

Package ‘ETDQualitizer’

May 7, 2026

Type Package

Title Automated Eye Tracking Data Quality Determination for
Screen-Based Eye Trackers

Version 0.9.0

Maintainer Diederick Niehorster <diederick_c.niehorster@humlab.lu.se>

Description Compute common data quality metrics for accuracy, precision and data loss
for screen-based eye trackers. Supports input data both in pixels on the screen and
in degrees, output measures are (where appropriate) expressed as angles in degrees.

License MIT + file LICENSE

URL <https://github.com/dcnieho/ETDQualitizer>

BugReports <https://github.com/dcnieho/ETDQualitizer/issues>

Encoding UTF-8

Imports R6, stats

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.2

NeedsCompilation no

Author Diederick Niehorster [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0002-4672-8756>>)

Repository CRAN

Date/Publication 2025-09-09 13:50:07 UTC

Contents

accuracy	2
bcea	3
compute_data_quality_from_validation	3
DataQuality	4
data_loss_from_expected	9
data_loss_from_invalid	10

effective_frequency	10
ETDQ_version	11
Fick_to_vector	11
precision_using_moving_window	12
report_data_quality_table	13
rms_s2s	14
ScreenConfiguration	14
std	18
vector_to_Fick	19

Index	20
--------------	-----------

accuracy	<i>Compute Gaze Accuracy</i>
----------	------------------------------

Description

Calculates the angular offset between gaze and target directions.

Usage

```
accuracy(azi, ele, target_azi, target_ele, central_tendency_fun = mean)
```

Arguments

azi	Gaze azimuth in degrees.
ele	Gaze elevation in degrees.
target_azi	Target azimuth in degrees.
target_ele	Target elevation in degrees.
central_tendency_fun	Function to compute central tendency (default: mean).

Value

A list with offset, offset_azi, and offset_ele, the total, horizontal and vertical offset of gaze from the target (in degrees).

Examples

```
accuracy(c(1, 2), c(1, 2), 0, 0)
```

bcea *Bivariate Contour Ellipse Area (BCEA)*

Description

Computes BCEA and ellipse parameters for gaze precision.

Usage

```
bcea(azi, ele, P = 0.68)
```

Arguments

azi	Azimuth values in degrees.
ele	Elevation values in degrees.
P	Cumulative probability (default: 0.68).

Value

A list with the BCEA (area) and additional info about the BCEA ellipse: orientation, ax1, ax2, and aspect_ratio.

Examples

```
bcea(rnorm(100), rnorm(100))
```

compute_data_quality_from_validation
Compute Data Quality Metrics from Validation Data

Description

This function computes a set of data quality metrics for gaze data collected during the PsychoPy validation procedure that is provided in the ETDQualitizer repository on github (<https://github.com/dnieho/ETDQualitizer/tree/>). It evaluates accuracy, precision, and optionally data loss and effective sampling frequency, per eye and per target.

Usage

```
compute_data_quality_from_validation(  
  gaze,  
  unit,  
  screen = NULL,  
  advanced = FALSE,  
  include_data_loss = FALSE  
)
```

Arguments

<code>gaze</code>	A 'data.frame' containing gaze data. Must include columns 'target_id', 'tar_x', 'tar_y', 'timestamp', and eye-specific columns such as 'left_x', 'left_y', 'right_x', 'right_y'. Timestamps should be provided in milliseconds.
<code>unit</code>	A character string specifying the unit of measurement for gaze and target coordinates in the gaze data.frame. Must be either "pixels" or "degrees".
<code>screen</code>	An optional 'ScreenConfiguration' object or numeric scalar used to convert pixel coordinates to degrees. Required if 'unit == "pixels"'.
<code>advanced</code>	Logical. If 'TRUE', all available metrics are returned. If 'FALSE', only a simplified subset is included (default is FALSE).
<code>include_data_loss</code>	Logical. If 'TRUE', includes data loss and effective frequency metrics in the output (default is FALSE).

Details

This function uses the following methods in the 'DataQuality' class to compute the returned results: 'accuracy()', 'precision_RMS_S2S()', 'precision_STD()', 'precision_BCEA()', 'data_loss_from_invalid()', and 'effective_frequency()'.

Value

A 'data.frame' with one row per eye-target combination, containing computed metrics: - 'eye', 'target_id': identifiers - 'offset', 'offset_x', 'offset_y': accuracy metrics ('offset_x', 'offset_y' only if 'advanced' is 'TRUE') - 'rms_s2s', 'rms_s2s_x', 'rms_s2s_y': precision (RMS sample-to-sample) ('rms_s2s_x', 'rms_s2s_y' only if 'advanced' is 'TRUE') - 'std', 'std_x', 'std_y': precision (standard deviation) ('std_x', 'std_y' only if 'advanced' is 'TRUE') - 'bcea', 'bcea_orientation', 'bcea_ax1', 'bcea_ax2', 'bcea_aspect_ratio': precision (BCEA metrics) ('bcea_orientation', 'bcea_ax1', 'bcea_ax2', 'bcea_aspect_ratio' only if 'advanced' is 'TRUE') - 'data_loss', 'effective_frequency': optional metrics if 'include_data_loss = TRUE'

Examples

```
## Not run:
dq <- compute_data_quality_from_validation(gaze_data, unit = "pixels", screen = my_screen_config)

## End(Not run)
```

DataQuality

R6 class for calculating Data Quality from a gaze data segment

Description

Provides methods for assessing the quality of gaze data, including accuracy, precision, data loss, and effective sampling frequency.

Public fields

- `timestamps` Vector of timestamps in seconds. Samples with missing data should not be removed, or the RMS calculation would be incorrect.
- `azi` Vector of azimuth angles in degrees (Fick angles). Missing data should be coded as NA, not using some special value such as (0,0) or (-xres,-yres).
- `ele` Vector of elevation angles in degrees (Fick angles). Missing data should be coded as NA, not using some special value such as (0,0) or (-xres,-yres).

Methods**Public methods:**

- `DataQuality$new()`
- `DataQuality$accuracy()`
- `DataQuality$precision_RMS_S2S()`
- `DataQuality$precision_STD()`
- `DataQuality$precision_BCEA()`
- `DataQuality$data_loss_from_invalid()`
- `DataQuality$data_loss_from_expected()`
- `DataQuality$effective_frequency()`
- `DataQuality$get_duration()`
- `DataQuality$precision_using_moving_window()`
- `DataQuality$clone()`

Method `new()`: Creates a new `DataQuality` object from gaze data and timestamps.

Usage:

```
DataQuality$new(gaze_x, gaze_y, timestamps, unit, screen = NULL)
```

Arguments:

`gaze_x` Horizontal gaze positions (pixels or degrees).

`gaze_y` Vertical gaze positions (pixels or degrees).

`timestamps` Vector of timestamps in seconds.

`unit` Unit of gaze data: either "pixels" or "degrees".

`screen` Optional `ScreenConfiguration` object, required if unit is "pixels".

Returns: A new `DataQuality` object.

Examples:

```
dq <- DataQuality$new(gaze_x, gaze_y, timestamps, unit = "pixels", screen = sc)
```

Method `accuracy()`: Calculates the accuracy of gaze data relative to a known target location.

Usage:

```
DataQuality$accuracy(target_azi, target_ele, central_tendency_fun = mean)
```

Arguments:

`target_azi` Target azimuth in degrees.

`target_ele` Target elevation in degrees.

central_tendency_fun Function to compute central tendency (e.g., mean, median).

Returns: Accuracy in degrees.

Examples:

```
dq$accuracy(0, 0)
```

Method precision_RMS_S2S(): Calculates precision as root mean square of sample-to-sample distances

Usage:

```
DataQuality$precision_RMS_S2S(central_tendency_fun = mean)
```

Arguments:

central_tendency_fun Function to compute central tendency (e.g., mean, median).

Returns: Precision in degrees.

Examples:

```
dq$precision_RMS_S2S()
```

Method precision_STD(): Calculates precision as standard deviation of gaze positions.

Usage:

```
DataQuality$precision_STD()
```

Returns: Standard deviation in degrees.

Examples:

```
dq$precision_STD()
```

Method precision_BCEA(): Calculates the Bivariate Contour Ellipse Area (BCEA) and ellipse parameters for gaze precision.

Usage:

```
DataQuality$precision_BCEA(P = 0.68)
```

Arguments:

P Proportion of data to include in the ellipse (default is 0.68).

Returns: BCEA in degrees-squared.

Examples:

```
dq$precision_BCEA()
```

Method data_loss_from_invalid(): Calculates the proportion of missing data (coded as NA).

Usage:

```
DataQuality$data_loss_from_invalid()
```

Returns: Proportion of missing samples.

Examples:

```
dq$data_loss_from_invalid()
```

Method data_loss_from_expected(): Estimates data loss based on expected number of samples given the duration and sampling frequency.

Usage:

```
DataQuality$data_loss_from_expected(frequency)
```

Arguments:

frequency Expected sampling frequency in Hz.

Returns: Proportion of missing samples.

Examples:

```
dq$data_loss_from_expected(500)
```

Method `effective_frequency()`: Calculates the effective sampling frequency based on timestamps.

Usage:

```
DataQuality$effective_frequency()
```

Returns: Effective frequency in Hz.

Examples:

```
dq$effective_frequency()
```

Method `get_duration()`: Computes the total duration of the gaze recording, including the last sample.

Usage:

```
DataQuality$get_duration()
```

Returns: Duration in seconds.

Examples:

```
dq$get_duration()
```

Method `precision_using_moving_window()`: Calculates precision using a moving window approach.

Usage:

```
DataQuality$precision_using_moving_window(  
  window_length,  
  metric,  
  aggregation_fun = median,  
  ...  
)
```

Arguments:

window_length Length of the moving window in number of samples.

metric Precision metric to use ("RMS-S2S", "STD", or "BCEA").

aggregation_fun Function to aggregate windowed precision values (e.g., median).

... Additional arguments passed to the precision metric function.

Returns: Precision value.

Examples:

```
dq$precision_using_moving_window(0.2, "RMS-S2S")
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
DataQuality$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
sc <- ScreenConfiguration$new(500, 300, 1920, 1080, 600)
gaze_x <- c(0, 1, -1)
gaze_y <- c(0, 1, -1)
timestamps <- c(0, 1, 2)
dq <- DataQuality$new(gaze_x, gaze_y, timestamps, unit = "pixels", screen = sc)
dq$accuracy(0, 0)
dq$precision_RMS_S2S()
dq$data_loss_from_invalid()

## -----
## Method `DataQuality$new`
## -----

dq <- DataQuality$new(gaze_x, gaze_y, timestamps, unit = "pixels", screen = sc)

## -----
## Method `DataQuality$accuracy`
## -----

dq$accuracy(0, 0)

## -----
## Method `DataQuality$precision_RMS_S2S`
## -----

dq$precision_RMS_S2S()

## -----
## Method `DataQuality$precision_STD`
## -----

dq$precision_STD()

## -----
## Method `DataQuality$precision_BCEA`
## -----

dq$precision_BCEA()

## -----
## Method `DataQuality$data_loss_from_invalid`
## -----
```

```

dq$data_loss_from_invalid()

## -----
## Method `DataQuality$data_loss_from_expected`
## -----

dq$data_loss_from_expected(500)

## -----
## Method `DataQuality$effective_frequency`
## -----

dq$effective_frequency()

## -----
## Method `DataQuality$get_duration`
## -----

dq$get_duration()

## -----
## Method `DataQuality$precision_using_moving_window`
## -----

dq$precision_using_moving_window(0.2, "RMS-S2S")

```

data_loss_from_expected

Compute Data Loss from Expected Sample Count

Description

Calculates data loss based on expected number of samples.

Usage

```
data_loss_from_expected(a, b, duration, frequency)
```

Arguments

a	Horizontal gaze values (e.g. azimuth or horizontal coordinate in pixels or mm).
b	Vertical gaze values (e.g. azimuth or horizontal coordinate in pixels or mm).
duration	Duration in seconds.
frequency	Sampling frequency in Hz.

Value

Percentage of data loss.

Examples

```
data_loss_from_expected(c(1, NA, 3), c(1, 2, NA), duration = 1, frequency = 3)
```

```
data_loss_from_invalid
```

Compute Data Loss from number of invalid samples.

Description

Calculates percentage of missing gaze samples.

Usage

```
data_loss_from_invalid(a, b)
```

Arguments

a Horizontal gaze values (e.g. azimuth or horizontal coordinate in pixels or mm).
b Vertical gaze values (e.g. azimuth or horizontal coordinate in pixels or mm).

Value

Percentage of missing samples.

Examples

```
data_loss_from_invalid(c(1, NA, 3), c(1, 2, NA))
```

```
effective_frequency    Compute Effective Sampling Frequency
```

Description

Calculates effective frequency based on valid samples.

Usage

```
effective_frequency(a, b, duration)
```

Arguments

a Horizontal gaze values (e.g. azimuth or horizontal coordinate in pixels or mm).
b Vertical gaze values (e.g. azimuth or horizontal coordinate in pixels or mm).
duration Duration in seconds.

Value

Effective frequency in Hz.

Examples

```
effective_frequency(c(1, NA, 3), c(1, 2, NA), duration = 1)
```

ETDQ_version	<i>Get ETDQualitizer Version</i>
--------------	----------------------------------

Description

Returns the current version string of the ETDQualitizer tool.

Usage

```
ETDQ_version()
```

Value

A character string representing the version number.

Examples

```
ETDQ_version()
```

Fick_to_vector	<i>Convert Fick Angles to 3D Vector</i>
----------------	---

Description

Converts azimuth and elevation angles (in degrees) to a 3D unit vector.

Usage

```
Fick_to_vector(azi, ele, rho = 1)
```

Arguments

azi	Azimuth angle in degrees.
ele	Elevation angle in degrees.
rho	Radius (default is 1.0).

Value

A list with components x, y, and z.

Examples

```
Fick_to_vector(30, 10)
```

```
precision_using_moving_window
```

Precision Using Moving Window

Description

Computes gaze precision using a moving window and selected metric.

Usage

```
precision_using_moving_window(  
  azi,  
  ele,  
  window_length,  
  metric,  
  aggregation_fun = median,  
  ...  
)
```

Arguments

azi	Azimuth values.
ele	Elevation values.
window_length	Window size in samples.
metric	Precision metric: "RMS-S2S", "STD", or "BCEA".
aggregation_fun	Function to aggregate precision values across the windows (default: median).
...	Additional arguments passed to metric function.

Value

Aggregated precision value.

Examples

```
precision_using_moving_window(rnorm(100), rnorm(100), 10, "STD")
```

`report_data_quality_table`*Summarize and Report Data Quality Metrics*

Description

This function summarizes data quality metrics from a validation procedure by computing averages per participant and generating descriptive statistics across participants. It also returns a formatted textual summary suitable for reporting.

Usage

```
report_data_quality_table(dq_table)
```

Arguments

`dq_table` A 'data.frame' containing data quality metrics. Must include columns 'file', 'eye', 'target_id', and relevant numeric metrics such as 'offset', 'rms_s2s', and 'std'. This would generally be created by concatenating the output of the `compute_data_quality_from_validation()` for multiple files.

Details

The summary text excludes BCEA and data loss metrics. BCEA is considered a niche metric and data loss is best reported across the full dataset rather than just the validation subset.

Value

A named list with two elements:

txt A character string summarizing key metrics (accuracy, RMS-S2S precision, STD precision).

measures A list containing:

- **all**: A data frame with per-participant averages (grouped by 'file').
- **mean, std, min, max**: Named numeric vectors with summary statistics across participants.

Examples

```
## Not run:
result <- report_data_quality_table(dq_table)
cat(result$txt)
head(result$measures$all)

## End(Not run)
```

 rms_s2s

RMS of Sample-to-Sample Differences

Description

Computes root mean square of differences between successive gaze samples.

Usage

```
rms_s2s(azi, ele, central_tendency_fun = mean)
```

Arguments

`azi` Azimuth values in degrees.
`ele` Elevation values in degrees.
`central_tendency_fun` Function to compute central tendency (default: mean).

Value

A list with `rms`, `rms_azi`, and `rms_ele`, the total RMS of sample-to-sample distances and that of the azimuthal and elevation components (all in degrees).

Examples

```
rms_s2s(c(1, 2, 3), c(1, 2, 3))
```

 ScreenConfiguration *R6 Screen Configuration Class*

Description

Provides methods for converting between pixel, millimeter, and degree units.

Public fields

`screen_size_x_mm` Screen width in mm.
`screen_size_y_mm` Screen height in mm.
`screen_res_x_pix` Horizontal screen resolution in pixels.
`screen_res_y_pix` Vertical screen resolution in pixels.
`viewing_distance_mm` Viewing distance in mm.

Methods**Public methods:**

- `ScreenConfiguration$new()`
- `ScreenConfiguration$pix_to_mm()`
- `ScreenConfiguration$pix_to_deg()`
- `ScreenConfiguration$mm_to_deg()`
- `ScreenConfiguration$mm_to_pix()`
- `ScreenConfiguration$deg_to_mm()`
- `ScreenConfiguration$deg_to_pix()`
- `ScreenConfiguration$screen_extents()`
- `ScreenConfiguration$clone()`

Method `new()`: Creates a new `ScreenConfiguration` object with screen and viewing distance parameters.

Usage:

```
ScreenConfiguration$new(  
  screen_size_x_mm,  
  screen_size_y_mm,  
  screen_res_x_pix,  
  screen_res_y_pix,  
  viewing_distance_mm  
)
```

Arguments:

`screen_size_x_mm` Screen width in millimeters.
`screen_size_y_mm` Screen height in millimeters.
`screen_res_x_pix` Horizontal screen resolution in pixels.
`screen_res_y_pix` Vertical screen resolution in pixels.
`viewing_distance_mm` Viewing distance in millimeters.

Returns: A new `ScreenConfiguration` object.

Examples:

```
sc <- ScreenConfiguration$new(500, 300, 1920, 1080, 600)
```

Method `pix_to_mm()`: Converts pixel coordinates to millimeter coordinates on the screen.

Usage:

```
ScreenConfiguration$pix_to_mm(x, y)
```

Arguments:

`x` Horizontal pixel coordinate.
`y` Vertical pixel coordinate.

Returns: A list with `x` and `y` in millimeters.

Examples:

```
sc$pix_to_mm(960, 540)
```

Method `pix_to_deg()`: Converts pixel coordinates to an angular gaze direction in degrees.

Usage:

```
ScreenConfiguration$pix_to_deg(x, y)
```

Arguments:

x Horizontal pixel coordinate.

y Vertical pixel coordinate.

Returns: A list with azimuth ("azi") and elevation ("ele") in degrees.

Examples:

```
sc$pix_to_deg(960, 540)
```

Method `mm_to_deg()`: Converts millimeter coordinates to an angular gaze direction in degrees.

Usage:

```
ScreenConfiguration$mm_to_deg(x, y)
```

Arguments:

x Horizontal position in millimeters.

y Vertical position in millimeters.

Returns: A list with azimuth ("azi") and elevation ("ele") in degrees.

Examples:

```
sc$mm_to_deg(100, 50)
```

Method `mm_to_pix()`: Converts millimeter coordinates on the screen to pixel coordinates.

Usage:

```
ScreenConfiguration$mm_to_pix(x, y)
```

Arguments:

x Horizontal position in millimeters.

y Vertical position in millimeters.

Returns: A list with x and y in pixels.

Examples:

```
sc$mm_to_pix(100, 50)
```

Method `deg_to_mm()`: Converts an angular gaze direction in degrees to millimeter coordinates on the screen.

Usage:

```
ScreenConfiguration$deg_to_mm(azi, ele)
```

Arguments:

azi Azimuth in degrees (Fick angles).

ele Elevation in degrees (Fick angles).

Returns: A list with x and y in millimeters.

Examples:

```
sc$deg_to_mm(2, 1)
```

Method `deg_to_pix()`: Converts an angular gaze direction in degrees to pixel coordinates.

Usage:

```
ScreenConfiguration$deg_to_pix(azi, ele)
```

Arguments:

`azi` Azimuth in degrees (Fick angles).

`ele` Elevation in degrees (Fick angles).

Returns: A list with x and y in pixels.

Examples:

```
sc$deg_to_pix(2, 1)
```

Method `screen_extents()`: Computes the horizontal and vertical extents of the screen (in degrees).

Usage:

```
ScreenConfiguration$screen_extents()
```

Returns: A list with width and height in degrees.

Examples:

```
sc$screen_extents()
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ScreenConfiguration$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
sc <- ScreenConfiguration$new(500, 300, 1920, 1080, 600)
sc$pix_to_deg(960, 540)

## -----
## Method `ScreenConfiguration$new`
## -----

sc <- ScreenConfiguration$new(500, 300, 1920, 1080, 600)

## -----
## Method `ScreenConfiguration$pix_to_mm`
## -----

sc$pix_to_mm(960, 540)

## -----
## Method `ScreenConfiguration$pix_to_deg`
## -----

sc$pix_to_deg(960, 540)
```

```

## -----
## Method `ScreenConfiguration$mm_to_deg`
## -----

sc$mm_to_deg(100, 50)

## -----
## Method `ScreenConfiguration$mm_to_pix`
## -----

sc$mm_to_pix(100, 50)

## -----
## Method `ScreenConfiguration$deg_to_mm`
## -----

sc$deg_to_mm(2, 1)

## -----
## Method `ScreenConfiguration$deg_to_pix`
## -----

sc$deg_to_pix(2, 1)

## -----
## Method `ScreenConfiguration$screen_extents`
## -----

sc$screen_extents()

```

std

Standard Deviation of Gaze Samples

Description

Computes standard deviation of azimuth and elevation.

Usage

```
std(azi, ele)
```

Arguments

azi	Azimuth values in degrees.
ele	Elevation values in degrees.

Value

A list with `std`, `std_azi`, and `std_ele`, the total STD of sample-to-sample distances and that of the azimuthal and elevation components (all in degrees).

Examples

```
std(c(1, 2, 3), c(1, 2, 3))
```

vector_to_Fick	<i>Convert 3D Vector to Fick Angles</i>
----------------	---

Description

Converts a 3D vector to azimuth and elevation angles (in degrees).

Usage

```
vector_to_Fick(x, y, z)
```

Arguments

x	X component of the vector.
y	Y component of the vector.
z	Z component of the vector.

Value

A list with components azi and ele.

Examples

```
vector_to_Fick(0.5, 0.2, 0.8)
```

Index

accuracy, [2](#)

bcea, [3](#)

compute_data_quality_from_validation,
[3](#)

data_loss_from_expected, [9](#)

data_loss_from_invalid, [10](#)

DataQuality, [4](#)

effective_frequency, [10](#)

ETDQ_version, [11](#)

Fick_to_vector, [11](#)

precision_using_moving_window, [12](#)

report_data_quality_table, [13](#)

rms_s2s, [14](#)

ScreenConfiguration, [14](#)

std, [18](#)

vector_to_Fick, [19](#)