
  pupil_left = c(3.0, 3.2, 3.4, NA, 3.3, 3.1)
)
smooth_gazepoint_pupil(d, pupil_cols = "pupil_left", window = 3)
```

split_gazepoint_sessions_pspm_style

Split Gazepoint recordings into PsPM-style sessions

Description

Split Gazepoint recordings into PsPM-style sessions

Usage

```
split_gazepoint_sessions_pspm_style(
  data,
  time_col = NULL,
  gap_seconds = NULL,
  session_col = "pspm_session",
  reset_time = TRUE
)
```

Arguments

data	Gazepoint data frame.
time_col	Time column.
gap_seconds	Gap threshold. If NULL, inferred from sampling interval.
session_col	Output session column.
reset_time	If TRUE, add session-relative time.

Value

List with annotated data, sessions, and split data.

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_column_names

Standardize common Gazepoint column names

Description

Maps common Gazepoint-style export aliases to canonical names such as time_s, participant, trial, pupil_left, pupil_right, gaze_x, gaze_y, GSR, and PPG.

Usage

```
standardize_gazepoint_column_names(
  data,
  dictionary = NULL,
  conflict = c("suffix", "error", "keep"),
  ignore_case = TRUE
)
```

Arguments

data	Data frame, or a named list of data frames.
dictionary	Optional named list mapping canonical names to aliases.
conflict	How to handle rename conflicts: "suffix", "error", or "keep".
ignore_case	If TRUE, match aliases case-insensitively.

Value

Data frame, or list of data frames, with a gazepoint_column_standardization attribute containing the rename audit.

standardize_gazepoint_columns

Standardize Gazepoint column names

Description

Compatibility alias for `standardize_gazepoint_column_names()`. This exact name is provided for users who search for a shorter column-standardization helper.

Usage

```
standardize_gazepoint_columns(data, ...)
```

Arguments

`data`	Data frame or supported object passed to `standardize_gazepoint_column_names()`.
`...`	Additional arguments passed to `standardize_gazepoint_column_names()`.

Value

Standardized data object returned by `standardize_gazepoint_column_names()`.

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_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_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_decision_log`
Summarise a Gazepoint analysis decision log

Description

Summarise the number of recorded decisions by workflow stage, object type, decision label, and function name.

Usage

```
summarise_gazepoint_decision_log(log)
```

Arguments

`log`	A decision log created with `create_gazepoint_analysis_decision_log()`.

Value

A list with class `"gazepoint_analysis_decision_log_summary"`.

`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_fixations_by_aoi
Summarise Gazepoint fixations by area of interest

Description

Produces participant-, trial-, and AOI-level fixation counts, duration summaries, dwell proportions, and optional first-fixation latency.

Usage

```
summarise_gazepoint_fixations_by_aoi(
  fixations,
  aoi_col = NULL,
  participant_col = NULL,
  trial_col = NULL,
  group_cols = NULL,
  start_col = NULL,
  end_col = NULL,
  duration_col = NULL,
  event_onset_col = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
  duration_unit = c("auto", "seconds", "milliseconds", "samples"),
  sampling_rate_hz = NULL,
  include_unassigned = FALSE,
  unassigned_label = "UNASSIGNED"
)
```

Arguments

fixations	Fixation-level data frame.
aoi_col	AOI label column.
participant_col	Optional participant column.
trial_col	Optional trial column.
group_cols	Additional grouping columns.
start_col	Fixation-start column.
end_col	Optional fixation-end column.
duration_col	Optional fixation-duration column.
event_onset_col	Optional event/stimulus onset column.
time_unit	Unit for start, end, and event-onset values.
duration_unit	Unit for duration values.

`sampling_rate_hz`
: Sampling frequency required when time or duration values are represented as sample indices.

`include_unassigned`
: Include rows with missing or empty AOI labels.

`unassigned_label`
: Label used for retained unassigned rows.

Value

A data frame of class `"gazepoint_fixation_aoi_summary"`.

`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_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_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_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`.

summarize_gazepoint_aoi_dwell

Summarize AOI dwell time and entries

Description

Computes AOI dwell time, entry counts, first-entry latency, and valid-sample ratios from gaze samples or fixation/AOI tables.

Usage

```
summarize_gazepoint_aoi_dwell(
  data,
  time_col = NULL,
  aoi_col = NULL,
  duration_col = NULL,
  group_cols = NULL,
  valid_col = NULL
)
```

Arguments

data	Data frame containing AOI labels.
time_col	Optional time column for sample-level data.
aoi_col	AOI column.
duration_col	Optional duration column for fixation-level data.
group_cols	Optional grouping columns such as participant/trial.
valid_col	Optional validity column.

Value

Data frame with one row per group and AOI.

summarize_gazepoint_audit_trail

Summarize a Gazepoint audit trail

Description

Summarizes pass, warning, failure, not-checked, recorded, and other status counts from a Gazepoint audit index. The summary is descriptive only and is intended to support transparent reporting of audit outputs.

Usage

```
summarize_gazepoint_audit_trail(audit_index, by = NULL)
```

Arguments

audit_index	A gazepoint_audit_index object, a data frame, or an object accepted by create_gazepoint_audit_index().
by	Optional character vector of grouping columns in audit_index, such as "audit_id", "source_table", or "domain".

Value

A data frame with class gazepoint_audit_trail_summary.

---

summarize_gazepoint_beat_corrections
Summarize beat-correction logs

Description

Summarizes the correction log returned by correct_gazepoint_beats(). The summary is intended for transparent reporting and does not imply automatic exclusion or interpretive conclusions.

Usage

```
summarize_gazepoint_beat_corrections(correction, by = NULL)
```

Arguments

correction	A gazepoint_beat_correction object or a correction log data frame.
by	Optional character vector of grouping columns.

Value

A data frame.

summarize_gazepoint_eventlocked_multimodal
Summarize event-locked multimodal Gazepoint data

Description

Creates event-locked sample windows and trial-level summary metrics for one or more numeric Gazepoint modalities such as EDA, PPG, HR, IBI, pupil, gaze, or derived signals.

Usage

```
summarize_gazepoint_eventlocked_multimodal(
  data,
  events,
  time_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  group_cols = NULL,
  signal_cols = NULL,
  pre_s = 1,
  post_s = 3,
  baseline_window_s = c(-1, 0),
  summary_window_s = c(0, 3)
)
```

Arguments

data	Data frame or named list of data frames.
events	Event table or numeric event times.
time_col	Optional time column for data-frame input.
event_time_col	Optional event-time column.
event_id_col	Optional event identifier column.
group_cols	Optional grouping columns used to match events to samples.
signal_cols	Optional character vector, or named list for list input.
pre_s	Seconds before each event.
post_s	Seconds after each event.
baseline_window_s	Two-element baseline window relative to event.
summary_window_s	Two-element summary window relative to event.

Value

Object of class gazepoint_eventlocked_multimodal.

summarize_gazepoint_export_inventory

Summarize Gazepoint export inventory

Description

Creates a descriptive file inventory for a Gazepoint export folder or a vector of files. The function summarizes paths, extensions, file sizes, likely export types, and empty-file flags. It is intended for reproducibility and audit reporting only.

Usage

```
summarize_gazepoint_export_inventory(
  path,
  recursive = TRUE,
  include_hidden = FALSE,
  classify = TRUE
)
```

Arguments

path	Directory path, file path, or character vector of file paths.
recursive	Logical. If TRUE, directories are searched recursively.
include_hidden	Logical. If TRUE, hidden files are included.
classify	Logical. If TRUE, likely Gazepoint export types are inferred from file names using conservative keyword rules.

Value

A data frame with class gazepoint_export_inventory.

summarize_gazepoint_feature_coverage

Summarize gpbiometrics feature coverage

Description

Creates a descriptive feature-coverage table from exported function names. Domains are assigned using transparent name patterns. The summary is intended for release review and documentation planning only.

Usage

```
summarize_gazepoint_feature_coverage(
  path = ".",
  exports = NULL,
  patterns = NULL
)
```

Arguments

path	Package root.
exports	Optional character vector of exported functions. If NULL, exports are read from NAMESPACE.
patterns	Optional named list of regular-expression patterns used to assign exported functions to domains.

Value

A data frame with one row per domain.

---

summarize_gazepoint_fixations

Summarize Gazepoint fixation metrics

Description

Computes fixation-level summary metrics by participant, trial, AOI, or any user-specified grouping columns.

Usage

```
summarize_gazepoint_fixations(
  fixDF,
  duration_col = NULL,
  x_col = NULL,
  y_col = NULL,
  participant_col = NULL,
  trial_col = NULL,
  aoi_col = NULL,
  group_cols = NULL,
  duration_unit = c("auto", "seconds", "milliseconds")
)
```

Arguments

fixDF	Fixation data frame.
duration_col	Fixation duration column. If NULL, common Gazepoint names such as FPOGD are detected automatically.
x_col	Fixation x-coordinate column.
y_col	Fixation y-coordinate column.
participant_col	Optional participant column.
trial_col	Optional trial column.
aoi_col	Optional AOI column.
group_cols	Optional grouping columns. Overrides detected participant, trial, and AOI columns.
duration_unit	"auto", "seconds", or "milliseconds".

Value

Data frame with fixation count, duration, and dispersion metrics.

Examples

```
fix <- data.frame(trial = "T1", AOI = c("A", "A"), FPOGD = c(.2, .3),
                  FPOGX = c(.1, .2), FPOGY = c(.3, .4))
summarize_gazepoint_fixations(fix)
```

---

summarize_gazepoint_fixations_by_aoi

American-spelling alias for fixation summaries by AOI

Description

American-spelling alias for fixation summaries by AOI

Usage

```
summarize_gazepoint_fixations_by_aoi(...)
```

Arguments

...	Arguments passed to summarise_gazepoint_fixations_by_aoi().

Value

A data frame of class "gazepoint_fixation_aoi_summary".

summarize_gazepoint_missingness
Summarize missingness and gap structure in Gazepoint data

Description

Computes missingness rates, missing-run counts, longest missing run, longest missing duration, and optional long-gap counts for one or more columns. This is intended as a reviewer-friendly audit helper before interpolation, exclusion, or event-locked analysis.

Usage

```
summarize_gazepoint_missingness(
  data,
  signal_cols = NULL,
  time_col = NULL,
  group_cols = NULL,
  long_gap_s = NULL,
  count_nonfinite = TRUE
)
```

Arguments

data	Data frame containing signal columns.
signal_cols	Optional character vector of columns to audit. If omitted, all columns except time_col and group_cols are audited.
time_col	Optional time column. If supplied or detected, missing-run durations are reported in seconds.
group_cols	Optional grouping columns such as participant, session, or trial.
long_gap_s	Optional threshold in seconds used to count long missing gaps.
count_nonfinite	If TRUE, non-finite numeric values are counted as missing.

Value

Data frame with one row per group and signal column.

summarize_gazepoint_nonwear

Summarize signal-dropout detections

Description

Aggregates the summary table returned by `detect_gazepoint_nonwear()`. The result is intended for QC reporting and does not imply automatic exclusion.

Usage

```
summarize_gazepoint_nonwear(nonwear, by = "signal")
```

Arguments

`nonwear`	A `gazepoint_nonwear_detection` object or a compatible summary data frame.
`by`	Character vector of columns used for aggregation.

Value

A data frame.

summarize_gazepoint_preregistration_readiness

Summarize preregistration readiness

Description

Summarizes checklist or audit results into compact readiness counts and proportions. The summary is intended for reporting and reviewer-readiness documentation.

Usage

```
summarize_gazepoint_preregistration_readiness(audit, by = NULL)
```

Arguments

`audit`	A `gazepoint_preregistration_audit` object or an item-level audit data frame.
`by`	Optional grouping column, such as `"domain"`.

Value

A data frame with class `gazepoint_preregistration_readiness`.

summarize_gazepoint_pupil_events
Summarize event-related pupil responses

Description

Computes event-level baseline-corrected pupil metrics, including peak dilation, peak latency, mean response, and area under the curve.

Usage

```
summarize_gazepoint_pupil_events(
  data,
  events,
  pre = 1,
  post = 3,
  time_col = NULL,
  pupil_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  baseline_window = NULL,
  response_window = c(0, 3)
)
```

Arguments

data	Pupil data frame.
events	Event timestamps, event data frame, or event-log path.
pre	Seconds before event onset.
post	Seconds after event onset.
time_col	Time column.
pupil_col	Pupil column.
event_time_col	Event-time column in events.
event_id_col	Event identifier column in events.
baseline_window	Baseline window relative to event onset.
response_window	Response window relative to event onset.

Value

Data frame with one row per event.

summarize_gazepoint_qc_overview

Summarize QC overview tables

Description

Creates a compact QC overview table from row-level or session-level QC data. The summary can include counts of flags, mean/minimum quality index values, and simple descriptive summaries for selected metrics.

Usage

```
summarize_gazepoint_qc_overview(
  data,
  group_cols = NULL,
  quality_index_col = NULL,
  flag_cols = NULL,
  metric_cols = NULL
)
```

Arguments

`data`	A data frame.
`group_cols`	Optional grouping columns.
`quality_index_col`	Optional numeric quality-index column.
`flag_cols`	Optional logical flag columns. If `NULL`, logical columns containing `"flag"` in their names are used when available.
`metric_cols`	Optional numeric metric columns to summarize.

Value

A data frame with class `gazepoint_qc_overview`.

summarize_gazepoint_scanpath_metrics

Summarize simple scanpath metrics

Description

Computes simple gaze/fixation-path summaries including path length, saccade count, regression-like leftward movements, fixation count, AOI transitions, and transition entropy.

Usage

```
summarize_gazepoint_scanpath_metrics(
  data,
  x_col = NULL,
  y_col = NULL,
  time_col = NULL,
  aoi_col = NULL,
  fixation_id_col = NULL,
  group_cols = NULL,
  min_saccade_distance = 0.02
)
```

Arguments

`data`	Gaze or fixation data frame.
`x_col`	Gaze/fixation x column.
`y_col`	Gaze/fixation y column.
`time_col`	Optional time column.
`aoi_col`	Optional AOI column.
`fixation_id_col`	Optional fixation identifier column.
`group_cols`	Optional grouping columns.
`min_saccade_distance`	Minimum Euclidean movement counted as saccade.

Value

Data frame with one row per group.

`summarize_gazepoint_scr_recovery` *Summarize SCR recovery around events*

Description

Computes event-level baseline, peak amplitude, peak latency, recovery latency, and post-peak recovery slope from EDA/GSR data.

Usage

```
summarize_gazepoint_scr_recovery(
  data,
  events,
  pre = 1,
  post = 6,
```

```
  time_col = NULL,
  signal_col = NULL,
  event_time_col = NULL,
  event_id_col = NULL,
  baseline_window = NULL,
  peak_window = c(0.5, 4),
  recovery_fraction = 0.5
)
```

Arguments

data	EDA/GSR data frame.
events	Event timestamps, event data frame, or event-log path.
pre	Seconds before event onset.
post	Seconds after event onset.
time_col	Time column in data.
signal_col	EDA/GSR signal column.
event_time_col	Event-time column in events.
event_id_col	Event identifier column in events.
baseline_window	Baseline window relative to event onset.
peak_window	Peak-search window relative to event onset.
recovery_fraction	Fraction of peak amplitude used as recovery target.

Value

Data frame with one row per event.

---

summarize_gazepoint_signal_quality

Summarize signal-quality indicators

Description

Summarizes signal-quality rows by user-selected columns. The result is a reporting table and does not imply automatic exclusion or interpretation.

Usage

```
summarize_gazepoint_signal_quality(quality, by = "signal")
```

Arguments

quality	A data frame returned by compute_gazepoint_signal_quality() or classify_gazepoint_signal_q
by	Character vector of grouping columns. Defaults to "signal".

Value

A data frame.

`summarize_gazepoint_time_clusters`
Summarize Gazepoint time clusters

Description

Returns a compact cluster-level summary table from a `gazepoint_cluster_permutation` object.

Usage

```
summarize_gazepoint_time_clusters(x, alpha = NULL)
```

Arguments

`x`	Object returned by `run_gazepoint_cluster_permutation()`.
`alpha`	Optional cluster-level alpha. If `NULL`, the alpha stored in the object is used.

Value

A data frame.

`summarize_gazepoint_tracking`
Summarize Gazepoint tracking validity

Description

Computes valid-pupil, valid-gaze, and combined tracking ratios overall or by participant/trial/group.

Usage

```
summarize_gazepoint_tracking(
  data,
  pupil_cols = NULL,
  x_col = NULL,
  y_col = NULL,
  group_cols = NULL,
  screen_bounds = c(0, 1, 0, 1),
  nonpositive_is_invalid = TRUE
)
```

Arguments

data	Eye-tracking data frame.
pupil_cols	Optional pupil columns.
x_col	Optional gaze x column.
y_col	Optional gaze y column.
group_cols	Optional grouping columns.
screen_bounds	Numeric vector c(x_min, x_max, y_min, y_max).
nonpositive_is_invalid	If TRUE, non-positive pupil values are invalid.

Value

Data frame of tracking ratios by group.

---

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.

`sync_gazepoint_signals_via_lsl` *Synchronize imported Gazepoint LSL/XDF streams*

Description

Applies explicit clock offsets and known latency corrections to already imported LSL/XDF streams. Input may be named data frames or pyxdf-style stream lists containing `time_stamps` and `time_series`.

Usage

```
sync_gazepoint_signals_via_lsl(
  streams,
  reference = NULL,
  time_cols = NULL,
  clock_offsets_s = NULL,
  known_lags_s = NULL,
  relative_zero = c("reference", "global", "none"),
  dejitter = c("none", "linear"),
  nominal_rates_hz = NULL,
  merge = c("none", "nearest"),
  tolerance_s = NULL
)
```

Arguments

`streams`	Named list of imported streams.
`reference`	Name of the reference stream.
`time_cols`	Optional named time-column vector or list.
`clock_offsets_s`	Values added to stream timestamps to place them in the reference clock domain.
`known_lags_s`	Known acquisition latencies subtracted from timestamps.
`relative_zero`	Origin for relative synchronized time.
`dejitter`	Optional transparent linear timestamp regularization.
`nominal_rates_hz`	Optional named nominal sampling frequencies.
`merge`	Whether to return separate streams or a nearest-neighbour merged table on the reference stream.
`tolerance_s`	Optional nearest-neighbour tolerance.

Value

A "gazepoint_lsl_sync" object.

See Also

import_gazepoint_lsl_xdf()

---

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`.

`trim_gazepoint_biometrics_pspm_style`

Trim Gazepoint biometrics data PsPM-style

Description

Trim Gazepoint biometrics data PsPM-style

Usage

```
trim_gazepoint_biometrics_pspm_style(
  data,
  start_s = NULL,
  end_s = NULL,
  time_col = NULL,
  reset_time = FALSE
)
```

Arguments

`data`	Gazepoint data frame.
`start_s`	Start time in seconds.
`end_s`	End time in seconds.
`time_col`	Time column.
`reset_time`	If TRUE, reset trimmed time to start at zero.

Value

Trimmed data frame.

`upsample_gazepoint_data`

Regularize signal data to an evenly spaced time grid

Description

Uses `stats::approx()` to interpolate selected numeric signals onto a regular time grid within each group. The function records an interpolation log and does not extrapolate beyond the observed time range.

Usage

```
upsample_gazepoint_data(
  data,
  time_col,
  signal_cols = NULL,
  group_cols = NULL,
  interval = NULL,
  method = c("linear", "constant")
)
```

Arguments

data	A data frame.
time_col	Numeric time column.
signal_cols	Optional character vector of numeric signal columns. If NULL, all numeric columns except grouping and time columns are used.
group_cols	Optional grouping columns.
interval	Numeric interval for the output time grid, in the same units as time_col. If NULL, the median positive time difference is used within each group.
method	Interpolation method passed to stats::approx(): "linear" or "constant".

Value

A data frame with class gazepoint_upsampled_data.

---

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"`.

`validate_gazepoint_format` *Validate a Gazepoint-format data frame*

Description

Lightweight format validator for Gazepoint-style biometric exports. This wrapper checks required and optional columns and, when available, attaches schema and audit outputs from existing package helpers.

Usage

```
validate_gazepoint_format(
  data,
  required_cols = NULL,
  optional_cols = NULL,
  expected_modalities = NULL,
  standardize = FALSE,
  strict = FALSE,
  ...
)
```

Arguments

`data`	Data frame to validate.
`required_cols`	Required column names.
`optional_cols`	Optional column names to report as present or absent.
`expected_modalities`	Optional expected modalities passed to `audit_gazepoint_biometrics_file()` when available.
`standardize`	If TRUE, standardize column names before validation.
`strict`	If TRUE, audit warnings make the returned `valid` field FALSE.
`...`	Reserved for future extensions.

Value

Object of class `gazepoint_format_validation`.

validate_gazepoint_gaze *Validate Gazepoint gaze data*

Description

Consolidates coordinate, validity, missingness, timestamp, sampling, and large-gap checks for sample-level Gazepoint gaze data.

Usage

```
validate_gazepoint_gaze(
  data,
  time_col = NULL,
  x_col = NULL,
  y_col = NULL,
  validity_cols = NULL,
  group_cols = NULL,
  coordinate_system = c("auto", "normalized", "pixels", "degrees"),
  screen_width_px = NULL,
  screen_height_px = NULL,
  time_unit = c("auto", "seconds", "milliseconds", "samples"),
  sampling_rate_hz = NULL,
  expected_sampling_rate_hz = NULL,
  sampling_tolerance = 0.2,
  missing_threshold = 0.2,
  gap_multiplier = 3
)
```

Arguments

data	A sample-level data frame.
time_col	Time column. If NULL, a likely column is inferred.
x_col	Horizontal gaze-coordinate column.
y_col	Vertical gaze-coordinate column.
validity_cols	Optional validity columns. Logical values and positive numeric values are treated as valid.
group_cols	Optional grouping columns, such as participant and trial.
coordinate_system	Coordinate system: "auto", "normalized", "pixels", or "degrees".
screen_width_px, screen_height_px	Screen dimensions required for bounded pixel-coordinate checks.
time_unit	Time unit: "auto", "seconds", "milliseconds", or "samples".
sampling_rate_hz	Sampling rate required for sample-index time.

`expected_sampling_rate_hz`
: Optional expected sampling rate.

`sampling_tolerance`
: Relative tolerance around the expected sampling interval.

`missing_threshold`
: Maximum acceptable missing-gaze proportion.

`gap_multiplier`
: A gap is flagged when it exceeds this multiple of the expected or median interval.

Value

A `"gazepoint_gaze_validation"` object containing row-level flags, group summaries, checks, an overall summary, and resolved settings.

`validate_gazepoint_metadata`
Validate Gazepoint metadata and required columns

Description

Performs transparent metadata checks for Gazepoint workflow data frames. The function checks required columns, optional expected columns, missing IDs, duplicate key rows, and time ordering within groups. It returns a structured report rather than modifying the data.

Usage

```
validate_gazepoint_metadata(
  data,
  required_cols = character(),
  expected_cols = character(),
  id_cols = NULL,
  time_col = NULL,
  unique_cols = NULL,
  allow_missing_ids = FALSE
)
```

Arguments

`data`
: A data frame.

`required_cols`
: Character vector of required columns.

`expected_cols`
: Optional character vector of expected but non-fatal columns.

`id_cols`
: Optional character vector naming participant/session/trial identifiers.

`time_col`
: Optional time column used for ordering checks.

`unique_cols`
: Optional character vector defining a row-level key that should be unique.

`allow_missing_ids`
: Logical. If `FALSE`, missing values in `id_cols` are reported as problems.

Value

A list containing status, problems, warnings, and a summary table. The object has class "gazepoint_metadata_validation

Examples

```
d <- data.frame(
  participant = c("P01", "P01"),
  time = c(1, 2),
  pupil_left = c(3.1, 3.2)
)
validate_gazepoint_metadata(
  d,
  required_cols = c("participant", "time"),
  id_cols = "participant",
  time_col = "time"
)
```

---

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.

`write_gazepoint_decision_log`
Write a Gazepoint analysis decision log to disk

Description

Write a decision log to CSV, optionally with a compact text summary.

Usage

```
write_gazepoint_decision_log(log, path, summary_path = NULL, overwrite = FALSE)
```

Arguments

`log`	A decision log created with `create_gazepoint_analysis_decision_log()`.
`path`	Output CSV file path.
`summary_path`	Optional output path for a text summary. If `NULL`, only the CSV file is written.
`overwrite`	Logical. If `FALSE`, existing files are not overwritten.

Value

A data frame listing written files.

`write_gazepoint_export_profile`
Write a Gazepoint export-folder profile to disk

Description

Export profile tables and a compact text summary.

Usage

```
write_gazepoint_export_profile(
  profile,
  path,
  prefix = "gazepoint_export_profile",
  overwrite = FALSE
)
```

Arguments

profile	Object returned by profile_gazepoint_export_folder().
path	Output folder.
prefix	File prefix.
overwrite	Logical. If FALSE, existing files are not overwritten.

Value

A data frame listing written files.

---

write_gazepoint_interoperability_audit *Write an interoperability audit*

Description

Writes only machine-readable aggregate compatibility information. No source data, participant identifiers, input filenames, or local data paths are included.

Usage

```
write_gazepoint_interoperability_audit(
  x,
  output_dir,
  prefix = "gpbiometrics-interoperability",
  overwrite = FALSE
)
```

Arguments

x	Object returned by audit_gazepoint_interoperability_versions().
output_dir	Output directory.
prefix	Filename prefix.
overwrite	Logical. Permit replacement of existing files.

Value

Invisibly returns a named character vector containing the four written file paths.

Examples

```
audit <- audit_gazepoint_interoperability_versions(
  include_python = FALSE
)
output <- tempfile("gpbiometrics-interoperability-")
files <- write_gazepoint_interoperability_audit(
  audit,
  output
)
basename(files)
```

`write_gazepoint_mne_fif` *Write prepared Gazepoint data to an MNE FIF file*

Description

Uses an external Python installation containing `numpy` and `mne`. Python is invoked only when `execute = TRUE`; no Python package is an R dependency.

Usage

```
write_gazepoint_mne_fif(
  x,
  fname,
  events = NULL,
  overwrite = FALSE,
  fmt = c("single", "double"),
  python = NULL,
  execute = TRUE,
  keep_intermediate = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`x`	A `"gazepoint_mne_input"` object or a data frame accepted by `prepare_gazepoint_mne_input()`.
`fname`	Output filename. Use an MNE-compatible suffix such as `"_raw.fif"` or `"_raw.fif.gz"`.
`events`	Optional `"gazepoint_mne_events"` object or three-column event matrix. Prepared event objects are also attached as MNE annotations.
`overwrite`	Overwrite an existing file.
`fmt`	FIF numeric format: `"single"` or `"double"`.
`python`	Optional Python executable or Windows py launcher.
`execute`	If `FALSE`, return a dry-run specification.

keep_intermediate
: Retain temporary CSV, TSV, and Python files.

verbose
: Show Python/MNE output.

...
: Arguments passed to prepare_gazepoint_mne_input() when x is a data frame.

Value

A "gazepoint_mne_fif_export" object.

---

write_gazepoint_real_data_smoke

Write privacy-safe real-data smoke-test summaries

Description

Writes only aggregate dataset summaries, sanitized condition information, runtime metadata, and non-sensitive settings.

Usage

```
write_gazepoint_real_data_smoke(
  x,
  output_dir,
  prefix = "gpbiometrics-real-data-smoke",
  overwrite = FALSE,
  protect_repository = TRUE
)
```

Arguments

x
: Object returned by run_gazepoint_real_data_smoke().

output_dir
: External output directory.

prefix
: Output filename prefix.

overwrite
: Permit replacement of existing files?

protect_repository
: Reject output directories inside the current package repository?

Value

Invisibly returns a named character vector of written files.

Index

add_gazepoint_decision, 10
align_gazepoint_biometrics_to_ttl, 11
align_gazepoint_streams_by_events, 13
align_gazepoint_to_eeg, 14
analyze_gazepoint_ac_susceptance, 16
analyze_gazepoint_cardiorespiratory_causality, 17
analyze_gazepoint_skin_potential, 18
anonymize_gazepoint_data, 19
assert_gazepoint_columns, 19
assess_gazepoint_hrp_waveform_quality, 20
assess_gazepoint_sampling_irregularity, 21
assign_gazepoint_aoi, 22
assign_gazepoint_aoi(), *252*, *255*
audit_gazepoint_beats, 25
audit_gazepoint_biometric_missingness, 26
audit_gazepoint_biometric_sampling, 26
audit_gazepoint_biometric_sync_drift, 27
audit_gazepoint_biometrics_file, 28
audit_gazepoint_condition_balance, 29
audit_gazepoint_dataset_structure, 30
audit_gazepoint_distributional_drift, 31
audit_gazepoint_eda_artifacts, 32
audit_gazepoint_engagement_dial, 33
audit_gazepoint_event_coverage, 34
audit_gazepoint_experiment_design, 35
audit_gazepoint_export_schema, 36
audit_gazepoint_gsr_quality, 37
audit_gazepoint_gsr_units, 37
audit_gazepoint_hr_quality, 38
audit_gazepoint_ibi_quality, 39
audit_gazepoint_interoperability_versions, 40
audit_gazepoint_interoperability_versions(), *201*, *367*
audit_gazepoint_pipeline_steps, 41
audit_gazepoint_preregistration_consistency, 42
audit_gazepoint_pupil_luminance, 43
audit_gazepoint_release_readiness, 43
audit_gazepoint_session_comparability, 44
audit_gazepoint_signal_activity, 45
audit_gazepoint_signal_activity(), *223*, *238*
audit_gazepoint_smoke_privacy, 46
audit_gazepoint_stabilization_period, 46
audit_gazepoint_time_resets, 47
audit_gazepoint_time_resets(), *28*, *223*, *240*
audit_gazepoint_timecourse_grid, 48

barplot(), *222*
baseline_correct_gazepoint_gsr, 49
baseline_correct_gazepoint_hr, 50
baseline_correct_gazepoint_pupil, 51
baseline_correct_gazepoint_pupil(), *266*
build_gazepoint_aoi_timecourse, 52

calculate_gazepoint_rsa, 53
check_gazepoint_bids, 54
check_gazepoint_bids(), *165*
check_gazepoint_biometric_columns, 55
check_gazepoint_plot_contract, 55
check_gazepoint_ppg_binary_quality, 56
check_gazepoint_pyhrv_interval, 57
check_gazepoint_pyhrv_interval(), *269*
chunk_gazepoint_biometrics, 57
classify_gazepoint_eda_response_pattern, 58
classify_gazepoint_scr_intervals, 59
classify_gazepoint_signal_quality, 60

clean_gazepoint_pupil, 60
clean_gazepoint_pupil_signal, 61
clean_gazepoint_pupil_signal(), *255*, *266*
clean_gazepoint_rr_intervals, 63
combine_gazepoint_marker_channels_pspm_style, 63
compare_gazepoint_conditions_bootstrap, 64
compare_gazepoint_export_profiles, 65
compare_gazepoint_hr_ibi_consistency, 66
compare_gazepoint_pyhrv_psd_methods, 67
compare_hrv_prv_devices, 67
compute_gazepoint_engagement_index, 68
compute_gazepoint_hrv_wavelet_psd, 69
compute_gazepoint_ppg_frequency_measures, 70
compute_gazepoint_ppg_measures, 71
compute_gazepoint_ppg_template_similarity, 71
compute_gazepoint_pyhrv_ar_psd, 72
compute_gazepoint_pyhrv_dfa, 73
compute_gazepoint_pyhrv_frequency_domain, 73
compute_gazepoint_pyhrv_heart_rate, 74
compute_gazepoint_pyhrv_hr_parameters, 74
compute_gazepoint_pyhrv_lomb_psd, 75
compute_gazepoint_pyhrv_nn20, 76
compute_gazepoint_pyhrv_nn50, 76
compute_gazepoint_pyhrv_nn_diff, 75
compute_gazepoint_pyhrv_nni_differences_parameters, 77
compute_gazepoint_pyhrv_nni_parameters, 77
compute_gazepoint_pyhrv_nnxx, 78
compute_gazepoint_pyhrv_nonlinear, 78
compute_gazepoint_pyhrv_poincare, 79
compute_gazepoint_pyhrv_psd_waterfall, 79
compute_gazepoint_pyhrv_rmssd, 80
compute_gazepoint_pyhrv_sample_entropy, 80
compute_gazepoint_pyhrv_sdann, 81
compute_gazepoint_pyhrv_sdnn, 81
compute_gazepoint_pyhrv_sdnn_index, 82
compute_gazepoint_pyhrv_sdsd, 82
compute_gazepoint_pyhrv_time_domain, 83
compute_gazepoint_pyhrv_tinn, 83
compute_gazepoint_pyhrv_triangular_index, 84
compute_gazepoint_pyhrv_welch_psd, 84
compute_gazepoint_quality_index, 85
compute_gazepoint_scr_habituation, 86
compute_gazepoint_scr_latency, 86
compute_gazepoint_signal_band_power, 87
compute_gazepoint_signal_correlation, 88
compute_gazepoint_signal_lag_matrix, 89
compute_gazepoint_signal_phase_locking, 90
compute_gazepoint_signal_power_spectrum, 90
compute_gazepoint_signal_quality, 91
convert_gazepoint_gsr_to_conductance, 92
correct_gazepoint_beats, 93
correct_gazepoint_eda_temperature, 94
correct_gazepoint_ppg_hampel, 95
correct_gazepoint_rri_artifacts_local, 95
create_gazepoint_analysis_decision_log, 96
create_gazepoint_analysis_manifest, 97
create_gazepoint_audit_index, 98
create_gazepoint_audit_report_section, 99
create_gazepoint_biometrics_checklist, 99
create_gazepoint_biometrics_feature_inventory, 100
create_gazepoint_biometrics_methods_text, 101
create_gazepoint_biometrics_report, 101
create_gazepoint_biometrics_report_tables, 103
create_gazepoint_dictionary, 104
create_gazepoint_eda_analysis_pipeline, 105
create_gazepoint_eye_methods_text, 106

create_gazepoint_heartpy_report, 107
create_gazepoint_methods_section, 108
create_gazepoint_pipeline_map, 109
create_gazepoint_preregistration_checklist, 110
create_gazepoint_preregistration_template, 110
create_gazepoint_pspm_glm_design, 111
create_gazepoint_pyhrv_time_vector, 112
create_gazepoint_qc_supplement, 113
create_gazepoint_quality_dashboard, 113
create_gazepoint_release_checklist, 114
create_gazepoint_reproducibility_statement, 115
create_gazepoint_sidecar_template, 116
create_gazepoint_trial_regressors, 116

decompose_gazepoint_eda, 118
denoise_gazepoint_eda_autoencoder, 119
denoise_gazepoint_eda_wavelet, 120
denoise_gazepoint_ppg_autoencoder, 121
denoise_gazepoint_quantization_noise, 122
detect_active_biometric_channels, 123
detect_gazepoint_biometric_schema, 123
detect_gazepoint_biometric_timebase, 124
detect_gazepoint_blinks, 125
detect_gazepoint_doubly_stochastic_changepoints, 126
detect_gazepoint_fixations, 127
detect_gazepoint_fixations(), *134*
detect_gazepoint_nonwear, 129
detect_gazepoint_ppg_onsets, 130
detect_gazepoint_ppg_peaks, 131
detect_gazepoint_pupil_blinks, 132
detect_gazepoint_pupil_blinks(), *255*, *266*
detect_gazepoint_saccades, 134
detect_gazepoint_saccades(), *129*
detect_gazepoint_scr_events, 135
detect_gazepoint_scr_peaks, 136
detect_gazepoint_time_columns, 137
detrend_gazepoint_rri_window, 138
detrend_gazepoint_signal, 138
diagnose_gazepoint_biometrics_workflow, 139
diagnose_gazepoint_biometrics_workflow(), *302*
diagnose_gazepoint_cluster_design, 140
diagnose_gazepoint_sync_drift, 141
downsample_gazepoint_data, 142
downsample_gazepoint_data(), *255*

enhance_gazepoint_ppg_peaks, 143
epoch_gazepoint_scr, 144
estimate_gazepoint_breathing_rate_from_ibi, 145
estimate_gazepoint_cluster_offset, 146
estimate_gazepoint_cluster_onset, 146
estimate_gazepoint_eda_recovery_times, 147
estimate_gazepoint_lsl_clock_offsets, 148
estimate_gazepoint_respiration_from_ppg, 149
estimate_gazepoint_samplerate_datetime, 149
estimate_gazepoint_samplerate_mstimer, 150
estimate_gazepoint_signal_lag, 151
export_gazepoint_audit_trail_markdown, 152
export_gazepoint_biometrics_report_bundle, 153
export_gazepoint_cluster_results, 154
export_gazepoint_heartpy_input, 154
export_gazepoint_mne_cluster_input, 155
export_gazepoint_permuco_cluster_input, 156
export_gazepoint_permutes_cluster_input, 157
export_gazepoint_pipeline_dot, 158
export_gazepoint_pspm_model_estimates, 159
export_gazepoint_pyhrv_results, 159
export_gazepoint_rhrv_input, 160
export_gazepoint_rhrv_input(), *270*, *271*
export_gazepoint_to_bids, 161
export_gazepoint_to_bids(), *243*, *244*
extract_gazepoint_beats_kmeans, 165
extract_gazepoint_bilateral_eda_asymmetry, 166

extract_gazepoint_eda_complexity, 167
extract_gazepoint_eda_events_biosppy_style, 168
extract_gazepoint_eda_spectral_power, 169
extract_gazepoint_eda_tvsymp, 170
extract_gazepoint_edr_pca, 171
extract_gazepoint_hrv_asymmetry, 172
extract_gazepoint_hrv_features, 172
extract_gazepoint_hrv_fragmentation, 173
extract_gazepoint_hrv_fuzzy_csi, 174
extract_gazepoint_hrv_geometric, 175
extract_gazepoint_hrv_nonlinear, 176
extract_gazepoint_hrv_rcmse, 177
extract_gazepoint_hrv_rqa, 178
extract_gazepoint_markerinfo_pspm_style, 179
extract_gazepoint_pdr_signals, 180
extract_gazepoint_ppg_morphology, 181
extract_gazepoint_ppg_templates, 182
extract_gazepoint_pyhrv_nn_intervals, 183
extract_gazepoint_respiration_ceemdan, 183
extract_gazepoint_scr_recovery_times, 184
extract_gazepoint_segments_pspm_style, 185
extract_gazepoint_ttl_events, 186

filter_gazepoint_gaze, 187
filter_gazepoint_ibi_implausible, 188
filter_gazepoint_ppg_butterworth, 189
filter_gazepoint_ppg_signal, 190
filter_gazepoint_signal, 191
fit_gazepoint_convolution_glm, 192
flag_gazepoint_artifacts_svm, 192
flag_gazepoint_biometric_dropouts, 193
flag_gazepoint_biometric_dropouts(), *222*
flag_gazepoint_hrv_segments, 194
flag_gazepoint_mad_artifacts, 195
flag_gazepoint_ppg_quality, 196
flag_gazepoint_rr_outliers, 197
flag_kleckner_eda_artifacts, 198
flip_gazepoint_ppg_signal, 199
format_gazepoint_biometrics_feature_inventory, 200
fuse_gazepoint_respiration_kalman, 200

gazepoint_interoperability_manifest, 201
gazepoint_interoperability_manifest(), *40*
generate_gazepoint_manifest, 202
get_gazepoint_plot_data, 203
get_gazepoint_plot_settings, 203
gpbiometrics_info, 204

import_gazepoint_biometric_folder, 204
import_gazepoint_biometrics, 205
import_gazepoint_data, 206
import_gazepoint_data_summary, 207
import_gazepoint_event_log, 207
import_gazepoint_lsl_xdf, 208
import_gazepoint_lsl_xdf(), *359*
import_gazepoint_pyhrv_results, 209
impute_gazepoint_missing, 209
interpolate_gazepoint_pupil_blinks, 210

join_gazepoint_biometrics_to_gp3tools, 211
join_gazepoint_biometrics_to_master, 212
join_gazepoint_biometrics_to_master(), *211*, *212*

match_gazepoint_events_to_biometrics, 213
matplot(), *225*
merge_gazepoint_recordings_pspm_style, 214
model_gazepoint_eda_point_process, 214
model_gazepoint_hr_point_process, 215
model_gazepoint_hrv_ipfm, 216

normalize_gazepoint_scr, 217

optimize_gazepoint_cvxeda_tau, 218

physiology_validation_ladder, 219
pipeline_comparison_dashboard, 219
plot_gazepoint_aoi_biometrics, 221
plot_gazepoint_biometric_quality, 221
plot_gazepoint_biometric_report_dashboard, 223
plot_gazepoint_biometric_signals, 224

plot_gazepoint_cluster_null_distribution, 225
plot_gazepoint_cluster_permutation, 226
plot_gazepoint_design_coverage, 226
plot_gazepoint_eda_decomposition, 227
plot_gazepoint_eda_gram, 228
plot_gazepoint_export_profile, 229
plot_gazepoint_missingness, 229
plot_gazepoint_multimodal_timeline, 230
plot_gazepoint_ppg_breathing, 232
plot_gazepoint_ppg_peak_detection, 232
plot_gazepoint_ppg_poincare, 233
plot_gazepoint_ppg_segmentwise, 233
plot_gazepoint_pyhrv_hr_heatplot, 234
plot_gazepoint_pyhrv_radar_chart, 234
plot_gazepoint_pyhrv_tachogram, 235
plot_gazepoint_saccade_main_sequence, 235
plot_gazepoint_saccade_main_sequence(), *129*
plot_gazepoint_scr_events, 236
plot_gazepoint_scr_specification_curve, 237
plot_gazepoint_signal_activity, 238
plot_gazepoint_signal_quality, 239
plot_gazepoint_time_resets, 240
ppg_topology_features, 240
prepare_gazepoint_aoi_biometrics_model_data, 241
prepare_gazepoint_artifact_svm_features, 242
prepare_gazepoint_bids_eye, 243
prepare_gazepoint_bids_physio, 243
prepare_gazepoint_biometrics_lme_data, 244
prepare_gazepoint_biosppy_input, 245
prepare_gazepoint_ctsi_input, 247
prepare_gazepoint_cvxeda_input, 249
prepare_gazepoint_eyetrackingr_input, 250
prepare_gazepoint_gazer_input, 252
prepare_gazepoint_heartpy_input, 255
prepare_gazepoint_ledalab_input, 256
prepare_gazepoint_mixed_model_data, 257
prepare_gazepoint_mne_events, 258
prepare_gazepoint_mne_input, 260
prepare_gazepoint_mne_input(), *368, 369*
prepare_gazepoint_multimodal_model_data, 261
prepare_gazepoint_neurokit_eda_input, 262
prepare_gazepoint_pspm_dcm_input, 262
prepare_gazepoint_pspm_input, 263
prepare_gazepoint_pupillometryr_input, 264
prepare_gazepoint_pyhrv_input, 267
prepare_gazepoint_pyppg_input, 269
prepare_gazepoint_rhrv_input, 270
prepare_gazepoint_rhrv_input(), *269*
prepare_gazepoint_scr_hurdle_model_data, 271
prepare_gazepoint_timecourse_test_data, 272
prepare_gazepoint_timecourse_test_data(), *291, 292*
preprocess_gazepoint_all, 274
preprocess_gazepoint_scr_pspm_style, 275
print.gazepoint_biometrics_audit *(*audit_gazepoint_biometrics_file*)*, 28
process_gazepoint_ppg_heartpy_style, 276
process_gazepoint_ppg_segmentwise, 277
profile_gazepoint_export_folder, 278

recommend_gazepoint_biometric_exclusions, 279
reconstruct_gazepoint_ppg_clipping, 280
regress_gazepoint_pupil_luminance, 281
reject_gazepoint_ppg_peaks, 282
remove_gazepoint_ppg_baseline_wander, 282
report_gazepoint_cluster_permutation, 283
report_gazepoint_data_quality, 284
respiration_from_ppg, 285
run_gazepoint_automated_statistics, 285
run_gazepoint_biometrics_real_data_readiness, 286
run_gazepoint_biometrics_workflow, 287

run_gazepoint_biometrics_workflow(), *302*
run_gazepoint_biosppy_eda, 289
run_gazepoint_biosppy_eda(), *247*
run_gazepoint_biosppy_ppg, 290
run_gazepoint_biosppy_ppg(), *247*
run_gazepoint_cluster_permutation, 291
run_gazepoint_cluster_permutation(), *48*, *154*, *225*, *226*, *283*, *294*, *295*, *356*
run_gazepoint_cluster_permutation_anova, 292
run_gazepoint_cluster_permutation_covariate_adjusted, 293
run_gazepoint_cluster_permutation_lmer, 293
run_gazepoint_cluster_permutation_parallel, 294
run_gazepoint_cluster_threshold_sensitivity, 294
run_gazepoint_eda_analysis_pipeline, 295
run_gazepoint_heartpy_crosscheck, 297
run_gazepoint_multidimensional_cluster_permutation, 298
run_gazepoint_neurokit_eda_crosscheck, 298
run_gazepoint_online_design_optimization, 299
run_gazepoint_pyhrv_style, 300
run_gazepoint_pyhrv_style(), *269*
run_gazepoint_real_data_smoke, 301
run_gazepoint_real_data_smoke(), *46*, *369*
run_gazepoint_scr_multiverse, 303
run_gazepoint_scr_threshold_sensitivity, 304
run_gazepoint_tfce, 306
run_gpbiometrics_shiny, 306
run_gpbiometrics_shiny_annotator, 307

scale_gazepoint_ppg_sections, 307
scale_gazepoint_ppg_signal, 308
scr_responsivity_sensitivity, 309
screen_gazepoint_eda_nonresponders, 309
segment_gazepoint_pyhrv_nni, 310
session_info_gazepoint, 311
simulate_gazepoint_artifact, 312
simulate_gazepoint_biometrics, 313
simulate_gazepoint_cluster_timecourse_data, 314
simulate_gazepoint_eye_data, 315
simulate_gazepoint_multimodal_data, 316
smooth_gazepoint_biometrics, 316
smooth_gazepoint_ppg_signal, 317
smooth_gazepoint_pupil, 318
split_gazepoint_sessions_pspm_style, 319
standardise_gazepoint_adaptive_ema, 319
standardise_gazepoint_biometric_names, 321
standardise_gazepoint_biometrics_within_unit *(*standardize_gazepoint_biometrics_within_unit*)*, 324
standardise_gazepoint_plot_contract, 322
standardise_gazepoint_plot_contract(), *327*
standardise_gazepoint_range_correction, 322
standardise_gazepoint_zscore, 323
standardize_gazepoint_adaptive_ema *(*standardise_gazepoint_adaptive_ema*)*, 319
standardize_gazepoint_biometrics_within_unit, 324
standardize_gazepoint_column_names, 326
standardize_gazepoint_columns, 327
standardize_gazepoint_plot_contracts, 327
standardize_gazepoint_range_correction *(*standardise_gazepoint_range_correction*)*, 322
standardize_gazepoint_zscore *(*standardise_gazepoint_zscore*)*, 323
stats::cor(), *28*, *151*
summarise_gazepoint_aoi_biometrics, 328
summarise_gazepoint_aoi_biometrics(), *24*
summarise_gazepoint_biometric_validity, 329
summarise_gazepoint_biometrics_feature_inventory,

330
summarise_gazepoint_biometrics_workflow, 330
summarise_gazepoint_biometrics_workflow(), *302*
summarise_gazepoint_decision_log, 331
summarise_gazepoint_dial_windows, 331
summarise_gazepoint_engagement_windows, 332
summarise_gazepoint_engagement_windows(), *331*, *332*
summarise_gazepoint_fixations_by_aoi, 333
summarise_gazepoint_fixations_by_aoi(), *349*
summarise_gazepoint_full_biometric_windows, 334
summarise_gazepoint_gsr_tonic_phasic, 335
summarise_gazepoint_gsr_windows, 336
summarise_gazepoint_hr_windows, 337
summarise_gazepoint_hrv_features, 337
summarise_gazepoint_ibi_hrv_windows, 339
summarise_gazepoint_ibi_windows, 340
summarise_gazepoint_multimodal_windows, 341
summarise_gazepoint_multimodal_windows(), *261*
summarise_gazepoint_scr_event_windows, 342
summarize_gazepoint_aoi_dwell, 344
summarize_gazepoint_aoi_dwell(), *24*
summarize_gazepoint_audit_trail, 344
summarize_gazepoint_beat_corrections, 345
summarize_gazepoint_eventlocked_multimodal, 346
summarize_gazepoint_export_inventory, 347
summarize_gazepoint_feature_coverage, 347
summarize_gazepoint_fixations, 348
summarize_gazepoint_fixations(), *24*, *129*
summarize_gazepoint_fixations_by_aoi, 349
summarize_gazepoint_missingness, 350
summarize_gazepoint_nonwear, 351
summarize_gazepoint_preregistration_readiness, 351
summarize_gazepoint_pupil_events, 352
summarize_gazepoint_qc_overview, 353
summarize_gazepoint_scanpath_metrics, 353
summarize_gazepoint_scr_recovery, 354
summarize_gazepoint_signal_quality, 355
summarize_gazepoint_time_clusters, 356
summarize_gazepoint_tracking, 356
summary.gazepoint_biometrics_audit *(*audit_gazepoint_biometrics_file*)*, 28
sync_gazepoint_biometrics_with_gaze, 357
sync_gazepoint_signals_via_lsl, 358
sync_gazepoint_signals_via_lsl(), *148*

test_gazepoint_hrv_nonlinearity, 359
trim_gazepoint_biometrics_pspm_style, 360

upsample_gazepoint_data, 360

validate_gazepoint_biometrics, 361
validate_gazepoint_format, 362
validate_gazepoint_gaze, 363
validate_gazepoint_metadata, 364

write_gazepoint_biometrics_report_tables, 365
write_gazepoint_decision_log, 366
write_gazepoint_export_profile, 366
write_gazepoint_interoperability_audit, 367
write_gazepoint_mne_fif, 368
write_gazepoint_real_data_smoke, 369