Package: duet (via r-universe)

February 25, 2025

Title Analysing Non-Verbal Communication in Dyadic Interactions from Video Data

Version 0.1.1

Description Analyzes non-verbal communication by processing data extracted from video recordings of dyadic interactions. It supports integration with open source tools, currently limited to 'OpenPose' (Cao et al. (2019) <doi:10.1109/TPAMI.2019.2929257>), converting its outputs into CSV format for further analysis. The package includes functions for data pre-processing, visualization, and computation of motion indices such as velocity, acceleration, and jerkiness (Cook et al. (2013) <doi:10.1093/brain/awt208>), facilitating the analysis of non-verbal cues in paired interactions and contributing to research on human communication dynamics.

License MIT + file LICENSE

SystemRequirements FFmpeg (https://ffmpeg.org/)

Imports dplyr, ggplot2, ggthemes, graphics, grDevices, kza, parallel, patchwork, reshape2, rjson, rlang, signal, stats, stringr, tidyr, tidyselect, tools, utils, zoo

Suggests spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs libfftw3-dev libicu-dev

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

RemoteUrl https://github.com/themisefth/duet

RemoteRef HEAD

RemoteSha 820896948895aee4ed6f3230c2656e9b8c9bfd38

Contents

`op_animate_dyad` *Animate OpenPose data for a dyad across a range of frames (Video)*

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

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

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

data	A data frame containing the columns to process.
fps	Frames per second, used to compute acceleration.
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 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_jerk *Compute Jerk*

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_velocity` *Compute Velocity*

Description

This function calculates the velocity 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 velocity.

Usage

```
op_compute_velocity(
  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 velocity.
`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 velocity columns added and 'c' columns removed.

Examples

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

# Compute velocity
result <- op_compute_velocity(
  data = data,
  fps = 30,
  overwrite = FALSE,
  merge_xy = TRUE
)

print(result)
```

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 = FALSE,
  include_labels = FALSE,
  frame_width = 1920,
  export_type = "dyad"
)
```

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".

Value

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

Examples

```
# Path to example JSON files included with the package
input_path <- system.file("extdata/json_files", package = "duet")

# Temporary output directory
output_path <- tempfile("output_csv")
```

```
dir.create(output_path)

# Run the function using the provided data
op_create_csv(
  input_path = input_path,
  output_path = output_path,
  model = "body",
  include_filename = TRUE,
  include_labels = TRUE,
  frame_width = 1920,
  export_type = "dyad"
)
```

`op_interpolate` *Interpolate missing or low-confidence values in a dataset*

Description

This function performs interpolation for x and y coordinate columns in a dataset based on confidence thresholds. It groups the data by `person` and `region` and uses spline interpolation to estimate missing or low-confidence values.

Usage

```
op_interpolate(
  data,
  confidence_threshold,
  missing = TRUE,
  treat_na_conf_as_low = FALSE
)
```

Arguments

`data`	A data frame containing x, y, confidence columns, and grouping columns (`person`, `region`).
`confidence_threshold`	A numeric value specifying the confidence threshold below which values will be interpolated.
`missing`	Logical. If TRUE, interpolate missing values (`NA`) in addition to low-confidence values.
`treat_na_conf_as_low`	Logical. If TRUE, treat NA in the confidence column as low confidence.

Value

A modified data frame with interpolated x and y values for low-confidence or missing rows.

Examples

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

# Interpolate missing or low-confidence values
result <- op_interpolate(
  data = data,
  confidence_threshold = 0.5,
  missing = TRUE,
  treat_na_conf_as_low = TRUE
)

print(result)
```

`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

`data`	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.
`frame_num`	A numeric value specifying the frame number to plot.

person	A character string specifying which person to plot: "left", "right", or "both". Default is "both".
lines	A logical value indicating whether to draw lines between keypoints. Default is TRUE.
keylabels	A logical value indicating whether to display keypoint labels. Default is FALSE.
label_type	A character string specifying the type of labels to display: "names" or "numbers". Default is "names".
hide_labels	A logical value indicating whether to hide axis labels and plot titles. Default is FALSE.
left_color	A character string specifying the color for the left person. Default is "blue".
right_color	A character string specifying the color for the right person. Default is "red".
background_color	A character string specifying the background color of the plot. Default is "white".
background_colour	A character string specifying the background colour of the plot (UK spelling). Default is NULL.
line_width	A numeric value specifying the width of the lines between keypoints. Default is 2.
point_size	A numeric value specifying the size of the keypoint markers. Default is 1.5.
text_color	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 *Plot Data Quality (Confidence Ratings or Completeness)*

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 the specified keypoints over time with facet wrapping. It allows the user to overlay axes or separate them, and to filter the data by person. If the number of facets exceeds the `max_facets` limit, a warning is issued, and the function will not return a plot.

Usage

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

Arguments

`data`	Data frame containing the keypoint data.
`keypoints`	Character vector of keypoints to include in the plot. When `overlay_axes = TRUE`, input keypoints as "0", "1", "2", etc. When `overlay_axes = FALSE`, input keypoints as "x0", "y0", "x1", "y1", etc.
`free_y`	Boolean indicating if the y-axis should be free in facet_wrap (default is TRUE).
`overlay_axes`	Boolean indicating if 'x' and 'y' columns should be overlaid in the same plot (default is FALSE).
`person`	Character string specifying which person to plot ("left", "right", or "both", default is "both").
`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.

Value

A ggplot object or NULL if the maximum number of facets is exceeded.

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)

# Plot with overlay_axes = TRUE
op_plot_timeseries(data = data, keypoints = c("0", "1"), overlay_axes = TRUE, person = "both")
```

```
# Plot with overlay_axes = FALSE
op_plot_timeseries(data = data, keypoints = c("0", "1"), overlay_axes = FALSE, person = "left")
```

`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)
```

Index