

# Package ‘TwoRegression’

November 19, 2025

**Type** Package

**Title** Develop and Apply Two-Regression Algorithms

**Version** 1.1.1

**Depends** R (>= 3.5.0)

**Description** Facilitates development and application of two-regression algorithms for research-grade wearable devices. It provides an easy way for users to access previously-developed algorithms, and also to develop their own. Initial motivation came from Hibbing PR, LaMunion SR, Kaplan AS, & Crouter SE (2018) <[doi:10.1249/MSS.0000000000001532](https://doi.org/10.1249/MSS.0000000000001532)>. However, other algorithms are now supported. Please see the associated references in the package documentation for full details of the algorithms that are supported.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** dplyr (>= 0.5.0), ggplot2 (>= 3.4.0), gridExtra (>= 2.3), lubridate, magrittr (>= 1.5), PAutilities (>= 1.1.0), pROC (>= 1.16.0), RcppRoll, rlang, stats, tidyr

**RoxygenNote** 7.1.2

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**URL** <https://github.com/paulhibbing/TwoRegression>

**BugReports** <https://github.com/paulhibbing/TwoRegression/issues>

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Paul R. Hibbing [aut, cre],  
Vincent T. van Hees [ctb]

**Maintainer** Paul R. Hibbing <paulhibbing@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-11-19 19:50:13 UTC

## Contents

all_data . . . . .	2
count_data . . . . .	3
fit_2rm . . . . .	3
imu_to_check . . . . .	5
imu_to_collapse . . . . .	6
plot.TwoRegression . . . . .	7
predict.TwoRegression . . . . .	9
raw_for_cv . . . . .	11
raw_to_collapse . . . . .	12
smooth_2rm . . . . .	12
TwoRegression-Package . . . . .	13
<b>Index</b>	<b>15</b>

---

all_data	<i>Two-regression-ready data frame</i>
----------	--

---

### Description

A dataset with pre-processed primary accelerometer and IMU data that is ready for applying a two-regression algorithm.

### Usage

all\_data

### Format

A data frame with 299 rows and 17 variables:

**PID** Participant ID

**file\_source\_PrimaryAccel** The filename of the primary accelerometer file

**date\_processed\_PrimaryAccel** The date the primary accelerometer file was processed

**file\_source\_IMU** The filename of the IMU file

**date\_processed\_IMU** The date the IMU file was processed

**Timestamp** The corresponding time for each row of data

**day\_of\_year** The numeric day of the year, i.e., the Julian date

**minute\_of\_day** The numeric minute of the day

**ENMO** Euclidian Norm Minus One, in milli-g

**Gyroscope\_VM\_DegPerS** Gyroscope vector magnitude, in degrees per second

**mean\_abs\_Gyroscope\_x\_DegPerS** Rotation in x axis, degrees per second

**mean\_abs\_Gyroscope\_y\_DegPerS** Rotation in y axis, degrees per second

**mean\_abs\_Gyroscope\_z\_DegPerS** Rotation in z axis, degrees per second

**mean\_magnetometer\_direction** Cardinal direction of magnetometer signal, averaged over one second

**ENMO\_CV10s** Coefficient of variation per 10-s, applied to Euclidian Norm Minus One

**GVM\_CV10s** Coefficient of variation per 10-s, applied to gyroscope vector magnitude

**Direction** Direction changes per 5-s

---

count\_data

*Activity count data for demonstrating prior two-regression models*

---

### Description

A small amount of 10-s epoch activity counts for code examples

### Usage

count\_data

### Format

A data frame with 30 rows and 5 variables:

**time** POSIX. The timestamp

**Axis1** numeric. The vertical axis activity counts

**Axis2** numeric. The horizontal axis

**Axis3** numeric. The lateral axis

**Vector.Magnitude** numeric. The vector magnitude of all three axes

---

fit\_2rm

*Develop a two-regression algorithm*

---

### Description

Develop a two-regression algorithm

Check if an object has class TwoRegression

**Usage**

```

fit_2rm(
  data,
  activity_var,
  sed_cp_activities,
  sed_activities,
  sed_cp_var,
  sed_METs,
  walkrun_activities,
  walkrun_cp_var,
  met_var,
  walkrun_formula,
  intermittent_formula,
  method = "user_unspecified"
)

is.TwoRegression(x)

```

**Arguments**

<code>data</code>	The data with which to develop the algorithm
<code>activity_var</code>	Character scalar. Name of the variable defining which activity is being performed
<code>sed_cp_activities</code>	Character vector. Activities to be included in the process of forming the sedentary classifier
<code>sed_activities</code>	Character vector. Actual sedentary activities
<code>sed_cp_var</code>	Character scalar. Name of the variable on which the sedentary cut-point is defined
<code>sed_METs</code>	Numeric scalar. Metabolic equivalent value to apply to sedentary activities
<code>walkrun_activities</code>	Character vector. Actual ambulatory activities
<code>walkrun_cp_var</code>	Character scalar. Name of the variable on which the walk/run cut-point is defined
<code>met_var</code>	Character scalar. Name of the variable giving actual energy expenditure (in metabolic equivalents)
<code>walkrun_formula</code>	Character scalar. Formula to use for developing the walk/run regression model
<code>intermittent_formula</code>	Character scalar. Formula to use for developing the intermittent activity regression model
<code>method</code>	character scalar. Optional name for the model, potentially useful for printing.
<code>x</code>	object to be tested

**Value**

An object of class ‘TwoRegression’

**See Also**

[predict.TwoRegression](#) [summary.TwoRegression](#) [plot.TwoRegression](#)

**Examples**

```
set.seed(307)

data(all_data, package = "TwoRegression")
fake_sed <- c("Lying", "Sitting")
fake_lpa <- c("Sweeping", "Dusting")
fake_cwr <- c("Walking", "Running")
fake_ila <- c("Tennis", "Basketball")

fake_activities <- c(fake_sed, fake_lpa, fake_cwr, fake_ila)

all_data$Activity <- sample(fake_activities, nrow(all_data), TRUE)

all_data$fake_METs <- ifelse(
  all_data$Activity %in% c(fake_sed, fake_lpa),
  runif(nrow(all_data), 1, 2),
  runif(nrow(all_data), 2.5, 8)
)

fit_2rm(
  data = all_data,
  activity_var = "Activity",
  sed_cp_activities = c(fake_sed, fake_lpa),
  sed_activities = fake_sed,
  sed_cp_var = "ENMO",
  sed_METs = 1.25,
  walkrun_activities = fake_cwr,
  walkrun_cp_var = "ENMO_CV10s",
  met_var = "fake_METs",
  walkrun_formula = "fake_METs ~ ENMO",
  intermittent_formula = "fake_METs ~ ENMO + I(ENMO^2) + I(ENMO^3)"
)
```

---

imu\_to\_check

*IMU data to check*

---

**Description**

A dataset for demonstrating checks that are applied to IMU data.

**Usage**

```
imu_to_check
```

**Format**

A data frame with 300 rows and 8 variables:

**file\_source\_IMU** The filename of the IMU file

**date\_processed\_IMU** The date the IMU file was processed

**Timestamp** The corresponding time for each row of data

**Gyroscope\_VM\_DegPerS** Gyroscope vector magnitude, in degrees per second

**mean\_abs\_Gyroscope\_x\_DegPerS** Rotation in x axis, degrees per second

**mean\_abs\_Gyroscope\_y\_DegPerS** Rotation in y axis, degrees per second

**mean\_abs\_Gyroscope\_z\_DegPerS** Rotation in z axis, degrees per second

**mean\_magnetometer\_direction** Cardinal direction of magnetometer signal, averaged over one second

---

imu_to_collapse	<i>IMU data to collapse</i>
-----------------	-----------------------------

---

**Description**

A partially-processed IMU dataset ready to be collapsed from raw samples to one-second summaries.

**Usage**

```
imu_to_collapse
```

**Format**

A data frame with 1500 rows and 17 variables:

**Timestamp** The corresponding time for each row of data

**Accelerometer.X** Secondary accelerometer x-axis data, in G

**Accelerometer.Y** Secondary accelerometer y-axis data, in G

**Accelerometer.Z** Secondary accelerometer z-axis data, in G

**Temperature** Temperature of the IMU, in Celcius

**Gyroscope.X** Gyroscope x-axis data, in degrees per second

**Gyroscope.Y** Gyroscope y-axis data, in degrees per second

**Gyroscope.Z** Gyroscope z-axis data, in degrees per second

**Magnetometer.X** Magnetometer x-axis data, in micro-Teslas

**Magnetometer.Y** Magnetometer y-axis data, in micro-Teslas

**Magnetometer.Z** Magnetometer z-axis data, in micro-Teslas  
**file\_source\_IMU** The filename of the IMU file  
**date\_processed\_IMU** The date the IMU file was processed  
**ms** The millisecond value of the timestamp  
**mean\_Accel\_VM** Vector magnitude of the secondary accelerometer signal, in G  
**Gyroscope\_VM\_DegPerS** Gyroscope vector magnitude, in degrees per second  
**Magnetometer\_VM\_MicroT** Vector magnitude of the magnetometer signal, in micro-Teslas

---

plot.TwoRegression      *Create summary plots for TwoRegression objects*

---

### Description

Four plots are generated: a threshold plot for both cut-points, and a model plot for both regression models

### Usage

```
## S3 method for class 'TwoRegression'
plot(
  x = NULL,
  object = NULL,
  sed_cp_activities,
  sed_activities,
  sed_cpVar = NULL,
  activity_var,
  met_var,
  walkrun_activities,
  walkrun_cpVar,
  x_sed = NULL,
  y_sed = NULL,
  x_walkrun = NULL,
  y_walkrun = NULL,
  print = TRUE,
  ...
)
```

### Arguments

x	passed from generic function but not used in the method
object	the TwoRegression object
sed_cp_activities	Character vector. Activities to be included in the process of forming the sedentary classifier

<code>sed_activities</code>	Character vector. Actual sedentary activities
<code>sed_cpVar</code>	character scalar. The name of the variable on which the cut-point is based
<code>activity_var</code>	Character scalar. Name of the variable defining which activity is being performed
<code>met_var</code>	character scalar. The name of the variable containing energy expenditure values, in metabolic equivalents
<code>walkrun_activities</code>	Character vector. Actual ambulatory activities
<code>walkrun_cpVar</code>	character scalar giving the name of the variable on which the walk/run cut-point is based
<code>x_sed</code>	numeric scalar giving x coordinate for label placement in sedentary cut-point plot
<code>y_sed</code>	numeric scalar giving y coordinate for label placement in sedentary cut-point plot
<code>x_walkrun</code>	numeric scalar giving x coordinate for label placement in walk/run cut-point plot
<code>y_walkrun</code>	numeric scalar giving y coordinate for label placement in walk/run cut-point plot
<code>print</code>	logical. Should the plot be arranged in a grid? If false, the panels will be returned in a list of <code>gg/ggplot</code> objects.
<code>...</code>	further arguments passed to plotting calls

**Value**

A two-by-two grid of summary plots

**Examples**

```

data(all_data, package = "TwoRegression")
all_data$PID <-
  rep(
    c("Test1", "Test2"),
    each = ceiling(nrow(all_data) / 2))[seq(nrow(all_data))]

fake_sed <- c("Lying", "Sitting")
fake_lpa <- c("Sweeping", "Dusting")
fake_cwr <- c("Walking", "Running")
fake_ila <- c("Tennis", "Basketball")

fake_activities <- c(fake_sed, fake_lpa, fake_cwr, fake_ila)

all_data$Activity <-
  sample(fake_activities, nrow(all_data), TRUE)

all_data$fake_METs <-
  ifelse(all_data$Activity %in% c(fake_sed, fake_lpa),
    runif(nrow(all_data), 1, 2),
    runif(nrow(all_data), 2.5, 8)
  )

```



```

ex_2rm <- fit_2rm(
  data = all_data,
  activity_var = "Activity",
  sed_cp_activities = c(fake_sed, fake_lpa),
  sed_activities = fake_sed,
  sed_cp_var = "ENMO",
  sed_METs = 1.25,
  walkrun_activities = fake_cwr,
  walkrun_cp_var = "ENMO_CV10s",
  met_var = "fake_METs",
  walkrun_formula = "fake_METs ~ ENMO",
  intermittent_formula = "fake_METs ~ ENMO + I(ENMO^2) + I(ENMO^3)"
)

model_plot_list <- plot(
  object = ex_2rm,
  sed_cp_activities = c(fake_sed, fake_lpa),
  sed_activities = fake_sed,
  sed_cpVar = "ENMO",
  activity_var = "Activity",
  met_var = "fake_METs",
  walkrun_activities = fake_cwr,
  walkrun_cpVar = "ENMO_CV10s",
  print = FALSE
)

print(model_plot_list$sed_cut_point)
print(model_plot_list$walkrun_cut_point)
print(model_plot_list$walkrun_regression)
print(model_plot_list$intermittent_regression)

plot(
  object = ex_2rm,
  sed_cp_activities = c(fake_sed, fake_lpa),
  sed_activities = fake_sed,
  sed_cpVar = "ENMO",
  activity_var = "Activity",
  met_var = "fake_METs",
  walkrun_activities = fake_cwr,
  walkrun_cpVar = "ENMO_CV10s",
  print = TRUE
)

```

---

predict.TwoRegression *Predict metabolic equivalents from a TwoRegression object*

---

### Description

Predict metabolic equivalents from a TwoRegression object

**Usage**

```
## S3 method for class 'TwoRegression'
predict(
  object,
  newdata,
  min_mets = object$sed_METs,
  max_mets = 20,
  warn_high_low = TRUE,
  verbose = FALSE,
  ...
)
```

**Arguments**

<code>object</code>	the <code>TwoRegression</code> object
<code>newdata</code>	the data on which to predict metabolic equivalents (METs)
<code>min_mets</code>	the minimum allowable value for MET predictions. Defaults to the value stored in <code>object\$sed_METs</code>
<code>max_mets</code>	the maximum allowable value for MET predictions. There is no value embedded in object. The default is 20
<code>warn_high_low</code>	logical. Issue warnings about values less than <code>min_mets</code> or greater than <code>max_mets</code> ?
<code>verbose</code>	logical. Print processing updates?
<code>...</code>	further arguments passed to or from other methods

**Value**

A two-column data frame giving the activity classification (sedentary, walk/run, or intermittent activity) and the corresponding metabolic equivalent prediction

**Examples**

```
data(all_data, package = "TwoRegression")
all_data$PID <-
  rep(
    c("Test1", "Test2"),
    each = ceiling(nrow(all_data) / 2))[seq(nrow(all_data))]

train_data <- all_data[all_data$PID != "Test2", ]
test_data <- all_data[all_data$PID == "Test2", ]

fake_sed <- c("Lying", "Sitting")
fake_lpa <- c("Sweeping", "Dusting")
fake_cwr <- c("Walking", "Running")
fake_ila <- c("Tennis", "Basketball")

fake_activities <- c(fake_sed, fake_lpa, fake_cwr, fake_ila)

train_data$Activity <-
```

```

    sample(fake_activities, nrow(train_data), TRUE)

train_data$fake_METs <-
  ifelse(train_data$Activity %in% c(fake_sed, fake_lpa),
    runif(nrow(train_data), 1, 2),
    runif(nrow(train_data), 2.5, 8)
  )

ex_2rm <- fit_2rm(
  data = train_data,
  activity_var = "Activity",
  sed_cp_activities = c(fake_sed, fake_lpa),
  sed_activities = fake_sed,
  sed_cp_var = "ENMO",
  sed_METs = 1.25,
  walkrun_activities = fake_cwr,
  walkrun_cp_var = "ENMO_CV10s",
  met_var = "fake_METs",
  walkrun_formula = "fake_METs ~ ENMO",
  intermittent_formula = "fake_METs ~ ENMO + I(ENMO^2) + I(ENMO^3)"
)

predict(ex_2rm, test_data)

```

---

raw_for_cv	<i>Primary accelerometer data to calculate coefficient of variation per 10-s</i>
------------	--

---

### Description

A partially-processed primary accelerometer dataset ready to calculate the coefficient of variation per 10-s

### Usage

```
raw_for_cv
```

### Format

A data frame with 299 rows and 2 variables:

**Block** A vestigial variable synonymous with row number

**ENMO** Euclidian Norm Minus One, in milli-g

---

raw_to_collapse	<i>Primary accelerometer data to collapse</i>
-----------------	---

---

### Description

A partially-processed primary accelerometer dataset ready to be collapsed from raw samples to one-second summaries.

### Usage

```
raw_to_collapse
```

### Format

A data frame with 24000 rows and 3 variables:

**Accelerometer X** Primary accelerometer x-axis data, in G

**Accelerometer Y** Primary accelerometer y-axis data, in G

**Accelerometer Z** Primary accelerometer z-axis data, in G

---

smooth_2rm	<i>Smooth two-regression estimates over specified periods</i>
------------	---

---

### Description

Smooth two-regression estimates over specified periods

### Usage

```
smooth_2rm(AG, time_var = "Timestamp", unit = "60 sec", verbose = FALSE, ...)
```

### Arguments

AG	data frame of ActiGraph data
time_var	character scalar. Name of the timestamp variable (required to verify that input epoch length is 10 seconds)
unit	the interval to use for smoothing (see <a href="#">floor_date</a> ). Default is "60 sec"
verbose	logical. Print updates to console?
...	currently unused

### Value

Smoothed data, collapsed in the specified intervals

## Examples

```
data(all_data, package = "TwoRegression")

result <- TwoRegression(
  all_data, "Hibbing 2018", gyro_var = "Gyroscope_VM_DegPerS",
  direction_var = "mean_magnetometer_direction",
  site = c("Left Ankle", "Right Ankle"), algorithm = 1:2
)

smooth_2rm(result)
```

---

TwoRegression-Package *Develop and Apply Two-Regression Algorithms*

---

## Description

The TwoRegression package is designed to make working with two-regression algorithms quick, easy, and accurate.

## Details

Originally, the package was designed to house the algorithms created by Hibbing et al. (2018). Since then, support has been added for other algorithms, including Crouter et al. (2006), Crouter et al. (2010), and Crouter et al. (2012). Functionality has also been added to develop and cross-validate new two-regression algorithms. The package RcppRoll has also been invoked to speed up rolling coefficient of variation calculations.

## Associated References

Hibbing PR, LaMunion SR, Kaplan AS, & Crouter SE (2018). Estimating energy expenditure with ActiGraph GT9X Inertial Measurement Unit. *Medicine and Science in Sports and Exercise*. 50(5), 1093-1102. doi: 10.1249/MSS.0000000000001532

Crouter, S. E., Clowers, K. G., & Bassett Jr, D. R. (2006). A novel method for using accelerometer data to predict energy expenditure. *Journal of Applied Physiology*, 100(4), 1324-1331.

Crouter, S. E., Kuffel, E., Haas, J. D., Frongillo, E. A., & Bassett Jr, D. R. (2010). Refined Two-Regression Model for the ActiGraph Accelerometer. *Medicine and Science in Sports and Exercise*, 42(5), 1029.

Crouter, S. E., Horton, M., & Bassett Jr, D. R. (2012). Use of a Two-regression model for estimating energy expenditure in children. *Medicine and Science in Sports and Exercise*, 44(6), 1177.

## Examples

```
## Datasets

data(count_data, package = "TwoRegression")
```

```
data(all_data, package = "TwoRegression")

## Crouter 2006-2012 models

TwoRegression(
  count_data, "Crouter 2006",
  movement_var = "Axis1", time_var = "time"
)

TwoRegression(
  count_data, "Crouter 2010",
  movement_var = "Axis1", time_var = "time"
)

TwoRegression(
  count_data, "Crouter 2012", movement_var = "Axis1",
  time_var = "time", model = "VA", check = FALSE
)

TwoRegression(
  count_data, "Crouter 2012", movement_var = "Vector.Magnitude",
  time_var = "time", model = "VM", check = FALSE
)

## Hibbing 2018 models (can be vectorized)

all_data$ENMO_CV10s <- NULL
all_data$GVM_CV10s <- NULL
all_data$Direction <- NULL

result <- TwoRegression(
  all_data, "Hibbing 2018", gyro_var = "Gyroscope_VM_DegPerS",
  direction_var = "mean_magnetometer_direction",
  site = c("Left Ankle", "Right Ankle"), algorithm = 1:2
)

utils::head(result)
```

# Index

## \* datasets

- all\_data, [2](#)
- count\_data, [3](#)
- imu\_to\_check, [5](#)
- imu\_to\_collapse, [6](#)
- raw\_for\_cv, [11](#)
- raw\_to\_collapse, [12](#)

all\_data, [2](#)

count\_data, [3](#)

fit\_2rm, [3](#)

floor\_date, [12](#)

imu\_to\_check, [5](#)

imu\_to\_collapse, [6](#)

is.TwoRegression (fit\_2rm), [3](#)

plot.TwoRegression, [5, 7](#)

predict.TwoRegression, [5, 9](#)

raw\_for\_cv, [11](#)

raw\_to\_collapse, [12](#)

smooth\_2rm, [12](#)

summary.TwoRegression, [5](#)

TwoRegression-Package, [13](#)