

Package ‘gp3tools’

August 21, 2026

Type Package

Title Import, Inspect, Analyse, and Report Gazeport GP3 Exports

Version 2.3.0

Description Tools for importing, inspecting, cleaning, summarising, modelling, and reporting Gazeport GP3 and Gazeport Analysis CSV exports. The package supports offline workflows for all-gaze, fixation, pupil, area-of-interest, transition, time-course, quality-audit, and manuscript-reporting analyses.
The package methodology is described in the peer-reviewed software paper <[doi:10.3390/jemr19040076](https://doi.org/10.3390/jemr19040076)>.

License MIT + file LICENSE

LazyData true

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports dplyr, ggplot2, readr, rlang, stringr, tibble, tidyr

Suggests brms, cluster, DHARMA, emmeans, eyetools, glmmTMB, knitr, lme4, magick, mgcv, png, pracma, rmarkdown, shiny, testthat (>= 3.0.0), TraMineR

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

URL <https://github.com/stefanosbalaskas/gp3tools>,
<https://stefanosbalaskas.github.io/gp3tools/>,
<https://doi.org/10.5281/zenodo.21292384>,
<https://doi.org/10.3390/jemr19040076>

BugReports <https://github.com/stefanosbalaskas/gp3tools/issues>

Config/Needs/website rmarkdown

NeedsCompilation no

Author Stefanos Balaskas [aut, cre] (ORCID:
<https://orcid.org/0000-0003-2444-9796>)
Maintainer Stefanos Balaskas <s.balaskas@ac.upatras.gr>
Repository CRAN
Date/Publication 2026-08-21 05:43:15 UTC

Contents

add_gazepoint_aoi	8
add_gazepoint_dynamic_aoi	9
add_gazepoint_polygon_aoi	11
analyse_gazepoint_binocular_sensitivity	12
analyze_gazepoint_window	14
as_gazepoint_master	15
audit_gazepoint_aoi_coding_matrix	16
audit_gazepoint_aoi_geometry	19
audit_gazepoint_aoi_margin_sensitivity	20
audit_gazepoint_aoi_overlap	22
audit_gazepoint_aoi_screen_coverage	23
audit_gazepoint_aoi_window_denominators	24
audit_gazepoint_binocular_reconstruction	26
audit_gazepoint_condition_quality_imbalance	27
audit_gazepoint_design_balance	28
audit_gazepoint_dynamic_aoi_coverage	29
audit_gazepoint_event_sync	30
audit_gazepoint_exclusion_flow	31
audit_gazepoint_face_quality	32
audit_gazepoint_face_sync	33
audit_gazepoint_fixation_reliability	35
audit_gazepoint_gaze_signal_quality	36
audit_gazepoint_master	38
audit_gazepoint_naming_consistency	39
audit_gazepoint_post_exclusion_balance	39
audit_gazepoint_pupil_baseline	41
audit_gazepoint_pupil_drift	42
audit_gazepoint_pupil_gaps	44
audit_gazepoint_pupil_imbalance	45
audit_gazepoint_pupil_overlap_risk	46
audit_gazepoint_pupil_reliability	47
audit_gazepoint_screen_bounds	49
audit_gazepoint_stimulus_luminance	50
audit_gazepoint_timecourse_grid	51
baseline_correct_gazepoint_pupil	51
benchmark_gazepoint_event_detectors	53
benchmark_gazepoint_export_performance	55
bootstrap_gazepoint_scanpath_clusters	56
bootstrap_gazepoint_timecourse	58

check_gazepoint_bayesian_readiness	59
check_gazepoint_file_pairs	60
check_gazepoint_model_convergence	60
check_gazepoint_model_overdispersion	61
check_gazepoint_model_singularity	62
check_gazepoint_performance_regression	62
check_gazepoint_real_data_readiness	63
check_sampling_rate	65
classify_gazepoint_events_hmm	65
classify_gazepoint_export	66
clean_gazepoint_by_trackloss	67
cluster_gazepoint_scanpaths	68
collect_gazepoint_qc_summaries	70
combine_gazepoint_eyes	71
compare_gazepoint_event_detectors	72
compare_gazepoint_nested_models	73
compute_gazepoint_aoi_entropy	74
compute_gazepoint_aoi_sequence_metrics	75
compute_gazepoint_aoi_transition_matrix	77
compute_gazepoint_saccade_metrics	78
compute_gazepoint_scanpath_geometry	79
compute_gazepoint_scanpath_similarity	80
compute_gazepoint_sequence_complexity	81
compute_gazepoint_sequence_distance	82
compute_gazepoint_sequence_recurrence	83
compute_gazepoint_time_varying_transition_matrix	84
compute_gazepoint_transition_network_metrics	85
compute_transition_matrix	86
construct_gazepoint_combined_pupil	86
create_gazepoint_analysis_decision_audit	88
create_gazepoint_bayesian_sap	89
create_gazepoint_brms_template	90
create_gazepoint_cross_package_report	90
create_gazepoint_event_review_template	91
create_gazepoint_face_reporting_checklist	92
create_gazepoint_hddm_fit_script	93
create_gazepoint_markovchain_object	94
create_gazepoint_master	96
create_gazepoint_preprocessing_multiverse	97
create_gazepoint_preprocessing_registry	98
create_gazepoint_report	100
create_gazepoint_reporting_checklist	101
detect_gazepoint_blinks	102
detect_gazepoint_fixations_ivt	103
detect_gazepoint_fixations_velocity	104
diagnose_gazepoint_binocular_pupil	105
diagnose_gazepoint_cluster_design	107
diagnose_gazepoint_gamm	108

diagnose_gazepoint_glm	109
downsample_gazepoint_pupil	110
estimate_gazepoint_cluster_offset	111
estimate_gazepoint_cluster_onset	111
estimate_gazepoint_divergence_point	112
export_gazepoint_cluster_results	113
export_gazepoint_heatmap_png	114
export_gazepoint_master_audit	115
export_gazepoint_mne_cluster_input	116
export_gazepoint_model_tables	117
export_gazepoint_permuco_cluster_input	118
export_gazepoint_permutes_cluster_input	119
export_gazepoint_tables	120
export_gazepoint_to_bids	120
extract_gazepoint_representative_scanpaths	121
filter_gazepoint_cnn_uncertainty	122
fit_gazepoint_aoi_brms	123
fit_gazepoint_aoi_gamm	124
fit_gazepoint_aoi_model_sensitivity	126
fit_gazepoint_aoi_window_glm	127
fit_gazepoint_binocular_calibration	129
fit_gazepoint_brms_model	130
fit_gazepoint_face_window_lmm	131
fit_gazepoint_gca	132
fit_gazepoint_multimodal_response_model	133
fit_gazepoint_pupil_gamm	135
fit_gazepoint_pupil_pfe_gamm	136
fit_gazepoint_pupil_window_lmm	137
fit_gazepoint_pupil_window_sensitivity	139
fit_gazepoint_transition_count_nb_sensitivity	141
flag_gazepoint_pupil	142
flag_gazepoint_pupil_artifacts	143
flag_gazepoint_pupil_hampel	146
flag_gazepoint_sequence_anomalies	147
flag_tracking_quality	148
gazepoint_example_aoi_geometry	149
gazepoint_example_aoi_windows	150
gazepoint_example_fixations	150
gazepoint_example_master	151
gazepoint_example_pupil_windows	152
gp3tools_naming_policy	152
gp3tools_performance_limits	153
harmonize_gazepoint_screen_coordinates	153
impute_gazepoint_pupil_gp	154
inspect_gazepoint_columns	155
interpolate_gazepoint_blinks	156
interpolate_gazepoint_pupil	157
interpolate_gazepoint_pupil_pchip	158

launch_gazepoint_qc_dashboard	160
mean_gazepoint_pupil	160
plot_gazepoint_aoi_gamm	161
plot_gazepoint_aoi_timeline	163
plot_gazepoint_aoi_transition_matrix	164
plot_gazepoint_aoi_verification	165
plot_gazepoint_binocular_diagnostics	167
plot_gazepoint_cluster_null_distribution	168
plot_gazepoint_cluster_permutation	169
plot_gazepoint_cluster_results	169
plot_gazepoint_event_detector_agreement	171
plot_gazepoint_event_detector_benchmark	171
plot_gazepoint_face_quality	172
plot_gazepoint_gca	173
plot_gazepoint_heatmap	174
plot_gazepoint_heatmap_overlay	176
plot_gazepoint_missingness_profile	177
plot_gazepoint_model_predictions	178
plot_gazepoint_model_residuals	180
plot_gazepoint_multiverse_results	180
plot_gazepoint_phase_timeline	181
plot_gazepoint_pupil_preprocessing	182
plot_gazepoint_pupil_status	184
plot_gazepoint_pupil_timecourse	185
plot_gazepoint_qc_overview	186
plot_gazepoint_scanpath	187
plot_gazepoint_scanpaths	188
plot_gazepoint_scanpath_clusters	190
plot_gazepoint_scanpath_cluster_stability	191
plot_gazepoint_stimulus_layout_qc	192
plot_gazepoint_time_series	194
plot_gazepoint_time_varying_effect	195
plot_sampling_rate	196
plot_tracking_quality	197
plot_transition_heatmap	197
prepare_gazepoint_aoi_gamm_data	198
prepare_gazepoint_aoi_glmm_data	200
prepare_gazepoint_aoi_sequences	201
prepare_gazepoint_cluster_data	202
prepare_gazepoint_eyetools_data	204
prepare_gazepoint_eyetrackingr_data	205
prepare_gazepoint_fixation_aligned_data	207
prepare_gazepoint_gazer_data	208
prepare_gazepoint_gca_data	210
prepare_gazepoint_gpbiometrics_bridge	211
prepare_gazepoint_hddm_export	212
prepare_gazepoint_heatmap_data	213
prepare_gazepoint_hmm_data	214

prepare_gazepoint_multimodal_data	215
prepare_gazepoint_pupillometryr_data	217
prepare_gazepoint_pupil_gamm_data	218
prepare_gazepoint_pupil_window_model_data	220
prepare_gazepoint_semimarkov_data	221
prepare_gazepoint_timecourse_test_data	223
prepare_gazepoint_traminer_data	224
preprocess_gazepoint_signals	225
read_gazepoint	227
read_gazepoint_face_export	228
read_gazepoint_folder	229
read_gazepoint_summary	230
recalibrate_gazepoint_gaze	230
recommend_gazepoint_exclusions	232
recommend_gazepoint_model_family	233
reconstruct_gazepoint_binocular_pupil	234
regress_gazepoint_pupils	236
report_gazepoint_cluster_permutation	237
report_gazepoint_face_qc	238
report_gazepoint_missingness	239
report_gazepoint_multiverse	240
report_gazepoint_phase_coverage	241
report_gazepoint_qc_overview	242
run_gazepoint_aoi_multiverse	242
run_gazepoint_cluster_permutation	244
run_gazepoint_cluster_permutation_anova	245
run_gazepoint_cluster_permutation_covariate_adjusted	245
run_gazepoint_cluster_permutation_lmer	246
run_gazepoint_cluster_permutation_parallel	246
run_gazepoint_cluster_threshold_sensitivity	247
run_gazepoint_eyetools_fixation_detection	247
run_gazepoint_gazer_crosscheck	249
run_gazepoint_gpbiometrics_workflow	250
run_gazepoint_model_leave_one_out	252
run_gazepoint_multidimensional_cluster_permutation	253
run_gazepoint_pupil_multiverse	253
run_gazepoint_tfce	254
run_gazepoint_workflow	255
save_gazepoint_plots	256
segment_gazepoint_task_phases	257
select_gazepoint_adaptive_trial	259
select_gazepoint_scanpath_clusters	260
simulate_gazepoint_cluster_timecourse_data	261
simulate_gazepoint_data	262
simulate_gazepoint_fixations	263
simulate_gazepoint_pupil_data	264
smooth_gazepoint_coordinate	265
smooth_gazepoint_pupil	267

standardise_gazepoint_names	268
standardize_gazepoint_face_columns	269
stress_test_gazepoint_binocular_reconstruction	270
summarise_aoi_samples	271
summarise_fixations	272
summarise_gazepoint_aoi	272
summarise_gazepoint_aoi_entries	273
summarise_gazepoint_aoi_transitions	274
summarise_gazepoint_aoi_trial_features	275
summarise_gazepoint_aoi_windows	276
summarise_gazepoint_binocular_reporting	278
summarise_gazepoint_clusters	279
summarise_gazepoint_coordinate_coverage	280
summarise_gazepoint_emmeans	280
summarise_gazepoint_event_detector_agreement	281
summarise_gazepoint_event_detector_benchmark	282
summarise_gazepoint_face_reactivity	282
summarise_gazepoint_face_windows	283
summarise_gazepoint_fixation_trials	283
summarise_gazepoint_fixed_effects	285
summarise_gazepoint_markovchain	286
summarise_gazepoint_multiverse_results	287
summarise_gazepoint_pupil	287
summarise_gazepoint_pupil_response_features	288
summarise_gazepoint_pupil_trial_features	289
summarise_gazepoint_pupil_windows	290
summarise_gazepoint_scanpath_cluster_stability	292
summarise_gazepoint_semimarkov	293
summarise_gazepoint_time_clusters	294
summarise_gazepoint_workflow	294
summarise_tracking_quality	295
summarize_gazepoint_coordinate_coverage	295
summarize_gazepoint_face_quality	297
summarize_gazepoint_face_reactivity	297
summarize_gazepoint_face_windows	299
summarize_gazepoint_missingness	301
summarize_gazepoint_phase_coverage	302
summarize_gazepoint_pupil_response_features	303
summarize_gazepoint_qc_status	304
summarize_gazepoint_time_clusters	304
sync_gazepoint_face_data	305
tidy_gazepoint_model_summary	307
transform_gazepoint_aoi_empirical_logit	308
validate_gazepoint_binocular_reconstruction	309
validate_gazepoint_master	311
write_gazepoint_naming_audit	313
write_gazepoint_outputs	313
write_gazepoint_performance_benchmark	314

add_gazepoint_aoi	<i>Add rectangular AOI membership to gaze data</i>
-------------------	--

Description

Labels gaze samples using one or more rectangular AOI definitions.

Usage

```
add_gazepoint_aoi(  
  master_df,  
  aoi_defs,  
  x_col = "FPOGX",  
  y_col = "FPOGY",  
  aoi_name = NULL,  
  output = c("logical", "label", "both"),  
  prefix = "aoi_",  
  label_col = "aoi_current",  
  outside_label = "outside",  
  overlap = c("first", "last", "error"),  
  include_overlap_count = TRUE  
)
```

Arguments

master_df	A sample-level gaze data frame.
aoi_defs	AOI definition data frame. Recognised aliases include name/aoi_name, L/left/xmin, R/right/xmax, T/top/ymin, and B/bottom/ymax.
x_col, y_col	Gaze-coordinate columns.
aoi_name	Optional AOI name selecting one row from aoi_defs.
output	Add logical AOI columns, a single label column, or both.
prefix	Prefix for logical AOI columns.
label_col	Name of the single AOI-label column.
outside_label	Label for samples outside all AOIs.
overlap	Overlap handling for the label column.
include_overlap_count	Add the number of AOIs containing each sample.

Value

The input data with AOI membership columns.

Examples

```
gaze <- data.frame(
  FPOGX = c(0.2, 0.5, 0.8),
  FPOGY = c(0.2, 0.5, 0.8)
)
defs <- data.frame(
  name = c("top_left", "bottom_right"),
  L = c(0, 0.6),
  R = c(0.4, 1),
  T = c(0, 0.6),
  B = c(0.4, 1)
)
add_gazepoint_aoi(gaze, defs, output = "both")
```

add_gazepoint_dynamic_aoi

Add time-varying AOI membership to Gazepoint data

Description

Match each gaze sample to a time-indexed AOI definition and classify the sample against rectangular or polygon geometry. Definitions can also be grouped by participant, stimulus, trial, or other shared columns.

Usage

```
add_gazepoint_dynamic_aoi(
  master_df,
  aoi_defs,
  x_col = "FPOGX",
  y_col = "FPOGY",
  time_col = "TIME",
  aoi_time_col = "aoi_time",
  aoi_name_col = "aoi_name",
  shape = c("auto", "rectangle", "polygon"),
  group_cols = NULL,
  match = c("nearest", "previous", "next"),
  max_time_gap = Inf,
  left_col = "left",
  right_col = "right",
  top_col = "top",
  bottom_col = "bottom",
  vertex_x_col = "vertex_x",
  vertex_y_col = "vertex_y",
  vertex_order_col = NULL,
  output = c("label", "logical", "both"),
  prefix = "aoi_",
```

```

    label_col = "aoi_current",
    outside_label = "outside",
    overlap = c("first", "last", "error"),
    boundary = c("inside", "outside"),
    definition_time_col = "aoi_definition_time",
    time_gap_col = "aoi_time_gap",
    include_overlap_count = TRUE
)

```

Arguments

master_df	A sample-level gaze data frame.
aoi_defs	Time-indexed AOI definitions.
x_col, y_col	Gaze-coordinate columns in master_df.
time_col	Sample timestamp column.
aoi_time_col	Definition timestamp column.
aoi_name_col	AOI-name column.
shape	Geometry type: "auto", "rectangle", or "polygon".
group_cols	Columns shared by master_df and aoi_defs that define independent dynamic-definition streams.
match	Time-matching rule: nearest, previous, or next definition.
max_time_gap	Maximum permitted absolute gap between a sample and its matched definition timestamp, in native time units.
left_col, right_col, top_col, bottom_col	Rectangle boundary columns.
vertex_x_col, vertex_y_col	Polygon vertex columns.
vertex_order_col	Optional polygon vertex-order column.
output	Add logical columns, a label column, or both.
prefix	Prefix for logical AOI columns.
label_col	AOI-label column.
outside_label	Label used outside all AOIs.
overlap	Overlap handling.
boundary	Polygon-boundary handling.
definition_time_col, time_gap_col	Names of dynamic-match diagnostics.
include_overlap_count	Add the number of containing AOIs.

Value

The input data with dynamic AOI membership and match-diagnostic columns.

add_gazepoint_polygon_aoi

Add polygon AOI membership to Gazepoint data

Description

Classify sample-level gaze coordinates against one or more polygon AOIs using a base-R ray-casting implementation. Polygon vertices are grouped by AOI name and ordered either by row order or an explicit vertex-order column.

Usage

```
add_gazepoint_polygon_aoi(
  master_df,
  vertices,
  x_col = "FPOGX",
  y_col = "FPOGY",
  aoi_col = "aoi_name",
  vertex_x_col = "vertex_x",
  vertex_y_col = "vertex_y",
  vertex_order_col = NULL,
  output = c("label", "logical", "both"),
  prefix = "aoi_",
  label_col = "aoi_current",
  outside_label = "outside",
  overlap = c("first", "last", "error"),
  boundary = c("inside", "outside"),
  include_overlap_count = TRUE
)
```

Arguments

master_df	A sample-level gaze data frame.
vertices	Polygon-vertex data frame.
x_col, y_col	Gaze-coordinate columns.
aoi_col	AOI-name column in vertices.
vertex_x_col, vertex_y_col	Vertex-coordinate columns.
vertex_order_col	Optional vertex-order column.
output	Add logical AOI columns, one label column, or both.
prefix	Prefix for logical AOI columns.
label_col	Name of the AOI-label column.
outside_label	Label used outside all polygons.

overlap Overlap handling for the label column.
 boundary Should points on polygon boundaries be treated as inside?
 include_overlap_count Add the number of containing AOIs.

Value

The input data with polygon AOI membership columns.

Examples

```

gaze <- data.frame(
  FPOGX = c(0.2, 0.7, 0.9),
  FPOGY = c(0.2, 0.7, 0.1)
)

triangle <- data.frame(
  aoi_name = "triangle",
  vertex_x = c(0, 1, 0),
  vertex_y = c(0, 0, 1)
)

add_gazepoint_polygon_aoi(
  gaze,
  triangle,
  output = "both"
)

```

analyse_gazepoint_binocular_sensitivity

Compare pupil-construction policies as a sensitivity analysis

Description

Constructs the same pupil stream under multiple declared policies and compares data retention, descriptive distribution summaries, condition means/contrasts, and pairwise series correlations. It deliberately avoids converting policy disagreement into an automatic inferential verdict.

Usage

```

analyse_gazepoint_binocular_sensitivity(
  data,
  left_col,
  right_col,
  policies = c("complete_case", "available_eye", "reconstructed_mean", "left_only",
    "right_only"),
  prefix = "gp3_binocular",
  group_cols = NULL,

```

```

    condition_col = NULL,
    valid_min = NULL,
    valid_max = NULL
  )

```

Arguments

<code>data</code>	Raw data or reconstruction output. <code>reconstructed_mean</code> requires reconstruction columns.
<code>left_col, right_col</code>	Original pupil channels.
<code>policies</code>	Any of "complete_case", "available_eye", "reconstructed_mean", "left_only", and "right_only".
<code>prefix</code>	Reconstruction prefix.
<code>group_cols</code>	Optional grouping columns for descriptive summaries.
<code>condition_col</code>	Optional condition column. When supplied, simple descriptive mean contrasts relative to the first observed condition are returned; no p-values are calculated.
<code>valid_min, valid_max</code>	Optional raw-channel bounds.

Value

A `gp3_binocular_sensitivity` object containing policy summaries, pairwise correlations, optional condition summaries/contrasts, and the constructed long series.

See Also

[audit_gazepoint_binocular_reconstruction\(\)](#)

Examples

```

dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 24)
dat$pupil_right[20:24] <- NA_real_
rec <- reconstruct_gazepoint_binocular_pupil(
  dat, "pupil_left", "pupil_right", group_cols = "subject", min_pairs = 20
)
analyse_gazepoint_binocular_sensitivity(
  rec, "pupil_left", "pupil_right", condition_col = "condition"
)

```

analyze_gazepoint_window

Summarise gaze or pupil measures in sliding time windows

Description

Creates overlapping or non-overlapping time windows and calculates selected summary statistics for numeric gaze or pupil columns.

Usage

```
analyze_gazepoint_window(
  et_data,
  window_size = 50,
  step = 10,
  summary_stats = c("mean", "sd"),
  by = "USER_ID",
  condition_col = NULL,
  value_cols = NULL,
  ts_col = "TIME",
  window_unit = c("milliseconds", "seconds", "native"),
  time_unit = c("auto", "seconds", "milliseconds"),
  include_partial = FALSE
)
```

Arguments

et_data	A sample-level data frame.
window_size	Window width.
step	Distance between consecutive window starts.
summary_stats	Statistics to calculate. Supported values are "mean", "sd", "median", "min", "max", "sum", and "valid_prop".
by	Grouping columns defining independent time series.
condition_col	Optional condition column appended to by.
value_cols	Numeric columns to summarise. When NULL, common gaze and pupil columns are detected.
ts_col	Timestamp column.
window_unit	Unit used by window_size and step.
time_unit	Unit of the timestamp column.
include_partial	Include a final window that is shorter than window_size.

Value

A tibble with one row per group and time window.

Examples

```
pupil <- data.frame(
  USER_ID = "P01",
  TIME = seq(0, 0.99, by = 0.01),
  mean_pupil = sin(seq(0, 2 * pi, length.out = 100))
)
analyze_gazepoint_window(
  pupil,
  window_size = 100,
  step = 50,
  value_cols = "mean_pupil"
)
```

as_gazepoint_master	<i>Convert Gazepoint all-gaze data to a master sample table</i>
---------------------	---

Description

Converts a Gazepoint all-gaze export into a standard sample-level table with one row per gaze sample. The returned table keeps Gazepoint identifiers but also adds analysis-friendly columns such as subject, media_id, time_ms, x, y, left_pupil, right_pupil, mean_pupil, valid_sample, missing_gaze, missing_pupil, trackloss, blink, aoi_current, message, and event_type.

Usage

```
as_gazepoint_master(
  data,
  screen_width_px = NULL,
  screen_height_px = NULL,
  source_col = "USER_FILE",
  media_col = "MEDIA_ID",
  media_name_col = "MEDIA_NAME",
  time_col = "TIME",
  coordinate_unit = c("auto", "normalised", "pixels"),
  event_latency_offset_ms = 0
)
```

Arguments

data	A Gazepoint all-gaze data frame, usually results\$all_gaze.
screen_width_px	Optional screen width in pixels. If supplied and gaze coordinates are detected as normalised 0-1 coordinates, x coordinates are converted to pixels.
screen_height_px	Optional screen height in pixels. If supplied and gaze coordinates are detected as normalised 0-1 coordinates, y coordinates are converted to pixels.
source_col	Column identifying the source/user file.

media_col	Column identifying the Gazepoint media/stimulus.
media_name_col	Column identifying the Gazepoint media/stimulus name.
time_col	Gazepoint time column, usually TIME.
coordinate_unit	One of "auto", "normalised", or "pixels". "auto" detects normalised coordinates when coordinate values are mostly between 0 and 1.
event_latency_offset_ms	Optional timing correction in milliseconds. Positive values shift event/sample time forward.

Details

This function is intended as a bridge between raw Gazepoint exports and more advanced eye-tracking workflows. It does not require an external trial log. Later, experiment-level information such as condition, trial ID, response, accuracy, or reaction time can be joined to the returned table.

Value

A tibble with one row per sample and standardised sample-level eye-tracking columns.

Examples

```
## Not run:
results <- run_gazepoint_workflow(
  export_dir = "C:/Users/YourName/Desktop/gp3_test_exports",
  output_dir = "C:/Users/YourName/Desktop/gp3_outputs"
)

master <- as_gazepoint_master(
  results$all_gaze,
  screen_width_px = 1920,
  screen_height_px = 1080
)

dplyr::glimpse(master)

## End(Not run)
```

```
audit_gazepoint_aoi_coding_matrix
```

Audit AOI coding against geometry

Description

Validate observed sample-level AOI labels against AOI labels derived from gaze coordinates and AOI geometry.

Usage

```

audit_gazepoint_aoi_coding_matrix(
  gaze_data,
  aoi_geometry,
  observed_aoi_col = NULL,
  gaze_x_col = NULL,
  gaze_y_col = NULL,
  gaze_stimulus_col = NULL,
  sample_id_cols = NULL,
  geometry_aoi_col = NULL,
  geometry_stimulus_col = NULL,
  x_min_col = NULL,
  y_min_col = NULL,
  x_max_col = NULL,
  y_max_col = NULL,
  x_col = NULL,
  y_col = NULL,
  width_col = NULL,
  height_col = NULL,
  screen_x_range = c(0, 1),
  screen_y_range = c(0, 1),
  tie_method = c("ambiguous", "first"),
  outside_label = "outside",
  ambiguous_label = "ambiguous",
  missing_label = "missing_coordinate",
  observed_outside_values = c("outside", "none", "no_aoi", "non_aoi", "background",
    "off_aoi"),
  max_mismatch_prop = 0.05,
  max_ambiguous_prop = 0.05,
  max_missing_coordinate_prop = 0.2,
  ignore_invalid_geometry = TRUE
)

```

Arguments

<code>gaze_data</code>	A data frame containing gaze samples and observed AOI labels.
<code>aoi_geometry</code>	A data frame containing AOI geometry definitions.
<code>observed_aoi_col</code>	Observed AOI label column in <code>gaze_data</code> .
<code>gaze_x_col</code>	Gaze x-coordinate column. If <code>NULL</code> , common aliases are detected automatically.
<code>gaze_y_col</code>	Gaze y-coordinate column. If <code>NULL</code> , common aliases are detected automatically.
<code>gaze_stimulus_col</code>	Optional gaze stimulus/media column.
<code>sample_id_cols</code>	Optional columns to carry into the sample-level coding audit table.
<code>geometry_aoi_col</code>	AOI label/name column in <code>aoi_geometry</code> .

geometry_stimulus_col	Optional stimulus/media column in aoi_geometry.
x_min_col	Optional AOI left/x-min column.
y_min_col	Optional AOI top/y-min column.
x_max_col	Optional AOI right/x-max column.
y_max_col	Optional AOI bottom/y-max column.
x_col	Optional AOI left/x column used with width_col.
y_col	Optional AOI top/y column used with height_col.
width_col	Optional AOI width column.
height_col	Optional AOI height column.
screen_x_range	Numeric length-2 vector defining the screen x range.
screen_y_range	Numeric length-2 vector defining the screen y range.
tie_method	How to handle samples falling in multiple AOIs. Use "ambiguous" to label them as ambiguous or "first" to use the first matching AOI.
outside_label	Label used for samples outside all AOIs.
ambiguous_label	Label used for samples inside multiple AOIs when tie_method = "ambiguous".
missing_label	Label used for samples with missing/non-finite gaze coordinates.
observed_outside_values	Character values in observed_aoi_col treated as outside/non-AOI labels.
max_mismatch_prop	Maximum acceptable proportion of comparable samples where observed and geometry-derived AOI labels differ.
max_ambiguous_prop	Maximum acceptable proportion of samples with ambiguous geometry-derived AOI assignment.
max_missing_coordinate_prop	Maximum acceptable proportion of samples with missing/non-finite gaze coordinates.
ignore_invalid_geometry	Logical. If TRUE, AOIs with invalid coordinates or dimensions are excluded before coding validation.

Value

A list with class `gp3_aoi_coding_matrix_audit` containing overview, geometry_summary, sample_coding, coding_matrix, observed_summary, derived_summary, flagged_samples, and settings tables.

audit_gazepoint_aoi_geometry
Audit AOI geometry definitions

Description

Create a publication-level audit of AOI geometry definitions, including AOI size, area, coordinate validity, screen-bound checks, and duplicate geometry.

Usage

```
audit_gazepoint_aoi_geometry(
  data,
  aoi_col = NULL,
  stimulus_col = NULL,
  x_min_col = NULL,
  y_min_col = NULL,
  x_max_col = NULL,
  y_max_col = NULL,
  x_col = NULL,
  y_col = NULL,
  width_col = NULL,
  height_col = NULL,
  screen_x_range = c(0, 1),
  screen_y_range = c(0, 1),
  min_width = 0,
  min_height = 0,
  min_area = 0,
  max_area_prop = 1,
  require_within_screen = TRUE
)
```

Arguments

data	A data frame containing AOI geometry definitions.
aoi_col	AOI label/name column. If NULL, common AOI-name aliases are detected automatically.
stimulus_col	Optional stimulus/media column.
x_min_col	Optional AOI left/x-min column.
y_min_col	Optional AOI top/y-min column.
x_max_col	Optional AOI right/x-max column.
y_max_col	Optional AOI bottom/y-max column.
x_col	Optional AOI left/x column used with width_col.
y_col	Optional AOI top/y column used with height_col.

width_col	Optional AOI width column.
height_col	Optional AOI height column.
screen_x_range	Numeric length-2 vector defining the screen x range.
screen_y_range	Numeric length-2 vector defining the screen y range.
min_width	Minimum acceptable AOI width.
min_height	Minimum acceptable AOI height.
min_area	Minimum acceptable AOI area.
max_area_prop	Maximum acceptable AOI area as a proportion of screen area.
require_within_screen	Logical. If TRUE, AOIs extending outside the screen range are flagged.

Value

A list with class `gp3_aoi_geometry_audit` containing overview, geometry_summary, size_summary, flagged_aois, duplicate_geometry, and settings tables.

audit_gazepoint_aoi_margin_sensitivity
<i>Audit AOI margin sensitivity</i>

Description

Audit whether sample-level AOI assignments are sensitive to small expansions or shrinkages of AOI boundaries.

Usage

```
audit_gazepoint_aoi_margin_sensitivity(
  gaze_data,
  aoi_geometry,
  gaze_x_col = NULL,
  gaze_y_col = NULL,
  gaze_stimulus_col = NULL,
  sample_id_cols = NULL,
  geometry_aoi_col = NULL,
  geometry_stimulus_col = NULL,
  x_min_col = NULL,
  y_min_col = NULL,
  x_max_col = NULL,
  y_max_col = NULL,
  x_col = NULL,
  y_col = NULL,
  width_col = NULL,
  height_col = NULL,
  margins = c(-0.02, 0, 0.02),
```

```

    screen_x_range = c(0, 1),
    screen_y_range = c(0, 1),
    tie_method = c("ambiguous", "first"),
    outside_label = "outside",
    ambiguous_label = "ambiguous",
    missing_label = "missing_coordinate",
    max_margin_change_prop = 0.1,
    max_ambiguous_prop = 0.05,
    ignore_invalid_geometry = TRUE
  )

```

Arguments

<code>gaze_data</code>	A data frame containing gaze samples.
<code>aoi_geometry</code>	A data frame containing AOI geometry definitions.
<code>gaze_x_col</code>	Gaze x-coordinate column. If NULL, common aliases are detected automatically.
<code>gaze_y_col</code>	Gaze y-coordinate column. If NULL, common aliases are detected automatically.
<code>gaze_stimulus_col</code>	Optional gaze stimulus/media column.
<code>sample_id_cols</code>	Optional columns to carry into the sample-level audit table.
<code>geometry_aoi_col</code>	AOI label/name column in <code>aoi_geometry</code> .
<code>geometry_stimulus_col</code>	Optional stimulus/media column in <code>aoi_geometry</code> .
<code>x_min_col</code>	Optional AOI left/x-min column.
<code>y_min_col</code>	Optional AOI top/y-min column.
<code>x_max_col</code>	Optional AOI right/x-max column.
<code>y_max_col</code>	Optional AOI bottom/y-max column.
<code>x_col</code>	Optional AOI left/x column used with <code>width_col</code> .
<code>y_col</code>	Optional AOI top/y column used with <code>height_col</code> .
<code>width_col</code>	Optional AOI width column.
<code>height_col</code>	Optional AOI height column.
<code>margins</code>	Numeric vector of AOI boundary margins. Positive values expand AOIs; negative values shrink AOIs. A zero-margin baseline is always added.
<code>screen_x_range</code>	Numeric length-2 vector defining the screen x range.
<code>screen_y_range</code>	Numeric length-2 vector defining the screen y range.
<code>tie_method</code>	How to handle samples falling in multiple AOIs. Use "ambiguous" to label them as ambiguous or "first" to use the first matching AOI.
<code>outside_label</code>	Label for samples outside all AOIs.
<code>ambiguous_label</code>	Label for samples inside multiple AOIs when <code>tie_method = "ambiguous"</code> .
<code>missing_label</code>	Label for samples with missing/non-finite gaze coordinates.

max_margin_change_prop
Maximum acceptable proportion of samples whose AOI assignment changes from the zero-margin baseline.

max_ambiguous_prop
Maximum acceptable proportion of ambiguous samples.

ignore_invalid_geometry
Logical. If TRUE, AOIs with invalid coordinates or dimensions are excluded before margin coding.

Value

A list with class `gp3_aoi_margin_sensitivity_audit` containing overview, geometry_summary, sample_sensitivity, margin_summary, aoi_margin_summary, flagged_samples, and settings tables.

audit_gazepoint_aoi_overlap
Audit AOI overlap

Description

Create a publication-level audit of pairwise AOI overlap within each stimulus.

Usage

```
audit_gazepoint_aoi_overlap(
  data,
  aoi_col = NULL,
  stimulus_col = NULL,
  x_min_col = NULL,
  y_min_col = NULL,
  x_max_col = NULL,
  y_max_col = NULL,
  x_col = NULL,
  y_col = NULL,
  width_col = NULL,
  height_col = NULL,
  screen_x_range = c(0, 1),
  screen_y_range = c(0, 1),
  min_overlap_area = 0,
  min_overlap_prop = 0,
  ignore_invalid_geometry = TRUE
)
```

Arguments

<code>data</code>	A data frame containing AOI geometry definitions.
<code>aoi_col</code>	AOI label/name column. If NULL, common AOI-name aliases are detected automatically by <code>audit_gazepoint_aoi_geometry()</code> .
<code>stimulus_col</code>	Optional stimulus/media column.
<code>x_min_col</code>	Optional AOI left/x-min column.
<code>y_min_col</code>	Optional AOI top/y-min column.
<code>x_max_col</code>	Optional AOI right/x-max column.
<code>y_max_col</code>	Optional AOI bottom/y-max column.
<code>x_col</code>	Optional AOI left/x column used with <code>width_col</code> .
<code>y_col</code>	Optional AOI top/y column used with <code>height_col</code> .
<code>width_col</code>	Optional AOI width column.
<code>height_col</code>	Optional AOI height column.
<code>screen_x_range</code>	Numeric length-2 vector defining the screen x range.
<code>screen_y_range</code>	Numeric length-2 vector defining the screen y range.
<code>min_overlap_area</code>	Minimum overlap area above which an AOI pair is flagged.
<code>min_overlap_prop</code>	Minimum overlap proportion above which an AOI pair is flagged. This is computed relative to the smaller AOI in the pair.
<code>ignore_invalid_geometry</code>	Logical. If TRUE, AOIs with invalid geometry are excluded from pairwise overlap calculations.

Value

A list with class `gp3_aoi_overlap_audit` containing `overview`, `geometry_summary`, `pairwise_overlap`, `overlap_summary`, `flagged_overlaps`, and `settings` tables.

`audit_gazepoint_aoi_screen_coverage`

Audit AOI coverage against screen bounds

Description

Checks rectangular AOIs against declared screen or stimulus dimensions. The helper reports missing geometry, invalid rectangle order, off-screen AOIs, raw AOI area, clipped on-screen area, and screen-coverage rates. Coverage summaries are descriptive and do not correct for AOI overlap.

Usage

```
audit_gazepoint_aoi_screen_coverage(
  data,
  screen_width,
  screen_height,
  aoi_col = NULL,
  x_min_col = "x_min",
  x_max_col = "x_max",
  y_min_col = "y_min",
  y_max_col = "y_max",
  margin = 0
)
```

Arguments

data A data frame containing AOI geometry.

screen_width, screen_height Numeric screen or stimulus dimensions.

aoi_col Optional AOI identifier column.

x_min_col, x_max_col, y_min_col, y_max_col Character names of rectangle boundary columns.

margin Numeric tolerance around screen bounds.

Value

A list with AOI-level summary, overall summary, and settings.

Examples

```
aoi <- data.frame(
  aoi = c("left", "right"),
  x_min = c(100, 1200),
  x_max = c(500, 1700),
  y_min = c(100, 100),
  y_max = c(400, 400)
)
audit_gazepoint_aoi_screen_coverage(aoi, 1920, 1080, aoi_col = "aoi")
```

audit_gazepoint_aoi_window_denominators

Audit AOI window denominators before binomial modelling

Description

Audit AOI-window sample denominators before confirmatory binomial or logistic mixed-effects modelling. The function is designed for output from `summarise_gazepoint_aoi_windows()`.

Usage

```
audit_gazepoint_aoi_window_denominators(
  data,
  window_col = "window_label",
  window_start_col = "window_start_ms",
  window_end_col = "window_end_ms",
  denominator_col = "n_valid_denominator_samples",
  total_col = "n_window_samples",
  target_col = "n_target_samples",
  condition_col = "condition",
  group_cols = NULL,
  min_denominator_samples = 5,
  min_valid_denominator_prop = 0.7,
  max_denominator_cv = 0.25,
  max_condition_ratio = 2
)
```

Arguments

<code>data</code>	AOI-window summary data.
<code>window_col</code>	Name of the AOI-window label column.
<code>window_start_col</code>	Optional window-start column.
<code>window_end_col</code>	Optional window-end column.
<code>denominator_col</code>	Name of the denominator column to audit.
<code>total_col</code>	Name of the total window-sample column.
<code>target_col</code>	Name of the target-success count column.
<code>condition_col</code>	Optional condition column.
<code>group_cols</code>	Optional grouping columns for row-level audit context.
<code>min_denominator_samples</code>	Minimum acceptable denominator count.
<code>min_valid_denominator_prop</code>	Minimum acceptable valid-denominator proportion relative to total window samples.
<code>max_denominator_cv</code>	Maximum acceptable denominator coefficient of variation within each window.
<code>max_condition_ratio</code>	Maximum acceptable ratio between the largest and smallest mean denominator across conditions within a window.

Value

A named list containing overview, row audit, window summary, condition-window summary, denominator-imbalance summary, flagged rows, and settings.

audit_gazepoint_binocular_reconstruction

Audit binocular reconstruction burden and imbalance

Description

Summarises how much of the retained pupil signal depends on model-based reconstruction, where reconstruction was blocked, whether reconstruction is uneven across declared groups, and how much reconstruction shifts a simple available-eye combined signal.

Usage

```
audit_gazepoint_binocular_reconstruction(
  data,
  by = NULL,
  prefix = "gp3_binocular",
  max_reconstruction_prop = NULL,
  max_group_rate_difference = NULL
)
```

Arguments

data	Output from reconstruct_gazepoint_binocular_pupil() .
by	Optional condition, participant, stimulus, trial, or other columns used to assess reconstruction-rate imbalance.
prefix	Reconstruction prefix.
max_reconstruction_prop	Optional descriptive threshold above which overall reconstruction burden is flagged for review. NULL reports burden without imposing a universal cutoff.
max_group_rate_difference	Optional descriptive threshold for the maximum minus minimum reconstruction fraction across by groups. NULL reports imbalance without imposing a universal cutoff.

Details

Thresholds are governance flags, not statistical tests and not universal validity cutoffs.

Value

A `gp3_binocular_audit` object with overall burden, grouped burden, status counts, model diagnostics, and audit status.

See Also

[analyse_gazepoint_binocular_sensitivity\(\)](#), [summarise_gazepoint_binocular_reporting\(\)](#)

Examples

```

dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 23)
dat$pupil_left[35:38] <- NA_real_
rec <- reconstruct_gazepoint_binocular_pupil(
  dat, "pupil_left", "pupil_right", group_cols = "subject", min_pairs = 20
)
audit_gazepoint_binocular_reconstruction(rec, by = "condition")

```

```

audit_gazepoint_condition_quality_imbalance
  Audit condition-level quality imbalance

```

Description

Create a publication-level audit of whether gaze, pupil, retention, or other quality metrics differ across experimental conditions.

Usage

```

audit_gazepoint_condition_quality_imbalance(
  data,
  condition_col = "condition",
  quality_cols = NULL,
  subject_col = NULL,
  min_units_per_condition = 1L,
  max_mean_difference = 0.1,
  max_condition_ratio = 2,
  lower_is_better = c("missing_gaze_prop", "offscreen_prop", "excluded_prop",
    "failure_prop", "artifact_prop")
)

```

Arguments

<code>data</code>	A data frame containing condition-level, unit-level, or subject-condition-level quality metrics.
<code>condition_col</code>	Condition column.
<code>quality_cols</code>	Numeric quality-metric columns. If NULL, common quality columns are detected automatically.
<code>subject_col</code>	Optional subject column.
<code>min_units_per_condition</code>	Minimum number of rows/units expected per condition.
<code>max_mean_difference</code>	Maximum acceptable absolute difference between condition means for each quality metric.

max_condition_ratio	Maximum acceptable ratio between the largest and smallest non-zero condition mean for each quality metric.
lower_is_better	Optional character vector naming metrics where lower values indicate better quality, such as missing-gaze or exclusion metrics.

Value

A list with class `gp3_condition_quality_imbalance_audit` containing overview, condition_summary, metric_summary, flagged_metrics, and settings tables.

audit_gazepoint_design_balance	<i>Audit Gazepoint experimental design balance</i>
--------------------------------	--

Description

Create a publication-level audit of observed design balance across subjects, conditions, and optional stimulus/trial identifiers.

Usage

```
audit_gazepoint_design_balance(
  data,
  subject_col = "subject",
  condition_col = "condition",
  unit_cols = c("media_id", "trial_global"),
  expected_conditions = NULL,
  min_units_per_condition = 1L,
  max_condition_ratio = 2,
  require_all_conditions_per_subject = TRUE
)
```

Arguments

data	A data frame containing trial-level, window-level, or sample-level Gazepoint-derived data.
subject_col	Subject/participant identifier column.
condition_col	Experimental condition column.
unit_cols	Optional columns defining the repeated unit to count within each subject and condition, such as media, trial, block, or window.
expected_conditions	Optional character vector of expected condition labels.
min_units_per_condition	Minimum number of observed units expected per subject-condition cell.

max_condition_ratio	Maximum allowed ratio between a subject's largest and smallest non-zero condition counts.
require_all_conditions_per_subject	Logical. If TRUE, flag subjects who do not have all expected or observed conditions.

Value

A list with class gp3_design_balance_audit containing overview, subject_summary, condition_summary, cell_summary, imbalance_summary, flagged_cells, and settings tables.

audit_gazepoint_dynamic_aoi_coverage
<i>Audit time-varying AOI coverage</i>

Description

Summarise whether gaze samples could be matched to dynamic AOI definitions, how large the definition-time gaps were, and how often samples fell inside versus outside defined AOIs.

Usage

```
audit_gazepoint_dynamic_aoi_coverage(  
  data,  
  label_col = "aoi_current",  
  definition_time_col = "aoi_definition_time",  
  time_gap_col = "aoi_time_gap",  
  group_cols = NULL,  
  outside_label = "outside",  
  max_time_gap = Inf,  
  x_col = NULL,  
  y_col = NULL  
)
```

Arguments

data	Output from add_gazepoint_dynamic_aoi() .
label_col	AOI-label column.
definition_time_col	Matched-definition timestamp column.
time_gap_col	Definition-time-gap column.
group_cols	Optional summary grouping columns.
outside_label	Outside-AOI label.
max_time_gap	Optional audit threshold for definition-time gaps.
x_col, y_col	Optional coordinate columns used to flag missing gaze.

Value

An object of class "gp3_dynamic_aoi_coverage_audit" containing overview, group, AOI, and flagged-row tables plus settings.

audit_gazepoint_event_sync

Audit Gazepoint event and timing synchronisation

Description

Create a publication-level audit of event timing, trial timing, and event availability in a Gazepoint master table or sample-level export.

Usage

```
audit_gazepoint_event_sync(
  data,
  time_col = "time",
  event_col = NULL,
  group_cols = c("subject", "media_id", "trial_global"),
  condition_col = NULL,
  expected_event_labels = NULL,
  onset_event_label = NULL,
  response_event_label = NULL,
  min_samples_per_unit = 1L,
  max_time_gap_ms = NULL
)
```

Arguments

data	A data frame containing sample-level Gazepoint data.
time_col	Name of the time column.
event_col	Optional event-label column. If NULL, the function tries to detect a common event column.
group_cols	Columns defining a trial or recording unit.
condition_col	Optional condition column.
expected_event_labels	Optional character vector of expected event labels.
onset_event_label	Optional event label identifying trial/stimulus onset.
response_event_label	Optional event label identifying response events.
min_samples_per_unit	Minimum number of samples expected per unit.
max_time_gap_ms	Optional maximum allowed within-unit time gap in milliseconds.

Value

A list with class `gp3_event_sync_audit` containing overview, unit_summary, event_summary, expected_event_summary, flagged_units, and settings tables.

audit_gazepoint_exclusion_flow

Audit Gazepoint exclusion and retention flow

Description

Create a publication-level audit of retained and excluded analysis units.

Usage

```
audit_gazepoint_exclusion_flow(
  data,
  subject_col = "subject",
  condition_col = NULL,
  unit_cols = c("media_id", "trial_global"),
  include_col = NULL,
  exclude_col = NULL,
  status_col = NULL,
  reason_col = NULL,
  included_values = c("included", "include", "kept", "keep", "retained", "ok", "ready",
    "complete", "completed"),
  excluded_values = c("excluded", "exclude", "drop", "dropped", "removed", "fail",
    "failed", "not_ready", "review", "invalid"),
  min_retained_prop = 0.7,
  max_condition_exclusion_ratio = 2
)
```

Arguments

<code>data</code>	A data frame containing row-, trial-, window-, or unit-level data.
<code>subject_col</code>	Subject/participant identifier column.
<code>condition_col</code>	Optional condition column.
<code>unit_cols</code>	Optional columns defining the analysis unit, such as media, trial, block, or window.
<code>include_col</code>	Optional logical/numeric/character column indicating rows or units retained for analysis.
<code>exclude_col</code>	Optional logical/numeric/character column indicating rows or units excluded from analysis.
<code>status_col</code>	Optional status column used to infer inclusion or exclusion.
<code>reason_col</code>	Optional exclusion-reason column.

included_values	Character values in status_col treated as retained.
excluded_values	Character values in status_col treated as excluded.
min_retained_prop	Minimum acceptable retained-unit proportion.
max_condition_exclusion_ratio	Maximum allowed ratio between condition exclusion proportions.

Value

A list with class `gp3_exclusion_flow_audit` containing overview, `unit_flow`, `reason_summary`, `condition_summary`, `subject_summary`, `flagged_units`, and settings tables.

audit_gazepoint_face_quality	<i>Audit external facial-behaviour data quality</i>
------------------------------	---

Description

Audits the quality of standardised external facial-behaviour data imported with `read_gazepoint_face_export()` and standardised with `standardize_gazepoint_face_columns()`. The helper checks face-detection validity, confidence, success, duplicate frame indices, and basic timing continuity. It does not infer facial expressions or emotional states.

Usage

```
audit_gazepoint_face_quality(
  data,
  group_cols = c("participant_id", "face_file"),
  confidence_threshold = 0.8,
  min_valid_percent = 70,
  warning_valid_percent = 85,
  max_time_gap_sec = NULL,
  max_duplicate_frame_percent = 1,
  standardize = TRUE
)
```

Arguments

data	A data frame returned by <code>standardize_gazepoint_face_columns()</code> , a data frame that can be standardised by that function, or a path to a CSV file readable by <code>read_gazepoint_face_export()</code> .
group_cols	Character vector of grouping columns for quality summaries. Columns not present in data are ignored. Use <code>NULL</code> for an overall-only audit.

confidence_threshold	Minimum face-detection confidence used when standardising unstandardised data.
min_valid_percent	Minimum valid-row percentage below which a group is marked as "fail".
warning_valid_percent	Valid-row percentage below which a group is marked as "warn" when it is still above min_valid_percent.
max_time_gap_sec	Optional maximum allowed time gap in seconds. If supplied, groups with larger observed positive gaps are marked as "warn".
max_duplicate_frame_percent	Maximum tolerated percentage of duplicate non-missing frame indices before a group is marked as "warn".
standardize	Should unstandardised data be passed through <code>standardize_gazepoint_face_columns()</code> before auditing?

Value

A list with overview, group_summary, issue_summary, data, and settings. The returned object has class `gp3_face_quality_audit`.

Examples

```
face <- data.frame(
  frame = 1:3,
  timestamp = c(0, 0.033, 0.066),
  confidence = c(0.95, 0.90, 0.40),
  success = c(1, 1, 1),
  AU12_r = c(0.1, 0.2, 0.3)
)

audit_gazepoint_face_quality(face)
```

audit_gazepoint_face_sync

Audit synchronisation between Gazepoint and external facial-behaviour data

Description

Summarises the output of `sync_gazepoint_face_data()` by reporting matched, unmatched, outside-tolerance, and missing-timing/frame rows. For nearest-time synchronisation, it also reports absolute time-difference summaries. The helper audits alignment quality only; it does not infer facial expressions or emotional states.

Usage

```
audit_gazepoint_face_sync(
  data,
  group_cols = NULL,
  min_matched_percent = 70,
  warning_matched_percent = 85,
  max_abs_diff_sec = NULL
)
```

Arguments

<code>data</code>	A data frame returned by <code>sync_gazepoint_face_data()</code> .
<code>group_cols</code>	Optional character vector of grouping columns. Columns not present in data are ignored. Use <code>NULL</code> for an overall-only audit.
<code>min_matched_percent</code>	Minimum percentage of rows that must have <code>face_sync_status == "matched"</code> for a group to pass.
<code>warning_matched_percent</code>	Percentage below which a group is marked as "warn" when still above <code>min_matched_percent</code> .
<code>max_abs_diff_sec</code>	Optional maximum allowed absolute synchronisation difference in seconds. Only evaluated when <code>face_sync_abs_diff_sec</code> is available.

Value

A list with overview, group_summary, issue_summary, data, and settings. The returned object has class `gp3_face_sync_audit`.

Examples

```
gaze <- data.frame(
  subject_id = "P001",
  time_sec = c(0.00, 0.03, 0.07)
)

face <- data.frame(
  participant_id = "P001",
  frame = 1:3,
  timestamp = c(0.00, 0.033, 0.066),
  confidence = c(0.95, 0.94, 0.93),
  success = c(1, 1, 1)
)

synced <- sync_gazepoint_face_data(
  gaze,
  face,
  by = c(subject_id = "participant_id"),
  gaze_time_col = "time_sec"
)
```

```
audit_gazepoint_face_sync(synced)
```

```
audit_gazepoint_fixation_reliability
```

Audit split-half reliability of fixation or AOI metrics

Description

Computes odd-even or random split-half reliability for common fixation and AOI-derived metrics. The audit returns the split-half correlation and the Spearman-Brown corrected reliability estimate.

Usage

```
audit_gazepoint_fixation_reliability(
  data,
  subject_col,
  trial_col,
  metric = c("fixation_count", "mean_fixation_duration", "total_fixation_duration",
    "aoi_dwell_prop", "transition_count", "entropy_score"),
  duration_col = NULL,
  aoi_col = NULL,
  target_aoi = NULL,
  time_col = NULL,
  group_cols = NULL,
  min_trials = 4,
  split_method = c("odd_even", "random"),
  seed = NULL,
  correlation_method = c("pearson", "spearman")
)
```

Arguments

<code>data</code>	A fixation-level or AOI-event-level data frame.
<code>subject_col</code>	Character scalar. Subject identifier column.
<code>trial_col</code>	Character scalar. Trial identifier column.
<code>metric</code>	Metric to audit. Supported values are "fixation_count", "mean_fixation_duration", "total_fixation_duration", "aoi_dwell_prop", "transition_count", and "entropy_score".
<code>duration_col</code>	Optional duration column. Required for duration metrics. Optional for "aoi_dwell_prop"; if omitted, row proportions are used.
<code>aoi_col</code>	Optional AOI column. Required for AOI, transition, and entropy metrics.
<code>target_aoi</code>	Optional AOI label required when <code>metric = "aoi_dwell_prop"</code> .
<code>time_col</code>	Optional time column used to order AOI sequences.

group_cols	Optional grouping columns. Reliability is computed separately within each group.
min_trials	Minimum number of trials per subject required for inclusion.
split_method	"odd_even" or "random".
seed	Optional random seed used when split_method = "random".
correlation_method	Correlation method passed to <code>stats::cor()</code> .

Value

A data frame containing split-half reliability diagnostics.

Examples

```
dat <- expand.grid(
  subject = paste0("S", 1:6),
  trial = paste0("T", 1:4),
  KEEP.OUT.ATTRS = FALSE
)
dat$duration <- rep(seq_len(6), each = 4) + rep(c(0, 0.1, 0, 0.1), 6)

audit_gazepoint_fixation_reliability(
  dat,
  subject_col = "subject",
  trial_col = "trial",
  metric = "total_fixation_duration",
  duration_col = "duration"
)
```

```
audit_gazepoint_gaze_signal_quality
```

Audit Gazepoint gaze-signal quality

Description

Create a publication-level audit of gaze coordinate availability, validity, off-screen samples, and optional pupil availability.

Usage

```
audit_gazepoint_gaze_signal_quality(
  data,
  subject_col = "subject",
  condition_col = NULL,
  group_cols = c("subject", "media_id", "trial_global"),
  x_col = NULL,
  y_col = NULL,
```

```

    validity_cols = NULL,
    pupil_col = NULL,
    screen_x_range = c(0, 1),
    screen_y_range = c(0, 1),
    min_gaze_valid_prop = 0.7,
    max_missing_gaze_prop = 0.3,
    max_offscreen_prop = 0.3,
    min_pupil_valid_prop = 0.7
  )

```

Arguments

<code>data</code>	A sample-level Gazepoint data frame.
<code>subject_col</code>	Subject/participant identifier column.
<code>condition_col</code>	Optional condition column.
<code>group_cols</code>	Columns defining a recording, stimulus, trial, or analysis unit.
<code>x_col</code>	Optional gaze-x coordinate column. If NULL, common Gazepoint aliases are detected.
<code>y_col</code>	Optional gaze-y coordinate column. If NULL, common Gazepoint aliases are detected.
<code>validity_cols</code>	Optional gaze-validity columns. If NULL, common Gazepoint validity columns are detected.
<code>pupil_col</code>	Optional pupil column. If NULL, common pupil aliases are detected.
<code>screen_x_range</code>	Numeric length-2 vector defining plausible on-screen x range.
<code>screen_y_range</code>	Numeric length-2 vector defining plausible on-screen y range.
<code>min_gaze_valid_prop</code>	Minimum acceptable gaze-validity proportion.
<code>max_missing_gaze_prop</code>	Maximum acceptable missing-gaze proportion.
<code>max_offscreen_prop</code>	Maximum acceptable off-screen proportion.
<code>min_pupil_valid_prop</code>	Minimum acceptable valid-pupil proportion when a pupil column is available.

Value

A list with class `gp3_gaze_signal_quality_audit` containing `overview`, `unit_summary`, `subject_summary`, `condition_summary`, `signal_issue_summary`, `flagged_units`, and `settings` tables.

 audit_gazepoint_master

Audit a Gazepoint master sample table

Description

Creates compact quality-audit tables from a master sample-level table created by `as_gazepoint_master()` or `create_gazepoint_master()`. The audit summarises missing gaze, missing pupil, off-screen gaze, AOI states, pupil availability, coordinate ranges, and subject/media-level quality.

Usage

```
audit_gazepoint_master(master)
```

Arguments

master A master sample-level table created by `as_gazepoint_master()` or `create_gazepoint_master()`.

Value

A named list of tibbles:

overview One-row overview of the master table.

by_subject Quality summary by participant/source.

by_media Quality summary by media/stimulus.

by_subject_media Quality summary by participant/source and media/stimulus.

aoi_states Counts and percentages of AOI states.

pupil_summary Pupil summary by participant/source and media/stimulus.

coordinate_summary Coordinate range and off-screen summary.

Examples

```
## Not run:
results <- run_gazepoint_workflow(
  export_dir = "C:/Users/YourName/Desktop/gp3_test_exports",
  output_dir = "C:/Users/YourName/Desktop/gp3_outputs"
)

master <- create_gazepoint_master(
  gaze_data = results$all_gaze,
  screen_width_px = 1920,
  screen_height_px = 1080
)

audit <- audit_gazepoint_master(master)

audit$overview
```

```

audit$by_subject
audit$aoi_states

## End(Not run)

```

```
audit_gazepoint_naming_consistency
```

Audit British and American summary-function spellings

Description

Audit British and American summary-function spellings

Usage

```
audit_gazepoint_naming_consistency(exports = NULL)
```

Arguments

`exports` Optional character vector of exported function names. When NULL, current gp3tools namespace exports are inspected.

Value

A "gazepoint_naming_audit" object containing paired names and an overall status.

```
audit_gazepoint_post_exclusion_balance
```

Audit post-exclusion condition balance

Description

Create a publication-level audit of whether the retained analysis sample remains balanced across subjects and experimental conditions after exclusions.

Usage

```

audit_gazepoint_post_exclusion_balance(
  data,
  subject_col = "subject",
  condition_col = "condition",
  unit_cols = c("media_id", "trial_global"),
  retained_col = NULL,
  include_col = NULL,
  exclude_col = NULL,

```

```

status_col = NULL,
expected_conditions = NULL,
included_values = c("included", "include", "kept", "keep", "retained", "ok", "ready",
  "complete", "completed"),
excluded_values = c("excluded", "exclude", "drop", "dropped", "removed", "fail",
  "failed", "not_ready", "review", "invalid"),
min_retained_units_per_condition = 1L,
min_retained_units_per_subject_condition = 1L,
max_condition_count_ratio = 2,
max_subject_condition_ratio = 2,
require_all_conditions_per_subject = TRUE
)

```

Arguments

<code>data</code>	A data frame containing row-, sample-, trial-, or unit-level data.
<code>subject_col</code>	Subject/participant identifier column.
<code>condition_col</code>	Experimental condition column.
<code>unit_cols</code>	Optional columns defining the analysis unit, such as media, trial, block, or window.
<code>retained_col</code>	Optional logical/numeric/character column indicating retained units.
<code>include_col</code>	Optional logical/numeric/character inclusion column.
<code>exclude_col</code>	Optional logical/numeric/character exclusion column.
<code>status_col</code>	Optional status column used to infer retained/excluded units.
<code>expected_conditions</code>	Optional character vector of expected conditions.
<code>included_values</code>	Character values in <code>status_col</code> treated as retained.
<code>excluded_values</code>	Character values in <code>status_col</code> treated as excluded.
<code>min_retained_units_per_condition</code>	Minimum retained units required per condition.
<code>min_retained_units_per_subject_condition</code>	Minimum retained units required per subject-condition cell.
<code>max_condition_count_ratio</code>	Maximum allowed ratio between condition-level retained counts.
<code>max_subject_condition_ratio</code>	Maximum allowed within-subject retained condition-count ratio.
<code>require_all_conditions_per_subject</code>	Logical. If TRUE, flag subjects missing retained units in one or more expected conditions.

Value

A list with class `gp3_post_exclusion_balance_audit` containing `overview`, `unit_flow`, `cell_summary`, `condition_summary`, `subject_summary`, `flagged_cells`, `flagged_subjects`, and `settings` tables.

audit_gazepoint_pupil_baseline

Audit Gazepoint pupil baseline quality

Description

Summarise baseline quality after `baseline_correct_gazepoint_pupil()`.

Usage

```
audit_gazepoint_pupil_baseline(
  data,
  group_cols = c("subject", "media_id"),
  time_col = "time",
  pupil_col = "pupil_interpolated",
  baseline_n_col = "pupil_baseline_n",
  baseline_status_col = "pupil_baseline_status",
  baseline_available_col = "pupil_baseline_available",
  baseline_used_col = "pupil_baseline_used",
  baseline_window_start_col = "pupil_baseline_window_start",
  baseline_window_end_col = "pupil_baseline_window_end",
  baseline_flag_col = NULL,
  interpolated_col = "pupil_was_interpolated",
  artifact_col = NULL,
  artifact_reason_col = NULL,
  min_baseline_samples = 1,
  max_missing_pct = 50,
  max_interpolated_pct = 50,
  max_artifact_pct = 50
)
```

Arguments

<code>data</code>	A data frame returned by <code>baseline_correct_gazepoint_pupil()</code> or a later pupil-preprocessing step.
<code>group_cols</code>	Character vector of grouping columns. Use <code>character(0)</code> for an overall audit.
<code>time_col</code>	Name of the time column.
<code>pupil_col</code>	Name of the pupil column used to evaluate missingness.
<code>baseline_n_col</code>	Name of the baseline valid-sample count column.
<code>baseline_status_col</code>	Name of the baseline-status column.
<code>baseline_available_col</code>	Name of the baseline-availability column.
<code>baseline_used_col</code>	Name of the logical column indicating whether a row used a baseline value.

baseline_window_start_col	Name of the baseline-window start column.
baseline_window_end_col	Name of the baseline-window end column.
baseline_flag_col	Optional logical column identifying baseline rows. If NULL, baseline rows are detected from the time column and baseline window start/end columns.
interpolated_col	Name of the logical interpolation flag column.
artifact_col	Optional artifact flag column. If NULL, the function tries to detect pupil_artifact_flag, pupil_flag_invalid, or artifact_flag.
artifact_reason_col	Optional artifact-reason column. If NULL, the function tries to detect pupil_artifact_reason, pupil_flag_reason, or artifact_reason.
min_baseline_samples	Minimum acceptable number of valid baseline samples before a group is flagged as low quality.
max_missing_pct	Maximum acceptable percentage of missing baseline samples.
max_interpolated_pct	Maximum acceptable percentage of interpolated baseline samples.
max_artifact_pct	Maximum acceptable percentage of artifact-flagged baseline samples.

Details

The function reports baseline-row counts, valid/missing baseline samples, interpolated baseline samples, artifact-flagged baseline samples, no-baseline cases, and low-quality baseline flags by subject, media, trial, condition, or any selected grouping variables.

Value

A tibble with one row per group.

audit_gazepoint_pupil_drift
<i>Audit Gazepoint pupil drift</i>

Description

Summarise tonic pupil/time-on-task drift in processed Gazepoint pupil data.

Usage

```
audit_gazepoint_pupil_drift(
  data,
  group_cols = "subject",
  pupil_col = NULL,
  time_col = "time",
  order_col = "trial",
  condition_col = "condition",
  exclude_col = "excluded_trial",
  include_excluded = FALSE,
  min_valid_samples = 3,
  max_abs_slope_per_min = 1,
  max_condition_time_mean_diff_ms = 1000,
  max_condition_order_mean_diff = 1
)
```

Arguments

<code>data</code>	A data frame from a Gazepoint pupil preprocessing pipeline.
<code>group_cols</code>	Character vector of grouping columns for the main drift audit. The default is "subject".
<code>pupil_col</code>	Name of the pupil column to analyse. If NULL, the function automatically tries <code>pupil_smoothed</code> , <code>pupil_baseline_corrected</code> , <code>pupil_interpolated</code> , <code>pupil_clean</code> , and <code>pupil</code> .
<code>time_col</code>	Name of the within-trial or sample-time column.
<code>order_col</code>	Optional trial/order column used to assess time-on-task imbalance. If NULL, order-based summaries are skipped.
<code>condition_col</code>	Optional condition column used to summarise condition drift and time-on-task imbalance. If NULL, condition summaries are skipped.
<code>exclude_col</code>	Optional logical exclusion column. If present and <code>include_excluded = FALSE</code> , excluded rows are removed before analysis.
<code>include_excluded</code>	Logical. If FALSE, rows marked by <code>exclude_col</code> are excluded when that column exists.
<code>min_valid_samples</code>	Minimum valid pupil samples required to estimate a drift slope.
<code>max_abs_slope_per_min</code>	Maximum acceptable absolute pupil slope per minute before a drift warning is raised.
<code>max_condition_time_mean_diff_ms</code>	Maximum acceptable difference in mean sample time across conditions.
<code>max_condition_order_mean_diff</code>	Maximum acceptable difference in mean trial/order value across conditions.

Details

The function estimates simple linear pupil trends over time within selected grouping variables, usually subjects. It also reports subject-level drift, condition-level drift, and possible condition imbalance in time-on-task.

Value

A named list containing `by_group`, `by_subject`, `by_condition`, `condition_balance`, and summary tibbles.

`audit_gazepoint_pupil_gaps`

Audit Gazepoint pupil interpolation gaps

Description

Summarise observed, interpolated, and remaining missing pupil samples, together with gap-level counts and gap duration/sample-size summaries.

Usage

```
audit_gazepoint_pupil_gaps(
  data,
  group_cols = c("subject", "media_id"),
  status_col = "pupil_interpolation_status",
  gap_id_col = "pupil_gap_id",
  gap_n_samples_col = "pupil_gap_n_samples",
  gap_duration_col = "pupil_gap_duration_ms",
  interpolated_col = "pupil_was_interpolated",
  pupil_col = "pupil_interpolated"
)
```

Arguments

<code>data</code>	A data frame containing pupil interpolation status columns.
<code>group_cols</code>	Character vector of grouping columns. Use <code>character(0)</code> for an overall audit.
<code>status_col</code>	Name of the interpolation status column.
<code>gap_id_col</code>	Name of the gap identifier column.
<code>gap_n_samples_col</code>	Name of the column containing gap size in samples.
<code>gap_duration_col</code>	Name of the column containing gap duration in ms.
<code>interpolated_col</code>	Name of the logical column indicating whether a sample was interpolated.
<code>pupil_col</code>	Name of the pupil column after interpolation.

Details

This function is intended for data returned by `interpolate_gazepoint_pupil()`, but it can also be used with any table that contains compatible interpolation-status and gap columns.

Value

A tibble with one row per group, or one row overall when `group_cols = character(0)`.

audit_gazepoint_pupil_imbalance
<i>Audit Gazepoint pupil preprocessing imbalance</i>

Description

Check whether pupil preprocessing loss differs across experimental conditions or other grouping variables.

Usage

```
audit_gazepoint_pupil_imbalance(
  data,
  group_cols = "condition",
  pupil_col = "pupil_interpolated",
  interpolated_col = "pupil_was_interpolated",
  interpolation_status_col = "pupil_interpolation_status",
  artifact_col = NULL,
  artifact_reason_col = NULL,
  min_group_n = 1,
  max_valid_pct_diff = 10,
  max_artifact_pct_diff = 10,
  max_missing_pct_diff = 10,
  max_interpolated_pct_diff = 10
)
```

Arguments

<code>data</code>	A data frame from a pupil preprocessing pipeline.
<code>group_cols</code>	Character vector of grouping columns. By default, summaries are produced by condition.
<code>pupil_col</code>	Name of the post-preprocessing pupil column used to define remaining valid and missing samples.
<code>interpolated_col</code>	Name of the logical interpolation flag column.
<code>interpolation_status_col</code>	Name of the interpolation-status column.

artifact_col	Optional artifact flag column. If NULL, the function tries to detect pupil_artifact_flag, pupil_flag_invalid, or artifact_flag.
artifact_reason_col	Optional artifact-reason column. If NULL, the function tries to detect pupil_artifact_reason, pupil_flag_reason, or artifact_reason.
min_group_n	Minimum group size below which a group is flagged.
max_valid_pct_diff	Maximum acceptable range in valid-sample percentage across groups.
max_artifact_pct_diff	Maximum acceptable range in artifact percentage across groups.
max_missing_pct_diff	Maximum acceptable range in remaining-missing percentage across groups.
max_interpolated_pct_diff	Maximum acceptable range in interpolated percentage across groups.

Details

The function summarises valid pupil samples, interpolated samples, artifact-flagged samples, and remaining missing samples. It also adds simple imbalance flags based on differences between groups.

Value

A tibble with one row per group and imbalance-warning columns.

audit_gazepoint_pupil_overlap_risk
<i>Audit Gazepoint pupil-response overlap risk</i>

Description

Check whether event-related pupil-response windows may overlap.

Usage

```
audit_gazepoint_pupil_overlap_risk(
  data,
  group_cols = "subject",
  trial_col = "trial_global",
  time_col = "time",
  event_time_cols = c("stimulus_onset_time", "target_onset_time", "response_time"),
  window_start_ms = 0,
  window_end_ms = 2000,
  min_event_gap_ms = 1000,
  exclude_col = "excluded_trial",
  include_excluded = FALSE
)
```

Arguments

data	A Gazepoint sample-level data frame.
group_cols	Character vector of grouping columns, usually "subject".
trial_col	Name of the trial identifier column.
time_col	Name of the within-trial time column.
event_time_cols	Character vector of event-time columns, in ms.
window_start_ms	Response-window start relative to each event, in ms.
window_end_ms	Response-window end relative to each event, in ms.
min_event_gap_ms	Minimum acceptable event-to-event gap in ms.
exclude_col	Optional logical exclusion column.
include_excluded	Logical. If FALSE, rows marked by exclude_col are removed before the audit when that column exists.

Details

This function is designed as a deconvolution/readiness gate. It checks whether events within the same trial are too close together and whether their response windows overlap. If no usable event-time values are found, the function returns a clean audit with `overlap_assessment_status = "no_usable_event_times"`.

Value

A named list containing events, event_gaps, by_trial, and summary tibbles.

audit_gazepoint_pupil_reliability

Audit split-half reliability for Gazepoint pupil outcomes

Description

Create a split-half reliability audit for trial-level or window-level pupil outcomes. The helper is intended for publication-readiness checks when pupil features are interpreted as stable participant-level outcomes or individual difference measures.

Usage

```
audit_gazepoint_pupil_reliability(
  data,
  outcome_cols = NULL,
  participant_col = NULL,
  trial_col = NULL,
  split_col = NULL,
  by_cols = NULL,
  split_method = c("odd_even", "first_second"),
  aggregate_function = c("mean", "median"),
  correlation_method = c("pearson", "spearman"),
  min_trials_per_split = 2,
  name = "gazepoint_pupil_reliability"
)
```

Arguments

<code>data</code>	A data frame containing trial-level or window-level pupil outcomes.
<code>outcome_cols</code>	Character vector of pupil outcome columns. If NULL, common pupil outcome columns are detected automatically.
<code>participant_col</code>	Participant/subject column. If NULL, common participant columns are detected automatically.
<code>trial_col</code>	Trial/order column. If NULL, common trial columns are detected automatically when available. If no trial column is available, row order within participant is used.
<code>split_col</code>	Optional pre-existing split column. If supplied, it must have exactly two non-missing levels.
<code>by_cols</code>	Optional grouping columns for separate reliability audits, such as "condition" or "window".
<code>split_method</code>	Split method used when <code>split_col = NULL</code> . Options are "odd_even" and "first_second".
<code>aggregate_function</code>	Function used to aggregate trial-level values within participant and split. Options are "mean" and "median".
<code>correlation_method</code>	Correlation method for split-half association. Options are "pearson" and "spearman".
<code>min_trials_per_split</code>	Minimum number of non-missing outcome values required in each split for a participant to contribute to the reliability estimate.
<code>name</code>	Character label stored in the audit object.

Value

A list with class `gp3_pupil_reliability_audit`.

audit_gazepoint_screen_bounds

Audit Gazepoint gaze coordinates against screen bounds

Description

Checks whether gaze coordinates are missing, equal to (0, 0), or outside the expected screen/stimulus bounds. The helper is intended for transparent quality-control reporting, not for automatic exclusion decisions.

Usage

```
audit_gazepoint_screen_bounds(
  data,
  x_col,
  y_col,
  screen_width,
  screen_height,
  group_cols = NULL,
  margin = 0,
  treat_zero_zero_as_out_of_bounds = TRUE
)
```

Arguments

data	A data frame.
x_col, y_col	Character names of gaze-coordinate columns.
screen_width, screen_height	Numeric screen or stimulus dimensions.
group_cols	Optional grouping columns for group-level summaries.
margin	Numeric tolerance around the screen bounds. A positive value allows coordinates slightly outside the nominal screen area.
treat_zero_zero_as_out_of_bounds	If TRUE, (0, 0) coordinates are flagged separately and counted as invalid.

Value

A list with row-level flags, group-level summary, overall summary, and settings.

Examples

```
x <- simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)
audit_gazepoint_screen_bounds(x, "gaze_x", "gaze_y", 1920, 1080)
```

audit_gazepoint_stimulus_luminance

Audit stimulus luminance and brightness for Gazepoint studies

Description

Compute stimulus-level luminance, brightness, and contrast summaries for image stimuli used in Gazepoint eye-tracking and pupillometry studies. This helper is intended as a publication-readiness audit because pupil size is strongly affected by stimulus brightness.

Usage

```
audit_gazepoint_stimulus_luminance(
  data,
  stimulus_file_col = NULL,
  stimulus_id_col = NULL,
  condition_col = NULL,
  image_dir = NULL,
  recursive = TRUE,
  name = "gazepoint_stimulus_luminance"
)
```

Arguments

data	A data frame containing at least a stimulus image/file column.
stimulus_file_col	Name of the stimulus image/file column. If NULL, common file-column names are detected automatically.
stimulus_id_col	Optional stimulus identifier column. If NULL, common stimulus/media/item identifier columns are detected automatically.
condition_col	Optional experimental condition column. If NULL, common condition columns are detected automatically. If no condition column exists, all rows are assigned to "all_data".
image_dir	Optional directory prepended to relative stimulus paths.
recursive	If TRUE, unresolved relative file names are searched for recursively under image_dir.
name	Character label stored in the audit object.

Value

A list with class `gp3_stimulus_luminance_audit`.

audit_gazepoint_timecourse_grid

Audit a Gazepoint time-course grid for cluster-permutation readiness

Description

Check whether a prepared or raw long-format time-course data set has the subject-by-condition-by-time structure expected by the conservative two-condition cluster-permutation workflow.

Usage

```
audit_gazepoint_timecourse_grid(
  data,
  subject_col = ".gp3_cluster_subject",
  condition_col = ".gp3_cluster_condition",
  time_col = ".gp3_cluster_time_bin",
  outcome_col = ".gp3_cluster_outcome"
)
```

Arguments

data	A data frame.
subject_col	Subject column. Defaults to the internal prepared column.
condition_col	Condition column. Defaults to the internal prepared column.
time_col	Time-bin column. Defaults to the internal prepared column.
outcome_col	Outcome column. Defaults to the internal prepared column.

Value

A list containing grid counts, missing cells, duplicate cells, and readiness flags.

baseline_correct_gazepoint_pupil

Baseline-correct Gazepoint pupil data

Description

Computes baseline-corrected pupil columns from a Gazepoint pupil time series, typically after [flag_gazepoint_pupil\(\)](#) and [interpolate_gazepoint_pupil\(\)](#). Baselines can be defined either by a time window, such as `c(-200, 0)`, or by a user-supplied logical baseline/pre-stimulus flag column.

Usage

```
baseline_correct_gazepoint_pupil(
  data,
  pupil_col = NULL,
  time_col = NULL,
  baseline_time_col = NULL,
  baseline_window = c(-200, 0),
  baseline_flag_col = NULL,
  group_cols = c("subject", "media_id"),
  baseline_method = c("mean", "median"),
  min_baseline_samples = 1
)
```

Arguments

data	A Gazepoint master table, preferably after <code>interpolate_gazepoint_pupil()</code> .
pupil_col	Optional name of the pupil column to baseline-correct. If NULL, the function detects one of <code>pupil_interpolated</code> , <code>pupil_for_preprocessing</code> , <code>mean_pupil</code> , <code>pupil</code> , <code>pupil_raw</code> , <code>left_pupil</code> , or <code>right_pupil</code> .
time_col	Optional name of the main time column. If NULL, the function detects one of <code>time_ms</code> , <code>time</code> , <code>time_orig</code> , or <code>time_orig_ms</code> .
baseline_time_col	Optional name of the time column used for selecting baseline samples. If NULL, the function detects relative-time columns first, then falls back to <code>time_col</code> .
baseline_window	Numeric vector of length two giving the baseline window in milliseconds. Defaults to <code>c(-200, 0)</code> . This can also be set to post-onset or early-window values such as <code>c(0, 200)</code> when no pre-stimulus period is available.
baseline_flag_col	Optional logical column identifying baseline samples. If supplied, this takes priority over <code>baseline_window</code> .
group_cols	Character vector of grouping columns used to compute one baseline per independent time series. Defaults to <code>c("subject", "media_id")</code> . Columns such as <code>"trial"</code> or <code>"trial_global"</code> can be added when available. Use <code>character(0)</code> for one global baseline.
baseline_method	Baseline statistic. One of <code>"mean"</code> or <code>"median"</code> . Defaults to <code>"mean"</code> .
min_baseline_samples	Minimum number of valid baseline samples required to compute a baseline. Defaults to 1.

Value

A tibble containing the original data plus baseline-correction columns.

Examples

```
## Not run:
flagged <- flag_gazepoint_pupil(master)
interpolated <- interpolate_gazepoint_pupil(flagged)

corrected <- baseline_correct_gazepoint_pupil(
  interpolated,
  baseline_window = c(-200, 0)
)

corrected <- baseline_correct_gazepoint_pupil(
  interpolated,
  baseline_window = c(0, 200)
)

## End(Not run)
```

benchmark_gazepoint_event_detectors

Benchmark Gazepoint event detectors against reviewed events

Description

Compare standardized fixation intervals from one or more detectors with a manually reviewed or synthetic reference-event table. Matching is one-to-one within each participant/trial sequence and is based on interval intersection-over-union. Results quantify methodological agreement with the supplied reference annotations; they do not establish a universally correct detector.

Usage

```
benchmark_gazepoint_event_detectors(
  x,
  reviewed_events,
  sequence_cols = NULL,
  reviewed_start_col = "start_time",
  reviewed_end_col = "end_time",
  reviewed_id_col = "review_event_id",
  review_status_col = "review_status",
  accepted_status = c("accepted", "include", "reviewed", "confirmed"),
  min_overlap = 0.5,
  time_unit = c("seconds", "milliseconds")
)
```

Arguments

x	An object returned by <code>compare_gazepoint_event_detectors()</code> or a standardized detector-event data frame.
---	---

reviewed_events	A data frame containing reviewed reference intervals.
sequence_cols	Sequence identifier columns. When <code>x</code> is a comparison object, its stored sequence columns are used by default.
reviewed_start_col, reviewed_end_col	Start and end columns in <code>reviewed_events</code> .
reviewed_id_col	Optional reviewed-event identifier column.
review_status_col	Optional review-status column. When present, only rows whose status is included in <code>accepted_status</code> are used.
accepted_status	Character values treated as accepted reviews.
min_overlap	Minimum interval intersection-over-union required for a true-positive match.
time_unit	Unit used by event start/end values. This controls conversion of timing errors to milliseconds.

Value

An object of class `"gp3_event_detector_benchmark"` containing detector-level metrics, sequence-level metrics, one-to-one matches, unmatched-event diagnostics, accepted reviewed events, detector events, detector-run information, and settings.

Examples

```

reviewed <- data.frame(
  USER_ID = "P01",
  trial = "T01",
  review_event_id = 1:2,
  start_time = c(0, 2),
  end_time = c(1, 3)
)

detected <- data.frame(
  USER_ID = "P01",
  trial = "T01",
  detector = "velocity_10",
  family = "velocity",
  threshold = 10,
  event_id = 1:2,
  start_time = c(0.05, 2.05),
  end_time = c(1.05, 3.05),
  duration_ms = 1000
)

benchmark <- benchmark_gazepoint_event_detectors(
  detected,
  reviewed,
  sequence_cols = c("USER_ID", "trial")
)

```

```
benchmark$detector_metrics
```

```
benchmark_gazepoint_export_performance
```

Benchmark gp3tools on increasingly large Gazepoint exports

Description

Generates deterministic Gazepoint-like all-gaze exports and benchmarks selected package operations across increasing row and file counts. Ordinary unit tests should use small scales. The script installed under `inst/benchmarks/` runs the large-export profile.

Usage

```
benchmark_gazepoint_export_performance(
  scales = data.frame(total_rows = c(10000L, 50000L, 200000L), n_files = c(1L, 4L, 8L)),
  operations = c("generate", "import", "master", "sampling", "quality"),
  trials = 3L,
  seed = 20260717L,
  limits = gp3tools_performance_limits(),
  stop_on_regression = FALSE,
  output_dir = NULL,
  keep_exports = FALSE,
  on_error = c("record", "stop")
)
```

Arguments

<code>scales</code>	Data frame with integer <code>total_rows</code> and <code>n_files</code> columns.
<code>operations</code>	Any of "generate", "import", "master", "sampling", and "quality".
<code>trials</code>	Number of repetitions per scale.
<code>seed</code>	Integer random seed.
<code>limits</code>	Performance limits returned by gp3tools_performance_limits() or a compatible data frame.
<code>stop_on_regression</code>	Stop when a completed benchmark exceeds a limit.
<code>output_dir</code>	Optional directory used for generated exports.
<code>keep_exports</code>	Retain generated CSV exports.
<code>on_error</code>	Whether operation errors are recorded or stop the benchmark.

Value

A "gazepoint_performance_benchmark" object containing trial-level measurements, aggregated summaries, regression checks, settings, and session metadata.

bootstrap_gazepoint_scanpath_clusters

Bootstrap scanpath-cluster stability

Description

Evaluate the stability of scanpath clustering by repeatedly subsampling scanpaths, refitting the clustering solution, and recording co-clustering, adjusted Rand agreement, and representative-scanpath selection.

Usage

```
bootstrap_gazepoint_scanpath_clusters(
  x,
  k = 3L,
  n_boot = 200L,
  sample_fraction = 0.8,
  method = c("hierarchical", "pam"),
  linkages = "average",
  seed = NULL,
  aoi_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  distance_col = "normalized_distance",
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  max_sequences = 200
)
```

Arguments

x	Long-format AOI data, pairwise distance data, a square numeric distance matrix, or a "dist" object.
k	Number of clusters.
n_boot	Number of subsampling iterations per specification.
sample_fraction	Proportion of scanpaths retained in each iteration. The resolved sample size is always at least $k + 1$.
method	Clustering method: "hierarchical" or "pam".
linkages	Hierarchical linkage methods to compare. Ignored for PAM.
seed	Optional integer seed. The caller's random-number state is restored on exit.
aoi_col	AOI column when x is long-format AOI data.
group_cols	Columns identifying independent scanpaths.

time_col	Optional ordering column.
distance_col	Distance column for pairwise-distance data.
include_missing	Should missing AOI labels be retained?
missing_label	Label used for retained missing AOIs.
collapse_repeats	Should consecutive repeated AOIs be collapsed?
max_sequences	Maximum number of scanpaths permitted when pairwise distances must be calculated.

Details

The routine reuses the distance formats accepted by `cluster_gazepoint_scanpaths()`. Hierarchical solutions can be compared across multiple linkage methods. PAM remains optional through `cluster`.

Value

An object of class "gp3_scanpath_cluster_bootstrap" containing full-data reference fits, co-clustering and pair-coverage matrices, iteration-level adjusted Rand results, representative stability, the reusable distance object, and resolved settings.

Examples

```
latent <- rep(1:3, each = 2)
d <- outer(
  latent,
  latent,
  FUN = function(x, y) ifelse(x == y, 0.1, 1)
)
diag(d) <- 0
dimnames(d) <- list(LETTERS[1:6], LETTERS[1:6])

stability <- bootstrap_gazepoint_scanpath_clusters(
  d,
  k = 3,
  n_boot = 10,
  seed = 1
)

stability$iteration_summary
```

bootstrap_gazepoint_timecourse

Bootstrap time-course summaries

Description

Compute simple nonparametric bootstrap confidence intervals for a time-varying gaze or pupil measure. The function is intentionally lightweight and returns a tidy data frame that can be plotted or used as a descriptive robustness check alongside model-based analyses.

Usage

```
bootstrap_gazepoint_timecourse(
  data,
  time_col,
  value_col,
  group_col = NULL,
  subject_col = NULL,
  n_boot = 1000,
  ci = 0.95,
  statistic = c("mean", "median"),
  difference_groups = NULL,
  seed = NULL
)
```

Arguments

data	A data frame.
time_col	Name of the time column.
value_col	Name of the numeric outcome column.
group_col	Optional grouping column, for example condition.
subject_col	Optional subject/participant column. If supplied, bootstrap resampling is performed over subjects within each time and group cell; otherwise rows are re-sampled.
n_boot	Number of bootstrap draws.
ci	Confidence level for the interval.
statistic	Summary statistic, either mean or median.
difference_groups	Optional character vector of length two. If supplied with group_col, the function also returns a bootstrap difference curve for group 1 minus group 2.
seed	Optional random seed.

Value

A data frame with time, group/contrast, estimate, lower and upper bootstrap interval limits, sample size, and status columns.

`check_gazepoint_bayesian_readiness`*Check readiness of a Gazepoint-derived dataset for Bayesian or advanced models*

Description

Performs lightweight structural checks before Bayesian, GAMM, HDDM, or other advanced modelling workflows. The function does not fit any model.

Usage

```
check_gazepoint_bayesian_readiness(  
  data,  
  outcome,  
  subject,  
  trial = NULL,  
  time = NULL,  
  condition = NULL,  
  metric_type = "continuous",  
  baseline_window = NULL,  
  min_observations_per_subject = 10,  
  max_missing_trial_prop = 0.2  
)
```

Arguments

<code>data</code>	A data frame.
<code>outcome</code>	Name of the outcome column.
<code>subject</code>	Name of the subject/participant column.
<code>trial</code>	Optional trial column.
<code>time</code>	Optional time column.
<code>condition</code>	Optional condition column.
<code>metric_type</code>	Character scalar describing the planned metric/model type.
<code>baseline_window</code>	Optional numeric vector of length two.
<code>min_observations_per_subject</code>	Minimum number of observations expected per subject.
<code>max_missing_trial_prop</code>	Maximum acceptable missingness proportion per subject-trial cell before a warning is raised.

Value

A data frame of checks, status values, and messages.

```
check_gazepoint_file_pairs
```

Check Gazepoint all-gaze and fixation file pairs

Description

Checks whether each Gazepoint participant/source has both an all-gaze CSV file and a fixation CSV file before running the full workflow.

Usage

```
check_gazepoint_file_pairs(
  folder,
  all_gaze_pattern = "_all_gaze\\.csv$",
  fixation_pattern = "_fixations\\.csv$",
  recursive = FALSE
)
```

Arguments

folder	Folder containing Gazepoint CSV export files.
all_gaze_pattern	Regular expression for selecting all-gaze files.
fixation_pattern	Regular expression for selecting fixation files.
recursive	Logical. If TRUE, search subfolders recursively.

Value

A tibble with one row per detected participant/source and file-pair status.

```
check_gazepoint_model_convergence
```

Check model convergence

Description

Check convergence status for fitted models used in gp3tools workflows.

Usage

```
check_gazepoint_model_convergence(model, model_name = NULL)
```

Arguments

model	A fitted model object, or a gp3tools fit object containing a \$model element.
model_name	Optional model label used in the returned table.

Details

This helper supports lme4 mixed models, mgcv GAM/GAMM objects, glm objects, and lm objects where convergence is meaningful. It returns a compact diagnostic table instead of printing model-specific messages.

Value

A tibble with model class, convergence status, diagnostic status, and message.

check_gazepoint_model_overdispersion
<i>Check model overdispersion</i>

Description

Compute a Pearson-residual overdispersion diagnostic for models where this is meaningful, especially binomial and count models.

Usage

```
check_gazepoint_model_overdispersion(
  model,
  ratio_threshold = 1.2,
  model_name = NULL
)
```

Arguments

model	A fitted model object, or a gp3tools fit object containing a \$model element.
ratio_threshold	Numeric threshold above which the model is flagged as overdispersed.
model_name	Optional model label used in the returned table.

Details

This helper supports glm, lme4 GLMMs, and mgcv GAM/GAMM objects when their family is binomial, quasibinomial, Poisson, quasipoisson, or negative-binomial-like. Gaussian lm, lmer, and Gaussian GAM models return a structured not_applicable diagnostic row.

Value

A tibble with Pearson chi-square, residual degrees of freedom, dispersion ratio, overdispersion flag, diagnostic status, and message.

```
check_gazepoint_model_singularity
  Check model singularity
```

Description

Check whether a fitted mixed model has a singular random-effects structure.

Usage

```
check_gazepoint_model_singularity(model, tolerance = 1e-04, model_name = NULL)
```

Arguments

<code>model</code>	A fitted model object, or a <code>gp3tools</code> fit object containing a <code>\$model</code> element.
<code>tolerance</code>	Numeric tolerance passed to <code>lme4::isSingular()</code> .
<code>model_name</code>	Optional model label used in the returned table.

Details

This helper is primarily intended for `lme4` mixed models. For model classes where singularity is not meaningful, such as `lm`, `glm`, and `mgcv` GAM objects, it returns a structured `not_applicable` diagnostic row.

Value

A tibble with model class, singular-fit status, diagnostic status, and message.

```
check_gazepoint_performance_regression
  Check gp3tools performance results against regression limits
```

Description

Applies explicit absolute limits and, optionally, compares matched rows with a saved baseline. Scaling exponents are estimated from median elapsed times across row-count levels.

Usage

```
check_gazepoint_performance_regression(
  x,
  limits = gp3tools_performance_limits(),
  baseline = NULL,
  elapsed_ratio_limit = 1.5,
  memory_ratio_limit = 1.5
)
```

Arguments

x	A benchmark object or summary data frame.
limits	Explicit operation-level limits.
baseline	Optional prior benchmark object or summary data frame.
elapsed_ratio_limit	Maximum allowed elapsed-time ratio to baseline.
memory_ratio_limit	Maximum allowed heap-growth ratio to baseline.

Value

A "gazepoint_performance_regression" object.

check_gazepoint_real_data_readiness

Check real-data readiness before Gazepoint analysis

Description

Create an explicit final readiness gate for real Gazepoint or gp3tools master data before confirmatory analysis. The gate checks required identifiers, trial/time structure, analysis-specific columns, missingness, basic design balance, and optional upstream audit objects.

Usage

```
check_gazepoint_real_data_readiness(
  data,
  analysis_type = c("general", "pupil", "aoi", "combined"),
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  condition_col = NULL,
  stimulus_col = NULL,
  aoi_col = NULL,
  pupil_col = NULL,
  gaze_x_col = NULL,
  gaze_y_col = NULL,
  tracking_valid_col = NULL,
  required_cols = NULL,
  audit_objects = NULL,
  min_rows = 1L,
  min_participants = 1L,
  min_trials = 1L,
  max_missing_pupil_prop = 0.4,
  max_missing_gaze_prop = 0.4,
```

```

    max_condition_imbalance_ratio = 3,
    name = "gazepoint_real_data_readiness_gate"
)

```

Arguments

<code>data</code>	A Gazepoint or gp3tools data frame.
<code>analysis_type</code>	Analysis target. Options are "general", "pupil", "aoi", and "combined".
<code>participant_col</code>	Optional participant column. If NULL, common names are detected.
<code>trial_col</code>	Optional trial column. If NULL, common names are detected.
<code>time_col</code>	Optional time column. If NULL, common names are detected.
<code>condition_col</code>	Optional condition/group column. If NULL, common names are detected when present.
<code>stimulus_col</code>	Optional stimulus/media column. If NULL, common names are detected when present.
<code>aoi_col</code>	Optional AOI/state column. If NULL, common names are detected when present.
<code>pupil_col</code>	Optional pupil column. If NULL, common names are detected when present.
<code>gaze_x_col</code>	Optional horizontal gaze coordinate column.
<code>gaze_y_col</code>	Optional vertical gaze coordinate column.
<code>tracking_valid_col</code>	Optional tracking-validity column.
<code>required_cols</code>	Additional required columns.
<code>audit_objects</code>	Optional list of upstream audit/validation objects.
<code>min_rows</code>	Minimum acceptable number of rows.
<code>min_participants</code>	Minimum acceptable number of participants.
<code>min_trials</code>	Minimum acceptable number of participant-trial units.
<code>max_missing_pupil_prop</code>	Maximum acceptable pupil missingness proportion when pupil data are required.
<code>max_missing_gaze_prop</code>	Maximum acceptable gaze-coordinate missingness proportion when gaze coordinate columns are present.
<code>max_condition_imbalance_ratio</code>	Warning threshold for condition imbalance.
<code>name</code>	Character label stored in the returned object.

Details

This helper is a final decision wrapper. It complements, but does not replace, `validate_gazepoint_master()` or `create_gazepoint_analysis_decision_audit()`.

Value

A list with class `gp3_real_data_readiness_gate`.

check_sampling_rate	<i>Check sampling rate by group</i>
---------------------	-------------------------------------

Description

Check sampling rate by group

Usage

```
check_sampling_rate(data, group_cols = "MEDIA_ID", time_col = "TIME")
```

Arguments

data	A Gazeport all-gaze data frame.
group_cols	Character vector of grouping columns, usually MEDIA_ID or c("participant_id", "MEDIA_ID").
time_col	Name of elapsed-time column.

Value

A tibble with sample interval and estimated Hz.

classify_gazeport_events_hmm	<i>Classify gaze events with a lightweight unsupervised HMM</i>
------------------------------	---

Description

Estimates gaze velocity, initializes hidden states by k-means, estimates a Gaussian-emission HMM, and decodes the most likely sequence with Viterbi. This is a lightweight package-internal HMM classifier and not a replacement for validated laboratory event-detection software.

Usage

```
classify_gazeport_events_hmm(
  data,
  x,
  y,
  time,
  subject = NULL,
  n_states = 3,
  state_labels = NULL
)
```

Arguments

data	A data frame.
x	X-coordinate column.
y	Y-coordinate column.
time	Time column.
subject	Optional subject column for within-subject sequences.
n_states	Number of hidden states.
state_labels	Optional labels for states. If NULL, states are ordered by increasing mean velocity.

Value

The input data with velocity, hmm_state, and hmm_event columns.

`classify_gazepoint_export`*Classify a Gazepoint export*

Description

Uses the filename and header structure to classify a file as all_gaze, fixations, summary, or unknown.

Usage

```
classify_gazepoint_export(path)
```

Arguments

path	File path.
------	------------

Value

A single character string.

clean_gazepoint_by_trackloss

Flag or filter Gazepoint data by trackloss

Description

Computes trackloss rates globally or within user-specified grouping columns, then flags or removes groups exceeding a transparent threshold. Trackloss can be supplied directly through a validity/tracking column or inferred from missing/out-of-range gaze coordinates.

Usage

```
clean_gazepoint_by_trackloss(
  data,
  group_cols = NULL,
  tracking_col = NULL,
  x_col = NULL,
  y_col = NULL,
  max_trackloss = 0.25,
  action = c("flag", "filter"),
  treat_zero_zero_as_loss = TRUE,
  rate_col = ".gp3_trackloss_rate",
  exclude_col = ".gp3_trackloss_exclude"
)
```

Arguments

data	A data frame.
group_cols	Optional character vector of grouping columns, for example participant and trial identifiers.
tracking_col	Optional tracking/validity column. Logical, numeric, and character encodings are supported.
x_col, y_col	Optional gaze coordinate columns used when tracking_col is not supplied.
max_trackloss	Maximum allowed trackloss proportion per group.
action	Either "flag" to retain all rows and add diagnostic columns, or "filter" to remove groups above the threshold.
treat_zero_zero_as_loss	If TRUE, (0, 0) gaze coordinates are treated as trackloss when using x_col and y_col.
rate_col, exclude_col	Names of the added diagnostic columns.

Value

A data frame with diagnostic columns. If action = "filter", rows from excluded groups are removed. A compact summary is stored in the "gp3_trackloss_summary" attribute.

Examples

```
x <- data.frame(
  participant = c("P1", "P1", "P2", "P2"),
  trial = c(1, 1, 1, 1),
  valid = c(1, 0, 1, 1)
)
clean_gazepoint_by_trackloss(
  x,
  group_cols = c("participant", "trial"),
  tracking_col = "valid",
  max_trackloss = 0.25
)
```

cluster_gazepoint_scanpaths

Cluster Gazepoint AOI scanpaths

Description

Cluster scanpaths from long-format AOI observations, pairwise scanpath distances, a numeric distance matrix, or a "dist" object.

Usage

```
cluster_gazepoint_scanpaths(
  x,
  k = 3,
  method = c("hierarchical", "pam"),
  linkage = c("average", "complete", "single", "ward.D2", "ward.D", "mcquitty", "median",
    "centroid"),
  aoi_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  distance_col = "normalized_distance",
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  max_sequences = 200
)
```

Arguments

- x One of:
- a long-format AOI data frame;
 - output from [compute_gazepoint_scanpath_similarity](#);
 - a square numeric distance matrix; or
 - a "dist" object.

k	Integer number of clusters. Must be at least 2 and smaller than the number of scanpaths.
method	Clustering method: "hierarchical" or "pam".
linkage	Hierarchical linkage method passed to hclust .
aoi_col	AOI column when x is long-format AOI data.
group_cols	Columns identifying independent scanpaths when x is long-format AOI data.
time_col	Optional ordering column for long-format AOI data.
distance_col	Distance column when x is a pairwise-distance data frame. Defaults to "normalized_distance".
include_missing	Should missing AOI labels be retained as a state?
missing_label	Label used when retaining missing AOIs.
collapse_repeats	Should consecutive repeated AOI labels be collapsed before pairwise distances are calculated?
max_sequences	Maximum number of grouped scanpaths permitted when pairwise distances must be calculated.

Details

Long-format AOI data are converted to pairwise normalized edit distances with [compute_gazepoint_scanpath_similarity](#). Hierarchical clustering uses base R. Partitioning around medoids requires the optional **cluster** package.

Value

An object of class "gp3_scanpath_clusters" containing:

- assignments: scanpath identifiers and cluster assignments;
- distance: the clustering distance object;
- model: the fitted hierarchical or PAM model;
- medoids: PAM medoid identifiers, when applicable;
- silhouette: scanpath-level silhouette diagnostics when **cluster** is available;
- clustering settings and status fields.

Examples

```
distance_matrix <- matrix(
  c(
    0, 1, 5, 6,
    1, 0, 6, 5,
    5, 6, 0, 1,
    6, 5, 1, 0
  ),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(
```

```

      c("scanpath_1", "scanpath_2", "scanpath_3", "scanpath_4"),
      c("scanpath_1", "scanpath_2", "scanpath_3", "scanpath_4")
    )
  )

  result <- cluster_gazepoint_scanpaths(
    distance_matrix,
    k = 2,
    method = "hierarchical"
  )

  result$assignments

```

```
collect_gazepoint_qc_summaries
```

Collect Gazepoint QC summaries

Description

Collects compact overview/status information from gp3tools audit, workflow, checklist, readiness, diagnostic, or reporting objects. The helper is intended to make existing QC outputs easier to review together; it does not rerun checks or define exclusion rules.

Usage

```

collect_gazepoint_qc_summaries(
  objects,
  object_names = NULL,
  name = "gazepoint_qc_summary_bundle",
  include_overview_rows = TRUE
)

```

Arguments

<code>objects</code>	A list of gp3tools objects, audit objects, overview data frames, or a single such object.
<code>object_names</code>	Optional character names for unnamed objects.
<code>name</code>	Character label stored in the returned object.
<code>include_overview_rows</code>	Logical. If TRUE, returns a combined long-form table of interpretable overview rows.

Value

A list with overview, object_summary, overview_rows, and settings.

Examples

```
audit <- list(
  overview = data.frame(
    audit_status = "ok",
    message = "Example audit passed."
  )
)
collect_gazepoint_qc_summaries(list(example_audit = audit))
```

 combine_gazepoint_eyes

Combine left and right Gazepoint eye channels

Description

Combines two numeric eye-specific columns into a single analysis column. The helper is intentionally simple and transparent: it can average available left/right values, prefer one eye with fallback to the other, or select the globally less-missing eye as a pragmatic "best eye" rule.

Usage

```
combine_gazepoint_eyes(
  data,
  left_col,
  right_col,
  output_col = "combined_eye",
  method = c("mean", "left", "right", "prefer_left", "prefer_right", "best"),
  valid_min = NULL,
  valid_max = NULL
)
```

Arguments

data	A data frame.
left_col, right_col	Character names of the left- and right-eye columns.
output_col	Character name of the combined output column.
method	Combination rule. One of "mean", "left", "right", "prefer_left", "prefer_right", or "best".
valid_min, valid_max	Optional numeric bounds. Values outside these bounds are treated as missing before combination.

Value

A copy of data with output_col added.

Examples

```
x <- data.frame(left_pupil = c(3.1, NA, 3.4), right_pupil = c(3.3, 3.2, NA))
combine_gazepoint_eyes(x, "left_pupil", "right_pupil", "pupil")
```

```
compare_gazepoint_event_detectors
```

Compare Gazepoint event-detection workflows

Description

Run native velocity-threshold detection, the lightweight gp3tools HMM classifier, and an optional eyetools branch on the same sample-level gaze data. Detector failures are recorded without invalidating successful branches.

Usage

```
compare_gazepoint_event_detectors(
  data,
  id_col = "USER_ID",
  trial_col = NULL,
  group_cols = NULL,
  x_col = "FPOGX",
  y_col = "FPOGY",
  time_col = "TIME",
  methods = c("velocity", "hmm", "eyetools"),
  velocity_thresholds = c(5, 10, 20),
  min_duration = 50,
  hmm_states = 3L,
  eyetools_method = c("vti", "dispersion"),
  run_optional_eyetools = FALSE,
  min_overlap = 0.5,
  velocity_args = list(),
  hmm_args = list(),
  eyetools_args = list()
)
```

Arguments

<code>data</code>	A sample-level gaze data frame.
<code>id_col</code>	Participant identifier column.
<code>trial_col</code>	Optional trial identifier column.
<code>group_cols</code>	Optional additional sequence columns.
<code>x_col, y_col</code>	Gaze-coordinate columns.
<code>time_col</code>	Timestamp column.
<code>methods</code>	Detector branches to run: "velocity", "hmm", and "eyetools".

velocity_thresholds One or more positive velocity thresholds.

min_duration Minimum fixation duration in milliseconds for the native velocity branch.

hmm_states Number of HMM states.

eyetools_method eyetools detector method.

run_optional_eyetools Should the optional eyetools branch be run?

min_overlap Minimum intersection-over-union used to classify overlapping events as matched.

velocity_args Named list overriding native velocity-detector defaults.

hmm_args Named list overriding HMM-classifier defaults.

eyetools_args Named list overriding eyetools-wrapper defaults.

Value

An object of class "gp3_event_detector_comparison" containing standardized event tables, detector-run status, detector summaries, pairwise agreement, unmatched events, raw detector outputs, and settings.

Examples

```
gaze <- data.frame(
  USER_ID = rep("P01", 80),
  trial = rep("T01", 80),
  TIME = seq(0, by = 0.01, length.out = 80),
  FPOGX = c(rep(0.2, 30), seq(0.2, 0.8, length.out = 10), rep(0.8, 40)),
  FPOGY = 0.5
)

comparison <- compare_gazepoint_event_detectors(
  gaze,
  trial_col = "trial",
  methods = "velocity",
  velocity_thresholds = c(5, 10)
)

comparison$detector_summary
```

compare_gazepoint_nested_models

Compare nested Gazepoint models

Description

Compare a sequence of nested models, such as null, main-effect, time, condition, and interaction models. The helper returns model-level fit indices, likelihood-ratio comparisons, ranking information, and extraction statuses.

Usage

```
compare_gazepoint_nested_models(
  models,
  model_names = NULL,
  comparison = c("sequential", "against_first"),
  name = "gazepoint_nested_model_comparison"
)
```

Arguments

models	A list of fitted model objects.
model_names	Optional character vector of model names. If NULL, names are taken from models or generated as model_1, model_2, etc.
comparison	Comparison strategy. "sequential" compares each model with the previous model. "against_first" compares each model with the first model.
name	Character label stored in the returned object.

Details

This helper is useful for GCA, confirmatory LMM/GLMM workflows, AOI GLMMs, pupil LMMs, and other fitted model workflows where reviewers expect explicit model-comparison evidence.

Value

A list with class `gp3_nested_model_comparison`.

```
compute_gazepoint_aoi_entropy
```

Compute AOI entropy metrics

Description

Computes spatial AOI entropy, directed transition entropy, and conditional transition entropy for Gazepoint-style AOI sequences. The function is useful for quantifying how concentrated, dispersed, or predictable gaze allocation is across Areas of Interest.

Usage

```
compute_gazepoint_aoi_entropy(
  data,
  aoi_col,
  group_cols = NULL,
  time_col = NULL,
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  log_base = 2
)
```

Arguments

<code>data</code>	A data frame containing AOI observations.
<code>aoi_col</code>	Character scalar. Column containing AOI labels.
<code>group_cols</code>	Optional character vector of grouping columns, such as participant, trial, stimulus, or condition columns.
<code>time_col</code>	Optional character scalar. If supplied, observations are ordered by this column within each group before transitions are computed.
<code>include_missing</code>	Logical. If TRUE, missing or empty AOI labels are retained as <code>missing_label</code> ; otherwise they are removed.
<code>missing_label</code>	Character scalar used when <code>include_missing = TRUE</code> .
<code>collapse_repeats</code>	Logical. If TRUE, consecutive identical AOI labels are collapsed before transition entropy is computed.
<code>log_base</code>	Numeric scalar. Base of the logarithm used for entropy.

Value

A data frame with one row per group and entropy/count columns.

Examples

```

dat <- data.frame(
  subject = "S01",
  trial = "T01",
  time = 1:6,
  AOI = c("A", "A", "B", "C", "B", "A")
)

compute_gazeport_aoi_entropy(
  dat,
  aoi_col = "AOI",
  group_cols = c("subject", "trial"),
  time_col = "time"
)

```

```
compute_gazeport_aoi_sequence_metrics
```

Compute AOI sequence metrics

Description

Computes compact scanpath-style descriptors from Gazeport AOI sequences, including sequence length, AOI visits, transitions, revisits, first and last AOI, dominant AOI, and run-length summaries.

Usage

```
compute_gazepoint_aoi_sequence_metrics(
  data,
  aoi_col,
  group_cols = NULL,
  time_col = NULL,
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = TRUE
)
```

Arguments

<code>data</code>	A data frame containing AOI observations.
<code>aoi_col</code>	Character scalar. Column containing AOI labels.
<code>group_cols</code>	Optional character vector of grouping columns.
<code>time_col</code>	Optional character scalar. If supplied, observations are ordered by this column within each group.
<code>include_missing</code>	Logical. If TRUE, missing or empty AOI labels are retained as <code>missing_label</code> ; otherwise they are removed.
<code>missing_label</code>	Character scalar used when <code>include_missing = TRUE</code> .
<code>collapse_repeats</code>	Logical. If TRUE, consecutive identical AOI labels are collapsed before visit, transition, and revisit metrics are computed.

Value

A data frame with one row per group and sequence-metric columns.

Examples

```
dat <- data.frame(
  subject = "S01",
  trial = "T01",
  time = 1:6,
  AOI = c("A", "A", "B", "A", "C", "C")
)

compute_gazepoint_aoi_sequence_metrics(
  dat,
  aoi_col = "AOI",
  group_cols = c("subject", "trial"),
  time_col = "time"
)
```

```
compute_gazepoint_aoi_transition_matrix
```

Compute Gazepoint AOI transition matrices

Description

Compute AOI transition count and probability matrices from sample-level Gazepoint AOI data, AOI-entry tables, or AOI-sequence tables. The function returns both matrix and long-table forms.

Usage

```
compute_gazepoint_aoi_transition_matrix(
  data,
  aoi_col = NULL,
  time_col = "time",
  group_cols = c("subject", "MEDIA_ID", "trial_global"),
  by_cols = NULL,
  include_non_aoi = TRUE,
  include_self_transitions = TRUE,
  states = NULL,
  time_window = NULL,
  non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
    "missing", "missing_aoi"),
  missing_aoi_label = "missing_aoi"
)
```

Arguments

<code>data</code>	A Gazepoint sample-level data frame, AOI-entry table, or AOI-sequence table.
<code>aoi_col</code>	Name of the AOI-state column. Used only when data is sample-level data. If NULL, the function tries <code>aoi_current</code> , <code>AOI</code> , and <code>aoi_state</code> .
<code>time_col</code>	Name of the time column, in milliseconds. Used only when data is sample-level data.
<code>group_cols</code>	Character vector of columns defining independent AOI sequences, usually subject/media/trial.
<code>by_cols</code>	Optional character vector of columns used to compute separate matrices, for example condition or MEDIA_ID.
<code>include_non_aoi</code>	Logical. If TRUE, non-AOI/background states are included in the transition matrix.
<code>include_self_transitions</code>	Logical. If TRUE, same-state transitions are retained. These can occur after non-AOI states are removed.
<code>states</code>	Optional character vector giving the desired row/column order for the transition matrices.

time_window	Optional numeric vector of length 2 giving an entry-start time window in milliseconds.
non_aoi_values	Character vector of AOI labels treated as background or non-AOI states.
missing_aoi_label	Label used when the AOI value is missing.

Value

A list containing count matrices, probability matrices, and long-form transition counts/probabilities.

compute_gazepoint_saccade_metrics

Compute basic saccade metrics from fixation coordinates

Description

Compute between-fixation displacement metrics from ordered fixation coordinates. The function does not perform raw event detection; it assumes that fixation-level coordinates are already available, for example from Gazepoint fixation exports.

Usage

```
compute_gazepoint_saccade_metrics(
  data,
  x_col,
  y_col,
  group_cols = NULL,
  time_col = NULL,
  start_time_col = NULL,
  end_time_col = NULL,
  distance_scale = 1,
  drop_missing = TRUE
)
```

Arguments

data	A fixation-level data frame.
x_col	Name of the horizontal fixation-coordinate column.
y_col	Name of the vertical fixation-coordinate column.
group_cols	Optional columns defining independent scanpaths, such as participant and trial.
time_col	Optional column used to order fixations and compute inter-fixation time differences.
start_time_col	Optional fixation-start column.
end_time_col	Optional fixation-end column. If both start and end columns are supplied, saccade duration is computed as the next fixation start minus the current fixation end.

distance_scale Multiplicative scale applied to coordinate distances.
 drop_missing Should rows with missing coordinates be dropped?

Value

A data frame with one row per between-fixation movement.

compute_gazepoint_scanpath_geometry

Compute scanpath geometry features by subject and trial

Description

Computes scanpath length, straight-line distance, efficiency, convex-hull area, and spatial dispersion from sequential gaze/fixation coordinates.

Usage

```
compute_gazepoint_scanpath_geometry(  
  data,  
  x,  
  y,  
  subject,  
  trial,  
  time = NULL,  
  condition = NULL  
)
```

Arguments

data	A data frame.
x	X-coordinate column.
y	Y-coordinate column.
subject	Subject column.
trial	Trial column.
time	Optional time column used for ordering.
condition	Optional condition column to carry forward.

Value

A trial-level data frame of scanpath geometry features.

```
compute_gazepoint_scanpath_similarity
```

Compute AOI scanpath similarity

Description

Compute pairwise AOI-sequence similarity between grouped scanpaths using a lightweight Levenshtein edit-distance implementation. Similarity is reported as $1 - \text{normalized_distance}$, where normalized distance is divided by the longer sequence length.

Usage

```
compute_gazepoint_scanpath_similarity(
  data,
  aoi_col,
  group_cols,
  time_col = NULL,
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  max_sequences = 200
)
```

Arguments

<code>data</code>	A data frame containing AOI observations.
<code>aoi_col</code>	Name of the AOI column.
<code>group_cols</code>	Columns defining each scanpath, for example subject and trial.
<code>time_col</code>	Optional time/order column.
<code>include_missing</code>	Should missing AOI labels be retained as a state?
<code>missing_label</code>	Label used when retaining missing AOIs.
<code>collapse_repeats</code>	Should consecutive repeated AOI labels be collapsed?
<code>max_sequences</code>	Maximum number of grouped sequences to compare.

Value

A long-format data frame containing pairwise edit distances, normalized distances, and similarities.

```
compute_gazepoint_sequence_complexity
```

Compute AOI sequence complexity metrics

Description

Compute compact sequence-complexity summaries from AOI labels. This helper complements transition and entropy summaries by returning simple, interpretable indices such as type-token ratio, transition density, normalized entropy, and a combined complexity index.

Usage

```
compute_gazepoint_sequence_complexity(  
  data = NULL,  
  sequence = NULL,  
  aoi_col = NULL,  
  group_cols = NULL,  
  time_col = NULL,  
  include_missing = FALSE,  
  missing_label = "missing",  
  collapse_repeats = FALSE  
)
```

Arguments

<code>data</code>	Optional data frame containing AOI observations.
<code>sequence</code>	Optional AOI sequence vector. Used when data is not supplied.
<code>aoi_col</code>	Name of the AOI column when data is supplied.
<code>group_cols</code>	Optional grouping columns.
<code>time_col</code>	Optional time/order column.
<code>include_missing</code>	Should missing AOI labels be retained as a state?
<code>missing_label</code>	Label used when retaining missing AOIs.
<code>collapse_repeats</code>	Should consecutive repeated AOI labels be collapsed?

Value

A data frame with sequence length, unique-state count, entropy, transition density, type-token ratio, and complexity index.

```
compute_gazepoint_sequence_distance
```

Compute AOI sequence distance

Description

Computes a lightweight edit distance between two AOI sequences using a vector-based Levenshtein distance. This provides a simple scanpath dissimilarity measure without requiring heavy sequence-analysis dependencies.

Usage

```
compute_gazepoint_sequence_distance(
  sequence_a,
  sequence_b,
  ignore_missing = TRUE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  substitution_cost = 1,
  insertion_cost = 1,
  deletion_cost = 1
)
```

Arguments

<code>sequence_a</code>	Character, factor, or atomic vector representing the first AOI sequence.
<code>sequence_b</code>	Character, factor, or atomic vector representing the second AOI sequence.
<code>ignore_missing</code>	Logical. If TRUE, missing and empty labels are removed.
<code>missing_label</code>	Character scalar used when <code>ignore_missing = FALSE</code> .
<code>collapse_repeats</code>	Logical. If TRUE, consecutive identical labels are collapsed before distance is computed.
<code>substitution_cost</code>	Numeric scalar substitution cost.
<code>insertion_cost</code>	Numeric scalar insertion cost.
<code>deletion_cost</code>	Numeric scalar deletion cost.

Value

A one-row data frame with edit distance, normalized distance, and sequence lengths.

Examples

```
compute_gazepoint_sequence_distance(
  sequence_a = c("Claim", "Evidence", "CTA"),
  sequence_b = c("Claim", "CTA", "Evidence")
)
```

`compute_gazepoint_sequence_recurrence`*Compute simple categorical sequence recurrence metrics*

Description

Compute lightweight recurrence metrics from AOI/state sequences. This is a compact categorical recurrence helper, not a full replacement for dedicated CRQA/RQA packages.

Usage

```
compute_gazepoint_sequence_recurrence(  
  data = NULL,  
  sequence = NULL,  
  aoi_col = NULL,  
  group_cols = NULL,  
  time_col = NULL,  
  min_line = 2,  
  include_missing = FALSE,  
  missing_label = "missing"  
)
```

Arguments

<code>data</code>	Optional long-format data frame.
<code>sequence</code>	Optional AOI/state vector used when data is absent.
<code>aoi_col</code>	AOI/state column when data is supplied.
<code>group_cols</code>	Optional grouping columns.
<code>time_col</code>	Optional ordering column.
<code>min_line</code>	Minimum diagonal-line length for determinism.
<code>include_missing</code>	Should missing states be retained?
<code>missing_label</code>	Label used for retained missing states.

Value

A data frame of recurrence metrics.

```
compute_gazepoint_time_varying_transition_matrix
```

Compute time-varying Gazepoint transition matrices

Description

Compute transition-count and transition-probability matrices across time windows. This helper is a convenience wrapper for studies where AOI/state transitions are expected to vary over the course of a stimulus, trial, or analysis window.

Usage

```
compute_gazepoint_time_varying_transition_matrix(
  data,
  from_col = NULL,
  to_col = NULL,
  time_col = NULL,
  window_col = NULL,
  window_size_ms = NULL,
  by_cols = NULL,
  count_col = NULL,
  states = NULL,
  complete_states = TRUE,
  drop_self_transitions = FALSE,
  normalise = c("row", "global", "none"),
  name = "gazepoint_time_varying_transition_matrix"
)
```

Arguments

<code>data</code>	A data frame containing transition-level rows.
<code>from_col</code>	Transition origin column. If NULL, common origin columns are detected automatically.
<code>to_col</code>	Transition destination column. If NULL, common destination columns are detected automatically.
<code>time_col</code>	Optional numeric time column used to construct windows when <code>window_col = NULL</code> .
<code>window_col</code>	Optional existing time-window column.
<code>window_size_ms</code>	Numeric window size used when <code>window_col = NULL</code> .
<code>by_cols</code>	Optional grouping columns, such as subject, condition, trial, or stimulus.
<code>count_col</code>	Optional count/weight column. If NULL, each row contributes one transition.
<code>states</code>	Optional character vector of allowed states/AOIs. If NULL, states are detected from <code>from_col</code> and <code>to_col</code> .

complete_states	If TRUE, complete all state-pair combinations within each time window and group.
drop_self_transitions	If TRUE, remove transitions where origin and destination are the same.
normalise	Probability normalisation. Options are "row", "global", and "none".
name	Character label stored in the returned object.

Value

A list with class `gp3_time_varying_transition_matrix`.

`compute_gazepoint_transition_network_metrics`
Compute lightweight AOI transition-network metrics

Description

Compute graph-style summaries from AOI transitions without requiring network packages. Metrics include state count, edge count, density, self-loops, and in/out-degree summaries.

Usage

```
compute_gazepoint_transition_network_metrics(
  data,
  aoi_col = NULL,
  from_col = NULL,
  to_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  include_self_loops = TRUE
)
```

Arguments

<code>data</code>	Optional data frame of AOI observations or transition rows.
<code>aoi_col</code>	AOI column for raw sequence data.
<code>from_col</code>	Source-state column for transition data.
<code>to_col</code>	Destination-state column for transition data.
<code>group_cols</code>	Optional grouping columns for raw sequence data.
<code>time_col</code>	Optional ordering column.
<code>include_self_loops</code>	Should self-transitions be included?

Value

A list with graph-level and state-level summaries.

```
compute_transition_matrix
```

Compute an AOI transition matrix

Description

Compute an AOI transition matrix

Usage

```
compute_transition_matrix(
  data,
  group_cols = "MEDIA_ID",
  aoi_col = "AOI",
  time_col = "TIME",
  collapse_repeats = TRUE
)
```

Arguments

<code>data</code>	A Gazepoint data frame with AOI labels.
<code>group_cols</code>	Columns defining independent sequences.
<code>aoi_col</code>	AOI column name.
<code>time_col</code>	Time column name.
<code>collapse_repeats</code>	If TRUE, consecutive identical AOI labels are reduced to one visit before transitions are counted.

Value

A tibble with from, to, n, and prob.

```
construct_gazepoint_combined_pupil
```

Construct an explicitly governed combined pupil signal

Description

Builds a combined pupil series under a declared policy while retaining a per-row source label. This function replaces opaque `rowMeans(..., na.rm = TRUE)` decisions with explicit provenance.

Usage

```
construct_gazepoint_combined_pupil(
  data,
  left_col,
  right_col,
  policy = c("complete_case", "available_eye", "reconstructed_mean", "left_only",
    "right_only"),
  prefix = "gp3_binocular",
  output_col = "pupil_binocular",
  status_col = "pupil_binocular_status",
  valid_min = NULL,
  valid_max = NULL,
  overwrite = FALSE
)
```

Arguments

<code>data</code>	Raw data or output from reconstruct_gazepoint_binocular_pupil() .
<code>left_col, right_col</code>	Original pupil column names.
<code>policy</code>	One of "complete_case", "available_eye", "reconstructed_mean", "left_only", or "right_only".
<code>prefix</code>	Reconstruction prefix used by reconstructed_mean.
<code>output_col</code>	Name of the constructed pupil column.
<code>status_col</code>	Name of the provenance column.
<code>valid_min, valid_max</code>	Optional bounds when working directly from raw channels.
<code>overwrite</code>	Whether existing output columns may be replaced.

Value

data with `output_col` and `status_col` appended.

See Also

[reconstruct_gazepoint_binocular_pupil\(\)](#), [analyse_gazepoint_binocular_sensitivity\(\)](#)

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 3, n_trials = 2, seed = 14)
dat$pupil_right[30:33] <- NA_real_
construct_gazepoint_combined_pupil(
  dat, "pupil_left", "pupil_right", policy = "available_eye"
)
```

```
create_gazepoint_analysis_decision_audit
```

Create a final Gazepoint analysis-decision audit

Description

Create a final audit table for a Gazepoint analysis workflow. The function records which analysis branches were run, classifies them as confirmatory, sensitivity, exploratory, diagnostic, preprocessing, or reporting branches, summarises available diagnostics, flags interpretation cautions, and creates a final analysis-readiness table.

Usage

```
create_gazepoint_analysis_decision_audit(
  ...,
  results = NULL,
  branch_roles = NULL,
  required_confirmatory = character(),
  diagnostics_required = TRUE,
  require_clean_diagnostics = FALSE
)
```

Arguments

...	Named analysis result objects. Each named object is treated as one analysis branch.
results	Optional named list of analysis result objects. This can be used instead of, or together with, ...
branch_roles	Optional data frame describing branch roles. It must contain branch_name and decision_type. Optional columns include analysis_family, interpretation_scope, and notes.
required_confirmatory	Character vector of confirmatory branches that must be present for the analysis to be considered complete.
diagnostics_required	Logical. If TRUE, confirmatory model branches without extractable diagnostics are flagged with a caution.
require_clean_diagnostics	Logical. If TRUE, diagnostic warnings in required confirmatory branches make the final readiness status not_ready.

Value

A list with class gp3_analysis_decision_audit containing overview, branch audit, diagnostics summary, interpretation cautions, readiness, and settings tables.

`create_gazepoint_bayesian_sap`*Create a Bayesian ocular Statistical Analysis Plan checklist*

Description

Generates a structured checklist for Bayesian or advanced eye-tracking and pupillometry analysis planning.

Usage

```
create_gazepoint_bayesian_sap(  
  outcome,  
  design,  
  primary_model,  
  baseline_window = NULL,  
  analysis_window = NULL,  
  missingness_threshold = 0.2,  
  blink_padding_ms = 50,  
  output = c("data.frame", "markdown")  
)
```

Arguments

<code>outcome</code>	Outcome name or outcome family.
<code>design</code>	Study design description.
<code>primary_model</code>	Planned primary model.
<code>baseline_window</code>	Optional baseline window.
<code>analysis_window</code>	Optional analysis window.
<code>missingness_threshold</code>	Trial-level missingness threshold.
<code>blink_padding_ms</code>	Blink padding in milliseconds.
<code>output</code>	Either "data.frame" or "markdown".

Value

A data frame or markdown character vector.

```
create_gazepoint_brms_template
```

Create brms formula and prior templates for Gazepoint-derived metrics

Description

Returns formula, family, prior, and reporting-note templates. The function does not require or call brms.

Usage

```
create_gazepoint_brms_template(
  metric_type,
  outcome,
  time = NULL,
  condition = NULL,
  subject = NULL,
  item = NULL
)
```

Arguments

metric_type	Metric type, such as "pupil_timecourse", "fixation_duration", "fixation_count", "aoi_proportion", or "binary_choice".
outcome	Outcome column name.
time	Optional time column.
condition	Optional condition column.
subject	Optional subject column.
item	Optional item/stimulus column.

Value

A list containing formula, family, priors, and notes.

```
create_gazepoint_cross_package_report
```

Create a combined gp3tools-gpbiometrics workflow report

Description

Create a combined gp3tools-gpbiometrics workflow report

Usage

```
create_gazepoint_cross_package_report(x, output_file = NULL)
```

Arguments

`x` A "gazepoint_cross_package_workflow" object.
`output_file` Optional Markdown output path.

Value

Character vector containing a compact Markdown report.

```
create_gazepoint_event_review_template
```

Create a manual event-review template

Description

Create a sequence-level CSV-ready template for manually annotating fixation intervals. The returned table contains one or more placeholder rows per participant/trial sequence together with the observed sequence boundaries. Reviewers should fill `start_time` and `end_time`, set `review_status` to "accepted", and add rows when a sequence contains multiple events.

Usage

```
create_gazepoint_event_review_template(  
  data,  
  id_col = "USER_ID",  
  trial_col = NULL,  
  group_cols = NULL,  
  time_col = "TIME",  
  rows_per_sequence = 1L,  
  event_type = "fixation",  
  reviewer = NA_character_  
)
```

Arguments

`data` Sample-level gaze data.
`id_col` Participant identifier column.
`trial_col` Optional trial identifier column.
`group_cols` Optional additional sequence columns.
`time_col` Timestamp column.
`rows_per_sequence` Number of placeholder rows created per sequence.
`event_type` Event label inserted in the template.
`reviewer` Optional reviewer identifier.

Value

A data frame suitable for export to CSV and manual review.

Examples

```
gaze <- data.frame(
  USER_ID = rep(c("P01", "P02"), each = 10),
  trial = rep("T01", 20),
  TIME = rep(seq(0, 0.09, by = 0.01), 2),
  FPOGX = 0.5,
  FPOGY = 0.5
)

template <- create_gazepoint_event_review_template(
  gaze,
  trial_col = "trial"
)

template
```

```
create_gazepoint_face_reporting_checklist
```

Create a reporting checklist for external facial-behaviour workflows

Description

Creates a compact checklist for reporting external facial-behaviour analyses alongside Gazepoint data. The checklist is designed for reviewer-facing transparency: it records whether import, quality auditing, synchronisation, window summaries, reactivity summaries, and modelling outputs are available. It also includes interpretation cautions. The helper does not infer facial expressions or emotional states.

Usage

```
create_gazepoint_face_reporting_checklist(
  face_data = NULL,
  quality_audit = NULL,
  sync_audit = NULL,
  window_summary = NULL,
  reactivity_summary = NULL,
  multimodal_model = NULL,
  include_interpretation_cautions = TRUE
)
```

Arguments

face_data	Optional imported or standardised face-analysis data.
quality_audit	Optional object returned by audit_gazepoint_face_quality().
sync_audit	Optional object returned by audit_gazepoint_face_sync().
window_summary	Optional object returned by summarize_gazepoint_face_windows().
reactivity_summary	Optional object returned by summarize_gazepoint_face_reactivity().
multimodal_model	Optional object returned by fit_gazepoint_face_window_lmm() or fit_gazepoint_multimodal_res.
include_interpretation_cautions	Should interpretation-caution checklist items be included?

Value

A tibble with class gp3_face_reporting_checklist.

Examples

```
quality <- list(
  overview = data.frame(
    n_rows = 10,
    valid_percent = 95,
    face_quality_status = "pass"
  ),
  issue_summary = data.frame(
    issue = "missing_confidence",
    n_groups_affected = 0
  )
)
class(quality) <- c("gp3_face_quality_audit", "list")

create_gazepoint_face_reporting_checklist(quality_audit = quality)
```

```
create_gazepoint_hddm_fit_script
```

Create a Python HDDM fitting script from a Gazepoint HDDM export

Description

Writes a Python script for fitting an HDDMRegressor model. The function does not fit HDDM inside R and does not require Python. It creates a reproducible script that can be run in a Python/HDDM environment.

Usage

```
create_gazepoint_hddm_fit_script(
  data_file,
  output_file = "fit_gazepoint_hddm.py",
  regressions = c(v = "target_dwell_ms_z", a = "pupil_peak_z"),
  include = c("v", "a", "t"),
  draws = 5000,
  burn = 2000,
  dbname = "hddm_traces.db"
)
```

Arguments

<code>data_file</code>	Path to a CSV file prepared with <code>prepare_gazepoint_hddm_export()</code> .
<code>output_file</code>	Path where the Python script should be written.
<code>regressions</code>	Named character vector. Names should be DDM parameters such as "v", "a", "t", or "z"; values should be predictor terms.
<code>include</code>	Character vector of DDM parameters to include.
<code>draws</code>	Number of posterior draws.
<code>burn</code>	Number of burn-in samples.
<code>dbname</code>	HDDM trace database name.

Value

Invisibly returns `output_file`.

```
create_gazepoint_markovchain_object
```

Create a Gazepoint AOI Markov-chain object

Description

Create a dependency-free Markov-chain-style object from AOI/state sequences. The function computes transition counts, transition probabilities, and matrix representations from ordered gaze/AOI states. It does not require the external `markovchain` package; instead, it returns a lightweight `gp3tools` object that can be inspected, exported, or converted later.

Usage

```
create_gazepoint_markovchain_object(
  data,
  state_col = NULL,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
```

```

sequence_id_cols = NULL,
state_order = NULL,
exclude_states = c("missing", "missing_aoi", "missing_coordinate", "trackloss",
  "track_loss"),
missing_state_label = NULL,
include_self_transitions = TRUE,
laplace = 0,
empty_state_handling = c("self", "zero", "NA"),
name = "gazepoint_markovchain"
)

```

Arguments

<code>data</code>	A data frame containing ordered AOI/state observations.
<code>state_col</code>	AOI/state column. If NULL, common AOI/state column names are detected automatically.
<code>participant_col</code>	Optional participant/subject column.
<code>trial_col</code>	Optional trial/sequence column.
<code>time_col</code>	Optional time/order column.
<code>sequence_id_cols</code>	Optional character vector of columns defining separate sequences. If NULL, participant and trial columns are used when available.
<code>state_order</code>	Optional character vector giving the preferred state order in the output matrices.
<code>exclude_states</code>	Character vector of states to exclude before transition calculation.
<code>missing_state_label</code>	Optional label used to retain missing states. If NULL, missing/blank states are removed.
<code>include_self_transitions</code>	Logical. If FALSE, transitions from a state to the same state are removed.
<code>laplace</code>	Numeric smoothing value added to all transition cells when computing probabilities.
<code>empty_state_handling</code>	How to handle states with no outgoing transitions: "self" creates an absorbing self-transition, "zero" leaves a zero row, and "NA" returns NA probabilities for that row.
<code>name</code>	Character label stored in the returned object.

Value

A list with class `gp3_markovchain_object`.

```
create_gazepoint_master
```

Create a master long-format dataset from Gazepoint all-gaze data

Description

Converts Gazepoint all-gaze exports imported with `read_gazepoint()` or `read_gazepoint_folder()` into a master sample-level structure suitable for quality checks, AOI summaries, pupil preprocessing, time-course analyses, and publication reporting.

Usage

```
create_gazepoint_master(
  gaze_data,
  screen_width_px = NULL,
  screen_height_px = NULL,
  screen_width_cm = NULL,
  screen_height_cm = NULL,
  viewing_distance_cm = NULL,
  time_unit = c("seconds", "milliseconds"),
  user_col = "USER_FILE",
  media_col = "MEDIA_ID",
  media_name_col = "MEDIA_NAME",
  tracker_model = "Gazepoint",
  tracker_sampling_rate = 60,
  event_latency_offset_ms = 0,
  baseline_window = NULL,
  analysis_window = NULL
)
```

Arguments

<code>gaze_data</code>	Gazepoint all-gaze data frame.
<code>screen_width_px</code>	Optional screen width in pixels. If provided and gaze coordinates are normalised, x-coordinates are converted to pixels.
<code>screen_height_px</code>	Optional screen height in pixels. If provided and gaze coordinates are normalised, y-coordinates are converted to pixels.
<code>screen_width_cm</code>	Optional physical screen width in centimetres.
<code>screen_height_cm</code>	Optional physical screen height in centimetres.
<code>viewing_distance_cm</code>	Optional viewing distance in centimetres.
<code>time_unit</code>	Unit of the Gazepoint TIME column. Usually "seconds".

user_col	Column identifying the source/user file.
media_col	Column identifying the stimulus/media.
media_name_col	Column identifying the media/stimulus name.
tracker_model	Tracker label stored in the master data.
tracker_sampling_rate	Expected tracker sampling rate.
event_latency_offset_ms	Optional event-latency correction in milliseconds.
baseline_window	Optional numeric vector of length 2 giving baseline start and end in milliseconds.
analysis_window	Optional numeric vector of length 2 giving analysis start and end in milliseconds.

Value

A tibble with one row per Gazepoint sample and publication-oriented master columns.

```
create_gazepoint_preprocessing_multiverse
```

Create a Gazepoint preprocessing multiverse

Description

Create a preprocessing multiverse specification for Gazepoint pupil and AOI workflows. The returned object defines alternative preprocessing decisions that can later be passed to pupil or AOI multiverse runners.

Usage

```
create_gazepoint_preprocessing_multiverse(
  pupil_max_gap_ms = c(75, 150, 250),
  pupil_smoothing_window_samples = c(3L, 5L, 7L),
  pupil_baseline_windows = list(c(0, 200), c(-200, 0)),
  pupil_artifact_padding_ms = c(0, 50),
  aoi_denominators = c("valid", "all", "aoi_only"),
  aoi_min_denominator_samples = c(1L, 5L, 10L),
  include_pupil = TRUE,
  include_aoi = TRUE,
  label_prefix = "mv"
)
```

Arguments

pupil_max_gap_ms	Numeric vector of maximum pupil-interpolation gap durations in milliseconds.
pupil_smoothing_window_samples	Integer vector of pupil smoothing window sizes in samples.
pupil_baseline_windows	List of numeric vectors of length 2 defining pupil baseline windows in milliseconds.
pupil_artifact_padding_ms	Numeric vector of artifact-padding values in milliseconds.
aoi_denominators	Character vector of AOI denominator definitions. Typical values are "valid", "all", and "aoi_only".
aoi_min_denominator_samples	Integer vector of minimum denominator sample thresholds for AOI-window modelling.
include_pupil	Logical. If TRUE, create pupil preprocessing branches.
include_aoi	Logical. If TRUE, create AOI preprocessing branches.
label_prefix	Character prefix used for branch identifiers.

Value

A list with class `gp3_preprocessing_multiverse` containing overview, pupil grid, AOI grid, combined grid, and settings tables.

```
create_gazepoint_preprocessing_registry
```

Create a Gazepoint pupil-preprocessing registry

Description

Creates a compact registry of commonly used preprocessing parameters for Gazepoint pupil and gaze analyses. The registry is designed to make preprocessing choices explicit, auditable, and easy to report.

Usage

```
create_gazepoint_preprocessing_registry(
  blink_padding_pre_ms = 100,
  blink_padding_post_ms = 100,
  max_interpolation_gap_ms = 150,
  smoothing_window_ms = 50,
  baseline_start_ms = -200,
  baseline_end_ms = 0,
  pupil_physiological_min = 1,
```

```

    pupil_physiological_max = 9,
    pupil_speed_mad_k = 6,
    binocular_mad_k = 6,
    baseline_missing_prop_threshold = 0.3,
    baseline_interpolated_prop_threshold = 0.3,
    baseline_artifact_prop_threshold = 0.3,
    overlap_trial_duration_ms = 3000,
    overlap_event_gap_ms = 1000
)

```

Arguments

blink_padding_pre_ms
Padding before bad pupil samples, blinks, or tracking artifacts, in milliseconds. Defaults to 100.

blink_padding_post_ms
Padding after bad pupil samples, blinks, or tracking artifacts, in milliseconds. Defaults to 100.

max_interpolation_gap_ms
Maximum missing-pupil gap duration to interpolate, in milliseconds. Defaults to 150.

smoothing_window_ms
Rolling smoothing window, in milliseconds. Defaults to 50.

baseline_start_ms
Baseline-window start, in milliseconds. Defaults to -200.

baseline_end_ms
Baseline-window end, in milliseconds. Defaults to 0.

pupil_physiological_min
Minimum plausible pupil value when the pupil unit is known to be millimetres. Defaults to 1.

pupil_physiological_max
Maximum plausible pupil value when the pupil unit is known to be millimetres. Defaults to 9.

pupil_speed_mad_k
MAD multiplier for pupil-speed outlier detection. Defaults to 6.

binocular_mad_k
MAD multiplier for left-right pupil disagreement. Defaults to 6.

baseline_missing_prop_threshold
Baseline missingness threshold used for baseline-quality audits. Defaults to 0.30.

baseline_interpolated_prop_threshold
Baseline interpolation threshold used for baseline-quality audits. Defaults to 0.30.

baseline_artifact_prop_threshold
Baseline artifact threshold used for baseline-quality audits. Defaults to 0.30.

overlap_trial_duration_ms
Trial-duration threshold below which pupil overlap/deconvolution risk should be considered. Defaults to 3000.

overlap_event_gap_ms

Event-gap threshold below which pupil-response overlap should be considered.
Defaults to 1000.

Value

A tibble with one row per preprocessing parameter.

Examples

```
registry <- create_gazepoint_preprocessing_registry()
registry
```

create_gazepoint_report

Create a Gazepoint HTML diagnostic report

Description

Creates a lightweight HTML report from a gp3tools workflow result object. The report includes dataset dimensions, sampling-rate checks, flagged recordings, AOI summaries, and standard diagnostic plots.

Usage

```
create_gazepoint_report(
  results,
  output_file,
  title = "Gazepoint diagnostic report",
  overwrite = TRUE,
  max_rows = 30,
  save_plots = TRUE,
  plot_dir = NULL
)
```

Arguments

results	A named list returned by run_gazepoint_workflow().
output_file	Path to the HTML report file to create.
title	Report title.
overwrite	Logical. If FALSE, stop when the report file already exists.
max_rows	Maximum number of rows to show in preview tables.
save_plots	Logical. If TRUE, save and include diagnostic plots.
plot_dir	Optional folder where report plot files should be saved. If NULL, a folder next to the report is created.

Value

A tibble with the written report path and plot folder.

```
create_gazepoint_reporting_checklist
```

Create a Gazepoint reporting checklist

Description

Create an auto-generated reporting checklist for Gazepoint/gp3tools analyses. The checklist summarises whether key dataset, preprocessing, quality-control, AOI, pupil, modelling, sensitivity, and reproducibility elements are present or still need reporting.

Usage

```
create_gazepoint_reporting_checklist(  
  data = NULL,  
  objects = NULL,  
  analysis_type = c("general", "pupil", "aoi", "combined"),  
  study_title = NULL,  
  required_sections = NULL,  
  include_optional = TRUE,  
  name = "gazepoint_reporting_checklist"  
)
```

Arguments

<code>data</code>	Optional Gazepoint or gp3tools data frame.
<code>objects</code>	Optional list of gp3tools audit, model, workflow, readiness, or external-check objects.
<code>analysis_type</code>	Analysis target. Options are "general", "pupil", "aoi", and "combined".
<code>study_title</code>	Optional study title or short label.
<code>required_sections</code>	Optional character vector of checklist item IDs that should be treated as required. If NULL, a default set is used.
<code>include_optional</code>	Logical. If TRUE, include optional advanced-methods reporting items.
<code>name</code>	Character label stored in the returned object.

Details

This helper is intended as a reporting aid. It does not replace the underlying audit, preprocessing, modelling, or readiness-gate functions.

Value

A list with class `gp3_reporting_checklist`.

detect_gazepoint_blinks

Detect blink intervals from pupil measurements

Description

Detects blink-like periods using missing or non-positive pupil samples and optional robust drop/recovery rules. Multiple pupil columns can be supplied; their row-wise mean is used for detection.

Usage

```
detect_gazepoint_blinks(
  all_gaze,
  pupil_col = NULL,
  ts_col = "TIME",
  id_col = "USER_ID",
  group_cols = NULL,
  min_duration = 50,
  z_thresh = 4,
  zero_threshold = 0,
  merge_gap_ms = 20,
  time_unit = c("auto", "seconds", "milliseconds"),
  include_rapid_changes = TRUE,
  return = c("events", "samples", "both")
)
```

Arguments

all_gaze	A data frame containing sample-level pupil data.
pupil_col	Pupil column or columns. When NULL, common gp3tools and Gazepoint pupil names are detected automatically.
ts_col	Timestamp column.
id_col	Participant identifier column.
group_cols	Optional additional grouping columns.
min_duration	Minimum retained blink duration in milliseconds.
z_thresh	Robust threshold for low values and rapid changes.
zero_threshold	Values at or below this threshold are invalid.
merge_gap_ms	Merge blink candidates separated by no more than this duration.
time_unit	Timestamp unit.
include_rapid_changes	Include robust rapid drop and recovery flags.
return	Return event intervals, sample labels, or both.

Value

A blink-event tibble, a labelled sample table, or both.

Examples

```
pupil <- data.frame(
  USER_ID = "P01",
  TIME = seq(0, 0.19, by = 0.01),
  mean_pupil = c(rep(3.2, 7), NA, NA, NA, rep(3.2, 10))
)
detect_gazepoint_blinks(pupil, min_duration = 20)
```

```
detect_gazepoint_fixations_ivt
```

Detect simple I-VT fixations from gaze samples

Description

Apply a lightweight velocity-threshold fixation detector to ordered gaze samples. This helper is intended for exploratory checks and teaching; it does not replace dedicated event-detection packages or Gazepoint's native fixation export.

Usage

```
detect_gazepoint_fixations_ivt(
  data,
  x_col,
  y_col,
  time_col,
  group_cols = NULL,
  velocity_threshold = 0.01,
  min_duration_ms = 60,
  distance_scale = 1,
  time_scale = 1
)
```

Arguments

<code>data</code>	A sample-level gaze data frame.
<code>x_col</code>	Horizontal gaze coordinate column.
<code>y_col</code>	Vertical gaze coordinate column.
<code>time_col</code>	Time column, in milliseconds by default.
<code>group_cols</code>	Optional columns defining independent recordings/trials.
<code>velocity_threshold</code>	Maximum velocity for a sample-to-sample interval to be treated as fixation-like.

min_duration_ms Minimum fixation duration.

distance_scale Multiplicative scale applied to coordinate distances.

time_scale Multiplicative scale applied to time differences.

Value

A fixation-level data frame.

detect_gazepoint_fixations_velocity
<i>Detect fixations with a velocity-threshold algorithm</i>

Description

Converts sample-level gaze coordinates into fixation events using an I-VT-style velocity threshold. The function is intended as a high-level event-table companion to [detect_gazepoint_fixations_ivt\(\)](#).

Usage

```
detect_gazepoint_fixations_velocity(  
  all_gaze,  
  id_col = "USER_ID",  
  x_col = "FPOGX",  
  y_col = "FPOGY",  
  ts_col = "TIME",  
  vmax = 10,  
  min_duration = 50,  
  group_cols = NULL,  
  time_unit = c("auto", "seconds", "milliseconds"),  
  x_scale = 1,  
  y_scale = 1,  
  return = c("events", "samples", "both"),  
  keep_single_sample = FALSE  
)
```

Arguments

all_gaze A data frame containing sample-level gaze data.

id_col Participant identifier column.

x_col Horizontal gaze-coordinate column.

y_col Vertical gaze-coordinate column.

ts_col Timestamp column.

vmax Maximum velocity classified as fixation. The threshold is in scaled coordinate units per second.

<code>min_duration</code>	Minimum fixation duration in milliseconds.
<code>group_cols</code>	Optional additional grouping columns, such as stimulus or trial identifiers.
<code>time_unit</code>	Timestamp unit. "auto" infers seconds versus milliseconds from positive timestamp differences.
<code>x_scale, y_scale</code>	Multipliers applied to coordinate differences before velocity is calculated. Use these to convert native coordinates to visual degrees when an appropriate conversion is available.
<code>return</code>	Return fixation "events", sample labels, or "both".
<code>keep_single_sample</code>	Retain single-sample events when they satisfy <code>min_duration</code> .

Value

A tibble of fixation events, a labelled sample table, or a list containing both.

Examples

```
gaze <- data.frame(
  USER_ID = "P01",
  TIME = seq(0, 0.19, by = 0.01),
  FPOGX = c(rep(0.25, 10), rep(0.75, 10)),
  FPOGY = 0.50
)
detect_gazepoint_fixations_velocity(
  gaze,
  vmax = 5,
  min_duration = 40
)
```

`diagnose_gazepoint_binocular_pupil`

Diagnose binocular pupil quality and agreement

Description

Quantifies binocular availability, monocular loss, left/right agreement, systematic offset, regression diagnostics, and temporal missingness runs before any model-based reconstruction is attempted. Source pupil columns are never modified.

Usage

```
diagnose_gazepoint_binocular_pupil(
  data,
  left_col,
  right_col,
  time_col = NULL,
```

```

group_cols = NULL,
time_unit = c("auto", "milliseconds", "seconds"),
valid_min = NULL,
valid_max = NULL,
min_pairs = 30L,
min_unique = 5L,
disagreement_mad_k = 6
)

```

Arguments

<code>data</code>	A data frame containing left and right pupil measurements.
<code>left_col, right_col</code>	Names of numeric pupil columns.
<code>time_col</code>	Optional numeric time column. When supplied, gap durations and time-order diagnostics are reported.
<code>group_cols</code>	Optional grouping columns, typically participant and/or session.
<code>time_unit</code>	Unit for <code>time_col</code> : "auto", "milliseconds", or "seconds".
<code>valid_min, valid_max</code>	Optional physiological or instrument-specific bounds. Values outside the declared bounds are treated as unavailable for diagnostics; the source data remain unchanged.
<code>min_pairs</code>	Minimum bilateral observations used to label a group as calibration-eligible.
<code>min_unique</code>	Minimum unique values required in each eye for eligibility.
<code>disagreement_mad_k</code>	Robust multiplier used to describe unusually large absolute between-eye differences. This is a diagnostic flag, not an exclusion threshold.

Details

The Bland-Altman-style limits returned here are descriptive mean-difference ± 1.96 SD limits. They do not establish interchangeability of the two measurements. `correlation` is Pearson correlation and `rank_correlation` is Spearman correlation on bilateral observations.

Value

An object of class `gp3_binocular_diagnostics` containing summary, gaps, and settings components.

See Also

[fit_gazepoint_binocular_calibration\(\)](#), [reconstruct_gazepoint_binocular_pupil\(\)](#)

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 11)
dat$pupil_left[25:28] <- NA_real_
diagnose_gazepoint_binocular_pupil(
  dat, "pupil_left", "pupil_right",
  time_col = "timestamp_ms", group_cols = "subject", min_pairs = 20
)
```

```
diagnose_gazepoint_cluster_design
```

Diagnose the design assumptions of a Gazepoint cluster-permutation workflow

Description

Provide a compact diagnostic summary for the conservative two-condition, within-subject, one-dimensional cluster-permutation workflow.

Usage

```
diagnose_gazepoint_cluster_design(
  data,
  subject_col = ".gp3_cluster_subject",
  condition_col = ".gp3_cluster_condition",
  time_col = ".gp3_cluster_time_bin",
  outcome_col = ".gp3_cluster_outcome"
)
```

Arguments

<code>data</code>	A prepared or raw long-format time-course data frame.
<code>subject_col</code>	Subject column.
<code>condition_col</code>	Condition column.
<code>time_col</code>	Time-bin column.
<code>outcome_col</code>	Outcome column.

Value

A data frame of design checks and cautious interpretations.

diagnose_gazepoint_gamm

Diagnose GAM and BAM models

Description

Run a compact diagnostics bundle for mgcv GAM/BAM models used in gp3tools workflows. The function combines convergence, basis-dimension checks, overdispersion checks, and optional DHARMA simulation-based residual diagnostics.

Usage

```
diagnose_gazepoint_gamm(
  model,
  model_name = NULL,
  check_convergence = TRUE,
  check_basis = TRUE,
  check_overdispersion = TRUE,
  use_dharma = FALSE,
  dharma_simulations = 250,
  seed = 123
)
```

Arguments

model	A fitted GAM/BAM object, a gp3tools fit object containing \$model, or a named list of fitted model objects.
model_name	Optional model label used in returned tables.
check_convergence	Logical. If TRUE, run convergence diagnostics.
check_basis	Logical. If TRUE, run mgcv::k.check() basis-dimension diagnostics when available.
check_overdispersion	Logical. If TRUE, run overdispersion diagnostics when meaningful for the model family.
use_dharma	Logical. If TRUE, try to run optional DHARMA diagnostics.
dharma_simulations	Number of DHARMA simulations.
seed	Random seed used before DHARMA simulation.

Details

The function accepts raw mgcv::gam() / mgcv::bam() model objects, gp3tools fit objects containing a \$model element, or a named list of fitted model objects.

Value

A list with overview, convergence, basis, overdispersion, DHARMA diagnostics, and settings.

diagnose_gazepoint_glmm

Diagnose GLMM, LMM, and GLM models

Description

Run a compact diagnostics bundle for model objects used in gp3tools workflows. The function combines convergence, singularity, overdispersion, and optional DHARMA simulation-based residual diagnostics.

Usage

```
diagnose_gazepoint_glmm(
  model,
  model_name = NULL,
  check_convergence = TRUE,
  check_singularity = TRUE,
  check_overdispersion = TRUE,
  use_dharma = TRUE,
  dharma_simulations = 250,
  seed = 123
)
```

Arguments

model	A fitted model object, a gp3tools fit object containing \$model, or a named list of fitted model objects.
model_name	Optional model label used in returned tables.
check_convergence	Logical. If TRUE, run convergence diagnostics.
check_singularity	Logical. If TRUE, run singularity diagnostics.
check_overdispersion	Logical. If TRUE, run overdispersion diagnostics.
use_dharma	Logical. If TRUE, try to run optional DHARMA diagnostics.
dharma_simulations	Number of DHARMA simulations.
seed	Random seed used before DHARMA simulation.

Details

The function accepts raw fitted models, gp3tools fit objects containing a \$model element, or a named list of fitted models. DHARMA diagnostics are optional and are skipped cleanly when DHARMA is not installed.

Value

A list with overview, convergence, singularity, overdispersion, DHARMA diagnostics, and settings.

downsample_gazepoint_pupil

Downsample pupil data by integer-factor aggregation

Description

Aggregates consecutive samples within independent sequences. The default mean method is safer than simple decimation because it reduces aliasing and preserves a representative timestamp for each bin.

Usage

```
downsample_gazepoint_pupil(
  master_df,
  factor = 2,
  pupil_cols = NULL,
  id_col = "USER_ID",
  group_cols = NULL,
  ts_col = "TIME",
  method = c("mean", "first"),
  keep_bin = FALSE
)
```

Arguments

master_df	A sample-level data frame.
factor	Positive integer downsampling factor.
pupil_cols	Pupil columns to aggregate. When NULL, common pupil columns are detected automatically.
id_col	Participant identifier.
group_cols	Optional additional independent-sequence columns.
ts_col	Optional timestamp column. Its bin value is the finite mean.
method	"mean" aggregation or "first"-sample decimation.
keep_bin	Keep the generated downsample-bin identifier.

Value

A downsampled data frame.

Examples

```
pupil <- data.frame(  
  USER_ID = "P01",  
  TIME = seq(0, 0.09, by = 0.01),  
  mean_pupil = seq(3, 4, length.out = 10)  
)  
downsample_gazepoint_pupil(pupil, factor = 2)
```

`estimate_gazepoint_cluster_offset`*Guardrail for exact cluster-offset estimation*

Description

This function intentionally fails safely. Cluster-permutation results should not be used as exact effect-offset estimates.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

`estimate_gazepoint_cluster_onset`*Guardrail for exact cluster-onset estimation*

Description

This function intentionally fails safely. Cluster-permutation results should not be used as exact effect-onset estimates.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

```
estimate_gazepoint_divergence_point
```

Estimate a bootstrapped divergence point between two Gazepoint time courses

Description

Estimate the earliest time point at which two condition-level time courses reliably diverge. The helper computes observed condition curves, bootstraps the condition difference, identifies the first time point where the bootstrap confidence interval excludes the null value for a requested number of consecutive time points, and returns a bootstrap uncertainty interval for the divergence onset.

Usage

```
estimate_gazepoint_divergence_point(
  data,
  outcome_col,
  time_col,
  condition_col,
  participant_col = NULL,
  trial_col = NULL,
  comparison = NULL,
  bootstrap_unit = c("participant", "trial", "row"),
  summary_function = c("mean", "median"),
  n_boot = 1000L,
  ci = 0.95,
  consecutive_points = 1L,
  null_value = 0,
  min_abs_difference = 0,
  direction = c("two_sided", "positive", "negative"),
  seed = NULL,
  keep_bootstrap = TRUE,
  name = "gazepoint_divergence_point"
)
```

Arguments

<code>data</code>	A data frame containing time-course observations.
<code>outcome_col</code>	Outcome column, for example pupil size, fixation probability, gaze proportion, or AOI time-course value.
<code>time_col</code>	Time column.
<code>condition_col</code>	Condition column. Exactly two conditions are compared unless comparison is supplied.

participant_col	Optional participant column used for participant-level bootstrap resampling.
trial_col	Optional trial column used for trial-level bootstrap resampling.
comparison	Optional character vector of two condition values. The estimated difference is $\text{comparison}[2] - \text{comparison}[1]$.
bootstrap_unit	Resampling unit. Options are "participant", "trial", and "row".
summary_function	Function used to summarise observations within condition-by-time cells. Options are "mean" and "median".
n_boot	Number of bootstrap resamples.
ci	Confidence level for bootstrap intervals.
consecutive_points	Number of consecutive time points required before declaring divergence.
null_value	Null difference value. Default is 0.
min_abs_difference	Optional minimum absolute observed difference required at a time point.
direction	Direction of divergence. "two_sided" checks whether the bootstrap interval excludes null_value in either direction. "positive" checks whether $\text{comparison}[2] > \text{comparison}[1]$. "negative" checks whether $\text{comparison}[2] < \text{comparison}[1]$.
seed	Optional random seed for reproducible bootstrap resampling.
keep_bootstrap	Logical. If TRUE, return bootstrap differences for each time point.
name	Character label stored in the returned object.

Details

This helper complements cluster-permutation analysis. Cluster permutation asks where a reliable time window exists; divergence-point analysis asks when the condition difference first emerges.

Value

A list with class `gp3_divergence_point_analysis`.

`export_gazepoint_cluster_results`

Export Gazepoint cluster-permutation results

Description

Export cluster tables, optional null distributions, settings, and cautious report text to a folder of CSV/TXT files.

Usage

```
export_gazepoint_cluster_results(result, outdir, overwrite = FALSE)
```

Arguments

result	A result object returned by run_gazepoint_cluster_permutation().
outdir	Output directory.
overwrite	Should an existing directory be reused?

Value

A data frame listing written files.

export_gazepoint_heatmap_png
<i>Export a Gazepoint heatmap plot to PNG</i>

Description

export_gazepoint_heatmap_png() saves a ggplot heatmap to a PNG file.

Usage

```
export_gazepoint_heatmap_png(  
  plot,  
  filename,  
  width = 8,  
  height = 5,  
  units = "in",  
  dpi = 300,  
  create_dir = TRUE,  
  ...  
)
```

Arguments

plot	A ggplot object, usually returned by plot_gazepoint_heatmap() or plot_gazepoint_heatmap_overlap().
filename	Output file path.
width, height	Plot size passed to ggplot2::ggsave().
units	Units passed to ggplot2::ggsave().
dpi	Resolution passed to ggplot2::ggsave().
create_dir	Logical. If TRUE, the output directory is created when needed.
...	Additional arguments passed to ggplot2::ggsave().

Value

Invisibly returns the output path.

Examples

```

gaze <- data.frame(
  x = c(0.20, 0.25, 0.70),
  y = c(0.30, 0.35, 0.60)
)

p <- plot_gazepoint_heatmap(
  gaze,
  x_col = "x",
  y_col = "y",
  bins = 10
)

out <- tempfile(fileext = ".png")
export_gazepoint_heatmap_png(p, out, width = 4, height = 3)

```

export_gazepoint_master_audit

Export a Gazepoint master table, audit tables, and validation tables

Description

Exports a Gazepoint master sample-level table together with the output of [audit_gazepoint_master\(\)](#) and, optionally, [validate_gazepoint_master\(\)](#). This function is useful after creating a master table with [as_gazepoint_master\(\)](#) or [create_gazepoint_master\(\)](#).

Usage

```

export_gazepoint_master_audit(
  master,
  audit = NULL,
  validation = NULL,
  output_dir = ".",
  prefix = "gazepoint",
  export_master = TRUE,
  export_audit = TRUE,
  export_validation = TRUE,
  overwrite = TRUE,
  na = ""
)

```

Arguments

master	A Gazepoint master sample-level table.
audit	Optional audit list returned by audit_gazepoint_master() . If NULL, the audit is created automatically.

validation	Optional validation list returned by <code>validate_gazepoint_master()</code> . If NULL and <code>export_validation = TRUE</code> , validation is created automatically.
output_dir	Directory where CSV files should be written.
prefix	File prefix used for all exported CSV files.
export_master	Logical. If TRUE, exports the full master table.
export_audit	Logical. If TRUE, exports audit tables.
export_validation	Logical. If TRUE, exports validation tables.
overwrite	Logical. If FALSE, the function aborts when any target file already exists.
na	String used for missing values in CSV output.

Value

A tibble listing the exported files, table names, and dimensions.

Examples

```
## Not run:
master <- create_gazepoint_master(
  gaze_data = results$all_gaze,
  screen_width_px = 1920,
  screen_height_px = 1080
)

audit <- audit_gazepoint_master(master)
validation <- validate_gazepoint_master(master)

exported <- export_gazepoint_master_audit(
  master = master,
  audit = audit,
  validation = validation,
  output_dir = "gazepoint_outputs",
  prefix = "study1"
)

exported

## End(Not run)
```

export_gazepoint_mne_cluster_input

Export Gazepoint time-course data for MNE-style cluster workflows

Description

Write a conservative long-format CSV file and README for continuation in an external Python/MNE workflow. This helper prepares data only; it does not run MNE, construct adjacency matrices, validate exchangeability, or validate the external analysis.

Usage

```
export_gazepoint_mne_cluster_input(
  data,
  outdir,
  subject_col = ".gp3_cluster_subject",
  condition_col = ".gp3_cluster_condition",
  time_col = ".gp3_cluster_time_bin",
  outcome_col = ".gp3_cluster_outcome",
  overwrite = FALSE
)
```

Arguments

data	A prepared or raw long-format time-course data frame.
outdir	Output directory.
subject_col	Subject column.
condition_col	Condition column.
time_col	Time-bin column.
outcome_col	Outcome column.
overwrite	Should an existing directory be reused?

Value

A data frame listing written files.

```
export_gazepoint_model_tables
```

Export manuscript-ready model tables

Description

Export model-summary tables and optional estimated marginal means tables to CSV files. The function is designed for objects returned by `tidy_gazepoint_model_summary()` and `summarise_gazepoint_emmeans()`.

Usage

```
export_gazepoint_model_tables(
  model_summary = NULL,
  emmeans_summary = NULL,
  output_dir,
  prefix = "gazepoint_model",
  overwrite = TRUE,
  include_diagnostics = TRUE
)
```

Arguments

model_summary	Optional object returned by tidy_gazepoint_model_summary().
emmeans_summary	Optional object returned by summarise_gazepoint_emmeans().
output_dir	Output directory.
prefix	File-name prefix.
overwrite	Logical. If FALSE, existing output files cause an error.
include_diagnostics	Logical. If TRUE, export available diagnostic component tables from model_summary.

Value

A tibble indexing the written files.

export_gazepoint_permuco_cluster_input	<i>Export Gazepoint time-course data for permuco-style cluster workflows</i>
--	--

Description

Write a conservative long-format CSV file and README for continuation in an external R/permuco workflow. This helper prepares data only; it does not run permuco or validate the external model specification.

Usage

```
export_gazepoint_permuco_cluster_input(
  data,
  outdir,
  subject_col = ".gp3_cluster_subject",
  condition_col = ".gp3_cluster_condition",
  time_col = ".gp3_cluster_time_bin",
  outcome_col = ".gp3_cluster_outcome",
  overwrite = FALSE
)
```

Arguments

data	A prepared or raw long-format time-course data frame.
outdir	Output directory.
subject_col	Subject column.
condition_col	Condition column.
time_col	Time-bin column.
outcome_col	Outcome column.
overwrite	Should an existing directory be reused?

Value

A data frame listing written files.

```
export_gazepoint_permutes_cluster_input
```

Export Gazepoint time-course data for permutes-style cluster work-flows

Description

Write a conservative long-format CSV file and README for continuation in an external R/permutes workflow. This helper prepares data only; it does not run permutes or validate the external model specification.

Usage

```
export_gazepoint_permutes_cluster_input(  
  data,  
  outdir,  
  subject_col = ".gp3_cluster_subject",  
  condition_col = ".gp3_cluster_condition",  
  time_col = ".gp3_cluster_time_bin",  
  outcome_col = ".gp3_cluster_outcome",  
  overwrite = FALSE  
)
```

Arguments

<code>data</code>	A prepared or raw long-format time-course data frame.
<code>outdir</code>	Output directory.
<code>subject_col</code>	Subject column.
<code>condition_col</code>	Condition column.
<code>time_col</code>	Time-bin column.
<code>outcome_col</code>	Outcome column.
<code>overwrite</code>	Should an existing directory be reused?

Value

A data frame listing written files.

`export_gazepoint_tables`*Export Gazepoint analysis tables to CSV files*

Description

Writes a named list of analysis tables to CSV files in an output folder.

Usage

```
export_gazepoint_tables(  
  tables,  
  output_dir,  
  prefix = NULL,  
  overwrite = TRUE,  
  na = ""  
)
```

Arguments

<code>tables</code>	A named list of data frames or tibbles.
<code>output_dir</code>	Folder where CSV files should be written.
<code>prefix</code>	Optional filename prefix.
<code>overwrite</code>	Logical. If FALSE, the function stops when a target file already exists.
<code>na</code>	Value used for missing values in the exported CSV files.

Value

A tibble with table names and written file paths.

`export_gazepoint_to_bids`*Export Gazepoint data to a lightweight BIDS-style folder*

Description

Write a conservative BIDS-style eye-tracking folder from a Gazepoint-style data frame. This helper creates standard-looking TSV and JSON sidecar files for sharing and inspection, but it does not claim full validation against a specific evolving BIDS eye-tracking validator.

Usage

```

export_gazepoint_to_bids(
  data,
  outdir,
  subject_col,
  task = "gazepoint",
  session = NULL,
  time_col,
  x_col,
  y_col,
  pupil_col = NULL,
  trial_col = NULL,
  aoi_col = NULL,
  overwrite = FALSE
)

```

Arguments

data	A gaze-sample data frame.
outdir	Output directory.
subject_col	Subject column.
task	Task label used in file names.
session	Optional session label.
time_col	Time column.
x_col	Horizontal gaze coordinate column.
y_col	Vertical gaze coordinate column.
pupil_col	Optional pupil column.
trial_col	Optional trial column.
aoi_col	Optional AOI column.
overwrite	Should an existing output directory be reused?

Value

A data frame listing written files.

```

extract_gazepoint_representative_scanpaths
      Extract representative scanpaths

```

Description

Select representative observed scanpaths by minimizing mean distance to other members of the same fitted cluster.

Usage

```
extract_gazepoint_representative_scanpaths(x, n_per_cluster = 1L)
```

Arguments

x An object returned by `cluster_gazepoint_scanpaths()`.
n_per_cluster Number of representatives to return per cluster.

Value

A data frame containing cluster, representative rank, sequence identifier, mean within-cluster distance, cluster size, and PAM medoid status.

Examples

```
d <- matrix(
  c(
    0, 0.1, 1, 1,
    0.1, 0, 1, 1,
    1, 1, 0, 0.1,
    1, 1, 0.1, 0
  ),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(LETTERS[1:4], LETTERS[1:4])
)

fit <- cluster_gazepoint_scanpaths(d, k = 2)
extract_gazepoint_representative_scanpaths(fit)
```

```
filter_gazepoint_cnn_uncertainty
```

Apply uncertainty filtering to Bayesian CNN or webcam gaze outputs

Description

Provides a lightweight post-processing helper for externally generated webcam/CNN gaze predictions. The function does not train a CNN. It filters or down-weights frame-level gaze estimates using an uncertainty column.

Usage

```
filter_gazepoint_cnn_uncertainty(
  data,
  x,
  y,
  uncertainty = NULL,
  max_uncertainty = NULL,
```

```

    weight_output = "cnn_uncertainty_weight",
    valid_output = "cnn_valid_frame"
  )

```

Arguments

<code>data</code>	A data frame containing frame-level gaze predictions.
<code>x</code>	Predicted x-coordinate column.
<code>y</code>	Predicted y-coordinate column.
<code>uncertainty</code>	Optional uncertainty column.
<code>max_uncertainty</code>	Optional maximum allowed uncertainty.
<code>weight_output</code>	Name of the output weight column.
<code>valid_output</code>	Name of the output validity column.

Value

Data frame with uncertainty weights and validity flags.

```
fit_gazepoint_aoi_brms
```

Fit an optional Bayesian AOI model with brms

Description

Prepare and optionally fit a Bayesian AOI model using **brms**. The dependency is optional: the function checks for **brms** only when `dry_run = FALSE`. Tests and lightweight workflows can use `dry_run = TRUE` to inspect the formula and data without running Stan.

Usage

```

fit_gazepoint_aoi_brms(
  data,
  response,
  predictors,
  subject_col = NULL,
  family = "bernoulli",
  prior = NULL,
  dry_run = TRUE,
  ...
)

```

Arguments

data	A data frame.
response	Response column.
predictors	Character vector of fixed-effect predictors.
subject_col	Optional grouping column for a random intercept.
family	brms family specification as a character string.
prior	Optional brms prior object.
dry_run	If TRUE, return the prepared call components without fitting.
...	Additional arguments passed to <code>brms::brm()</code> when fitting.

Value

A dry-run specification list or a `brmsfit` object.

fit_gazepoint_aoi_gamm

Fit AOI time-course GAMMs

Description

Fit binomial GAMMs for AOI target-looking time courses prepared by `prepare_gazepoint_aoi_gamm_data()`. The model uses target-looking successes and failures over time and can include condition effects, condition-specific smooths, and subject random-effect smooths.

Usage

```
fit_gazepoint_aoi_gamm(
  data,
  include_condition = TRUE,
  condition_smooths = TRUE,
  random_subject = TRUE,
  random_subject_time = FALSE,
  time_k = 10,
  subject_time_k = 5,
  family = stats::binomial(),
  method = "fREML",
  discrete = FALSE,
  select = FALSE,
  drop_non_ok = TRUE,
  min_rows = 10,
  min_subjects = 2,
  min_time_bins = 4,
  ...
)
```

Arguments

<code>data</code>	A data frame returned by <code>prepare_gazepoint_aoi_gamm_data()</code> .
<code>include_condition</code>	Logical. If TRUE, include condition as a parametric fixed effect when two or more conditions are available.
<code>condition_smooths</code>	Logical. If TRUE, fit condition-specific time smooths when two or more conditions are available.
<code>random_subject</code>	Logical. If TRUE, include a subject random-effect smooth.
<code>random_subject_time</code>	Logical. If TRUE, include subject-specific factor-smooth time deviations. This can be useful for repeated-measures time-course data but may be too heavy for very small datasets.
<code>time_k</code>	Basis dimension for the main time smooth.
<code>subject_time_k</code>	Basis dimension for subject-specific factor-smooth time deviations.
<code>family</code>	Model family. Defaults to <code>stats::binomial()</code> .
<code>method</code>	Smoothing-parameter estimation method passed to <code>mgcv::bam()</code> .
<code>discrete</code>	Logical passed to <code>mgcv::bam()</code> .
<code>select</code>	Logical passed to <code>mgcv::bam()</code> .
<code>drop_non_ok</code>	Logical. If TRUE, keep only rows with <code>.gp3_aoi_gamm_status == "ok"</code> before fitting.
<code>min_rows</code>	Minimum number of rows required for model fitting.
<code>min_subjects</code>	Minimum number of subjects required for model fitting.
<code>min_time_bins</code>	Minimum number of time bins required for model fitting.
<code>...</code>	Additional arguments passed to <code>mgcv::bam()</code> .

Details

This function is intended for AOI time-course modelling. It is separate from confirmatory AOI-window GLMMs and from cluster-based permutation tests.

Value

A list containing the fitted model, formula, model status, diagnostics, parametric table, smooth table, and settings.

```
fit_gazepoint_aoi_model_sensitivity
```

Fit AOI-window model-family sensitivity checks

Description

Fit a compact set of sensitivity models for AOI-window outcomes. The main model is a binomial GLMM. Additional checks can include an empirical-logit LMM, a weighted proportion LMM, and a fixed-effects quasibinomial GLM.

Usage

```
fit_gazepoint_aoi_model_sensitivity(
  data,
  success_col = "aoi_glmm_success",
  failure_col = "aoi_glmm_failure",
  denominator_col = "aoi_glmm_denominator",
  proportion_col = "aoi_glmm_prop",
  subject_col = "aoi_glmm_subject",
  condition_col = "aoi_glmm_condition",
  window_col = "aoi_glmm_window",
  model_types = c("binomial_glmm", "empirical_logit_lmm", "proportion_lmm",
    "quasibinomial_glm"),
  include_condition = TRUE,
  include_window = TRUE,
  include_interaction = TRUE,
  random_intercept = TRUE,
  optimizer = "bobyqa",
  maxfun = 2e+05,
  nAGQ = 0,
  empirical_logit_correction = 0.5,
  drop_missing = TRUE
)
```

Arguments

<code>data</code>	AOI GLMM data returned by <code>prepare_gazepoint_aoi_glmm_data()</code> .
<code>success_col</code>	Success-count column.
<code>failure_col</code>	Failure-count column.
<code>denominator_col</code>	Denominator column.
<code>proportion_col</code>	Proportion column.
<code>subject_col</code>	Subject column.
<code>condition_col</code>	Condition column.
<code>window_col</code>	Window column.

model_types	Character vector of model types. Supported values are "binomial_glmm", "empirical_logit_lmm", "proportion_lmm", and "quasibinomial_glm".
include_condition	Logical. Include condition fixed effect when possible.
include_window	Logical. Include window fixed effect when possible.
include_interaction	Logical. Include condition-by-window interaction when both condition and window are included.
random_intercept	Logical. Include subject random intercept in mixed sensitivity models.
optimizer	Optimizer for lme4 mixed models.
maxfun	Maximum optimizer evaluations.
nAGQ	Number of adaptive Gauss-Hermite quadrature points for the binomial GLMM.
empirical_logit_correction	Small correction added to success and failure counts for empirical-logit models.
drop_missing	Logical. Drop rows with missing model variables.

Value

A list containing fitted models, formulas, comparison table, fixed effects table, settings, and status information.

fit_gazepoint_aoi_window_glmm
<i>Fit an AOI-window binomial GLMM</i>

Description

Fit a confirmatory AOI-window mixed-effects logistic regression from data prepared by prepare_gazepoint_aoi_glmm_data

Usage

```
fit_gazepoint_aoi_window_glmm(  
  data,  
  success_col = "aoi_glmm_success",  
  failure_col = "aoi_glmm_failure",  
  subject_col = "aoi_glmm_subject",  
  condition_col = "aoi_glmm_condition",  
  window_col = "aoi_glmm_window",  
  include_condition = TRUE,  
  include_window = TRUE,  
  include_interaction = TRUE,  
  random_intercept = TRUE,  
  random_window_slopes = FALSE,  
  fallback_on_singular = TRUE,
```

```

optimizer = "bobyqa",
maxfun = 2e+05,
nAGQ = 0,
drop_missing = TRUE
)

```

Arguments

<code>data</code>	AOI GLMM data returned by <code>prepare_gazepoint_aoi_glmm_data()</code> .
<code>success_col</code>	Success-count column.
<code>failure_col</code>	Failure-count column.
<code>subject_col</code>	Subject factor/column.
<code>condition_col</code>	Condition factor/column.
<code>window_col</code>	AOI-window factor/column.
<code>include_condition</code>	Logical. Include condition fixed effects when at least two conditions are available.
<code>include_window</code>	Logical. Include window fixed effects when at least two windows are available.
<code>include_interaction</code>	Logical. Include condition-by-window interaction when both condition and window fixed effects are included.
<code>random_intercept</code>	Logical. Include subject random intercept.
<code>random_window_slopes</code>	Logical. Attempt subject-level random slopes for AOI window.
<code>fallback_on_singular</code>	Logical. If TRUE, fall back to a simpler random intercept model when a random-slope model is singular or fails.
<code>optimizer</code>	Optimizer passed to <code>lme4::glmerControl()</code> .
<code>maxfun</code>	Maximum optimizer evaluations.
<code>nAGQ</code>	Number of adaptive Gauss-Hermite quadrature points.
<code>drop_missing</code>	Logical. Drop rows with missing model variables before fitting.

Value

A list with fitted model, attempted model, formulas, comparison table, settings, status fields, and model data.

fit_gazepoint_binocular_calibration

Fit audited cross-eye pupil calibration models

Description

Fits separate left-from-right and right-from-left linear calibration models using only bilateral observations. Optional fallback levels permit transparent participant/session-to-pooled fallback without silently mixing calibration scopes.

Usage

```
fit_gazepoint_binocular_calibration(
  data,
  left_col,
  right_col,
  group_cols = NULL,
  fallback_group_cols = NULL,
  valid_min = NULL,
  valid_max = NULL,
  min_pairs = 30L,
  min_unique = 5L,
  min_r2 = NULL,
  allow_negative_slope = FALSE,
  max_abs_slope = NULL
)
```

Arguments

data	A data frame containing pupil channels.
left_col, right_col	Numeric pupil columns.
group_cols	Primary calibration grouping columns.
fallback_group_cols	Optional list of fallback grouping specifications. By default a pooled fallback is added when group_cols is non-empty. Use <code>list()</code> to disable fallback.
valid_min, valid_max	Optional measurement bounds.
min_pairs	Minimum bilateral training observations per model.
min_unique	Minimum unique predictor and outcome values.
min_r2	Optional minimum in-sample R-squared. NULL reports R-squared without using it as a gate.
allow_negative_slope	Whether a non-positive cross-eye slope can be eligible. The conservative default is FALSE.
max_abs_slope	Optional absolute slope ceiling; use NULL for none.

Details

R-squared is a diagnostic, not evidence that reconstructed values are measured observations. Calibration eligibility is explicit and can be reviewed before reconstruction.

Value

An object of class `gp3_binocular_calibration` with a flat `models` table, per-level model tables, and settings.

References

Ong J, He W, Maglanque P, Jiang X, Gillman LM, Vergis A, Hardy K (2025). A Preprocessing Pipeline for Pupillometry Signal from Multimodal iMotion Data. *Sensors*, 25(15), 4737. doi:10.3390/s25154737

See Also

`validate_gazepoint_binocular_reconstruction()`, `reconstruct_gazepoint_binocular_pupil()`

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 12)
fit_gazepoint_binocular_calibration(
  dat, "pupil_left", "pupil_right", group_cols = "subject", min_pairs = 20
)
```

```
fit_gazepoint_brms_model
```

Fit an optional brms model for Gazepoint-derived data

Description

Fits a Bayesian model using brms when brms is installed. brms is treated as an optional external backend; gp3tools does not import or require it.

Usage

```
fit_gazepoint_brms_model(
  data,
  formula,
  family = "gaussian",
  prior = NULL,
  chains = 4,
  iter = 2000,
  warmup = floor(iter/2),
  cores = 1,
  backend = NULL,
```

```
    ...
  )
```

Arguments

<code>data</code>	A data frame.
<code>formula</code>	A model formula.
<code>family</code>	brms family specification as a character string or brms family object.
<code>prior</code>	Optional brms prior specification.
<code>chains</code>	Number of MCMC chains.
<code>iter</code>	Total iterations per chain.
<code>warmup</code>	Warmup iterations per chain.
<code>cores</code>	Number of cores.
<code>backend</code>	Optional brms backend, e.g. "cmdstanr".
<code>...</code>	Additional arguments passed to <code>brms::brm()</code> .

Value

A fitted brms model.

```
fit_gazepoint_face_window_lmm
```

Fit a facial-behaviour window mixed or linear model

Description

Fits an explicit model to a face-window or multimodal analysis table. If `random_effects` is supplied, the model is fitted with `lme4::lmer()`; otherwise `stats::lm()` is used. The helper is a modelling convenience wrapper and does not interpret facial-behaviour variables as emotions.

Usage

```
fit_gazepoint_face_window_lmm(
  data,
  outcome,
  predictors,
  covariates = NULL,
  random_effects = NULL,
  na_action = c("na.omit", "na.exclude"),
  REML = FALSE
)
```

Arguments

<code>data</code>	A face-window or multimodal data frame.
<code>outcome</code>	Outcome column name.
<code>predictors</code>	Character vector of fixed-effect predictor columns.
<code>covariates</code>	Optional character vector of covariate columns.
<code>random_effects</code>	Optional random-effects formula component, for example "(1 participant_id)".
<code>na_action</code>	Missing-data handling. One of "na.omit" or "na.exclude".
<code>REML</code>	Passed to <code>lme4::lmer()</code> when a mixed model is fitted.

Value

A list with model, formula, data, and settings. The object has class `gp3_face_window_lmm`.

Examples

```
dat <- data.frame(
  participant_id = c("P001", "P002", "P003"),
  AU12_r_mean = c(0.1, 0.2, 0.3),
  rating = c(3, 4, 5)
)

fit_gazepoint_face_window_lmm(
  dat,
  outcome = "rating",
  predictors = "AU12_r_mean"
)
```

<code>fit_gazepoint_gca</code>	<i>Fit a Gazepoint Growth Curve Analysis mixed model</i>
--------------------------------	--

Description

Fit a Growth Curve Analysis (GCA) mixed model to prepared pupil time-course data. The function first attempts a random-intercept plus random-time-slopes model and, if the model fails or is singular, falls back to a random-intercept model.

Usage

```
fit_gazepoint_gca(
  data,
  outcome_col = "gca_pupil",
  subject_col = "subject",
  condition_col = "condition",
  time_terms = NULL,
  degree = NULL,
  weights_col = "gca_weight",
```

```

    use_weights = TRUE,
    random_slopes = TRUE,
    fallback_on_singular = TRUE,
    REML = FALSE,
    optimizer = "bobyqa",
    maxfun = 2e+05,
    drop_missing = TRUE
  )

```

Arguments

<code>data</code>	A data frame created by <code>prepare_gazepoint_gca_data()</code> .
<code>outcome_col</code>	Name of the GCA outcome column.
<code>subject_col</code>	Name of the subject column.
<code>condition_col</code>	Name of the condition column.
<code>time_terms</code>	Optional character vector of polynomial time-term columns. If NULL, terms named <code>time_poly_1</code> , <code>time_poly_2</code> , ... are detected.
<code>degree</code>	Optional number of polynomial terms to use. If supplied and <code>time_terms = NULL</code> , the function uses <code>time_poly_1</code> through <code>time_poly_degree</code> .
<code>weights_col</code>	Optional weights column. Use NULL for unweighted models.
<code>use_weights</code>	Logical. If TRUE, uses <code>weights_col</code> when available.
<code>random_slopes</code>	Logical. If TRUE, first attempts random slopes for all polynomial time terms.
<code>fallback_on_singular</code>	Logical. If TRUE, falls back to a random-intercept model when the random-slope model is singular.
<code>REML</code>	Logical passed to <code>lme4::lmer()</code> .
<code>optimizer</code>	Optimizer passed to <code>lme4::lmerControl()</code> .
<code>maxfun</code>	Maximum optimizer function evaluations.
<code>drop_missing</code>	Logical. If TRUE, rows with missing model variables are removed before fitting.

Value

A list of class `gp3_gca_model` containing the fitted model, attempted and final formulas, model comparison information, settings, and status.

```
fit_gazepoint_multimodal_response_model
```

Fit a multimodal response model

Description

Fits an explicit response model using multimodal predictors. Linear models, generalised linear models, and mixed-effects models are supported depending on `family` and `random_effects`. The helper prepares and fits the model only; it does not make causal, diagnostic, or emotion-inference claims.

Usage

```
fit_gazepoint_multimodal_response_model(
  data,
  outcome,
  predictors,
  covariates = NULL,
  random_effects = NULL,
  family = NULL,
  na_action = c("na.omit", "na.exclude"),
  REML = FALSE,
  ...
)
```

Arguments

<code>data</code>	A multimodal analysis data frame, usually returned by <code>prepare_gazepoint_multimodal_data()</code> .
<code>outcome</code>	Outcome column name.
<code>predictors</code>	Character vector of fixed-effect predictor columns.
<code>covariates</code>	Optional character vector of covariate columns.
<code>random_effects</code>	Optional random-effects formula component, for example <code>"(1 participant_id)"</code> .
<code>family</code>	Optional model family. If <code>NULL</code> , <code>stats::lm()</code> or <code>lme4::lmer()</code> is used. If supplied, <code>stats::glm()</code> or <code>lme4::glmer()</code> is used.
<code>na_action</code>	Missing-data handling. One of <code>"na.omit"</code> or <code>"na.exclude"</code> .
<code>REML</code>	Passed to <code>lme4::lmer()</code> when a linear mixed model is fitted.
<code>...</code>	Additional arguments passed to the model-fitting function.

Value

A list with model, formula, data, and settings. The object has class `gp3_multimodal_response_model`.

Examples

```
dat <- data.frame(
  participant_id = c("P001", "P002", "P003"),
  AU12_r_mean = c(0.1, 0.2, 0.3),
  dwell_time = c(1.2, 1.4, 1.1),
  rating = c(3, 4, 5)
)

fit_gazepoint_multimodal_response_model(
  dat,
  outcome = "rating",
  predictors = c("AU12_r_mean", "dwell_time")
)
```

fit_gazepoint_pupil_gamm

Fit a Gazepoint pupil GAMM

Description

Fit a generalized additive mixed model for binned pupil time-course data using `mgcv::bam()`. The function is designed to work with data prepared by `prepare_gazepoint_pupil_gamm_data()`.

Usage

```
fit_gazepoint_pupil_gamm(
  data,
  pupil_col = "mean_pupil",
  time_col = "time_bin_center_ms",
  subject_col = "subject",
  condition_col = "condition",
  n_time_basis = 10,
  use_condition_smooths = TRUE,
  include_subject_random_effect = TRUE,
  family = c("gaussian", "scat"),
  method = "fREML",
  discrete = TRUE,
  rho = NULL,
  ar_start_col = "AR.start",
  weights_col = NULL,
  drop_missing = TRUE
)
```

Arguments

<code>data</code>	A binned pupil time-course data frame.
<code>pupil_col</code>	Name of the dependent pupil column.
<code>time_col</code>	Name of the time-bin centre column.
<code>subject_col</code>	Name of the subject column.
<code>condition_col</code>	Name of the condition column.
<code>n_time_basis</code>	Basis dimension for smooth time terms.
<code>use_condition_smooths</code>	Logical. If TRUE, condition-specific smooths are added when the condition column has more than one level.
<code>include_subject_random_effect</code>	Logical. If TRUE, adds a subject random-effect smooth.
<code>family</code>	Model family. Use "gaussian" for the default Gaussian model or "scat" for mgcv's scaled-t family.
<code>method</code>	Smoothing-parameter estimation method passed to <code>mgcv::bam()</code> .

discrete	Logical passed to <code>mgcv::bam()</code> .
rho	Optional AR(1) correlation parameter passed to <code>mgcv::bam()</code> .
ar_start_col	Optional AR-start column. If present and rho is not NULL, it is passed to <code>mgcv::bam()</code> as <code>AR.start</code> .
weights_col	Optional weights column.
drop_missing	Logical. If TRUE, rows with missing model variables are removed before fitting.

Value

A list of class `gp3_pupil_gamm` containing the fitted model, formula, data, settings, and status information.

```
fit_gazepoint_pupil_pfe_gamm
```

Fit a gaze-position-adjusted pupil GAMM sensitivity model

Description

Fit and compare a main pupil GAMM with a gaze-position-adjusted sensitivity model. The adjusted model adds a two-dimensional tensor-product smooth over mean gaze x/y position using `te(mean_x, mean_y)`.

Usage

```
fit_gazepoint_pupil_pfe_gamm(
  data,
  pupil_col = "mean_pupil",
  time_col = "time_bin_center_ms",
  subject_col = "subject",
  condition_col = "condition",
  x_col = "mean_x",
  y_col = "mean_y",
  n_time_basis = 10,
  n_position_basis = 8,
  use_condition_smooths = TRUE,
  include_subject_random_effect = TRUE,
  family = c("gaussian", "scat"),
  method = "fREML",
  discrete = TRUE,
  rho = NULL,
  ar_start_col = "AR.start",
  weights_col = NULL,
  drop_missing = TRUE
)
```

Arguments

<code>data</code>	A binned pupil time-course data frame, usually created by <code>prepare_gazepoint_pupil_gamm_data()</code> .
<code>pupil_col</code>	Name of the dependent pupil column.
<code>time_col</code>	Name of the time-bin centre column.
<code>subject_col</code>	Name of the subject column.
<code>condition_col</code>	Name of the condition column.
<code>x_col</code>	Name of the mean gaze x-position column.
<code>y_col</code>	Name of the mean gaze y-position column.
<code>n_time_basis</code>	Basis dimension for time smooths.
<code>n_position_basis</code>	Basis dimension for gaze-position smooths.
<code>use_condition_smooths</code>	Logical. If TRUE, condition-specific time smooths are used when multiple conditions are present.
<code>include_subject_random_effect</code>	Logical. If TRUE, adds a subject random-effect smooth.
<code>family</code>	Model family. Use "gaussian" or "scat".
<code>method</code>	Smoothing-parameter estimation method passed to <code>mgcv::bam()</code> .
<code>discrete</code>	Logical passed to <code>mgcv::bam()</code> .
<code>rho</code>	Optional AR(1) correlation parameter passed to <code>mgcv::bam()</code> .
<code>ar_start_col</code>	Optional AR-start column.
<code>weights_col</code>	Optional weights column.
<code>drop_missing</code>	Logical. If TRUE, rows with missing model variables are removed before fitting.

Value

A list of class `gp3_pupil_pfe_gamm` containing the main model, the gaze-position-adjusted model, formulas, comparison table, settings, and status information.

```
fit_gazepoint_pupil_window_lmm
```

Fit confirmatory pupil-window linear mixed models

Description

Fit the main confirmatory trial/window-level pupil model from data prepared with `prepare_gazepoint_pupil_window_model()`. The default model is a linear mixed model with pupil outcome as the continuous dependent variable, condition and/or window fixed effects when available, and a subject random intercept when feasible.

Usage

```
fit_gazepoint_pupil_window_lmm(
  data,
  formula = NULL,
  outcome_col = "pupil_model_outcome",
  subject_col = "pupil_model_subject",
  condition_col = "pupil_model_condition",
  window_col = "pupil_model_window",
  weights_col = "pupil_model_weight",
  use_weights = FALSE,
  include_condition = TRUE,
  include_window = TRUE,
  include_interaction = TRUE,
  random_intercept = TRUE,
  random_window_slopes = FALSE,
  fallback_on_singular = TRUE,
  REML = FALSE,
  optimizer = "bobyqa",
  maxfun = 2e+05,
  drop_missing = TRUE,
  ...
)
```

Arguments

<code>data</code>	Pupil-window model data, usually produced by <code>prepare_gazepoint_pupil_window_model_data()</code> .
<code>formula</code>	Optional model formula. If <code>NULL</code> , a formula is constructed automatically.
<code>outcome_col</code>	Outcome column.
<code>subject_col</code>	Subject column.
<code>condition_col</code>	Condition column.
<code>window_col</code>	Window column.
<code>weights_col</code>	Optional weights column.
<code>use_weights</code>	Logical. If <code>TRUE</code> , use <code>weights_col</code> as model weights.
<code>include_condition</code>	Logical. Include condition fixed effects when more than one condition level is available.
<code>include_window</code>	Logical. Include window fixed effects when more than one window level is available.
<code>include_interaction</code>	Logical. Include the condition-by-window interaction when both condition and window are used.
<code>random_intercept</code>	Logical. Include a subject random intercept when feasible.
<code>random_window_slopes</code>	Logical. Attempt subject-level random window slopes when feasible.

fallback_on_singular	Logical. If TRUE, fall back from a random-slope model to a random-intercept model when the attempted model is singular or fails.
REML	Logical. Passed to <code>lme4::lmer()</code> .
optimizer	Optimizer passed to <code>lme4::lmerControl()</code> .
maxfun	Maximum optimizer iterations passed to <code>lme4::lmerControl()</code> .
drop_missing	Logical. If TRUE, rows with missing or non-finite model inputs are removed before fitting.
...	Additional arguments passed to <code>lme4::lmer()</code> or <code>stats::lm()</code> .

Value

A list containing the fitted model, formula, attempted model, fallback information, fixed effects, comparison table, settings, and model diagnostics.

`fit_gazepoint_pupil_window_sensitivity`

Run sensitivity models for confirmatory pupil-window analyses

Description

Run a compact set of sensitivity models for confirmatory pupil-window analyses. Supported model families are the main linear mixed model, a weighted linear mixed model, a fixed-effects linear model, and a weighted fixed-effects linear model. Weighted models use the prepared valid-sample count column as weights by default.

Usage

```
fit_gazepoint_pupil_window_sensitivity(
  data,
  outcome_col = "pupil_model_outcome",
  subject_col = "pupil_model_subject",
  condition_col = "pupil_model_condition",
  window_col = "pupil_model_window",
  weights_col = "pupil_model_weight",
  model_types = c("lmm", "weighted_lmm", "lm", "weighted_lm"),
  include_condition = TRUE,
  include_window = TRUE,
  include_interaction = TRUE,
  random_intercept = TRUE,
  random_window_slopes = FALSE,
  fallback_on_singular = TRUE,
  REML = FALSE,
  optimizer = "bobyqa",
  maxfun = 2e+05,
  drop_missing = TRUE,
  ...
)
```

Arguments

<code>data</code>	Pupil-window model data, usually produced by <code>prepare_gazepoint_pupil_window_model_data()</code> .
<code>outcome_col</code>	Outcome column.
<code>subject_col</code>	Subject column.
<code>condition_col</code>	Condition column.
<code>window_col</code>	Window column.
<code>weights_col</code>	Optional weights column.
<code>model_types</code>	Character vector of model types to fit. Supported values are "lmm", "weighted_lmm", "lm", and "weighted_lm".
<code>include_condition</code>	Logical. Include condition fixed effects when more than one condition level is available.
<code>include_window</code>	Logical. Include window fixed effects when more than one window level is available.
<code>include_interaction</code>	Logical. Include the condition-by-window interaction when both condition and window are used.
<code>random_intercept</code>	Logical. Include a subject random intercept for LMM model types when feasible.
<code>random_window_slopes</code>	Logical. Attempt subject-level random window slopes for LMM model types when feasible.
<code>fallback_on_singular</code>	Logical. If TRUE, LMM model types may fall back from random-window-slope models to random-intercept models when needed.
<code>REML</code>	Logical. Passed to <code>lme4::lmer()</code> .
<code>optimizer</code>	Optimizer passed to <code>lme4::lmerControl()</code> .
<code>maxfun</code>	Maximum optimizer iterations passed to <code>lme4::lmerControl()</code> .
<code>drop_missing</code>	Logical. If TRUE, rows with missing or non-finite model inputs are removed before fitting.
<code>...</code>	Additional arguments passed to <code>fit_gazepoint_pupil_window_lmm()</code> .

Value

A list containing fitted models, formulas, fixed effects, a comparison table, settings, and model-status information.

fit_gazepoint_transition_count_nb_sensitivity

Fit optional negative-binomial transition-count sensitivity models

Description

Fit an optional negative-binomial sensitivity model for AOI/state transition counts using `glmmTMB` when it is installed. This helper is intended as a publication sensitivity branch for overdispersed transition-count outcomes.

Usage

```
fit_gazepoint_transition_count_nb_sensitivity(
  data,
  count_col = NULL,
  from_col = NULL,
  to_col = NULL,
  condition_cols = NULL,
  random_effect_cols = NULL,
  exposure_col = NULL,
  offset_col = NULL,
  formula = NULL,
  family = c("nbinom2", "nbinom1"),
  zero_inflation = FALSE,
  ziformula = NULL,
  dispformula = NULL,
  control = NULL,
  name = "gazepoint_transition_count_nb_sensitivity"
)
```

Arguments

<code>data</code>	A data frame containing transition-count rows.
<code>count_col</code>	Transition-count outcome column. If <code>NULL</code> , common count columns are detected automatically.
<code>from_col</code>	Transition origin column. If <code>NULL</code> , common origin columns are detected automatically.
<code>to_col</code>	Transition destination column. If <code>NULL</code> , common destination columns are detected automatically.
<code>condition_cols</code>	Optional fixed-effect condition columns.
<code>random_effect_cols</code>	Optional random-intercept grouping columns.
<code>exposure_col</code>	Optional positive exposure column. If supplied, the model includes <code>offset(log(exposure_col))</code> .
<code>offset_col</code>	Optional numeric offset column. Use either <code>exposure_col</code> or <code>offset_col</code> , not both.

formula	Optional model formula. If NULL, a formula is constructed from transition origin, destination, condition columns, optional offset, and random intercepts.
family	Negative-binomial family. Options are "nbinom2" and "nbinom1".
zero_inflation	Logical. If TRUE, use ziformula = ~1 unless ziformula is supplied.
ziformula	Optional zero-inflation formula passed to glmmTMB.
dispformula	Optional dispersion formula passed to glmmTMB.
control	Optional control object passed to glmmTMB.
name	Character label stored in the returned object.

Details

The helper keeps glmmTMB optional. If glmmTMB is not installed, it returns a structured skipped object rather than failing.

Value

A list with class gp3_transition_count_nb_sensitivity.

flag_gazepoint_pupil	<i>Flag invalid, missing, implausible, and outlying Gazepoint pupil samples</i>
----------------------	---

Description

Adds pupil-quality flags to a Gazepoint master sample-level table created by [as_gazepoint_master\(\)](#) or [create_gazepoint_master\(\)](#). This function is intended as a preprocessing step before interpolation, filtering, baseline correction, or pupil-based modelling.

Usage

```
flag_gazepoint_pupil(
  master,
  pupil_col = NULL,
  time_col = NULL,
  missing_pupil_col = NULL,
  group_cols = c("subject", "media_id"),
  min_pupil = 0,
  max_pupil = Inf,
  outlier_k = 1.5,
  flag_iqr_outliers = TRUE
)
```

Arguments

master	A Gazepoint master sample-level table.
pupil_col	Optional name of the pupil column to flag. If NULL, the function detects one of mean_pupil, pupil, pupil_raw, left_pupil, or right_pupil.
time_col	Optional name of the time column. If NULL, the function detects one of time_ms, time, time_orig, or time_orig_ms.
missing_pupil_col	Optional name of the missing-pupil flag column. If NULL, the function uses missing_pupil when available.
group_cols	Character vector of grouping columns used for IQR-based outlier detection. Defaults to c("subject", "media_id") using internally standardised names. Use character(0) for global outlier detection.
min_pupil	Minimum plausible pupil value. Defaults to 0.
max_pupil	Maximum plausible pupil value. Defaults to Inf. Use narrower values, such as 1 and 9, only when the pupil column is known to be measured in millimetres.
outlier_k	Multiplier for IQR-based outlier detection. Defaults to 1.5.
flag_iqr_outliers	Logical. If TRUE, IQR-based outliers are flagged. Defaults to TRUE.

Value

A tibble containing the original master table plus pupil-flagging columns.

Examples

```
## Not run:
flagged <- flag_gazepoint_pupil(master)

dplyr::count(flagged, pupil_flag_reason)

## End(Not run)
```

flag_gazepoint_pupil_artifacts

Flag Gazepoint pupil artifacts before interpolation

Description

Adds pupil-specific artifact flags for blink/trackloss contamination, physiological implausibility when pupil units are millimetres, pupil-speed outliers, left-right binocular pupil disagreement, and temporal padding around bad samples. The function preserves raw pupil columns and creates pupil_clean, which can be used as input for interpolation.

Usage

```
flag_gazepoint_pupil_artifacts(
  data,
  pupil_col = NULL,
  left_pupil_col = NULL,
  right_pupil_col = NULL,
  time_col = NULL,
  blink_col = NULL,
  trackloss_col = NULL,
  missing_pupil_col = NULL,
  pupil_unit_col = NULL,
  group_cols = c("subject", "media_id"),
  registry = NULL,
  blink_padding_pre_ms = NULL,
  blink_padding_post_ms = NULL,
  pupil_min_mm = NULL,
  pupil_max_mm = NULL,
  pupil_speed_mad_k = NULL,
  binocular_mad_k = NULL,
  max_physio_outlier_prop = 0.8,
  flag_speed_outliers = TRUE,
  flag_binocular_disagreement = TRUE,
  flag_physiological_outliers = TRUE
)
```

Arguments

<code>data</code>	A Gazepoint master table or pupil-processing table.
<code>pupil_col</code>	Optional name of the pupil column to clean. If NULL, the function detects one of <code>mean_pupil</code> , <code>pupil_raw</code> , <code>pupil</code> , <code>left_pupil</code> , or <code>right_pupil</code> .
<code>left_pupil_col</code>	Optional left-pupil column. If NULL, <code>left_pupil</code> is used when available.
<code>right_pupil_col</code>	Optional right-pupil column. If NULL, <code>right_pupil</code> is used when available.
<code>time_col</code>	Optional time column. If NULL, the function detects one of <code>time_ms</code> , <code>time</code> , <code>time_orig</code> , or <code>time_orig_ms</code> .
<code>blink_col</code>	Optional blink column. If NULL, <code>blink</code> is used when available.
<code>trackloss_col</code>	Optional trackloss column. If NULL, one of <code>trackloss</code> or <code>Trackloss</code> is used when available.
<code>missing_pupil_col</code>	Optional missing-pupil column. If NULL, <code>missing_pupil</code> is used when available.
<code>pupil_unit_col</code>	Optional pupil-unit column. If NULL, <code>pupil_unit</code> is used when available.
<code>group_cols</code>	Character vector of grouping columns used for speed outlier detection and artifact-padding windows. Defaults to <code>c("subject", "media_id")</code> . Use <code>character(0)</code> for global processing.
<code>registry</code>	Optional preprocessing registry created by create_gazepoint_preprocessing_registry() .

blink_padding_pre_ms	Padding before bad samples, in milliseconds. If NULL, taken from registry or defaults to 100.
blink_padding_post_ms	Padding after bad samples, in milliseconds. If NULL, taken from registry or defaults to 100.
pupil_min_mm	Minimum plausible pupil value when units are millimetres. If NULL, taken from registry or defaults to 1.
pupil_max_mm	Maximum plausible pupil value when units are millimetres. If NULL, taken from registry or defaults to 9.
pupil_speed_mad_k	MAD multiplier for pupil-speed outlier detection. If NULL, taken from registry or defaults to 6.
binocular_mad_k	MAD multiplier for binocular-disagreement detection. If NULL, taken from registry or defaults to 6.
max_physio_outlier_prop	Maximum allowed proportion of non-missing millimetre-labelled pupil samples that may be rejected by the physiological rule before the rule is automatically suppressed. Defaults to 0.80. This prevents raw-unit Gazepoint exports from being silently erased when the unit label suggests millimetres but the numeric scale is not compatible with ordinary 1–9 mm thresholds.
flag_speed_outliers	Logical. If TRUE, pupil-speed outliers are flagged. Defaults to TRUE.
flag_binocular_disagreement	Logical. If TRUE, left-right pupil disagreement is flagged when both eyes are available. Defaults to TRUE.
flag_physiological_outliers	Logical. If TRUE, millimetre-based physiological thresholds are applied only when the pupil unit is identified as millimetres. Defaults to TRUE.

Value

A tibble containing the original data plus pupil-artifact columns.

Examples

```
## Not run:
registry <- create_gazepoint_preprocessing_registry()

artifact_pupil <- flag_gazepoint_pupil_artifacts(
  master,
  registry = registry
)

dplyr::count(artifact_pupil, pupil_artifact_reason)

## End(Not run)
```

flag_gazepoint_pupil_hampel

Flag pupil artifacts with a Hampel filter

Description

Apply a rolling Hampel filter to a Gazepoint pupil column. The helper computes a rolling median and median absolute deviation (MAD) within a centred sample window, then flags pupil samples whose absolute deviation from the local median exceeds $k * MAD$.

Usage

```
flag_gazepoint_pupil_hampel(
  data,
  pupil_col,
  time_col = NULL,
  grouping_cols = NULL,
  window_size_samples = 7L,
  k = 3,
  min_valid_samples = 3L,
  scale_mad = 1.4826,
  flag_col = "pupil_hampel_outlier",
  median_col = "pupil_hampel_median",
  mad_col = "pupil_hampel_mad",
  threshold_col = "pupil_hampel_threshold",
  corrected_col = NULL,
  status_col = "pupil_hampel_status",
  overwrite = FALSE,
  name = "gazepoint_pupil_hampel"
)
```

Arguments

data	A data frame containing pupil observations.
pupil_col	Pupil column to screen.
time_col	Optional time column used to order samples within groups.
grouping_cols	Optional grouping columns, for example participant and trial.
window_size_samples	Odd integer rolling-window size in samples.
k	Hampel threshold multiplier.
min_valid_samples	Minimum finite pupil samples required inside a rolling window.
scale_mad	Scaling factor applied to MAD. The default 1.4826 makes MAD comparable to the standard deviation under normality.
flag_col	Name of the logical Hampel-flag output column.

median_col	Name of the rolling median output column.
mad_col	Name of the rolling MAD output column.
threshold_col	Name of the rolling threshold output column.
corrected_col	Optional name of a corrected pupil column. If supplied, flagged samples are replaced with the local rolling median.
status_col	Name of the row-level Hampel status column.
overwrite	Logical. If FALSE, the function errors when output columns already exist.
name	Character label stored in object attributes.

Details

This helper is intended as an optional sensitivity/artifact-flagging branch. It complements existing pupil artifact checks and should not automatically replace confirmatory preprocessing decisions.

Value

A tibble with Hampel-filter columns added. The object has class `gp3_pupil_hampel_flags`.

```
flag_gazepoint_sequence_anomalies
```

Flag unusual AOI sequences

Description

Identify grouped AOI sequences that are unusually short, unusually long, have high missingness, or contain very few unique AOI states. The function is intended as a lightweight quality-control helper rather than a definitive exclusion rule.

Usage

```
flag_gazepoint_sequence_anomalies(
  data,
  aoi_col,
  group_cols,
  time_col = NULL,
  min_length = 2,
  max_length = NULL,
  max_missing_prop = 0.5,
  z_threshold = 3,
  min_unique_aoi = 1
)
```

Arguments

data	A data frame containing AOI observations.
aoi_col	Name of the AOI column.
group_cols	Columns defining each sequence.
time_col	Optional time/order column.
min_length	Minimum acceptable non-missing sequence length.
max_length	Optional maximum acceptable non-missing sequence length.
max_missing_prop	Maximum acceptable missing AOI proportion.
z_threshold	Absolute z-score threshold for unusual sequence length.
min_unique_aoi	Minimum number of unique AOI labels expected.

Value

A data frame with sequence diagnostics and anomaly flags.

flag_tracking_quality *Flag low-quality Gazepoint recordings*

Description

Combines tracking-quality and sampling-rate summaries and flags rows with low gaze validity, low pupil validity, abnormal sampling rate, or short duration.

Usage

```
flag_tracking_quality(
  quality,
  sampling,
  by = c("USER_FILE", "MEDIA_ID"),
  min_gaze_valid_pct = 70,
  min_pupil_valid_pct = 70,
  expected_hz = 60,
  hz_tolerance = 5,
  min_duration_sec = NULL
)
```

Arguments

quality	Tracking-quality table from summarise_tracking_quality().
sampling	Sampling-rate table from check_sampling_rate().
by	Columns used to join quality and sampling.
min_gaze_valid_pct	Minimum acceptable FPOGV validity percentage.

min_pupil_valid_pct	Minimum acceptable pupil validity percentage.
expected_hz	Expected sampling rate.
hz_tolerance	Allowed deviation from the expected sampling rate.
min_duration_sec	Minimum acceptable recording duration in seconds.

Value

A tibble with quality, sampling, flag columns, and an overall review flag.

gazeport_example_aoi_geometry
<i>Example AOI geometry table</i>

Description

A lightweight synthetic AOI geometry table for AOI-verification examples. Coordinates are normalised to a 0–1 screen coordinate system.

Usage

```
gazeport_example_aoi_geometry
```

Format

A tibble with one row per stimulus and AOI, including:

media_id Synthetic stimulus identifier.

aoi Synthetic AOI label.

x_min, y_min, x_max, y_max Normalised rectangular AOI boundaries.

Examples

```
data(gazeport_example_aoi_geometry)
gazeport_example_aoi_geometry
```

```
gazeport_example_aoi_windows
```

Example AOI-window summary table

Description

A lightweight synthetic AOI-window summary table created from gazeport_example_master. It can be used in examples for AOI-window denominator checks, GLMM preparation, and AOI-window modelling.

Usage

```
gazeport_example_aoi_windows
```

Format

A tibble with one row per participant, stimulus/trial, and AOI time window.

Examples

```
data(gazeport_example_aoi_windows)
head(gazeport_example_aoi_windows)
```

```
gazeport_example_fixations
```

Example Gazeport fixation table

Description

A lightweight synthetic Gazeport-style fixation table for examples, tests, README workflows, and vignettes. The data are artificial and are not from a real participant study.

Usage

```
gazeport_example_fixations
```

Format

A tibble with fixation-level rows and columns including:

USER_FILE Synthetic participant/file identifier.

subject Synthetic participant identifier.

MEDIA_ID Synthetic stimulus identifier.

trial_global Synthetic trial identifier.

condition Synthetic experimental condition.
FPOGID Synthetic fixation identifier.
FPOGS Synthetic fixation start time.
FPOGD Synthetic fixation duration.
FPOGX, FPOGY Synthetic fixation coordinates.
FPOGV Synthetic fixation validity flag.
AOI Synthetic AOI label.

Examples

```
data(gazeport_example_fixations)
head(gazeport_example_fixations)
```

```
gazeport_example_master
```

Example Gazeport master table

Description

A lightweight synthetic Gazeport-style sample-level master table for examples, tests, README workflows, and vignettes. The data are artificial and are not from a real participant study.

Usage

```
gazeport_example_master
```

Format

A tibble with sample-level rows and columns including:

subject Synthetic participant identifier.
USER_FILE Gazeport-style participant/file identifier.
MEDIA_ID Synthetic stimulus identifier.
trial_global Synthetic trial identifier.
condition Synthetic experimental condition.
time Sample time in milliseconds.
x, y Normalised gaze coordinates.
pupil Synthetic pupil value.
valid Logical gaze/pupil validity flag.
artifact Logical synthetic pupil-artifact flag.
aoi_current Synthetic AOI state.
is_fixation, is_saccade Synthetic fixation and saccade indicators.
event_label Synthetic event marker.

Examples

```
data(gazepoint_example_master)
head(gazepoint_example_master)
```

```
gazepoint_example_pupil_windows
```

Example pupil-window summary table

Description

A lightweight synthetic pupil-window summary table created from `gazepoint_example_master`. It can be used in examples for pupil-window model-data preparation and confirmatory pupil-window modelling.

Usage

```
gazepoint_example_pupil_windows
```

Format

A tibble with one row per participant, stimulus/trial, condition, and pupil time window.

Examples

```
data(gazepoint_example_pupil_windows)
head(gazepoint_example_pupil_windows)
```

```
gp3tools_naming_policy
```

gp3tools user-facing naming policy

Description

Documents the package policy that British English `summarise_*` names are canonical for new summary helpers. Existing American English `summarize_*` exports remain available for backward compatibility.

Usage

```
gp3tools_naming_policy()
```

Value

A list describing the policy and migration rules.

`gp3tools_performance_limits`*Default performance limits for gp3tools export workflows*

Description

Returns conservative, explicit limits for elapsed time, approximate R-heap growth, and scaling behaviour. The limits are intended as regression gates, not as hardware-independent claims about absolute package speed.

Usage

```
gp3tools_performance_limits()
```

Value

A data frame with one row per benchmarked operation.

`harmonize_gazepoint_screen_coordinates`*Harmonize Gazepoint screen coordinates across resolutions*

Description

Rescales gaze coordinates from one screen or stimulus resolution to another. This is a deterministic coordinate transformation for harmonizing exports before plotting, AOI checks, or descriptive summaries. It does not recalibrate gaze data or correct measurement error.

Usage

```
harmonize_gazepoint_screen_coordinates(  
  data,  
  x_col,  
  y_col,  
  from_width,  
  from_height,  
  to_width,  
  to_height,  
  output_x_col = "gaze_x_harmonized",  
  output_y_col = "gaze_y_harmonized",  
  keep_original = TRUE  
)
```

Arguments

data A data frame.
x_col, y_col Character names of source coordinate columns.
from_width, from_height Original screen or stimulus dimensions.
to_width, to_height Target screen or stimulus dimensions.
output_x_col, output_y_col Names of the rescaled output columns.
keep_original If TRUE, original coordinate columns are retained. If FALSE, the original columns are removed when output column names differ.

Value

A copy of data with harmonized coordinate columns.

Examples

```

x <- data.frame(gaze_x = c(0, 960, 1920), gaze_y = c(0, 540, 1080))
harmonize_gazepoint_screen_coordinates(
  x,
  x_col = "gaze_x",
  y_col = "gaze_y",
  from_width = 1920,
  from_height = 1080,
  to_width = 1280,
  to_height = 720
)

```

impute_gazepoint_pupil_gp

Impute missing pupil samples with a lightweight Gaussian-process smoother

Description

Performs within-subject/trial Gaussian-process interpolation using a squared exponential kernel. This helper is intended for short missing segments after blink detection, not for reconstructing long unusable trials.

Usage

```

impute_gazepoint_pupil_gp(
  data,
  pupil,
  time,

```

```

    subject = NULL,
    trial = NULL,
    length_scale = NULL,
    noise = 1e-04,
    max_train = 300,
    output = "pupil_gp_imputed",
    flag = "pupil_was_gp_imputed"
  )

```

Arguments

data	A data frame.
pupil	Pupil column.
time	Time column.
subject	Optional subject column.
trial	Optional trial column.
length_scale	Kernel length scale in the same unit as time.
noise	Observation noise variance.
max_train	Maximum number of observed samples used per sequence.
output	Name of the imputed output column.
flag	Name of the logical imputation flag column.

Value

Data frame with imputed pupil values and an imputation flag.

inspect_gazepoint_columns
Inspect Gazepoint columns

Description

Inspect Gazepoint columns

Usage

```
inspect_gazepoint_columns(x)
```

Arguments

x	A data frame or path to a Gazepoint CSV export.
---	---

Value

A tibble describing column names, semantic groups, and missingness.

interpolate_gazepoint_blinks

Interpolate pupil values across detected blink intervals

Description

Masks samples inside blink intervals and interpolates only bounded internal gaps. Long or edge gaps remain missing.

Usage

```
interpolate_gazepoint_blinks(
  master_df,
  blink_df,
  pupil_cols = NULL,
  id_col = "USER_ID",
  group_cols = NULL,
  ts_col = "TIME",
  start_col = "start_time",
  end_col = "end_time",
  method = c("linear", "spline"),
  max_gap_ms = 500,
  suffix = "_blink_interp",
  keep_mask = TRUE,
  time_unit = c("auto", "seconds", "milliseconds")
)
```

Arguments

master_df	A sample-level data frame.
blink_df	Blink intervals returned by detect_gazepoint_blinks() .
pupil_cols	Pupil columns to interpolate. When NULL, common pupil columns are detected automatically.
id_col	Participant identifier shared by both inputs.
group_cols	Optional additional grouping columns shared by both inputs.
ts_col	Timestamp column in master_df.
start_col, end_col	Blink interval boundaries.
method	Interpolation method.
max_gap_ms	Maximum blink duration eligible for interpolation.
suffix	Suffix used for interpolated columns.
keep_mask	Add blink_interpolated and blink_masked columns.
time_unit	Timestamp unit.

Value

The sample table with interpolated pupil columns.

Examples

```
pupil <- data.frame(
  USER_ID = "P01",
  TIME = seq(0, 0.09, by = 0.01),
  mean_pupil = c(3, 3.1, 3.2, NA, NA, 3.3, 3.4, 3.5, 3.6, 3.7)
)
blinks <- detect_gazepoint_blinks(
  pupil,
  min_duration = 10
)
interpolate_gazepoint_blinks(pupil, blinks)
```

interpolate_gazepoint_pupil

Interpolate short missing gaps in Gazepoint pupil data

Description

Performs linear interpolation over short internal gaps in Gazepoint pupil data. This function is intended to be used after [flag_gazepoint_pupil\(\)](#) or [flag_gazepoint_pupil_artifacts\(\)](#). When available, `pupil_clean` is used as the preferred default input column, followed by `pupil_for_preprocessing`. Leading gaps, trailing gaps, long gaps, non-finite time values, and groups with too few valid pupil samples are not interpolated.

Usage

```
interpolate_gazepoint_pupil(
  data,
  pupil_col = NULL,
  time_col = NULL,
  group_cols = c("subject", "media_id"),
  max_gap_ms = 150,
  max_gap_samples = Inf,
  min_valid_points = 2
)
```

Arguments

<code>data</code>	A Gazepoint master table, preferably after flag_gazepoint_pupil() or flag_gazepoint_pupil_artifacts() .
<code>pupil_col</code>	Optional name of the pupil column to interpolate. If <code>NULL</code> , the function detects one of <code>pupil_clean</code> , <code>pupil_for_preprocessing</code> , <code>mean_pupil</code> , <code>pupil</code> , <code>pupil_raw</code> , <code>left_pupil</code> , or <code>right_pupil</code> .

<code>time_col</code>	Optional name of the time column. If NULL, the function detects one of <code>time_ms</code> , <code>time</code> , <code>time_orig</code> , or <code>time_orig_ms</code> .
<code>group_cols</code>	Character vector of grouping columns used to keep interpolation within independent time series. Defaults to <code>c("subject", "media_id")</code> using internally standardised names. Use <code>character(0)</code> for global interpolation.
<code>max_gap_ms</code>	Maximum duration, in milliseconds, of a gap that may be interpolated. The duration is measured between the valid samples immediately before and after the gap. Defaults to 150.
<code>max_gap_samples</code>	Maximum number of consecutive missing samples that may be interpolated. Defaults to Inf.
<code>min_valid_points</code>	Minimum number of valid samples required within a group before interpolation is attempted. Defaults to 2.

Value

A tibble containing the original data plus interpolation columns.

Examples

```
## Not run:
flagged <- flag_gazepoint_pupil(master)

interpolated <- interpolate_gazepoint_pupil(flagged)

dplyr::count(interpolated, pupil_interpolation_status)

artifact_flagged <- flag_gazepoint_pupil_artifacts(master)

artifact_interpolated <- interpolate_gazepoint_pupil(artifact_flagged)

dplyr::count(artifact_interpolated, pupil_interpolation_status)

## End(Not run)
```

```
interpolate_gazepoint_pupil_pchip
```

Interpolate Gazepoint pupil data using PCHIP

Description

Apply shape-preserving piecewise cubic Hermite interpolation to short internal gaps in Gazepoint pupil time series. This helper is intended as a sensitivity branch alongside the default linear interpolation workflow. It does not overwrite the original pupil column.

Usage

```

interpolate_gazepoint_pupil_pchip(
  data,
  pupil_col = NULL,
  time_col = NULL,
  grouping_cols = NULL,
  max_gap_ms = 500,
  max_gap_samples = NULL,
  min_valid_points = 3,
  output_col = "pupil_interpolated_pchip",
  flag_col = "interpolated_pupil_pchip",
  status_col = "pchip_interpolation_status"
)

```

Arguments

<code>data</code>	A data frame containing pupil time-series data.
<code>pupil_col</code>	Name of the pupil column to interpolate. If <code>NULL</code> , common processed pupil columns are detected automatically.
<code>time_col</code>	Name of the time column. If <code>NULL</code> , common time columns are detected automatically.
<code>grouping_cols</code>	Optional character vector of grouping columns. If <code>NULL</code> , common participant/trial/media grouping columns are detected automatically. Use <code>character(0)</code> to interpolate the full data as one sequence.
<code>max_gap_ms</code>	Maximum internal gap duration, in milliseconds, eligible for interpolation. If both <code>max_gap_ms</code> and <code>max_gap_samples</code> are supplied, both criteria must be satisfied.
<code>max_gap_samples</code>	Maximum number of consecutive missing samples eligible for interpolation.
<code>min_valid_points</code>	Minimum number of valid non-missing points required within a group before PCHIP interpolation is attempted.
<code>output_col</code>	Name of the interpolated pupil output column.
<code>flag_col</code>	Name of the logical flag column indicating samples filled by PCHIP interpolation.
<code>status_col</code>	Name of the interpolation-status column.

Value

A tibble with PCHIP interpolation columns added.

```
launch_gazepoint_qc_dashboard
```

Launch or describe a lightweight QC dashboard

Description

Provide a minimal optional Shiny dashboard launcher for inspecting a data frame. With `launch = FALSE`, the function returns a dashboard specification without requiring Shiny.

Usage

```
launch_gazepoint_qc_dashboard(  
  data = NULL,  
  title = "gp3tools QC dashboard",  
  launch = FALSE  
)
```

Arguments

<code>data</code>	Optional data frame to inspect.
<code>title</code>	Dashboard title.
<code>launch</code>	Should a Shiny app be launched?

Value

A dashboard specification, or a Shiny app object when launched.

```
mean_gazepoint_pupil
```

Calculate mean binocular pupil size

Description

Calculate mean binocular pupil size

Usage

```
mean_gazepoint_pupil(  
  master_df,  
  lp_col = "LPupil",  
  rp_col = "RPupil",  
  output_col = "mean_pupil",  
  min_eyes = 1  
)
```

Arguments

master_df	A sample-level pupil data frame.
lp_col, rp_col	Left- and right-pupil columns.
output_col	Name of the generated column.
min_eyes	Minimum number of finite eye measurements required.

Value

The input data with a binocular mean-pupil column.

Examples

```
pupil <- data.frame(
  LPupil = c(3, NA, 4),
  RPupil = c(3.2, 3.5, NA)
)
mean_gazepoint_pupil(pupil)
```

```
plot_gazepoint_aoi_gamm
```

Plot AOI time-course GAMM results

Description

Plot observed AOI target-looking proportions and fitted GAMM trajectories from a model returned by `fit_gazepoint_aoi_gamm()`.

Usage

```
plot_gazepoint_aoi_gamm(
  fit,
  n_time_points = 100,
  include_observed = TRUE,
  include_fitted = TRUE,
  show_ci = TRUE,
  ci_level = 0.95,
  exclude_random_effects = TRUE,
  observed_summary = c("pooled"),
  point_size = 1.8,
  point_alpha = 0.65,
  line_width = 0.8,
  ribbon_alpha = 0.15,
  title = NULL,
  subtitle = NULL,
  x_label = "Time (ms)",
  y_label = "Target AOI looking probability",
  y_limits = c(0, 1)
)
```

Arguments

<code>fit</code>	A result object returned by <code>fit_gazepoint_aoi_gamm()</code> .
<code>n_time_points</code>	Number of time points used for the fitted prediction grid. If NULL, the observed time bins are used.
<code>include_observed</code>	Logical. If TRUE, plot observed binned proportions.
<code>include_fitted</code>	Logical. If TRUE, plot fitted GAMM trajectories.
<code>show_ci</code>	Logical. If TRUE, plot fitted confidence intervals.
<code>ci_level</code>	Confidence level for fitted intervals.
<code>exclude_random_effects</code>	Logical. If TRUE, exclude subject random-effect smooths from fitted predictions.
<code>observed_summary</code>	Character. Currently "pooled" pools successes and denominators by condition and time.
<code>point_size</code>	Size of observed points.
<code>point_alpha</code>	Transparency for observed points.
<code>line_width</code>	Width of fitted trajectory lines.
<code>ribbon_alpha</code>	Transparency for fitted confidence ribbons.
<code>title</code>	Optional plot title.
<code>subtitle</code>	Optional plot subtitle.
<code>x_label</code>	X-axis label.
<code>y_label</code>	Y-axis label.
<code>y_limits</code>	Optional numeric vector of length 2 for y-axis limits.

Details

The plot supports single-condition fallback models and multi-condition AOI time-course GAMMs. By default, fitted trajectories are population-level predictions with subject random-effect smooths excluded.

Value

A ggplot object with prediction and observed data stored as attributes.

```
plot_gazepoint_aoi_timeline
```

Plot an AOI timeline

Description

Creates a scarf-style timeline plot showing the current AOI over time for each subject, trial, or user-defined row grouping.

Usage

```
plot_gazepoint_aoi_timeline(
  data,
  aoi_col,
  time_col,
  y_col = NULL,
  subject_col = NULL,
  trial_col = NULL,
  group_cols = NULL,
  include_missing = FALSE,
  missing_label = "missing",
  sample_width = NULL,
  title = NULL,
  x_label = "Time",
  y_label = NULL,
  aoi_label = "AOI"
)
```

Arguments

<code>data</code>	A data frame containing AOI observations.
<code>aoi_col</code>	Character scalar. Column containing AOI labels.
<code>time_col</code>	Character scalar. Column containing time values.
<code>y_col</code>	Optional character scalar used directly as the y-axis row.
<code>subject_col</code>	Optional subject column used to construct the y-axis row.
<code>trial_col</code>	Optional trial column used to construct the y-axis row.
<code>group_cols</code>	Optional character vector used to construct the y-axis row when <code>y_col</code> is not supplied.
<code>include_missing</code>	Logical. If TRUE, missing or empty AOI labels are retained as <code>missing_label</code> ; otherwise they are removed.
<code>missing_label</code>	Character scalar used when <code>include_missing</code> = TRUE.
<code>sample_width</code>	Optional numeric tile width. If omitted, a median time difference is estimated.
<code>title</code>	Optional plot title.

x_label	X-axis label.
y_label	Optional y-axis label.
aoi_label	Fill legend label.

Value

A ggplot object.

Examples

```
dat <- data.frame(
  subject = rep(c("S01", "S02"), each = 4),
  trial = "T01",
  time = rep(1:4, 2),
  AOI = c("A", "A", "B", "C", "A", "B", "B", "C")
)

plot_gazepoint_aoi_timeline(
  dat,
  aoi_col = "AOI",
  time_col = "time",
  subject_col = "subject",
  trial_col = "trial"
)
```

```
plot_gazepoint_aoi_transition_matrix
```

Plot Gazepoint AOI transition matrix

Description

Plot a heatmap of AOI transition counts or probabilities from the output of `compute_gazepoint_aoi_transition_matrix()` or from a compatible long-form transition table.

Usage

```
plot_gazepoint_aoi_transition_matrix(
  transitions,
  value = c("prob", "n"),
  state_order = NULL,
  by_cols = NULL,
  include_zero = TRUE,
  show_labels = TRUE,
  label_digits = 2,
  label_size = 3,
  facet = TRUE,
  title = NULL
)
```

Arguments

transitions	A gp3_aoi_transition_matrix object, a long-form transition table with from, to, n, and/or prob columns, or a numeric matrix with AOI states as row and column names.
value	Which value to plot: "prob" for transition probabilities or "n" for transition counts.
state_order	Optional character vector defining the AOI order on the heatmap axes.
by_cols	Optional character vector of grouping columns to facet by. If NULL, the function uses grouping columns stored in a gp3_aoi_transition_matrix object, when available.
include_zero	Logical. If TRUE, all possible state-to-state cells are shown, with missing transitions displayed as zero.
show_labels	Logical. If TRUE, cell values are printed inside tiles.
label_digits	Number of digits used when labelling probabilities.
label_size	Text size for cell labels.
facet	Logical. If TRUE, grouped transition tables are faceted.
title	Optional plot title.

Value

A ggplot2 plot object.

plot_gazepoint_aoi_verification

Plot AOI geometry for visual verification

Description

Create a visual verification plot of AOI rectangles, with optional gaze samples overlaid.

Usage

```
plot_gazepoint_aoi_verification(
  aoi_geometry,
  gaze_data = NULL,
  geometry_aoi_col = NULL,
  geometry_stimulus_col = NULL,
  x_min_col = NULL,
  y_min_col = NULL,
  x_max_col = NULL,
  y_max_col = NULL,
  x_col = NULL,
  y_col = NULL,
  width_col = NULL,
```

```

height_col = NULL,
gaze_x_col = NULL,
gaze_y_col = NULL,
gaze_stimulus_col = NULL,
screen_x_range = c(0, 1),
screen_y_range = c(0, 1),
facet_by_stimulus = TRUE,
show_labels = TRUE,
show_gaze = TRUE,
invert_y = TRUE,
point_alpha = 0.35,
point_size = 1.2,
line_width = 0.8,
label_size = 3
)

```

Arguments

<code>aoi_geometry</code>	A data frame containing AOI geometry definitions.
<code>gaze_data</code>	Optional data frame containing gaze samples to overlay.
<code>geometry_aoi_col</code>	AOI label/name column in <code>aoi_geometry</code> .
<code>geometry_stimulus_col</code>	Optional stimulus/media column in <code>aoi_geometry</code> .
<code>x_min_col</code>	Optional AOI left/x-min column.
<code>y_min_col</code>	Optional AOI top/y-min column.
<code>x_max_col</code>	Optional AOI right/x-max column.
<code>y_max_col</code>	Optional AOI bottom/y-max column.
<code>x_col</code>	Optional AOI left/x column used with <code>width_col</code> .
<code>y_col</code>	Optional AOI top/y column used with <code>height_col</code> .
<code>width_col</code>	Optional AOI width column.
<code>height_col</code>	Optional AOI height column.
<code>gaze_x_col</code>	Optional gaze x-coordinate column.
<code>gaze_y_col</code>	Optional gaze y-coordinate column.
<code>gaze_stimulus_col</code>	Optional gaze stimulus/media column.
<code>screen_x_range</code>	Numeric length-2 vector defining the screen x range.
<code>screen_y_range</code>	Numeric length-2 vector defining the screen y range.
<code>facet_by_stimulus</code>	Logical. If TRUE, facet by stimulus/media when a stimulus column is available.
<code>show_labels</code>	Logical. If TRUE, draw AOI labels at AOI centres.
<code>show_gaze</code>	Logical. If TRUE, overlay gaze samples when <code>gaze_data</code> is supplied.
<code>invert_y</code>	Logical. If TRUE, reverse the y-axis so screen origin is at the top-left.

point_alpha	Alpha transparency for gaze points.
point_size	Size of gaze points.
line_width	Width of AOI rectangle borders.
label_size	Size of AOI labels.

Value

A ggplot object.

plot_gazepoint_binocular_diagnostics
<i>Plot binocular pupil diagnostics, reconstruction, and sensitivity</i>

Description

Provides a compact plotting interface for binocular traces, agreement, Bland-Altman-style differences, artificial-missingness validation, validation residuals, provenance timelines, reconstruction burden, sensitivity summaries, and reconstructed gap durations. Every non-dashboard call returns a ggplot object that can be further customised.

Usage

```
plot_gazepoint_binocular_diagnostics(  
  x,  
  type = c("trace", "agreement", "bland_altman", "validation", "residuals", "timeline",  
           "burden", "sensitivity", "gaps", "dashboard"),  
  left_col = NULL,  
  right_col = NULL,  
  time_col = NULL,  
  prefix = "gp3_binocular",  
  point_alpha = 0.35,  
  bins = 30L  
)
```

Arguments

x	A data frame/reconstruction result, or an object produced by diagnose_gazepoint_binocular_pupil() , validate_gazepoint_binocular_reconstruction() , audit_gazepoint_binocular_reconstruction() , or analyse_gazepoint_binocular_sensitivity() .
type	Plot type: "trace", "agreement", "bland_altman", "validation", "residuals", "timeline", "burden", "sensitivity", "gaps", or "dashboard".
left_col, right_col, time_col	Column names for data-frame plots. For reconstruction outputs these are inferred from metadata when omitted.
prefix	Reconstruction prefix.
point_alpha	Point transparency for dense scatterplots.
bins	Histogram bins for type = "gaps".

Value

A ggplot2 object. `type = "dashboard"` returns a named list of ggplot objects rather than introducing a layout dependency.

See Also

[reconstruct_gazepoint_binocular_pupil\(\)](#), [validate_gazepoint_binocular_reconstruction\(\)](#)

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 3, n_trials = 2, seed = 31)
dat$pupil_left[20:24] <- NA_real_
rec <- reconstruct_gazepoint_binocular_pupil(
  dat, "pupil_left", "pupil_right", time_col = "timestamp_ms",
  group_cols = "subject", min_pairs = 20
)
plot_gazepoint_binocular_diagnostics(rec, type = "agreement")
```

plot_gazepoint_cluster_null_distribution

Plot the cluster-permutation null distribution

Description

Plot the null distribution of maximum cluster statistics from a cluster-permutation result when available.

Usage

```
plot_gazepoint_cluster_null_distribution(
  result,
  observed_line = TRUE,
  title = NULL
)
```

Arguments

<code>result</code>	A result object returned by <code>run_gazepoint_cluster_permutation()</code> .
<code>observed_line</code>	Should observed cluster statistics be added when available?
<code>title</code>	Optional plot title.

Value

A ggplot object.

`plot_gazepoint_cluster_permutation`*Plot a Gazepoint cluster-permutation result*

Description

`plot_gazepoint_cluster_permutation()` is a compatibility wrapper around `plot_gazepoint_cluster_results()`. It uses the existing validated `gp3tools` plotting engine while providing a name aligned with the cluster-permutation workflow.

Usage

```
plot_gazepoint_cluster_permutation(result, ...)
```

Arguments

<code>result</code>	A result object returned by <code>run_gazepoint_cluster_permutation()</code> .
<code>...</code>	Additional arguments passed to <code>plot_gazepoint_cluster_results()</code> .

Value

A ggplot object.

`plot_gazepoint_cluster_results`*Plot cluster-based permutation results*

Description

Create a publication-ready time-course plot from the output of `run_gazepoint_cluster_permutation()`. The plot can show the mean condition difference, the time-wise test statistic, or both. Candidate time bins and cluster-level significant windows can be highlighted.

Usage

```
plot_gazepoint_cluster_results(  
  result,  
  plot_type = c("both", "difference", "statistic"),  
  alpha = 0.05,  
  significant_only = TRUE,  
  show_clusters = TRUE,  
  show_candidates = TRUE,  
  show_threshold = TRUE,  
  show_zero_line = TRUE,  
  title = NULL,
```

```

    subtitle = NULL,
    x_label = "Time (ms)",
    y_label = NULL,
    line_width = 0.7,
    point_size = 1.8,
    cluster_alpha = 0.12
  )

```

Arguments

<code>result</code>	A result object returned by <code>run_gazepoint_cluster_permutation()</code> .
<code>plot_type</code>	Character. One of "both", "difference", or "statistic".
<code>alpha</code>	Cluster-level significance threshold used to decide which clusters are significant for plotting.
<code>significant_only</code>	Logical. If TRUE, only significant clusters are shaded. If FALSE, all observed clusters are shaded.
<code>show_clusters</code>	Logical. If TRUE, shade cluster windows.
<code>show_candidates</code>	Logical. If TRUE, mark time bins exceeding the cluster-forming threshold.
<code>show_threshold</code>	Logical. If TRUE, show the cluster-forming threshold on the statistic panel.
<code>show_zero_line</code>	Logical. If TRUE, add a horizontal zero reference line.
<code>title</code>	Optional plot title.
<code>subtitle</code>	Optional plot subtitle.
<code>x_label</code>	X-axis label.
<code>y_label</code>	Optional y-axis label. If NULL, a label is chosen automatically.
<code>line_width</code>	Width of the time-course line.
<code>point_size</code>	Size of candidate-bin points.
<code>cluster_alpha</code>	Transparency for shaded cluster windows.

Details

Cluster-based permutation tests are intended for time-course inference. They should not be used to discover a confirmatory time window and then test that same window again in a second confirmatory model.

Value

A ggplot object.

plot_gazepoint_event_detector_agreement
Plot event-detector comparison diagnostics

Description

Plot event-detector comparison diagnostics

Usage

```
plot_gazepoint_event_detector_agreement(  
  x,  
  plot = c("counts", "durations", "agreement"),  
  main = NULL,  
  ylab = NULL,  
  las = 2  
)
```

Arguments

x	An object returned by compare_gazepoint_event_detectors() .
plot	Plot type: "counts", "durations", or "agreement".
main	Optional plot title.
ylab	Optional vertical-axis label.
las	Axis-label orientation passed to base graphics.

Value

Invisibly returns the plotted data.

plot_gazepoint_event_detector_benchmark
Plot event-detector benchmark diagnostics

Description

Plot event-detector benchmark diagnostics

Usage

```
plot_gazepoint_event_detector_benchmark(  
  x,  
  plot = c("f1", "precision_recall", "overlap", "timing_error", "counts"),  
  main = NULL,  
  ylab = NULL,  
  las = 2  
)
```

Arguments

x	An object returned by <code>benchmark_gazepoint_event_detectors()</code> .
plot	Plot type: "f1", "precision_recall", "overlap", "timing_error", or "counts".
main	Optional plot title.
ylab	Optional vertical-axis label.
las	Axis-label orientation passed to base graphics.

Value

Invisibly returns the detector-level data used by the plot.

plot_gazepoint_face_quality
Plot external facial-behaviour data quality

Description

Creates descriptive quality-control plots from `audit_gazepoint_face_quality()` output. The plots summarise face-data validity, confidence, status, or timing gaps. They do not infer facial expressions or emotional states.

Usage

```
plot_gazepoint_face_quality(
  data,
  plot_type = c("status", "validity", "confidence", "time_gaps"),
  group_col = NULL,
  title = NULL,
  ...
)
```

Arguments

data	A <code>gp3_face_quality_audit</code> object, a standardised face data frame, an unstandardised face data frame, or a CSV path.
plot_type	One of "status", "validity", "confidence", or "time_gaps".
group_col	Optional grouping column to use on the y-axis for group-level plots. Defaults to <code>face_quality_group</code> .
title	Optional plot title.
...	Additional arguments passed to <code>audit_gazepoint_face_quality()</code> when data is not already an audit object.

Value

A ggplot object.

Examples

```
face <- data.frame(
  frame = 1:3,
  timestamp = c(0, 0.033, 0.066),
  confidence = c(0.95, 0.90, 0.40),
  success = c(1, 1, 1)
)

plot_gazepoint_face_quality(face)
```

plot_gazepoint_gca	<i>Plot observed and fitted Growth Curve Analysis trajectories</i>
--------------------	--

Description

Plot observed and fitted pupil trajectories from a `gp3_gca_model` object. The function aggregates observed and fitted values by condition and time, and returns a `ggplot2` object.

Usage

```
plot_gazepoint_gca(
  model,
  data = NULL,
  time_col = "gca_time",
  observed_col = "gca_pupil",
  fitted_col = "gca_fitted",
  condition_col = "condition",
  subject_col = "subject",
  summarise = TRUE,
  show_observed = TRUE,
  show_fitted = TRUE,
  show_subjects = FALSE,
  interval = TRUE,
  title = NULL,
  point_size = 1.6,
  line_width = 0.8,
  alpha = 0.75
)
```

Arguments

<code>model</code>	A fitted object returned by <code>fit_gazepoint_gca()</code> , or a data frame containing observed and fitted values.
<code>data</code>	Optional data frame. If <code>NULL</code> and <code>model</code> is a <code>gp3_gca_model</code> , the model data are used.
<code>time_col</code>	Name of the time column.

observed_col	Name of the observed outcome column.
fitted_col	Name of the fitted-value column. If unavailable and model is a gp3_gca_model, fitted values are computed from the model.
condition_col	Name of the condition column.
subject_col	Optional subject column, used only when show_subjects = TRUE.
summarise	Logical. If TRUE, plot mean trajectories by condition and time. If FALSE, plot row-level values.
show_observed	Logical. If TRUE, include observed trajectory.
show_fitted	Logical. If TRUE, include fitted trajectory.
show_subjects	Logical. If TRUE, add faint subject-level observed trajectories when a subject column is available.
interval	Logical. If TRUE, add a standard-error ribbon around the observed mean trajectory when summarise = TRUE.
title	Optional plot title.
point_size	Point size for observed means.
line_width	Line width for trajectories.
alpha	Alpha value for observed points/lines.

Value

A ggplot2 object.

plot_gazepoint_heatmap

Plot a Gazepoint gaze or fixation heatmap

Description

plot_gazepoint_heatmap() creates a binned spatial heatmap from gaze or fixation coordinates. Points may optionally be weighted by duration.

Usage

```
plot_gazepoint_heatmap(
  data,
  x_col = NULL,
  y_col = NULL,
  weight_col = NULL,
  display_width = NULL,
  display_height = NULL,
  coordinate_space = c("auto", "normalized", "pixel"),
  bins = 60,
  alpha = 0.85,
```

```

    normalize = TRUE,
    show_points = FALSE,
    show_legend = TRUE
  )

```

Arguments

<code>data</code>	A data frame or an object returned by <code>prepare_gazepoint_heatmap_data()</code> .
<code>x_col, y_col</code>	Character strings giving x and y coordinate columns. These may be omitted when data is already prepared by <code>prepare_gazepoint_heatmap_data()</code> .
<code>weight_col</code>	Optional non-negative weight column, such as fixation duration.
<code>display_width, display_height</code>	Display width and height in pixels.
<code>coordinate_space</code>	One of "auto", "normalized", or "pixel".
<code>bins</code>	Number of bins. Either one integer or two integers for x and y.
<code>alpha</code>	Heatmap layer transparency.
<code>normalize</code>	Logical. If TRUE, bin intensities are scaled to the range 0–1.
<code>show_points</code>	Logical. If TRUE, raw points are added over the heatmap.
<code>show_legend</code>	Logical. If TRUE, the fill legend is shown.

Value

A ggplot object.

Examples

```

gaze <- data.frame(
  x = c(0.20, 0.25, 0.27, 0.70, 0.75),
  y = c(0.30, 0.32, 0.34, 0.55, 0.60),
  duration = c(120, 180, 160, 90, 100)
)

plot_gazepoint_heatmap(
  gaze,
  x_col = "x",
  y_col = "y",
  weight_col = "duration",
  display_width = 1920,
  display_height = 1080,
  bins = 20
)

```

plot_gazepoint_heatmap_overlay

Plot a Gazepoint heatmap over a background image

Description

plot_gazepoint_heatmap_overlay() overlays a binned gaze or fixation heatmap on a PNG background image, such as a stimulus screenshot. The png package is used only when this helper is called.

Usage

```
plot_gazepoint_heatmap_overlay(
  data,
  background_image,
  x_col = NULL,
  y_col = NULL,
  weight_col = NULL,
  display_width = NULL,
  display_height = NULL,
  coordinate_space = c("auto", "normalized", "pixel"),
  bins = 60,
  heatmap_alpha = 0.7,
  background_alpha = 1,
  normalize = TRUE,
  show_legend = TRUE
)
```

Arguments

data	A data frame or prepared heatmap data.
background_image	Path to a PNG background image.
x_col, y_col	Character strings giving x and y coordinate columns.
weight_col	Optional non-negative weight column.
display_width, display_height	Display width and height in pixels. If omitted, the PNG image dimensions are used.
coordinate_space	One of "auto", "normalized", or "pixel".
bins	Number of heatmap bins. Either one integer or two integers.
heatmap_alpha	Heatmap layer transparency.
background_alpha	Background image transparency.
normalize	Logical. If TRUE, bin intensities are scaled to 0–1.
show_legend	Logical. If TRUE, the fill legend is shown.

Value

A ggplot object.

Examples

```
gaze <- data.frame(
  x = c(0.20, 0.25, 0.70),
  y = c(0.30, 0.35, 0.60),
  duration = c(120, 200, 80)
)

if (requireNamespace("png", quietly = TRUE)) {
  bg <- tempfile(fileext = ".png")
  img <- array(1, dim = c(200, 300, 3))
  png::writePNG(img, bg)

  plot_gazepoint_heatmap_overlay(
    gaze,
    background_image = bg,
    x_col = "x",
    y_col = "y",
    weight_col = "duration"
  )
}
```

plot_gazepoint_missingness_profile

Plot a Gazepoint missingness profile

Description

Creates a descriptive plot of missingness rates from raw data or from the output of `summarize_gazepoint_missingness()`. The plot is intended for quality-control review and reporting.

Usage

```
plot_gazepoint_missingness_profile(
  data,
  cols = NULL,
  group_cols = NULL,
  plot_type = c("bar", "tile"),
  title = NULL,
  y_label = "Missingness rate"
)
```

Arguments

data	A data frame or a missingness summary produced by <code>summarize_gazepoint_missingness()</code> .
cols	Optional columns to summarize when data is raw data.
group_cols	Optional grouping columns when data is raw data.
plot_type	Either "bar" or "tile".
title	Optional plot title.
y_label	Optional y-axis label for bar plots.

Value

A ggplot object.

Examples

```
x <- simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)
plot_gazepoint_missingness_profile(
  x,
  cols = c("pupil_left", "pupil_right", "pupil"),
  group_cols = "condition"
)
```

plot_gazepoint_model_predictions

Plot observed summaries and model-implied predictions

Description

Create a reporting plot that overlays observed outcome summaries with fitted/model-predicted trajectories. The helper is intentionally generic and can be used with linear models, GLMs, mixed models, GAMMs, GCA-style models, AOI GLMMs, and pupil LMMs when the fitted object supports `predict()`.

Usage

```
plot_gazepoint_model_predictions(
  data,
  model = NULL,
  x_col,
  outcome_col,
  condition_col = NULL,
  group_cols = NULL,
  facet_cols = NULL,
  newdata = NULL,
  prediction_type = c("response", "link"),
  include_random_effects = FALSE,
  observed_summary_function = c("mean", "median"),
```

```

ci = 0.95,
show_observed = TRUE,
show_observed_ci = TRUE,
show_predictions = TRUE,
show_prediction_ci = TRUE,
point_alpha = 0.55,
line_width = 1,
name = "gazepoint_model_predictions"
)

```

Arguments

<code>data</code>	A data frame containing the observed data.
<code>model</code>	Optional fitted model object. If supplied, predictions are computed using <code>predict()</code> .
<code>x_col</code>	Column used on the x-axis, usually time or time bin.
<code>outcome_col</code>	Observed outcome column.
<code>condition_col</code>	Optional condition column used for colour/grouping.
<code>group_cols</code>	Optional additional grouping columns for observed and predicted trajectories.
<code>facet_cols</code>	Optional columns used for faceting.
<code>newdata</code>	Optional prediction grid. If NULL, predictions are computed on data and then summarised by <code>x/group/facet</code> .
<code>prediction_type</code>	Prediction scale passed to <code>predict()</code> . Common values are "response" and "link".
<code>include_random_effects</code>	Logical. For lme4 mixed models, FALSE requests population-level predictions via <code>re.form = NA</code> ; TRUE includes conditional random effects where possible.
<code>observed_summary_function</code>	Summary for observed outcomes. Options are "mean" and "median".
<code>ci</code>	Confidence level for observed and prediction intervals when standard errors are available.
<code>show_observed</code>	Logical. Plot observed summaries.
<code>show_observed_ci</code>	Logical. Plot observed normal-approximation intervals.
<code>show_predictions</code>	Logical. Plot model predictions when <code>model</code> is supplied.
<code>show_prediction_ci</code>	Logical. Plot prediction intervals when standard errors are available from <code>predict()</code> .
<code>point_alpha</code>	Alpha value for observed points.
<code>line_width</code>	Line width for prediction trajectories.
<code>name</code>	Character label stored in plot attributes.

Value

A ggplot object with attributes containing the observed summary, prediction summary, overview, and settings.

plot_gazepoint_model_residuals
<i>Plot model residual diagnostics</i>

Description

Create a compact residual diagnostic plot from either a fitted model object with residuals() and fitted() methods, or a data frame that already contains fitted values and residuals.

Usage

```
plot_gazepoint_model_residuals(  
  model = NULL,  
  data = NULL,  
  fitted_col = NULL,  
  residual_col = NULL,  
  type = c("residuals_fitted", "qq"),  
  title = NULL  
)
```

Arguments

model	Optional fitted model object.
data	Optional data frame containing fitted and residual columns.
fitted_col	Fitted-value column when data is supplied.
residual_col	Residual column when data is supplied.
type	Diagnostic plot type: residuals-versus-fitted or QQ plot.
title	Optional plot title.

Value

A ggplot object.

plot_gazepoint_multiverse_results
<i>Plot Gazepoint preprocessing multiverse results</i>

Description

Create diagnostic plots from preprocessing multiverse summaries or from pupil/AOI multiverse result objects.

Usage

```
plot_gazepoint_multiverse_results(
  x,
  plot = c("status", "rows", "pupil_parameters", "aoi_denominators"),
  family = c("all", "pupil", "aoi"),
  title = NULL,
  show_labels = TRUE
)
```

Arguments

<code>x</code>	A <code>gp3_multiverse_summary_results</code> , <code>gp3_pupil_multiverse_results</code> , or <code>gp3_aoi_multiverse_results</code> object.
<code>plot</code>	Character. Plot type. One of "status", "rows", "pupil_parameters", or "aoi_denominators".
<code>family</code>	Character. Which family to show. One of "all", "pupil", or "aoi".
<code>title</code>	Optional plot title.
<code>show_labels</code>	Logical. If TRUE, show branch labels on the y-axis.

Value

A ggplot object.

```
plot_gazepoint_phase_timeline
```

Plot a Gazepoint task-phase timeline

Description

Creates a descriptive timeline plot from segmented Gazepoint-style data or from a phase-coverage summary. The plot is intended for visual inspection of task-phase coverage and timing, not for inferential analysis.

Usage

```
plot_gazepoint_phase_timeline(
  data,
  phase_col = "task_phase",
  group_cols = NULL,
  time_col = NULL,
  title = NULL
)
```

Arguments

<code>data</code>	A segmented data frame or a summary produced by <code>summarize_gazepoint_phase_coverage()</code> .
<code>phase_col</code>	Character name of the phase column when data is raw data.
<code>group_cols</code>	Optional grouping columns when data is raw data.
<code>time_col</code>	Optional time column when data is raw data.
<code>title</code>	Optional plot title.

Value

A ggplot object.

Examples

```
x <- data.frame(time_ms = c(0, 250, 750, 1250))
windows <- data.frame(
  phase = c("baseline", "stimulus"),
  start = c(0, 500),
  end = c(500, 1500)
)
segmented <- segment_gazepoint_task_phases(x, "time_ms", windows)
plot_gazepoint_phase_timeline(segmented, time_col = "time_ms")
```

plot_gazepoint_pupil_preprocessing

Plot Gazepoint pupil preprocessing for one trial

Description

Create a visual audit plot for one selected subject, media item, trial, or trial-global identifier. The plot can show raw pupil, cleaned pupil, interpolated pupil, baseline-corrected pupil, smoothed pupil, and artifact flags.

Usage

```
plot_gazepoint_pupil_preprocessing(
  data,
  subject = NULL,
  media_id = NULL,
  trial = NULL,
  trial_global = NULL,
  condition = NULL,
  subject_col = "subject",
  media_col = "MEDIA_ID",
  trial_col = "trial",
  trial_global_col = "trial_global",
  condition_col = "condition",
```

```

time_col = "time",
raw_pupil_col = "pupil",
clean_pupil_col = "pupil_clean",
interpolated_pupil_col = "pupil_interpolated",
baseline_pupil_col = "pupil_baseline_corrected",
smoothed_pupil_col = "pupil_smoothed",
artifact_col = NULL,
artifact_reason_col = NULL,
status_col = "pupil_interpolation_status",
plot_style = c("faceted", "overlaid"),
bin_width_ms = 50,
max_event_marks = 150,
point_size = 0.8,
line_width = 0.35,
alpha = 0.95
)

```

Arguments

<code>data</code>	A Gazepoint pupil data frame.
<code>subject</code>	Optional subject value to filter.
<code>media_id</code>	Optional media identifier value to filter.
<code>trial</code>	Optional trial value to filter.
<code>trial_global</code>	Optional global trial identifier value to filter.
<code>condition</code>	Optional condition value to filter.
<code>subject_col</code>	Name of the subject column.
<code>media_col</code>	Name of the media identifier column.
<code>trial_col</code>	Name of the trial column.
<code>trial_global_col</code>	Name of the global trial identifier column.
<code>condition_col</code>	Name of the condition column.
<code>time_col</code>	Name of the time column.
<code>raw_pupil_col</code>	Optional raw pupil column.
<code>clean_pupil_col</code>	Optional cleaned pupil column.
<code>interpolated_pupil_col</code>	Optional interpolated pupil column.
<code>baseline_pupil_col</code>	Optional baseline-corrected pupil column.
<code>smoothed_pupil_col</code>	Optional smoothed pupil column.
<code>artifact_col</code>	Optional artifact flag column. If NULL, the function tries <code>pupil_artifact_flag</code> , <code>pupil_flag_invalid</code> , and <code>artifact_flag</code> .

artifact_reason_col	Optional artifact-reason column. If NULL, the function tries pupil_artifact_reason, pupil_flag_reason, and artifact_reason.
status_col	Optional interpolation-status column.
plot_style	Either "faceted" or "overlaid".
bin_width_ms	Width of time bins in milliseconds. This is used only for visual smoothing of dense sample-level traces.
max_event_marks	Maximum number of artifact/interpolation rug marks to draw. Event marks are evenly thinned if there are more events.
point_size	Size control for artifact/interpolation rug marks.
line_width	Line width for pupil series.
alpha	Line and marker transparency.

Value

A ggplot2 plot object.

plot_gazepoint_pupil_status
Plot Gazepoint pupil preprocessing status

Description

Visualise observed, interpolated, missing, artifact, and other pupil-sample statuses over time or as grouped percentages.

Usage

```
plot_gazepoint_pupil_status(
  data,
  time_col = "time",
  pupil_col = NULL,
  status_col = "pupil_interpolation_status",
  interpolated_col = "pupil_was_interpolated",
  artifact_col = NULL,
  artifact_reason_col = NULL,
  group_cols = c("subject", "trial_global"),
  facet_cols = NULL,
  plot_type = c("timeline", "summary"),
  point_size = 0.7,
  alpha = 0.8,
  max_points = 50000
)
```

Arguments

<code>data</code>	A Gazepoint pupil data frame.
<code>time_col</code>	Name of the time column.
<code>pupil_col</code>	Optional pupil column used to detect remaining missing samples. If NULL, the function tries <code>pupil_smoothed</code> , <code>pupil_baseline_corrected</code> , <code>pupil_interpolated</code> , <code>pupil_clean</code> , and <code>pupil</code> .
<code>status_col</code>	Optional interpolation-status column.
<code>interpolated_col</code>	Optional logical interpolation flag column.
<code>artifact_col</code>	Optional artifact flag column. If NULL, the function tries <code>pupil_artifact_flag</code> , <code>pupil_flag_invalid</code> , and <code>artifact_flag</code> .
<code>artifact_reason_col</code>	Optional artifact-reason column. If NULL, the function tries <code>pupil_artifact_reason</code> , <code>pupil_flag_reason</code> , and <code>artifact_reason</code> .
<code>group_cols</code>	Character vector used to define timeline rows or summary groups.
<code>facet_cols</code>	Optional character vector of columns used for faceting.
<code>plot_type</code>	Either "timeline" or "summary".
<code>point_size</code>	Point size for timeline plots.
<code>alpha</code>	Point/column transparency.
<code>max_points</code>	Maximum number of rows to plot in timeline mode. If the input has more rows, rows are evenly thinned for plotting only.

Value

A ggplot2 plot object.

`plot_gazepoint_pupil_timecourse`
Plot Gazepoint pupil time course

Description

Plot a binned pupil time course with a mean line and confidence band. The function can plot one overall time course or condition-wise time courses, with optional faceting by variables such as condition, media, AOI, subject, or trial.

Usage

```
plot_gazepoint_pupil_timecourse(
  data,
  pupil_col = NULL,
  time_col = "time",
  condition_col = "condition",
```

```

    facet_cols = NULL,
    bin_width_ms = 100,
    ci_level = 0.95,
    min_samples = 1,
    band_alpha = 0.2,
    line_width = 0.8
  )

```

Arguments

<code>data</code>	A Gazepoint pupil data frame.
<code>pupil_col</code>	Name of the pupil column to plot. If NULL, the function tries <code>pupil_smoothed</code> , <code>pupil_baseline_corrected</code> , <code>pupil_baseline_percent_change</code> , <code>pupil_interpolated</code> , <code>pupil_clean</code> , and <code>pupil</code> .
<code>time_col</code>	Name of the time column.
<code>condition_col</code>	Optional condition column used for separate lines. If the column is missing or contains only missing values, the function plots a single "all_data" time course.
<code>facet_cols</code>	Optional character vector of columns used for faceting.
<code>bin_width_ms</code>	Width of time bins in milliseconds.
<code>ci_level</code>	Confidence level for the band.
<code>min_samples</code>	Minimum number of valid pupil samples required per time bin.
<code>band_alpha</code>	Transparency of the confidence band.
<code>line_width</code>	Width of the mean time-course line.

Value

A ggplot2 plot object.

```
plot_gazepoint_qc_overview
```

Plot a Gazepoint QC overview

Description

Creates a descriptive QC-status plot from a QC bundle, QC status summary, or object-summary table produced by `collect_gazepoint_qc_summaries()`. The plot is intended for quick review and reporting support; it does not replace the underlying audit outputs.

Usage

```

plot_gazepoint_qc_overview(
  qc_bundle,
  plot_type = c("status_counts", "objects"),
  title = NULL
)

```

Arguments

qc_bundle	A gp3_qc_summary_bundle, gp3_qc_status_summary, object-summary data frame, or list of objects that can be passed to collect_gazepoint_qc_summaries().
plot_type	Either "status_counts" or "objects".
title	Optional plot title.

Value

A ggplot object.

Examples

```
x <- list(
  pass = list(overview = data.frame(audit_status = "ok")),
  warn = list(overview = data.frame(audit_status = "review"))
)
plot_gazepoint_qc_overview(x)
```

plot_gazepoint_scanpath

Plot a fixation scanpath

Description

Plot fixation coordinates connected in temporal order. This is a static ggplot2 scanpath helper for fixation-level Gazepoint exports. It does not require a stimulus image, but the returned plot can be extended by users with additional ggplot2 layers.

Usage

```
plot_gazepoint_scanpath(
  data,
  x_col,
  y_col,
  group_cols = NULL,
  time_col = NULL,
  fixation_index_col = NULL,
  label_col = NULL,
  point_size = 2,
  line_width = 0.4,
  show_order = TRUE,
  add_arrows = TRUE,
  reverse_y = FALSE,
  title = NULL
)
```

Arguments

<code>data</code>	A fixation-level data frame.
<code>x_col</code>	Name of the horizontal fixation-coordinate column.
<code>y_col</code>	Name of the vertical fixation-coordinate column.
<code>group_cols</code>	Optional columns defining scanpaths.
<code>time_col</code>	Optional column used to order fixations.
<code>fixation_index_col</code>	Optional fixation index column used for labels.
<code>label_col</code>	Optional label column. If supplied, labels use this column.
<code>point_size</code>	Size of fixation points.
<code>line_width</code>	Width of connecting path lines.
<code>show_order</code>	Should fixation order labels be drawn?
<code>add_arrows</code>	Should arrows be added to scanpath lines?
<code>reverse_y</code>	Should the y-axis be reversed for screen-coordinate plots?
<code>title</code>	Optional plot title.

Value

A ggplot object.

plot_gazepoint_scanpaths

Plot multiple Gazepoint scanpaths

Description

Creates a descriptive multi-scanpath plot from gaze coordinates. This helper is intended for visual quality review, participant/trial inspection, and documentation examples. It should not be interpreted as an inferential scanpath-comparison method.

Usage

```
plot_gazepoint_scanpaths(
  data,
  x_col,
  y_col,
  order_col = NULL,
  group_cols = NULL,
  colour_col = NULL,
  facet_col = NULL,
  screen_width = NULL,
  screen_height = NULL,
  reverse_y = TRUE,
```

```

    show_points = TRUE,
    alpha = 0.45,
    linewidth = 0.4,
    point_size = 0.7,
    title = NULL
  )

```

Arguments

<code>data</code>	A data frame.
<code>x_col, y_col</code>	Character names of gaze coordinate columns.
<code>order_col</code>	Optional column used to order gaze samples before plotting.
<code>group_cols</code>	Optional character vector used to define separate paths.
<code>colour_col</code>	Optional column used for colour.
<code>facet_col</code>	Optional column used for faceting.
<code>screen_width, screen_height</code>	Optional screen dimensions. If supplied, axis limits are set to the screen bounds.
<code>reverse_y</code>	If TRUE, reverses the y-axis so the origin is displayed at the top-left, matching common screen-coordinate conventions.
<code>show_points</code>	If TRUE, sample points are added on top of paths.
<code>alpha</code>	Line opacity.
<code>linewidth</code>	Line width.
<code>point_size</code>	Point size when <code>show_points = TRUE</code> .
<code>title</code>	Optional plot title.

Value

A ggplot object.

Examples

```

x <- simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)
plot_gazepoint_scanpaths(
  x,
  x_col = "gaze_x",
  y_col = "gaze_y",
  order_col = "time_bin",
  group_cols = c("subject", "trial"),
  colour_col = "condition"
)

```

plot_gazepoint_scanpath_clusters

Plot fitted scanpath clusters

Description

Create base-R MDS, dendrogram, or silhouette diagnostics for an object returned by `cluster_gazepoint_scanpaths()`.

Usage

```
plot_gazepoint_scanpath_clusters(
  x,
  plot = c("mds", "dendrogram", "silhouette"),
  labels = TRUE,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  point_cex = 1.2,
  label_cex = 0.8
)
```

Arguments

<code>x</code>	An object returned by <code>cluster_gazepoint_scanpaths()</code> .
<code>plot</code>	Plot type: "mds", "dendrogram", or "silhouette".
<code>labels</code>	Should scanpath identifiers be displayed?
<code>main</code>	Optional plot title.
<code>xlab</code>	Optional horizontal-axis label.
<code>ylab</code>	Optional vertical-axis label.
<code>point_cex</code>	Point-size multiplier for the MDS display.
<code>label_cex</code>	Label-size multiplier.

Details

These displays describe distance structure and cluster separation. They do not establish distinct cognitive or psychological strategies.

Value

Invisibly returns a list containing the plot type, plot data, clustering method, and number of clusters.

Examples

```
d <- matrix(
  c(
    0, 0.1, 1, 1,
    0.1, 0, 1, 1,
    1, 1, 0, 0.1,
    1, 1, 0.1, 0
  ),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(LETTERS[1:4], LETTERS[1:4])
)

fit <- cluster_gazepoint_scanpaths(d, k = 2)
plot_gazepoint_scanpath_clusters(fit, plot = "mds")
```

```
plot_gazepoint_scanpath_cluster_stability
      Plot scanpath-cluster stability
```

Description

Create base-R diagnostics for co-clustering probabilities, iteration-level adjusted Rand indices, or sequence-level stability.

Usage

```
plot_gazepoint_scanpath_cluster_stability(
  x,
  plot = c("coclustering", "ari", "sequence"),
  specification = NULL,
  min_pair_coverage = 0.5,
  stable_threshold = 0.75,
  main = NULL,
  label_cex = 0.8
)
```

Arguments

<code>x</code>	An object returned by <code>bootstrap_gazepoint_scanpath_clusters()</code> .
<code>plot</code>	Plot type: "coclustering", "ari", or "sequence".
<code>specification</code>	Optional specification name. The first specification is used by default.
<code>min_pair_coverage</code>	Pair-coverage threshold used in sequence summaries.
<code>stable_threshold</code>	Stability threshold used in sequence summaries.
<code>main</code>	Optional plot title.
<code>label_cex</code>	Axis-label size multiplier.

Value

Invisibly returns a list containing the plot type, specification, and plotted data.

Examples

```
latent <- rep(1:3, each = 2)
d <- outer(
  latent,
  latent,
  FUN = function(x, y) ifelse(x == y, 0.1, 1)
)
diag(d) <- 0
dimnames(d) <- list(LETTERS[1:6], LETTERS[1:6])

stability <- bootstrap_gazepoint_scanpath_clusters(
  d,
  k = 3,
  n_boot = 10,
  seed = 1
)

plot_gazepoint_scanpath_cluster_stability(
  stability,
  plot = "coclustering"
)
```

plot_gazepoint_stimulus_layout_qc

Plot stimulus-layout quality control

Description

Draws screen/stimulus bounds, rectangular AOIs, and optional gaze coordinates. The plot is intended for visual quality review of coordinate systems, AOI placement, and gaze coverage. It is descriptive and should not be interpreted as an inferential attention analysis.

Usage

```
plot_gazepoint_stimulus_layout_qc(
  aoi_data,
  screen_width,
  screen_height,
  aoi_col = NULL,
  x_min_col = "x_min",
  x_max_col = "x_max",
  y_min_col = "y_min",
  y_max_col = "y_max",
  gaze_data = NULL,
```

```

    gaze_x_col = NULL,
    gaze_y_col = NULL,
    reverse_y = TRUE,
    show_aoi_labels = TRUE,
    show_gaze = TRUE,
    gaze_alpha = 0.25,
    gaze_point_size = 0.7,
    title = NULL
  )

```

Arguments

<code>aoi_data</code>	A data frame containing rectangular AOI geometry.
<code>screen_width, screen_height</code>	Numeric screen or stimulus dimensions.
<code>aoi_col</code>	Optional AOI identifier column.
<code>x_min_col, x_max_col, y_min_col, y_max_col</code>	Character names of AOI rectangle boundary columns.
<code>gaze_data</code>	Optional data frame containing gaze coordinates.
<code>gaze_x_col, gaze_y_col</code>	Character names of gaze-coordinate columns in <code>gaze_data</code> .
<code>reverse_y</code>	If TRUE, reverses the y-axis to match common screen coordinate conventions with the origin at the top-left.
<code>show_aoi_labels</code>	If TRUE, AOI labels are drawn at rectangle centres.
<code>show_gaze</code>	If TRUE, gaze points are shown when <code>gaze_data</code> is supplied.
<code>gaze_alpha</code>	Opacity of gaze points.
<code>gaze_point_size</code>	Size of gaze points.
<code>title</code>	Optional plot title.

Value

A ggplot object.

Examples

```

aoi <- data.frame(
  aoi = c("left", "right"),
  x_min = c(100, 1200),
  x_max = c(500, 1700),
  y_min = c(100, 100),
  y_max = c(400, 400)
)
gaze <- simulate_gazepoint_pupil_data(n_subjects = 1, n_trials = 1, n_time_bins = 10, seed = 1)
plot_gazepoint_stimulus_layout_qc(
  aoi,

```

```

    screen_width = 1920,
    screen_height = 1080,
    aoi_col = "aoi",
    gaze_data = gaze,
    gaze_x_col = "gaze_x",
    gaze_y_col = "gaze_y"
  )

```

```
plot_gazepoint_time_series
```

Plot a Gazepoint-style time series

Description

Creates a compact line plot for pupil, gaze, AOI, or other time-varying Gazepoint-derived measures. The helper is intentionally descriptive: it does not smooth, model, or infer effects unless the user has already prepared the plotted values.

Usage

```

plot_gazepoint_time_series(
  data,
  time_col,
  value_col,
  group_cols = NULL,
  colour_col = NULL,
  facet_col = NULL,
  alpha = 0.55,
  linewidth = 0.4,
  title = NULL,
  x_label = NULL,
  y_label = NULL
)

```

Arguments

<code>data</code>	A data frame.
<code>time_col</code>	Character name of the time column.
<code>value_col</code>	Character name of the value column.
<code>group_cols</code>	Optional character vector of grouping columns used to draw separate trajectories.
<code>colour_col</code>	Optional character name of a column used for colour.
<code>facet_col</code>	Optional character name of a column used for faceting.
<code>alpha</code>	Line opacity.
<code>linewidth</code>	Line width.
<code>title</code>	Optional plot title.
<code>x_label, y_label</code>	Optional axis labels.

Value

A ggplot object.

Examples

```
x <- simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)
plot_gazepoint_time_series(
  x,
  time_col = "time_bin",
  value_col = "pupil",
  group_cols = c("subject", "trial"),
  colour_col = "condition"
)
```

plot_gazepoint_time_varying_effect

Plot a time-varying effect curve

Description

Plot a time-varying effect, difference curve, or model-prediction contrast from a tidy data frame. This helper is intentionally lightweight: it does not refit a model, but visualises already computed estimates and optional interval bounds from GAMM, GCA, cluster, bootstrap, or prediction workflows.

Usage

```
plot_gazepoint_time_varying_effect(
  data,
  time_col,
  estimate_col,
  lower_col = NULL,
  upper_col = NULL,
  group_col = NULL,
  zero_line = TRUE,
  title = NULL,
  x_label = NULL,
  y_label = NULL
)
```

Arguments

data	A data frame containing time-varying estimates.
time_col	Name of the time column.
estimate_col	Name of the estimate/effect column.
lower_col	Optional lower interval column.

upper_col	Optional upper interval column.
group_col	Optional grouping/contrast column.
zero_line	Should a horizontal zero reference line be shown?
title	Optional plot title.
x_label	Optional x-axis label.
y_label	Optional y-axis label.

Value

A ggplot object.

plot_sampling_rate	<i>Plot Gazepoint sampling-rate diagnostics</i>
--------------------	---

Description

Creates a diagnostic plot of estimated sampling rate by participant/file and media stimulus.

Usage

```
plot_sampling_rate(  
  sampling,  
  user_col = "USER_FILE",  
  media_col = "MEDIA_ID",  
  expected_hz = 60,  
  hz_tolerance = 5  
)
```

Arguments

sampling	Sampling-rate table, usually from check_sampling_rate().
user_col	Column identifying the source/user file.
media_col	Column identifying the media/stimulus.
expected_hz	Expected sampling rate.
hz_tolerance	Allowed deviation from the expected sampling rate.

Value

A ggplot object.

plot_tracking_quality *Plot Gazepoint tracking-quality diagnostics*

Description

Creates a readable diagnostic plot of selected gaze and pupil validity percentages by participant/file and media stimulus.

Usage

```
plot_tracking_quality(
  data,
  metric_cols = NULL,
  user_col = "USER_FILE",
  media_col = "MEDIA_ID",
  review_col = "review_required",
  min_valid_pct = 70
)
```

Arguments

data	A tracking-quality or flagged-quality table, usually from summarise_tracking_quality() or flag_tracking_quality().
metric_cols	Validity percentage columns to plot. If NULL, the default is FPOGV_valid_pct and RPV_valid_pct, which provide a compact diagnostic view of gaze and right-pupil validity.
user_col	Column identifying the source/user file.
media_col	Column identifying the media/stimulus.
review_col	Optional column indicating whether a row requires review.
min_valid_pct	Vertical threshold line for acceptable validity.

Value

A ggplot object.

plot_transition_heatmap
Plot an AOI transition heatmap

Description

Plot an AOI transition heatmap

Usage

```
plot_transition_heatmap(transitions)
```

Arguments

transitions Output of compute_transition_matrix().

Value

A ggplot object.

```
prepare_gazepoint_aoi_gamm_data
```

Prepare AOI time-course data for GAMM analysis

Description

Prepare sample-level or binned Gazepoint AOI data for AOI time-course GAMM analysis. The function creates binned subject-by-condition-by-time summaries with binomial success/failure counts for target-AOI looking over time.

Usage

```
prepare_gazepoint_aoi_gamm_data(
  data,
  aoi_col = "aoi_current",
  target_aoi_values = NULL,
  outcome_col = NULL,
  subject_col = "subject",
  condition_col = "condition",
  time_col = "time",
  trial_col = NULL,
  time_bin_col = NULL,
  conditions = NULL,
  time_window = NULL,
  bin_size_ms = 50,
  denominator = c("valid", "all", "aoi_only"),
  valid_aoi_values = NULL,
  non_aoi_values = c("non_aoi"),
  missing_aoi_values = c("missing", ""),
  min_denominator_samples = 1,
  drop_invalid = TRUE,
  missing_condition_label = "all_data",
  outcome_label = "target_aoi"
)
```

Arguments

<code>data</code>	A data frame containing sample-level or binned AOI data.
<code>aoi_col</code>	Name of the AOI-state column. Used when <code>outcome_col</code> = NULL.
<code>target_aoi_values</code>	Character vector identifying target AOI values. Required when <code>outcome_col</code> = NULL.
<code>outcome_col</code>	Optional logical or 0/1 numeric column indicating target AOI looking at the sample level. If supplied, this takes priority over <code>aoi_col</code> and <code>target_aoi_values</code> .
<code>subject_col</code>	Name of the subject/participant column.
<code>condition_col</code>	Name of the condition column. If unavailable, a single fallback condition is created.
<code>time_col</code>	Name of the time column, in milliseconds.
<code>trial_col</code>	Optional trial identifier column.
<code>time_bin_col</code>	Optional existing time-bin column. If NULL, time bins are created from <code>time_col</code> using <code>bin_size_ms</code> .
<code>conditions</code>	Optional character vector of condition levels to retain and order.
<code>time_window</code>	Optional numeric vector of length 2 defining the retained time window in milliseconds.
<code>bin_size_ms</code>	Time-bin width in milliseconds when <code>time_bin_col</code> = NULL.
<code>denominator</code>	Denominator definition. "valid" uses non-missing AOI states, "all" uses all retained rows, and "aoi_only" uses only explicit AOI states.
<code>valid_aoi_values</code>	Optional character vector defining explicit AOI values for "aoi_only" denominators. If NULL, values beginning with "AOI" are treated as explicit AOIs, excluding non_aoi_values.
<code>non_aoi_values</code>	Character vector identifying non-AOI/background states.
<code>missing_aoi_values</code>	Character vector identifying missing AOI-state labels.
<code>min_denominator_samples</code>	Minimum number of denominator samples required per subject-condition-time bin.
<code>drop_invalid</code>	Logical. If TRUE, bins with zero or low denominators are removed from the returned data.
<code>missing_condition_label</code>	Fallback condition label when no usable condition column is available.
<code>outcome_label</code>	Descriptive label for the AOI-GAMM outcome.

Details

This helper is intended for modelling AOI time-course trajectories, such as target-AOI looking probability over time. It is separate from confirmatory AOI-window GLMMs and from cluster-based permutation tests.

Value

A tibble with standardised AOI-GAMM columns.

```
prepare_gazepoint_aoi_glmm_data
```

Prepare AOI-window data for binomial GLMMs

Description

Prepare AOI-window summaries for confirmatory binomial mixed-effects modelling. The function creates success, failure, denominator, proportion, subject, condition, and window columns from output produced by `summarise_gazepoint_aoi_windows()`.

Usage

```
prepare_gazepoint_aoi_glmm_data(
  data,
  success_col = "n_target_samples",
  denominator = c("valid", "all", "aoi", "custom"),
  denominator_col = NULL,
  valid_denominator_col = "n_valid_denominator_samples",
  all_denominator_col = "n_window_samples",
  aoi_denominator_col = "n_aoi_samples",
  subject_col = "subject",
  condition_col = "condition",
  window_col = "window_label",
  window_start_col = "window_start_ms",
  window_end_col = "window_end_ms",
  group_cols = NULL,
  min_denominator_samples = 1,
  drop_invalid = TRUE,
  missing_condition_label = "all_data",
  outcome_label = "target"
)
```

Arguments

<code>data</code>	AOI-window summary data.
<code>success_col</code>	Column containing the success count. For target-looking models this is usually <code>n_target_samples</code> .
<code>denominator</code>	Denominator definition. Use "valid" for valid AOI-window denominator samples, "all" for all window samples, "aoi" for AOI-only samples, or "custom" with <code>denominator_col</code> .
<code>denominator_col</code>	Custom denominator column when <code>denominator = "custom"</code> .
<code>valid_denominator_col</code>	Column used when <code>denominator = "valid"</code> .
<code>all_denominator_col</code>	Column used when <code>denominator = "all"</code> .

aoi_denominator_col	Column used when denominator = "aoi".
subject_col	Subject/participant column.
condition_col	Optional condition column.
window_col	AOI-window label column.
window_start_col	Optional window-start column.
window_end_col	Optional window-end column.
group_cols	Optional extra grouping columns to keep/check.
min_denominator_samples	Minimum acceptable denominator.
drop_invalid	Logical. If TRUE, rows with invalid binomial counts or too-small denominators are removed.
missing_condition_label	Label used when condition is missing.
outcome_label	Label stored in the output to identify the modelled AOI outcome.

Value

A tibble of GLMM-ready AOI-window rows.

```
prepare_gazepoint_aoi_sequences
```

Prepare Gazepoint AOI sequences

Description

Create ordered AOI-state sequences from sample-level Gazepoint AOI data or from the output of `summarise_gazepoint_aoi_entries()`. The output is transition-ready and includes the current AOI state, previous state, next state, dwell time before transition, and self-transition flags.

Usage

```
prepare_gazepoint_aoi_sequences(
  data,
  aoi_col = NULL,
  time_col = "time",
  group_cols = c("subject", "MEDIA_ID", "trial_global"),
  include_non_aoi = TRUE,
  non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
    "missing", "missing_aoi"),
  missing_aoi_label = "missing_aoi",
  include_terminal = TRUE
)
```

Arguments

<code>data</code>	A Gazepoint sample-level data frame or an AOI-entry table created by <code>summarise_gazepoint_aoi_entries</code> .
<code>aoi_col</code>	Name of the AOI-state column. Used only when data is sample-level data. If NULL, the function tries <code>aoi_current</code> , <code>AOI</code> , and <code>aoi_state</code> .
<code>time_col</code>	Name of the time column, in milliseconds. Used only when data is sample-level data.
<code>group_cols</code>	Character vector of columns defining independent AOI sequences, usually subject/media/trial.
<code>include_non_aoi</code>	Logical. If TRUE, non-AOI/background states are retained. If FALSE, non-AOI/background states are removed before sequence and transition fields are computed.
<code>non_aoi_values</code>	Character vector of AOI labels treated as background or non-AOI states.
<code>missing_aoi_label</code>	Label used when the AOI value is missing.
<code>include_terminal</code>	Logical. If TRUE, the final state of each sequence is retained with <code>next_state = NA</code> . If FALSE, terminal states are removed so that each output row represents an observed transition.

Value

A tibble with ordered AOI sequence and transition fields.

```
prepare_gazepoint_cluster_data
```

Prepare time-course data for cluster-based permutation tests

Description

Prepare sample-level or already binned Gazepoint time-course data for cluster-based permutation testing. The function standardises subject, condition, time-bin, outcome, sample-count, trial-count, and status columns. It can be used for AOI proportions, pupil time-course outcomes, or other continuous time-varying measures.

Usage

```
prepare_gazepoint_cluster_data(
  data,
  outcome_col,
  subject_col = "subject",
  condition_col = "condition",
  time_col = "time",
  trial_col = NULL,
  time_bin_col = NULL,
  conditions = NULL,
```

```

    time_window = NULL,
    bin_size_ms = 50,
    aggregation = c("mean", "proportion", "sum", "median"),
    min_samples_per_bin = 1,
    paired = TRUE,
    drop_invalid = TRUE,
    missing_condition_label = "all_data",
    outcome_label = "outcome"
  )

```

Arguments

<code>data</code>	A data frame containing sample-level or binned time-course data.
<code>outcome_col</code>	Column containing the outcome to test. For AOI analyses this is often a 0/1 or logical AOI column. For pupil analyses this is often a processed pupil column.
<code>subject_col</code>	Subject/participant column.
<code>condition_col</code>	Optional condition column.
<code>time_col</code>	Time column in milliseconds.
<code>trial_col</code>	Optional trial identifier column.
<code>time_bin_col</code>	Optional existing time-bin column. If <code>NULL</code> , time bins are created from <code>time_col</code> and <code>bin_size_ms</code> .
<code>conditions</code>	Optional character vector of condition levels to keep. Cluster tests are usually pairwise, so this is typically length 2.
<code>time_window</code>	Optional numeric vector of length 2 giving the time range to retain, in milliseconds.
<code>bin_size_ms</code>	Bin size in milliseconds when <code>time_bin_col = NULL</code> .
<code>aggregation</code>	How to aggregate samples within subject-condition-time bins. Supported values are "mean", "proportion", "sum", and "median". "proportion" is equivalent to the mean of a numeric/logical 0/1 outcome.
<code>min_samples_per_bin</code>	Minimum number of samples required per subject-condition-time bin.
<code>paired</code>	Logical. If <code>TRUE</code> , retain only subjects with all retained condition levels.
<code>drop_invalid</code>	Logical. If <code>TRUE</code> , rows and bins that are not suitable for cluster testing are removed.
<code>missing_condition_label</code>	Label used when condition is missing or <code>condition_col</code> is unavailable.
<code>outcome_label</code>	Label stored in the output to identify the outcome.

Details

Cluster-based permutation tests are intended for time-course inference. They should not be used to discover a time window and then test that same window again as a confirmatory analysis.

Value

A tibble with standardised cluster-test preparation columns.

```
prepare_gazepoint_eyetools_data
```

Prepare Gazepoint master data for eyetools-style workflows

Description

Convert a gp3tools master table into a dependency-free, eyetools-friendly sample-level table. The returned data frame keeps one row per sample and creates standard participant, trial, time, gaze-coordinate, binocular coordinate, pupil, AOI, fixation, event, validity, trackloss, and status columns.

Usage

```
prepare_gazepoint_eyetools_data(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  x_col = NULL,
  y_col = NULL,
  left_x_col = NULL,
  left_y_col = NULL,
  right_x_col = NULL,
  right_y_col = NULL,
  pupil_col = NULL,
  left_pupil_col = NULL,
  right_pupil_col = NULL,
  media_col = NULL,
  condition_col = NULL,
  aoi_col = NULL,
  fixation_col = NULL,
  event_col = NULL,
  validity_cols = NULL,
  trackloss_col = NULL,
  screen_x_range = c(0, 1),
  screen_y_range = c(0, 1),
  missing_aoi_label = "missing_aoi",
  keep_original_cols = TRUE
)
```

Arguments

<code>data</code>	A Gazepoint master table or sample-level gaze data frame.
<code>participant_col</code>	Participant/subject identifier column.
<code>trial_col</code>	Trial identifier column. If NULL, a trial identifier is created from <code>media_col</code> when available.

time_col	Sample time column.
x_col	Optional primary gaze x-coordinate column.
y_col	Optional primary gaze y-coordinate column.
left_x_col	Optional left-eye x-coordinate column.
left_y_col	Optional left-eye y-coordinate column.
right_x_col	Optional right-eye x-coordinate column.
right_y_col	Optional right-eye y-coordinate column.
pupil_col	Optional primary pupil column.
left_pupil_col	Optional left-eye pupil column.
right_pupil_col	Optional right-eye pupil column.
media_col	Optional media/stimulus identifier column.
condition_col	Optional condition/grouping column.
aoi_col	Optional AOI label/state column.
fixation_col	Optional fixation identifier column.
event_col	Optional event/marker column.
validity_cols	Optional validity columns used to define trackloss.
trackloss_col	Optional existing trackloss column. If supplied, it is used directly after coercion to logical.
screen_x_range	Numeric length-2 vector defining the screen x range.
screen_y_range	Numeric length-2 vector defining the screen y range.
missing_aoi_label	Label used for missing AOI values.
keep_original_cols	Logical. If TRUE, original columns are retained after the standard adapter columns.

Value

A tibble with class `gp3_eyetools_data`.

```
prepare_gazepoint_eyetrackingr_data
```

Prepare Gazepoint master data for eyetrackingR-style workflows

Description

Convert a `gp3tools` master table into a dependency-free, `eyetrackingR`-friendly sample-level table. The returned data frame keeps one row per gaze sample and creates standard participant, trial, time, gaze-coordinate, AOI, trackloss, and AOI-indicator columns.

Usage

```
prepare_gazepoint_eyetrackingr_data(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  aoi_col = NULL,
  x_col = NULL,
  y_col = NULL,
  media_col = NULL,
  condition_col = NULL,
  validity_cols = NULL,
  aoi_values = NULL,
  aoi_prefix = "aoi_",
  missing_aoi_label = "missing_aoi",
  non_aoi_values = c("outside", "none", "no_aoi", "non_aoi", "background", "off_aoi",
    "missing", "NA"),
  trackloss_col = NULL,
  keep_original_cols = TRUE
)
```

Arguments

<code>data</code>	A Gazepoint master table or sample-level gaze data frame.
<code>participant_col</code>	Participant/subject identifier column.
<code>trial_col</code>	Trial identifier column. If NULL, a trial identifier is created from <code>media_col</code> when available.
<code>time_col</code>	Sample time column.
<code>aoi_col</code>	AOI label/state column.
<code>x_col</code>	Optional gaze x-coordinate column.
<code>y_col</code>	Optional gaze y-coordinate column.
<code>media_col</code>	Optional media/stimulus identifier column.
<code>condition_col</code>	Optional condition/grouping column.
<code>validity_cols</code>	Optional validity columns used to define trackloss.
<code>aoi_values</code>	Optional AOI values for which logical indicator columns should be created. If NULL, values are detected from <code>aoi_col</code> .
<code>aoi_prefix</code>	Prefix for generated AOI indicator columns.
<code>missing_aoi_label</code>	Label used for missing AOI values.
<code>non_aoi_values</code>	Character values treated as non-AOI/background states.
<code>trackloss_col</code>	Optional existing trackloss column. If supplied, it is used directly after coercion to logical.
<code>keep_original_cols</code>	Logical. If TRUE, original columns are retained after the standard adapter columns.

Value

A tibble with class `gp3_eyetrackingr_data`.

```
prepare_gazepoint_fixation_aligned_data
```

Prepare fixation- or saccade-contingent aligned Gazept data

Description

Align Gazept observations to a within-trial event such as first fixation, first target-AOI entry, first fixation to a target AOI, first saccade to a target AOI, or a custom event marker. The helper returns the original data with event-aligned time, event metadata, pre-event/post-event flags, and trial-level summaries that help separate event-driven looking from looks that were already present before the event.

Usage

```
prepare_gazepoint_fixation_aligned_data(
  data,
  time_col,
  participant_col = NULL,
  trial_col = NULL,
  aoi_col = NULL,
  target_aoi = NULL,
  fixation_col = NULL,
  saccade_col = NULL,
  event_col = NULL,
  event_value = NULL,
  alignment_event = c("first_target_entry", "first_fixation_to_target",
    "first_saccade_to_aoi", "first_fixation", "custom"),
  baseline_window = NULL,
  analysis_window = NULL,
  keep_unaligned = FALSE,
  name = "gazepoint_fixation_aligned_data"
)
```

Arguments

<code>data</code>	A data frame containing Gazept samples, fixation rows, or trial-level time-course rows.
<code>time_col</code>	Time column.
<code>participant_col</code>	Optional participant column.
<code>trial_col</code>	Optional trial column.
<code>aoi_col</code>	Optional AOI column.

target_aoi	Optional character vector identifying the target AOI(s).
fixation_col	Optional fixation indicator column.
saccade_col	Optional saccade indicator column.
event_col	Optional custom event indicator column.
event_value	Optional value(s) in event_col defining the custom event. If NULL and alignment_event = "custom", event_col is interpreted as a logical-like indicator.
alignment_event	Alignment event. Options are "first_target_entry", "first_fixation_to_target", "first_saccade_to_aoi", "first_fixation", and "custom".
baseline_window	Optional numeric vector of length two giving the aligned-time baseline window, for example c(-200, 0).
analysis_window	Optional numeric vector of length two giving the aligned-time analysis window, for example c(0, 1000).
keep_unaligned	Logical. If FALSE, groups without an alignment event are removed from aligned_data. Their status remains in event_table.
name	Character label stored in the returned object.

Value

A list with class gp3_fixation_aligned_data.

```
prepare_gazepoint_gazer_data
```

Prepare Gazepoint master data for gazer-style workflows

Description

Convert a gp3tools master table into a dependency-free, gazer-friendly sample-level table. The returned data frame keeps one row per gaze sample and creates standard participant, trial, time, gaze-coordinate, pupil, AOI, fixation, validity, trackloss, and screen-bound status columns.

Usage

```
prepare_gazepoint_gazer_data(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  x_col = NULL,
  y_col = NULL,
  pupil_col = NULL,
  media_col = NULL,
  condition_col = NULL,
```

```

    aoi_col = NULL,
    fixation_col = NULL,
    validity_cols = NULL,
    trackloss_col = NULL,
    screen_x_range = c(0, 1),
    screen_y_range = c(0, 1),
    missing_aoi_label = "missing_aoi",
    keep_original_cols = TRUE
  )

```

Arguments

<code>data</code>	A Gazepoint master table or sample-level gaze data frame.
<code>participant_col</code>	Participant/subject identifier column.
<code>trial_col</code>	Trial identifier column. If NULL, a trial identifier is created from <code>media_col</code> when available.
<code>time_col</code>	Sample time column.
<code>x_col</code>	Gaze x-coordinate column.
<code>y_col</code>	Gaze y-coordinate column.
<code>pupil_col</code>	Optional pupil column.
<code>media_col</code>	Optional media/stimulus identifier column.
<code>condition_col</code>	Optional condition/grouping column.
<code>aoi_col</code>	Optional AOI label/state column.
<code>fixation_col</code>	Optional fixation identifier column.
<code>validity_cols</code>	Optional validity columns used to define trackloss.
<code>trackloss_col</code>	Optional existing trackloss column. If supplied, it is used directly after coercion to logical.
<code>screen_x_range</code>	Numeric length-2 vector defining the screen x range.
<code>screen_y_range</code>	Numeric length-2 vector defining the screen y range.
<code>missing_aoi_label</code>	Label used for missing AOI values.
<code>keep_original_cols</code>	Logical. If TRUE, original columns are retained after the standard adapter columns.

Value

A tibble with class `gp3_gazer_data`.

```
prepare_gazepoint_gca_data
```

Prepare Gazepoint Growth Curve Analysis data

Description

Prepare binned pupil time-course data for Growth Curve Analysis (GCA). The function creates orthogonal polynomial time terms, preserves subject and condition information, and standardises key columns for later mixed-model fitting.

Usage

```
prepare_gazepoint_gca_data(
  data,
  pupil_col = "mean_pupil",
  time_col = "time_bin_center_ms",
  subject_col = "subject",
  condition_col = "condition",
  degree = 3,
  orthogonal = TRUE,
  time_window = NULL,
  valid_samples_col = "n_valid_samples",
  min_valid_samples = 1,
  weights_col = NULL,
  missing_condition_label = "all_data",
  drop_missing = TRUE
)
```

Arguments

<code>data</code>	A binned pupil time-course data frame, usually created by <code>prepare_gazepoint_pupil_gamm_data()</code> .
<code>pupil_col</code>	Name of the pupil outcome column.
<code>time_col</code>	Name of the time column.
<code>subject_col</code>	Name of the subject column.
<code>condition_col</code>	Name of the condition column. If unavailable or entirely missing, a single condition label is used.
<code>degree</code>	Number of polynomial time terms to create.
<code>orthogonal</code>	Logical. If TRUE, use orthogonal polynomial terms from <code>stats::poly()</code> . If FALSE, use raw powers of z-scored time.
<code>time_window</code>	Optional numeric vector of length 2 giving the time window to retain.
<code>valid_samples_col</code>	Optional column containing valid sample counts.
<code>min_valid_samples</code>	Minimum valid samples required per row when <code>valid_samples_col</code> is available.

weights_col Optional weights column to preserve for later modelling.
 missing_condition_label Label used when condition values are missing.
 drop_missing Logical. If TRUE, rows with missing outcome/time/subject values are removed.

Value

A tibble of class gp3_gca_data with standard GCA columns and polynomial time terms.

```
prepare_gazepoint_gpbiometrics_bridge
```

Prepare gp3tools gaze output for a gpbiometrics workflow

Description

Standardises participant, trial, time, gaze, AOI, pupil, and validity fields without changing the underlying measurements.

Usage

```
prepare_gazepoint_gpbiometrics_bridge(  
  gaze_data,  
  participant_col = NULL,  
  trial_col = NULL,  
  time_col = NULL,  
  time_unit = c("auto", "seconds", "milliseconds"),  
  x_col = NULL,  
  y_col = NULL,  
  aoi_col = NULL,  
  pupil_col = NULL,  
  validity_col = NULL,  
  keep_cols = NULL  
)
```

Arguments

gaze_data Sample-level gp3tools gaze/master data.
 participant_col, trial_col, time_col Explicit source columns.
 time_unit Time unit for the source time column.
 x_col, y_col, aoi_col, pupil_col, validity_col Optional source columns.
 keep_cols Additional source columns to retain.

Value

A data frame of class "gazepoint_gpbiometrics_bridge".

`prepare_gazepoint_hddm_export`*Prepare a trial-level export for Python HDDM*

Description

Prepares a clean trial-level CSV-compatible data frame for Python HDDM or related drift-diffusion modelling workflows. This function does not fit HDDM.

Usage

```
prepare_gazepoint_hddm_export(  
  data,  
  subject,  
  rt,  
  response,  
  predictors = NULL,  
  zscore_within_subject = TRUE,  
  drop_missing = TRUE,  
  file = NULL  
)
```

Arguments

<code>data</code>	Trial-level data frame.
<code>subject</code>	Subject identifier column.
<code>rt</code>	Response-time column.
<code>response</code>	Binary response column coded as 0/1.
<code>predictors</code>	Optional continuous predictors to include.
<code>zscore_within_subject</code>	Logical; z-score predictors within subject.
<code>drop_missing</code>	Logical; drop rows with missing required values.
<code>file</code>	Optional CSV path. If supplied, the export is written to disk.

Value

A data frame ready for HDDM-style workflows.

```
prepare_gazepoint_heatmap_data
```

Prepare gaze or fixation coordinates for heatmap plotting

Description

`prepare_gazepoint_heatmap_data()` standardises gaze or fixation coordinates for spatial heatmap visualisation. It supports normalised Gazepoint-style coordinates in the range 0–1 and pixel coordinates.

Usage

```
prepare_gazepoint_heatmap_data(
  data,
  x_col,
  y_col,
  weight_col = NULL,
  display_width = NULL,
  display_height = NULL,
  coordinate_space = c("auto", "normalized", "pixel")
)
```

Arguments

<code>data</code>	A data frame containing gaze or fixation coordinates.
<code>x_col, y_col</code>	Character strings giving the x and y coordinate columns.
<code>weight_col</code>	Optional character string giving a non-negative weight column, such as fixation duration. If NULL, each point receives equal weight.
<code>display_width, display_height</code>	Display width and height in pixels. For normalised coordinates, these values are used to convert coordinates to pixel space. If omitted for normalised coordinates, a unit display is used. If omitted for pixel coordinates, bounds are inferred from the observed coordinates.
<code>coordinate_space</code>	One of "auto", "normalized", or "pixel". "auto" treats coordinates as normalised only when all finite x and y values fall between 0 and 1.

Value

A data frame with the original retained rows and standardised columns `.gp3_x_px`, `.gp3_y_px`, and `.gp3_weight`.

Examples

```
gaze <- data.frame(
  x = c(0.20, 0.25, 0.75),
  y = c(0.30, 0.35, 0.60),
```

```

    duration = c(120, 200, 80)
  )

  prepare_gazepoint_heatmap_data(
    gaze,
    x_col = "x",
    y_col = "y",
    weight_col = "duration",
    display_width = 1920,
    display_height = 1080
  )

```

```
prepare_gazepoint_hmm_data
```

Prepare Gazepoint AOI/state sequences for HMM-style workflows

Description

Convert ordered Gazepoint AOI/state observations into a dependency-free hidden-Markov-model-ready structure. The helper creates ordered sequence data, transition tables, initial-state probabilities, transition-probability matrices, and observation/emission summaries. It does not fit an HMM and does not import external HMM packages.

Usage

```

prepare_gazepoint_hmm_data(
  data,
  state_col = NULL,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  observation_cols = NULL,
  sequence_id_cols = NULL,
  covariate_cols = NULL,
  state_order = NULL,
  exclude_states = c("missing", "missing_aoi", "missing_coordinate", "trackloss",
    "track_loss"),
  missing_state_label = NULL,
  scale_numeric_observations = FALSE,
  include_terminal_state = FALSE,
  terminal_state_label = "END",
  name = "gazepoint_hmm_data"
)

```

Arguments

<code>data</code>	A data frame containing ordered AOI/state observations.
-------------------	---

state_col	AOI/state column. If NULL, common AOI/state columns are detected automatically.
participant_col	Optional participant/subject column.
trial_col	Optional trial/sequence column.
time_col	Optional time/order column.
observation_cols	Optional observation columns to carry into the HMM data. If NULL, common gaze, pupil, fixation, and validity columns are detected automatically.
sequence_id_cols	Optional character vector of columns defining separate sequences. If NULL, participant and trial columns are used when available.
covariate_cols	Optional covariate columns to carry into the HMM data.
state_order	Optional preferred hidden-state order.
exclude_states	Character vector of states to exclude before sequence construction.
missing_state_label	Optional label used to retain missing states. If NULL, missing/blank states are removed.
scale_numeric_observations	Logical. If TRUE, z-scored versions of numeric observation columns are added with suffix <code>_z</code> .
include_terminal_state	Logical. If TRUE, each sequence contributes a final transition to <code>terminal_state_label</code> .
terminal_state_label	Terminal-state label.
name	Character label stored in the returned object.

Value

A list with class `gp3_hmm_data`.

```
prepare_gazepoint_multimodal_data
```

Prepare multimodal Gazepoint and external face-window data

Description

Joins facial-behaviour window summaries with optional Gazepoint-derived summaries, response variables, or covariates. The helper is intentionally conservative: it prepares transparent analysis tables and optional scaled predictors, but it does not infer emotional states or causal effects.

Usage

```
prepare_gazepoint_multimodal_data(
  face_windows,
  gaze_data = NULL,
  response_data = NULL,
  by = NULL,
  gaze_by = NULL,
  response_by = NULL,
  predictor_cols = NULL,
  outcome_cols = NULL,
  covariate_cols = NULL,
  scale_predictors = TRUE,
  scaled_suffix = "_z",
  drop_missing_outcomes = FALSE,
  keep_all = TRUE
)
```

Arguments

<code>face_windows</code>	A data frame, usually returned by <code>summarize_gazepoint_face_windows()</code> or <code>summarize_gazepoint_face_reactivity()</code> .
<code>gaze_data</code>	Optional Gazepoint-derived data frame to join.
<code>response_data</code>	Optional response/outcome data frame to join.
<code>by</code>	Character vector of join columns shared across tables. If <code>NULL</code> , common identifier-like columns are detected.
<code>gaze_by</code>	Optional named join mapping passed to <code>merge()</code> for <code>gaze_data</code> . If <code>NULL</code> , <code>by</code> is used.
<code>response_by</code>	Optional named join mapping passed to <code>merge()</code> for <code>response_data</code> . If <code>NULL</code> , <code>by</code> is used.
<code>predictor_cols</code>	Optional predictor columns to mark for modelling. If <code>NULL</code> , numeric non-identifier columns from the joined table are used.
<code>outcome_cols</code>	Optional outcome columns to mark for modelling.
<code>covariate_cols</code>	Optional covariate columns to mark for modelling.
<code>scale_predictors</code>	Should numeric predictor columns be z-scaled?
<code>scaled_suffix</code>	Suffix for scaled predictor columns.
<code>drop_missing_outcomes</code>	Should rows with missing values in any <code>outcome_cols</code> be dropped?
<code>keep_all</code>	Should all rows from <code>face_windows</code> be retained during joins?

Value

A tibble with class `gp3_multimodal_data`. Attributes contain join settings, selected predictors, outcomes, covariates, and scaling metadata.

Examples

```

face_windows <- data.frame(
  participant_id = c("P001", "P002"),
  trial_id = c(1, 1),
  AU12_r_mean = c(0.2, 0.3),
  face_confidence_mean = c(0.95, 0.94)
)

responses <- data.frame(
  participant_id = c("P001", "P002"),
  trial_id = c(1, 1),
  rating = c(4, 5)
)

prepare_gazepoint_multimodal_data(
  face_windows,
  response_data = responses,
  by = c("participant_id", "trial_id"),
  outcome_cols = "rating",
  predictor_cols = "AU12_r_mean"
)

```

```
prepare_gazepoint_pupillometryr_data
```

Prepare Gazepoint master data for pupillometryR-style workflows

Description

Convert a gp3tools master table into a dependency-free, pupillometryR-friendly sample-level pupil table. The returned data frame keeps one row per sample and creates standard participant, trial, time, pupil, condition, event, baseline, validity, trackloss, and status columns.

Usage

```

prepare_gazepoint_pupillometryr_data(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  pupil_col = NULL,
  media_col = NULL,
  condition_col = NULL,
  event_col = NULL,
  baseline_col = NULL,
  validity_cols = NULL,
  pupil_status_col = NULL,
  trackloss_col = NULL,
  invalid_pupil_status = c("missing", "artifact", "blink", "trackloss", "track_loss",

```

```

      "invalid", "excluded", "bad", "outlier"),
    keep_original_cols = TRUE
  )

```

Arguments

<code>data</code>	A Gazepoint master table or sample-level gaze/pupil data frame.
<code>participant_col</code>	Participant/subject identifier column.
<code>trial_col</code>	Trial identifier column. If NULL, a trial identifier is created from <code>media_col</code> when available.
<code>time_col</code>	Sample time column.
<code>pupil_col</code>	Pupil column to export.
<code>media_col</code>	Optional media/stimulus identifier column.
<code>condition_col</code>	Optional condition/grouping column.
<code>event_col</code>	Optional event/marker column.
<code>baseline_col</code>	Optional baseline-period indicator column.
<code>validity_cols</code>	Optional validity columns used to define trackloss.
<code>pupil_status_col</code>	Optional pupil-status column used to mark invalid pupil samples.
<code>trackloss_col</code>	Optional existing trackloss column. If supplied, it is used directly after coercion to logical.
<code>invalid_pupil_status</code>	Character values in <code>pupil_status_col</code> treated as invalid pupil samples.
<code>keep_original_cols</code>	Logical. If TRUE, original columns are retained after the standard adapter columns.

Value

A tibble with class `gp3_pupillometryr_data`.

```
prepare_gazepoint_pupil_gamm_data
```

Prepare Gazepoint pupil GAMM data

Description

Prepare binned pupil time-course data for GAMM modelling with `mgcv::bam()`. The function aggregates processed sample-level pupil data into subject-by- condition-by-time-bin rows and creates an `AR.start` indicator for autoregressive GAMM models.

Usage

```
prepare_gazepoint_pupil_gamm_data(
  data,
  pupil_col = NULL,
  time_col = "time",
  subject_col = "subject",
  condition_col = "condition",
  x_col = NULL,
  y_col = NULL,
  group_cols = c("subject", "condition"),
  bin_width_ms = 50,
  time_window = NULL,
  min_valid_samples = 1,
  missing_condition_label = "all_data"
)
```

Arguments

<code>data</code>	A Gazepoint sample-level data frame, usually after pupil preprocessing, interpolation, baseline correction, and optional smoothing.
<code>pupil_col</code>	Name of the pupil column to aggregate. If NULL, the function tries common processed pupil columns such as <code>pupil_smoothed</code> , <code>pupil_baseline_corrected</code> , <code>pupil_interpolated</code> , <code>pupil_clean</code> , and <code>pupil_for_preprocessing</code> .
<code>time_col</code>	Name of the time column in milliseconds. If the requested column is not available, the function tries common alternatives.
<code>subject_col</code>	Name of the subject column. If unavailable, the function tries common participant identifiers.
<code>condition_col</code>	Name of the condition column. If unavailable or entirely missing, a single condition label is used.
<code>x_col</code>	Optional gaze x-coordinate column. If NULL, common x-coordinate columns are auto-detected when available.
<code>y_col</code>	Optional gaze y-coordinate column. If NULL, common y-coordinate columns are auto-detected when available.
<code>group_cols</code>	Columns defining independent time series before binning. Defaults to <code>c("subject", "condition")</code> .
<code>bin_width_ms</code>	Width of time bins in milliseconds.
<code>time_window</code>	Optional numeric vector of length 2 giving the time window to retain before binning.
<code>min_valid_samples</code>	Minimum number of valid pupil samples required for a bin to be retained.
<code>missing_condition_label</code>	Label used when condition values are missing or when no usable condition column is available.

Value

A tibble with binned pupil time-course data for GAMM modelling.

```
prepare_gazepoint_pupil_window_model_data
```

Prepare pupil-window data for confirmatory mixed models

Description

Prepare pupil-window summaries or pupil trial-feature tables for confirmatory window-level modelling. The function standardises subject, condition, window, trial/media identifiers, outcome, valid-sample counts, total-sample counts, valid-sample proportions, weights, and model-readiness status columns.

Usage

```
prepare_gazepoint_pupil_window_model_data(
  data,
  outcome_col = "mean_pupil",
  subject_col = "subject",
  condition_col = "condition",
  window_col = "window_label",
  window_start_col = "window_start_ms",
  window_end_col = "window_end_ms",
  trial_col = NULL,
  media_col = "media_id",
  valid_samples_col = "n_valid_pupil",
  total_samples_col = "n_samples",
  min_valid_samples = 5,
  min_valid_prop = 0.7,
  drop_invalid = TRUE,
  missing_condition_label = "all_data",
  outcome_label = "pupil"
)
```

Arguments

<code>data</code>	Pupil-window summary data.
<code>outcome_col</code>	Column containing the pupil outcome to model. The default is <code>mean_pupil</code> .
<code>subject_col</code>	Subject/participant column.
<code>condition_col</code>	Optional condition column. Common aliases such as <code>condition</code> , <code>Condition</code> , and <code>CONDITION</code> are detected when available.
<code>window_col</code>	Pupil-window label column.
<code>window_start_col</code>	Optional window-start column.

window_end_col	Optional window-end column.
trial_col	Optional trial identifier column.
media_col	Optional media/stimulus identifier column. Common aliases such as media_id and MEDIA_ID are detected when available.
valid_samples_col	Optional column containing the number of valid pupil samples in the window. Common aliases such as n_valid_pupil and n_valid_samples are detected when available.
total_samples_col	Optional column containing the total number of samples in the window. Common aliases such as n_samples and n_window_samples are detected when available.
min_valid_samples	Minimum acceptable number of valid pupil samples.
min_valid_prop	Minimum acceptable valid-sample proportion.
drop_invalid	Logical. If TRUE, rows with invalid or low-quality model inputs are removed.
missing_condition_label	Label used when condition is missing.
outcome_label	Label stored in the output to identify the modelled pupil outcome.

Value

A tibble of pupil-window rows prepared for confirmatory modelling.

```
prepare_gazepoint_semimarkov_data
```

Prepare Gazept AOI sequences for semi-Markov modelling

Description

Convert ordered AOI/state observations into state-visit and transition-level semi-Markov data. Consecutive repeated states can be collapsed into dwell episodes, producing one row per state visit with dwell duration and next-state information.

Usage

```
prepare_gazepoint_semimarkov_data(
  data,
  state_col = NULL,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  duration_col = NULL,
  sequence_id_cols = NULL,
  covariate_cols = NULL,
```

```

exclude_states = c("missing", "missing_aoi", "missing_coordinate", "trackloss",
  "track_loss"),
missing_state_label = NULL,
collapse_repeated_states = TRUE,
include_terminal_states = TRUE,
terminal_next_state_label = "END",
name = "gazepoint_semimarkov_data"
)

```

Arguments

<code>data</code>	A data frame containing ordered AOI/state observations.
<code>state_col</code>	AOI/state column. If NULL, common AOI/state columns are detected automatically.
<code>participant_col</code>	Optional participant/subject column.
<code>trial_col</code>	Optional trial/sequence column.
<code>time_col</code>	Optional time/order column.
<code>duration_col</code>	Optional sample-duration column. If supplied, dwell durations are computed by summing this column within each state visit.
<code>sequence_id_cols</code>	Optional character vector of columns defining separate sequences. If NULL, participant and trial columns are used when available.
<code>covariate_cols</code>	Optional character vector of covariate columns to carry into the state-visit and transition tables using the first value within each state visit.
<code>exclude_states</code>	Character vector of states to exclude before creating state visits.
<code>missing_state_label</code>	Optional label used to retain missing states. If NULL, missing/blank states are removed.
<code>collapse_repeated_states</code>	Logical. If TRUE, consecutive repeated states within a sequence are collapsed into a single dwell episode.
<code>include_terminal_states</code>	Logical. If TRUE, the final state visit in each sequence is retained as a transition to <code>terminal_next_state_label</code> .
<code>terminal_next_state_label</code>	Label used for the terminal next state.
<code>name</code>	Character label stored in the returned object.

Value

A list with class `gp3_semimarkov_data`.

```
prepare_gazepoint_timecourse_test_data
```

Prepare time-course data for Gazepoint cluster-permutation testing

Description

`prepare_gazepoint_timecourse_test_data()` converts a long-format subject-by-condition-by-time data frame into the internal column contract used by `run_gazepoint_cluster_permutation()`. It is intended for conservative two-condition, within-subject, one-dimensional time-course workflows.

Usage

```
prepare_gazepoint_timecourse_test_data(
  data,
  subject_col,
  condition_col,
  time_col,
  outcome_col,
  condition_order = NULL,
  aggregate_fun = mean,
  complete_only = TRUE
)
```

Arguments

<code>data</code>	A long-format data frame.
<code>subject_col</code>	Column identifying participants or paired units.
<code>condition_col</code>	Column identifying the two within-subject conditions.
<code>time_col</code>	Numeric time or time-bin column.
<code>outcome_col</code>	Numeric outcome column.
<code>condition_order</code>	Optional character vector of length two giving the condition order used for contrasts.
<code>aggregate_fun</code>	Function used when duplicate subject-condition-time rows are present. Defaults to mean.
<code>complete_only</code>	If TRUE, keep only subject-by-time cells with both conditions present.

Value

A data frame with internal cluster columns: `.gp3_cluster_subject`, `.gp3_cluster_condition`, `.gp3_cluster_time_bin`, and `.gp3_cluster_outcome`.

Examples

```
d <- data.frame(
  subject = rep(1:4, each = 6),
  condition = rep(rep(c("A", "B"), each = 3), 4),
  time = rep(1:3, 8),
  value = rnorm(24)
)

prepare_gazepoint_timecourse_test_data(
  d,
  subject_col = "subject",
  condition_col = "condition",
  time_col = "time",
  outcome_col = "value"
)
```

```
prepare_gazepoint_traminer_data
```

Prepare AOI sequences for TraMineR-style workflows

Description

Convert long AOI observations into one row per sequence and one column per ordered state position. If **TraMineR** is installed and `as_traminer = TRUE`, the function also returns a TraMineR sequence object.

Usage

```
prepare_gazepoint_traminer_data(
  data,
  aoi_col,
  sequence_cols,
  time_col = NULL,
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  state_prefix = "state_",
  as_traminer = FALSE
)
```

Arguments

<code>data</code>	Long-format AOI data.
<code>aoi_col</code>	AOI/state column.
<code>sequence_cols</code>	Columns defining each sequence.
<code>time_col</code>	Optional ordering column.

include_missing	Should missing AOIs be kept as a state?
missing_label	Label used for retained missing AOIs.
collapse_repeats	Should consecutive repeated states be collapsed?
state_prefix	Prefix for wide state columns.
as_traminer	Should TraMineR::seqdef() be called if available?

Value

A list containing wide data, state columns, alphabet, and optionally a TraMineR sequence object.

```
preprocess_gazepoint_signals
```

Run an integrated Gazepoint signal-preprocessing workflow

Description

Orchestrate transparent blink, pupil, coordinate, downsampling, and velocity-based fixation-processing steps while preserving the original input columns. Every requested operation is recorded in a decision log.

Usage

```
preprocess_gazepoint_signals(
  data,
  id_col = "USER_ID",
  group_cols = NULL,
  time_col = "TIME",
  x_col = "FPOGX",
  y_col = "FPOGY",
  left_pupil_col = NULL,
  right_pupil_col = NULL,
  pupil_col = NULL,
  pupil_mode = c("mean", "regression", "none"),
  detect_blinks = TRUE,
  interpolate_blinks = TRUE,
  smooth_pupil = TRUE,
  smooth_coordinates = TRUE,
  downsample_factor = 1L,
  detect_fixations = TRUE,
  blink_args = list(),
  interpolation_args = list(),
  pupil_args = list(),
  pupil_smoothing_args = list(),
  coordinate_smoothing_args = list(),
```

```

    downsampling_args = list(),
    fixation_args = list()
)

```

Arguments

<code>data</code>	A sample-level Gazepoint or gp3tools data frame.
<code>id_col</code>	Participant identifier column.
<code>group_cols</code>	Optional additional columns defining independent time series, such as trial or stimulus.
<code>time_col</code>	Timestamp column.
<code>x_col, y_col</code>	Gaze-coordinate columns.
<code>left_pupil_col, right_pupil_col</code>	Optional binocular pupil columns. When NULL, common Gazepoint and gp3tools names are detected.
<code>pupil_col</code>	Optional existing monocular or fused pupil column used when <code>pupil_mode = "none"</code> .
<code>pupil_mode</code>	Binocular fusion mode: "mean", "regression", or "none".
<code>detect_blinks</code>	Should <code>detect_gazepoint_blinks()</code> be run?
<code>interpolate_blinks</code>	Should detected blink intervals be interpolated?
<code>smooth_pupil</code>	Should <code>smooth_gazepoint_pupil()</code> be run?
<code>smooth_coordinates</code>	Should <code>smooth_gazepoint_coordinate()</code> be run?
<code>downsample_factor</code>	Positive integer downsampling factor. Use 1 to retain the original sample count.
<code>detect_fixations</code>	Should <code>detect_gazepoint_fixations_velocity()</code> be run on the final full-resolution coordinates?
<code>blink_args</code>	Named list overriding blink-detection defaults.
<code>interpolation_args</code>	Named list overriding blink-interpolation defaults.
<code>pupil_args</code>	Named list overriding binocular fusion defaults.
<code>pupil_smoothing_args</code>	Named list overriding pupil-smoothing defaults.
<code>coordinate_smoothing_args</code>	Named list overriding coordinate-smoothing defaults.
<code>downsampling_args</code>	Named list overriding downsampling defaults.
<code>fixation_args</code>	Named list overriding fixation-detection defaults.

Value

An object of class "gp3_signal_preprocessing_result" containing processed data, detected blinks, detected fixations, diagnostic tables, a `decision_log`, and resolved settings.

Examples

```
pupil <- data.frame(
  USER_ID = rep("P01", 30),
  trial = rep("T01", 30),
  TIME = seq(0, 0.29, by = 0.01),
  FPOGX = c(rep(0.25, 15), rep(0.75, 15)),
  FPOGY = 0.50,
  LPupil = c(rep(3.2, 10), NA, NA, rep(3.2, 18)),
  RPupil = c(rep(3.1, 10), NA, NA, rep(3.1, 18))
)

result <- preprocess_gazepoint_signals(
  pupil,
  group_cols = "trial",
  downsample_factor = 2
)

result$decision_log
```

read_gazepoint

*Read a Gazepoint all-gaze or fixation CSV export***Description**

Reads Gazepoint all-gaze and fixation CSV exports, standardises timestamped column names, and removes empty trailing columns produced by Gazepoint exports.

Usage

```
read_gazepoint(path, standardise_names = TRUE, drop_empty_cols = TRUE)
```

Arguments

path Path to a Gazepoint CSV export.

standardise_names Logical. If TRUE, standardise TIME(...) and TIMETICK(...) headers.

drop_empty_cols Logical. If TRUE, remove empty trailing or unnamed columns created by Gazepoint export formatting.

Value

A tibble with attributes gp3_file_type and gp3_source_file.

read_gazepoint_face_export

Read external facial-analysis exports

Description

Reads one or more external facial-analysis CSV files into a single tidy table. The helper is designed for outputs produced outside Gazepoint, such as OpenFace-, py-feat-, MediaPipe-, FaceReader-, or generic frame-level facial behaviour exports. It does not infer facial expressions from Gazepoint CSV files.

Usage

```
read_gazepoint_face_export(
  path,
  source = c("auto", "openface", "pyfeat", "mediapipe", "facereader", "generic"),
  participant_id = NULL,
  session_id = NULL,
  recursive = TRUE,
  trim_names = TRUE,
  encoding = "UTF-8",
  na = c("", "NA", "NaN"),
  ...
)
```

Arguments

path	Path to one CSV file, several CSV files, or a directory containing CSV files.
source	Facial-analysis source. Use "auto" to infer a likely source from column names, or one of "openface", "pyfeat", "mediapipe", "facereader", or "generic".
participant_id	Optional participant identifier. Either length one or the same length as the number of files read.
session_id	Optional session identifier. Either length one or the same length as the number of files read.
recursive	If path is a directory, should CSV files be searched recursively?
trim_names	Should leading and trailing whitespace be removed from column names?
encoding	File encoding passed to <code>utils::read.csv()</code> .
na	Character values treated as missing.
...	Additional arguments passed to <code>utils::read.csv()</code> .

Value

A tibble with metadata columns identifying the source file and detected source. The returned object has class `gp3_face_export`.

Examples

```

tmp <- tempfile(fileext = ".csv")
write.csv(
  data.frame(
    frame = 1:2,
    timestamp = c(0, 0.033),
    confidence = c(0.98, 0.97),
    success = c(1, 1),
    AU12_r = c(0.1, 0.2)
  ),
  tmp,
  row.names = FALSE
)

read_gazepoint_face_export(tmp)

```

read_gazepoint_folder *Read multiple Gazepoint CSV exports from a folder*

Description

Reads all Gazepoint all-gaze or fixation CSV exports in a folder that match a filename pattern and combines them into one tibble.

Usage

```

read_gazepoint_folder(
  folder,
  pattern = "\\*.csv$",
  source_col = "USER_FILE",
  recursive = FALSE,
  ...
)

```

Arguments

folder	Path to the folder containing Gazepoint CSV exports.
pattern	Regular expression used to select files. For example, "_all_gaze*.csv\$" or "_fixations*.csv\$".
source_col	Name of the column storing the source filename.
recursive	Logical. If TRUE, search subfolders recursively.
...	Additional arguments passed to read_gazepoint().

Value

A tibble containing all matching files combined row-wise.

```
read_gazepoint_summary
```

Read a Gazepoint Analysis Data Summary export

Description

Parses the multi-section Data_Summary_export_*.csv file into metadata, aoi_summary, and aoi_by_user tables.

Usage

```
read_gazepoint_summary(path)
```

Arguments

path Path to Data_Summary_export_*.csv.

Value

A list with metadata, aoi_summary, and aoi_by_user.

```
recalibrate_gazepoint_gaze
```

Offline gaze recalibration using known target coordinates

Description

Apply an offline drift-correction shift to Gazepoint gaze coordinates using known fixation/check-target coordinates. For each group, the helper estimates the horizontal and vertical gaze offset from valid target samples and applies the correction to all gaze samples in the same group.

Usage

```
recalibrate_gazepoint_gaze(
  data,
  x_col,
  y_col,
  target_x_col,
  target_y_col,
  time_col = NULL,
  grouping_cols = NULL,
  calibration_col = NULL,
  calibration_value = NULL,
  method = c("median_shift", "mean_shift"),
  min_valid_points = 3L,
```

```

    max_shift = NULL,
    output_x_col = "gaze_x_recalibrated",
    output_y_col = "gaze_y_recalibrated",
    dx_col = "gaze_recalibration_dx",
    dy_col = "gaze_recalibration_dy",
    shift_col = "gaze_recalibration_shift",
    error_before_col = "gaze_error_before_recalibration",
    error_after_col = "gaze_error_after_recalibration",
    status_col = "gaze_recalibration_status",
    overwrite = FALSE,
    name = "gazepoint_gaze_recalibration"
)

```

Arguments

<code>data</code>	A data frame containing gaze and target coordinates.
<code>x_col</code>	Horizontal gaze coordinate column.
<code>y_col</code>	Vertical gaze coordinate column.
<code>target_x_col</code>	Known horizontal target coordinate column.
<code>target_y_col</code>	Known vertical target coordinate column.
<code>time_col</code>	Optional time column used only for stable ordering.
<code>grouping_cols</code>	Optional grouping columns used to estimate one correction per participant, trial, block, stimulus, or other unit.
<code>calibration_col</code>	Optional column identifying rows to use for estimating the correction.
<code>calibration_value</code>	Optional value in <code>calibration_col</code> identifying calibration/check rows. If <code>calibration_col</code> is supplied and <code>calibration_value = NULL</code> , logical TRUE rows are used.
<code>method</code>	Shift estimator. "median_shift" uses median target-minus-gaze offsets; "mean_shift" uses mean offsets.
<code>min_valid_points</code>	Minimum valid target/gaze pairs required per group.
<code>max_shift</code>	Optional maximum Euclidean correction shift. If exceeded, the shift is reported but not applied.
<code>output_x_col</code>	Corrected horizontal gaze output column.
<code>output_y_col</code>	Corrected vertical gaze output column.
<code>dx_col</code>	Estimated horizontal correction column.
<code>dy_col</code>	Estimated vertical correction column.
<code>shift_col</code>	Estimated Euclidean shift-distance column.
<code>error_before_col</code>	Row-wise gaze-to-target error before correction.
<code>error_after_col</code>	Row-wise gaze-to-target error after correction.
<code>status_col</code>	Row-level recalibration status column.
<code>overwrite</code>	Logical. If FALSE, existing output columns are protected.
<code>name</code>	Character label stored in object attributes.

Details

This helper is useful only when known target coordinates are available, for example from calibration checks, fixation targets, validation targets, or drift-check trials.

Value

A tibble with recalibrated gaze columns and recalibration attributes.

```
recommend_gazepoint_exclusions
```

Recommend trial and participant exclusions

Description

Create explicit trial-level and participant-level exclusion recommendations from Gazepoint sample-level quality information. The helper can use validity flags, gaze-coordinate missingness, pupil missingness, and optional artifact flags to produce transparent exclusion tables.

Usage

```
recommend_gazepoint_exclusions(
  data,
  participant_col,
  trial_col = NULL,
  condition_col = NULL,
  validity_col = NULL,
  x_col = NULL,
  y_col = NULL,
  pupil_col = NULL,
  artifact_col = NULL,
  min_trial_samples = 10L,
  max_trial_missing_prop = 0.5,
  max_trial_artifact_prop = 0.5,
  min_participant_trials = 2L,
  min_participant_valid_trials = 1L,
  max_participant_missing_prop = 0.5,
  max_participant_artifact_prop = 0.5,
  require_both_gaze_coordinates = TRUE,
  name = "gazepoint_exclusion_recommendations"
)
```

Arguments

`data` A data frame containing sample-level or trial-level data.

`participant_col` Participant identifier column.

trial_col	Optional trial identifier column.
condition_col	Optional condition column retained in summaries.
validity_col	Optional logical/numeric/character validity column.
x_col	Optional horizontal gaze coordinate column.
y_col	Optional vertical gaze coordinate column.
pupil_col	Optional pupil column.
artifact_col	Optional logical/numeric/character artifact flag column.
min_trial_samples	Minimum samples required per trial.
max_trial_missing_prop	Maximum missing/unusable sample proportion per trial.
max_trial_artifact_prop	Maximum artifact proportion per trial.
min_participant_trials	Minimum total trials required per participant.
min_participant_valid_trials	Minimum retained trials required per participant.
max_participant_missing_prop	Maximum missing/unusable sample proportion per participant.
max_participant_artifact_prop	Maximum artifact proportion per participant.
require_both_gaze_coordinates	Logical. If both gaze columns are supplied, should a sample be usable only when both coordinates are finite?
name	Character label stored in object attributes.

Details

This function recommends exclusions only. It does not remove rows.

Value

A list with overview, trial recommendations, participant recommendations, an explicit exclusion table, and settings.

recommend_gazepoint_model_family

Recommend model families for Gazepoint-derived metrics

Description

Maps common eye-tracking and pupillometry metrics to suitable statistical model families, transformations, and modelling notes. The helper is intended for planning and reporting; it does not fit a model.

Usage

```
recommend_gazepoint_model_family(metric = NULL)
```

Arguments

metric	Character vector of metric names. If NULL, all available recommendations are returned.
--------	--

Value

A data frame with metric, data property, recommended family, common transformation, and notes.

```
reconstruct_gazepoint_binocular_pupil
```

Reconstruct temporarily unavailable binocular pupil channels

Description

Applies a declared reconstruction policy without overwriting the original eye channels. Linear-regression reconstruction uses separately fitted left-from-right and right-from-left models, explicit eligibility gates, gap-duration restrictions, extrapolation controls, physiological bounds, and row-level provenance.

Usage

```
reconstruct_gazepoint_binocular_pupil(
  data,
  left_col,
  right_col,
  time_col = NULL,
  group_cols = NULL,
  gap_group_cols = NULL,
  method = c("linear_regression", "available_eye", "none"),
  calibration = NULL,
  fallback_group_cols = NULL,
  min_pairs = 30L,
  min_unique = 5L,
  min_r2 = NULL,
  time_unit = c("auto", "milliseconds", "seconds"),
  max_gap_ms = Inf,
  allow_edge_gaps = TRUE,
  allow_extrapolation = FALSE,
  valid_min = NULL,
  valid_max = NULL,
  exclude_flag_cols = NULL,
  prefix = "gp3_binocular",
  overwrite = FALSE
)
```

Arguments

<code>data</code>	A data frame containing pupil channels.
<code>left_col, right_col</code>	Numeric pupil columns.
<code>time_col</code>	Optional numeric time column. Required when <code>max_gap_ms</code> is finite.
<code>group_cols</code>	Primary calibration grouping columns.
<code>gap_group_cols</code>	Optional grouping columns used only to define temporal missing-eye runs. When <code>NULL</code> , <code>group_cols</code> are used. This permits, for example, participant-level calibration with participant-by-trial gap gates.
<code>method</code>	Reconstruction policy: "linear_regression", "available_eye", or "none". The latter two do not synthesize a missing eye; they retain explicit monocular provenance for downstream construction.
<code>calibration</code>	Optional result from <code>fit_gazepoint_binocular_calibration()</code> . When omitted and <code>method = "linear_regression"</code> , calibration is fitted from data.
<code>fallback_group_cols, min_pairs, min_unique, min_r2</code>	Calibration settings used only when calibration is not supplied.
<code>time_unit</code>	Unit for <code>time_col</code> .
<code>max_gap_ms</code>	Maximum contiguous missing-eye run eligible for model-based reconstruction. <code>Inf</code> disables the duration gate. No study-specific cutoff is imposed by default.
<code>allow_edge_gaps</code>	Whether missing runs touching a group boundary may be reconstructed from the simultaneously observed contralateral eye.
<code>allow_extrapolation</code>	Whether predictions outside the calibration predictor range are allowed.
<code>valid_min, valid_max</code>	Optional bounds applied both to observed values used by this workflow and to predictions.
<code>exclude_flag_cols</code>	Optional logical/numeric flag columns. Rows flagged in any supplied column are not reconstructed.
<code>prefix</code>	Prefix for added provenance and reconstructed-channel columns.
<code>overwrite</code>	Whether existing output columns with this prefix may be replaced.

Details

Reconstructed values are predictions from the contralateral eye; they are never labelled as measurements. Temporal interpolation and cross-eye reconstruction solve different missing-data problems and are deliberately kept separate.

Value

The input data with additional observed, final-channel, reconstruction, model, gap, and status columns. Original pupil columns are untouched. A `gp3_binocular_reconstruction` metadata attribute records the declared policy and calibration object.

References

Ong J, He W, Maglanque P, Jiang X, Gillman LM, Vergis A, Hardy K (2025). A Preprocessing Pipeline for Pupillometry Signal from Multimodal iMotion Data. *Sensors*, 25(15), 4737. doi:[10.3390/s25154737](https://doi.org/10.3390/s25154737)

See Also

`construct_gazepoint_combined_pupil()`, `validate_gazepoint_binocular_reconstruction()`

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 13)
dat$pupil_left[40:43] <- NA_real_
rec <- reconstruct_gazepoint_binocular_pupil(
  dat, "pupil_left", "pupil_right", time_col = "timestamp_ms",
  group_cols = "subject", min_pairs = 20
)
table(rec$gp3_binocular_status)
```

regress_gazepoint_pupils

Fuse binocular pupil traces using cross-eye regression

Description

Fits cross-eye regressions within independent sequences and creates a regression-smoothed binocular pupil trace. The function is diagnostic and preprocessing-oriented; it does not imply that one eye causally predicts the other.

Usage

```
regress_gazepoint_pupils(
  master_df,
  lp_col = "LPupil",
  rp_col = "RPupil",
  id_col = "USER_ID",
  group_cols = NULL,
  direction = c("bidirectional", "right_on_left", "left_on_right"),
  output_col = "pupil_regressed",
  residual_col = "pupil_regression_residual",
  min_complete = 10
)
```

Arguments

master_df	A sample-level pupil data frame.
lp_col, rp_col	Left- and right-pupil columns.
id_col	Participant identifier.
group_cols	Optional additional independent-sequence columns.
direction	Regression direction. "bidirectional" fits both right-on-left and left-on-right models.
output_col	Name of the fused pupil column.
residual_col	Name of the right-on-left residual column.
min_complete	Minimum complete binocular samples required for regression. Groups below this threshold use the binocular mean.

Value

The input data with regression-smoothed pupil columns.

Examples

```
pupil <- data.frame(
  USER_ID = rep("P01", 20),
  LPupil = seq(3, 4, length.out = 20),
  RPupil = seq(3.1, 4.1, length.out = 20)
)
regress_gazepoint_pupils(pupil)
```

```
report_gazepoint_cluster_permutation
```

Report a Gazepoint cluster-permutation result

Description

Create a compact, cautious, text-ready report from a cluster-permutation result. The report avoids exact onset/offset claims and describes detected clusters as time ranges surviving the specified cluster procedure.

Usage

```
report_gazepoint_cluster_permutation(result, alpha = 0.05)
```

Arguments

result	A result object returned by run_gazepoint_cluster_permutation().
alpha	Significance threshold.

Value

A list with cluster table, settings, and report text.

report_gazepoint_face_qc

Report external facial-behaviour QC and reporting readiness

Description

Creates a compact markdown or list report from facial-behaviour quality, synchronisation, window-summary, reactivity, and modelling objects. The report is designed for transparent methods/supplementary reporting. It does not infer facial expressions or emotional states.

Usage

```
report_gazepoint_face_qc(
  face_data = NULL,
  quality_audit = NULL,
  sync_audit = NULL,
  window_summary = NULL,
  reactivity_summary = NULL,
  multimodal_model = NULL,
  checklist = NULL,
  output = c("markdown", "list"),
  include_cautions = TRUE
)
```

Arguments

face_data	Optional imported or standardised face-analysis data.
quality_audit	Optional object returned by <code>audit_gazepoint_face_quality()</code> .
sync_audit	Optional object returned by <code>audit_gazepoint_face_sync()</code> .
window_summary	Optional object returned by <code>summarize_gazepoint_face_windows()</code> .
reactivity_summary	Optional object returned by <code>summarize_gazepoint_face_reactivity()</code> .
multimodal_model	Optional object returned by a <code>gp3tools</code> multimodal modelling helper.
checklist	Optional checklist returned by <code>create_gazepoint_face_reporting_checklist()</code> . If NULL, one is created.
output	Output format. "markdown" returns a character vector; "list" returns a structured list.
include_cautions	Should interpretation cautions be included?

Value

A markdown character vector with class `gp3_face_qc_report`, or a list with class `gp3_face_qc_report_list`.

Examples

```
quality <- list(
  overview = data.frame(
    n_rows = 10,
    valid_percent = 95,
    face_quality_status = "pass"
  ),
  issue_summary = data.frame(
    issue = "missing_confidence",
    n_groups_affected = 0
  )
)
class(quality) <- c("gp3_face_quality_audit", "list")

report_gazepoint_face_qc(quality_audit = quality)
```

```
report_gazepoint_missingness
```

Report Gazepoint missingness

Description

Produces a compact, cautious text summary of missingness rates. The report is intended for transparent methods/results documentation and does not recommend exclusions.

Usage

```
report_gazepoint_missingness(
  data,
  cols = NULL,
  group_cols = NULL,
  digits = 1,
  max_variables = 5
)
```

Arguments

<code>data</code>	A data frame or a missingness summary produced by <code>summarize_gazepoint_missingness()</code> .
<code>cols</code>	Optional columns to summarize when data is raw data.
<code>group_cols</code>	Optional grouping columns when data is raw data.
<code>digits</code>	Number of decimal places for percentages.
<code>max_variables</code>	Maximum number of highest-missingness variables to name in the report text.

Value

A list with `summary`, `overall`, and `report_text`.

Examples

```
x <- simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)
report_gazepoint_missingness(
  x,
  cols = c("pupil_left", "pupil_right", "pupil")
)
```

```
report_gazepoint_multiverse
```

Report multiverse-analysis results

Description

Create compact branch-, status-, and term-level summaries from a multiverse result object. This helper is intentionally generic and can summarise the package's multiverse output after it has been tidied to data frames.

Usage

```
report_gazepoint_multiverse(
  multiverse_results,
  branch_col = NULL,
  term_col = NULL,
  estimate_col = NULL,
  p_col = NULL,
  status_col = NULL,
  alpha = 0.05
)
```

Arguments

<code>multiverse_results</code>	A data frame, or a list containing data frames.
<code>branch_col</code>	Optional branch/specification column.
<code>term_col</code>	Optional model term column.
<code>estimate_col</code>	Optional estimate/effect column.
<code>p_col</code>	Optional p-value column.
<code>status_col</code>	Optional status column.
<code>alpha</code>	Significance threshold used for descriptive counts.

Value

A list with branch, status, and term summaries.

```
report_gazepoint_phase_coverage
```

Report task-phase coverage

Description

Produces compact, cautious text describing task-phase data coverage. The report is intended for methods/results documentation and does not define exclusion rules.

Usage

```
report_gazepoint_phase_coverage(
  data,
  phase_col = "task_phase",
  group_cols = NULL,
  time_col = NULL,
  value_cols = NULL,
  digits = 1
)
```

Arguments

data A raw data frame or a summary produced by `summarize_gazepoint_phase_coverage()`.

phase_col, group_cols, time_col, value_cols Arguments used when data is raw data.

digits Number of decimal places used in report percentages.

Value

A list with summary, overall, and report_text.

Examples

```
x <- data.frame(time_ms = c(0, 250, 750, 1250), value = c(1, NA, 3, 4))
windows <- data.frame(phase = c("baseline", "stimulus"), start = c(0, 500), end = c(500, 1500))
segmented <- segment_gazepoint_task_phases(x, "time_ms", windows)
report_gazepoint_phase_coverage(segmented, phase_col = "task_phase", time_col = "time_ms")
```

```
report_gazepoint_qc_overview
```

Report Gazepoint QC overview

Description

Produces compact, cautious text from a QC summary bundle. The report describes available QC outputs and status patterns, but it does not replace the underlying audit, readiness-gate, checklist, or exclusion-recommendation functions.

Usage

```
report_gazepoint_qc_overview(qc_bundle, max_objects = 5)
```

Arguments

`qc_bundle` A `gp3_qc_summary_bundle`, object-summary data frame, or list of objects that can be passed to `collect_gazepoint_qc_summaries()`.

`max_objects` Maximum number of non-pass objects to name in the report.

Value

A list with `summary`, `object_summary`, and `report_text`.

Examples

```
x <- list(overview = data.frame(audit_status = "review", message = "Check coverage."))
report_gazepoint_qc_overview(list(example = x))
```

```
run_gazepoint_aoi_multiverse
```

Run a Gazepoint AOI preprocessing multiverse

Description

Run all AOI preprocessing branches defined by `create_gazepoint_preprocessing_multiverse()`. Each branch creates AOI-window summaries and then prepares binomial AOI GLMM data using the branch-specific denominator and minimum denominator threshold.

Usage

```
run_gazepoint_aoi_multiverse(
  data,
  multiverse,
  branch_ids = NULL,
  windows,
  time_col = "time",
  aoi_col = "aoi_current",
  subject_col = "subject",
  condition_col = NULL,
  group_cols = NULL,
  target_aoi_values,
  distractor_aoi_values = NULL,
  success_col = "n_target_samples",
  outcome_label = "target",
  keep_outputs = TRUE,
  stop_on_error = FALSE
)
```

Arguments

<code>data</code>	A Gazepoint master table or sample-level AOI table.
<code>multiverse</code>	A <code>gp3_preprocessing_multiverse</code> object returned by <code>create_gazepoint_preprocessing_multiverse()</code> .
<code>branch_ids</code>	Optional character vector of AOI branch IDs to run.
<code>windows</code>	Numeric vector or labelled window table passed to <code>summarise_gazepoint_aoi_windows()</code> .
<code>time_col</code>	Time column.
<code>aoi_col</code>	AOI-state column.
<code>subject_col</code>	Subject column.
<code>condition_col</code>	Optional condition column.
<code>group_cols</code>	Optional grouping columns for AOI-window summaries.
<code>target_aoi_values</code>	Target AOI values.
<code>distractor_aoi_values</code>	Optional distractor AOI values.
<code>success_col</code>	Success-count column passed to <code>prepare_gazepoint_aoi_glmm_data()</code> .
<code>outcome_label</code>	Outcome label passed to AOI helpers.
<code>keep_outputs</code>	Logical. If TRUE, keep branch outputs in <code>branch_outputs</code> .
<code>stop_on_error</code>	Logical. If TRUE, stop when a branch fails. If FALSE, record the branch error and continue.

Value

A list with class `gp3_aoi_multiverse_results` containing overview, branch results, optional branch outputs, and settings.

run_gazepoint_cluster_permutation

Run paired cluster-based permutation tests

Description

Run a paired cluster-based permutation test on time-course data prepared by `prepare_gazepoint_cluster_data()`. The function tests whether two conditions diverge over time while controlling cluster-level inference using a permutation distribution of maximum cluster statistics.

Usage

```
run_gazepoint_cluster_permutation(
  data,
  condition_order = NULL,
  n_permutations = 1000,
  cluster_threshold = 2,
  tail = c("two_sided", "greater", "less"),
  cluster_stat = c("sum_abs_t", "sum_t", "size"),
  min_time_bins = 1,
  seed = NULL,
  paired = TRUE
)
```

Arguments

<code>data</code>	Cluster-ready data produced by <code>prepare_gazepoint_cluster_data()</code> .
<code>condition_order</code>	Optional character vector of length 2 defining the two conditions and their order. The tested difference is condition 2 minus condition 1.
<code>n_permutations</code>	Number of sign-flip permutations.
<code>cluster_threshold</code>	Absolute t-statistic threshold for forming candidate clusters. For <code>tail = "greater"</code> or <code>tail = "less"</code> , the same positive threshold is used in the requested direction.
<code>tail</code>	Direction of the test. <code>"two_sided"</code> tests positive and negative clusters. <code>"greater"</code> tests condition 2 greater than condition 1. <code>"less"</code> tests condition 2 less than condition 1.
<code>cluster_stat</code>	Cluster statistic. <code>"sum_abs_t"</code> sums absolute t-statistics within a cluster. <code>"sum_t"</code> sums signed t-statistics and then uses the absolute value for cluster-level inference. <code>"size"</code> uses the number of time bins.
<code>min_time_bins</code>	Minimum number of adjacent time bins required for a cluster to be retained.
<code>seed</code>	Optional random seed for reproducible permutations.
<code>paired</code>	Logical. Currently only paired within-subject sign-flip permutation is supported.

Details

Cluster-based permutation tests are intended for time-course inference. They should not be used to discover a confirmatory time window and then test that same window again in a second confirmatory model.

Value

A list containing observed time-course statistics, observed clusters, the permutation distribution, settings, and status fields.

run_gazepoint_cluster_permutation_anova
Guardrail for cluster-permutation ANOVA

Description

This function intentionally fails safely. Cluster-permutation ANOVA is not implemented as an active inferential engine in gp3tools because it requires additional design, exchangeability, and error-term choices that are outside the currently validated two-condition workflow.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

run_gazepoint_cluster_permutation_covariate_adjusted
Guardrail for covariate-adjusted cluster permutation

Description

This function intentionally fails safely. Covariate-adjusted cluster permutation is not implemented as an active inferential engine in gp3tools.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

```
run_gazepoint_cluster_permutation_lmer
```

Guardrail for mixed-model cluster permutation

Description

This function intentionally fails safely. Mixed-model cluster permutation is not implemented as an active inferential engine in gp3tools.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

```
run_gazepoint_cluster_permutation_parallel
```

Guardrail for parallel cluster permutation

Description

This function intentionally fails safely. Parallel cluster permutation is not implemented as a separate active engine in gp3tools.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

`run_gazepoint_cluster_threshold_sensitivity`*Run threshold-sensitivity checks for Gazepoint cluster permutation*

Description

Re-run `run_gazepoint_cluster_permutation()` across a small set of cluster-forming thresholds and summarize how many clusters are detected.

Usage

```
run_gazepoint_cluster_threshold_sensitivity(  
  data,  
  thresholds = c(1.5, 2, 2.5),  
  ...  
)
```

Arguments

<code>data</code>	Prepared cluster-permutation data.
<code>thresholds</code>	Numeric vector of cluster-forming thresholds.
<code>...</code>	Additional arguments passed to <code>run_gazepoint_cluster_permutation()</code> .

Value

A list containing threshold-level summaries and full result objects.

`run_gazepoint_eyetools_fixation_detection`*Run optional eyetools fixation and saccade detection*

Description

Prepare Gazepoint sample-level gaze data for the optional `eyetools` package and, when `eyetools` is installed, run fixation and/or saccade detection using `eyetools::fixation_dispersion()`, `eyetools::fixation_VTI()` and/or `eyetools::saccade_VTI()`.

Usage

```
run_gazepoint_eyetools_fixation_detection(  
  data,  
  participant_col = NULL,  
  trial_col = NULL,  
  time_col = NULL,  
  x_col = NULL,
```

```

y_col = NULL,
condition_col = NULL,
stimulus_col = NULL,
method = c("dispersion", "vti", "saccade", "all"),
sample_rate = NULL,
threshold = 100,
min_dur = 150,
min_dur_sac = 20,
disp_tol = 100,
NA_tol = 0.25,
smooth = FALSE,
drop_missing = TRUE,
progress = FALSE,
name = "gazepoint_eyetools_fixation_detection"
)

```

Arguments

<code>data</code>	A data frame containing sample-level gaze data.
<code>participant_col</code>	Participant/subject column. If NULL, common names are detected.
<code>trial_col</code>	Trial column. If NULL, common names are detected.
<code>time_col</code>	Time column. If NULL, common names are detected.
<code>x_col</code>	Horizontal gaze coordinate column. If NULL, common names are detected.
<code>y_col</code>	Vertical gaze coordinate column. If NULL, common names are detected.
<code>condition_col</code>	Optional condition/group column.
<code>stimulus_col</code>	Optional stimulus/media column.
<code>method</code>	Detector branch. Options are "dispersion", "vti", "saccade", and "all".
<code>sample_rate</code>	Optional sample rate passed to velocity-threshold functions.
<code>threshold</code>	Velocity threshold passed to velocity-threshold functions.
<code>min_dur</code>	Minimum fixation duration in milliseconds.
<code>min_dur_sac</code>	Minimum saccade duration in milliseconds.
<code>disp_tol</code>	Dispersion tolerance in pixels.
<code>NA_tol</code>	Missing-data tolerance passed to <code>fixation_dispersion()</code> .
<code>smooth</code>	Logical; passed to <code>fixation_VTI()</code> .
<code>drop_missing</code>	Logical. If TRUE, rows with non-finite time, x, or y are removed before detector execution.
<code>progress</code>	Logical. Passed to eyetools detector functions.
<code>name</code>	Character label stored in the returned object.

Details

This helper is an optional external-detector branch. It does not replace the main `gp3tools` summaries or AOI/transition workflows.

Value

A list with class `gp3_eyetools_fixation_detection`.

```
run_gazepoint_gazer_crosscheck
```

Run an optional gazeR pupil-preprocessing cross-check

Description

Prepare Gazepoint pupil data for the optional gazer package and, when gazer is installed, run a conservative pupil-preprocessing cross-check using gazeR-style blink extension, smoothing/interpolation, optional baseline correction, and optional downsampling.

Usage

```
run_gazepoint_gazer_crosscheck(
  data,
  participant_col = NULL,
  trial_col = NULL,
  time_col = NULL,
  pupil_col = NULL,
  condition_col = NULL,
  message_col = NULL,
  blink_col = NULL,
  hz = 60,
  fillback = 100,
  fillforward = 100,
  smooth_n = 5,
  step_first = c("smooth", "interpolate"),
  interpolation_type = "linear",
  maxgap = Inf,
  baseline_window = NULL,
  baseline_event = NULL,
  baseline_dur = 100,
  baseline_method = "sub",
  bin_length = NULL,
  name = "gazepoint_gazer_crosscheck"
)
```

Arguments

<code>data</code>	A data frame containing Gazepoint or gp3tools pupil time-series data.
<code>participant_col</code>	Participant/subject column. If NULL, common names are detected.
<code>trial_col</code>	Trial column. If NULL, common names are detected.
<code>time_col</code>	Time column. If NULL, common names are detected.

pupil_col	Pupil column. If NULL, common processed/raw pupil columns are detected.
condition_col	Optional condition column. If NULL, common names are detected; otherwise "all_data" is used.
message_col	Optional message/event column.
blink_col	Optional blink/trackloss column.
hz	Sampling rate passed to gazeR functions.
fillback	Blink-extension window before missing/blink samples, in ms.
fillforward	Blink-extension window after missing/blink samples, in ms.
smooth_n	Smoothing window parameter passed to <code>gazer::smooth_interpolate_pupil()</code> .
step_first	Processing order passed to <code>gazer::smooth_interpolate_pupil()</code> .
interpolation_type	Interpolation type passed to <code>gazer::smooth_interpolate_pupil()</code> .
maxgap	Maximum gap passed to <code>gazer::smooth_interpolate_pupil()</code> .
baseline_window	Optional numeric vector of length 2 passed to <code>gazer::baseline_correction_pupil()</code> .
baseline_event	Optional event label passed to <code>gazer::baseline_correction_pupil_msg()</code> .
baseline_dur	Baseline duration used with <code>baseline_event</code> .
baseline_method	Baseline method used with <code>baseline_event</code> .
bin_length	Optional bin length passed to <code>gazer::downsample_gaze()</code> .
name	Character label stored in the returned object.

Details

This helper is a cross-check branch. It is not intended to replace the main `gp3tools` pupil preprocessing pipeline.

Value

A list with class `gp3_gazer_crosscheck`.

```
run_gazepoint_gpbiometrics_workflow
```

Run a tested gp3tools-to-gpbiometrics integration workflow

Description

Aligns `gp3tools` gaze output with an already imported biometric data frame, computes AOI/event-contingent signal summaries, and returns a combined audit and cautious report text. A user-supplied adapter can replace the native nearest-timestamp join without making `gpbiometrics` a mandatory dependency.

Usage

```
run_gazepoint_gpbiometrics_workflow(
  gaze_data,
  biometrics_data,
  gaze_args = list(),
  biometric_participant_col = NULL,
  biometric_trial_col = NULL,
  biometric_time_col = NULL,
  biometric_time_unit = c("auto", "seconds", "milliseconds"),
  signal_cols = NULL,
  event_col = NULL,
  tolerance_s = NULL,
  adapter = NULL,
  include_unmatched = TRUE
)
```

Arguments

<code>gaze_data</code>	Raw gp3tools gaze/master data or a prepared bridge.
<code>biometrics_data</code>	Biometric samples, including participant, trial, time, and one or more numeric signal columns.
<code>gaze_args</code>	Named list forwarded to prepare_gazepoint_gpbiometrics_bridge() .
<code>biometric_participant_col, biometric_trial_col, biometric_time_col</code>	Explicit biometric source columns.
<code>biometric_time_unit</code>	Unit for the biometric time column.
<code>signal_cols</code>	Numeric biometric signals to summarise.
<code>event_col</code>	Optional biometric event column.
<code>tolerance_s</code>	Maximum absolute time difference for a match. When NULL, it is estimated conservatively from biometric sampling intervals.
<code>adapter</code>	Optional function with arguments <code>gaze</code> , <code>biometrics</code> , and <code>tolerance_s</code> ; it must return a synchronized data frame.
<code>include_unmatched</code>	Retain unmatched gaze rows.

Value

A "gazepoint_cross_package_workflow" object.

```
run_gazepoint_model_leave_one_out
```

Run leave-one-unit model sensitivity analysis

Description

Refit the same model repeatedly while removing one participant, item, stimulus, trial, or other analysis unit at a time. The helper compares leave-one-out estimates with the full-data model to assess whether a key effect is driven by a single unit.

Usage

```
run_gazepoint_model_leave_one_out(
  data,
  unit_col,
  fit_function,
  extract_function = NULL,
  effect_terms = NULL,
  min_rows = 2L,
  keep_models = FALSE,
  name = "gazepoint_model_leave_one_out"
)
```

Arguments

<code>data</code>	A data frame used for model fitting.
<code>unit_col</code>	Column identifying the unit to leave out, for example subject, participant, item, stimulus, or trial.
<code>fit_function</code>	Function that takes one data frame argument and returns a fitted model.
<code>extract_function</code>	Optional function that takes a fitted model and returns a data frame of effects. If NULL, a default coefficient extractor is used for common model objects.
<code>effect_terms</code>	Optional character vector of terms/effects to retain in the sensitivity summary.
<code>min_rows</code>	Minimum number of rows required after leaving one unit out.
<code>keep_models</code>	Logical. If TRUE, keep the full model and refitted models.
<code>name</code>	Character label stored in the returned object.

Details

This is a generic robustness wrapper. It can be used with linear models, GLMs, mixed models, GAMMs, GCA models, AOI GLMMs, pupil LMMs, or any custom model as long as a fitting function is supplied.

Value

A list with class `gp3_model_leave_one_out_sensitivity`.

`run_gazepoint_multidimensional_cluster_permutation`*Guardrail for multidimensional cluster permutation*

Description

This function intentionally fails safely. Multidimensional cluster permutation is not implemented as an active inferential engine in gp3tools.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

`run_gazepoint_pupil_multiverse`*Run a Gazepoint pupil preprocessing multiverse*

Description

Run all pupil preprocessing branches defined by `create_gazepoint_preprocessing_multiverse()`. Each branch can apply pupil artifact flagging, interpolation, baseline correction, smoothing, and optional pupil-window summarisation.

Usage

```
run_gazepoint_pupil_multiverse(  
  data,  
  multiverse,  
  branch_ids = NULL,  
  pupil_col = NULL,  
  time_col = NULL,  
  group_cols = NULL,  
  summarise_windows = FALSE,  
  windows = NULL,  
  keep_outputs = TRUE,  
  stop_on_error = FALSE  
)
```

Arguments

data	A Gazepoint master table or processed pupil table.
multiverse	A gp3_preprocessing_multiverse object returned by create_gazepoint_preprocessing_multiverse.
branch_ids	Optional character vector of pupil branch IDs to run.
pupil_col	Optional pupil column passed to downstream preprocessing helpers when supported.
time_col	Optional time column passed to downstream preprocessing helpers when supported.
group_cols	Optional grouping columns passed to downstream preprocessing helpers when supported.
summarise_windows	Logical. If TRUE, summarise each processed pupil branch into pupil analysis windows.
windows	Optional windows passed to summarise_gazepoint_pupil_windows() when summarise_windows = TRUE.
keep_outputs	Logical. If TRUE, keep processed branch data in branch_outputs.
stop_on_error	Logical. If TRUE, stop when a branch fails. If FALSE, record the branch error and continue.

Value

A list with class gp3_pupil_multiverse_results containing overview, branch results, optional branch outputs, and settings.

run_gazepoint_tfce	<i>Guardrail for threshold-free cluster enhancement</i>
--------------------	---

Description

This function intentionally fails safely. TFCE is not implemented as an active inferential engine in gp3tools.

Usage

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

Arguments

... Arguments reserved for a future implementation.

Value

This function always stops with an explanatory error.

`run_gazepoint_workflow`*Run a complete Gazepoint analysis workflow*

Description

Reads Gazepoint all-gaze and fixation exports from a folder, computes sampling-rate checks, tracking-quality summaries, quality flags, and AOI-level metrics, and optionally exports output tables, diagnostic plots, and an HTML diagnostic report.

Usage

```
run_gazepoint_workflow(  
  export_dir,  
  all_gaze_pattern = "_all_gaze\\.csv$",  
  fixation_pattern = "_fixations\\.csv$",  
  check_file_pairs = TRUE,  
  group_cols = c("USER_FILE", "MEDIA_ID"),  
  user_col = "USER_FILE",  
  sample_rate = 60,  
  min_gaze_valid_pct = 70,  
  min_pupil_valid_pct = 70,  
  expected_hz = 60,  
  hz_tolerance = 5,  
  min_duration_sec = NULL,  
  output_dir = NULL,  
  prefix = "gazepoint",  
  overwrite = TRUE,  
  save_plots = FALSE,  
  plot_output_dir = NULL,  
  create_report = FALSE,  
  report_file = NULL,  
  report_title = "Gazepoint diagnostic report",  
  report_plot_dir = NULL,  
  report_max_rows = 30  
)
```

Arguments

<code>export_dir</code>	Folder containing Gazepoint CSV export files.
<code>all_gaze_pattern</code>	Regular expression for selecting all-gaze files.
<code>fixation_pattern</code>	Regular expression for selecting fixation files.
<code>check_file_pairs</code>	Logical. If TRUE, check that each participant/source has both an all-gaze file and a fixation file before importing.

group_cols	Columns used for grouped sampling and tracking-quality summaries.
user_col	Column name used to identify the source/user file.
sample_rate	Sampling rate used for approximate sample-based AOI viewed time.
min_gaze_valid_pct	Minimum acceptable FPOGV validity percentage.
min_pupil_valid_pct	Minimum acceptable pupil validity percentage.
expected_hz	Expected sampling rate.
hz_tolerance	Allowed deviation from the expected sampling rate.
min_duration_sec	Optional minimum acceptable recording duration in seconds.
output_dir	Optional folder where output CSV files should be written.
prefix	Filename prefix used when exporting output tables, plots, and report.
overwrite	Logical. If FALSE, stop when output files already exist.
save_plots	Logical. If TRUE, save standard diagnostic plots.
plot_output_dir	Optional folder where diagnostic plots should be saved. If NULL, output_dir is used.
create_report	Logical. If TRUE, create an HTML diagnostic report.
report_file	Optional path to the HTML report. If NULL, a report file is created in output_dir using prefix.
report_title	Title used in the HTML diagnostic report.
report_plot_dir	Optional folder for plots used inside the HTML report.
report_max_rows	Maximum number of rows shown in report preview tables.

Value

A named list containing file-pair checks, imported data, analysis tables, quality flags, written table paths, written plot paths, and written report path.

save_gazepoint_plots	<i>Save standard Gazepoint diagnostic plots</i>
----------------------	---

Description

Saves standard diagnostic plots produced from gp3tools workflow outputs.

Usage

```
save_gazepoint_plots(  
  flagged_quality = NULL,  
  sampling = NULL,  
  output_dir,  
  prefix = "gazepoint",  
  overwrite = TRUE,  
  width = 9,  
  height_quality = 6,  
  height_sampling = 5,  
  dpi = 300  
)
```

Arguments

flagged_quality	Flagged tracking-quality table, usually from flag_tracking_quality() or run_gazepoint_workflow()
sampling	Sampling-rate table, usually from check_sampling_rate() or run_gazepoint_workflow().
output_dir	Folder where plot files should be saved.
prefix	Filename prefix used for saved plot files.
overwrite	Logical. If FALSE, stop when output plot files already exist.
width	Plot width in inches.
height_quality	Tracking-quality plot height in inches.
height_sampling	Sampling-rate plot height in inches.
dpi	Plot resolution.

Value

A tibble with plot names and written file paths.

segment_gazepoint_task_phases
<i>Segment Gazepoint-style data into task phases</i>

Description

Assigns each row of a Gazepoint-style data set to a task phase using a user-supplied phase-window table. The helper is deterministic and descriptive: it labels rows according to declared time windows and does not infer phases.

Usage

```
segment_gazepoint_task_phases(
  data,
  time_col,
  phase_windows,
  phase_col = "task_phase",
  window_phase_col = "phase",
  window_start_col = "start",
  window_end_col = "end",
  outside_label = "outside",
  include_lower = TRUE,
  include_upper = FALSE,
  keep_window_metadata = FALSE
)
```

Arguments

<code>data</code>	A data frame.
<code>time_col</code>	Character name of the time column.
<code>phase_windows</code>	A data frame containing phase labels and start/end times.
<code>phase_col</code>	Output column name for assigned phases.
<code>window_phase_col, window_start_col, window_end_col</code>	Column names in <code>phase_windows</code> .
<code>outside_label</code>	Label assigned to rows outside all phase windows. If <code>NULL</code> , outside rows receive <code>NA_character_</code> .
<code>include_lower, include_upper</code>	Logical values controlling whether phase window boundaries are closed on the lower and upper sides.
<code>keep_window_metadata</code>	If <code>TRUE</code> , adds assigned phase-window start and end columns.

Value

A copy of data with phase labels and assignment diagnostics.

Examples

```
x <- data.frame(time_ms = c(0, 250, 750, 1250, 1750))
windows <- data.frame(
  phase = c("baseline", "stimulus", "response"),
  start = c(0, 500, 1000),
  end = c(500, 1000, 2000)
)
segment_gazepoint_task_phases(x, "time_ms", windows)
```

`select_gazepoint_adaptive_trial`*Select the next adaptive trial from candidate stimuli*

Description

Provides a lightweight Bayesian-optimization-style acquisition helper for adaptive testing. It assumes candidate-level posterior means and standard deviations are already available or supplied by the user.

Usage

```
select_gazepoint_adaptive_trial(  
  candidates,  
  mean,  
  sd,  
  acquisition = c("ucb", "uncertainty", "expected_improvement"),  
  kappa = 2,  
  best_observed = NULL,  
  maximize = TRUE  
)
```

Arguments

<code>candidates</code>	A data frame of candidate stimuli/trials.
<code>mean</code>	Column containing posterior mean utility or expected information.
<code>sd</code>	Column containing posterior uncertainty.
<code>acquisition</code>	Acquisition rule: "ucb", "uncertainty", or "expected_improvement".
<code>kappa</code>	Exploration weight for UCB.
<code>best_observed</code>	Best observed value for expected improvement.
<code>maximize</code>	Logical; select maximum acquisition value if TRUE.

Value

One-row data frame corresponding to the selected candidate, with an added acquisition score.

```
select_gazepoint_scanpath_clusters
```

Select the number of scanpath clusters

Description

Compare candidate cluster counts using mean silhouette width. Input formats and sequence-preparation arguments match [cluster_gazepoint_scanpaths\(\)](#).

Usage

```
select_gazepoint_scanpath_clusters(
  x,
  k_values = NULL,
  method = c("hierarchical", "pam"),
  linkage = c("average", "complete", "single", "ward.D2", "ward.D", "mcquitty", "median",
    "centroid"),
  aoi_col = NULL,
  group_cols = NULL,
  time_col = NULL,
  distance_col = "normalized_distance",
  include_missing = FALSE,
  missing_label = "missing",
  collapse_repeats = FALSE,
  max_sequences = 200
)
```

Arguments

<code>x</code>	Long-format AOI data, a pairwise-distance data frame, a square numeric distance matrix, or a dist object.
<code>k_values</code>	Candidate cluster counts. When NULL, values from 2 through the smaller of 6 or one fewer than the number of scanpaths are evaluated.
<code>method</code>	Clustering method: "hierarchical" or "pam".
<code>linkage</code>	Hierarchical linkage method passed to stats::hclust() .
<code>aoi_col</code>	AOI column when <code>x</code> is long-format AOI data.
<code>group_cols</code>	Columns identifying independent scanpaths.
<code>time_col</code>	Optional ordering column.
<code>distance_col</code>	Distance column for pairwise-distance data.
<code>include_missing</code>	Should missing AOI labels be retained?
<code>missing_label</code>	Label used for retained missing AOIs.
<code>collapse_repeats</code>	Should consecutive repeated AOIs be collapsed?
<code>max_sequences</code>	Maximum number of scanpaths permitted when pairwise distances must be calculated.

Value

An object of class `gp3_scanpath_cluster_selection` containing candidate diagnostics, the recommended number of clusters, all fitted solutions, the recommended fit, and the distance object.

Examples

```
d <- matrix(
  c(
    0, 0.1, 1, 1,
    0.1, 0, 1, 1,
    1, 1, 0, 0.1,
    1, 1, 0.1, 0
  ),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(LETTERS[1:4], LETTERS[1:4])
)

if (requireNamespace("cluster", quietly = TRUE)) {
  result <- select_gazepoint_scanpath_clusters(
    d,
    k_values = 2:3
  )
  result$diagnostics
}
```

`simulate_gazepoint_cluster_timecourse_data`

Simulate simple Gazepoint cluster time-course data

Description

Generate synthetic two-condition within-subject time-course data for examples and tests. The simulation is intentionally simple and should not be treated as a realistic model of gaze, pupil, or biometric time-series data.

Usage

```
simulate_gazepoint_cluster_timecourse_data(
  n_subjects = 20,
  n_time_bins = 60,
  conditions = c("control", "treatment"),
  effect_start = 25,
  effect_end = 40,
  effect_size = 0.5,
  subject_sd = 0.3,
  noise_sd = 0.4,
  seed = NULL
)
```

Arguments

n_subjects	Number of subjects.
n_time_bins	Number of time bins.
conditions	Two condition labels.
effect_start	First time bin with an injected treatment effect.
effect_end	Last time bin with an injected treatment effect.
effect_size	Added treatment effect inside the effect window.
subject_sd	Standard deviation of subject random shifts.
noise_sd	Standard deviation of observation noise.
seed	Optional random seed.

Value

A long-format data frame.

simulate_gazepoint_data

Simulate simple Gazepoint-style gaze data

Description

Generate a compact synthetic Gazepoint-style data set for examples, tests, teaching, and workflow demonstrations. The simulation is intentionally simple and should not be treated as a realistic generative model of visual attention.

Usage

```
simulate_gazepoint_data(  
  n_subjects = 12,  
  n_trials = 8,  
  trial_duration_ms = 2000,  
  sampling_rate_hz = 60,  
  conditions = c("control", "treatment"),  
  aoi_labels = c("target", "other"),  
  effect_size = 0.5,  
  target_aoi = aoi_labels[1L],  
  seed = NULL,  
  include_fixations = TRUE  
)
```

Arguments

n_subjects	Number of synthetic participants.
n_trials	Number of trials per participant.
trial_duration_ms	Trial duration in milliseconds.
sampling_rate_hz	Sampling rate in Hz.
conditions	Character vector of condition labels.
aoi_labels	Character vector of AOI labels.
effect_size	Logit-scale increase in target-AOI probability for the second and later conditions.
target_aoi	AOI receiving the simulated condition effect.
seed	Optional random seed.
include_fixations	Should a simple fixation-level table be returned?

Value

A list containing all_gaze, aoi_windows, optional fixations, and simulation metadata.

```
simulate_gazepoint_fixations
```

Simulate Gazepoint-like fixation events

Description

Generates event-level fixation data with bounded random spatial drift.

Usage

```
simulate_gazepoint_fixations(
  n_subjects = 10,
  n_fix = 50,
  sd = 10,
  coordinate_system = c("pixels", "normalized"),
  screen_width = 1920,
  screen_height = 1080,
  duration_mean = 250,
  duration_sd = 80,
  saccade_gap_mean = 40,
  seed = NULL
)
```

Arguments

n_subjects	Number of simulated participants.
n_fix	Number of fixations per participant.
sd	Standard deviation of spatial random-walk increments.
coordinate_system	Coordinate representation.
screen_width, screen_height	Pixel dimensions when coordinate_system = "pixels".
duration_mean, duration_sd	Mean and standard deviation of fixation duration in milliseconds.
saccade_gap_mean	Mean interval between fixations in milliseconds.
seed	Optional random seed.

Value

A tibble resembling a Gazepoint fixation export.

Examples

```
simulate_gazepoint_fixations(
  n_subjects = 2,
  n_fix = 10,
  seed = 1
)
```

```
simulate_gazepoint_pupil_data
```

Simulate Gazepoint-like pupil data

Description

Generates a privacy-safe synthetic pupil data set with balanced conditions, left/right pupil channels, a combined pupil column, blink/trackloss flags, and simple gaze coordinates. The generator is intended for examples and unit tests, not for claims about empirical pupil physiology.

Usage

```
simulate_gazepoint_pupil_data(
  n_subjects = 12,
  n_trials = 8,
  n_time_bins = 60,
  conditions = c("control", "treatment"),
  baseline_mean = 3.5,
  condition_effect = 0.15,
  noise_sd = 0.08,
```

```

    subject_sd = 0.25,
    blink_probability = 0.03,
    seed = NULL
  )

```

Arguments

n_subjects	Number of synthetic participants.
n_trials	Number of trials per participant.
n_time_bins	Number of time bins per trial.
conditions	Character vector of condition labels.
baseline_mean	Mean pupil size around which synthetic data are generated.
condition_effect	Numeric effect added to non-reference conditions. If a single value is supplied, it is applied to all non-reference conditions. If multiple values are supplied, they are recycled across conditions.
noise_sd	Standard deviation of sample-level noise.
subject_sd	Standard deviation of participant-level random offsets.
blink_probability	Probability that a sample is marked as blink/trackloss.
seed	Optional random seed.

Value

A data frame with synthetic Gazepoint-like pupil and gaze columns.

Examples

```
simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)
```

smooth_gazepoint_coordinate

Smooth gaze coordinates within independent sequences

Description

Applies a centred rolling median or moving average to gaze coordinates.

Usage

```
smooth_gazepoint_coordinate(
  all_gaze,
  method = c("median", "mean"),
  window = 5,
  x_col = "FPOGX",
  y_col = "FPOGY",
  id_col = "USER_ID",
  group_cols = NULL,
  suffix = "_smooth",
  min_valid = 1,
  preserve_missing = TRUE
)
```

Arguments

<code>all_gaze</code>	A sample-level gaze data frame.
<code>method</code>	Smoothing method.
<code>window</code>	Positive integer rolling-window width.
<code>x_col, y_col</code>	Coordinate columns.
<code>id_col</code>	Participant identifier.
<code>group_cols</code>	Additional independent-sequence columns.
<code>suffix</code>	Suffix for generated columns.
<code>min_valid</code>	Minimum finite samples required in a rolling window.
<code>preserve_missing</code>	Keep smoothed values missing where the original coordinate is missing.

Value

The input data with smoothed coordinate columns.

Examples

```
gaze <- data.frame(
  USER_ID = "P01",
  FPOGX = c(0.1, 0.11, 0.5, 0.12, 0.13),
  FPOGY = c(0.2, 0.21, 0.6, 0.22, 0.23)
)
smooth_gazepoint_coordinate(gaze, window = 3)
```

smooth_gazepoint_pupil

Smooth Gazepoint pupil data

Description

Applies sample-based rolling smoothing to a Gazepoint pupil time series, typically after `flag_gazepoint_pupil()`, `interpolate_gazepoint_pupil()`, and optionally `baseline_correct_gazepoint_pupil()`. The function preserves the original pupil column and adds smoothed-output columns.

Usage

```
smooth_gazepoint_pupil(
  data,
  pupil_col = NULL,
  time_col = NULL,
  group_cols = c("subject", "media_id"),
  window_samples = 5,
  method = c("mean", "median"),
  align = c("center", "right", "left"),
  min_points = 1,
  preserve_missing = TRUE
)
```

Arguments

<code>data</code>	A Gazepoint master table, preferably after pupil preprocessing.
<code>pupil_col</code>	Optional name of the pupil column to smooth. If NULL, the function detects one of <code>pupil_baseline_corrected</code> , <code>pupil_baseline_percent_change</code> , <code>pupil_interpolated</code> , <code>pupil_for_preprocessing</code> , <code>mean_pupil</code> , <code>pupil</code> , <code>pupil_raw</code> , <code>left_pupil</code> , or <code>right_pupil</code> .
<code>time_col</code>	Optional name of the time column. If NULL, the function detects one of <code>time_ms</code> , <code>time</code> , <code>time_orig</code> , or <code>time_orig_ms</code> .
<code>group_cols</code>	Character vector of grouping columns used to keep smoothing within independent time series. Defaults to <code>c("subject", "media_id")</code> . Columns such as <code>"trial"</code> or <code>"trial_global"</code> can be added when available. Use <code>character(0)</code> for global smoothing.
<code>window_samples</code>	Number of samples in the rolling smoothing window. Defaults to 5.
<code>method</code>	Smoothing statistic. One of <code>"mean"</code> or <code>"median"</code> . Defaults to <code>"mean"</code> .
<code>align</code>	Window alignment. One of <code>"center"</code> , <code>"right"</code> , or <code>"left"</code> . Defaults to <code>"center"</code> .
<code>min_points</code>	Minimum number of finite values required inside a window to return a smoothed value. Defaults to 1.
<code>preserve_missing</code>	Logical. If TRUE, rows with missing/non-finite input remain missing in <code>pupil_smoothed</code> . Defaults to TRUE.

Value

A tibble containing the original data plus pupil-smoothing columns.

Examples

```
## Not run:
smoothed <- smooth_gazepoint_pupil(
  baseline_corrected,
  pupil_col = "pupil_baseline_corrected",
  window_samples = 5,
  method = "mean"
)

dplyr::count(smoothed, pupil_smoothing_status)

## End(Not run)
```

standardise_gazepoint_names

Standardise Gazepoint column names

Description

Converts timestamped Gazepoint headers such as TIME(2026/02/20 00:53:57.275) to TIME, converts TIMETICK(f=10000000) to TIMETICK, trims whitespace, and removes empty columns created by trailing commas in Gazepoint exports.

Usage

```
standardise_gazepoint_names(x)
```

Arguments

x A data frame or character vector of column names.

Value

If x is a data frame, the same data frame with standardised names and empty Gazepoint columns removed. If x is a character vector, a character vector.

standardize_gazepoint_face_columns

Standardise external facial-analysis columns

Description

Adds common gp3tools-friendly facial-analysis columns to external face-analysis outputs. The function keeps the original columns by default and prepends standard columns such as face_frame, face_time_sec, face_confidence, face_success, and face_valid.

Usage

```
standardize_gazepoint_face_columns(
  data,
  source = c("auto", "openface", "pyfeat", "mediapipe", "facereader", "generic"),
  participant_id_col = NULL,
  frame_col = NULL,
  time_col = NULL,
  confidence_col = NULL,
  success_col = NULL,
  face_id_col = NULL,
  file_col = NULL,
  confidence_threshold = 0.8,
  keep_original_columns = TRUE
)
```

Arguments

data	A data frame returned by read_gazepoint_face_export(), a plain data frame, or a CSV path readable by read_gazepoint_face_export().
source	Facial-analysis source. Use "auto" to infer a likely source.
participant_id_col	Optional participant identifier column.
frame_col	Optional frame-index column.
time_col	Optional time column, preferably in seconds.
confidence_col	Optional face-detection confidence column.
success_col	Optional face-detection success column.
face_id_col	Optional face identifier column.
file_col	Optional file/source column.
confidence_threshold	Minimum confidence required for face_valid when a confidence column is available.
keep_original_columns	Should original facial-analysis columns be kept?

Value

A tibble with standardised facial-analysis columns. The returned object has class `gp3_face_data`.

Examples

```
face <- data.frame(
  frame = 1:2,
  timestamp = c(0, 0.033),
  confidence = c(0.98, 0.40),
  success = c(1, 1),
  AU12_r = c(0.1, 0.2)
)

standardize_gazepoint_face_columns(face)
```

```
stress_test_gazepoint_binocular_reconstruction
```

Stress-test binocular reconstruction across missingness levels

Description

Repeats artificial monocular-loss validation across declared missingness levels and random/contiguous masking modes to show where cross-eye reconstruction begins to degrade.

Usage

```
stress_test_gazepoint_binocular_reconstruction(
  data,
  left_col,
  right_col,
  time_col = NULL,
  group_cols = NULL,
  gap_group_cols = NULL,
  fallback_group_cols = NULL,
  missingness = c(0.05, 0.1, 0.2, 0.3),
  mask_modes = c("random", "contiguous"),
  block_size = 6L,
  repeats = 3L,
  seed = 1L,
  min_pairs = 30L,
  min_unique = 5L,
  min_r2 = NULL,
  max_gap_ms = Inf,
  valid_min = NULL,
  valid_max = NULL
)
```

Arguments

data, left_col, right_col, time_col, group_cols, gap_group_cols, fallback_group_cols
 See [validate_gazepoint_binocular_reconstruction\(\)](#).
 missingness Numeric proportions strictly between 0 and 1.
 mask_modes One or both of "random" and "contiguous".
 block_size Contiguous mask size in samples.
 repeats Repeats per stress-test cell.
 seed Base seed; deterministic offsets are used across cells.
 min_pairs, min_unique, min_r2, max_gap_ms, valid_min, valid_max
 Passed to validation.

Value

A gp3_binocular_stress_test object containing cell-level results and individual validation objects.

See Also

[validate_gazepoint_binocular_reconstruction\(\)](#)

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 22)
stress_test_gazepoint_binocular_reconstruction(
  dat, "pupil_left", "pupil_right", time_col = "timestamp_ms",
  group_cols = "subject", missingness = c(0.05, 0.10),
  mask_modes = "random", repeats = 1, min_pairs = 20
)
```

summarise_aoi_samples *Summarise sample-level AOI viewing*

Description

Computes transparent AOI metrics from sample-level rows. These may not exactly reproduce Gaze-point Analysis summary metrics; use [read_gazepoint_summary\(\)](#) when official Gaze-point summary values are available.

Usage

```
summarise_aoi_samples(
  data,
  group_cols = "MEDIA_ID",
  aoi_col = "AOI",
  time_col = "TIME"
)
```

Arguments

data	A Gazepoint all-gaze data frame.
group_cols	Grouping columns.
aoi_col	AOI column name.
time_col	Time column name.

Value

A tibble with AOI sample count, TTFF, and approximate dwell time.

summarise_fixations	<i>Summarise fixation-level AOI metrics</i>
---------------------	---

Description

Summarise fixation-level AOI metrics

Usage

```
summarise_fixations(data, group_cols = "MEDIA_ID", aoi_col = "AOI")
```

Arguments

data	A Gazepoint fixation data frame.
group_cols	Grouping columns.
aoi_col	AOI column name.

Value

A tibble with fixation counts and summed fixation duration by AOI.

summarise_gazepoint_aoi	<i>Summarise Gazepoint AOI metrics from gaze and fixation exports</i>
-------------------------	---

Description

Combines sample-level AOI viewing information from all-gaze data with fixation-level AOI metrics from fixation data.

Usage

```
summarise_gazepoint_aoi(
  gaze_data,
  fixation_data,
  user_col = "USER_FILE",
  sample_rate = 60
)
```

Arguments

`gaze_data` A Gazepoint all-gaze data frame imported with `read_gazepoint()`.

`fixation_data` A Gazepoint fixation data frame imported with `read_gazepoint()`.

`user_col` Name of the column identifying the user file. Default is "USER_FILE".

`sample_rate` Assumed sampling rate used to approximate viewed time from sample counts.

Value

A tibble with one row per user, media, and AOI.

```
summarise_gazepoint_aoi_entries
```

Summarise Gazepoint AOI entry episodes

Description

Convert sample-level AOI states into AOI entry episodes. An entry starts whenever the AOI state changes within a subject, media, trial, or other grouping unit.

Usage

```
summarise_gazepoint_aoi_entries(
  data,
  aoi_col = NULL,
  time_col = "time",
  group_cols = c("subject", "MEDIA_ID", "trial_global"),
  include_non_aoi = TRUE,
  non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
    "missing", "missing_aoi"),
  missing_aoi_label = "missing_aoi"
)
```

Arguments

<code>data</code>	A Gazepoint master/sample-level data frame.
<code>aoi_col</code>	Name of the AOI-state column. If NULL, the function tries <code>aoi_current</code> , <code>AOI</code> , and <code>aoi_state</code> .
<code>time_col</code>	Name of the time column, in milliseconds.
<code>group_cols</code>	Character vector of columns defining independent sequences, usually subject/media/trial.
<code>include_non_aoi</code>	Logical. If TRUE, non-AOI/background episodes are retained. If FALSE, they are removed after entry order and neighbouring states have been computed.
<code>non_aoi_values</code>	Character vector of AOI labels treated as background or non-AOI states.
<code>missing_aoi_label</code>	Label used when the AOI value is missing.

Value

A tibble with one row per AOI entry episode.

<code>summarise_gazepoint_aoi_transitions</code>	<i>Summarise Gazepoint AOI transition features</i>
--	--

Description

Summarise AOI transitions at the trial or group level. The function can work from sample-level Gazepoint AOI data, AOI-entry tables created by `summarise_gazepoint_aoi_entries()`, or AOI-sequence tables created by `prepare_gazepoint_aoi_sequences()`.

Usage

```
summarise_gazepoint_aoi_transitions(
  data,
  aoi_col = NULL,
  time_col = "time",
  group_cols = c("subject", "MEDIA_ID", "trial_global"),
  include_non_aoi = TRUE,
  target_aoi_values = NULL,
  distractor_aoi_values = NULL,
  non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
    "missing", "missing_aoi"),
  missing_aoi_label = "missing_aoi"
)
```

Arguments

<code>data</code>	A Gazepoint sample-level data frame, AOI-entry table, or AOI-sequence table.
<code>aoi_col</code>	Name of the AOI-state column. Used only when data is sample-level data. If NULL, the function tries <code>aoi_current</code> , <code>AOI</code> , and <code>aoi_state</code> .
<code>time_col</code>	Name of the time column, in milliseconds. Used only when data is sample-level data.
<code>group_cols</code>	Character vector of columns defining independent AOI sequences, usually subject/media/trial.
<code>include_non_aoi</code>	Logical. If TRUE, non-AOI/background states are retained before transition summaries are computed. This is useful for background-to-target and target-to-background features.
<code>target_aoi_values</code>	Optional character vector defining target AOI labels.
<code>distractor_aoi_values</code>	Optional character vector defining distractor AOI labels.
<code>non_aoi_values</code>	Character vector of AOI labels treated as background or non-AOI states.
<code>missing_aoi_label</code>	Label used when the AOI value is missing.

Value

A tibble with one row per group and trial-level AOI transition features.

```
summarise_gazepoint_aoi_trial_features
```

Summarise Gazepoint AOI trial features

Description

Create trial-level AOI features from sample-level Gazepoint AOI data or from AOI-entry tables created by `summarise_gazepoint_aoi_entries()`. The output includes AOI dwell, entry, TTFF, revisit, and transition features.

Usage

```
summarise_gazepoint_aoi_trial_features(
  data,
  aoi_col = NULL,
  time_col = "time",
  group_cols = c("subject", "MEDIA_ID", "trial_global"),
  include_non_aoi = TRUE,
  target_aoi_values = NULL,
  distractor_aoi_values = NULL,
```

```

    non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
      "missing", "missing_aoi"),
    missing_aoi_label = "missing_aoi"
  )

```

Arguments

<code>data</code>	A Gazepoint sample-level data frame, AOI-entry table, or compatible AOI table.
<code>aoi_col</code>	Name of the AOI-state column. Used only when data is sample-level data. If NULL, the function tries <code>aoi_current</code> , <code>AOI</code> , and <code>aoi_state</code> .
<code>time_col</code>	Name of the time column, in milliseconds. Used only when data is sample-level data.
<code>group_cols</code>	Character vector of columns defining independent trials, usually subject/media/trial.
<code>include_non_aoi</code>	Logical. If TRUE, non-AOI/background states are kept when computing trial-duration and transition features.
<code>target_aoi_values</code>	Optional character vector defining target AOI labels.
<code>distractor_aoi_values</code>	Optional character vector defining distractor AOI labels.
<code>non_aoi_values</code>	Character vector of AOI labels treated as background or non-AOI states.
<code>missing_aoi_label</code>	Label used when the AOI value is missing.

Value

A tibble with one row per trial/group and AOI trial-level features.

```
summarise_gazepoint_aoi_windows
```

Summarise Gazepoint AOI samples within predefined time windows

Description

Summarise sample-level AOI states into predefined analysis windows. This is intended for confirmatory AOI window modelling, especially binomial target-looking models where target samples are modelled relative to a denominator such as all valid window samples.

Usage

```

summarise_gazepoint_aoi_windows(
  data,
  windows,
  time_col = "time",
  aoi_col = NULL,

```

```

  subject_col = "subject",
  condition_col = "condition",
  group_cols = NULL,
  target_aoi_values = NULL,
  distractor_aoi_values = NULL,
  non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
    "missing", "missing_aoi"),
  window_label_col = "window_label",
  window_start_col = "window_start_ms",
  window_end_col = "window_end_ms",
  include_right_endpoint = FALSE,
  missing_condition_label = "all_data",
  missing_aoi_label = "missing_aoi"
)

```

Arguments

<code>data</code>	A Gazepoint master table or sample-level gaze data.
<code>windows</code>	Numeric breakpoints, for example <code>c(0, 500, 1000, 2000)</code> , or a data frame with window labels and start/end columns.
<code>time_col</code>	Name of the time column.
<code>aoi_col</code>	Optional AOI-state column. If <code>NULL</code> , the function attempts to detect <code>aoi_current</code> , <code>AOI</code> , or <code>aoi_state</code> .
<code>subject_col</code>	Name of the subject column.
<code>condition_col</code>	Optional condition column.
<code>group_cols</code>	Additional grouping columns. Defaults to subject, condition, media, trial-global, and trial columns when available.
<code>target_aoi_values</code>	Character vector identifying target AOIs.
<code>distractor_aoi_values</code>	Character vector identifying distractor AOIs.
<code>non_aoi_values</code>	Character vector identifying background/non-AOI states.
<code>window_label_col</code>	Window label column when <code>windows</code> is a data frame.
<code>window_start_col</code>	Window start column when <code>windows</code> is a data frame.
<code>window_end_col</code>	Window end column when <code>windows</code> is a data frame.
<code>include_right_endpoint</code>	Logical. If <code>TRUE</code> , include the right endpoint of each window.
<code>missing_condition_label</code>	Label used when condition is missing.
<code>missing_aoi_label</code>	Label used when AOI state is missing.

Value

A tibble with one row per group and AOI window.

summarise_gazepoint_binocular_reporting

Create manuscript-ready binocular reconstruction reporting

Description

Produces structured reporting fields plus conservative prose describing the declared calibration, reconstruction burden, and optional pseudo-missing validation. The text never describes predicted values as measured or as the biological truth.

Usage

```
summarise_gazepoint_binocular_reporting(
  data,
  audit = NULL,
  validation = NULL,
  by = NULL,
  prefix = "gp3_binocular"
)
```

Arguments

data	Output from reconstruct_gazepoint_binocular_pupil() .
audit	Optional result from audit_gazepoint_binocular_reconstruction() .
validation	Optional result from validate_gazepoint_binocular_reconstruction() .
by	Optional audit grouping used when audit is not supplied.
prefix	Reconstruction prefix.

Value

A gp3_binocular_reporting object with summary, models, validation, text, and limitations.

See Also

[audit_gazepoint_binocular_reconstruction\(\)](#)

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 4, n_trials = 2, seed = 25)
dat$pupil_left[30:33] <- NA_real_
rec <- reconstruct_gazepoint_binocular_pupil(
  dat, "pupil_left", "pupil_right", group_cols = "subject", min_pairs = 20
)
summarise_gazepoint_binocular_reporting(rec)$text
```

summarise_gazepoint_clusters
<i>Summarise cluster-based permutation results</i>

Description

Create compact reporting tables from the output of `run_gazepoint_cluster_permutation()`. The function returns an overview table, all observed clusters, significant clusters, time-course summary, permutation-distribution summary, settings table, and circularity warning.

Usage

```
summarise_gazepoint_clusters(  
  result,  
  alpha = 0.05,  
  round_digits = NULL,  
  include_timecourse = TRUE  
)
```

Arguments

result	A result object returned by <code>run_gazepoint_cluster_permutation()</code> .
alpha	Cluster-level significance threshold.
round_digits	Optional number of digits for rounding numeric reporting columns. If NULL, no rounding is applied.
include_timecourse	Logical. If TRUE, include the full observed time-course table in the returned object.

Details

Cluster-based permutation tests are intended for time-course inference. They should not be used to discover a confirmatory time window and then test that same window again in a second confirmatory model.

Value

A list of summary tables.

summarise_gazepoint_coordinate_coverage
British-spelling alias for summarize_gazepoint_coordinate_coverage()

Description

British-spelling alias for `summarize_gazepoint_coordinate_coverage()`

Usage

```
summarise_gazepoint_coordinate_coverage(...)
```

Arguments

... Arguments passed to `summarize_gazepoint_coordinate_coverage()`.

Value

The result of `summarize_gazepoint_coordinate_coverage()`.

summarise_gazepoint_emmeans
Summarise estimated marginal means and contrasts

Description

Create manuscript-ready estimated marginal means and pairwise contrast tables from fitted models used in gp3tools workflows.

Usage

```
summarise_gazepoint_emmeans(  
  model,  
  specs,  
  by = NULL,  
  model_name = NULL,  
  type = "response",  
  contrast_method = "pairwise",  
  adjust = "tukey",  
  conf_level = 0.95,  
  include_contrasts = TRUE  
)
```

Arguments

<code>model</code>	A fitted model object, or a <code>gp3tools</code> fit object containing a <code>\$model</code> element.
<code>specs</code>	Character vector or formula passed to <code>emmeans::emmeans()</code> .
<code>by</code>	Optional character vector of grouping variables passed to <code>emmeans::emmeans()</code> .
<code>model_name</code>	Optional model label used in returned tables.
<code>type</code>	Scale passed to <code>emmeans</code> summaries. Common values are "link" and "response".
<code>contrast_method</code>	Contrast method passed to <code>emmeans::contrast()</code> .
<code>adjust</code>	Multiplicity adjustment for contrasts.
<code>conf_level</code>	Confidence level.
<code>include_contrasts</code>	Logical. If TRUE, compute contrasts.

Details

The function uses the optional `emmeans` package. If `emmeans` is not installed, the function returns structured skipped tables rather than failing. This keeps `emmeans` as an optional suggested dependency.

Value

A list with overview, `emmeans`, contrasts, and settings. The returned object has class `gp3_emmeans_summary`.

```
summarise_gazepoint_event_detector_agreement
```

Summarise agreement between event detectors

Description

Compare standardized fixation intervals using event-level intersection-over-union. Agreement is methodological and does not identify a uniquely correct detector.

Usage

```
summarise_gazepoint_event_detector_agreement(x, min_overlap = 0.5)
```

Arguments

<code>x</code>	An object returned by <code>compare_gazepoint_event_detectors()</code> or a standardized event data frame containing detector, sequence, start, end, and duration columns.
<code>min_overlap</code>	Minimum interval intersection-over-union required for a match.

Value

A list containing detector summaries, pairwise agreement, unmatched events, and settings.

`summarise_gazepoint_event_detector_benchmark`*Summarise an event-detector benchmark*

Description

Extract detector-level, sequence-level, matched-event, or unmatched-event tables from an event-detector benchmark.

Usage

```
summarise_gazepoint_event_detector_benchmark(  
  x,  
  level = c("detector", "sequence", "matches", "errors"),  
  sort = TRUE  
)
```

Arguments

<code>x</code>	An object returned by <code>benchmark_gazepoint_event_detectors()</code> .
<code>level</code>	Summary level to return.
<code>sort</code>	Should detector summaries be ordered by decreasing F1 score?

Value

A data frame.

`summarise_gazepoint_face_reactivity`*British-spelling alias for `summarize_gazepoint_face_reactivity()`*

Description

British-spelling alias for `summarize_gazepoint_face_reactivity()`

Usage

```
summarise_gazepoint_face_reactivity(...)
```

Arguments

<code>...</code>	Arguments passed to <code>summarize_gazepoint_face_reactivity()</code> .
------------------	--

Value

The result of `summarize_gazepoint_face_reactivity()`.

summarise_gazepoint_face_windows

British-spelling alias for summarize_gazepoint_face_windows()

Description

British-spelling alias for `summarize_gazepoint_face_windows()`

Usage

```
summarise_gazepoint_face_windows(...)
```

Arguments

... Arguments passed to `summarize_gazepoint_face_windows()`.

Value

The result of `summarize_gazepoint_face_windows()`.

summarise_gazepoint_fixation_trials

Summarise Gazeport fixation trial features

Description

Create trial-level fixation features from Gazeport fixation-level data. The function supports common Gazeport fixation export columns such as FPOGS, FPOGD, FPOGX, FPOGY, FPOGID, FPOGV, and AOI, as well as already-standardised columns.

Usage

```
summarise_gazepoint_fixation_trials(
  data,
  group_cols = NULL,
  fixation_id_col = NULL,
  start_col = NULL,
  duration_col = NULL,
  x_col = NULL,
  y_col = NULL,
  valid_col = NULL,
  aoi_col = NULL,
  start_time_unit = c("auto", "ms", "s"),
  duration_unit = c("auto", "ms", "s"),
  valid_only = TRUE,
```

```

include_non_aoi = TRUE,
target_aoi_values = NULL,
distractor_aoi_values = NULL,
non_aoi_values = c("non_aoi", "none", "background", "outside", "outside_aoi",
  "missing", "missing_aoi"),
missing_aoi_label = "missing_aoi"
)

```

Arguments

<code>data</code>	A Gazepoint fixation-level data frame.
<code>group_cols</code>	Character vector of columns defining independent trials. If NULL, the function tries to infer sensible grouping columns from participant, media, and trial columns.
<code>fixation_id_col</code>	Optional fixation ID column. If NULL, the function tries FPOGID, <code>fixation_id</code> , and related names.
<code>start_col</code>	Optional fixation start-time column. If NULL, the function tries FPOGS, <code>fixation_start_time</code> , <code>time</code> , <code>TIME</code> , and <code>TIMETICK</code> .
<code>duration_col</code>	Optional fixation-duration column. If NULL, the function tries FPOGD, <code>fixation_duration_ms</code> , <code>fixation_duration</code> , and related names.
<code>x_col</code>	Optional fixation x-coordinate column.
<code>y_col</code>	Optional fixation y-coordinate column.
<code>valid_col</code>	Optional fixation-validity column. If detected and <code>valid_only = TRUE</code> , invalid fixations are removed.
<code>aoi_col</code>	Optional AOI column. If NULL, the function tries AOI, <code>aoi_current</code> , and <code>aoi_state</code> .
<code>start_time_unit</code>	Unit for the start-time column: "auto", "ms", or "s".
<code>duration_unit</code>	Unit for the duration column: "auto", "ms", or "s".
<code>valid_only</code>	Logical. If TRUE, invalid fixations are removed when a validity column is available.
<code>include_non_aoi</code>	Logical. If TRUE, non-AOI/background fixations are included. If FALSE, they are removed before summaries are computed.
<code>target_aoi_values</code>	Optional character vector defining target AOI labels.
<code>distractor_aoi_values</code>	Optional character vector defining distractor AOI labels.
<code>non_aoi_values</code>	Character vector of AOI labels treated as background or non-AOI states.
<code>missing_aoi_label</code>	Label used when the AOI value is missing.

Value

A tibble with one row per trial/group and fixation-level features.

`summarise_gazepoint_fixed_effects`*Summarise fixed effects from fitted models*

Description

Create a compact manuscript-ready fixed-effect summary table from common models used in gp3tools workflows.

Usage

```
summarise_gazepoint_fixed_effects(  
  model,  
  model_name = NULL,  
  conf_level = 0.95,  
  exponentiate = FALSE,  
  drop_intercept = FALSE  
)
```

Arguments

<code>model</code>	A fitted model object, or a gp3tools fit object containing a <code>\$model</code> element.
<code>model_name</code>	Optional model label used in the returned table.
<code>conf_level</code>	Confidence level for Wald confidence intervals.
<code>exponentiate</code>	Logical. If TRUE, exponentiate estimates and confidence intervals. This is useful for logistic models when reporting odds ratios.
<code>drop_intercept</code>	Logical. If TRUE, remove the intercept row.

Details

The function supports `lm`, `glm`, `lme4` mixed models, and `mgcv` GAM/BAM objects. It can also accept a gp3tools fit object containing a `$model` element. Confidence intervals are computed using a Wald approximation from the estimate and standard error so that the function remains lightweight and fast for mixed models.

Value

A tibble with fixed-effect estimates, standard errors, test statistics, p-values when available, confidence intervals, significance stars, and status fields.

`summarise_gazepoint_markovchain`*Summarise a Gazepoint Markov-chain object*

Description

Convert a Gazepoint Markov-chain object, transition matrix, or transition data frame into a tidy transition summary. The function is deliberately permissive so that it can summarise objects created by `create_gazepoint_markovchain_object()` as well as simple matrices used in examples or tests.

Usage

```
summarise_gazepoint_markovchain(  
  markov_object,  
  include_zero = FALSE,  
  from_col = NULL,  
  to_col = NULL,  
  count_col = NULL,  
  probability_col = NULL  
)
```

Arguments

<code>markov_object</code>	A Markov-chain object, matrix, table, list, or data frame containing transition information.
<code>include_zero</code>	Should zero-valued transitions be retained?
<code>from_col</code>	Optional source-state column when <code>markov_object</code> is a data frame.
<code>to_col</code>	Optional destination-state column when <code>markov_object</code> is a data frame.
<code>count_col</code>	Optional transition-count column when available.
<code>probability_col</code>	Optional transition-probability column when available.

Value

A data frame with source state, destination state, transition count or weight, row total, transition probability, and status columns.

summarise_gazepoint_multiverse_results

Summarise Gazepoint preprocessing multiverse results

Description

Summarise pupil and/or AOI preprocessing multiverse result objects created by `run_gazepoint_pupil_multiverse()` and `run_gazepoint_aoi_multiverse()`.

Usage

```
summarise_gazepoint_multiverse_results(..., results = NULL)
```

Arguments

... One or more multiverse result objects.
results Optional named list of multiverse result objects.

Value

A list with class `gp3_multiverse_summary_results` containing overview, branch summary, failure summary, and settings tables.

summarise_gazepoint_pupil

Summarise Gazepoint pupil data

Description

Creates compact pupil-quality and pupil-distribution summaries from a Gazepoint master sample-level table created by [as_gazepoint_master\(\)](#) or [create_gazepoint_master\(\)](#). This function is intended as the first pupil preprocessing gate before interpolation, filtering, baseline correction, or pupil-based modelling.

Usage

```
summarise_gazepoint_pupil(
  master,
  group_cols = c("subject", "media_id"),
  pupil_col = NULL,
  time_col = NULL,
  missing_pupil_col = NULL,
  min_pupil = 0,
  max_pupil = Inf,
  outlier_k = 1.5
)
```

Arguments

<code>master</code>	A Gazepoint master sample-level table.
<code>group_cols</code>	Character vector of grouping columns. Defaults to <code>c("subject", "media_id")</code> using internally standardised names. Use <code>character(0)</code> for an overall summary.
<code>pupil_col</code>	Optional name of the pupil column to summarise. If <code>NULL</code> , the function detects one of <code>mean_pupil</code> , <code>pupil</code> , <code>pupil_raw</code> , <code>left_pupil</code> , or <code>right_pupil</code> .
<code>time_col</code>	Optional name of the time column. If <code>NULL</code> , the function detects one of <code>time_ms</code> , <code>time</code> , <code>time_orig</code> , or <code>time_orig_ms</code> .
<code>missing_pupil_col</code>	Optional name of the missing-pupil flag column. If <code>NULL</code> , the function uses <code>missing_pupil</code> when available.
<code>min_pupil</code>	Minimum plausible pupil value. Defaults to 0.
<code>max_pupil</code>	Maximum plausible pupil value. Defaults to <code>Inf</code> . Use narrower values, such as 1 and 9, only when the pupil column is known to be measured in millimetres.
<code>outlier_k</code>	Multiplier for IQR-based outlier detection. Defaults to 1.5.

Value

A tibble with pupil-quality and pupil-distribution summaries.

Examples

```
## Not run:
master <- create_gazepoint_master(
  gaze_data = results$all_gaze,
  screen_width_px = 1920,
  screen_height_px = 1080
)

summarise_gazepoint_pupil(master)
summarise_gazepoint_pupil(master, group_cols = "subject")
summarise_gazepoint_pupil(master, group_cols = character(0))

## End(Not run)
```

`summarise_gazepoint_pupil_response_features`

British-spelling alias for `summarize_gazepoint_pupil_response_features()`

Description

British-spelling alias for `summarize_gazepoint_pupil_response_features()`

Usage

```
summarise_gazepoint_pupil_response_features(...)
```

Arguments

... Arguments passed to `summarize_gazepoint_pupil_response_features()`.

Value

The result of `summarize_gazepoint_pupil_response_features()`.

```
summarise_gazepoint_pupil_trial_features
```

Summarise Gazepoint pupil trial-level features

Description

Convert sample-level Gazepoint pupil time series into trial-level pupil features for statistical modelling.

Usage

```
summarise_gazepoint_pupil_trial_features(
  data,
  group_cols = c("subject", "trial_global"),
  pupil_col = NULL,
  time_col = "time",
  interpolated_col = "pupil_was_interpolated",
  artifact_col = NULL,
  artifact_reason_col = NULL,
  early_window = c(0, 500),
  middle_window = c(500, 1500),
  late_window = c(1500, 3000),
  min_valid_samples = 1
)
```

Arguments

<code>data</code>	A Gazepoint pupil data frame.
<code>group_cols</code>	Character vector of grouping columns. The default is <code>c("subject", "trial_global")</code> .
<code>pupil_col</code>	Name of the processed pupil column to summarise. If <code>NULL</code> , the function tries <code>pupil_smoothed</code> , <code>pupil_baseline_corrected</code> , <code>pupil_baseline_percent_change</code> , <code>pupil_interpolated</code> , <code>pupil_clean</code> , and <code>pupil</code> .
<code>time_col</code>	Name of the time column.
<code>interpolated_col</code>	Optional logical interpolation flag column.

artifact_col	Optional artifact flag column. If NULL, the function tries to detect pupil_artifact_flag, pupil_flag_invalid, or artifact_flag.
artifact_reason_col	Optional artifact-reason column. If NULL, the function tries to detect pupil_artifact_reason, pupil_flag_reason, or artifact_reason.
early_window	Numeric vector of length 2 defining the early window in milliseconds.
middle_window	Numeric vector of length 2 defining the middle window in milliseconds.
late_window	Numeric vector of length 2 defining the late window in milliseconds.
min_valid_samples	Minimum number of valid pupil samples required for a trial to be labelled "ok".

Details

The function summarises one row per trial or other user-defined grouping. It computes mean pupil, peak pupil, time-to-peak, AUC, early/middle/late window means, valid-sample percentage, interpolation percentage, artifact percentage, and missingness summaries.

Value

A tibble with one row per trial/group.

summarise_gazepoint_pupil_windows

Summarise Gazepoint pupil responses within time windows

Description

Aggregates processed Gazepoint pupil data into user-defined analysis windows, typically after [flag_gazepoint_pupil\(\)](#), [interpolate_gazepoint_pupil\(\)](#), [baseline_correct_gazepoint_pupil\(\)](#), and [smooth_gazepoint_pupil\(\)](#). The function can summarise raw, interpolated, baseline-corrected, percent-change, or smoothed pupil columns.

Usage

```
summarise_gazepoint_pupil_windows(
  data,
  pupil_col = NULL,
  time_col = NULL,
  windows = c(0, 500, 1000, 2000),
  group_cols = c("subject", "media_id"),
  include_window_end = FALSE,
  min_valid_samples = 1
)
```

Arguments

<code>data</code>	A Gazepoint master table or processed pupil table.
<code>pupil_col</code>	Optional name of the pupil column to summarise. If NULL, the function detects one of <code>pupil_smoothed</code> , <code>pupil_baseline_corrected</code> , <code>pupil_baseline_percent_change</code> , <code>pupil_interpolated</code> , <code>pupil_for_preprocessing</code> , <code>mean_pupil</code> , <code>pupil</code> , <code>pupil_raw</code> , <code>left_pupil</code> , or <code>right_pupil</code> .
<code>time_col</code>	Optional name of the time column used for assigning samples to windows. If NULL, the function detects one of <code>time_relative_ms</code> , <code>relative_time_ms</code> , <code>event_time_ms</code> , <code>time_ms</code> , <code>time</code> , <code>time_orig</code> , or <code>time_orig_ms</code> .
<code>windows</code>	Window definitions. Either a numeric vector of breakpoints, such as <code>c(0, 500, 1000, 2000)</code> , or a data frame with window start and end columns. Supported names include <code>window_start_ms</code> , <code>window_start</code> , <code>start_ms</code> , or <code>start</code> , and <code>window_end_ms</code> , <code>window_end</code> , <code>end_ms</code> , or <code>end</code> . A <code>window_label</code> or <code>label</code> column is optional.
<code>group_cols</code>	Character vector of grouping columns. Standard roles such as <code>"subject"</code> , <code>"media_id"</code> , <code>"trial"</code> , and <code>"trial_global"</code> are internally standardised when available. Other columns, such as <code>"condition"</code> or <code>"AOI"</code> , can also be used if present in data. Use <code>character(0)</code> for overall window summaries.
<code>include_window_end</code>	Logical. If FALSE, windows are left-closed and right-open: <code>[start, end)</code> . If TRUE, the end point is included: <code>[start, end]</code> . Defaults to FALSE.
<code>min_valid_samples</code>	Minimum number of finite pupil samples required for a window to be labelled <code>"valid"</code> . Defaults to 1.

Value

A tibble with one row per group-by-window combination present in the data.

Examples

```
## Not run:
pupil_windows <- summarise_gazepoint_pupil_windows(
  smoothed_pupil,
  pupil_col = "pupil_smoothed",
  windows = c(0, 500, 1000, 2000),
  group_cols = c("subject", "media_id")
)

## End(Not run)
```

```
summarise_gazepoint_scanpath_cluster_stability
```

Summarise scanpath-cluster stability

Description

Convert bootstrap scanpath-clustering output into overview, sequence-level, pairwise, and representative-stability tables.

Usage

```
summarise_gazepoint_scanpath_cluster_stability(
  x,
  min_pair_coverage = 0.5,
  stable_threshold = 0.75
)
```

Arguments

x An object returned by `bootstrap_gazepoint_scanpath_clusters()`.

min_pair_coverage Minimum proportion of iterations in which a pair must co-occur before it contributes to summaries.

stable_threshold Within-reference-cluster co-clustering threshold used to count stable scanpaths.

Value

An object of class "gp3_scanpath_cluster_stability_summary" containing overview, sequence, pairwise, and representative tables.

Examples

```
latent <- rep(1:3, each = 2)
d <- outer(
  latent,
  latent,
  FUN = function(x, y) ifelse(x == y, 0.1, 1)
)
diag(d) <- 0
dimnames(d) <- list(LETTERS[1:6], LETTERS[1:6])

stability <- bootstrap_gazepoint_scanpath_clusters(
  d,
  k = 3,
  n_boot = 10,
  seed = 1
)
```

```
summarise_gazepoint_scanpath_cluster_stability(
  stability
)
```

summarise_gazepoint_semimarkov
<i>Summarise Gazepoint semi-Markov data</i>

Description

Summarise state visits and state-to-state transitions from semi-Markov preparation output or a compatible data frame. The function returns a list with a state-duration summary and a transition summary.

Usage

```
summarise_gazepoint_semimarkov(
  semimarkov_data,
  state_col = NULL,
  duration_col = NULL,
  sequence_col = NULL,
  time_col = NULL,
  from_col = NULL,
  to_col = NULL
)
```

Arguments

semimarkov_data	Output from prepare_gazepoint_semimarkov_data() or a compatible data frame/list.
state_col	Optional state/AOI column.
duration_col	Optional state-duration column.
sequence_col	Optional sequence, subject, or trial column.
time_col	Optional time/order column.
from_col	Optional transition source-state column.
to_col	Optional transition destination-state column.

Value

A list with state_summary, transition_summary, columns, and summary_status.

`summarise_gazepoint_time_clusters`*British-spelling alias for `summarize_gazepoint_time_clusters()`*

Description

British-spelling alias for `summarize_gazepoint_time_clusters()`

Usage

```
summarise_gazepoint_time_clusters(...)
```

Arguments

... Arguments passed to `summarize_gazepoint_time_clusters()`.

Value

The result of `summarize_gazepoint_time_clusters()`.

`summarise_gazepoint_workflow`*Summarise a Gazepoint workflow result*

Description

Creates a compact one-row summary from a result object returned by `run_gazepoint_workflow()`. This is useful for quickly checking how many rows, file pairs, flagged recordings, exported tables, exported plots, and reports were produced by the workflow.

Usage

```
summarise_gazepoint_workflow(results)
```

Arguments

results A named list returned by `run_gazepoint_workflow()`.

Value

A tibble with one row containing workflow-level summary counts.

Examples

```
## Not run:
results <- run_gazepoint_workflow(
  export_dir = "C:/Users/YourName/Desktop/gp3_test_exports",
  output_dir = "C:/Users/YourName/Desktop/gp3_outputs",
  prefix = "study1",
  save_plots = TRUE,
  create_report = TRUE
)

summarise_gazepoint_workflow(results)

## End(Not run)
```

`summarise_tracking_quality`*Summarise Gazepoint tracking quality*

Description

Summarise Gazepoint tracking quality

Usage

```
summarise_tracking_quality(data, group_cols = "MEDIA_ID")
```

Arguments

<code>data</code>	A Gazepoint data frame.
<code>group_cols</code>	Grouping columns.

Value

A tibble with validity percentages for available validity columns.

`summarize_gazepoint_coordinate_coverage`*Summarize gaze-coordinate coverage over a screen grid*

Description

Summarizes how much of the screen area is represented by valid gaze coordinates. The helper reports valid-coordinate rates, coordinate ranges, and the proportion of occupied grid cells. It is intended for quality review and documentation, not for inference about attention.

Usage

```
summarize_gazepoint_coordinate_coverage(  
  data,  
  x_col,  
  y_col,  
  screen_width,  
  screen_height,  
  group_cols = NULL,  
  grid_n_x = 10,  
  grid_n_y = 10,  
  include_out_of_bounds = FALSE  
)
```

Arguments

<code>data</code>	A data frame.
<code>x_col, y_col</code>	Character names of gaze-coordinate columns.
<code>screen_width, screen_height</code>	Numeric screen or stimulus dimensions.
<code>group_cols</code>	Optional grouping columns.
<code>grid_n_x, grid_n_y</code>	Number of grid cells along the x and y dimensions.
<code>include_out_of_bounds</code>	If TRUE, finite out-of-bounds coordinates are included in range summaries but not in grid-occupancy calculations.

Value

A data frame with one row per group.

Examples

```
x <- simulate_gazepoint_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)  
summarize_gazepoint_coordinate_coverage(  
  x,  
  x_col = "gaze_x",  
  y_col = "gaze_y",  
  screen_width = 1920,  
  screen_height = 1080,  
  group_cols = "condition"  
)
```

`summarize_gazepoint_face_quality`*Summarise external facial-behaviour data quality*

Description

Returns the overview table from `audit_gazepoint_face_quality()`. The helper accepts either an existing face-quality audit object or data that can be audited directly.

Usage

```
summarize_gazepoint_face_quality(data, ...)
```

```
summarise_gazepoint_face_quality(data, ...)
```

Arguments

<code>data</code>	A <code>gp3_face_quality_audit</code> object, a standardised face data frame, an unstandardised face data frame, or a CSV path.
<code>...</code>	Additional arguments passed to <code>audit_gazepoint_face_quality()</code> when data is not already an audit object.

Value

A tibble with class `gp3_face_quality_summary`.

Examples

```
face <- data.frame(  
  frame = 1:2,  
  timestamp = c(0, 0.033),  
  confidence = c(0.95, 0.90),  
  success = c(1, 1)  
)  
  
summarize_gazepoint_face_quality(face)
```

`summarize_gazepoint_face_reactivity`*Summarise facial-behaviour reactivity between two windows*

Description

Computes baseline-to-response differences from `summarize_gazepoint_face_windows()` output. The function returns one row per group and measure. Reactivity is reported as response minus baseline. The helper summarises technical facial-behaviour measures only and does not infer emotional states.

Usage

```
summarize_gazepoint_face_reactivity(
  data,
  baseline_window,
  response_window,
  group_cols = NULL,
  window_col = NULL,
  measure_cols = NULL,
  statistic = c("mean", "median")
)
```

Arguments

<code>data</code>	A window-summary table returned by <code>summarize_gazepoint_face_windows()</code> .
<code>baseline_window</code>	Value in <code>window_col</code> identifying the baseline window.
<code>response_window</code>	Value in <code>window_col</code> identifying the response window.
<code>group_cols</code>	Optional grouping columns.
<code>window_col</code>	Column identifying window labels or IDs. Auto-detected when possible.
<code>measure_cols</code>	Measure names or summary columns. If <code>NULL</code> , columns ending in the selected statistic suffix are detected automatically.
<code>statistic</code>	Statistic used for reactivity. One of "mean" or "median".

Value

A tibble with one row per group and measure. The returned object has class `gp3_face_reactivity_summary`.

Examples

```
face <- data.frame(
  participant_id = "P001",
  face_time_sec = c(0.00, 0.05, 0.10),
  face_valid = c(TRUE, TRUE, TRUE),
  AU12_r = c(0.1, 0.2, 0.3)
)

windows <- data.frame(
  participant_id = "P001",
  window = c("baseline", "response"),
  window_start_sec = c(0.00, 0.05),
  window_end_sec = c(0.05, 0.15)
)

summary <- summarize_gazepoint_face_windows(
  face,
  windows = windows,
  group_cols = "participant_id",
  window_label_col = "window"
```

```

)

summarize_gazepoint_face_reactivity(
  summary,
  baseline_window = "baseline",
  response_window = "response",
  group_cols = "participant_id"
)

```

```
summarize_gazepoint_face_windows
```

Summarise external facial-behaviour data within analysis windows

Description

Summarises numeric external facial-behaviour variables within time windows. The helper can use a separate window table or data rows that already contain window labels. It is intended for external face-analysis data imported, standardised, and optionally synchronised with Gazeport data. It does not infer facial expressions or emotional states.

Usage

```

summarize_gazepoint_face_windows(
  data,
  windows = NULL,
  time_col = NULL,
  window_start_col = "window_start_sec",
  window_end_col = "window_end_sec",
  group_cols = NULL,
  window_id_col = NULL,
  window_label_col = NULL,
  measure_cols = NULL,
  validity_col = NULL,
  confidence_col = NULL,
  require_valid = TRUE,
  include_empty_windows = TRUE
)

```

Arguments

<code>data</code>	A face-analysis data frame, usually returned by <code>standardize_gazepoint_face_columns()</code> or <code>sync_gazepoint_face_data()</code> .
<code>windows</code>	Optional data frame defining windows. When supplied, it must contain start and end columns.
<code>time_col</code>	Time column in data, in seconds. Auto-detected when possible.
<code>window_start_col</code>	Window-start column in seconds.

`window_end_col` Window-end column in seconds.

`group_cols` Optional grouping columns shared by data and windows.

`window_id_col` Optional window identifier column.

`window_label_col` Optional human-readable window label column.

`measure_cols` Numeric facial-behaviour columns to summarise. When NULL, likely facial-behaviour columns are detected automatically.

`validity_col` Optional validity column. Auto-detected when possible.

`confidence_col` Optional confidence column. Auto-detected when possible.

`require_valid` Should measure summaries use only rows where the validity column is TRUE when such a column is available?

`include_empty_windows` Should windows with no matching rows be kept in the output?

Value

A tibble with one row per group/window and summary columns for each measure. The returned object has class `gp3_face_window_summary`.

Examples

```
face <- data.frame(
  participant_id = "P001",
  face_time_sec = c(0.00, 0.05, 0.10),
  face_confidence = c(0.95, 0.94, 0.93),
  face_valid = c(TRUE, TRUE, TRUE),
  AU12_r = c(0.1, 0.2, 0.3)
)

windows <- data.frame(
  participant_id = "P001",
  window = c("baseline", "response"),
  window_start_sec = c(0.00, 0.05),
  window_end_sec = c(0.05, 0.15)
)

summarize_gazepoint_face_windows(
  face,
  windows = windows,
  group_cols = "participant_id",
  window_label_col = "window"
)
```

`summarize_gazeplot_missingness`*Summarize missingness in Gazeplot-style data*

Description

Computes missingness and observed-data rates for selected columns, optionally within grouping variables such as participant, trial, condition, or AOI. The helper is intended for transparent data-coverage reporting and does not make exclusion decisions.

Usage

```
summarize_gazeplot_missingness(  
  data,  
  cols = NULL,  
  group_cols = NULL,  
  include_group_cols = FALSE  
)
```

```
summarise_gazeplot_missingness(  
  data,  
  cols = NULL,  
  group_cols = NULL,  
  include_group_cols = FALSE  
)
```

Arguments

<code>data</code>	A data frame.
<code>cols</code>	Optional character vector of columns to summarize. If <code>NULL</code> , all non-grouping columns are summarized.
<code>group_cols</code>	Optional character vector of grouping columns.
<code>include_group_cols</code>	If <code>TRUE</code> , grouping columns can also be summarized when <code>cols = NULL</code> .

Value

A data frame with one row per variable and group.

Examples

```
x <- simulate_gazeplot_pupil_data(n_subjects = 2, n_trials = 2, n_time_bins = 5, seed = 1)  
summarize_gazeplot_missingness(  
  x,  
  cols = c("pupil_left", "pupil_right", "pupil"),  
  group_cols = "condition"  
)
```

```
summarize_gazepoint_phase_coverage
```

Summarize task-phase coverage

Description

Summarizes row counts, optional time coverage, and optional value completeness by task phase and optional grouping variables. The helper is intended for documenting task-stage data availability, not for inferential modelling.

Usage

```
summarize_gazepoint_phase_coverage(
  data,
  phase_col = "task_phase",
  group_cols = NULL,
  time_col = NULL,
  value_cols = NULL
)

summarise_gazepoint_phase_coverage(
  data,
  phase_col = "task_phase",
  group_cols = NULL,
  time_col = NULL,
  value_cols = NULL
)
```

Arguments

<code>data</code>	A data frame containing a phase column.
<code>phase_col</code>	Character name of the phase column.
<code>group_cols</code>	Optional grouping columns, such as subject, trial, or condition.
<code>time_col</code>	Optional time column used to summarize phase timing.
<code>value_cols</code>	Optional value columns used to summarize complete-value coverage within phases.

Value

A data frame with one row per group-phase combination.

Examples

```
x <- data.frame(time_ms = c(0, 250, 750, 1250), value = c(1, NA, 3, 4))
windows <- data.frame(phase = c("baseline", "stimulus"), start = c(0, 500), end = c(500, 1500))
segmented <- segment_gazepoint_task_phases(x, "time_ms", windows)
summarize_gazepoint_phase_coverage(segmented, phase_col = "task_phase", time_col = "time_ms")
```

summarize_gazepoint_pupil_response_features
<i>Summarize pupil response features by subject and trial</i>

Description

Extracts common pupil-response features from sample-level pupil data.

Usage

```
summarize_gazepoint_pupil_response_features(  
  data,  
  pupil,  
  time,  
  subject,  
  trial,  
  baseline_window,  
  response_window,  
  condition = NULL,  
  interpolated = NULL  
)
```

Arguments

data	A sample-level data frame.
pupil	Pupil column.
time	Time column.
subject	Subject column.
trial	Trial column.
baseline_window	Numeric vector of length two.
response_window	Numeric vector of length two.
condition	Optional condition column to carry forward.
interpolated	Optional logical/numeric interpolation flag column.

Value

A trial-level data frame of pupil features.

```
summarize_gazepoint_qc_status
```

Summarize Gazepoint QC status

Description

Summarizes pass/warn/fail/info/unknown status counts from a QC bundle or an object-summary table produced by `collect_gazepoint_qc_summaries()`.

Usage

```
summarize_gazepoint_qc_status(qc_bundle)

summarise_gazepoint_qc_status(qc_bundle)
```

Arguments

`qc_bundle` A `gp3_qc_summary_bundle`, object-summary data frame, or list of objects that can be passed to `collect_gazepoint_qc_summaries()`.

Value

A list with overview, status_counts, and object_summary.

Examples

```
x <- list(overview = data.frame(audit_status = "ok"))
summarize_gazepoint_qc_status(list(x))
```

```
summarize_gazepoint_time_clusters
```

Summarize Gazepoint time clusters

Description

`summarize_gazepoint_time_clusters()` provides a compact US-spelling summary of cluster-permutation output from `run_gazepoint_cluster_permutation()`.

Usage

```
summarize_gazepoint_time_clusters(result, alpha = 0.05)
```

Arguments

`result` A result object returned by `run_gazepoint_cluster_permutation()`.
`alpha` Significance threshold used for the descriptive cluster_significant flag.

Value

A data frame with one row per cluster. If no clusters are present, an empty data frame with the expected columns is returned.

Examples

```
# See run_gazepoint_cluster_permutation() for the inferential workflow.
```

```
sync_gazepoint_face_data
```

Synchronise external facial-behaviour data with Gazepoint data

Description

Aligns external facial-behaviour data to Gazepoint rows using either nearest time matching or exact frame matching. The helper is designed for facial data previously imported with `read_gazepoint_face_export()` and standardised with `standardize_gazepoint_face_columns()`. It does not infer facial expressions from Gazepoint exports and does not interpret facial behaviour as emotion.

Usage

```
sync_gazepoint_face_data(
  gazepoint_data,
  face_data,
  method = c("nearest_time", "frame_exact"),
  by = NULL,
  gaze_time_col = NULL,
  face_time_col = NULL,
  gaze_frame_col = NULL,
  face_frame_col = NULL,
  tolerance_sec = 0.05,
  prefix = "face_",
  keep_unmatched = TRUE,
  standardize_face = TRUE
)
```

Arguments

<code>gazepoint_data</code>	Gazepoint data frame, typically a master table, trial table, or sample-level table.
<code>face_data</code>	External facial-behaviour data frame, preferably returned by <code>standardize_gazepoint_face_columns()</code>
<code>method</code>	Synchronisation method. "nearest_time" matches each Gazepoint row to the nearest facial-data row within group. "frame_exact" matches by exact frame index within group.
<code>by</code>	Optional named character vector mapping Gazepoint grouping columns to facial-data grouping columns. For example, <code>c(subject_id = "participant_id")</code> . Use NULL for no grouping.

gaze_time_col	Gazepoint time column. Required for method = "nearest_time" unless auto-detected.
face_time_col	Facial-data time column. Defaults to face_time_sec when available.
gaze_frame_col	Gazepoint frame column. Required for method = "frame_exact" unless auto-detected.
face_frame_col	Facial-data frame column. Defaults to face_frame when available.
tolerance_sec	Maximum allowed absolute time difference in seconds for nearest-time matching.
prefix	Prefix added to non-standard facial-data columns before joining.
keep_unmatched	Should Gazepoint rows without a valid face-data match be retained?
standardize_face	Should face_data be passed through standardize_gazepoint_face_columns() before synchronisation when needed?

Value

A tibble with Gazepoint columns plus matched facial-behaviour columns and synchronisation meta-data. The returned object has class `gp3_face_sync`.

Examples

```
gaze <- data.frame(
  subject_id = "P001",
  time_sec = c(0.00, 0.03, 0.07),
  AOI = c("A", "A", "B")
)

face <- data.frame(
  participant_id = "P001",
  frame = 1:3,
  timestamp = c(0.00, 0.033, 0.066),
  confidence = c(0.95, 0.94, 0.93),
  success = c(1, 1, 1),
  AU12_r = c(0.1, 0.2, 0.3)
)

sync_gazepoint_face_data(
  gaze,
  face,
  by = c(subject_id = "participant_id"),
  gaze_time_col = "time_sec"
)
```

`tidy_gazepoint_model_summary`*Create a tidy model summary for manuscript tables*

Description

Create a compact model-summary object from common fitted models used in gp3tools workflows. The function combines model metadata, fixed-effect summaries, and optional model diagnostics into one structured object.

Usage

```
tidy_gazepoint_model_summary(  
  model,  
  model_name = NULL,  
  conf_level = 0.95,  
  exponentiate = FALSE,  
  drop_intercept = FALSE,  
  include_diagnostics = TRUE,  
  use_dharma = FALSE,  
  dharma_simulations = 250,  
  seed = 123  
)
```

Arguments

<code>model</code>	A fitted model object, or a gp3tools fit object containing a <code>\$model</code> element.
<code>model_name</code>	Optional model label used in returned tables.
<code>conf_level</code>	Confidence level for Wald confidence intervals.
<code>exponentiate</code>	Logical. If TRUE, exponentiate fixed-effect estimates and confidence intervals.
<code>drop_intercept</code>	Logical. If TRUE, remove the intercept from the fixed-effect table.
<code>include_diagnostics</code>	Logical. If TRUE, include model diagnostics when supported.
<code>use_dharma</code>	Logical. If TRUE, request optional DHARMA diagnostics.
<code>dharma_simulations</code>	Number of DHARMA simulations.
<code>seed</code>	Random seed used before DHARMA simulation.

Value

A list with overview, `model_info`, `fixed_effects`, `diagnostics`, and `settings`. The returned object has class `gp3_model_summary`.

transform_gazepoint_aoi_empirical_logit

Transform AOI proportions to empirical logits

Description

Convert bounded AOI proportions into empirical logits for linear mixed models, growth-curve analysis, or other approximately Gaussian time-course models.

Usage

```
transform_gazepoint_aoi_empirical_logit(
  data,
  numerator_col = NULL,
  denominator_col = NULL,
  proportion_col = NULL,
  correction = 0.5,
  pseudo_denominator = 1,
  output_col = "aoi_empirical_logit",
  adjusted_proportion_col = "aoi_proportion_adjusted",
  raw_proportion_col = "aoi_proportion_raw",
  numerator_output_col = "aoi_numerator",
  denominator_output_col = "aoi_denominator",
  status_col = "aoi_empirical_logit_status",
  overwrite = FALSE,
  name = "gazepoint_aoi_empirical_logit"
)
```

Arguments

data	A data frame containing AOI proportions or AOI count data.
numerator_col	Optional numerator column, for example number of samples or fixations inside the AOI.
denominator_col	Optional denominator column, for example total valid samples or total fixations in the window.
proportion_col	Optional bounded AOI proportion column. If numerator_col and denominator_col are supplied, the raw proportion is computed from those columns. If only proportion_col is supplied, a pseudo-denominator is used and recorded in the output.
correction	Positive correction constant added to numerator and non-AOI count. The common empirical-logit correction is 0.5.
pseudo_denominator	Positive pseudo-denominator used only when proportion_col is supplied without denominator_col.

output_col	Name of the empirical-logit output column.
adjusted_proportion_col	Name of the adjusted proportion output column.
raw_proportion_col	Name of the raw proportion output column.
numerator_output_col	Name of the numerator output column used in the transformation.
denominator_output_col	Name of the denominator output column used in the transformation.
status_col	Name of the row-level transformation status column.
overwrite	Logical. If FALSE, the function errors when output columns already exist in data.
name	Character label stored in object attributes.

Details

Binomial GLMMs are usually preferable when numerator and denominator counts are available. This helper is intended for sensitivity analyses, GCA-style models, and linear time-course summaries where a transformed AOI proportion is needed.

Value

A tibble with empirical-logit transformation columns added. The object has class `gp3_aoi_empirical_logit_data`.

validate_gazepoint_binocular_reconstruction

Validate binocular reconstruction using artificial monocular loss

Description

Uses bilateral observations as known reference values, temporarily masks one eye, refits cross-eye calibration without the masked target values, reconstructs the hidden observations, and compares predictions with the held-out values. This provides dataset-specific empirical reconstruction diagnostics rather than assuming that a cross-eye regression is adequate.

Usage

```
validate_gazepoint_binocular_reconstruction(
  data,
  left_col,
  right_col,
  time_col = NULL,
  group_cols = NULL,
  gap_group_cols = NULL,
  fallback_group_cols = NULL,
  direction = c("both", "left_from_right", "right_from_left"),
```

```

mask_prop = 0.2,
mask_mode = c("random", "contiguous"),
block_size = 6L,
repeats = 5L,
seed = 1L,
min_pairs = 30L,
min_unique = 5L,
min_r2 = NULL,
time_unit = c("auto", "milliseconds", "seconds"),
max_gap_ms = Inf,
allow_edge_gaps = TRUE,
allow_extrapolation = FALSE,
valid_min = NULL,
valid_max = NULL
)

```

Arguments

<code>data</code>	A data frame containing binocular pupil measurements.
<code>left_col, right_col</code>	Numeric pupil columns.
<code>time_col</code>	Optional time column; required for meaningful contiguous-gap validation and finite <code>max_gap_ms</code> .
<code>group_cols</code>	Primary calibration and masking groups.
<code>gap_group_cols</code>	Optional groups used to define temporal missing-eye runs during reconstruction; defaults to <code>group_cols</code> .
<code>fallback_group_cols</code>	Optional calibration fallback groups.
<code>direction</code>	"both", "left_from_right", or "right_from_left".
<code>mask_prop</code>	Proportion of bilateral observations masked in each repeat.
<code>mask_mode</code>	"random" masks individual observations; "contiguous" masks short ordered runs.
<code>block_size</code>	Number of samples per attempted contiguous block.
<code>repeats</code>	Number of repeated artificial-missingness evaluations.
<code>seed</code>	Random seed. The caller's RNG state is restored on exit.
<code>min_pairs, min_unique, min_r2</code>	Calibration gates.
<code>time_unit</code>	Time unit used by reconstruction.
<code>max_gap_ms, allow_edge_gaps, allow_extrapolation</code>	Reconstruction gates.
<code>valid_min, valid_max</code>	Optional bounds.

Details

Artificial masking evaluates prediction error where the hidden target is actually known. It does not prove that naturally missing observations are missing at random or that their unobserved values would follow the same error distribution.

Value

A `gp3_binocular_validation` object containing repeat-level metrics, row-level predictions, aggregated summary, and settings.

See Also

[fit_gazepoint_binocular_calibration\(\)](#), [stress_test_gazepoint_binocular_reconstruction\(\)](#)

Examples

```
dat <- simulate_gazepoint_pupil_data(n_subjects = 5, n_trials = 2, seed = 21)
val <- validate_gazepoint_binocular_reconstruction(
  dat, "pupil_left", "pupil_right", time_col = "timestamp_ms",
  group_cols = "subject", mask_prop = 0.1, repeats = 2,
  min_pairs = 20, seed = 9
)
val$summary
```

validate_gazepoint_master

Validate a Gazepoint master sample table

Description

Performs formal validation checks on a Gazepoint master sample-level table created by [as_gazepoint_master\(\)](#) or [create_gazepoint_master\(\)](#). This function is intended as a gate between master-table construction and more advanced steps such as pupil preprocessing, AOI modelling, or statistical analysis.

Usage

```
validate_gazepoint_master(
  master,
  min_valid_sample_pct = 75,
  max_missing_gaze_pct = 25,
  max_missing_pupil_pct = 50,
  max_offscreen_gaze_pct = 25,
  require_pupil = FALSE,
  require_aoi = FALSE,
  fail_on_error = FALSE
)
```

Arguments

<code>master</code>	A Gazepoint master sample-level table.
<code>min_valid_sample_pct</code>	Minimum acceptable percentage of valid gaze samples. Defaults to 75.
<code>max_missing_gaze_pct</code>	Maximum acceptable percentage of missing gaze samples. Defaults to 25.
<code>max_missing_pupil_pct</code>	Maximum acceptable percentage of missing pupil samples. Defaults to 50.
<code>max_offscreen_gaze_pct</code>	Maximum acceptable percentage of off-screen gaze samples. Defaults to 25.
<code>require_pupil</code>	Logical. If TRUE, the validation fails when no usable pupil column is present. Defaults to FALSE.
<code>require_aoi</code>	Logical. If TRUE, the validation fails when no real AOI samples are present. Defaults to FALSE.
<code>fail_on_error</code>	Logical. If TRUE, the function aborts when one or more validation checks fail. Defaults to FALSE.

Value

A named list with:

summary One-row validation summary.

checks A tibble containing all validation checks.

failed_checks Validation checks with status "fail".

warning_checks Validation checks with status "warning".

column_map Detected column mapping used for validation.

Examples

```
## Not run:
master <- create_gazepoint_master(
  gaze_data = results$all_gaze,
  screen_width_px = 1920,
  screen_height_px = 1080
)

validation <- validate_gazepoint_master(master)

validation$summary
validation$checks

## End(Not run)
```

`write_gazepoint_naming_audit`*Write a gp3tools naming-consistency audit*

Description

Write a gp3tools naming-consistency audit

Usage

```
write_gazepoint_naming_audit(x, output_file)
```

Arguments

<code>x</code>	A "gazepoint_naming_audit" object.
<code>output_file</code>	CSV output path.

Value

Normalized path to the written file.

`write_gazepoint_outputs`*Write standard Gazepoint analysis outputs*

Description

Convenience wrapper for exporting standard gp3tools outputs such as sampling checks, tracking quality summaries, flagged quality rows, and AOI tables.

Usage

```
write_gazepoint_outputs(  
  sampling = NULL,  
  quality = NULL,  
  flagged_quality = NULL,  
  aoi_table = NULL,  
  output_dir,  
  prefix = "gazepoint",  
  overwrite = TRUE  
)
```

Arguments

sampling	Sampling-rate table, usually from <code>check_sampling_rate()</code> .
quality	Tracking-quality table, usually from <code>summarise_tracking_quality()</code> .
flagged_quality	Flagged quality table, usually from <code>flag_tracking_quality()</code> .
aoi_table	AOI summary table, usually from <code>summarise_gazepoint_aoi()</code> .
output_dir	Folder where CSV files should be written.
prefix	Optional filename prefix.
overwrite	Logical. If FALSE, stop when files already exist.

Value

A tibble with table names and written file paths.

write_gazepoint_performance_benchmark	<i>Write gp3tools performance benchmark tables</i>
---------------------------------------	--

Description

Write gp3tools performance benchmark tables

Usage

```
write_gazepoint_performance_benchmark(
  x,
  output_dir,
  prefix = "gp3tools-performance"
)
```

Arguments

x	A "gazepoint_performance_benchmark" object.
output_dir	Output directory.
prefix	Filename prefix.

Value

Named character vector of written files.

Index

* datasets

- gazeport_example_aoi_geometry, [149](#)
- gazeport_example_aoi_windows, [150](#)
- gazeport_example_fixations, [150](#)
- gazeport_example_master, [151](#)
- gazeport_example_pupil_windows, [152](#)

- add_gazeport_aoi, [8](#)
- add_gazeport_dynamic_aoi, [9](#)
- add_gazeport_dynamic_aoi(), [29](#)
- add_gazeport_polygon_aoi, [11](#)
- analyse_gazeport_binocular_sensitivity, [12](#)
- analyse_gazeport_binocular_sensitivity(), [26](#), [87](#), [167](#)
- analyze_gazeport_window, [14](#)
- as_gazeport_master, [15](#)
- as_gazeport_master(), [38](#), [115](#), [142](#), [287](#), [311](#)
- audit_gazeport_aoi_coding_matrix, [16](#)
- audit_gazeport_aoi_geometry, [19](#)
- audit_gazeport_aoi_margin_sensitivity, [20](#)
- audit_gazeport_aoi_overlap, [22](#)
- audit_gazeport_aoi_screen_coverage, [23](#)
- audit_gazeport_aoi_window_denominators, [24](#)
- audit_gazeport_binocular_reconstruction, [26](#)
- audit_gazeport_binocular_reconstruction(), [13](#), [167](#), [278](#)
- audit_gazeport_condition_quality_imbalance, [27](#)
- audit_gazeport_design_balance, [28](#)
- audit_gazeport_dynamic_aoi_coverage, [29](#)
- audit_gazeport_event_sync, [30](#)
- audit_gazeport_exclusion_flow, [31](#)
- audit_gazeport_face_quality, [32](#)
- audit_gazeport_face_sync, [33](#)
- audit_gazeport_fixation_reliability, [35](#)
- audit_gazeport_gaze_signal_quality, [36](#)
- audit_gazeport_master, [38](#)
- audit_gazeport_master(), [115](#)
- audit_gazeport_naming_consistency, [39](#)
- audit_gazeport_post_exclusion_balance, [39](#)
- audit_gazeport_pupil_baseline, [41](#)
- audit_gazeport_pupil_drift, [42](#)
- audit_gazeport_pupil_gaps, [44](#)
- audit_gazeport_pupil_imbalance, [45](#)
- audit_gazeport_pupil_overlap_risk, [46](#)
- audit_gazeport_pupil_reliability, [47](#)
- audit_gazeport_screen_bounds, [49](#)
- audit_gazeport_stimulus_luminance, [50](#)
- audit_gazeport_timecourse_grid, [51](#)

- baseline_correct_gazeport_pupil, [51](#)
- baseline_correct_gazeport_pupil(), [41](#), [267](#), [290](#)
- benchmark_gazeport_event_detectors, [53](#)
- benchmark_gazeport_event_detectors(), [172](#), [282](#)
- benchmark_gazeport_export_performance, [55](#)
- bootstrap_gazeport_scanpath_clusters, [56](#)
- bootstrap_gazeport_scanpath_clusters(), [191](#), [292](#)
- bootstrap_gazeport_timecourse, [58](#)

- check_gazeport_bayesian_readiness, [59](#)
- check_gazeport_file_pairs, [60](#)
- check_gazeport_model_convergence, [60](#)

- check_gazepoint_model_overdispersion, 61
- check_gazepoint_model_singularity, 62
- check_gazepoint_performance_regression, 62
- check_gazepoint_real_data_readiness, 63
- check_sampling_rate, 65
- classify_gazepoint_events_hmm, 65
- classify_gazepoint_export, 66
- clean_gazepoint_by_trackloss, 67
- cluster_gazepoint_scanpaths, 68
- cluster_gazepoint_scanpaths(), 57, 122, 190, 260
- collect_gazepoint_qc_summaries, 70
- combine_gazepoint_eyes, 71
- compare_gazepoint_event_detectors, 72
- compare_gazepoint_event_detectors(), 53, 171, 281
- compare_gazepoint_nested_models, 73
- compute_gazepoint_aoi_entropy, 74
- compute_gazepoint_aoi_sequence_metrics, 75
- compute_gazepoint_aoi_transition_matrix, 77
- compute_gazepoint_saccade_metrics, 78
- compute_gazepoint_scanpath_geometry, 79
- compute_gazepoint_scanpath_similarity, 68, 69, 80
- compute_gazepoint_sequence_complexity, 81
- compute_gazepoint_sequence_distance, 82
- compute_gazepoint_sequence_recurrence, 83
- compute_gazepoint_time_varying_transition_matrix, 84
- compute_gazepoint_transition_network_metrics, 85
- compute_transition_matrix, 86
- construct_gazepoint_combined_pupil, 86
- construct_gazepoint_combined_pupil(), 236
- create_gazepoint_analysis_decision_audit, 88
- create_gazepoint_bayesian_sap, 89
- create_gazepoint_brms_template, 90
- create_gazepoint_cross_package_report, 90
- create_gazepoint_event_review_template, 91
- create_gazepoint_face_reporting_checklist, 92
- create_gazepoint_hddm_fit_script, 93
- create_gazepoint_markovchain_object, 94
- create_gazepoint_master, 96
- create_gazepoint_master(), 38, 115, 142, 287, 311
- create_gazepoint_preprocessing_multiverse, 97
- create_gazepoint_preprocessing_registry, 98
- create_gazepoint_preprocessing_registry(), 144
- create_gazepoint_report, 100
- create_gazepoint_reporting_checklist, 101
- detect_gazepoint_blinks, 102
- detect_gazepoint_blinks(), 156, 226
- detect_gazepoint_fixations_ivt, 103
- detect_gazepoint_fixations_ivt(), 104
- detect_gazepoint_fixations_velocity, 104
- detect_gazepoint_fixations_velocity(), 226
- diagnose_gazepoint_binocular_pupil, 105
- diagnose_gazepoint_binocular_pupil(), 167
- diagnose_gazepoint_cluster_design, 107
- diagnose_gazepoint_gamm, 108
- diagnose_gazepoint_glmm, 109
- downsample_gazepoint_pupil, 110
- estimate_gazepoint_cluster_offset, 111
- estimate_gazepoint_cluster_onset, 111
- estimate_gazepoint_divergence_point, 112
- export_gazepoint_cluster_results, 113
- export_gazepoint_heatmap_png, 114
- export_gazepoint_master_audit, 115
- export_gazepoint_mne_cluster_input, 116
- export_gazepoint_model_tables, 117

- export_gazepoint_permuco_cluster_input, 118
- export_gazepoint_permutes_cluster_input, 119
- export_gazepoint_tables, 120
- export_gazepoint_to_bids, 120
- extract_gazepoint_representative_scanpaths, 121
- filter_gazepoint_cnn_uncertainty, 122
- fit_gazepoint_aoi_brms, 123
- fit_gazepoint_aoi_gamm, 124
- fit_gazepoint_aoi_model_sensitivity, 126
- fit_gazepoint_aoi_window_glmm, 127
- fit_gazepoint_binocular_calibration, 129
- fit_gazepoint_binocular_calibration(), 106, 235, 311
- fit_gazepoint_brms_model, 130
- fit_gazepoint_face_window_lmm, 131
- fit_gazepoint_gca, 132
- fit_gazepoint_multimodal_response_model, 133
- fit_gazepoint_pupil_gamm, 135
- fit_gazepoint_pupil_pfe_gamm, 136
- fit_gazepoint_pupil_window_lmm, 137
- fit_gazepoint_pupil_window_sensitivity, 139
- fit_gazepoint_transition_count_nb_sensitivity, 141
- flag_gazepoint_pupil, 142
- flag_gazepoint_pupil(), 51, 157, 267, 290
- flag_gazepoint_pupil_artifacts, 143
- flag_gazepoint_pupil_artifacts(), 157
- flag_gazepoint_pupil_hampel, 146
- flag_gazepoint_sequence_anomalies, 147
- flag_tracking_quality, 148
- gazepoint_example_aoi_geometry, 149
- gazepoint_example_aoi_windows, 150
- gazepoint_example_fixations, 150
- gazepoint_example_master, 151
- gazepoint_example_pupil_windows, 152
- gp3tools_naming_policy, 152
- gp3tools_performance_limits, 153
- gp3tools_performance_limits(), 55
- harmonize_gazepoint_screen_coordinates, 153
- hclust, 69
- impute_gazepoint_pupil_gp, 154
- inspect_gazepoint_columns, 155
- interpolate_gazepoint_blinks, 156
- interpolate_gazepoint_pupil, 157
- interpolate_gazepoint_pupil(), 45, 51, 52, 267, 290
- interpolate_gazepoint_pupil_pchip, 158
- launch_gazepoint_qc_dashboard, 160
- mean_gazepoint_pupil, 160
- plot_gazepoint_aoi_gamm, 161
- plot_gazepoint_aoi_timeline, 163
- plot_gazepoint_aoi_transition_matrix, 164
- plot_gazepoint_aoi_verification, 165
- plot_gazepoint_binocular_diagnostics, 167
- plot_gazepoint_cluster_null_distribution, 168
- plot_gazepoint_cluster_permutation, 169
- plot_gazepoint_cluster_results, 169
- plot_gazepoint_event_detector_agreement, 171
- plot_gazepoint_event_detector_benchmark, 171
- plot_gazepoint_face_quality, 172
- plot_gazepoint_gca, 173
- plot_gazepoint_heatmap, 174
- plot_gazepoint_heatmap_overlay, 176
- plot_gazepoint_missingness_profile, 177
- plot_gazepoint_model_predictions, 178
- plot_gazepoint_model_residuals, 180
- plot_gazepoint_multiverse_results, 180
- plot_gazepoint_phase_timeline, 181
- plot_gazepoint_pupil_preprocessing, 182
- plot_gazepoint_pupil_status, 184
- plot_gazepoint_pupil_timecourse, 185
- plot_gazepoint_qc_overview, 186
- plot_gazepoint_scanpath, 187
- plot_gazepoint_scanpath_cluster_stability, 191
- plot_gazepoint_scanpath_clusters, 190

- plot_gazepoint_scanpaths, 188
- plot_gazepoint_stimulus_layout_qc, 192
- plot_gazepoint_time_series, 194
- plot_gazepoint_time_varying_effect, 195
- plot_sampling_rate, 196
- plot_tracking_quality, 197
- plot_transition_heatmap, 197
- prepare_gazepoint_aoi_gamm_data, 198
- prepare_gazepoint_aoi_glm_data, 200
- prepare_gazepoint_aoi_sequences, 201
- prepare_gazepoint_cluster_data, 202
- prepare_gazepoint_eyetools_data, 204
- prepare_gazepoint_eyetrackingr_data, 205
- prepare_gazepoint_fixation_aligned_data, 207
- prepare_gazepoint_gazer_data, 208
- prepare_gazepoint_gca_data, 210
- prepare_gazepoint_gpbiometrics_bridge, 211
- prepare_gazepoint_gpbiometrics_bridge(), 251
- prepare_gazepoint_hddm_export, 212
- prepare_gazepoint_heatmap_data, 213
- prepare_gazepoint_hmm_data, 214
- prepare_gazepoint_multimodal_data, 215
- prepare_gazepoint_pupil_gamm_data, 218
- prepare_gazepoint_pupil_window_model_data, 220
- prepare_gazepoint_pupillometryr_data, 217
- prepare_gazepoint_semimarkov_data, 221
- prepare_gazepoint_timecourse_test_data, 223
- prepare_gazepoint_traminer_data, 224
- preprocess_gazepoint_signals, 225
- read_gazepoint, 227
- read_gazepoint_face_export, 228
- read_gazepoint_folder, 229
- read_gazepoint_summary, 230
- recalibrate_gazepoint_gaze, 230
- recommend_gazepoint_exclusions, 232
- recommend_gazepoint_model_family, 233
- reconstruct_gazepoint_binocular_pupil, 234
- reconstruct_gazepoint_binocular_pupil(), 26, 87, 106, 130, 168, 278
- regress_gazepoint_pupils, 236
- report_gazepoint_cluster_permutation, 237
- report_gazepoint_face_qc, 238
- report_gazepoint_missingness, 239
- report_gazepoint_multiverse, 240
- report_gazepoint_phase_coverage, 241
- report_gazepoint_qc_overview, 242
- run_gazepoint_aoi_multiverse, 242
- run_gazepoint_cluster_permutation, 244
- run_gazepoint_cluster_permutation_anova, 245
- run_gazepoint_cluster_permutation_covariate_adjusted, 245
- run_gazepoint_cluster_permutation_lmer, 246
- run_gazepoint_cluster_permutation_parallel, 246
- run_gazepoint_cluster_threshold_sensitivity, 247
- run_gazepoint_eyetools_fixation_detection, 247
- run_gazepoint_gazer_crosscheck, 249
- run_gazepoint_gpbiometrics_workflow, 250
- run_gazepoint_model_leave_one_out, 252
- run_gazepoint_multidimensional_cluster_permutation, 253
- run_gazepoint_pupil_multiverse, 253
- run_gazepoint_tfce, 254
- run_gazepoint_workflow, 255
- run_gazepoint_workflow(), 294
- save_gazepoint_plots, 256
- segment_gazepoint_task_phases, 257
- select_gazepoint_adaptive_trial, 259
- select_gazepoint_scanpath_clusters, 260
- simulate_gazepoint_cluster_timecourse_data, 261
- simulate_gazepoint_data, 262
- simulate_gazepoint_fixations, 263
- simulate_gazepoint_pupil_data, 264
- smooth_gazepoint_coordinate, 265
- smooth_gazepoint_coordinate(), 226
- smooth_gazepoint_pupil, 267
- smooth_gazepoint_pupil(), 226, 290
- standardise_gazepoint_names, 268

- standardize_gazepoint_face_columns, 269
- stats::cor(), 36
- stats::hclust(), 260
- stress_test_gazepoint_binocular_reconstruction, 270
- stress_test_gazepoint_binocular_reconstruction(), 311
- summarise_aoi_samples, 271
- summarise_fixations, 272
- summarise_gazepoint_aoi, 272
- summarise_gazepoint_aoi_entries, 273
- summarise_gazepoint_aoi_transitions, 274
- summarise_gazepoint_aoi_trial_features, 275
- summarise_gazepoint_aoi_windows, 276
- summarise_gazepoint_binocular_reporting, 278
- summarise_gazepoint_binocular_reporting(), 26
- summarise_gazepoint_clusters, 279
- summarise_gazepoint_coordinate_coverage, 280
- summarise_gazepoint_emmeans, 280
- summarise_gazepoint_event_detector_agreement, 281
- summarise_gazepoint_event_detector_benchmark, 282
- summarise_gazepoint_face_quality
(summarize_gazepoint_face_quality), 297
- summarise_gazepoint_face_reactivity, 282
- summarise_gazepoint_face_windows, 283
- summarise_gazepoint_fixation_trials, 283
- summarise_gazepoint_fixed_effects, 285
- summarise_gazepoint_markovchain, 286
- summarise_gazepoint_missingness
(summarize_gazepoint_missingness), 301
- summarise_gazepoint_multiverse_results, 287
- summarise_gazepoint_phase_coverage
(summarize_gazepoint_phase_coverage), 302
- summarise_gazepoint_pupil, 287
- summarise_gazepoint_pupil_response_features, 288
- summarise_gazepoint_pupil_trial_features, 289
- summarise_gazepoint_pupil_windows, 290
- summarise_gazepoint_qc_status
(summarize_gazepoint_qc_status), 304
- summarise_gazepoint_scanpath_cluster_stability, 292
- summarise_gazepoint_semimarkov, 293
- summarise_gazepoint_time_clusters, 294
- summarise_gazepoint_workflow, 294
- summarise_tracking_quality, 295
- summarize_gazepoint_coordinate_coverage, 295
- summarize_gazepoint_coordinate_coverage(), 280
- summarize_gazepoint_face_quality, 297
- summarize_gazepoint_face_reactivity, 297
- summarize_gazepoint_face_reactivity(), 282
- summarize_gazepoint_face_windows, 299
- summarize_gazepoint_face_windows(), 283
- summarize_gazepoint_missingness, 301
- summarize_gazepoint_phase_coverage, 302
- summarize_gazepoint_pupil_response_features, 303
- summarize_gazepoint_pupil_response_features(), 289
- summarize_gazepoint_qc_status, 304
- summarize_gazepoint_time_clusters, 304
- summarize_gazepoint_time_clusters(), 294
- sync_gazepoint_face_data, 305
- tidy_gazepoint_model_summary, 307
- transform_gazepoint_aoi_empirical_logit, 308
- validate_gazepoint_binocular_reconstruction, 309
- validate_gazepoint_binocular_reconstruction(), 130, 167, 168, 236, 271, 278
- validate_gazepoint_master, 311
- validate_gazepoint_master(), 115, 116

`write_gazepoint_naming_audit`, [313](#)
`write_gazepoint_outputs`, [313](#)
`write_gazepoint_performance_benchmark`,
[314](#)