Package 'whippr'

October 12, 2022

Title Tools for Manipulating Gas Exchange Data

Version 0.1.2

Description Set of tools for manipulating gas exchange data from cardiopulmonary exercise testing.

License MIT + file LICENSE

URL `https://github.com/fmmattioni/whippr`

BugReports `https://github.com/fmmattioni/whippr/issues`

Encoding UTF-8

Imports readxl (>= 1.3.1), dplyr (>= 1.0.1), stringr (>= 1.4.0), lubridate (>= 1.7.9), magrittr, tibble, zoo, purrr, tidyr (>= 1.1.1), broom (>= 0.7.0), cli, ggplot2 (>= 3.3.2), glue, minpack.lm, patchwork (>= 1.0.1), rlang, usethis, nlstools, pillar

RoxygenNote 7.2.1

Suggests knitr, rmarkdown, fansi, collapsibleTree, testthat, shiny, miniUI, datapasta, rstudioapi, htmltools, readr, anomalize, ggforce, ggtext, forcats, covr

NeedsCompilation no

Author Felipe Mattioni Maturana [aut, cre] (<`https://orcid.org/0000-0002-4221-6104`>)

Maintainer Felipe Mattioni Maturana <`felipe.mattioni@med.uni-tuebingen.de`>

Repository CRAN

Date/Publication 2022-09-09 07:00:02 UTC

R topics documented:

`detect_outliers` *Detect outliers*

Description

It detects outliers based on prediction bands for the given level of confidence provided.

Usage

```
detect_outliers(
  .data,
  test_type = c("incremental", "kinetics"),
  vo2_column = "VO2",
  cleaning_level = 0.95,
  cleaning_baseline_fit,
  protocol_n_transitions,
  protocol_baseline_length,
  protocol_transition_length,
  method_incremental = c("linear", "anomaly"),
  verbose = TRUE,
  ...
)
```

Arguments

`.data`	Data retrieved from `read_data()` for a **kinetics** test, or the data retrieved from `incremental_normalize()` for a **incremental** test.
`test_type`	The test to be analyzed. Either 'incremental' or 'kinetics'.
`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to `VO2`.
`cleaning_level`	A numeric scalar between 0 and 1 giving the confidence level for the intervals to be calculated. Default to `0.95`.
`cleaning_baseline_fit`	For **kinetics** test only. A vector of the same length as the number in `protocol_n_transitions`, indicating what kind of fit to perform for each baseline. Vector accepts characters either 'linear' or 'exponential'.
`protocol_n_transitions`	For **kinetics** test only. Number of transitions performed.
`protocol_baseline_length`	For **kinetics** test only. The length of the baseline (in seconds).
`protocol_transition_length`	For **kinetics** test only. The length of the transition (in seconds).
`method_incremental`	The method to be used in detecting outliers from the incremental test. Either 'linear' or 'anomaly'. See `Details`.
`verbose`	A boolean indicating whether messages should be printed in the console. Default to `TRUE`.
`...`	Additional arguments. Currently ignored.

Details

TODO

Value

a tibble

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("example_cosmed.xlsx", package = "whippr")

## read data
df <- read_data(path = path_example, metabolic_cart = "cosmed")

## detect outliers
data_outliers <- detect_outliers(
  .data = df,
  test_type = "kinetics",
  vo2_column = "VO2",
```

```
  cleaning_level = 0.95,
  cleaning_baseline_fit = c("linear", "exponential", "exponential"),
  protocol_n_transitions = 3,
  protocol_baseline_length = 360,
  protocol_transition_length = 360,
  verbose = TRUE
 )

## get file path from example data
path_example_ramp <- system.file("ramp_cosmed.xlsx", package = "whippr")

## read data from ramp test
df_ramp <- read_data(path = path_example_ramp, metabolic_cart = "cosmed")

## normalize incremental test data
ramp_normalized <- df_ramp %>%
 incremental_normalize(
   .data = .,
   incremental_type = "ramp",
   has_baseline = TRUE,
   baseline_length = 240,
   work_rate_magic = TRUE,
   baseline_intensity = 20,
   ramp_increase = 25
 )

## detect ramp outliers
data_ramp_outliers <- detect_outliers(
  .data = ramp_normalized,
  test_type = "incremental",
  vo2_column = "VO2",
  cleaning_level = 0.95,
  method_incremental = "linear",
  verbose = TRUE
 )

## End(Not run)
```

`get_residuals` *Get residuals*

Description

Computes residuals from the VO2 kinetics model.

Usage

```
get_residuals(.model)
```

Arguments

`.model`	A model of class `nls`.

Value

a tibble containing the data passed to augment, and additional columns:

`.fitted`	The predicted response for that observation.
`.resid`	The residual for a particular point.
`standardized_residuals`	Standardized residuals.
`sqrt_abs_standardized_residuals`	The sqrt of absolute value of standardized residuals.
`lag_residuals`	The lag of the `.resid` column for plotting auto-correlation.

`incremental_normalize` *Normalize incremental test data*

Description

Detect protocol phases (baseline, ramp, steps), normalize work rate, and time-align baseline phase (baseline time becomes negative).

Usage

```
incremental_normalize(
  .data,
  incremental_type = c("ramp", "step"),
  has_baseline = TRUE,
  baseline_length = NULL,
  work_rate_magic = FALSE,
  baseline_intensity = NULL,
  ramp_increase = NULL,
  step_start = NULL,
  step_increase = NULL,
  step_length = NULL,
  ...
)
```

Arguments

`.data`	Data retrieved from `read_data()`.
`incremental_type`	The type of the incremental test performed. Either "ramp" or "step".
`has_baseline`	A boolean to indicate whether the data contains a baseline phase. This is used for an incremental test only. Default to `TRUE`.

`baseline_length`	The baseline length (in seconds) performed.
`work_rate_magic`	A boolean indicating whether to perform the work rate calculations. When set to `TRUE`, it will calculate the work rate throughout a ramp or step test. In the case of a step test, it will also perform a linear transformation of the work rate. If set to `TRUE`, the arguments below should be given. Default to `FALSE`.
`baseline_intensity`	A numeric atomic vector indicating the work rate of the baseline. If the baseline was performed at rest, indicate `0`.
`ramp_increase`	A numeric atomic vector indicating the ramp increase in watts per minute (W/min). For example, if the ramp was `30 W/min`, then pass the number `30` to this argument.
`step_start`	In case your baseline was performed at rest, you can set in this parameter at which intensity the step test started.
`step_increase`	A numeric atomic vector indicating the step increase, in watts. For example, if the step increase was `25 W` at each step, then pass the number `25` to this argument.
`step_length`	A numeric atomic vector indicating the length (in seconds) of each step in the step incremental test.
`...`	Additional arguments. Currently ignored.

Value

a tibble

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("ramp_cosmed.xlsx", package = "whippr")

## read data from ramp test
df <- read_data(path = path_example, metabolic_cart = "cosmed")

## normalize incremental test data
ramp_normalized <- df %>%
 incremental_normalize(
   .data = .,
   incremental_type = "ramp",
   has_baseline = TRUE,
   baseline_length = 240,
   work_rate_magic = TRUE,
   baseline_intensity = 20,
   ramp_increase = 25
 )

## get file path from example data
path_example_step <- system.file("step_cortex.xlsx", package = "whippr")
```

```
## read data from step test
df_step <- read_data(path = path_example_step, metabolic_cart = "cortex")

## normalize incremental test data
step_normalized <- df_step %>%
 incremental_normalize(
   .data = .,
   incremental_type = "step",
   has_baseline = TRUE,
   baseline_length = 120,
   work_rate_magic = TRUE,
   baseline_intensity = 0,
   step_start = 50,
   step_increase = 25,
   step_length = 180
 )

## End(Not run)
```

`interpolate` *Interpolate data from breath-by-breath into second-by-second*

Description

This function interpolates the data based on the time column. It takes the breath-by-breath data and transforms it into second-by-second.

Usage

```
interpolate(.data)
```

Arguments

`.data`	Data retrieved from `read_data()`.

Value

a tibble

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("example_cosmed.xlsx", package = "whippr")

## read data
df <- read_data(path = path_example, metabolic_cart = "cosmed")

df %>%
 interpolate()
```

```
## End(Not run)
```

`model_diagnostics` *Model diagnostics*

Description

Plots different model diagnostics for checking the model performance.

Usage

```
model_diagnostics(.residuals_tbl)
```

Arguments

`.residuals_tbl`	The data retrived from `get_residuals()`.

Value

a patchwork object

`normalize_first_breath` *Normalize first breath*

Description

This is needed specially when the data gets filtered. For example, if the data file does not only contain the baseline and transitions performed, we will have to normalize the time column. This function will make sure that in case the first breath does not start at zero, it will create a zero data point, duplicating the first breath. This will make sure the data does not get shifted (misalignment).

Usage

```
normalize_first_breath(.data)
```

Arguments

`.data`	Breath-by-breath data.

Value

a tibble

`normalize_time` *Normalize time column*

Description

Normalizes the the time column such that the baseline phase has negative time values. Point zero will then represent the start of the transition phase.

Usage

```
normalize_time(.data, protocol_baseline_length)
```

Arguments

`.data`	Breath-by-breath data.
`protocol_baseline_length`	The length of the baseline (in seconds).

Value

a tibble

`normalize_transitions` *Normalize transitions*

Description

Recognizes and normalizes the time column of each transition. It will also label the transitions into: 'baseline' or 'transition'.

Usage

```
normalize_transitions(
  .data,
  protocol_n_transitions,
  protocol_baseline_length,
  protocol_transition_length
)
```

Arguments

`.data`	Breath-by-breath data.
`protocol_n_transitions`	Number of transitions performed.
`protocol_baseline_length`	The length of the baseline (in seconds).
`protocol_transition_length`	The length of the transition (in seconds).

Value

a tibble

`perform_average` *Perform average on second-by-second data*

Description

This function performs either a bin- or a rolling-average on the interpolated data. You must specify the `type` of the average before continuing.

Usage

```
perform_average(
  .data,
  type = c("bin", "rolling", "ensemble"),
  bins = 30,
  rolling_window = 30
)
```

Arguments

`.data`	The second-by-second data retrieved from `interpolate()`.
`type`	The type of the average to perform. Either `bin`, `rolling`, or `ensemble`.
`bins`	If bin-average is chosen, here you can specify the size of the bin-average, in seconds. Default to 30-s bin-average.
`rolling_window`	If rolling-average is chosen, here you can specify the rolling-average window, in seconds. Default to 30-s rolling-average.

Details

Ensemble average is used in VO2 kinetics analysis, where a series of transitions from baseline to the moderate/heavy/severe intensity-domain is ensembled averaged into a single 'bout' for further data processing.

Value

a tibble

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("example_cosmed.xlsx", package = "whippr")

## read data
df <- read_data(path = path_example, metabolic_cart = "cosmed")
```

```
## interpolate and perform 30-s bin-average
df %>%
 interpolate() %>%
 perform_average(type = "bin", bins = 30)

## interpolate and perform 30-s rolling-average
df %>%
 interpolate() %>%
 perform_average(type = "rolling", rolling_window = 30)

## End(Not run)
```

`perform_kinetics` *Perform VO2 kinetics fitting*

Description

Performs the fitting process for the VO2 kinetics analysis. At this point, the data should already have been cleaned (outliers removed) and processed (interpolated, time-aligned, ensembled-averaged, and bin-averaged).

Usage

```
perform_kinetics(
  .data_processed,
  intensity_domain = c("moderate", "heavy", "severe"),
  fit_level = 0.95,
  fit_phase_1_length,
  fit_baseline_length,
  fit_transition_length,
  verbose = TRUE,
  ...
)
```

Arguments

`.data_processed`	The data retrived from `process_data()`.
`intensity_domain`	The exercise-intensity domain that the test was performed. Either *moderate*, *heavy*, or *severe*.
`fit_level`	A numeric scalar between 0 and 1 giving the confidence level for the parameter estimates in the final VO2 kinetics fit. Default to `0.95`.
`fit_phase_1_length`	The length of the phase I that you wish to exclude from the final exponential fit, in seconds. See `VO2 kinetics` section in `?vo2_kinetics` for more details.

`fit_baseline_length`	The length the baseline to perform the final linear fit, in seconds. See `VO2 kinetics` section `?vo2_kinetics` for more details.
`fit_transition_length`	The length of the transition to perform the final exponential fit, in seconds. See `VO2 kinetics` section `?vo2_kinetics` for more details.
`verbose`	A boolean indicating whether messages should be printed in the console. Default to `TRUE`.
`...`	Additional arguments when fiting VO2 kinetics in the heavy- or severe-intensity domains. Arguments may be the following: `TODO`

Details

See `?vo2_kinetics` for details.

Value

a tibble containing one row and the nested columns:

`data_fitted`	The data containing the time and VO2 columns, as well as the fitted data and its residuals for each data point.
`model`	A `nls` object. The model used in the VO2 kinetics fitting.
`model_summary`	The tidied summary of the `model`.
`model_residuals`	The residuals of the `model`.
`plot_model`	The final plot of the fitted `model`.
`plot_residuals`	The residuals plot for the `model` diagnostics.

`perform_max` *Perform VO2max calculation*

Description

It performs the calculation of VO2max, HRmax, and maximal RER. Additionally, it detects whether a plateau can be identified from your data.

Usage

```
perform_max(
  .data,
  vo2_column = "VO2",
  vo2_relative_column = NULL,
  heart_rate_column = NULL,
  rer_column = NULL,
```

```
  average_method = c("bin", "rolling"),
  average_length = 30,
  plot = TRUE,
  verbose = TRUE
)
```

Arguments

`.data`	The data retrieved either from `incremental_normalize()` or `detect_outliers()`.
`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to `"VO2"`.
`vo2_relative_column`	The name (quoted) of the column containing the relative to body weight oxygen uptake (VO2) data. Default to `NULL`.
`heart_rate_column`	The name (quoted) of the column containing the heart rate (HR) data. Default to `NULL`. If `NULL`, this parameter will not be calculated.
`rer_column`	The name (quoted) of the column containing the respiratory exchange ratio (RER) data. Default to `NULL`. If `NULL`, this parameter will not be calculated.
`average_method`	The average method to be used for VO2max calculation. One of `bin` or `rolling`.
`average_length`	The length, in seconds, of the average to be used. For example, if `average_method = bin`, and `average_length = 30`, it will perform a 30-s bin-average.
`plot`	A boolean indicating whether to produce a plot with the summary results. Default to `TRUE`.
`verbose`	A boolean indicating whether messages should be printed in the console. Default to `TRUE`.

Value

a tibble

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("ramp_cosmed.xlsx", package = "whippr")

## read data from ramp test
df <- read_data(path = path_example, metabolic_cart = "cosmed")

## normalize incremental test data
ramp_normalized <- df %>%
 incremental_normalize(
   .data = .,
   incremental_type = "ramp",
   has_baseline = TRUE,
   baseline_length = 240,
   work_rate_magic = TRUE,
```

```
    baseline_intensity = 20,
    ramp_increase = 25
  )

## detect outliers
data_ramp_outliers <- detect_outliers(
 .data = ramp_normalized,
 test_type = "incremental",
 vo2_column = "VO2",
 cleaning_level = 0.95,
 method_incremental = "linear",
 verbose = TRUE
)

## analyze VO2max
perform_max(
 .data = data_ramp_outliers,
 vo2_column = "VO2",
 vo2_relative_column = "VO2/Kg",
 heart_rate_column = "HR",
 rer_column = "R",
 average_method = "bin",
 average_length = 30,
 plot = TRUE,
 verbose = FALSE
)

## End(Not run)
```

`plot_incremental` *Plot incremental test work rate*

Description

Visualize what was done during the process of deriving the work rate from the incremental test protocol

Usage

```
plot_incremental(.data)
```

Arguments

`.data`	data retrieved from `incremental_normalize()`.

Value

a ggplot object

`plot_outliers` *Plot outliers*

Description

Plot outliers

Usage

```
plot_outliers(.data)
```

Arguments

`.data`	The data retrieved from `detect_outliers()`.

Value

a patchwork object

`predict_bands` *Extract confidence and prediction bands*

Description

It extracts confidence and prediction bands from the `nls` model. It is used only for data cleaning.

Usage

```
predict_bands(
  .data,
  time_column = "t",
  vo2_column = "VO2",
  cleaning_level = 0.95,
  cleaning_baseline_fit = c("linear", "exponential")
)
```

Arguments

`.data`	The nornalized data retrieved from `normalize_transitions()`.
`time_column`	The name (quoted) of the column containing the time. Depending on the language of your system, this column might not be "t". Therefore, you may specify it here. Default to "t".
`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to 'VO2'.

`cleaning_level`	A numeric scalar between 0 and 1 giving the confidence level for the intervals to be calculated.
`cleaning_baseline_fit`	A character indicating what kind of fit to perform for each baseline. Either 'linear' or 'exponential'.

Value

a tibble containing the following columns:

`x`	The provided time data.
`y`	The provided VO2 data.
`.fitted`	The predicted response for that observation.
`.resid`	The residual for a particular point.
`lwr_conf`	Lower limit of the confidence band.
`upr_conf`	Upper limit of the confidence band.
`lwr_pred`	Lower limit of the prediction band.
`upr_pred`	Upper limit of the prediction band.

`predict_bands_baseline` *Extract confidence and prediction bands for the baseline phase*

Description

Extract confidence and prediction bands for the baseline phase

Usage

```
predict_bands_baseline(
  .data,
  time_column,
  vo2_column,
  cleaning_level,
  cleaning_baseline_fit
)
```

Arguments

`.data`	The nornalized data retrieved from `normalize_transitions()`. The data should be filtered to only the 'baseline' phase before passing to the function.
`time_column`	The name (quoted) of the column containing the time. Depending on the language of your system, this column might not be "t". Therefore, you may specify it here. Default to "t".

`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to 'VO2'.
`cleaning_level`	A numeric scalar between 0 and 1 giving the confidence level for the intervals to be calculated.
`cleaning_baseline_fit`	A character indicating what kind of fit to perform for each baseline. Either 'linear' or 'exponential'.

Value

a tibble containing the following columns:

`x`	The provided time data.
`y`	The provided VO2 data.
`.fitted`	The predicted response for that observation.
`.resid`	The residual for a particular point.
`lwr_conf`	Lower limit of the confidence band.
`upr_conf`	Upper limit of the confidence band.
`lwr_pred`	Lower limit of the prediction band.
`upr_pred`	Upper limit of the prediction band.

`predict_bands_transition`

Extract confidence and prediction bands for the transition phase

Description

Extract confidence and prediction bands for the transition phase

Usage

```
predict_bands_transition(
  .data,
  time_column,
  vo2_column,
  cleaning_level,
  cleaning_model
)
```

Arguments

`.data`	The nornalized data retrieved from `normalize_transitions()`. The data should be filtered to only the 'transition' phase before passing to the function.
`time_column`	The name (quoted) of the column containing the time. Depending on the language of your system, this column might not be "t". Therefore, you may specify it here. Default to "t".
`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to 'VO2'.
`cleaning_level`	A numeric scalar between 0 and 1 giving the confidence level for the intervals to be calculated.
`cleaning_model`	The `nls` model to retrieve the bands from.

Value

a tibble containing the following columns:

`x`	The provided time data.
`y`	The provided VO2 data.
`.fitted`	The predicted response for that observation.
`.resid`	The residual for a particular point.
`lwr_conf`	Lower limit of the confidence band.
`upr_conf`	Upper limit of the confidence band.
`lwr_pred`	Lower limit of the prediction band.
`upr_pred`	Upper limit of the prediction band.

`print.whippr` *Whippr print method*

Description

Whippr print method

Usage

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

Arguments

`x`	A tibble with class 'whippr'
`...`	Extra arguments, not used.

`process_data` *Process data for VO2 kinetics fitting*

Description

It removes the outliers detected through `detect_outliers()`, interpolates each transition, ensemble-averages all the transitions into one, performs a bin-average, and normalizes the time column (time zero will indicate the end of baseline and the start of the transition phase).

Usage

```
process_data(.data_outliers, protocol_baseline_length, fit_bin_average)
```

Arguments

`.data_outliers`	The data retrived from `detect_outliers()`.
`protocol_baseline_length`	The length of the baseline (in seconds).
`fit_bin_average`	The bin average to be performed for the final fit.

Details

TODO

Value

a tibble with the time-aligned, ensembled-averaged, and bin-averaged data.

`read_data` *Read data from metabolic cart*

Description

It reads the raw data exported from the metabolic cart.

Usage

```
read_data(
  path,
 metabolic_cart = c("cosmed", "cortex", "nspire", "parvo", "geratherm", "cardiocoach",
    "custom"),
  time_column = "t",
  work_rate_column = NULL
)
```

Arguments

`path`	Path to read the file from.
`metabolic_cart`	Metabolic cart that was used for data collection. Currently, 'cosmed', 'cortex', 'nspire', 'parvo', 'geratherm', and 'cardiocoach' are supported. Additionaly, there is an option called 'custom' that supports files that do not have a metabolic cart-specific format.
`time_column`	The name (quoted) of the column containing the time. Depending on the language of your system, this column might not be "t". Therefore, you may specify it here. Default to "t".
`work_rate_column`	Default is `NULL`. In case your work rate column is coerced as a character column you can define here the name of this column in your data file. This happens because at the very beginning of the test the system may input a character like "-" to indicate no work rate. Therefore this is not going to get recognized as a numeric column. If your work rate column is called `WR`, for example, just pass `"WR"` to this argument.

Value

a tibble

`run_manual_cleaner` *Manual data cleaner*

Description

Usually manual data cleaning should be avoided. However, sometimes in gas exchange data there is the need to delete a few clear "bad breaths" (noise). In these situations you may use this function. Although it is encouraged that you use the `detect_outliers()` function, you may use this function at your own risk. This function can also be used to clean other kind of data, like heart rate data.

Usage

```
run_manual_cleaner(.data, width = 1200, height = 900)
```

Arguments

`.data`	The data to be manually cleaned. The first column will be always treated as the x-axis.
`width`	The width, in pixels, of the window.
`height`	the height, in pixels, of the window.

Value

The code to reproduce the manual data cleaning.

`theme_whippr` *Whippr ggplot2 theme*

Description

This theme was inspired by the plots from the Acta Physiologica Journal

Usage

```
theme_whippr(base_size = 14, base_family = "sans")
```

Arguments

`base_size`	base font size, given in pts. Default is `14`.
`base_family`	base font family. Default is `sans`.

Value

a ggplot2 object

`vo2_kinetics` *VO2 kinetics*

Description

It performs the whole process of the VO2 kinetics data analysis, which includes: data cleaning (`detect_outliers()`); outliers removal, interpolation, ensemble-averaging transitions and bin-avering final dataset (`process_data()`), and modelling VO2 kinetics (`perform_kinetics()`). This function is a general function that will call these separate functions. You can also call each one of them separately if you want.

Usage

```
vo2_kinetics(
  .data,
  intensity_domain = c("moderate", "heavy", "severe"),
  vo2_column = "VO2",
  protocol_n_transitions,
  protocol_baseline_length,
  protocol_transition_length,
  cleaning_level = 0.95,
  cleaning_baseline_fit,
  fit_level = 0.95,
  fit_bin_average,
  fit_phase_1_length,
```

```
  fit_baseline_length,
  fit_transition_length,
  verbose = TRUE,
  ...
)
```

Arguments

`.data`	Data retrieved from `read_data()`.
`intensity_domain`	The exercise-intensity domain that the test was performed. Either *moderate*, *heavy*, or *severe*.
`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to `"VO2"`.
`protocol_n_transitions`	Number of transitions performed.
`protocol_baseline_length`	The length of the baseline (in seconds).
`protocol_transition_length`	The length of the transition (in seconds).
`cleaning_level`	A numeric scalar between 0 and 1 giving the confidence level for the intervals to be calculated during the data cleaning process. Breaths lying outside the prediction bands will be excluded. Default to `0.95`.
`cleaning_baseline_fit`	A vector of the same length as the number in `protocol_n_transitions`, indicating what kind of fit to perform for each baseline. Either *linear* or *exponential*.
`fit_level`	A numeric scalar between 0 and 1 giving the confidence level for the parameter estimates in the final VO2 kinetics fit. Default to `0.95`.
`fit_bin_average`	The bin average to be performed for the final fit.
`fit_phase_1_length`	The length of the phase I that you wish to exclude from the final exponential fit, in seconds. See `VO2 kinetics` section for more details.
`fit_baseline_length`	The length the baseline to perform the final linear fit, in seconds. See `VO2 kinetics` section for more details.
`fit_transition_length`	The length of the transition to perform the final exponential fit, in seconds. See `VO2 kinetics` section for more details.
`verbose`	A boolean indicating whether messages should be printed in the console. Default to `TRUE`.
`...`	Additional arguments passed to `perform_kinetics()` when fitting VO2 kinetics in the heavy- or severe-intensity domains. See `?perform_kinetics` for more details.

Details

The function is a wrapper of smaller functions and has important arguments:

- **protocol_** = sets arguments related to the protocol used.
- **cleaning_** = sets arguments related to data cleaning.
- **fit_** = sets arguments related to VO2 kinetics fitting.

The function works like the following sequence:

`vo2_kinetics( )`:

- `detect_outliers( )` = separates the data into the number of transitions indicated, and fits each baseline and transition phase indiviudally, retrieving the predictions bands for the level indicated. Then it recognizes breaths lying outside the prediciton bands and flag them as outliers.
- `plot_outliers( )` = plots each transition identifying outliers.
- `process_data( )` = It removes the outliers detected through `detect_outliers()`, interpolates each transition, ensemble-averages all the transitions into one, performs a bin-average, and normalizes the time column (time zero will indicate the end of baseline and the start of the transition phase).
- `perform_kinetics( )` = performs the VO2 kinetics fitting based on the **fit_** parameters given. It also calculates the residuals, and plots the final fit as well as residuals for model diagnostics.

Value

a tibble containing one row and the nested columns:

`data_outliers`	The raw data containing additional columns that identify breaths as outliers.
`plot_outliers`	A `patchwork` object to display outliers from every transition.
`data_processed`	The processed data (time-aligned, ensembled-averaged, and bin-averaged).
`data_fitted`	The data containing the time and VO2 columns, as well as the fitted data and its residuals for each data point.
`model`	A `nls` object. The model used in the VO2 kinetics fitting.
`model_summary`	The tidied summary of the `model`.
`model_residuals`	The residuals of the `model`.
`plot_model`	The final plot of the fitted `model`.
`plot_residuals`	The residuals plot for the `model` diagnostics.

VO2 kinetics

VO2 kinetics, described as the rate of adjustment of the oxidative energy system to an instantaneous increase in the energy demand, is exponential in nature, and it is described by the oxygen uptake (VO2) time-constant (τVO2) (Murias, Spencer and Paterson (2014); Poole and Jones (2011)).

VO2 kinetics analysis provides understanding of the mechanisms that regulate the rate at which oxidative phosphorylation adapts to step changes in exercise intensities and ATP requirement. This is usually accomplished by performing step transitions from a baseline intensity to a higher work rate in either the **moderate-**, **heavy-**, or **severe-intensity domain** (Murias et al., 2011).

Three distinct phases may be observed in the VO2 response during on-transient exercise:

Phase I: also termed as the cardiodynamic phase, it represents the circulatory transit delay on the VO2 response as a result of the increase in the pulmonary blood flow that does not reflect the increase in oxygen extraction in the active muscles. The time-window of the Phase I is determined in the `fit_phase_1_length` argument, which will be internally passed into the `perform_kinetics()` function.

Phase II: also termed as the primary component, represents the exponential increase in VO2 related to the continued increase in pulmonary and muscle blood flow. The Phase II is described by the time-constant parameter (τ) in the mono-exponential model (see below), and it is defined as the duration of time (in seconds) for the VO2 response to increase to 63% of the required steady-state.

Phase III: represents the steady-state phase of the VO2 response during moderate-intensity exercise.

Moderate-intensity domain:

The on-transient response from baseline to a transition within the **moderate-intensity domain** is analyzed using a **mono-exponential model**:

$$VO_{2(t)} = baseline + amplitude \cdot \left(1 - e^{-\frac{(t-TD)}{tau}}\right)$$

where:

- `VO2(t)` = the oxygen uptake at any given time.
- `baseline` = the oxygen uptake associated with the baseline phase.
- `amplitude` = the steady-state increase increase in oxygen uptake above `baseline`.
- `TD` = the time delay.
- τ = the time constant defined as the duration of time for the oxygen uptake to increase to 63% of the steady-state increase.

The baseline value in the mono-exponential model is a **fixed** value and pre-determined as the mean of the VO2 response (i.e., linear model with the slope set as zero) during the baseline phase. The time window of the baseline period is determined in the `fit_baseline_length` argument, which will be internally passed into the `perform_kinetics()` function.

Diverse exercise protocols exist to determine VO2 kinetics in the moderate-intensity domain. Usually, the protocol consists of multiple transitions (typically 3 or 4) from a baseline exercise-intensity to an exercise-intensity below the gas exchange threshold (typically the power output associated with 90% of the gas exchange threshold). Bbaseline and transition phases are usually performed for 6 minutes each. The reason that 6 minutes is done for each phase is to give enough time for both to reach a steady-state response:

For example, for each multiple of the time-constant (τ), VO2 increases by 63% of the difference between the previous τ and the required steady-state. This means:

- `1` τ `= 63%` Δ.
- `2` τ `= 86%` Δ `[100% - 63% = 37%; (37% x 63%) + 63% = 86%]`.
- `3` τ `= 95%` Δ `[100% - 86% = 14%; (14% x 63%) + 86% = 95%]`.
- `4` τ `= 98%` Δ `[100% - 95% = 5%; (5% x 63%) + 95% = 98%]`.

In practical terms, let's imagine that a given participant has a τ = **60 seconds**. This means that this person would need **240 seconds** (`4 x 60`) to reach **steady-state** (98% of the response) in the **moderate-intensity domain**. This would leave other 120 seconds (2 minutes) of transition, so the protocol of performing 6-min transitions makes sure enough time is given.

Now let's imagine that another person has a τ = **20 seconds**. This means that this person would need **80 seconds** (`4 x 20`) to reach **steady-state** (98% of the response) in the **moderate-intensity domain**.

Given that there is enough time to reach a VO2 steady-state response with 6 minutes of transition, that means that for the final fit (when the transitions were cleaned, ensembled-averaged, and bin-averaged) there is no need to include the whole 6 minutes of the transition. This strategy avoids superfluous sections of the steady-state data, thus maximizing the quality of the fit during the exercise on-transient (Bell et al., 2001). This may be specified through the `fit_transition_length` argument, which will be internally passed into the `perform_kinetics()` function.

As for bin-averages in the final fit, usually the data are averaged into 5-s or 10-s bins, 5-s being the most common (Keir et al., 2014). This may be specified through the `fit_bin_average` argument, which will be internally passed into the `process_data()` function.

Heavy- and severe-intensity domains:

TODO

References

Bell, C., Paterson, D. H., Kowalchuk, J. M., Padilla, J., & Cunningham, D. A. (2001). A comparison of modelling techniques used to characterise oxygen uptake kinetics during the on-transient of exercise. Experimental Physiology, 86(5), 667-676.

Keir, D. A., Murias, J. M., Paterson, D. H., & Kowalchuk, J. M. (2014). Breath-by-breath pulmonary O2 uptake kinetics: effect of data processing on confidence in estimating model parameters. Experimental physiology, 99(11), 1511-1522.

Murias, J. M., Spencer, M. D., & Paterson, D. H. (2014). The critical role of O2 provision in the dynamic adjustment of oxidative phosphorylation. Exercise and sport sciences reviews, 42(1), 4-11.

Murias, J. M., Spencer, M. D., Kowalchuk, J. M., & Paterson, D. H. (2011). Influence of phase I duration on phase II VO2 kinetics parameter estimates in older and young adults. American Journal of Physiology-regulatory, integrative and comparative physiology, 301(1), R218-R224.

Poole, D. C., & Jones, A. M. (2011). Oxygen uptake kinetics. Comprehensive Physiology, 2(2), 933-996.

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("example_cosmed.xlsx", package = "whippr")

## read data
df <- read_data(path = path_example, metabolic_cart = "cosmed", time_column = "t")

## VO2 kinetics analysis
results_kinetics <- vo2_kinetics(
  .data = df,
```

```
  intensity_domain = "moderate",
  vo2_column = "VO2",
  protocol_n_transitions = 3,
  protocol_baseline_length = 360,
  protocol_transition_length = 360,
  cleaning_level = 0.95,
  cleaning_baseline_fit = c("linear", "exponential", "exponential"),
  fit_level = 0.95,
  fit_bin_average = 5,
  fit_phase_1_length = 20,
  fit_baseline_length = 120,
  fit_transition_length = 240,
  verbose = TRUE
)

## End(Not run)
```

`vo2_max` *VO2max*

Description

It performs the whole process of the VO2max data analysis, which includes: data standardization and normalization according to incremental protocol (`incremental_normalize()`), 'bad breaths' detection (`detect_outliers()`), mean response time calculation (`incremental_mrt()`) (currently ignored), and maximal values calculation (VO2, PO, HR, RER) (`perform_max()`).

Usage

```
vo2_max(
  .data,
  vo2_column = "VO2",
  vo2_relative_column = NULL,
  heart_rate_column = NULL,
  rer_column = NULL,
  detect_outliers = TRUE,
  average_method = c("bin", "rolling"),
  average_length = 30,
  mrt,
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`.data`	Data retrieved from `read_data()`.
`vo2_column`	The name (quoted) of the column containing the absolute oxygen uptake (VO2) data. Default to `"VO2"`.

`vo2_relative_column`	The name (quoted) of the column containing the relative to body weight oxygen uptake (VO2) data. Default to `NULL`.
`heart_rate_column`	The name (quoted) of the column containing the heart rate (HR) data. Default to `NULL`. If `NULL`, this parameter will not be calculated.
`rer_column`	The name (quoted) of the column containing the respiratory exchange ratio (RER) data. Default to `NULL`. If `NULL`, this parameter will not be calculated.
`detect_outliers`	A boolean indicating whether to detect outliers. Default to `TRUE`.
`average_method`	The average method to be used for VO2max calculation. One of `bin` or `rolling`.
`average_length`	The length, in seconds, of the average to be used. For example, if `average_method = bin`, and `average_length = 30`, it will perform a 30-s bin-average.
`mrt`	A boolean indicating whether to calculate the mean response time. To be implemented soon <- currently ignored.
`plot`	A boolean indicating whether to produce a plot with the summary results. Default to `TRUE`.
`verbose`	A boolean indicating whether messages should be printed in the console. Default to `TRUE`.
`...`	Additional arguments passed onto `incremental_normalize()`, `detect_outliers()` if `detect_outliers = TRUE`, and `incremental_mrt()` if `mrt = TRUE`.

Details

TODO

Value

a tibble containing one row and the following columns:

`VO2max_absolute`	The absolute VO2max.
`VO2max_relative`	The relative VO2max.
`POpeak`	The peak power output.
`HRmax`	The maximal heart rate.
`RERmax`	The maximal RER.
`plot`	The plot, if `plot = TRUE`.

Examples

```
## Not run:
## get file path from example data
path_example <- system.file("ramp_cosmed.xlsx", package = "whippr")

## read data from ramp test
```

```
df <- read_data(path = path_example, metabolic_cart = "cosmed")

## normalize incremental test data
ramp_normalized <- df %>%
 incremental_normalize(
   .data = .,
   incremental_type = "ramp",
   has_baseline = TRUE,
   baseline_length = 240,
   work_rate_magic = TRUE,
   baseline_intensity = 20,
   ramp_increase = 25
 )

## detect outliers
data_ramp_outliers <- detect_outliers(
 .data = ramp_normalized,
 test_type = "incremental",
 vo2_column = "VO2",
 cleaning_level = 0.95,
 method_incremental = "linear",
 verbose = TRUE
)

## analyze VO2max
perform_max(
 .data = data_ramp_outliers,
 vo2_column = "VO2",
 vo2_relative_column = "VO2/Kg",
 heart_rate_column = "HR",
 rer_column = "R",
 average_method = "bin",
 average_length = 30,
 plot = TRUE,
 verbose = FALSE
)

## End(Not run)
```

Index