

Package ‘trackeR’

January 12, 2017

Version 0.0.5

Date 2017-01-11

Title Infrastructure for Running and Cycling Data from GPS-Enabled Tracking Devices

Description The aim of this package is to provide infrastructure for handling running and cycling data from GPS-enabled tracking devices. After extraction and appropriate manipulation of the training or competition attributes, the data are placed into session-based and unit-aware data objects of class `trackeRdata` (S3 class). The information in the resulting data objects can then be visualised, summarised, and analysed through corresponding flexible and extensible methods.

Depends R (>= 2.10.0), zoo, ggplot2

Imports parallel, XML, RSQLite, jsonlite, raster, scam, fda, ggmap, leaflet, gridExtra, gtable, colorspace

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

License GPL-2 | GPL-3

URL <https://github.com/hfrick/trackeR>

BugReports <https://github.com/hfrick/trackeR/issues>

RoxygenNote 5.0.1

NeedsCompilation no

Author Hannah Frick [aut, cre],
Ioannis Kosmidis [aut]

Maintainer Hannah Frick <h.frick@ucl.ac.uk>

Repository CRAN

Date/Publication 2017-01-12 15:42:44

R topics documented:

append	3
append.conProfile	4

append.trackeRdata	4
c2d	5
changeUnits	5
changeUnits.conProfile	6
changeUnits.distrProfile	6
changeUnits.trackeRdata	7
changeUnits.trackeRdataSummary	7
changeUnits.trackeRWprime	8
concentrationProfile	8
conversions	9
decreasingSmoother	13
distance2speed	13
distributionProfile	14
fortify.conProfile	15
fortify.distrProfile	15
fortify.trackeRdata	16
fortify.trackeRdataSummary	16
fortify.trackeRWprime	17
funPCA	17
GC2trackeRdata	18
generateBaseUnits	19
generateVariableNames	19
getOperations	20
getOperations.conProfile	20
getOperations.distrProfile	21
getOperations.trackeRdata	21
getUnits	22
getUnits.conProfile	22
getUnits.distrProfile	23
getUnits.trackeRdata	23
getUnits.trackeRdataSummary	24
getUnits.trackeRWprime	24
imputeSpeeds	25
leafletRoute	26
nsessions	26
plot.conProfile	27
plot.distrProfile	27
plot.trackeRdata	28
plot.trackeRdataSummary	29
plot.trackeRdataZones	30
plot.trackeRfzca	31
plot.trackeRWprime	32
plotRoute	32
print.trackeRdataSummary	33
profile2fd	34
readContainer	35
readDirectory	36
readX	38

restingPeriods	39
run	40
runs	40
sanityChecks	40
scaled	41
scaled.distrProfile	41
smoother	42
smoother.conProfile	42
smoother.distrProfile	43
smoother.trackeRdata	43
smootherControl.distrProfile	44
smootherControl.trackeRdata	45
speed2distance	46
summary.trackeRdata	46
threshold	47
timeAboveThreshold	48
timeline	48
trackeR	49
trackeRdata	50
Wexp	52
Wprime	53
zones	54
Index	56

append

Generic function for appending data to existing files.

Description

Generic function for appending data to existing files.

Usage

```
append(object, file, ...)
```

Arguments

object	The object to be appended.
file	The file to which object is to be appended.
...	Arguments to be passed to methods.

append.conProfile *Append training profiles.*

Description

Generic function for appending training distribution or concentration profiles to existing files.

Usage

```
## S3 method for class 'conProfile'  
append(object, file, ...)
```

```
## S3 method for class 'distrProfile'  
append(object, file, ...)
```

Arguments

object	The object to be appended.
file	The file to which object is to be appended.
...	Currently not used.

append.trackeRdata *Append training sessions to existing file.*

Description

Append training sessions to existing file.

Usage

```
## S3 method for class 'trackeRdata'  
append(object, file, ...)
```

Arguments

object	The object to be appended.
file	The file to which object is to be appended.
...	Currently not used.

c2d	<i>Transform concentration profile to distribution profile.</i>
-----	---

Description

Transform concentration profile to distribution profile.

Usage

```
c2d(cp)
```

Arguments

cp	Single concentration profile as a zoo object.
----	---

changeUnits	<i>Generic function for changing the units of measurement.</i>
-------------	--

Description

Generic function for changing the units of measurement.

Usage

```
changeUnits(object, variable, unit, ...)
```

Arguments

object	The object of which the units of measurement are changed.
variable	The variable of which the units of measurement are changed.
unit	The unit of measurement to which is changed.
...	Arguments to be passed to methods.

`changeUnits.conProfile`*Change the units of the variables in an conProfile object.*

Description

Change the units of the variables in an conProfile object.

Usage

```
## S3 method for class 'conProfile'  
changeUnits(object, variable, unit, ...)
```

Arguments

<code>object</code>	An object of class conProfile as returned by concentrationProfile .
<code>variable</code>	A vector of variables to be changed.
<code>unit</code>	A vector with the units, corresponding to variable.
<code>...</code>	Currently not used.

`changeUnits.distrProfile`*Change the units of the variables in an distrProfile object.*

Description

Change the units of the variables in an distrProfile object.

Usage

```
## S3 method for class 'distrProfile'  
changeUnits(object, variable, unit, ...)
```

Arguments

<code>object</code>	An object of class distrProfile as returned by distributionProfile .
<code>variable</code>	A vector of variables to be changed.
<code>unit</code>	A vector with the units, corresponding to variable.
<code>...</code>	Currently not used.

`changeUnits.trackeRdata`*Change the units of the variables in an trackeRdata object.*

Description

Change the units of the variables in an trackeRdata object.

Usage

```
## S3 method for class 'trackeRdata'  
changeUnits(object, variable, unit, ...)
```

Arguments

<code>object</code>	An object of class trackeRdata .
<code>variable</code>	A vector of variables to be changed.
<code>unit</code>	A vector with the units, corresponding to variable.
<code>...</code>	Currently not used.

`changeUnits.trackeRdataSummary`*Change the units of the variables in an trackeRdataSummary object.*

Description

Change the units of the variables in an trackeRdataSummary object.

Usage

```
## S3 method for class 'trackeRdataSummary'  
changeUnits(object, variable, unit, ...)
```

Arguments

<code>object</code>	An object of class <code>trackeRdataSummary</code> .
<code>variable</code>	A vector of variables to be changed. Note, these are expected to be concepts like "speed" rather than variable names like "avgSpeed" or "avgSpeedMoving".
<code>unit</code>	A vector with the units, corresponding to variable.
<code>...</code>	Currently not used.

changeUnits.trackeRWprime

Change the units of the variables in an trackeRdata object.

Description

Change the units of the variables in an trackeRdata object.

Usage

```
## S3 method for class 'trackeRWprime'  
changeUnits(object, variable, unit, ...)
```

Arguments

object	An object of class trackeRdata .
variable	A vector of variables to be changed.
unit	A vector with the units, corresponding to variable.
...	Currently not used.

concentrationProfile *Generate training concentration profiles.*

Description

Generate training concentration profiles.

Usage

```
concentrationProfile(object, session = NULL, what = c("speed",  
  "heart.rate"), ...)
```

Arguments

object	An object of class distrProfile as returned by distributionProfile .
session	A numeric vector of the sessions to be used, defaults to all sessions.
what	The variables for which the concentration profiles should be generated.
...	Currently not used.

Value

An object of class [conProfile](#).

References

Kosmidis, I., and Passfield, L. (2015). Linking the Performance of Endurance Runners to Training and Physiological Effects via Multi-Resolution Elastic Net. *ArXiv e-print* arXiv:1506.01388.

Examples

```
data("run", package = "trackeR")
dProfile <- distributionProfile(run, what = "speed", grid = seq(0, 12.5, by = 0.05))
cProfile <- concentrationProfile(dProfile)
plot(cProfile, smooth = FALSE)
plot(cProfile)
```

conversions

Auxiliary conversion functions.

Description

Conversion functions for distance, duration, speed, pace, power, and cadence.

Usage

m2km(variable)

km2m(variable)

m2ft(variable)

ft2m(variable)

m2mi(variable)

mi2m(variable)

km2ft(variable)

ft2km(variable)

km2mi(variable)

mi2km(variable)

ft2mi(variable)

mi2ft(variable)

m2m(variable)

km2km(variable)
ft2ft(variable)
mi2mi(variable)
s2min(variable)
min2s(variable)
s2h(variable)
h2s(variable)
min2h(variable)
h2min(variable)
s2s(variable)
min2min(variable)
h2h(variable)
m_per_s2km_per_h(variable)
km_per_h2m_per_s(variable)
m_per_s2ft_per_min(variable)
ft_per_min2m_per_s(variable)
m_per_s2ft_per_s(variable)
ft_per_s2m_per_s(variable)
m_per_s2mi_per_h(variable)
mi_per_h2m_per_s(variable)
m_per_s2km_per_min(variable)
km_per_min2m_per_s(variable)
m_per_s2mi_per_min(variable)
mi_per_min2m_per_s(variable)

km_per_h2ft_per_min(variable)
ft_per_min2km_per_h(variable)
km_per_h2ft_per_s(variable)
ft_per_s2km_per_h(variable)
km_per_h2mi_per_h(variable)
mi_per_h2km_per_h(variable)
km_per_h2km_per_min(variable)
km_per_min2km_per_h(variable)
km_per_h2mi_per_min(variable)
mi_per_min2km_per_h(variable)
ft_per_min2ft_per_s(variable)
ft_per_s2ft_per_min(variable)
ft_per_min2mi_per_h(variable)
mi_per_h2ft_per_min(variable)
ft_per_min2km_per_min(variable)
km_per_min2ft_per_min(variable)
ft_per_min2mi_per_min(variable)
mi_per_min2ft_per_min(variable)
ft_per_s2mi_per_h(variable)
mi_per_h2ft_per_s(variable)
ft_per_s2km_per_min(variable)
km_per_min2ft_per_s(variable)
ft_per_s2mi_per_min(variable)
mi_per_min2ft_per_s(variable)

mi_per_h2km_per_min(variable)
km_per_min2mi_per_h(variable)
mi_per_h2mi_per_min(variable)
mi_per_min2mi_per_h(variable)
km_per_min2mi_per_min(variable)
mi_per_min2km_per_min(variable)
m_per_s2m_per_s(variable)
km_per_h2km_per_h(variable)
ft_per_min2ft_per_min(variable)
ft_per_s2ft_per_s(variable)
mi_per_h2mi_per_h(variable)
km_per_min2km_per_min(variable)
mi_per_min2mi_per_min(variable)
s_per_m2min_per_km(variable)
min_per_km2s_per_m(variable)
s_per_m2min_per_mi(variable)
min_per_mi2s_per_m(variable)
min_per_km2min_per_mi(variable)
min_per_mi2min_per_km(variable)
s_per_m2s_per_m(variable)
min_per_km2min_per_km(variable)
min_per_mi2min_per_mi(variable)
W2kW(variable)
kW2W(variable)

```

W2W(variable)
kW2kW(variable)
steps_per_min2steps_per_min(variable)
rev_per_min2rev_per_min(variable)

```

Arguments

variable Variable to be converted.

decreasingSmoother *Smooth a decreasing function.*

Description

This smoother ensures a positive response (Poisson) that is a monotone decreasing function of x .

Usage

```
decreasingSmoother(x, y, k = 30, len = NULL, sp = NULL, fam = "poisson")
```

Arguments

x	The regressor passed on to the formula argument of scam .
y	The response passed on to the formula argument of scam .
k	Number of knots.
len	If NULL, the default, x is used for prediction. Otherwise, prediction is done over the range of x with len equidistant points.
sp	A vector of smoothing parameters passed on to scam .
fam	A family object passed on to the family argument of scam .

distance2speed *Convert distance to speed.*

Description

Convert distance to speed.

Usage

```
distance2speed(distance, time, timeunit)
```

Arguments

distance	Distance in meters.
time	Time.
timeunit	Time unit in speed, e.g., "hours" for speed in *_per_h.

Value

Speed in meters per second.

distributionProfile *Generate training distribution profiles.*

Description

Generate training distribution profiles.

Usage

```
distributionProfile(object, session = NULL, what = c("speed", "heart.rate"),
  grid = list(speed = seq(0, 12.5, by = 0.05), heart.rate = seq(0, 250)),
  parallel = FALSE, cores = NULL)
```

Arguments

object	An object of class <code>trackeRdata</code> .
session	A numeric vector of the sessions to be used, defaults to all sessions.
what	The variables for which the distribution profiles should be generated.
grid	A named list containing the grid for the variables in <code>what</code> .
parallel	Logical. Should computation be carried out in parallel?
cores	Number of cores for parallel computing. If <code>NULL</code> , the number of cores is set to the value of <code>options("cores")</code> (on Windows) or <code>options("mc.cores")</code> (elsewhere), or, if the relevant option is unspecified, to half the number of cores detected.

Value

An object of class `distrProfile`.

References

Kosmidis, I., and Passfield, L. (2015). Linking the Performance of Endurance Runners to Training and Physiological Effects via Multi-Resolution Elastic Net. *ArXiv e-print* arXiv:1506.01388.

Examples

```
data("run", package = "trackeR")
dProfile <- distributionProfile(run, what = "speed", grid = seq(0, 12.5, by = 0.05))
plot(dProfile, smooth = FALSE)
```

fortify.conProfile *Fortify a conProfile object for plotting with ggplot2.*

Description

Fortify a conProfile object for plotting with ggplot2.

Usage

```
## S3 method for class 'conProfile'  
fortify(model, data, melt = FALSE, ...)
```

Arguments

model	The conProfile object.
data	Ignored.
melt	Logical. Should the data be melted into long format instead of the default wide format?
...	Ignored.

fortify.distrProfile *Fortify a distrProfile object for plotting with ggplot2.*

Description

Fortify a distrProfile object for plotting with ggplot2.

Usage

```
## S3 method for class 'distrProfile'  
fortify(model, data, melt = FALSE, ...)
```

Arguments

model	The distrProfile object.
data	Ignored.
melt	Logical. Should the data be melted into long format instead of the default wide format?
...	Ignored.

fortify.trackeRdata *Fortify a trackeRdata object for plotting with ggplot2.*

Description

Fortify a trackeRdata object for plotting with ggplot2.

Usage

```
## S3 method for class 'trackeRdata'  
fortify(model, data, melt = FALSE, ...)
```

Arguments

model	The trackeRdata object.
data	Ignored.
melt	Logical. Should the data be melted into long format instead of the default wide format?
...	Ignored.

fortify.trackeRdataSummary
Fortify a trackeRdataSummary object for plotting with ggplot2.

Description

Fortify a trackeRdataSummary object for plotting with ggplot2.

Usage

```
## S3 method for class 'trackeRdataSummary'  
fortify(model, data, melt = FALSE, ...)
```

Arguments

model	The trackeRdata object.
data	Ignored.
melt	Logical. Should the data be melted into long format instead of the default wide format?
...	Currently not used.

fortify.trackeRWprime *Fortify a trackeRWprime object for plotting with ggplot2.*

Description

Fortify a trackeRWprime object for plotting with ggplot2.

Usage

```
## S3 method for class 'trackeRWprime'
fortify(model, data, melt = FALSE, ...)
```

Arguments

model	The trackeRWprime object as returned by Wprime .
data	Ignored.
melt	Logical. Should the data be melted into long format instead of the default wide format?
...	Ignored.

funPCA	<i>Functional principal components analysis of distribution or concentration profiles.</i>
--------	--

Description

Functional principal components analysis of distribution or concentration profiles.

Generic function for functional principal components analysis.

Usage

```
## S3 method for class 'distrProfile'
funPCA(object, what, nharm = 4, ...)
```

```
## S3 method for class 'conProfile'
funPCA(object, what, nharm = 4, ...)
```

```
funPCA(object, ...)
```

Arguments

object	The object to which a functional principal components analysis is applied.
what	The variable for which the profiles should be analysed.
nharm	The number of principal components estimated.
...	Arguments to be passed to methods.

Details

The ... argument is passed on to [pca.fd](#).

Value

An object of class `trackeRfpca`.

References

Ramsay JO, Silverman BW (2005). *Functional Data Analysis*. Springer-Verlag New York.

Examples

```
data("runs", package = "trackeR")
dp <- distributionProfile(runs, what = "speed")
dp.pca <- funPCA(dp, what = "speed", nharm = 4)
## 1st harmonic captures vast majority of the variation
plot(dp.pca, harm = 1)
## time spent above speed = 0 is the characteristic distinguishing the profiles
sumRuns <- summary(runs)
plot(sumRuns$durationMoving, dp.pca$scores[,1])
```

GC2trackeRdata

Coercion function for use in Golden Cheetah

Description

Experimental state.

Usage

```
GC2trackeRdata(gc, cycling = TRUE, correctDistances = FALSE,
  country = NULL, mask = TRUE, fromDistances = FALSE, lgap = 30,
  lskip = 5, m = 11, silent = FALSE)
```

Arguments

<code>gc</code>	Output of <code>GC.activity</code> .
<code>cycling</code>	Logical. Does the data stem from cycling instead of running?
<code>correctDistances</code>	Logical. Should the distances be corrected for elevation?
<code>country</code>	ISO3 country code for downloading altitude data. If <code>NULL</code> , country is derived from longitude and latitude.
<code>mask</code>	Logical. Passed on to getData . Should only the altitudes for the specified country be extracted (<code>TRUE</code>) or also those for the neighboring countries (<code>FALSE</code>)?
<code>fromDistances</code>	Logical. Should the speeds be calculated from the distance recordings instead of taken from the speed recordings directly?

lgap	Time in seconds corresponding to the minimal sampling rate.
lskip	Time in seconds between the last observation before a small break and the first imputed speed or the last imputed speed and the first observation after a small break.
m	Number of imputed observations in each small break.
silent	Logical. Should warnings be generated if any of the sanity checks on the data are triggered?

See Also

[trackeRdata](#)

generateBaseUnits	<i>Generate base units.</i>
-------------------	-----------------------------

Description

Generate base units.

Usage

```
generateBaseUnits(cycling = FALSE, ...)
```

Arguments

cycling	Logical. Is the data from a cycling session rather than a running session?
...	Currently not used.

generateVariableNames	<i>Generate variables names for internal use in readX functions. the variables vecotrs need to correspond one by on interms of variable type</i>
-----------------------	--

Description

Generate variables names for internal use in readX functions. the variables vecotrs need to correspond one by on interms of variable type

Usage

```
generateVariableNames()
```

getOperations	<i>Generic function for retrieving the operation settings.</i>
---------------	--

Description

Generic function for retrieving the operation settings.

Usage

```
getOperations(object, ...)
```

Arguments

object	The object of which the units of measurement are retrieved.
...	Arguments to be passed to methods.

getOperations.conProfile	<i>Get the operation settings of an conProfile object.</i>
--------------------------	--

Description

Get the operation settings of an conProfile object.

Usage

```
## S3 method for class 'conProfile'  
getOperations(object, ...)
```

Arguments

object	An object of class conProfile as returned by concentrationProfile .
...	Currently not used.

`getOperations.distrProfile`*Get the operation settings of an distrProfile object.*

Description

Get the operation settings of an distrProfile object.

Usage

```
## S3 method for class 'distrProfile'  
getOperations(object, ...)
```

Arguments

<code>object</code>	An object of class <code>distrProfile</code> as returned by distributionProfile .
<code>...</code>	Currently not used.

`getOperations.trackeRdata`*Get the operation settings of an trackeRdata object.*

Description

Get the operation settings of an trackeRdata object.

Usage

```
## S3 method for class 'trackeRdata'  
getOperations(object, ...)
```

Arguments

<code>object</code>	An object of class trackeRdata .
<code>...</code>	Currently not used.

getUnits	<i>Generic function for retrieving the units of measurement.</i>
----------	--

Description

Generic function for retrieving the units of measurement.

Usage

```
getUnits(object, ...)
```

Arguments

object	The object of which the units of measurement are retrieved.
...	Arguments to be passed to methods.

getUnits.conProfile	<i>Get the units of the variables in an conProfile object.</i>
---------------------	--

Description

Get the units of the variables in an conProfile object.

Usage

```
## S3 method for class 'conProfile'  
getUnits(object, ...)
```

Arguments

object	An object of class conProfile.
...	Currently not used.

getUnits.distrProfile *Get the units of the variables in an distrProfile object.*

Description

Get the units of the variables in an distrProfile object.

Usage

```
## S3 method for class 'distrProfile'  
getUnits(object, ...)
```

Arguments

object	An object of class distrProfile.
...	Currently not used.

getUnits.trackerdata *Get the units of the variables in an trackerdata object.*

Description

Get the units of the variables in an trackerdata object.

Usage

```
## S3 method for class 'trackerdata'  
getUnits(object, ...)
```

Arguments

object	An object of class trackerdata .
...	Currently not used.

getUnits.trackeRdataSummary

Get the units of the variables in an trackeRdataSummary object.

Description

Get the units of the variables in an trackeRdataSummary object.

Usage

```
## S3 method for class 'trackeRdataSummary'  
getUnits(object, ...)
```

Arguments

object	An object of class trackeRdataSummary.
...	Currently not used.

getUnits.trackeRWprime

Get the units of the variables in an trackeRWprime object.

Description

Get the units of the variables in an trackeRWprime object.

Usage

```
## S3 method for class 'trackeRWprime'  
getUnits(object, ...)
```

Arguments

object	An object of class trackeRWprime.
...	Currently not used.

imputeSpeeds	<i>Impute speeds.</i>
--------------	-----------------------

Description

Impute speeds of 0 during small breaks within a session.

Usage

```
imputeSpeeds(sessionData, fromDistances = TRUE, lgap = 30, lskip = 5,  
             m = 11, cycling = FALSE, units = NULL)
```

Arguments

sessionData	A multivariate zoo object with observations of either distance or speed (named Distance or Speed, respectively).
fromDistances	Logical. Should the speeds be calculated from the distance recordings instead of taken from the speed recordings directly?
lgap	Time in seconds corresponding to the minimal sampling rate.
lskip	Time in seconds between the last observation before a small break and the first imputed speed or the last imputed speed and the first observation after a small break.
m	Number of imputed observations in each small break.
cycling	Logical. Are the data from a cycling session? If TRUE, power is imputed with 0, else with NA.
units	Units of measurement.

Value

A multivariate [zoo](#) object with imputed observations: 0 for speed, last known position for latitude, longitude and altitude, NA for all other variables. Distances are calculated based on speeds after imputation.

References

Kosmidis, I., and Passfield, L. (2015). Linking the Performance of Endurance Runners to Training and Physiological Effects via Multi-Resolution Elastic Net. *ArXiv e-print* arXiv:1506.01388.

leafletRoute	<i>Plot routes for training sessions.</i>
--------------	---

Description

Plot the route ran/cycled during training on an interactive map. Internet connection is required to download the background map. Icons are by Maps Icons Collection <https://mapicons.mapsmarker.com>

Usage

```
leafletRoute(x, session = NULL, threshold = TRUE, ...)
```

Arguments

x	A object of class trackeRdata .
session	A numeric vector of the sessions to be plotted. Defaults to all sessions.
threshold	Logical. Should thresholds be applied?
...	Additional arguments passed on to threshold .

Examples

```
## Not run:  
data("runs", package = "trackeR")  
leafletRoute(runs, session = 23:24)  
  
## End(Not run)
```

nsessions	<i>Generic function for calculating number of sessions.</i>
-----------	---

Description

Generic function for calculating number of sessions.

Usage

```
nsessions(object, ...)
```

Arguments

object	The object for which to calculate the number of sessions.
...	Arguments to be passed to methods.

plot.conProfile *Plot concentration profiles.*

Description

Plot concentration profiles.

Usage

```
## S3 method for class 'conProfile'
plot(x, session = NULL, what = c("speed",
  "heart.rate"), multiple = FALSE, smooth = TRUE, ...)
```

Arguments

x	An object of class conProfile as returned by concentrationProfile .
session	A vector of the sessions to be plotted, defaults to all sessions. Either a character vector with the session names, e.g., c("Session3", "Session4") or a numeric vector with the relative position of the session(s).
what	Which variables should be plotted?
multiple	Logical. Should all sessions be plotted in one panel?
smooth	Logical. Should unsmoothed profiles be smoothed before plotting?
...	Currently not used.

Examples

```
data("runs", package = "trackerR")
dProfile <- distributionProfile(runs, session = 1:3,
  what = "speed", grid = seq(0, 12.5, by = 0.05))
cProfile <- concentrationProfile(dProfile)
plot(cProfile, smooth = FALSE)
plot(cProfile)
```

plot.distrProfile *Plot distribution profiles.*

Description

Plot distribution profiles.

Usage

```
## S3 method for class 'distrProfile'
plot(x, session = NULL, what = c("speed",
  "heart.rate"), multiple = FALSE, smooth = TRUE, ...)
```

Arguments

x	An object of class <code>distrProfile</code> as returned by <code>distributionProfile</code> .
session	A numeric vector of the sessions to be plotted, defaults to all sessions.
what	Which variables should be plotted?
multiple	Logical. Should all sessions be plotted in one panel?
smooth	Logical. Should unsmoothed profiles be smoothed before plotting?
...	Further arguments to be passed to <code>smootherControl.distrProfile</code> .

Examples

```
data("runs", package = "tracker")
dProfile <- distributionProfile(runs, session = 1:2,
  what = "speed", grid = seq(0, 12.5, by = 0.05))
plot(dProfile, smooth = FALSE)
plot(dProfile, smooth = FALSE, multiple = TRUE)
plot(dProfile, multiple = TRUE)
```

plot.trackeRdata *Plot training sessions in form of trackeRdata objects.*

Description

Plot training sessions in form of trackeRdata objects.

Usage

```
## S3 method for class 'trackeRdata'
plot(x, session = NULL, what = c("pace",
  "heart.rate"), threshold = TRUE, smooth = FALSE, trend = TRUE,
  dates = TRUE, ...)
```

Arguments

x	An object of class <code>trackeRdata</code> .
session	A numeric vector of the sessions to be plotted, defaults to all sessions.
what	Which variables should be plotted?
threshold	Logical. Should thresholds be applied?
smooth	Logical. Should the data be smoothed?
trend	Logical. Should a smooth trend be plotted?
dates	Logical. Should the date of the session be used in the panel header?
...	Further arguments to be passed to <code>threshold</code> and <code>smootherControl.trackeRdata</code> .

Details

Note that a threshold is always applied to the pace. This (upper) threshold corresponds to a speed of 1.4 meters per second, the preferred walking speed of humans. The lower threshold is 0.

Examples

```
## Not run:
data("runs", package = "trackeR")
## plot heart rate and pace for the first 3 sessions
plot(runs, session = 1:3)
## plot raw speed data for session 4
plot(runs, session = 4, what = "speed", threshold = FALSE, smooth = FALSE)
## threshold speed variable
plot(runs, session = 4, what = "speed", threshold = TRUE, smooth = FALSE,
     variable = "speed", lower = 0, upper = 10)
## and smooth (thresholding with default values)
plot(runs, session = 4, what = "speed", threshold = TRUE,
     smooth = TRUE, width = 15, parallel = FALSE)

## End(Not run)
```

plot.trackeRdataSummary

Plot an object of class trackeRdataSummary.

Description

Plot an object of class trackeRdataSummary.

Usage

```
## S3 method for class 'trackeRdataSummary'
plot(x, date = TRUE, what = NULL,
     group = NULL, lines = TRUE, ...)
```

Arguments

x	An object of class trackeRdataSummary.
date	Should the date or the session number be used on the abscissa?
what	Name of variables which should be plotted. Default is all.
group	Which group of variables should be plotted? This can either be total or moving. Default is both.
lines	Should interpolating lines be plotted?
...	Currently not used.

See Also

[summary.trackeRdata](#)

Examples

```
data("runs", package = "tracker")
runSummary <- summary(runs)
plot(runSummary)
plot(runSummary, date = FALSE, group = "total",
      what = c("distance", "duration", "avgSpeed"))
```

plot.trackeRdataZones *Plot training zones.*

Description

Plot training zones.

Usage

```
## S3 method for class 'trackeRdataZones'
plot(x, percent = TRUE, ...)
```

Arguments

x	An object of class trackeRdataZones as returned by zones .
percent	Logical. Should the relative or absolute times spent training in the different zones be plotted?
...	Currently not used.

Examples

```
data("run", package = "tracker")
runZones <- zones(run, what = "speed", breaks = c(0, 2:6, 12.5))
plot(runZones, percent = FALSE)
```

plot.trackeRfpca	<i>Plot function for functional principal components analysis of distribution and concentration profiles.</i>
------------------	---

Description

Plot function for functional principal components analysis of distribution and concentration profiles.

Usage

```
## S3 method for class 'trackeRfpca'  
plot(x, harm = NULL, expand = NULL,  
     pointplot = TRUE, ...)
```

Arguments

x	An object of class trackeRfpca as returned by funPCA .
harm	A numerical vector of the harmonics to be plotted. Defaults to all harmonics.
expand	The factor used to generate suitable multiples of the harmonics. If NULL, the effect of +/- 2 standard deviations of each harmonic is plotted.
pointplot	Should the harmonics be plotted with + and - point characters? Otherwise, lines are used.
...	Currently not used.

References

Ramsay JO, Silverman BW (2005). Functional Data Analysis. Springer-Verlag New York.

See Also

[plot.pca.fd](#)

Examples

```
data("runs", package = "trackeR")  
dp <- distributionProfile(runs, what = "speed")  
dp.pca <- funPCA(dp, what = "speed", nharm = 4)  
## 1st harmonic captures vast majority of the variation  
plot(dp.pca)  
plot(dp.pca, harm = 1, pointplot = FALSE)
```

plot.trackerWprime *Plot W'.*

Description

Plot W'.

Usage

```
## S3 method for class 'trackerWprime'
plot(x, session = NULL, dates = TRUE,
      scaled = TRUE, ...)
```

Arguments

x	An object of class trackerWprime as returned by Wprime .
session	A numeric vector of the sessions to be plotted, defaults to all sessions.
dates	Logical. Should the date of the session be used in the panel header?
scaled	Logical. Should the W' be scaled to the movement variable (power or speed) which is then plotted in the background?
...	Currently not used.

Examples

```
data("runs", package = "trackerR")
wexp <- Wprime(runs, session = 1:3, cp = 4, version = "2012")
plot(wexp, session = 1:2)
```

plotRoute *Plot routes for training sessions.*

Description

Plot the route ran/cycled during training onto a background map. Internet connection is required to download the background map.

Usage

```
plotRoute(x, session = 1, zoom = NULL, speed = TRUE, threshold = TRUE,
          mfrow = NULL, ...)
```


Arguments

x	A object of class trackeRdata .
session	A numeric vector of the sessions to be plotted. Defaults to the first session, all sessions can be plotted by session = NULL.
zoom	The zoom level for the background map as passed on to get_map (2 corresponds roughly to continent level and 20 to building level).
speed	Logical. Should the trace be colored according to speed?
threshold	Logical. Should thresholds be applied?
mfrow	A vector of 2 elements, number of rows and number of columns, specifying the layout for multiple sessions.
...	Additional arguments passed on to threshold and get_map , e.g., source and maptype.

See Also

[get_map](#), [ggmap](#)

Examples

```
## Not run:
data("runs", package = "trackeR")
plotRoute(runs, session = 4, zoom = 13)
plotRoute(runs, session = 4, zoom = 13, maptype = "hybrid")
plotRoute(runs, session = 4, zoom = 13, source = "stamen")
## multiple sessions
plotRoute(runs, session = c(1:5, 8:11), source = "google")
## different zoom level per panel
plotRoute(runs, session = 6:7, source = "google", zoom = c(13, 14))

## End(Not run)
```

```
print.trackeRdataSummary
```

Print method for session summaries.

Description

Print method for session summaries.

Usage

```
## S3 method for class 'trackeRdataSummary'
print(x, ..., digits = 2)
```

Arguments

x	An object of class <code>trackerRdataSummary</code> .
...	Not used, for compatibility with generic <code>summary</code> method only.
digits	Number of digits to be printed.

profile2fd	<i>Transform distribution and concentration profiles to functional data objects of class <code>fd</code>.</i>
------------	---

Description

Transform distribution and concentration profiles to functional data objects of class `fd`.

Usage

```
profile2fd(object, what, ...)
```

Arguments

object	An object of class <code>distrProfile</code> or <code>conProfile</code> , as returned by distributionProfile and concentrationProfile , respectively.
what	The variable for which the profiles should be transformed to a functional data object.
...	Additional arguments passed on to Data2fd

Value

An object of class `fd`.

Examples

```
library("fda")
data("runs", package = "trackerR")
dp <- distributionProfile(runs, what = "speed")
dpFun <- profile2fd(dp, what = "speed",
  fdnames = list("speed", "sessions", "time above threshold"))
dp.pca <- pca.fd(dpFun, nharm = 4)
## 1st harmonic captures vast majority of the variation
dp.pca$varprop
## time spent above speed = 0 is the characteristic distinguishing the profiles
plot(dp.pca, harm = 1)
sumRuns <- summary(runs)
plot(sumRuns$durationMoving, dp.pca$scores[,1])
```

readContainer	<i>Read a GPS container file.</i>
---------------	-----------------------------------

Description

Read a GPS container file.

Usage

```
readContainer(file, type = c("tcx", "db3", "json"), table = "gps_data",
  timezone = "", sessionThreshold = 2, correctDistances = FALSE,
  country = NULL, mask = TRUE, fromDistances = NULL, speedunit = NULL,
  distanceunit = NULL, cycling = FALSE, lgap = 30, lskip = 5, m = 11,
  silent = FALSE, parallel = FALSE, cores = getOption("mc.cores", 2L))
```

Arguments

file	The path to the file.
type	The type of the GPS container file. Supported so far are tcx, db3, and json.
table	The name of the table in the database if type is set to db3, ignored otherwise.
timezone	The timezone of the observations as passed on to as.POSIXct . Ignored for JSON files.
sessionThreshold	The threshold in hours for the time difference between consecutive timestamps above which they are considered to belong to different training sessions.
correctDistances	Logical. Should the distances be corrected for elevation?
country	ISO3 country code for downloading altitude data. If NULL, country is derived from longitude and latitude.
mask	Logical. Passed on to getData . Should only the altitudes for the specified country be extracted (TRUE) or also those for the neighboring countries (FALSE)?
fromDistances	Logical. Should the speeds be calculated from the distance recordings instead of taken from the speed recordings directly. Defaults to TRUE for tcx and Golden Cheetah's json files and to FALSE for db3 files.
speedunit	Character string indicating the measurement unit of the speeds in the container file to be converted into meters per second. Default is m_per_s when type is tcx and km_per_h when type is db3 or json. See Details.
distanceunit	Character string indicating the measurement unit of the distance in the container file to be converted into meters. Default is m when type is tcx and km when type is db3 or json. See Details.
cycling	Logical. Do the data stem from cycling instead of running? If so, the unit of measurement for cadence is set to rev_per_min instead of steps_per_min.
lgap	Time in seconds corresponding to the minimal sampling rate.

lskip	Time in seconds between the last observation before a small break and the first imputed speed or the last imputed speed and the first observation after a small break.
m	Number of imputed observations in each small break.
silent	Logical. Should warnings be generated if any of the sanity checks on the data are triggered?
parallel	Logical. Should computation be carried out in parallel? (Not supported on Windows.)
cores	Number of cores for parallel computing.

Details

Available options for speedunit currently are km_per_h, m_per_s, mi_per_h, ft_per_min and ft_per_s. Available options for distanceunit currently are km, m, mi and ft.

Reading Golden Cheetah's JSON files is experimental.

Value

An object of class `trackerRdata`.

See Also

[trackerRdata](#), [readTCX](#), [readDB3](#), [readJSON](#)

Examples

```
## Not run:
filepath <- system.file("extdata", "2013-06-08-090442.TCX", package = "trackerR")
run <- readContainer(filepath, type = "tcx", timezone = "GMT")

## End(Not run)
```

readDirectory

Read all supported container files from a supplied directory.

Description

Read all supported container files from a supplied directory.

Usage

```
readDirectory(directory, aggregate = TRUE, table = "gps_data",
  timezone = "", sessionThreshold = 2, correctDistances = FALSE,
  country = NULL, mask = TRUE, fromDistances = NULL,
  speedunit = list(tcx = "m_per_s", db3 = "km_per_h", json = "km_per_h"),
  distanceunit = list(tcx = "m", db3 = "km", json = "km"), cycling = FALSE,
  lgap = 30, lskip = 5, m = 11, silent = FALSE, parallel = FALSE,
  cores = getOption("mc.cores", 2L), verbose = TRUE)
```

Arguments

directory	The path to the directory.
aggregate	Logical. Aggregate data from different files to the same session if observations are less than <code>sessionThreshold</code> hours apart? Alternatively, data from different files is stored in different sessions.
table	The name of the table in the database for db3 files.
timezone	The timezone of the observations as passed on to <code>as.POSIXct</code> . Ignored for JSON files.
sessionThreshold	The threshold in hours for the time difference between consecutive timestamps above which they are considered to belong to different training sessions.
correctDistances	Logical. Should the distances be corrected for elevation?
country	ISO3 country code for downloading altitude data. If NULL, country is derived from longitude and latitude.
mask	Logical. Passed on to <code>getData</code> . Should only the altitudes for the specified country be extracted (TRUE) or also those for the neighboring countries (FALSE)?
fromDistances	Logical. Should the speeds be calculated from the distance recordings instead of taken from the speed recordings directly. Defaults to TRUE for txc and Golden Cheetah's json files and to FALSE for db3 files.
speedunit	Character string indicating the measurement unit of the speeds in the container file to be converted into meters per second. Default is <code>m_per_s</code> for txc files and <code>km_per_h</code> for db3 and Golden Cheetah's json files. See Details.
distanceunit	Character string indicating the measurement unit of the distance in the container file to be converted into meters. Default is <code>m</code> for txc files and <code>km</code> for db3 and Golden Cheetah's json files. See Details.
cycling	Logical. Do the data stem from cycling instead of running? If so, the default unit of measurement for cadence is set to <code>rev_per_min</code> instead of <code>steps_per_min</code> and power is imputed with 0, else with NA.
lgap	Time in seconds corresponding to the minimal sampling rate.
lskip	Time in seconds between the last observation before a small break and the first imputed speed or the last imputed speed and the first observation after a small break.
m	Number of imputed observations in each small break.
silent	Logical. Should warnings be generated if any of the sanity checks on the data are triggered?
parallel	Logical. Should computation be carried out in parallel? (Not supported on Windows.)
cores	Number of cores for parallel computing.
verbose	Logical. Should progress reports be printed?

Details

Available options for speedunit currently are km_per_h, m_per_s, mi_per_h, ft_per_min and ft_per_s. Available options for distanceunit currently are km, m, mi and ft.

Reading Golden Cheetah's JSON files is experimental.

Value

An object of class [trackerData](#).

See Also

[trackerData](#), [readTCX](#), [readDB3](#), [readJSON](#)

readX

Read a training file in TCX, db3 or Golden Cheetah's JSON format.

Description

Read a training file in TCX, db3 or Golden Cheetah's JSON format.

Usage

```
readTCX(file, timezone = "", speedunit = "m_per_s", distanceunit = "m",
        parallel = FALSE, cores = getOption("mc.cores", 2L), ...)
```

```
readDB3(file, timezone = "", table = "gps_data", speedunit = "km_per_h",
        distanceunit = "km")
```

```
readJSON(file, timezone = "", speedunit = "km_per_h", distanceunit = "km")
```

Arguments

file	The path to the file.
timezone	The timezone of the observations as passed on to as.POSIXct . Ignored for JSON files.
speedunit	Character string indicating the measurement unit of the speeds in the container file to be converted into meters per second. See Details.
distanceunit	Character string indicating the measurement unit of the distance in the container file to be converted into meters. See Details.
parallel	Logical. Should computation be carried out in parallel? (Not supported on Windows.)
cores	Number of cores for parallel computing.
...	Currently not used.
table	Character string indicating the name of the table with the GPS data in the db3 container file.

Details

Available options for speedunit currently are km_per_h, m_per_s, mi_per_h, ft_per_min and ft_per_s. The default is m_per_s for TCX files and km_per_h for db3 and Golden Cheetah's json files. Available options for distanceunit currently are km, m, mi and ft. The default is m for TCX files and km for db3 and Golden Cheetah's json files.

Reading Golden Cheetah's JSON files is experimental.

Examples

```
## Not run:
## read raw data
filepath <- system.file("extdata", "2013-06-08-090442.TCX", package = "trackeR")
run <- readTCX(file = filepath, timezone = "GMT")

## turn into trackeRdata object
run <- trackeRdata(run, units = data.frame(variable = c("latitude", "longitude",
  "altitude", "distance", "heart.rate", "speed", "cadence", "power"),
  unit = c("degree", "degree", "m", "m", "bpm", "m_per_s", "steps_per_min", "W"),
  stringsAsFactors = FALSE))

## alternatively
run <- readContainer(filepath, type = "tcx", timezone = "GMT")

## End(Not run)
```

restingPeriods	<i>Extract resting period characteristics.</i>
----------------	--

Description

Extract resting period characteristics.

Usage

```
restingPeriods(times, sessionThreshold)
```

Arguments

times Timestamps.
sessionThreshold

The threshold in hours for the time difference between consecutive timestamps above which they are considered to belong to different training sessions.

Value

A list containing a dataframe with start, end, and duration for each session and the resting time between sessions, named "sessions" and "restingTime", respectively.

run	<i>Training session.</i>
-----	--------------------------

Description

Training session.

Usage

run

Format

A [trackeRdata](#) object containing one running training session.

runs	<i>Training sessions.</i>
------	---------------------------

Description

Training sessions.

Usage

runs

Format

A [trackeRdata](#) object containing 27 running training sessions.

sanityChecks	<i>Sanity checks for tracking data.</i>
--------------	---

Description

Heart rate measurements of 0 are set to NA, assuming the athlete is alive. Observations with missing or duplicated time stamps are removed.

Usage

```
sanityChecks(dat, silent)
```

Arguments

dat	Data set to be clean up.
silent	Logical. Should warnings be generated if any of the sanity checks on the data are triggered?

scaled	<i>Generic function for scaling.</i>
--------	--------------------------------------

Description

Generic function for scaling.

Usage

```
scaled(object, ...)
```

Arguments

object	The object to be scaled.
...	Arguments to be passed to methods.

scaled.distrProfile	<i>Scale the distribution profile relative to its maximum value</i>
---------------------	---

Description

Scale the distribution profile relative to its maximum value

Usage

```
## S3 method for class 'distrProfile'  
scaled(object, session = NULL, what = c("speed",  
  "heart.rate"), ...)
```

Arguments

object	An object of class <code>distrProfile</code> as returned by distributionProfile .
session	A numeric vector of the sessions to be plotted, defaults to all sessions.
what	Which variables should be scaled?
...	Currently not used.

smoother	<i>Generic function for smoothing.</i>
----------	--

Description

Generic function for smoothing.

Usage

```
smoother(object, ...)
```

Arguments

object	The object to be smoothed.
...	Arguments to be passed to methods.

smoother.conProfile	<i>Smoother for concentration profiles.</i>
---------------------	---

Description

To ensure positivity of the smoothed concentration profiles, the concentration profiles are transformed to distribution profiles before smoothing. The smoothed distribution profiles are then transformed to concentration profiles.

Usage

```
## S3 method for class 'conProfile'
smoother(object, session = NULL, control = list(...),
  ...)
```

Arguments

object	An object of class <code>conProfile</code> as returned by concentrationProfile .
session	A numeric vector of the sessions to be selected and smoothed. Defaults to all sessions.
control	A list of parameters for controlling the smoothing process. This is passed to smootherControl.distrProfile .
...	Arguments to be used to form the default <code>control</code> argument if it is not supplied directly.

See Also

[smootherControl.distrProfile](#)

smoother.distrProfile *Smoother for distribution profiles.*

Description

The distribution profiles are smoothed using a shape constrained additive model with Poisson responses to ensure that the smoothed distribution profile is positive and monotone decreasing.

Usage

```
## S3 method for class 'distrProfile'  
smoother(object, session = NULL, control = list(...),  
  ...)
```

Arguments

object	An object of class <code>distrProfile</code> as returned by distributionProfile .
session	A numeric vector of the sessions to be selected and smoothed. Defaults to all sessions.
control	A list of parameters for controlling the smoothing process. This is passed to smootherControl.distrProfile .
...	Arguments to be used to form the default <code>control</code> argument if it is not supplied directly.

References

Kosmidis, I., and Passfield, L. (2015). Linking the Performance of Endurance Runners to Training and Physiological Effects via Multi-Resolution Elastic Net. *ArXiv e-print* arXiv:1506.01388.

Pya, N. and Wood S. (2015). Shape Constrained Additive Models. *Statistics and Computing*, 25(3), 543–559.

See Also

[smootherControl.distrProfile](#)

smoother.trackeRdata *Smoother for [trackeRdata](#) objects.*

Description

Smoother for [trackeRdata](#) objects.

Usage

```
## S3 method for class 'trackeRdata'
smoother(object, session = NULL, control = list(...),
  ...)
```

Arguments

object	An object of class trackeRdata .
session	The sessions to be smoothed. Default is all sessions.
control	A list of parameters for controlling the smoothing process. This is passed to smootherControl.trackeRdata .
...	Arguments to be used to form the default control argument if it is not supplied directly.

Value

An object of class [trackeRdata](#).

See Also

[smootherControl.trackeRdata](#)

Examples

```
data("run", package = "trackeR")
## unsmoothed speeds
plot(run, smooth = FALSE)
## default smoothing
plot(run, smooth = TRUE, cores = 2)
## smoothed with some non-default options
runS <- smoother(run, fun = "median", width = 20, what = "speed", cores = 2)
plot(runS, smooth = FALSE)
```

smootherControl.distrProfile

Auxiliary function for [smoother.distrProfile](#). Typically used to construct a control argument for [smoother.distrProfile](#).

Description

Auxiliary function for [smoother.distrProfile](#). Typically used to construct a control argument for [smoother.distrProfile](#).

Usage

```
smootherControl.distrProfile(what = c("speed", "heart.rate"), k = 30,
  sp = NULL, parallel = FALSE, cores = NULL)
```

Arguments

what	Vector of the names of the variables which should be smoothed.
k	Number of knots.
sp	A vector of smoothing parameters passed on to scam .
parallel	Logical. Should computation be carried out in parallel?
cores	Number of cores for parallel computing. If NULL, the number of cores is set to the value of <code>options("corese")</code> (on Windows) or <code>options("mc.cores")</code> (elsewhere), or, if the relevant option is unspecified, to half the number of cores detected.

`smootherControl.trackeRdata`

Auxiliary function for [smoother.trackeRdata](#). Typically used to construct a control argument for [smoother.trackeRdata](#).

Description

Auxiliary function for [smoother.trackeRdata](#). Typically used to construct a control argument for [smoother.trackeRdata](#).

Usage

```
smootherControl.trackeRdata(fun = "mean", width = 10, parallel = FALSE,
  cores = NULL, what = c("speed", "heart.rate"), nsessions = NA, ...)
```

Arguments

fun	The name of the function to be matched and used to aggregate/smooth the data.
width	The width of the window in which the raw observations get aggregated via function fun.
parallel	Logical. Should computation be carried out in parallel?
cores	Number of cores for parallel computing. If NULL, the number of cores is set to the value of <code>options("corese")</code> (on Windows) or <code>options("mc.cores")</code> (elsewhere), or, if the relevant option is unspecified, to half the number of cores detected.
what	Vector of the names of the variables which should be smoothed.
nsessions	Vector containing the number of session. Default corresponds to all sessions belonging to the same group. Used only internally.
...	Currently not used.

See Also

[smoother.trackeRdata](#)

speed2distance *Convert speed to distance.*

Description

Convert speed to distance.

Usage

```
speed2distance(speed, time, timeunit, cumulative = TRUE)
```

Arguments

speed	Speed in meters per second.
time	Time.
timeunit	Time unit in speed, e.g., "hours" for speed in *_per_h.
cumulative	Logical. Should the cumulative distances be returned?

Value

Distance in meters.

summary.trackeRdata *Summary of training sessions.*

Description

Summary of training sessions.

Usage

```
## S3 method for class 'trackeRdata'
summary(object, session = NULL,
        movingThreshold = NULL, ...)
```

Arguments

object	An object of class <code>trackeRdata</code> .
session	A numeric vector of the sessions to be summarised, defaults to all sessions.
movingThreshold	The threshold above which speed an athlete is considered moving (given in the unit of the speed measurements in object. If NULL, the default, the threshold corresponds to a slow walking speed (1 m/s, converted to another speed unit, if necessary). For reference, the preferred walking speed for humans is around 1.4 m/s (Bohannon, 1997).
...	Currently not used.

Value

An object of class `trackerRdataSummary`.

References

Bohannon RW (1997). "Comfortable and Maximum Walking Speed of Adults Aged 20–79 Years: Reference Values and Determinants." *Age and Ageing*, 26(1), 15–19. doi: 10.1093/ageing/26.1.15.

See Also

[plot.trackerRdataSummary](#)

Examples

```
data("runs", package = "trackerR")
runSummary <- summary(runs, session = 1:2)
## print summary
runSummary
print(runSummary, digits = 3)
## change units
changeUnits(runSummary, variable = "speed", unit = "km_per_h")
## plot summary
runSummaryFull <- summary(runs)
plot(runSummaryFull)
plot(runSummaryFull, group = c("total", "moving"),
     what = c("avgSpeed", "distance", "duration", "avgHeartRate"))
```

threshold

Thresholding for variables in trackerRdata objects.

Description

Thresholding for variables in `trackerRdata` objects.

Usage

```
threshold(object, variable, lower, upper, ...)
```

Arguments

<code>object</code>	An object of class <code>trackerRdata</code> .
<code>variable</code>	A vector containing the names of the variables to which thresholding is applied. See Details.
<code>lower</code>	A vector containing the corresponding lower thresholds. See Details.
<code>upper</code>	A vector containing the corresponding upper thresholds. See Details.
<code>...</code>	Currently not used.

Details

Argument `variable` can also be a data frame containing the variable names, lower, and upper thresholds. If arguments `variable`, `lower`, and `upper` are all unspecified, the following default thresholds are employed: latitude [-90, 90] degrees, longitude [-180, 180] degrees, altitude [-500, 9000] m, distance [0, Inf] meters, heart rate [0, 250] bpm, power [0, Inf] W, pace [0, Inf] min per km, duration [0, Inf] seconds. The thresholds for speed differ for running, [0, 12.5] meters per second, and cycling, [0, 100] meters per second. Default thresholds are converted to the units of measurement of the object before they are applied.

Examples

```
data("runs", package = "tracker")
plot(runs, session = 4, what = "speed", threshold = FALSE)
runsT <- threshold(runs, variable = "speed", lower = 0, upper = 12.5)
plot(runsT, session = 4, what = "speed", threshold = FALSE)
```

`timeAboveThreshold` *Time spent above a certain threshold*

Description

Time spent above a certain threshold

Usage

```
timeAboveThreshold(object, threshold = -1, ge = TRUE)
```

Arguments

<code>object</code>	A (univariate) zoo object.
<code>threshold</code>	The threshold.
<code>ge</code>	Logical. Should time include the threshold (greater or equal to threshold) or not (greater only)?

`timeline` *Generic function for visualising the sessions on a time versus date plot.*

Description

Generic function for visualising the sessions on a time versus date plot.

Usage

```
timeline(object, lims, ...)
```


Arguments

object	An object of class <code>trackeRdata</code> or <code>trackeRdataSummary</code>
lims	An optional vector of two times in HH:MM format. Default is NULL. If supplied, the times are used to define the limits of the time axis.
...	Arguments passed to <code>summary.trackeRdata</code>

Examples

```
## Not run:
data("runs", package = "trackeR")
## timeline plot applied on the trackeRdata object directly and with
## inferred limits for the time axis
timeline(runs)

## the same timeline plot applied on the trackeRdataSummary object
runSummary <- summary(runs)
timeline(runSummary, lims = c("00:01", "23:59"))

## End(Not run)
```

trackeR	<i>trackeR: Infrastructure for running and cycling data from GPS-enabled tracking devices</i>
---------	---

Description

trackeR provides infrastructure for handling cycling and running data from GPS-enabled tracking devices. After extraction and appropriate manipulation of the training or competition attributes, the data are placed into session-aware data objects with an S3 class `trackeRdata`. The information in the resultant data objects can then be visualised, summarised and analysed through corresponding flexible and extensible methods.

Note

Core facilities in the trackeR package, including reading functions (see [readX](#)), data pre-processing strategies (see [trackeRdata](#)), and calculation of concentration and distribution profiles (see [distributionProfile](#) and [concentrationProfile](#)) are based on un-packaged R code that was developed by Ioannis Kosmidis for the requirements of the analyses in Kosmidis & Passfield (2015).

References

Kosmidis, I., and Passfield, L. (2015). Linking the Performance of Endurance Runners to Training and Physiological Effects via Multi-Resolution Elastic Net. *ArXiv e-print* arXiv:1506.01388.

 trackeRdata

 Create a trackeRdata object.

Description

Create a trackeRdata object from a data frame with observations being divided in separate training sessions. For breaks within a session observations are imputed.

Usage

```
trackeRdata(dat, units = NULL, cycling = FALSE, sessionThreshold = 2,
  correctDistances = FALSE, country = NULL, mask = TRUE,
  fromDistances = TRUE, lgap = 30, lskip = 5, m = 11, silent = FALSE)
```

Arguments

dat	A data frame.
units	A data frame containing the unit of measurement for all variables. See Details.
cycling	Logical. Do the data stem from cycling instead of running? If so, the default unit of measurement for cadence is set to rev_per_min instead of steps_per_min and power is imputed with 0, else with NA.
sessionThreshold	The threshold in hours for the time difference between consecutive timestamps above which they are considered to belong to different training sessions.
correctDistances	Logical. Should the distances be corrected for elevation?
country	ISO3 country code for downloading altitude data. If NULL, country is derived from longitude and latitude.
mask	Logical. Passed on to <code>getData</code> . Should only the altitudes for the specified country be extracted (TRUE) or also those for the neighboring countries (FALSE)?
fromDistances	Logical. Should the speeds be calculated from the distance recordings instead of taken from the speed recordings directly?
lgap	Time in seconds corresponding to the minimal sampling rate.
lskip	Time in seconds between the last observation before a small break and the first imputed speed or the last imputed speed and the first observation after a small break.
m	Number of imputed observations in each small break.
silent	Logical. Should warnings be generated if any of the sanity checks on the data are triggered?

Details

The `units` argument takes a data frame with two variables named `variable` and `unit`. Possible options include:

- variables `latitude` and `longitude` with unit `degree`
- variables `altitude`, `distance` with unit `m`, `km`, `mi` or `ft`
- variable `heart.rate` with unit `bpm`
- variable `speed` with unit `m_per_s`, `km_per_h`, `ft_per_min`, `ft_per_s` or `mi_per_h`
- variable `cadence` with unit `steps_per_min` or `rev_per_min`
- variable `power` with unit `W` or `kW`.

If the argument `units` is `NULL`, the default units are used. These are the first options, i.e., `m` for variables `altitude` and `distance`, `m_per_s` for variable `speed` as well as `W` for variable `power`. The default for variable `cadence` depends on the value of argument `cycling`.

During small breaks within a session, e.g., because the recording device was paused, observations are imputed the following way: 0 for speed, last known position for latitude, longitude and altitude, `NA` or 0 power for running or cycling session, respectively, and `NA` for all other variables. Distances are (re-)calculated based on speeds after imputation.

See Also

[readContainer](#) for reading `.tcx` and `.db3` files directly into `trackeRdata` objects.

Examples

```
## Not run:
## read raw data
filepath <- system.file("extdata", "2013-06-08-090442.TCX", package = "trackeR")
run <- readTCX(file = filepath, timezone = "GMT")

## turn into trackeRdata object
run <- trackeRdata(run, units = data.frame(variable = c("latitude", "longitude",
  "altitude", "distance", "heart.rate", "speed", "cadence", "power"),
  unit = c("degree", "degree", "m", "m", "bpm", "m_per_s", "steps_per_min", "W"),
  stringsAsFactors = FALSE))

## alternatively
run <- readContainer(filepath, type = "tcx", timezone = "GMT")

## End(Not run)
```

Wexp	<i>W' expended.</i>
------	---------------------

Description

Calculate W' expended, i.e., the work capacity above critical power which has been depleted and not yet been replenished.

Usage

```
Wexp(object, w0, cp, version = c("2015", "2012"), meanRecoveryPower = FALSE)
```

Arguments

object	Univariate zoo object containing the time stamped power output or speed values. (Power should be in Watts, speed in meters per second.)
w0	Initial capacity of W' , as calculated based on the critical power model by Monod and Scherrer (1965).
cp	Critical power/speed, i.e., the power/speed which can be maintained for longer period of time.
version	How should W' be replenished? Options include "2015" and "2012" for the versions presented in Skiba et al. (2015) and Skiba et al. (2012), respectively. See Details.
meanRecoveryPower	Should the mean of all power outputs below critical power be used as recovery power? See Details.

Details

Skiba et al. (2015) and Skiba et al. (2012) both describe an exponential decay of W' expended over an interval $[t_{i-1}, t_i]$ if the power output during this interval is below critical power: $W_{\text{exp}}(t_i) = W_{\text{exp}}(t_{i-1}) * \exp(\nu * (t_i - t_{i-1}))$. However, the factor ν differs: Skiba et al. (2012) describe it as $1/\tau$ with τ estimated as $\tau = 546 * \exp(-0.01 * (CP - P_i) + 316)$. Skiba et al. (2015) use $(P_i - CP) / W'_0$. Skiba et al. (2012) and Skiba et al. (2015) employ a constant recovery power (calculated as the mean over all power outputs below critical power). This rationale can be applied by setting the argument `meanRecoveryPower` to `TRUE`. Note that this employs information from the all observations with a power output below critical power, not just those prior to the current time point.

References

Monod H, Scherrer J (1965). "The Work Capacity of a Synergic Muscular Group." *Ergonomics*, 8(3), 329–338. Skiba PF, Chidnok W, Vanhatalo A, Jones AM (2012). "Modeling the Expenditure and Reconstitution of Work Capacity above Critical Power." *Medicine & Science in Sports & Exercise*, 44(8), 1526–1532. Skiba PF, Fulford J, Clarke DC, Vanhatalo A, Jones AM (2015). "Intramuscular Determinants of the Ability to Recover Work Capacity above Critical Power." *European Journal of Applied Physiology*, 115(4), 703–713.

Wprime

W': work capacity above critical power.

Description

Based on the critical power model for cycling (Monod and Scherrer, 1965), W' (read W prime) describes the finite work capacity above critical power (Skiba et al., 2012). While W' is depleted during exercise above critical power, it is replenished during exercise below critical power. Thus, it is of interest how much of this work capacity has been depleted and not yet been replenished again, named W' expended, or how much of this work capacity is still available, named W' balance. This principal is applied to runners by substituting power and critical power with speed and critical speed, respectively (Skiba et al., 2012).

Usage

```
Wprime(object, session = NULL, quantity = c("expended", "balance"), w0, cp,
       version = c("2015", "2012"), meanRecoveryPower = FALSE,
       parallel = FALSE, cores = NULL, ...)
```

Arguments

object	A trackeRdata object.
session	A numeric vector of the sessions to be used, defaults to all sessions.
quantity	Should W' "expended" or W' "balance" be returned?
w0	Initial capacity of W' , as calculated based on the critical power model by Monod and Scherrer (1965).
cp	Critical power/speed, i.e., the power/speed which can be maintained for longer period of time.
version	How should W' be replenished? Options include "2015" and "2012" for the versions presented in Skiba et al. (2015) and Skiba et al. (2012), respectively. See Details.
meanRecoveryPower	Should the mean of all power outputs below critical power be used as recovery power? See Details.
parallel	Logical. Should computation be carried out in parallel?
cores	Number of cores for parallel computing. If NULL, the number of cores is set to the value of <code>options("corese")</code> (on Windows) or <code>options("mc.cores")</code> (elsewhere), or, if the relevant option is unspecified, to half the number of cores detected.
...	Currently not used.

Details

Skiba et al. (2015) and Skiba et al. (2012) both describe an exponential decay of W' expended over an interval $[t_{i-1}, t_i]$ if the power output during this interval is below critical power: $W_exp(t_i) = W_exp(t_{i-1}) * \exp(\nu * (t_i - t_{i-1}))$. However, the factor ν differs: Skiba et al. (2012) describe it as $1/\tau$ with τ estimated as $\tau = 546 * \exp(-0.01 * (CP - P_i) + 316)$. Skiba et al. (2015) use $(P_i - CP) / W'_0$. Skiba et al. (2012) and Skiba et al. (2015) employ a constant recovery power (calculated as the mean over all power outputs below critical power). This rational can be applied by setting the argument `meanRecoveryPower` to `TRUE`. Note that this employes information from the all observations with a power output below critical power, not just those prior to the current time point.

Value

An object of class `trackerWprime`.

References

Monod H, Scherrer J (1965). "The Work Capacity of a Synergic Muscular Group." *Ergonomics*, 8(3), 329–338. Skiba PF, Chidnok W, Vanhatalo A, Jones AM (2012). "Modeling the Expenditure and Reconstitution of Work Capacity above Critical Power." *Medicine & Science in Sports & Exercise*, 44(8), 1526–1532. Skiba PF, Fulford J, Clarke DC, Vanhatalo A, Jones AM (2015). "Intramuscular Determinants of the Ability to Recover Work Capacity above Critical Power." *European Journal of Applied Physiology*, 115(4), 703–713.

Examples

```
data("runs", package = "trackerR")
wexp <- Wprime(runs, session = c(11,13), cp = 4, version = "2012")
plot(wexp)
```

zones

Time spent in training zones.

Description

Time spent in training zones.

Usage

```
zones(object, session = NULL, what = c("speed", "heart.rate"),
      breaks = list(speed = 0:10, heart.rate = c(0, seq(75, 225, by = 50), 250)),
      parallel = FALSE, cores = NULL, ...)
```

Arguments

object	An object of class trackeRdata .
session	A numeric vector of the sessions to be plotted, defaults to all sessions.
what	A vector of variable names.
breaks	A list of breakpoints between zones, corresponding to the variables in what.
parallel	Logical. Should computation be carried out in parallel?
cores	Number of cores for parallel computing. If NULL, the number of cores is set to the value of <code>options("corese")</code> (on Windows) or <code>options("mc.cores")</code> (elsewhere), or, if the relevant option is unspecified, to half the number of cores detected.
...	Currently not used.

Value

An object of class `trackeRdataZones`.

See Also

[plot.trackeRdataZones](#)

Examples

```
data("run", package = "trackeR")
runZones <- zones(run, what = "speed", breaks = list(speed = c(0, 2:6, 12.5)))
## if breaks is a named list, argument 'what' can be left unspecified
runZones <- zones(run, breaks = list(speed = c(0, 2:6, 12.5)))
## if only a single variable is to be evaluated, 'breaks' can also be a vector
runZones <- zones(run, what = "speed", breaks = c(0, 2:6, 12.5))
plot(runZones)
```

Index

*Topic **datasets**

- run, [40](#)
- runs, [40](#)

- append, [3](#)
- append.conProfile, [4](#)
- append.distrProfile
(append.conProfile), [4](#)
- append.trackeRdata, [4](#)
- as.POSIXct, [35](#), [37](#), [38](#)

- c2d, [5](#)
- changeUnits, [5](#)
- changeUnits.conProfile, [6](#)
- changeUnits.distrProfile, [6](#)
- changeUnits.trackeRdata, [7](#)
- changeUnits.trackeRdataSummary, [7](#)
- changeUnits.trackeRWprime, [8](#)
- concentrationProfile, [6](#), [8](#), [20](#), [27](#), [34](#), [42](#),
[49](#)
- conversions, [9](#)

- Data2fd, [34](#)
- decreasingSmoother, [13](#)
- distance2speed, [13](#)
- distributionProfile, [6](#), [8](#), [14](#), [21](#), [28](#), [34](#),
[41](#), [43](#), [49](#)

- fd, [34](#)
- fortify.conProfile, [15](#)
- fortify.distrProfile, [15](#)
- fortify.trackeRdata, [16](#)
- fortify.trackeRdataSummary, [16](#)
- fortify.trackeRWprime, [17](#)
- ft2ft (conversions), [9](#)
- ft2km (conversions), [9](#)
- ft2m (conversions), [9](#)
- ft2mi (conversions), [9](#)
- ft_per_min2ft_per_min (conversions), [9](#)
- ft_per_min2ft_per_s (conversions), [9](#)
- ft_per_min2km_per_h (conversions), [9](#)
- ft_per_min2km_per_min (conversions), [9](#)
- ft_per_min2m_per_s (conversions), [9](#)
- ft_per_min2mi_per_h (conversions), [9](#)
- ft_per_min2mi_per_min (conversions), [9](#)
- ft_per_s2ft_per_min (conversions), [9](#)
- ft_per_s2ft_per_s (conversions), [9](#)
- ft_per_s2km_per_h (conversions), [9](#)
- ft_per_s2km_per_min (conversions), [9](#)
- ft_per_s2m_per_s (conversions), [9](#)
- ft_per_s2mi_per_h (conversions), [9](#)
- ft_per_s2mi_per_min (conversions), [9](#)
- funPCA, [17](#), [31](#)

- GC2trackeRdata, [18](#)
- generateBaseUnits, [19](#)
- generateVariableNames, [19](#)
- get_map, [33](#)
- getData, [18](#), [35](#), [37](#), [50](#)
- getOperations, [20](#)
- getOperations.conProfile, [20](#)
- getOperations.distrProfile, [21](#)
- getOperations.trackeRdata, [21](#)
- getUnits, [22](#)
- getUnits.conProfile, [22](#)
- getUnits.distrProfile, [23](#)
- getUnits.trackeRdata, [23](#)
- getUnits.trackeRdataSummary, [24](#)
- getUnits.trackeRWprime, [24](#)
- ggmap, [33](#)

- h2h (conversions), [9](#)
- h2min (conversions), [9](#)
- h2s (conversions), [9](#)

- imputeSpeeds, [25](#)

- km2ft (conversions), [9](#)
- km2km (conversions), [9](#)
- km2m (conversions), [9](#)

- km2mi (conversions), 9
- km_per_h2ft_per_min (conversions), 9
- km_per_h2ft_per_s (conversions), 9
- km_per_h2km_per_h (conversions), 9
- km_per_h2km_per_min (conversions), 9
- km_per_h2m_per_s (conversions), 9
- km_per_h2mi_per_h (conversions), 9
- km_per_h2mi_per_min (conversions), 9
- km_per_min2ft_per_min (conversions), 9
- km_per_min2ft_per_s (conversions), 9
- km_per_min2km_per_h (conversions), 9
- km_per_min2km_per_min (conversions), 9
- km_per_min2m_per_s (conversions), 9
- km_per_min2mi_per_h (conversions), 9
- km_per_min2mi_per_min (conversions), 9
- kW2kW (conversions), 9
- kW2W (conversions), 9
- leafletRoute, 26
- m2ft (conversions), 9
- m2km (conversions), 9
- m2m (conversions), 9
- m2mi (conversions), 9
- m_per_s2ft_per_min (conversions), 9
- m_per_s2ft_per_s (conversions), 9
- m_per_s2km_per_h (conversions), 9
- m_per_s2km_per_min (conversions), 9
- m_per_s2m_per_s (conversions), 9
- m_per_s2mi_per_h (conversions), 9
- m_per_s2mi_per_min (conversions), 9
- mi2ft (conversions), 9
- mi2km (conversions), 9
- mi2m (conversions), 9
- mi2mi (conversions), 9
- mi_per_h2ft_per_min (conversions), 9
- mi_per_h2ft_per_s (conversions), 9
- mi_per_h2km_per_h (conversions), 9
- mi_per_h2km_per_min (conversions), 9
- mi_per_h2m_per_s (conversions), 9
- mi_per_h2mi_per_h (conversions), 9
- mi_per_h2mi_per_min (conversions), 9
- mi_per_min2ft_per_min (conversions), 9
- mi_per_min2ft_per_s (conversions), 9
- mi_per_min2km_per_h (conversions), 9
- mi_per_min2km_per_min (conversions), 9
- mi_per_min2m_per_s (conversions), 9
- mi_per_min2mi_per_h (conversions), 9
- mi_per_min2mi_per_min (conversions), 9
- min2h (conversions), 9
- min2min (conversions), 9
- min2s (conversions), 9
- min_per_km2min_per_km (conversions), 9
- min_per_km2min_per_mi (conversions), 9
- min_per_km2s_per_m (conversions), 9
- min_per_mi2min_per_km (conversions), 9
- min_per_mi2min_per_mi (conversions), 9
- min_per_mi2s_per_m (conversions), 9
- nsessions, 26
- pca.fd, 18
- plot.conProfile, 27
- plot.distrProfile, 27
- plot.pca.fd, 31
- plot.trackeRdata, 28
- plot.trackeRdataSummary, 29, 47
- plot.trackeRdataZones, 30, 55
- plot.trackeRfpca, 31
- plot.trackeRWprime, 32
- plotRoute, 32
- print.trackeRdataSummary, 33
- profile2fd, 34
- readContainer, 35, 51
- readDB3, 36, 38
- readDB3 (readX), 38
- readDirectory, 36
- readJSON, 36, 38
- readJSON (readX), 38
- readTCX, 36, 38
- readTCX (readX), 38
- readX, 38, 49
- restingPeriods, 39
- rev_per_min2rev_per_min (conversions), 9
- run, 40
- runs, 40
- s2h (conversions), 9
- s2min (conversions), 9
- s2s (conversions), 9
- s_per_m2min_per_km (conversions), 9
- s_per_m2min_per_mi (conversions), 9
- s_per_m2s_per_m (conversions), 9
- sanityChecks, 40
- scaled, 41
- scaled.distrProfile, 41
- scam, 13, 45

smoother, [42](#)
smoother.conProfile, [42](#)
smoother.distrProfile, [43](#), [44](#)
smoother.trackRdata, [43](#), [45](#)
smootherControl.distrProfile, [28](#), [42](#), [43](#),
[44](#)
smootherControl.trackRdata, [28](#), [44](#), [45](#)
speed2distance, [46](#)
steps_per_min2steps_per_min
(conversions), [9](#)
summary.trackRdata, [30](#), [46](#), [49](#)

threshold, [26](#), [28](#), [33](#), [47](#)
timeAboveThreshold, [48](#)
timeline, [48](#)
tracker, [49](#)
tracker-package (tracker), [49](#)
trackRdata, [7](#), [8](#), [14](#), [16](#), [19](#), [21](#), [23](#), [26](#), [28](#),
[33](#), [36](#), [38](#), [40](#), [43](#), [44](#), [46](#), [47](#), [49](#), [50](#),
[53](#), [55](#)

W2kW (conversions), [9](#)
W2W (conversions), [9](#)
Wexp, [52](#)
Wprime, [17](#), [32](#), [53](#)

zones, [30](#), [54](#)
zoo, [25](#), [52](#)