Package: GENEAclassify (via r-universe)

October 16, 2026

Type Package

Title Segmentation and Classification of Accelerometer Data

Version 1.5.6

Date 2024-02-07

Author Chris Campbell [aut], Aimee Gott [aut], Joss Langford [aut], Charles Sweetland [aut], Penny Sweetland [aut], Jia Ying Chua [aut, cre], Activinsights Ltd [cph]

Description Segmentation and classification procedures for data from the 'Activinsights GENEActiv' <https://activinsights.com/technology/geneactiv/> accelerometer that provides the user with a model to guess behaviour from test data where behaviour is missing. Includes a step counting algorithm, a function to create segmented data with custom features and a function to use recursive partitioning provided in the function rpart() of the 'rpart' package to create classification models.

License GPL

LazyData true

Depends R (>= 2.14.0), methods, utils, grDevices, MASS, GENEAread, changepoint, signal, rpart

Suggests waveslim, knitr, rmarkdown

RoxygenNote 7.2.3

VignetteBuilder knitr

NeedsCompilation yes

Maintainer Jia Ying Chua <jiayingc@activinsights.com>

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

Date/Publication 2025-05-24 08:20:31 UTC

RemoteUrl https://github.com/ActiveSense/GENEAclassify

RemoteRef HEAD

RemoteSha 9005fc310563bc54f041b7da1fdaa7f4b083d1f0

Contents

GENEAclassify-package 2
classifyGENEA 2
createGENEAmodel 5
dataImport 6
find_peaks 8
getGENEAsegments 8
segmentation 12
stepCounter 16
TrainingData 17
trainingFit 18

Index **19**

`GENEAclassify-package` *Classification of accelorometer data*

Description

This package provides tools to perform the segmentation and classification of GENEActiv accelorometer data. The high frequency time series data is split into segments based on activity changepoints. An **rpart** fit is trained against known activities at each segment. This fit can then then be used to guess behaviours from test data when activity at each time point has not been reported. This allows detailed behaviour profiles to be created for the wearer.

`classifyGENEA` *Classify Data into Categories defined in an rpart GENEA fit*

Description

Perform classification on segmented GENEActiv bin data using an rpart GENEA training fit.

Usage

```
classifyGENEA(
  testfile,
  start = NULL,
  end = NULL,
  Use.Timestamps = FALSE,
  radians = FALSE,
  mmap.load = (.Machine$sizeof.pointer >= 8),
  trainingfit = trainingFit,
  newdata,
  allprobs = FALSE,
  setinf = 100,
```

```
  outputname = "_classified",
  outputdir = "GENEAclassification",
  datacols = "default",
 changepoint = c("UpDownDegrees", "TempFreq", "UpDownFreq", "UpDownMean", "UpDownVar",
    "UpDownMeanVar", "DegreesMean", "DegreesVar", "DegreesMeanVar",
    "UpDownMeanVarDegreesMeanVar", "UpDownMeanVarMagMeanVar"),
  penalty = "Manual",
  pen.value1 = 40,
  pen.value2 = 400,
  intervalseconds = 30,
  mininterval = 5,
  samplefreq = 100,
  filterorder = 2,
  boundaries = c(0.5, 5),
  Rp = 3,
  plot.it = FALSE,
  hysteresis = 0.1,
  stft_win = 10,
  plot.seg = FALSE,
  plot.seg.outputfile = "Changepoint",
  verbose = TRUE,
  ...
)
```

Arguments

testfile	character string stating path to a GENEActiv bin file, or a folder containing GENEActiv bin files.
start	Where to start reading observations.
end	Where to end reading observations.
Use.Timestamps	To use timestamps as the start and end time values this has to be set to TRUE. (Default FALSE)
radians	calculate degrees rotation in radians.
mmap.load	Default is (.Machine$sizeof.pointer >= 8). see mmap for more details
trainingfit	a GENEA rpart object created by createGENEAmodel that gives the decision tree that was fitted from the training data. These are the parameters used to predict the new data.
newdata	a new data frame that is to be classified (provide instead of testfile). The data must contain the features named in the trainingfit.
allprobs	single logical should all estimated probabilities be reported rather than probability of selected class (default FALSE).
setinf	single numeric an arbitrary value to replace Inf in calculated columns or NA to ignore Inf values. (default 100). -setinf is used to replace -Inf. Alternatively, use setinf NULL to leave Inf as is.
outputname	file name root (excluding extension) for saving the classification output (default "classified").

outputdir	absolute or relative path to directory in which artifacts (plot and changes files) should be created or NULL (default "GENEAclassification").
datacols	a vector constructed 'column.summary' or 'default'. See segmentation for details.
changepoint	defines the change point analysis to use. UpDownDegrees performs the change point analysis on the variance of arm elevation and wrist rotation. TempFreq performs a change point on the variance in the temeprature and Frequency (Typically better for sleep behaviours)
penalty	single characgter, the penalty to use for changepoint detection. default ("SIC")
pen.value1	Value of the type 1 error required when penalty is "Asymptotic".
pen.value2	Default set as NULL and so equals pen.value1 if no input.
intervalseconds	An integer number of seconds between 5 and 30 during which at most one changepoint may occur.
mininterval	single numeric that defines the smallest changepoint initially found. Passed to cpt.var as the variable minseglen
samplefreq	The sampling frequency of the data, in hertz, when calculating the step number. (default 100)
filterorder	The order of the filter applied with respect to the cheby options.
boundaries	to passed to the filter in the step counting algorithm.
Rp	the decibel level that the cheby filter takes. see cheby1
plot.it	(logical) Creates a plot showing the zero crossings counted by the step counting algorithm#' @param Centre Centres the xz signal about 0 when set to True.
hysteresis	The hysteresis applied after zero crossing. (default 100mg)
stft_win	numeric for the window to calculate the frequency of an event using the stft function.
plot.seg	(logical) Creates a plot displaying the changepoint locations
plot.seg.outputfile	The name of the png file created that shows the change points on a positionals plots.
verbose	single logical should additional progress reporting be printed at the console (default TRUE).
...	other arguments to be passed to dataImport, segmentation and other functions

Details

This function will apply the rules determined by the rpart GENEA decision tree passed to argument trainingfit to the columns of newdata to classify into classes (view using "levels").

Value

The function will return the data frame that was provided as newdata with additional columns.

1. Class, a factor indicating that the predicted category of the segment

2. p.Class, estimated probability that the prediction is correct

Alternatively, by setting argument allprobs to TRUE, a column constructed 'p.level' containing the estimated probability of each possible class will be returned instead.

Examples

```
## segData <- read.csv(system.file(package = "GENEAclassify",
##       "testdata", "trainingData9.csv"))
## The training fit can be created by provided the file path to the training data
## in the function getTrainingData - see the help file for more details
## Uses the fitted decision tree to predict the segmented data
## class9 <- classifyGENEA(testfile = "trainingData9.csv",
##                         newdata = segData,
##                         trainingfit = trainingFit)
## head(class9)
## table(class9$Class)
```

---

createGENEAmodel *Create training data decision tree model*

Description

From data frame create a decision tree that can be used for classifying data into specified categories. The data frame may optionally contain a reserved column Source, specifying the provenance of the record. The data frame must contain a column, by default named Activity, specifing the classes into which the model fit should be classified.

Usage

```
createGENEAmodel(
  data,
  outputtree = NULL,
 features = c("Segment.Duration", "Principal.Frequency.mad", "UpDown.sd", "Degrees.sd"),
  category = "Activity",
  plot = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

data	data frame containing segmented GENEActiv bin data.
outputtree	name of the png file that shows the classification tree plot.
features	character vector naming independent variables to use in classification. Alternatively, a numeric vector specifying the variables to pass to the classification or NULL, in which case all variables are used in the order of the supplied training dataset. Note that including large numbers of variables (>7) may result in long run times.

category	single character naming the dependent variable to use (default 'Activity').
plot	a logical value indicating whether a plot of the classification tree should be plotted. The default is TRUE.
verbose	single logical should additional progress reporting be printed at the console? (default TRUE)
...	other arguments for rpart

Details

The function will create an rpart classification tree for the training data based upon the parameters passed to features. The model created, an GENEA rpart object can be used within the function "classifyGENEA" to classify GENEA bin files.

Value

A GENEA rpart fit.

See Also

The returned object can be interrogated with features, the variables used in defining the model, and "levels", the response categories predicted by the model.

Examples

```
## dataPath <- file.path(system.file(package = "GENEAclassify"),
##                                   "testdata",
##                                   "trainingData9.csv")
##
## t1 <- read.csv(file = dataPath)
##
## f1 <- createGENEAmodel(data = t1,
##                        features = c("Degrees.var",
##                                     "UpDown.mad",
##                                     "Magnitude.mean"),
##                        category = "Activity")
##
## class(f1)
## levels(f1)
## features(f1)
## plot(f1)
## text(f1)
```

---

dataImport *Data import function*

Description

Loads the data into R and creates format required for segmentation.

Usage

```
dataImport(
  binfile,
  downsample = 100,
  start = NULL,
  end = NULL,
  Use.Timestamps = FALSE,
  radians = FALSE,
  keep_raw_data = TRUE,
  mmap.load = (.Machine$sizeof.pointer >= 8),
  ...
)
```

Arguments

binfile	File path to binary data to be segmented.
downsample	Rate to downsample the data, defaults to every 100th observation. For no downsampling set NULL.
start	Where to start reading observations.
end	Where to end reading observations.
Use.Timestamps	To use timestamps as the startand end time values this has to be set to TRUE. (Default FALSE)
radians	calculate degrees rotation in radians.
keep_raw_data	Keep the raw data for calculating steps using stepcounter.
mmap.load	Default is (.Machine$sizeof.pointer >= 8). see mmap for more details
...	additional arguments passed through.

Details

Reads in the binary data file and extracts the information required for the segmentation procedure.

Value

Returns a list containing a matrix of the data including the x, y and z axis data, vectors of the up down (elevation) and degrees (rotation), a vector of time stamps, a vector of vector magnitudes and the serial number of the device.

Examples

```
##   segData = dataImport(bindata = file.path(system.file(package = "GENEAread"),
##                                                        "binfile",
##                                                        "TESTfile.bin"))
##
## segData1 = dataImport(AccData)
## names(segData)
```

`find_peaks` *Find_Peaks*

Description

Peak detect function

Usage

```
find_peaks(x, m = 3)
```

Arguments

x	timeseries signal
m	The number of points either side of the peak to required to be a peak.

Details

Finds the peaks and valleys within the signal passed to the function.

`getGENEAsegments` *import and segment one or more bin files.*

Description

Import and summarize GENEActiv bin data for manual classification.

Usage

```
getGENEAsegments(
  testfile,
  start = NULL,
  end = NULL,
  Use.Timestamps = FALSE,
  radians = FALSE,
  keep_raw_data = TRUE,
  mmap.load = (.Machine$sizeof.pointer >= 8),
  outputtoken = "_segmented",
  outputdir = "GENEAclassification",
  datacols = "default",
  decimalplaces = "default",
  filterWave = FALSE,
  filtername = "haar",
  j = 8,
 changepoint = c("UpDownDegrees", "TempFreq", "UpDownFreq", "UpDownMean", "UpDownVar",
```

```
    "UpDownMeanVar", "DegreesMean", "DegreesVar", "DegreesMeanVar",
    "UpDownMeanVarDegreesMeanVar", "UpDownMeanVarMagMeanVar"),
  penalty = "Manual",
  pen.value1 = 40,
  pen.value2 = 400,
  intervalseconds = 30,
  mininterval = 5,
  samplefreq = 100,
  filterorder = 2,
  boundaries = c(0.5, 5),
  Rp = 3,
  plot.it = FALSE,
  hysteresis = 0.1,
  stft_win = 10,
  plot_changepoints = FALSE,
  plot_changepoints_outputfile = "Changepoint",
  verbose = FALSE,
  ...
)
```

Arguments

testfile	character vector stating path to a GENEActiv bin file, or a folder containing GENEActiv bin files.
start	Where to start reading observations.
end	Where to end reading observations.
Use.Timestamps	To use timestamps as the start and end time values this has to be set to TRUE. (Default FALSE)
radians	calculate degrees rotation in radians.
keep_raw_data	Keep the raw data for calculating steps using stepcounter.
mmap.load	Default is (.Machine$sizeof.pointer >= 8). see mmap for more details
outputtoken	single character string to be appended to the file name for saving the segmenation output (default '_segmentated').
outputdir	The absolute or relative path to directory in which artifacts (plot and changes files) should be created, or NULL (default "GENEAclassification").
datacols	a vector constructed 'column.summary' or 'default'. See segmentation for details.
decimalplaces	named numeric vector of decimal places with which to round summary columns. NULL returns unrounded values. The length 1 character vector 'default' applies default roundings: • Start.Time = 0, • Degrees.mean = 3, • Degrees.median = 3, • Degrees.var = 3, • Degrees.sd = 3,

- Degrees.mad = 3,
- Magnitude.mean = 3,
- UpDown.mean = 3,
- UpDown.median = 3,
- UpDown.var = 3
- UpDown.sd = 3,
- UpDown.mad = 3,
- Principal.Frequency.median = 3,
- Principal.Frequency.mad = 3,
- Principal.Frequency.ratio = 3,
- Principal.Frequency.sumdiff = 3,
- Principal.Frequency.meandiff = 3,
- Principal.Frequency.abssumdiff = 3,
- Principal.Frequency.sddiff = 3,
- Light.mean = 0,
- Light.max = 0,
- Temp.mean = 1,
- Temp.sumdiff = 3
- Temp.meandiff = 3
- Temp.abssumdiff = 3
- Temp.sddiff = 3
- Step.count = 0
- Step.sd = 1
- Step.mean = 0
- Step.GENEAamplitude = 3
- Step.GENEAwavelength = 3
- Step.GENEAdistance = 3

This can be changed by using a named list. e.g decimalplaces = c(Start.Time = 2, Degrees.mean = 4).

`filterWave` single logical, should a smoothing filter from `wave.filter` be applied? (default FALSE).

`filtername` single character, the name of the wavelet to use for smoothing when filter is TRUE. (default "haar") Passed to `link[waveslim]{wave.filter}`.

`j` single numeric, the level to which to smooth. Passed to `link[waveslim]{wave.filter}` (default 8).

`changepoint` defines the change point analysis to use. UpDownDegrees performs the change point analysis on the variance of arm elevation and wrist rotation. TempFreq performs a change point on the variance in the temperature and frequency (Typically better for sleep behaviours).

`penalty` single character, the penalty to use for changepoint detection. default ("SIC").

`pen.value1` Value of the type 1 error required when penalty is "Asymptotic".

`pen.value2` Default set as NULL and so equals pen.value1 if no input.

intervalseconds
: An integer number of seconds between 5 and 30 during which at most one changepoint may occur.

mininterval
: single numeric that defines the smallest changepoint initially found. Passed to cpt.var as the variable minseglen

samplefreq
: The sampling frequency of the data, in hertz, when calculating the step number. (default 100).

filterorder
: The order of the filter applied with respect to the butter or cheby options. See cheby1 or butter.

boundaries
: to pass to the filter in the step counting algorithm.

Rp
: the decibel level that the cheby filter takes. See cheby1.

plot.it
: single logical, Creates a plot showing the zero crossings counted by the step counting algorithm#' @param Centre Centres the xz signal about 0 when set to True.

hysteresis
: The hysteresis applied after zero crossing. (default 100mg)

stft_win
: numeric for the window to calculate the frequency of an event using the stft function.

plot_changepoints
: single logical, Creates a plot displaying the changepoint locations.

plot_changepoints_outputfile
: The name of the png file created that shows the change points on a positionals plots.

verbose
: single logical should additional progress reporting be printed at the console? (default TRUE).

...
: other arguments to be passed to dataImport, segmentation and other functions with these functions.

Value

segmented data are returned

See Also

The returned object can be interrogated with head.

Examples

```
## testfile = file.path(system.file(package = "GENEAread"),
##                                  "binfile",
##                                  "TESTfile.bin")
##
## segData <- getGENEAsegments(testfile = testfile,
##                             outputdir = file.path(tempdir(), "GENEAclassification"),
##                             changepoint = "UpDownDegrees",
##                             pen.value1 = 1,
##                             pen.value2 = 1)
## head(segData)
## list.files(file.path(tempdir(), "GENEAclassification"))
```

segmentation *Perform Segmentation on GENEActiv Accelerometer Data*

Description

Perform segmentation of Activinsights accelerometer data. Data are smoothed by the second, or by 10 data points, whichever number of records is greater.

Filtering is performed by tools from **waveslim**. Options are passed to `wavelet.filter`.

Usage

```
segmentation(
  data,
  outputfile = "detectedChanges",
  outputdir = "GENEAclassification",
  datacols = "default",
  decimalplaces = "default",
  filterWave = FALSE,
  filtername = "haar",
  j = 8,
 changepoint = c("UpDownDegrees", "TempFreq", "UpDownFreq", "UpDownMean", "UpDownVar",
    "UpDownMeanVar", "DegreesMean", "DegreesVar", "DegreesMeanVar",
    "UpDownMeanVarDegreesMeanVar", "UpDownMeanVarMagMeanVar", "RadiansMean",
    "RadiansVar", "RadiansMeanVar", "UpDownMeanDegreesVar"),
  penalty = "Manual",
  pen.value1 = 40,
  pen.value2 = 400,
  intervalseconds = 30,
  mininterval = 5,
  samplefreq = 100,
  filterorder = 2,
  boundaries = c(0.5, 5),
  Rp = 3,
  plot.it = FALSE,
  hysteresis = 0.1,
  stft_win = 10,
  verbose = FALSE,
  verbose_timer = FALSE,
  ...
)
```

Arguments

data	the GENEActiv bin object to be segmented which should be the output of the `dataImport` function.
outputfile	single character, file name for saving the segmentation output as CSV (and if plot.it is TRUE, corresponding plot PNG). If NULL, create no files.

`outputdir` single character, the absolute or relative path to directory in which plot and changes files) should be created, or NULL (default "GENEAclassification"). Ignored if outputfile is NULL.

`datacols` character vector constructed 'column.summary'. This object specifies the data and summary to output for the classification. The first part of each element must name column in the GENEAbin datasets specified by filePath. Derived columns may also be selected:

- Step (zero-crossing step counter method),
- Principal.Frequency.

The second should be the name of a function that evaluates to lenth one. The functions must contain only alphabetical characters (no numbers, underscores or punctuation). The default matrix, specified using the length 1 character vector `'default'` is:

- UpDown.mean
- UpDown.sd
- UpDown.mad
- Degrees.mean
- Degrees.var
- Degrees.sd
- Light.mean
- Light.max
- Temp.mean
- Temp.sumdiff
- Temp.meandiff
- Temp.abssumdiff
- Temp.sddiff
- Magnitude.mean
- Step.GENEAcount
- Step.sd
- Step.mean
- Step.GENEAamplitude
- Step.GENEAwavelength
- Principal.Frequency.median
- Principal.Frequency.mad
- Principal.Frequency.ratio
- Principal.Frequency.sumdiff
- Principal.Frequency.meandiff
- Principal.Frequency.abssumdiff
- Principal.Frequency.sddiff

`decimalplaces` named numeric vector of decimal places with which to round summary columns. `NULL` returns unrounded values. The length 1 character vector 'default' applies default roundings:

- Start.Time = 0,

- Degrees.mean = 3,
- Degrees.median = 3,
- Degrees.var = 3,
- Degrees.sd = 3,
- Degrees.mad = 3,
- Magnitude.mean = 3,
- UpDown.mean = 3,
- UpDown.median = 3,
- UpDown.var = 3
- UpDown.sd = 3,
- UpDown.mad = 3,
- Principal.Frequency.median = 3,
- Principal.Frequency.mad = 3,
- Principal.Frequency.ratio = 3,
- Principal.Frequency.sumdiff = 3,
- Principal.Frequency.meandiff = 3,
- Principal.Frequency.abssumdiff = 3,
- Principal.Frequency.sddiff = 3,
- Light.mean = 0,
- Light.max = 0,
- Temp.mean = 1,
- Temp.sumdiff = 3
- Temp.meandiff = 3
- Temp.abssumdiff = 3
- Temp.sddiff = 3
- Step.GENEAcount = 0
- Step.sd = 1
- Step.mean = 0

`filterWave` single logical, should a smoothing filter from `wave.filter` be applied? (default FALSE).

`filtername` single character, the name of the wavelet to use for smoothing when filter is TRUE. (default "haar") Passed to `link[waveslim]{wave.filter}`.

`j` single numeric, the level to which to smooth. Passed to `link[waveslim]{wave.filter}` (default 8).

`changepoint` defines the change point analysis to use. UpDownDegrees performs the change point analysis on the variance of arm elevation and wrist rotation. TempFreq performs a change point on the variance in the temeprature and frequency (Typically better for sleep behaviours).

`penalty` single characgter, the penalty to use for changepoint detection. default ("SIC").

`pen.value1` Value of the type 1 error required when penalty is "Asymptotic".

`pen.value2` Default set as NULL and so equals pen.value1 if no input.

`intervalseconds`
: An integer number of seconds between 5 and 30 during which at most one changepoint may occur.

`mininterval`
: single numeric that defines the smallest changepoint initially found. Passed to `cpt.var` as the variable minseglen

`samplefreq`
: The sampling frequency of the data, in hertz, when calculating the step number. (default 100).

`filterorder`
: The order of the filter applied with respect to the butter or cheby options if stepCounter is used. The order of the moving average filter if step counter 2 is used. See `cheby1` or `butter`.

`boundaries`
: to pass to the filter in the step counting algorithm.

`Rp`
: the decibel level that the cheby filter takes. See `cheby1`.

`plot.it`
: single logical, Creates a plot showing the zero crossings counted by the step counting algorithm#' @param Centre Centres the xz signal about 0 when set to True.

`hysteresis`
: The hysteresis applied after zero crossing. (default 100mg)

`stft_win`
: numeric for the window to calculate the frequency of an event using the `stft` function.

`verbose`
: single logical to print additional progress reporting (default FALSE).

`verbose_timer`
: single logical tp print additional progress reporting on time for each section of the function (default FALSE).

`...`
: other arguments to be passed to `dataImport`, `segmentation` and other functions with these functions.

Details

Performs the segmentation procedure on the provided elevation data. Optionally a wavelet filter is first applied to smooth the data. The number of changes occuring in a given number of seconds may be controlled using the intervalseconds argument. Changes will be removed based on which segments are the closest match in terms of variance. A series of features for each of the segments will then be calculated and returned as a csv file.

Value

The segment data are returned invisibly. This data frame has columns:

- Serial.Number
- Start.Time
- Segment.Start.Time
- Segment.Duration
- UpDown.median
- UpDown.var
- Degrees.median
- Degrees.mad

In addition, the requested columns are included. Optionally, as a side effect a csv file is returned listing all of the segments found in the data along with a variety of features for that segment. Optionally a png file plotting the data and the detected changes can also be produced.

Examples

```
### Load the data to segment keeping only the first quarter of the data
## library(GENEAread)
## testfile = file.path(system.file(package = "GENEAread"),
##                                  "binfile",
##                                  "TESTfile.bin")
## segData <- dataImport(binfile = testfile,
##     downsample = 100, start = 0, end = 0.25)
## head(segData)
### Run loaded data through segmentation function
## segment <- segmentation(data = segData, outputfile = NULL)
## head(segment)
## segment2 <- segmentation(data = segData, outputfile = NULL,
##     datacols = "Degrees.skew")
## head(segment2)
```

`stepCounter` *Step Counter*

Description

Function to calculate the number and variance of the steps in the data.

Usage

```
stepCounter(
  AccData,
  samplefreq = 100,
  filterorder = 2,
  boundaries = c(0.5, 5),
  Rp = 3,
  plot.it = FALSE,
  hysteresis = 0.05,
  verbose = verbose,
  fun = c("GENEAcount", "mean", "sd", "mad")
)
```

Arguments

`AccData`	The data to use for calculating the steps. This should either an AccData object or a vector.
`samplefreq`	The sampling frequency of the data, in hertz, when calculating the step number (default 100).

filterorder	single integer, order of the Chebyshev bandpass filter, passed to argument n of cheby1.
boundaries	length 2 numeric vector specifying lower and upper bounds of Chebychev filter (default c(0.5, 5) Hz), passed to argument W of butter or cheby1.
Rp	the decibel level that the cheby filter takes, see cheby1.
plot.it	single logical create plot of data and zero crossing points (default FALSE).
hysteresis	The hysteresis applied after zero crossing. (default 100mg)
verbose	single logical should additional progress reporting be printed at the console? (default FALSE).
fun	character vector naming functions by which to summarize steps. "count" is an internally implemented summarizing function that returns step count.

Value

Returns a vector with length fun.

Examples

```
d1 <- sin(seq(0.1, 100, 0.1))/2 + rnorm(1000)/10 + 1
Steps4 = stepCounter(d1)
length(Steps4)
mean(Steps4)
sd(Steps4)
plot(Steps4)
```

TrainingData *Example Training Data set*

Description

A manually classified training data set provided with the package.

Format

Manually classified segmented data that can be used to build a classification model.

Source

Manually classified by Actvinsights.

See Also

"levels" features

Examples

```
data(TrainingData)
class(TrainingData)
```

trainingFit *Example classification tree*

Description

rpart object declaring the decision tree for the classification of GENEActiv bin data into typical groups.

Format

An `rpart` object with class `GENEA` containing the decision tree information required for prediction.

Source

Output of `createGENEAmodel` on experimental training data.

See Also

"levels" features

Examples

```
data(trainingFit)
class(trainingFit)
levels(trainingFit)
features(trainingFit)
plot(trainingFit)
text(trainingFit, cex = 0.5)
```

Index

∗ **Internal**
    find_peaks, 8
∗ **datasets**
    TrainingData, 17
    trainingFit, 18
∗ **package**
    GENEAclassify-package, 2

butter, *11*, *15*, *17*

cheby1, *4*, *11*, *15*, *17*
classifyGENEA, 2, *6*
cpt.var, *4*, *11*, *15*
createGENEAmodel, *3*, 5

dataImport, *4*, 6, *11, 12*, *15*

features, *3*, *17, 18*
find_peaks, 8

GENEAclassify-package, 2
getGENEAsegments, 8

head, *11*

levels, *4*, *6*, *17, 18*

mmap, *3*, *9*

segmentation, *4*, *9*, *11*, 12, *15*
stepCounter, 16
stft, *4*, *11*, *15*

TrainingData, 17
trainingFit, 18

wave.filter, *10*, *14*
wavelet.filter, *12*