Package: accelEE (via r-universe)

May 13, 2026

Type Package

Title Predict Energy Expenditure from Accelerometer Data

Version 0.3.1

Description Simplifies the application of various energy expenditure models. The package is intended as a hub that brings together methods from a variety of other, themed packages such as 'Sojourn' and 'TwoRegression'. Several methods are supported locally as well, including the linear methods of Hildebrand et al. (2014) <doi:10.1249/MSS.0000000000000289> and the non-linear adaptation by Ellingson et al. (2017) <doi:10.1088/1361-6579/aa6d00>. The package can combine output from different methods and produce standardized output in a range of units.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://github.com/paulhibbing/accelEE

BugReports https://github.com/paulhibbing/accelEE/issues

Imports digest, dplyr, lubridate, magrittr, nnet, PAutilities (>= 1.3.0), randomForest, rlang, Sojourn (>= 1.1.0), stats, tidyr, tree, TwoRegression (>= 1.0.0), utils

RoxygenNote 7.3.3

Depends R (>= 2.10)

Suggests EE.Data, PhysActBedRest (>= 1.1), PhysicalActivity (>= 0.2.4), read.gt3x (>= 1.0.2)

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-05-07 21:38:07 UTC

RemoteUrl https://github.com/paulhibbing/accelee

RemoteRef HEAD

RemoteSha 2dc9cc383cdb6e125c8433957a06e37cee5d577a

Contents

accelEE-function 2
ee_file 8
ee_summary 9
generic_features 10
hibbing23-summary 11
montoye_features 12
staudenmayer_features 13

Index **15**

`accelEE-function` *Predict energy expenditure for accelerometry data*

Description

Predict energy expenditure for accelerometry data

Usage

```
## Wrapper function:

  accelEE(
    d, method = c(
      "Crouter 2006", "Crouter 2010", "Crouter 2012", "Crouter 2015",
      "Hibbing 2018", "Hildebrand Linear", "Hildebrand Non-Linear",
      "Montoye 2017", "SIP", "Sojourn 1x", "Sojourn 3x",
      "Staudenmayer Linear", "Staudenmayer Random Forest"
    ), time_var = "Timestamp", output_epoch = "default",
    warn_high_low = TRUE, met_mlkgmin = 3.5, RER = 0.85,
    feature_calc = TRUE, shrink_output = TRUE, combine = TRUE,
    ee_vars = c("METs", "VO2", "kcal"), verbose = FALSE, ...
  )


## Internal applicator functions called by the wrapper, based on
## the value of the `method` argument (external functions listed
## under 'See Also'):

  wrap_2RM(
    d, time_var = "Timestamp", output_epoch = "default",
    max_mets = 20, warn_high_low = TRUE,
    met_mlkgmin = 3.5, RER = 0.85,
    feature_calc = TRUE, shrink_output = TRUE, verbose = FALSE,
    method = c(
      "Crouter 2006", "Crouter 2010",
      "Crouter 2012", "Hibbing 2018"
```

```
  ), ..., met_name = "METs", tag = ""
)

crouter15(
  d, time_var = "Timestamp", output_epoch = "default",
  min_mets = 1, max_mets = 20, warn_high_low = TRUE,
  met_mlkgmin = 3.5, RER = 0.85,
  shrink_output = TRUE, verbose = FALSE,
  model = c("VA", "VM"), movement_var = "Axis1",
  ...
)

hildebrand_linear(
  d, time_var = "Timestamp", output_epoch = "default",
  min_vo2_mlkgmin = 3, max_vo2_mlkgmin = 70, warn_high_low = TRUE,
  met_mlkgmin = 3.5, RER = 0.85,
  feature_calc = TRUE, shrink_output = TRUE, verbose = FALSE,
  age = c("youth", "adult"), monitor = c("ActiGraph", "GENEActiv"),
  location = c("hip", "wrist"), enmo_name = "ENMO", ...
)

hildebrand_nonlinear(
  d, time_var = "Timestamp", output_epoch = "default",
  min_vo2_mlkgmin = 3, max_vo2_mlkgmin = 70, warn_high_low = TRUE,
  met_mlkgmin = 3.5, RER = 0.85,
  feature_calc = TRUE, shrink_output = TRUE,
  verbose = FALSE, enmo_name = "ENMO", ...
)

montoye(
  d, time_var = "Timestamp", output_epoch = "default",
  min_mets = 1, max_mets = 20, warn_high_low = TRUE,
  met_mlkgmin = 3.5, RER = 0.85,
  feature_calc = TRUE, shrink_output = TRUE,
  verbose = FALSE, side = c("left", "right"),
  ...
)

sojourn(
  d,  time_var = "Timestamp", output_epoch = "default",
  min_mets = 1, max_mets = 20, warn_high_low = TRUE,
  met_mlkgmin = 3.5, RER = 0.85,
  shrink_output = TRUE, verbose = FALSE,
  axis1 = "Axis1", axis2 = "Axis2", axis3 = "Axis3",
  vector.magnitude = "Vector.Magnitude",
  method = c("SIP", "Sojourn 1x", "Sojourn 3x"),
  ..., met_name = "METs", tag = ""
)
```

```
staudenmayer(
  d, time_var = "Timestamp", output_epoch = "default",
  min_mets = 1, max_mets = 20, warn_high_low = TRUE,
  met_mlkgmin = 3.5, RER = 0.85,
  feature_calc = TRUE, shrink_output = TRUE,
  verbose = FALSE, ..., select = c("METs_lm", "METs_rf")
)
```

Arguments

d	data frame of data to use for generating predictions
method	the method(s) to use
time_var	character. Name of the column containing POSIX-formatted timestamps
output_epoch	character. The desired epoch length of output. Acceptable options are "default" or else a setting appropriate for the unit argument of lubridate::floor_date()
warn_high_low	logical. Issue warnings when corrections are applied to values that are out of range?
met_mlkgmin	conversion factor for transforming oxygen consumption (in ml/kg/min) into metabolic equivalents (METs)
RER	the respiratory exchange ratio. Used for determining conversion factors when calculating caloric expenditure from oxygen consumption
feature_calc	logical. Calculate features for the selected method(s)? If FALSE, the assumption is that features have already been calculated
shrink_output	logical. Reduce the number of columns in output by removing calculated feature columns? Default is TRUE. May only have an impact on output in certain cases, particularly when setting output_epoch = "default" and/or method = c("Montoye 2017", ...)
combine	logical. Combine results from each method into a single data frame? If TRUE (the default), the results will all be collapsed to a commonly-compatible epoch length, which may override output_epoch if a suitable selection is not given
ee_vars	character vector indicating which energy expenditure variables to return. Choose one or more of "METs", "VO2", and "kcal" (case insensitive)
verbose	logical. Print updates to console?
...	arguments passed to specific applicators and beyond. See details
min_mets	minimum allowable metabolic equivalent (MET) value. Values lower than this (if any) will be rounded up to it
max_mets	maximum allowable metabolic equivalent (MET) value. Values higher than this (if any) will be rounded down to it
min_vo2_mlkgmin	minimum allowable oxygen consumption value (in ml/kg/min). Values lower than this (if any) will be rounded up to it
max_vo2_mlkgmin	maximum allowable oxygen consumption value (in ml/kg/min). Values higher than this (if any) will be rounded down to it

model	character. Can be "VA" and/or "VM", specifying which of the Crouter 2015 model(s) to use
movement_var	character. name of the variable(s) on which the Crouter 2015 method should be applied. Length must match the length of model. It is assumed that the first elements of movement_var and model will correspond with one another, and the same for the second elements (if applicable)
age	the age group(s) of desired Hildebrand equation(s) to apply
monitor	the monitor being worn by the participant
location	the placement of the monitor on the body
enmo_name	name of the variable containing Euclidian Norm Minus One (ENMO) values
side	character vector or scalar indicating which side-specific wrist model(s) to implement for "Montoye 2017". Can be "left", "right", or c("left", "right")
axis1	for Sojourn 1x and Sojourn 3x, the name of the variable in d containing vertical axis activity counts
axis2	for Sojourn 3x, the name of the variable in d containing horizontal axis activity counts
axis3	for Sojourn 3x, the name of the variable in d containing lateral axis activity counts
vector.magnitude	for Sojourn 3x, the name of the variable in d containing vector magnitude activity counts
met_name	[for internal use] A character scalar giving the name of the column containing metabolic equivalent values (METs)
tag	[for internal use] A character scalar giving an informative tag to add when naming variables
select	[for internal use] A character scalar or vector indicating which Staudenmayer model(s) to run

Details

This is a wrapper and aggregator for applying different energy expenditure prediction methods. Depending on the value(s) specified in the method argument, calls are made to applicator functions (one per method). Most applicators require values to be passed in for additional variables. Thus, the signature for each applicator function is included above, in the usage section.

For TwoRegression methods, a customized internal wrapper (wrap_2RM) is used around TwoRegression. Additional arguments can be passed to that function directly through this one. Similarly for Sojourn methods, additional arguments can be passed directly to the corresponding functions from the Sojourn package. Links to those are below.

For Staudenmayer and Montoye methods, values can be passed directly to staudenmayer_features and montoye_features, respectively (if feature calculation is requested via the feature_calc argument).

Value

A data frame appended with new columns containing energy expenditure predictions

Note

Some things to be aware of:

1. Oxygen consumption values are converted to kcal using factors from the Lusk table (by default, 4.862 kcal/L, corresponding to RER of 0.85; see 'References' below).

2. Not all methods can necessarily be combined through a single call. This capability is dependent on the desired settings and format of the output. There are too many possibilities and contingencies to list in a single documentation file. Options and adaptations can be discussed on GitHub.

3. The `wrap_2RM` applicator does not have a formal `min_mets` argument because the default minima differ depending on the method being implemented. However, you can still pass a value for `min_mets`, and it will get forwarded to the `TwoRegression` package and applied as expected.

References

Crouter et al. (2006). doi:10.1152/japplphysiol.00818.2005

Crouter et al. (2010). doi:10.1249/MSS.0b013e3181c37458

Crouter et al. (2012), doi:10.1249/MSS.0b013e3182447825

Crouter et al. (2015). doi:10.1249/MSS.0000000000000502

Hibbing et al. (2018). doi:10.1249/MSS.0000000000001532

Hildebrand et al. (2014). doi:10.1249/MSS.0000000000000289

Hildebrand et al. (2017). doi:10.1111/sms.12795

Ellingson et al. (2017). doi:10.1088/13616579/aa6d00

Montoye et al. (2017). doi:10.1080/1091367X.2017.1337638

Lyden et al. (2014). doi:10.1249/MSS.0b013e3182a42a2d

Ellingson et al. (2016). doi:10.1249/MSS.0000000000000915

Staudenmayer et al. (2015). doi:10.1152/japplphysiol.00026.2015

See Also

Lusk, G. (1924). Analysis of the oxidation of mixtures of carbohydrate and fat: a correction. *Journal of Biological Chemistry*, 59, 41-42.

`TwoRegression`

`sojourn_3x_SIP`

`soj_1x_original`

`soj_3x_original`

Examples

```
#### Below, note the variations throughout the examples,
#### showing different ways you can customize the output



## Raw acceleration examples:
```

```
if (isTRUE(requireNamespace("read.gt3x", quietly = TRUE))) {

  f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")
  d <- stats::setNames(
    read.gt3x::read.gt3x(f, asDataFrame = TRUE, imputeZeroes = TRUE),
    c("Timestamp", "Accelerometer_X", "Accelerometer_Y", "Accelerometer_Z")
  )[1:30000, ]

  utils::head(
    accelEE(
      d, "Hibbing 2018", algorithm = 1,
      site = c("Left Wrist", "Right Wrist"),
      warn_high_low = FALSE, shrink_output = FALSE
    )
  )

  utils::head(
    accelEE(
      d, c("Hildebrand Linear", "Hildebrand Non-Linear"), age = "adult",
      monitor = "ActiGraph", location = "Wrist", warn_high_low = FALSE,
      ee_vars = c("METs", "kcal"), output_epoch = "60 sec"
    )
  )

  accelEE(
    d, c(
      "Montoye 2017", "Staudenmayer Linear",
      "Staudenmayer Random Forest"
    ), side = "left", ee_vars = "VO2", combine = FALSE
  )

}


## Activity count examples:

if (isTRUE(requireNamespace("TwoRegression", quietly = TRUE))) {

  data(count_data, package = "TwoRegression")

  utils::head(
    accelEE(
      count_data, c("Crouter 2006", "Crouter 2010"),
      movement_var = "Axis1", time_var = "time"
    )
  )

}


if (isTRUE(requireNamespace("Sojourn", quietly = TRUE))) {
```

```
  # Sojourn methods can't be implemented in a single call,
  # but you can chain them together, particularly with
  # `magrittr` piping although that is not shown below

  data(SIP_ag, package = "Sojourn")
  data(SIP_ap, package = "Sojourn")
  d <- Sojourn::enhance_actigraph(SIP_ag, SIP_ap)

  soj_results <- accelEE(d, "SIP", time_var = "Time", warn_high_low = FALSE)
  #^^Note that the SIP method causes a `Timestamp` variable to be
  #  silently populated, whereas the input data frame must have a column
  #  named `Time`. In general, the Sojourn methods (especially SIP) are
  #  currently coded somewhat inflexibly, often requiring specific
  #  variable names for the input. Your best bet for getting them to run
  #  is to copy, paste, and execute the package examples, then format your
  #  data to match the example data exactly.

  soj_results <- accelEE(
    soj_results, "Sojourn 1x", axis1 = "counts", time_var = "Time"
  )

  soj_results <- accelEE(
    soj_results, "Sojourn 3x", axis1 = "counts", axis2 = "axis2",
    axis3 = "axis3", vector.magnitude = "vm", output_epoch = "60 sec"
  )
  #^^Note that this collapses everything to one-minute epochs

  utils::head(soj_results)

}
```

`ee_file` *Run a pre-specified processing scheme*

Description

Pre-specified routines are designed to facilitate replication of methods from prior studies

Usage

```
ee_file(filename, scheme = "Hibbing 2023", ...)
```

Arguments

`filename`	character. Path to the file
`scheme`	character. Name of the routine to be executed. Currently the only option is `"Hibbing 2023"`.
`...`	Arguments passed to sub-routine functions

Value

A data frame whose contents are prepared according to the indicated scheme

See Also

hibbing23-file

Examples

```
f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")
ee_file(f)
```

`ee_summary` *Run a pre-specified summary scheme*

Description

Pre-specified routines are designed to facilitate replication of methods from prior studies

Usage

```
ee_summary(d, scheme = "Hibbing 2023", ...)
```

Arguments

`d`	input data (presumably a data frame)
`scheme`	character. Name of the routine to be executed. Currently the only option is `"Hibbing 2023"`.
`...`	Arguments passed to sub-routine functions

Value

A data frame whose contents are prepared according to the indicated scheme

See Also

Subroutine function(s): hibbing23-summary

Examples

```
f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")

## Read acceleration data
  accel <- ee_file(f)

## Generate some bogus count data
  cts <- data.frame(
    Timestamp = accel$Timestamp,
```

```
    Axis1 = 0, Axis2 = 0, Axis3 = 0,
    valid_status = "Non-Wear"
  )

## Returns NA with a warning about no complete days in the file
  suppressWarnings(ee_summary(
    merge(accel, cts)
  ))
```

---

generic_features *Calculate generic features for model application*

Description

Calculate generic features for model application

Usage

```
generic_features(
  d,
  time_var = "Timestamp",
  x_var = "Accelerometer_X",
  y_var = "Accelerometer_Y",
  z_var = "Accelerometer_Z",
  win_width_sec = 1,
  verbose = FALSE,
  ...
)
```

Arguments

d	data frame of data to use for generating predictions
time_var	character. Name of the column containing POSIX-formatted timestamps
x_var	character. Name of the X-axis variable
y_var	character. Name of the Y-axis variable
z_var	character. Name of the Z-axis variable
win_width_sec	desired window width for features
verbose	logical. Print updates to console?
...	currently unused

Value

A data frame of features in the specified epoch length

Examples

```
if (isTRUE(requireNamespace("read.gt3x"))) {

  f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")
  d <- stats::setNames(
    read.gt3x::read.gt3x(f, asDataFrame = TRUE, imputeZeroes = TRUE),
    c("Timestamp", "Accelerometer_X", "Accelerometer_Y", "Accelerometer_Z")
  )[1:30000, ]

  utils::head(generic_features(d))

}
```

---

hibbing23-summary *Run the Hibbing 2023 summary scheme*

Description

Run the Hibbing 2023 summary scheme

Usage

```
ee_summary_hibbing23(d, verbose = FALSE)
```

Arguments

d	input data (presumably a data frame)
verbose	logical. Print updates to console?

Value

A data frame whose contents are prepared according to the indicated scheme

Examples

```
f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")

## Read acceleration data
  accel <- ee_file(f)

## Generate some bogus count data
  cts <- data.frame(
    Timestamp = accel$Timestamp,
    Axis1 = 0, Axis2 = 0, Axis3 = 0,
    valid_status = "Non-Wear"
  )

## Returns NA with a warning about no complete days in the file
  suppressWarnings(ee_summary(
```

```
  merge(accel, cts)
))
```

`montoye_features` *Calculate features for Montoye's neural networks*

Description

Calculate features for Montoye's neural networks

Usage

```
montoye_features(
  d,
  time_var = "Timestamp",
  x_var = "Accelerometer_X",
  y_var = "Accelerometer_Y",
  z_var = "Accelerometer_Z",
  side = c("left", "right"),
  win_width_sec = 30,
  verbose = FALSE,
  ...
)
```

Arguments

`d`	data frame of data to use for generating predictions
`time_var`	character. Name of the column containing POSIX-formatted timestamps
`x_var`	character. Name of the X-axis variable
`y_var`	character. Name of the Y-axis variable
`z_var`	character. Name of the Z-axis variable
`side`	character vector or scalar indicating which side-specific wrist model(s) to implement for `"Montoye 2017"`. Can be `"left"`, `"right"`, or `c("left", "right")`
`win_width_sec`	desired window width for features
`verbose`	logical. Print updates to console?
`...`	currently unused

Value

A dataframe of features for entry into the neural networks

References

Montoye et al. (2017) doi:10.1080/1091367X.2017.1337638

Examples

```
if (isTRUE(requireNamespace("read.gt3x"))) {

  f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")
  d <- read.gt3x::read.gt3x(f, asDataFrame = TRUE, imputeZeroes = TRUE)

  montoye_features(d[1:12000, ], "time", "X", "Y","Z", "right")

}
```

`staudenmayer_features` *Calculate features for Staudenmayer models*

Description

Calculate features for Staudenmayer models

Usage

```
staudenmayer_features(
  d,
  time_var = "Timestamp",
  x_var = "Accelerometer_X",
  y_var = "Accelerometer_Y",
  z_var = "Accelerometer_Z",
  win_width_sec = 15,
  verbose = FALSE,
  ...
)
```

Arguments

`d`	data frame of data to use for generating predictions
`time_var`	character. Name of the column containing POSIX-formatted timestamps
`x_var`	character. Name of the X-axis variable
`y_var`	character. Name of the Y-axis variable
`z_var`	character. Name of the Z-axis variable
`win_width_sec`	desired window width for features
`verbose`	logical. Print updates to console?
`...`	currently unused

Value

Data frame containing features in the specified format

References

Staudenmayer et al. (2015), doi:10.1152/japplphysiol.00026.2015

Examples

```
if (isTRUE(requireNamespace("read.gt3x"))) {

  f <- system.file("extdata/TAS1H30182785_2019-09-17.gt3x", package = "read.gt3x")
  d <- read.gt3x::read.gt3x(f, asDataFrame = TRUE, imputeZeroes = TRUE)

  staudenmayer_features(d[1:12000, ], "time", "X", "Y", "Z")

}
```

Index

accelEE *(accelEE-function)*, 2
accelEE-function, 2

crouter15 *(accelEE-function)*, 2

ee_file, 8
ee_summary, 9
ee_summary_hibbing23
        (hibbing23-summary), 11

generic_features, 10

hibbing23-summary, 11
hildebrand_linear *(accelEE-function)*, 2
hildebrand_nonlinear
        (accelEE-function), 2

montoye *(accelEE-function)*, 2
montoye_features, *5*, 12

soj_1x_original, *6*
soj_3x_original, *6*
sojourn *(accelEE-function)*, 2
sojourn_3x_SIP, *6*
staudenmayer *(accelEE-function)*, 2
staudenmayer_features, *5*, 13

TwoRegression, *5, 6*

wrap_2RM *(accelEE-function)*, 2