

Package: duet (via r-universe)

October 3, 2026

Type Package

Title An R Package for Dyadic Movement Analysis

Version 0.2.0

Description A set of tools for analyzing dyadic movement data. It provides functions to process, visualize, and compute various movement-based metrics from OpenPose-generated keypoints, including velocity, acceleration, and jerkiness.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL <https://github.com/ThemisEfth/duet>

BugReports <https://github.com/ThemisEfth/duet/issues>

Depends R (>= 4.1.0)

Imports biwavelet, imputeTS, kza, rjson, moments, dplyr, ggplot2, ggthemes, patchwork, reshape2, rlang, signal, stringr, tidyr, tidyselect, zoo

Suggests cowplot, flextable, magick, rempsyc, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/pak/sysreqs libfftw3-dev libicu-dev libjpeg-dev libpng-dev libxml2-dev libssl-dev

Repository <https://themiseft.r-universe.dev>

Date/Publication 2025-07-31 14:46:42 UTC

RemoteUrl <https://github.com/themiseft/duet>

RemoteRef HEAD

RemoteSha bc20d1e4c328801fab92fe8f936bd455b3d3ec7e

Contents

op_animate_dyad	2
op_apply_keypoint_labels	4
op_batch_create_csv	5
op_compute_acceleration	6
op_compute_coherence	7
op_compute_coherence_batch	9
op_compute_jerk	10
op_compute_motionenergy	11
op_compute_velocity	13
op_create_csv	14
op_interpolate	15
op_merge_dyad	16
op_plot_openpose	17
op_plot_quality	19
op_plot_timeseries	20
op_remove_keypoints	21
op_smooth_timeseries	23
op_summarise	24
Index	27

op_animate_dyad	<i>Animate OpenPose data for a dyad across a range of frames (Video)</i>
-----------------	--

Description

This function generates a video of the OpenPose data for both persons in a dyad across a specified range of frames.

Usage

```
op_animate_dyad(  
  data,  
  output_file,  
  lines = FALSE,  
  keylabels = FALSE,  
  label_type = "names",  
  fps = 24,  
  min_frame = NULL,  
  max_frame = NULL,  
  hide_labels = FALSE,  
  left_color = "blue",  
  right_color = "red",  
  background_color = "white",  
  background_colour = NULL  
)
```

Arguments

<code>data</code>	A dataframe containing OpenPose data.
<code>output_file</code>	A character string specifying the path and filename for the output video file.
<code>lines</code>	A logical value indicating whether to draw lines connecting joints. Default is FALSE.
<code>keylabels</code>	A logical value indicating whether to label keypoints. Default is FALSE.
<code>label_type</code>	A character string specifying the type of labels to use: "names" or "numbers". Default is "names".
<code>fps</code>	An integer specifying the frames per second for the video. Default is 24.
<code>min_frame</code>	An optional integer specifying the minimum frame to include in the video. Default is the first frame in the data.
<code>max_frame</code>	An optional integer specifying the maximum frame to include in the video. Default is the last frame in the data.
<code>hide_labels</code>	A logical value indicating whether to hide the x and y axes, box, and title. Default is FALSE.
<code>left_color</code>	A character string specifying the color to use for the left person. Default is "blue".
<code>right_color</code>	A character string specifying the color to use for the right person. Default is "red".
<code>background_color</code>	A character string specifying the background color of the plot. Default is "white". (US English)
<code>background_colour</code>	A character string specifying the background colour of the plot. Default is "white". (UK English)

Value

No return value. This function generates a video file as a side effect, saved at the specified output path.

Examples

```
## Not run:
# Example OpenPose data
data <- data.frame(
  frame = rep(1:10, each = 2),
  person = rep(c("left", "right"), times = 10),
  x0 = runif(20, 0, 1920), y0 = runif(20, 0, 1080),
  x1 = runif(20, 0, 1920), y1 = runif(20, 0, 1080)
)

# Output file path
output_file <- tempfile("output_video", fileext = ".mp4")

# Generate video
```

```
op_animate_dyad(  
  data = data,  
  output_file = output_file,  
  fps = 24,  
  left_color = "blue",  
  right_color = "red"  
)  
  
## End(Not run)
```

op_apply_keypoint_labels

Rename Columns Based on Region

Description

This function renames columns of a dataframe based on the specified region.

Usage

```
op_apply_keypoint_labels(df)
```

Arguments

df Dataframe with columns to be renamed.

Value

Dataframe with renamed columns.

Examples

```
# Example dataframe  
df <- data.frame(  
  region = rep(c("body", "hand_left", "hand_right", "face"), each = 3),  
  x0 = rnorm(12), y0 = rnorm(12), c0 = rnorm(12),  
  x1 = rnorm(12), y1 = rnorm(12), c1 = rnorm(12)  
)  
  
# Apply keypoint labels  
df_renamed <- op_apply_keypoint_labels(df)  
print(df_renamed)
```

op_batch_create_csv *Process all Dyad Directories to Create CSV Files*

Description

This function processes all dyad directories in the specified input base path, applying the ‘op_create_csv’ function from the package, and saves the output in the corresponding directories in the output base path.

Usage

```
op_batch_create_csv(
  input_base_path,
  output_base_path,
  include_filename = TRUE,
  include_labels = FALSE,
  frame_width = 1920,
  export_type = "dyad",
  model = "all",
  overwrite = FALSE
)
```

Arguments

input_base_path	Character. The base path containing dyad directories with JSON files.
output_base_path	Character. The base path where the CSV files will be saved.
include_filename	Logical. Whether to include filenames in the CSV. Default is TRUE.
include_labels	Logical. Whether to include labels in the CSV. Default is FALSE.
frame_width	Numeric. The width of the video frame in pixels. Default is 1920.
export_type	Character. The type of export file, such as ‘dyad’ or other formats. Default is ‘dyad’.
model	Character. The model object to use for processing, e.g., ‘all’ or a specific model. Default is ‘all’.
overwrite	Logical. Whether to overwrite existing files. Default is FALSE.

Value

None. The function is called for its side effects.

`op_compute_acceleration`*Compute Acceleration*

Description

This function calculates the acceleration for each column that begins with 'x' and 'y' and removes all columns that start with 'c'. It takes either the fps or the video duration as input to compute the acceleration.

Usage

```
op_compute_acceleration(  
  data,  
  fps = NULL,  
  video_duration = NULL,  
  overwrite = FALSE,  
  merge_xy = FALSE  
)
```

Arguments

<code>data</code>	A data frame containing the columns to process.
<code>fps</code>	Frames per second, used to compute acceleration.
<code>video_duration</code>	Video duration in seconds, used to compute fps.
<code>overwrite</code>	Logical value indicating whether to remove original 'x' and 'y' columns.
<code>merge_xy</code>	Logical value indicating whether to merge x and y columns using Euclidean distance.

Value

A data frame with acceleration columns added and 'c' columns removed.

Examples

```
# Load example data from the package  
data_path <- system.file("extdata/csv_data/A-B_body_dyad_velocity.csv", package = "duet")  
data <- read.csv(data_path)  
  
# Compute acceleration  
result <- op_compute_acceleration(  
  data = data,  
  fps = 30,  
  overwrite = FALSE,  
  merge_xy = TRUE  
)  
  
print(result)
```

op_compute_coherence *Compute Cross-Wavelet Coherence for Dyadic Motion Energy Data*

Description

This function computes cross-wavelet coherence between two individuals in a dyad using motion energy data. It is designed to be robust, CRAN-compliant, and user-friendly, with automatic detection of parameters and dynamic calculation of frequency bands.

Usage

```
op_compute_coherence(
  data,
  dyad_id = NULL,
  region = NULL,
  person_ids = NULL,
  dyad_col = NULL,
  region_col = "region",
  person_col = "person",
  frame_col = "frame",
  motion_col = "motion_energy",
  freq_bands = list(`0.03-0.06Hz` = c(0.03125, 0.0625), `0.06-0.12Hz` = c(0.0625, 0.125),
    `0.12-0.25Hz` = c(0.125, 0.25), `0.25-0.5Hz` = c(0.25, 0.5), `0.5-1Hz` = c(0.5, 1),
    `1-2Hz` = c(1, 2), `2-4Hz` = c(2, 4)),
  start_frame = 1,
  end_frame = NULL,
  param = 8,
  nrands = 1000,
  plot_result = FALSE,
  return_raw = FALSE,
  verbose = TRUE
)
```

Arguments

data	A data frame containing motion energy data.
dyad_id	Character string for the dyad to analyze. If 'NULL' (default), the function will proceed only if a single dyad is present in 'data'.
region	Character string for the body region to analyze. If 'NULL' (default), proceeds only if a single region exists for the selected dyad.
person_ids	A vector of two character strings for the persons in the dyad. If 'NULL' (default), auto-detects the two persons.
dyad_col	Character string for the dyad identifier column. Defaults to "base_filename" or "dyad_id" if found.
region_col	Character string for the region column name (default: "region").

person_col	Character string for the person column name (default: "person").
frame_col	Character string for the frame/time column name (default: "frame").
motion_col	Character string for the motion energy column name (default: "motion_energy").
freq_bands	A named list of frequency bands in Hertz (Hz) . Each element is a numeric vector of length two specifying the lower and upper frequency bound (e.g., <code>list("slow_rhythm" = c(0.1, 0.5))</code>).
start_frame	Integer, the starting frame for analysis (default: 1).
end_frame	Integer, the ending frame for analysis. If <code>'NULL'</code> (default), uses all available frames.
param	Numeric, the mother wavelet parameter for <code>'biwavelet::wtc'</code> (default: 8).
nrands	Integer, the number of random simulations for significance testing (default: 1000).
plot_result	Logical, if <code>'TRUE'</code> , generates a plot of the wavelet coherence.
return_raw	Logical, if <code>'TRUE'</code> , includes the raw <code>'wtc'</code> object in the output.
verbose	Logical, if <code>'TRUE'</code> , prints informative messages during execution.

Details

This function is a wrapper around `'biwavelet::wtc'` that simplifies its application to dyadic motion data. It includes CRAN-compliant safety checks, such as replacing `'cat()'` with `'message()'` and safely managing graphical parameters with `'on.exit()'`.

The key improvement is the dynamic calculation of frequency bands. You specify bands in Hz, and the function identifies the corresponding indices from the wavelet transform's scale/period results, making the analysis independent of time series length and sampling rate.

Value

A list containing:

coherence_summary	A data frame with <code>'dyad_id'</code> and coherence statistics for each frequency band.
analysis_info	A list with metadata about the analysis.
wtc_object	If <code>'return_raw = TRUE'</code> , the raw object from <code>'biwavelet::wtc'</code> .

Examples

```
## Not run:
# Create sample data
sample_data <- data.frame(
  frame = rep(1:100, 2),
  dyad_id = "D01",
  region = "body",
  person = rep(c("P1", "P2"), each = 100),
  motion_energy = c(rnorm(100), rnorm(100))
)

# Define frequency bands in Hz
my_bands <- list(
```

```

    "slow" = c(0.1, 0.5), # 0.1 to 0.5 Hz
    "fast" = c(0.5, 1.0) # 0.5 to 1.0 Hz
  )

  # Run analysis (dyad_id and region are auto-detected)
  result <- op_compute_coherence(
    data = sample_data,
    freq_bands = my_bands,
    plot_result = TRUE
  )

  print(result$coherence_summary)

  ## End(Not run)

```

op_compute_coherence_batch

Process Multiple Dyads for Cross-Wavelet Coherence

Description

A wrapper function that processes multiple dyads and/or regions in a dataset, attaching unique identifiers to each result.

Usage

```

op_compute_coherence_batch(
  data,
  process_by = c("dyad"),
  unique_id_cols = NULL,
  parallel = FALSE,
  ...
)

```

Arguments

data	A data frame containing motion energy data for multiple dyads/regions.
process_by	Character vector specifying what to process separately. Options: c("dyad"), c("region"), or c("dyad", "region"). Default is c("dyad").
unique_id_cols	Character vector of column names to include as unique identifiers in the output. If NULL, uses the dyad column and region column.
parallel	Logical, whether to use parallel processing (requires 'parallel' package).
...	Additional arguments passed to op_compute_coherence()

Value

A list with elements:

`results` A list of results, one per dyad/region combination
`summary_table` A data frame combining all coherence summaries with unique IDs
`processing_log` A data frame with processing status for each combination

<code>op_compute_jerk</code>	<i>Compute Jerk</i>
------------------------------	---------------------

Description

This function calculates the jerk for each column that begins with 'x' and 'y' and removes all columns that start with 'c'. It takes either the fps or the video duration as input to compute the jerk.

Usage

```
op_compute_jerk(  
  data,  
  fps = NULL,  
  video_duration = NULL,  
  overwrite = FALSE,  
  merge_xy = FALSE  
)
```

Arguments

`data` A data frame containing the columns to process.
`fps` Frames per second, used to compute jerk.
`video_duration` Video duration in seconds, used to compute fps.
`overwrite` Logical value indicating whether to remove original 'x' and 'y' columns.
`merge_xy` Logical value indicating whether to merge x and y columns using Euclidean distance.

Value

A data frame with jerk columns added and 'c' columns removed.

Examples

```
# Load example data from the package  
data_path <- system.file("extdata/csv_data/A-B_body_dyad_accel.csv", package = "duet")  
data <- read.csv(data_path)  
  
# Compute jerk  
result <- op_compute_jerk(
```

```

    data = data,
    fps = 30,
    overwrite = FALSE,
    merge_xy = TRUE
  )

  print(result)

```

op_compute_motionenergy

Compute Motion Energy from OpenPose Data

Description

Performs frame differencing analysis on OpenPose keypoint data to calculate motion energy. This function computes the amount of movement between consecutive frames for each keypoint, with options for aggregation and filtering.

Usage

```

op_compute_motionenergy(
  data,
  id_cols = NULL,
  frame_col = "frame",
  aggregate_keypoints = TRUE,
  aggregate_coordinates = TRUE,
  method = c("absolute", "squared"),
  na_action = c("omit", "interpolate", "zero"),
  plot = FALSE,
  rmea_format = FALSE
)

```

Arguments

data	A data.frame containing OpenPose data with columns for keypoint coordinates (x0, y0, x1, y1, etc.) and grouping variables.
id_cols	Character vector of column names used for grouping. If NULL (default), automatically detects ID columns as all non-coordinate, non-frame columns (excludes columns starting with x, y, c and frame column).
frame_col	Character string specifying the frame column name. Default: "frame"
aggregate_keypoints	Logical. If TRUE, aggregates motion across all keypoints. If FALSE, returns motion energy per keypoint. Default: TRUE
aggregate_coordinates	Logical. If TRUE, combines x and y motion into single metric using Euclidean distance. If FALSE, keeps separate. Default: TRUE

method	Character. Method for calculating differences: • "absolute": Calculates the sum of absolute differences in coordinates between frames. This method provides a linear measure of change and is sensitive to smaller movements. The resulting values are directly interpretable as the magnitude of change. • "squared": Calculates the sum of squared differences in coordinates between frames. This method amplifies larger movements more significantly than smaller ones, making it potentially more sensitive to bursts of activity or more pronounced changes. It's often used when the impact of larger movements needs to be emphasized. Default: "absolute"
na_action	Character. How to handle missing values: "omit" removes frames with missing data, "interpolate" uses linear interpolation, "zero" treats missing as zero motion. Default: "omit"
plot	Logical. If TRUE, generates a plot of motion energy over frames when data is fully aggregated (aggregate_keypoints = TRUE and aggregate_coordinates = TRUE). The plot will be grouped by the 'person' column if it's one of the id_cols, otherwise by the first id_col. Default: FALSE
rmea_format	Logical. If TRUE, converts output to wide format with columns for region*person combinations, removing all other columns. Default: FALSE

Details

The function processes OpenPose data by: 1. Auto-detecting ID columns (if not specified) as columns that don't start with x, y, c 2. Grouping data by the ID columns 3. Computing frame-to-frame differences based on the chosen method 4. Aggregating results based on user preferences

Motion energy is calculated as the absolute or squared difference between consecutive frames. When aggregating coordinates, Euclidean distance is used: $\sqrt{x_diff^2 + y_diff^2}$ if method is "absolute" (applied after diff), or $x_diff^2 + y_diff^2$ if method is "squared" (as diffs are already squared). Note: The Euclidean combination for "squared" method is implicitly handled as $\sqrt{(x_diff^2)^2 + (y_diff^2)^2}$ if 'aggregate_coordinates' is TRUE *after* squaring, or more commonly, the squared differences are summed directly if that's the intent. The current implementation applies $\sqrt{x_motion^2 + y_motion^2}$ where x_motion/y_motion are either 'abs(diff)' or 'diff^2'. For "squared" method, this means $\sqrt{(x_diff^2)^2 + (y_diff^2)^2}$. If the intent for "squared" is $\text{sum}(x_diff^2 + y_diff^2)$ before sqrt, the logic in aggregation might need adjustment based on precise definition. Assuming current implementation is desired.

When aggregating keypoints, values are summed across all valid keypoints for each frame.

Value

A data.frame with motion energy values. Structure depends on aggregation parameters:

- If both aggregation options TRUE: ID columns + frame + motion_energy
- If aggregate_coordinates FALSE: adds x_motion, y_motion columns
- If aggregate_keypoints FALSE: adds keypoint column
- If rmea_format TRUE: wide format with region*person columns only

If 'plot = TRUE' and conditions are met, a ggplot object is also printed.

Note

The first frame of each group will have NA motion values since there's no previous frame for comparison. These are removed when `na_action = "omit"`. Requires `ggplot2` package for plotting.

`op_compute_velocity` *Compute Velocity and Optionally Adjust Boundary Values*

Description

This function calculates the velocity for each column that begins with 'x' and 'y'. The first row of the data frame (containing initial NA velocities) is removed. Optionally, the first and last calculated velocity values in the returned series can be set to NA if they are suspected to be artifacts. The function also removes all columns that start with 'c'.

Usage

```
op_compute_velocity(
  data,
  fps = NULL,
  video_duration = NULL,
  overwrite = FALSE,
  merge_xy = FALSE,
  boundary_velocity_treatment = "none"
)
```

Arguments

<code>data</code>	A data frame containing the columns to process. Must have at least 2 rows if velocity calculation is expected.
<code>fps</code>	Frames per second, used to compute velocity.
<code>video_duration</code>	Video duration in seconds, used to compute fps.
<code>overwrite</code>	Logical value indicating whether to remove original 'x' and 'y' columns after velocity calculation. Default is FALSE.
<code>merge_xy</code>	Logical value indicating whether to merge x and y columns using Euclidean distance for velocity. If FALSE, velocity is computed for x and y components separately. Default is FALSE.
<code>boundary_velocity_treatment</code>	Character string specifying how to treat the first and last calculated velocity values in the output series. Options: "none" (default - no change), "set_na" (sets the first and last calculated velocities to NA).

Value

A data frame with velocity columns. If `'boundary_velocity_treatment = "set_na"'`, the first and last rows of the velocity columns will have NA values. The overall data frame will have one less row than the input due to the removal of the initial NA velocity row.

Examples

```
# Create sample data
sample_data <- data.frame(
  frame = 1:10,
  x0 = cumsum(rnorm(10)), y0 = cumsum(rnorm(10)), c0 = rnorm(10),
  x1 = cumsum(rnorm(10)), y1 = cumsum(rnorm(10)), c1 = rnorm(10)
)

# Compute velocity, default boundary treatment
result_default <- op_compute_velocity(
  data = sample_data,
  fps = 30
)
print(result_default) # Should have 9 rows

# Compute velocity, set boundary velocities to NA
result_boundary_na <- op_compute_velocity(
  data = sample_data,
  fps = 30,
  boundary_velocity_treatment = "set_na"
)
print(result_boundary_na) # First and last velocity values should be NA
```

op_create_csv

Create CSV from JSON files

Description

This function reads JSON files from the specified directory, processes the pose keypoints, and saves the results into CSV files.

Usage

```
op_create_csv(
  input_path,
  output_path = input_path,
  model = "all",
  include_filename = TRUE,
  include_labels = FALSE,
  frame_width = 1920,
  export_type = "dyad",
  use_openpose_order = FALSE
)
```

Arguments

input_path	Path to the directory containing JSON files.
output_path	Path to the directory where CSV files will be saved. Defaults to the input path.

model	The model to use: "all", "body", "hands", or "face". Defaults to "all".
include_filename	Boolean indicating whether to include the base filename in column names. Defaults to FALSE.
include_labels	Boolean indicating whether to rename columns based on region labels. Defaults to FALSE.
frame_width	Width of the frame. Defaults to 1920.
export_type	Type of export: "individual" to export separate CSV files for each person, "dyad" to export both persons' data into a single CSV file. Defaults to "individual".
use_openpose_order	Logical. If TRUE, assigns person 1 and 2 based on the order given by OpenPose rather than position on screen (left/right). Defaults to FALSE.

Value

No return value. This function is called for its side effects, which include writing CSV files to the specified output directory.

op_interpolate	<i>Interpolate missing or low-quality values in OpenPose time-series data</i>
----------------	---

Description

This function performs various interpolation methods for x and y coordinate columns in OpenPose datasets based on confidence thresholds, missing values, or zero values. It groups the data by specified grouping variables and uses the selected interpolation method to estimate problematic values.

The function is designed to be robust, automatically detecting the relevant OpenPose columns (e.g., 'x1, y1, c1') and applying interpolation logic to each keypoint within each specified group (e.g., for each person).

Usage

```
op_interpolate(
  data,
  method = "median",
  confidence_threshold = 0.3,
  handle_missing = TRUE,
  handle_zeros = FALSE,
  treat_na_conf_as_low = TRUE,
  grouping_vars = c("person", "region"),
  polynomial_degree = 3,
  max_gap = Inf,
  smooth_factor = 0,
  extrapolation = "none",
  verbose = FALSE
)
```

Arguments

data	A data frame containing OpenPose keypoint data with x, y, and confidence columns.
method	Character string specifying the interpolation method. Options include: "spline", "linear", "polynomial", "kalman", "locf" (last observation carried forward), "nocb" (next observation carried backward), "mean", "median". Default is "median".
confidence_threshold	Numeric, NA, or FALSE. The confidence score below which data points are considered problematic and targeted for interpolation. If 'NA' or 'FALSE', this check is skipped. Default is 0.3.
handle_missing	Logical. If 'TRUE' (default), 'NA' values in coordinate columns will be targeted for interpolation.
handle_zeros	Logical. If 'TRUE', coordinate values of exactly 0 will be targeted for interpolation. Default is 'FALSE'.
treat_na_conf_as_low	Logical. If 'TRUE' (default), 'NA' values in a confidence column are treated as having low confidence (i.e., 0).
grouping_vars	Character vector of column names to group the data by before interpolation (e.g., c("person", "region")). Interpolation is performed independently for each group.
polynomial_degree	Integer. The degree of the polynomial to use when 'method = "polynomial"'. Default is 3.
max_gap	Integer. The maximum number of consecutive problematic frames to interpolate. Gaps larger than this value will be ignored. Default is 'Inf' (no limit).
smooth_factor	Numeric. A smoothing factor for spline interpolation (currently unused, for future compatibility). Default is 0.
extrapolation	Character string specifying how to handle values outside the range of good data. Not yet fully implemented. Default is "none".
verbose	Logical. If 'TRUE', prints detailed messages about the process.

Value

A data frame identical in structure to the input 'data', but with problematic values replaced by interpolated estimates. Two new columns are added: 'interpolated_points_count_per_row' and 'interpolation_method_used'.

op_merge_dyad

Merge CSV files for each dyad

Description

This function merges all CSV files in each dyad directory within the specified input base path.

Usage

```
op_merge_dyad(input_base_path, output_base_path)
```

Arguments

`input_base_path`
Character. The base path containing dyad directories with CSV files.

`output_base_path`
Character. The base path where the merged CSV files will be saved.

Value

None. The function is called for its side effects.

Examples

```
# Load example data paths from the package
input_base_path <- system.file("extdata/csv_data/dyad_1", package = "duet")
output_base_path <- tempfile("merged_dyads")

# Ensure input files exist
input_files <- list.files(input_base_path, pattern = "\\*.csv$", full.names = TRUE)
if (length(input_files) > 0) {
  # Merge CSV files for each dyad
  op_merge_dyad(input_base_path, output_base_path)

  # Check merged files
  merged_files <- list.files(output_base_path, pattern = "\\*.csv$", full.names = TRUE)
  print(merged_files)

  # Read and display merged data
  if (length(merged_files) > 0) {
    merged_data <- read.csv(merged_files[1])
    print(merged_data)
  } else {
    message("No merged files were created.")
  }
} else {
  message("No input files found to process.")
}
```

op_plot_openpose

Plot OpenPose Data for a Specified Frame

Description

This function visualizes keypoints and their connections from OpenPose data for a specified frame. The function allows customization of the plot, including the option to display labels, lines between keypoints, and different colours for left and right persons.

Usage

```

op_plot_openpose(
  data,
  frame_num,
  person = c("both", "left", "right"),
  lines = TRUE,
  keylabels = FALSE,
  label_type = c("names", "numbers"),
  hide_labels = FALSE,
  left_color = "blue",
  right_color = "red",
  background_color = "white",
  background_colour = NULL,
  line_width = 2,
  point_size = 1.5,
  text_color = "black"
)

```

Arguments

<code>data</code>	A data frame containing OpenPose data. The data frame should include columns for the frame number, person identifier, and x/y coordinates for each keypoint.
<code>frame_num</code>	A numeric value specifying the frame number to plot.
<code>person</code>	A character string specifying which person to plot: "left", "right", or "both". Default is "both".
<code>lines</code>	A logical value indicating whether to draw lines between keypoints. Default is TRUE.
<code>keylabels</code>	A logical value indicating whether to display keypoint labels. Default is FALSE.
<code>label_type</code>	A character string specifying the type of labels to display: "names" or "numbers". Default is "names".
<code>hide_labels</code>	A logical value indicating whether to hide axis labels and plot titles. Default is FALSE.
<code>left_color</code>	A character string specifying the color for the left person. Default is "blue".
<code>right_color</code>	A character string specifying the color for the right person. Default is "red".
<code>background_color</code>	A character string specifying the background color of the plot. Default is "white".
<code>background_colour</code>	A character string specifying the background colour of the plot (UK spelling). Default is NULL.
<code>line_width</code>	A numeric value specifying the width of the lines between keypoints. Default is 2.
<code>point_size</code>	A numeric value specifying the size of the keypoint markers. Default is 1.5.
<code>text_color</code>	A character string specifying the color of the text (labels and titles). Default is "black".

Value

No return value, called for side effects (plotting to screen).

Examples

```
# Path to example CSV file included with the package
file_path <- system.file("extdata/csv_data/A-B_body_dyad.csv", package = "duet")

# Load the data
data <- read.csv(file_path)

# Plot the data for the specified frame
op_plot_openpose(
  data = data,
  frame_num = 1,
  person = "both",
  lines = TRUE,
  keylabels = TRUE,
  label_type = "names",
  left_color = "blue",
  right_color = "red",
  background_colour = "grey90"
)
```

op_plot_quality	<i>Plot Data Quality (Confidence Ratings or Completeness)</i>
-----------------	---

Description

This function plots either the mean confidence ratings, the percentage of completeness (i.e., data present), or both for the given dataframe. It can handle data for one or multiple persons and regions, creating separate panels for each.

Usage

```
op_plot_quality(df, plot_type = "confidence", threshold_line = 50)
```

Arguments

df	A dataframe containing the confidence data, with columns for base_filename, region, person, and confidence values.
plot_type	Character. Either "confidence" to plot the mean confidence rating, "completeness" to plot the percentage of completeness, or "both" to plot both. Default is "confidence".
threshold_line	Numeric. The value at which to draw a dashed horizontal line. Default is 50.

Value

A ggplot object or a combined plot if "both" is selected.

Examples

```
# Example usage:
# Path to example CSV file included with the package
file_path <- system.file("extdata/csv_data/A-B_body_dyad.csv", package = "duet")

# Load the data
data <- read.csv(file_path)

# plot <- op_plot_data_quality(df, plot_type = "both", threshold_line = 75)
# print(plot)
```

op_plot_timeseries *Plot Keypoints with Facet Wrap*

Description

This function plots specified keypoints (or defaults) over time with facet wrapping. It handles columns starting with x, y, v_, a_, or j_. Color and linetype are used to distinguish overlaid persons or overlaid metric types.

Usage

```
op_plot_timeseries(
  data,
  keypoints = NULL,
  free_y = TRUE,
  overlay_axes = FALSE,
  person = "both",
  facet_by_person = TRUE,
  max_facets = 10,
  x_axis = "frame",
  verbose = FALSE
)
```

Arguments

data	Data frame containing the keypoint data. Must include a 'frame' column for the x-axis.
keypoints	Character vector of keypoint identifiers to plot (e.g., "0", "1", "Nose"). If NULL (default), the first four available keypoint identifiers are used.
free_y	Boolean indicating if the y-axis should be free in facet_wrap (default is TRUE).
overlay_axes	Boolean indicating if different metric types (x, y, v_, a_, j_) for the same keypoint ID should be overlaid in the same plot facet. Default is FALSE.

person	Character string specifying which person to plot. Options: "left", "right", or "both". Requires a 'person' column in 'data'. Default is "both".
facet_by_person	Boolean indicating if data for different persons (when 'person = "both"') should be in separate facets ('TRUE', default) or overlaid on the same facets ('FALSE').
max_facets	Integer indicating the maximum number of facets allowed (default is 10). If the total facets exceed this number, the function returns 'NULL' with a warning.
x_axis	Character string for the column to be used as the x-axis (time). Default is "frame".
verbose	Logical, if TRUE, prints messages about default keypoint selection. Default is FALSE.

Value

A ggplot object or NULL if the maximum number of facets is exceeded or no data can be plotted.

Examples

```
# Create sample data
sample_data <- data.frame(
  frame = 1:100,
  x0 = rnorm(100), y0 = rnorm(100), v_0 = rnorm(100, 5),
  x1 = rnorm(100, 2), y1 = rnorm(100, 2), a_1 = rnorm(100, 10),
  x2 = rnorm(100, -2), y2 = rnorm(100, -2), j_2 = rnorm(100, 1),
  x3 = rnorm(100, 1), y3 = rnorm(100, 1),
  person = rep(c("P1", "P2"), each = 50)
)

## Not run:
# Ex 1: Overlay axes, facet by person.
# Color by metric_type, linetype by metric_type.
op_plot_timeseries(data = sample_data, overlay_axes = TRUE, person = "both",
  facet_by_person = TRUE)

# Ex 2: Overlay axes, overlay persons.
# Color by person, linetype by person.
op_plot_timeseries(data = sample_data, overlay_axes = TRUE, person = "both",
  facet_by_person = FALSE)

## End(Not run)
```

op_remove_keypoints *Remove Keypoints Based on Various Criteria*

Description

This function removes keypoints and their corresponding columns based on several criteria: user-specified keypoints, low total confidence values over time, exceeding a threshold of missing/zero values, or if all data for a keypoint is missing (i.e., all zeros).

Usage

```
op_remove_keypoints(
  df,
  remove_specific_keypoints = NULL,
  remove_undetected_keypoints = FALSE,
  remove_keypoints_total_confidence = NULL,
  remove_keypoints_missing_data = NULL,
  apply_removal_equally = TRUE
)
```

Arguments

df A data frame containing the data to process. Keypoint columns are expected to include x, y, and c (confidence) columns with corresponding indices.

remove_specific_keypoints Character vector. Specifies the keypoint indices (e.g., "1") to remove. This will automatically remove corresponding 'x', 'y', and 'c' columns for those indices. Default is NULL.

remove_undetected_keypoints Logical. If TRUE, removes keypoints where all confidence values are zero across all rows. Default is FALSE.

remove_keypoints_total_confidence Numeric or FALSE. A threshold for the mean confidence values. Keypoints with a mean confidence below this threshold will be removed. If set to FALSE, behaves as NULL. Default is NULL.

remove_keypoints_missing_data Numeric or FALSE. A threshold (between 0 and 1) for the percentage of missing or zero values. Columns exceeding this threshold will be removed. If set to FALSE, behaves as NULL. Default is NULL.

apply_removal_equally Logical. If TRUE, the same columns will be removed across all rows of the dataset. If FALSE, removal criteria are applied separately for each combination of 'person' and 'region'. Default is TRUE.

Value

A data frame with specified keypoints and corresponding columns removed.

Examples

```
# Load example data from the package
data_path <- system.file("extdata/csv_data/dyad_1/A_body.csv", package = "duet")
df <- read.csv(data_path)

# Remove keypoints based on various criteria
result <- op_remove_keypoints(
  df = df,
  remove_specific_keypoints = c("1", "2"), # Remove specific keypoints (e.g., keypoints 1 and 2)
  remove_undetected_keypoints = TRUE,      # Remove keypoints with all zero confidence
```

```

remove_keypoints_total_confidence = 0.5, # Remove keypoints with mean confidence below 0.5
remove_keypoints_missing_data = 0.2,    # Remove keypoints with >20% missing data
apply_removal_equally = TRUE            # Apply removal equally across the dataset
)

# Display the result
print(result)

```

op_smooth_timeseries *Smooth Time Series Data with Various Methods*

Description

This function applies different smoothing techniques to time series data for the selected columns (keypoints), including moving average, Kalman-Ziegler Adaptive (KZA), Savitzky-Golay filter, and Butterworth filter. It can optionally plot the smoothed data alongside the original data, with faceting based on the 'person' and 'keypoint' columns.

Arguments

data	A data frame containing the time series data. Must include 'person', 'time', and keypoints (e.g., 'x0', 'y0', etc.).
method	The smoothing method to use. Options are "zoo" (moving average), "kza" (Kalman-Ziegler Adaptive), "savitzky" (Savitzky-Golay filter), and "butterworth" (Butterworth filter). Default is "zoo".
kza_k	Window size for the KZA method. Default is 3.
kza_m	Number of iterations for the KZA method. Default is 2.
rollmean_width	Width of the moving average window for the zoo method. Default is 3.
sg_window	Window size for the Savitzky-Golay filter. Default is 5.
sg_order	Polynomial order for the Savitzky-Golay filter. Default is 3.
butter_order	Order of the Butterworth filter. Default is 3.
butter_cutoff	Cutoff frequency for the Butterworth filter. Default is 0.1.
side	Character string indicating which side of the data to smooth. Options are "left", "right", or "both". Default is "both".
plot	Logical, if TRUE, the function will generate a plot comparing the original and smoothed data. If FALSE, the function returns only the smoothed data frame without plotting. Default is TRUE.
keypoints	Vector of keypoint column names (e.g., 'x0', 'x1') to be smoothed and included in the plot. If NULL, all keypoints beginning with 'x' or 'y' will be smoothed and plotted. Default is NULL.

Value

A data frame with the smoothed time series data for the specified keypoints. If 'plot = TRUE', a plot is displayed comparing the original and smoothed data.

Examples

```
# Load example data from the package
data_path <- system.file("extdata/csv_data/dyad_1/A_body.csv", package = "duet")
data <- read.csv(data_path)

# Smooth the time series data using the Savitzky-Golay filter
smoothed_data <- op_smooth_timeseries(
  data = data,
  method = "savitzky",
  sg_window = 5,
  sg_order = 3,
  plot = TRUE,
  keypoints = c("x0", "y0") # Specify keypoints to smooth
)

# Print the smoothed data
print(smoothed_data)
```

grouping_vars	A character vector of column names in data that identify unique groups for summarisation (e.g., <code>c("person", "region")</code>). If NULL (default), the function will attempt to use "person" and/or "region" if they exist in data and have more than one unique level. Otherwise, no grouping is applied beyond the pivoted names_to column.
cols	A character vector of column names to pivot from wide to long format. If NULL (default), all numeric columns not in grouping_vars (once determined) are pivoted.
names_to	A character string specifying the name of the new column storing the names of pivoted columns. Default is "keypoint".
values_to	A character string specifying the name of the new column storing the numeric values from pivoted columns. Default is "value".
metrics	A character vector specifying which metrics to calculate. Available options: "count", "na_count", "valid_count", "mean", "median", "sd", "variance", "iqr", "min", "max", "skewness", "kurtosis", "dominant_period". Default calculates all metrics.
plot	Logical indicating whether to generate summary plots of the calculated statistics. Default is FALSE.
dominant_period_min_points	Integer specifying minimum number of non-NA, non-constant data points required for dominant period calculation. Default is 10L.
dominant_period_args	List of additional arguments passed to spectrum for periodicity calculation. Default is NULL.

Details

The function performs the following steps:

1. Validates input parameters.
2. Determines grouping variables if not explicitly provided (checks for "person", "region").
3. Identifies numeric columns to pivot, excluding non-numeric columns with a warning.
4. Reshapes data from wide to long format using [pivot_longer](#).
5. Calculates requested summary statistics grouped by specified variables.
6. Optionally generates visualization plots of the summary statistics.

For dominant period calculation, the function uses [spectrum](#) to find the peak in the power spectrum density. Skewness and kurtosis require the **moments** package.

Value

A tibble with summary statistics. Each row corresponds to a unique combination of determined grouping_vars and values from names_to. Columns include grouping variables and requested metrics.

Examples

```
# Create sample data with a non-numeric column
sample_data <- data.frame(
  frame = 1:100,
  participant = rep(c("P1", "P2"), each = 50),
  region = rep(c("A", "B"), times = 50),
  notes = "some_metadata", # This non-numeric column will be ignored
  Nose_x = rnorm(100),
  Nose_y = rnorm(100),
  LEye_x = rnorm(100),
  LEye_y = rnorm(100)
)

# The function will now automatically ignore the 'notes' column and warn the user.
result_robust <- op_summarise(
  data = sample_data,
  grouping_vars = c("participant", "region"),
  metrics = c("mean", "sd"),
  plot = TRUE
)
print(result_robust)
```

Index

op_animate_dyad, [2](#)
op_apply_keypoint_labels, [4](#)
op_batch_create_csv, [5](#)
op_compute_acceleration, [6](#)
op_compute_coherence, [7](#)
op_compute_coherence_batch, [9](#)
op_compute_jerk, [10](#)
op_compute_motionenergy, [11](#)
op_compute_velocity, [13](#)
op_create_csv, [14](#)
op_interpolate, [15](#)
op_merge_dyad, [16](#)
op_plot_openpose, [17](#)
op_plot_quality, [19](#)
op_plot_timeseries, [20](#)
op_remove_keypoints, [21](#)
op_smooth_timeseries, [23](#)
op_summarise, [24](#)

pivot_longer, [25](#)

spectrum, [25](#)