Package 'animalTrack'

October 12, 2022

Type Package

Title Animal track reconstruction for high frequency 2-dimensional (2D) or 3-dimensional (3D) movement data.

Version 1.0.0

Date 2013-09-19

Author Ed Farrell and Lee Fuiman

Maintainer Ed Farrell <edward.farrell27@gmail.com>

Depends R (>= 2.10.0), rgl

Description

2D and 3D animal tracking data can be used to reconstruct tracks through time/space with correction based on known positions. 3D visualization of animal position and attitude.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2013-09-23 16:15:49

R topics documented:

`animalTrack-package` *Animal track reconstruction for high frequency 2-dimensional (2D) or 3-dimensional (3D) movement data.*

Description

2D and 3D animal tracking data can be used to reconstruct tracks through time/space with correction based on known positions. 3D visualization of animal position and attitude.

Details

Package:	animalTrack
Type:	Package
Version:	1.0.0
Date:	2013-09-19
License:	GPL (>=2)
LazyLoad:	yes

This package is intended for processing and visualization of high frequency animal tracking data, either 2D or 3D. As miniaturization technology has improved over the past 30 years, high resolution animal-borne data loggers have become increasingly used by scientists to study animal behavior. This package is intended to bridge the gap between raw multi-sensor data and meaningful calibrated variables (e.g. position, speed, attitude, heading) that can be used to understand natural behavior.

All calculations that are performed to estimate the pitch, roll, and yaw angles of the animal are in the north-east-down (NED) frame of reference. It is important that the user either record measurements in the NED frame, or shift the raw measurements of each axis to NED before any calculations using this package.

Author(s)

Author: Ed Farrell <edward.farrell27@gmail.com> and Lee Fuiman <lee.fuiman@utexas.edu>

Affiliation: The University of Texas at Austin, Marine Science Institute http://www.utmsi.utexas.edu/

Maintainer: Ed Farrell <edward.farrell27@gmail.com>

`accuracy` *Estimate accuracy of three axis accelerometer/magnetometer.*

Description

The accuracy of the accelerometer and magnetometer is crucial to estimating pitch, roll, and heading. The axes must be orthogonal.

Usage

```
accuracy(x, y, z)

accuracy.improve(x, y, z)
```

Arguments

x	a numeric vector of x-axis values.
y	a numeric vector of y-axis values.
z	a numeric vector of z-axis values.

Details

In the case of a three-axis accelerometer, the combined static acceleration (i.e. acceleration due to gravity) should always be 1**g**. Accelerometers measure the Earth's gravity as well as movement of the animal in gravitational units, denoted as **g** units. The sensor should be calibrated so that 1**g** = -9.8 m/s^2, which is the acceleration due to gravity (this varies locally). The combined acceleration will vary from 1**g** during dynamic acceleration (e.g. an animal accelerates to chase prey), or if the axes are not perfectly orthogonal. In the case of dynamic acceleration, an effort must be made to filter out the high frequency signal so that only the static signal (i.e. body orientation) remains. The `accuracy` function evaluates the accuracy of the accelerometer by calculating the vector norm of the acceleration, A, using the equation

$A = \sqrt{Ax^2 + Ay^2 + Az^2}$

and subsequently calculating the variance and standard deviation.

The `accuracy.improve` function provides an estimate of the Z-axis value of the accelerometer using the X and Y-axis values. The Z-axis estimate gives a value for Z that makes the vector norm 1boldg (i.e. all three axes are perfectly orthogonal and X and Y are assumed to be correct). This estimate for Z can then be used to evaluate error that may correspond to pitch angle, or may be a constant error. This will vary between sensors.

Value

For `accuracy()`, a list containing the raw accuracy values (which should equal 1, in the case of perfect accuracy), variance, and standard deviation

acc	vector containing the estimated accuracy values
var	variance of acc
std	standard deviation of acc

For `accuracy.improve()`, a list containing multiple components

z.est	vector of estimated Z-axis values.
z.resid	vector of residuals (z - z.est).
acc.input	vector of accuracy values of the input (i.e. vector norm of x, y, z). Same as accuracy().

`acc.input.var` accuracy variance of the input values.

`acc.input.std` accuracy standard deviation of the input values.

`acc.improve` vector of accuracy values using `z.est`. Same as `accuracy(x, y, z.est)`

`acc.improve.var` accuracy variance of `acc.improve`.

`acc.improve.std` accuracy standard deviation of `acc.improve`.

Note

The `accuracy.improve` function should be used to evaluate the accuracy of the Z-axis, assuming that the X and Y axes estimates are reliable. If X and Y are not reliable, then the estimate for Z will be erroneous. This function is especially useful if the accelerometer or magnetometer has a two axis X and Y module, and a one-axis Z module that may not be properly aligned within the instrument housing.

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Grygorenko, V. (2011), Sensing - magnetic compass with tilt compensation. Cypress Perform, AN2272, Document No. 001-32379 Rev. B.

Tuck, K. (2007), Tilt sensing using linear accelerometers. Freescale Semiconductor, AN3461 Rev. 2.

Examples

```
## Load seal dive data. see help(hugh) for details on variables.
## There are six dives, with surface intervals. here, we will
## look at the first 3 dives.
data(hugh)
## accuracy of the accelerometer.
A.acc.1 <- accuracy(hugh$Ax.dec[hugh$diven == 1],hugh$Ay.dec[hugh$diven == 1],
                    hugh$Az.dec[hugh$diven == 1])
A.acc.2 <- accuracy(hugh$Ax.dec[hugh$diven == 2],hugh$Ay.dec[hugh$diven == 2],
                    hugh$Az.dec[hugh$diven == 2])
A.acc.3 <- accuracy(hugh$Ax.dec[hugh$diven == 3],hugh$Ay.dec[hugh$diven == 3],
                    hugh$Az.dec[hugh$diven == 3])
## Plot the accuracy during the track.
## dive 1 has a high error due to some outliers. dive 2 looks much better.
## dive 3 has some outliers similar to dive 1.
plot(A.acc.1$acc)
plot(A.acc.2$acc)
plot(A.acc.3$acc)
## error angle in degrees
asin(A.acc.1$std)*(180/pi)
asin(A.acc.2$std)*(180/pi)
```

```
asin(A.acc.3$std)*(180/pi)

## See if there is a systematic error in the Z-axis. Warnings will occur
## if there are errors in the X, Y, or Z-axis (i.e. too large/small
## of a value).
acc.d1 <- accuracy.improve(hugh$Ax.dec[hugh$diven == 1],hugh$Ay.dec[hugh$diven == 1],
                           hugh$Az.dec[hugh$diven == 1])
acc.d2 <- accuracy.improve(hugh$Ax.dec[hugh$diven == 2],hugh$Ay.dec[hugh$diven == 2],
                           hugh$Az.dec[hugh$diven == 2])
acc.d3 <- accuracy.improve(hugh$Ax.dec[hugh$diven == 3],hugh$Ay.dec[hugh$diven == 3],
                           hugh$Az.dec[hugh$diven == 3])
## Plot the Z-axis values, the estimates, and the residuals
## We can see that there are large spikes. These spikes may
## coincide with movement. The Z values during these spikes
## could be filtered out, or a new filter to the Z accelerometer
## may be needed. This will vary according to the sensor,
## type of animal, etc.
plot(hugh$Az.dec[hugh$diven == 1],ylim=c(-.5,1.5))
points(acc.d1$z.est,col='red')
lines(acc.d1$z.resid,col='blue')
legend(400,-.05,legend=c("Z-axis","Z-axis estimates","Residuals"),
       lty=c(0,0,1),pch=c(1,1,NA),col=c("black","red","blue"),bty="n")
plot(hugh$Az.dec[hugh$diven == 2],ylim=c(-.5,1.5))
points(acc.d2$z.est,col='red')
lines(acc.d2$z.resid,col='blue')
legend(300,-.05,legend=c("Z-axis","Z-axis estimates","Residuals"),
       lty=c(0,0,1),pch=c(1,1,NA),col=c("black","red","blue"),bty="n")
plot(hugh$Az.dec[hugh$diven == 3],ylim=c(-.5,1.5))
points(acc.d3$z.est,col='red')
lines(acc.d3$z.resid,col='blue')
legend(400,-.05,legend=c("Z-axis","Z-axis estimates","Residuals"),
       lty=c(0,0,1),pch=c(1,1,NA),col=c("black","red","blue"),bty="n")
```

`animalplot3d` *Make a 3D rotated plot of animal orientation based on roll, pitch, and yaw angles.*

Description

3D rendering of animal attitude using the "rgl" package.

Usage

```
animalplot3d(roll, pitch, yaw, angle = "degree", add = FALSE, ...)
```

Arguments

roll	Roll angle of the animal
pitch	Pitch angle of the animal
yaw	Yaw angle of the animal
angle	Unit of angular measure for input. Default is "degree", but the function will accept "radian". All three input angles must be either "degree" or "radian".
add	Indicates if the plot will be added to an existing rgl plot.
...	Additional items to be passed on to the rgl plotting function.

Details

This plotting function uses rgl plotting functions to produce a new 3D plot, or to add to an existing 3D plot. Plotting parameters will be passed on the the plot3d() function for customization. The stationary "body frame" which uses the "North-East-Down" (NED) frame of reference is plotted using the color red. Each of the three axes is plotted with small spheres on the end. The X-axis points posterior to anterior (i.e. direction the animal is facing) and faces north when not rotated. The Y-axis is oriented to the east of the X-axis (points left to right). The Z-axis is the dorso-ventral axis and points downward when not rotated (toward the center of the Earth). The rotated axes are also plotted in blue. The X-axis is given a large sphere so that it is easy to represent the direction that the animal is facing.

Value

An object with two rotated matrices

rotmat	3x3 rotation matrix
hrotmat	homogenous rotation matrix

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

See Also

plot3d

Examples

```
## Animal with a roll angle of 30 degrees, and a pitch angle of 10 degrees
roll <- 30
pitch <- 10
yaw <- 0

animalplot3d(roll,pitch,yaw,angle = "degree")

## Make the plot a bit larger
par3d(windowRect = c(50,50,700,700))
```

```
## Add another animal orientation with the same roll and pitch, but a 40 degree yaw.
animalplot3d(roll,pitch,40,angle = "degree", add=TRUE)
```

`calibrate.axes` *Calibrate two accelerometer/magnetometer axes. Axes are scaled and centered.*

Description

Hard iron distortion compass calibration. Hard iron error arises from magnetized iron or steel on the compass platform that creates a constant offset for the compass axes. This offset is additive and constant.

Usage

```
calibrate.axes(x, y, scale = 1, xcal = NULL, ycal = NULL)
```

Arguments

`x`	a numeric vector of axis values. This should be calibration rotations (i.e. yaw, pitch, roll)
`y`	a numeric vector of axis values. This should be calibration rotations (i.e. yaw, pitch, roll)
`scale`	the maximum value `x` and `y` will have after being scaled (i.e. values will range from `-scale` to `+scale`. Defaults to 1.
`xcal`	a numeric vector of axis values (corresponding to `x`). This should be track data that should be scaled according to the scale factor, offset, and `scale` variable. Optional.
`ycal`	a numeric vector of axis values (corresponding to `y`). This should be track data that should be scaled according to the scale factor, offset, and `scale` variable. Optional.

Details

A common source of compass error is hard and/or soft iron disturbance. Hard iron error arises from magnetized iron or steel on the compass platform that creates a constant offset for the compass axes. To account for the additive shift, an offset value is determined for each axis and applied by adding or subtracting the offset value from `x` or `y`. Soft iron error results from the interaction of soft iron near the compass and the earth's magnetic field. Soft iron error is a much more complicated distortion to account for and is not handled in this function.

Value

returns a list containing scale factors, offset values, and scaled axes

`xsf`	scale factor for x.
`ysf`	scale factor for y.
`xoff`	offset value for x.
`yoff`	offset value for y.
`xh`	calibrated x value. Scale factor and offset applied.
`yh`	calibrated y value. Scale factor and offset applied.
`xhscaled`	calibrated x value. Scale factor and offset applied, and scaled using `scale`.
`yhscaled`	calibrated y value. Scale factor and offset applied, and scaled using `scale`.
`xcal`	calibrated independent `xcal` value. Scale factor and offset applied to `xcal`, and scaled using `scale`. Optional. Will be NULL if `xcal` is not provided.
`ycal`	calibrated independent `ycal` value. Scale factor and offset applied to `ycal`, and scaled using `scale`. Optional. Will be NULL if `ycal` is not provided.

Warning

If any dynamic acceleration is present during the calibration procedure or track, it will result in an erroneous calibration. This can be avoided by keeping the sensor still and by pausing during rotations. Once the calibration constants are known for the instrument, they can be applied to the raw signal from the deployment on the animal. The raw accelerometer signal will include both static and dynamic acceleration, so the accelerometer signal must be filtered using a low-pass filter to extract the "static" portion of the signal.

Note

If no distortion is present, `plot(x,y)` will be circular and centered on (0,0). If hard-iron distortion is present, the center of the circle will not be (0,0), it will be shifted along the x, y, or both axes. However, the plot should still be circular. If the plot is not circular (e.g. ellipsoid), then soft-iron effects may be at play. In which case, a more complex calibration is needed. See Caruso (2000) for a discussion of compass and accelerometer errors. The hard-iron calibration used in this function is commonly implemented and can be found in Grygorenko (2011), among others.

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Caruso, M. J. (2000), Applications of magnetic sensors for low cost compass systems. Proc. Position Location and Navigation Symp., IEEE 2000, Mar. 2000, pp. 177-184

Grygorenko, V. (2011), Sensing - magnetic compass with tilt compensation. Cypress Perform, AN2272, Document No. 001-32379 Rev. B.

See Also

calibrate.axis

Examples

```
## Import the missionbay_calib2 dataset. There is a calibration sequence
## at Mission Bay, CA. Use help(missionbay_calib2) for info.

data(missionbay_calib2)

# Uncalibrated raw values
mx <- missionbay_calib2$mx_raw
my <- missionbay_calib2$my_raw

plot(mx,col='blue',type='l',lwd=3)
lines(my,col='green',lwd=3)

## Scale and center the x and y magnetometer axes against one another. Also, make
## sure that each axis is in the NED (north-east-down) orientation.
cal <- calibrate.axes(mx, my)
attributes(cal)

plot(cal$xhscaled,col='blue',type='l',lwd=3)
lines(cal$yhscaled,col='green',lwd=3)
```

---

calibrate.axis *Scale and center an accelerometer/magnetometer axis.*

Description

Scale and center `x` using the minimum and maximum values of the sensor. `scale` value indicates the maximum value `x` will have after being scaled (i.e. values will range `-scale` to `+scale`).

Usage

```
calibrate.axis(x, max, min, scale = 1)
```

Arguments

`x`	a numeric vector of axis values.
`max`	maximum uncalibrated value of the axis. This should be measured empirically through a series of calibration rotations. For a magnetometer axis, `max` should be the value of the Earth's "total field" at the locality (i.e. when the axis is aligned with the Earth's total field vector). The `min` value would conversely be the value when the axis is pointed in the opposite direction. The same logic applied if using accelerometer axis, but instead of the Earth's magnetic field, the gravity vector of the Earth is used (i.e. vector pointing toward the center of the Earth).

`min`	see `max`. value when axis is pointed opposite of reference vector (e.g. Earth magnetic or gravity field).
`scale`	the maximum value `x` will have after being scaled (i.e. values will range from `-scale` to `+scale`. Defaults to -1 to +1.

Details

This function is useful for calibrating the axes of magnetometers/accelerometers and correcting for hard-iron distortion. Centering and scaling each axis will standardize and essentially calibrate each axis so that the simulaneous measurements of each will be trigonometrically accurate. This function effectively deals with hard iron distortion problems as well as small misalignments between axes. However, there are other factors that can cause discontinuity (e.g. temperature, electronic noise, etc.) that are not accounted for in this function. For a more complete discussion of dealing with compass errors see Caruso (2000).

Value

object of scaled/centered values of `x`.

Warning

If any dynamic acceleration is present during the calibration procedure or track, it will result in an erroneous calibration. This can be avoided by keeping the sensor still and by pausing during rotations. Once the calibration constants are known for the instrument, they can be applied to the raw signal from the deployment on the animal. But remember, the raw accelerometer signal will contain both static and dynamic acceleration, so the accelerometer signal must be filtered using a low-pass filter to extract the static signal.

Note

If no distortion is present, `plot(x,y)` will be circular and centered on (0,0). If hard-iron distortion is present, the center of the circle will not be (0,0), it will be shifted along the x, y, or both axes. However, the plot should still be circular. If the plot is not circular (e.g. ellipsoid), then soft-iron effects may be at play. In which case, a more complex calibration is needed. See Caruso (2000) for a discussion of compass and accelerometer errors. The hard-iron calibration used in this function is commonly implemented and can be found in Grygorenko (2011), among others.

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Caruso, M. J. (2000), Applications of magnetic sensors for low cost compass systems. Proc. Position Location and Navigation Symp., IEEE 2000, Mar. 2000, pp. 177-184

See Also

`calibrate.axes`

Examples

```
## Import the missionbay_calib2 dataset. There is a calibration sequence
## at Mission Bay, CA. Use help(missionbay_calib2) for info.

data(missionbay_calib2)
mx <- missionbay_calib2$mx_raw
my <- missionbay_calib2$my_raw
mz <- missionbay_calib2$mz_raw

ax <- missionbay_calib2$ax_raw
ay <- missionbay_calib2$ay_raw
az <- missionbay_calib2$az_raw
## Scale and center the magnetometer axes. Also, make sure that each axis
## is in the NED (north-east-down) orientation (this is the reason for the
## negative signs).

Mxscaled <- calibrate.axis(mx, max(mx), min(mx))
Myscaled <- -calibrate.axis(my, max(my), min(my))
Mzscaled <- -calibrate.axis(mz, max(mz), min(mz))

## Scale and center the accelerometer axes. Also, make sure that each axis
## is in the NED (north-east-down) orientation.

Axscaled <- -calibrate.axis(ax, max(ax), min(ax))
Ayscaled <- calibrate.axis(ay, max(ay), min(ay))
Azscaled <- calibrate.axis(az, max(az), min(az))

## Compare
par(mfrow=c(2,3))
plot(mx,type="l",col="blue",xlab="time (s)",ylab="bits",main="Mx Raw",lwd=3)
plot(my,type="l",col="blue",xlab="time (s)",ylab="bits",main="My Raw",lwd=3)
plot(mz,type="l",col="blue",xlab="time (s)",ylab="bits",main="Mz Raw",lwd=3)
plot(Mxscaled,type="l",col="red",xlab="time (s)",ylab="bits",main="Mx Calibrated",lwd=3)
plot(Myscaled,type="l",col="red",xlab="time (s)",ylab="bits",main="My Calibrated",lwd=3)
plot(Mzscaled,type="l",col="red",xlab="time (s)",ylab="bits",main="Mz Calibrated",lwd=3)
```

---

dead_reckoning *Create a course steered and course made good using the principles of navigation by dead reckoning.*

Description

Calculate past, present, or future position using "dead reckoning" Bowditch (1995). Specifically, the function calculates the "course steered" and the "course made good" (if end position is known).

Usage

```
dead_reckoning(speed, heading, angle = "degree", ret = TRUE, depth = NULL,
                    pitch = NULL, endcoords=NULL, speedhorizontal = "corrected")
```

Arguments

speed	an object containing the values for measured speed (i.e. speed through water). (Note: this is assumed to be horizontal speed (i.e. no pitch), unless specified in the speedhorizontal argument.
heading	an object containing the values for measured heading. Can be one of c("degree", "radian").
angle	Unit of angular measure for heading. Default is "degree", but will accept "radian".
ret	Does the animal return to the same position as where it started? (i.e. (0,0))
depth	an object containing the values for measured depth (e.g pressure sensor, or altitude sensor).optional.
pitch	an object containing the values for calculated pitch. Both the pitch or pitch2 functions can be used. (Note: must be in radians). Optional.
endcoords	Coordinates for known location at the end of the track, c(x,y). If ret == F, what are the ending cartesian coordinates (i.e. (*x*,*y*))? This is the location where dead_reckoning will correct the course to based on a constant set and drift.(NOTE: ret must be FALSE).
speedhorizontal	Indicates how the input speed values should be handled. Default is "corrected", which indicates that the input speed values are horizontal speeds (i.e. speed across a plane tangent to the surface of the earth). If "pitch", then speed is assumed to be a 3D vector and will be corrected to horizontal using $speedh = cos(\phi) * \text{speed}$. If "depth", then speed is assumed to be a 3D vector and will be corrected to horizontal using $speedh = \sqrt{(speed^2 - \Delta depth^2)}$. Optional.

Details

See Bowditch (1995) for a complete discussion of dead reckoning and navigation. This function extends traditional navigation calculations by providing the flexibility for 3-dimensional parameters common in animal tracking data.

Value

Object of class navigate, which is a list with the following components:

CSx	a numeric vector with the x-coordinates for the calculated course steered. Position ascertained through "dead reckoning".
CSy	a numeric vector with the y-coordinates for the calculated course steered. Position ascertained through "dead reckoning".
CMGx	a numeric vector with the x-coordinates for the calculated course made good. Will be NA if end location is unknown (i.e. ret == F and endcoords = NULL. This is what Bowdith refers to as an "estimated position".)
CMGy	a numeric vector with the y-coordinates for the calculated course made goodWill be NA if end location is unknown (i.e. ret == F and endcoords = NULL. This is what Bowdith refers to as an "estimated position".)

speedh	speed horizontal. A numeric vector of corrected horizontal speed values. If speedhorizontal = "corrected" , then this will be the same as the input speed
speedmg	speed made good. A numeric vector of speed made good values. These are the speeds between observations after correcting the animal track for a course made good
drift	the speed of the current that resulted in the course steered. This is the distance between the course steered end location and the known (i.e. real) end location, divided by time. This represents the speed of the prevailing current, which is assumed to be constant
errordistance	This is the distance between the course steered end location and the known (i.e. real) end location
set	compass direction of the current. $0 <= \texttt{set} < 2\pi$

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Bowditch, N. (1995), *The New American Practical Navigator*. Bethesda, MD: Defense Mapping Agency Hydrographic Topographic Center.

Examples

```
## Import the "missionbay2" dataset. See help(missionbay2)
## for full documentation.
data(missionbay2)

trial.1 <- missionbay2[missionbay2$trial == 1,]
trial.2 <- missionbay2[missionbay2$trial == 2,]
trial.3 <- missionbay2[missionbay2$trial == 3,]
trial.4 <- missionbay2[missionbay2$trial == 4,]

## Calculate the course made good for the four trials. Each returns
## to the starting position.
CS1 <- dead_reckoning(trial.1$speed, trial.1$heading_geo,angle="radian")
CS2 <- dead_reckoning(trial.2$speed, trial.2$heading_geo,angle="radian")
CS3 <- dead_reckoning(trial.3$speed, trial.3$heading_geo,angle="radian")
CS4 <- dead_reckoning(trial.4$speed, trial.4$heading_geo,angle="radian")

## Plot the course steered for each trial.
plot(CS1$CSx,CS1$CSy,type='l',col='blue',xlab="X-coordinate (unprojected)",
     ylab="Y-coordinate (unprojected)",ylim=c(-400,150))
lines(CS2$CSx,CS2$CSy,col='green')
lines(CS3$CSx,CS3$CSy,col='red')
lines(CS4$CSx,CS4$CSy,col='magenta')
legend(-300,100,legend=c("Run1","Run2","Run3","Run4"),col=c("blue","green",
       "red","magenta"),lty=c(1,1,1,1),bty="n")
title('Course Steered for Mission Bay Trials')
```

```
grid()

## Plot the course steered vs. course made good
plot(CS1$CSx,CS1$CSy,type='l',col='blue',xlab="X-coordinate (unprojected)",
     ylab="Y-coordinate (unprojected)",ylim=c(-400,150))
lines(CS1$CMGx,CS1$CMGy,col='black')
t.set <- paste("Track 1, Set Angle: ",as.character(round(CS1$set*(180/pi),2)))
t.drift <- paste("Track 1, Drift: ",as.character(round(CS1$drift,2))," m/s")
t.error <- paste("Track 1, Error Distance: ",as.character(round(CS1$errordistance,2))," m")
title(paste(t.set,"\n",t.drift,"\n",t.error))
legend(-300,100,legend=c("Course Steered","Course Made Good"),
       col=c("blue","black"),lty=c(1,1),bty="n")
grid()
```

---

hugh *Seal "hugh" dive data.*

Description

A sample of recorded field data from a seal. The seal makes 6 dives with 5 surface intervals. Data have been pre-filtered but are in somewhat of an intermediate form for the purposes of examples within the `animalTrack` package.

Usage

```
data(hugh)
```

Format

A data frame with 13,953 observations on the following 21 variables.

`Year` year

`Month` month

`Day` day

`Hour` hour

`Minute` minute

`Second` second

`MX` a numeric vector of magnetometer X-axis values (in bits)

`MY` a numeric vector of magnetometer Y-axis values (in bits)

`MZ` a numeric vector of magnetometer Z-axis values (in bits)

`depth.cal` depth (meters)

`MX2` a numeric vector of temperature calibrated magnetometer x-axis values (in bits)

`MY2` a numeric vector of temperature calibrated magnetometer Y-axis values (in bits)

`MZ2` a numeric vector of temperature calibrated magnetometer Z-axis values (in bits)

`MXone` a numeric vector of temperature calibrated magnetometer X-axis values scaled from -1 to +1

`MYone` a numeric vector of temperature calibrated magnetometer Y-axis values scaled from -1 to +1

`MZone` a numeric vector of temperature calibrated magnetometer Z-axis values scaled from -1 to +1

`speed.cal` calibrated speed (meters per second)

`Ax.dec` a numeric vector of calibrated accelerometer x-axis values scaled from -1 to +1g. A low-pass filter was applied in order to obtain "static" acceleration.

`Ay.dec` a numeric vector of calibrated accelerometer Y-axis values scaled from -1 to +1g. A low-pass filter was applied in order to obtain "static" acceleration.

`Az.dec` a numeric vector of calibrated accelerometer Z-axis values scaled from -1 to +1g. A low-pass filter was applied in order to obtain "static" acceleration.

`diven` dive number

Source

Data were collected under NSF Grant #0739390 and NOAA Marine Mammal Permit #984-1814.

Examples

```
data(hugh)
```

---

missionbay2 *Mission Bay navigation trials.*

Description

Data from calibrated field trials in Mission Bay, CA.

Usage

```
data(missionbay2)
```

Format

A data frame with 3,020 observations on the following 27 variables.

`date` a Date

`time` a POSIXlt

`year` year

`month` month

`day` day

`hour` hour

`minute` minute

`second` second

`speed` calibrated speed (meters per second)

`depth` depth (meters)

`mx_raw` a numeric vector of magnetometer X-axis values (in bits)

`my_raw` a numeric vector of magnetometer Y-axis values (in bits)

`mz_raw` a numeric vector of magnetometer Z-axis values (in bits)

`ax_raw` a numeric vector of accelerometer X-axis values (in bits)

`ay_raw` a numeric vector of accelerometer Y-axis values (in bits)

`az_raw` a numeric vector of accelerometer Z-axis values (in bits)

`mx_scaled` a numeric vector of calibrated magnetometer X-axis values scaled from -1 to +1

`my_scaled` a numeric vector of calibrated magnetometer Y-axis values scaled from -1 to +1

`mz_scaled` a numeric vector of calibrated magnetometer Z-axis values scaled from -1 to +1

`ax_scaled` a numeric vector of calibrated accelerometer X-axis values scaled from -1 to +1

`ay_scaled` a numeric vector of calibrated accelerometer Y-axis values scaled from -1 to +1

`az_scaled` a numeric vector of calibrated accelerometer Z-axis values scaled from -1 to +1

`roll` roll values in radians ($-\pi$ to $+\pi$)

`pitch` pitch values in radians ($-\pi/2$ to $+\pi/2$)

`heading_mag` magnetic heading direction in radians ($0 - 2\pi$)

`heading_geo` geographic heading (corrected for declination) direction in radians ($0 - 2\pi$)

`trial` trial number, 1-4

Details

Four trials were conducted in which an instrument that was recording speed, magnetic fields, acceleration, depth, and time was towed behind a boat to a series of known locations. During each trial, the boat returned to the starting location after a short trip.

Source

These data were recorded with great seamanship by Bill Hagey and Randall Davis

Examples

```
data(missionbay2)
```

missionbay_calib2 *Magnetometer/accelerometer calibration data.*

Description

Empirical magnetometer/accelerometer calibration data from Mission Bay, CA. These data are a sequence of 2 clockwise yaw rotations, 2 pitch rotations (nose down), and 2 roll rotations (nose facing west, starboard down).

Usage

```
data(missionbay_calib2)
```

Format

A data frame with 275 observations on the following 15 variables.

`date` a Date

`time` a POSIXlt

`year` a numeric vector

`month` a numeric vector

`day` a numeric vector

`hour` a numeric vector

`minute` a numeric vector

`second` a numeric vector

`mx_raw` a numeric vector. Magnetometer x-axis value in bits

`my_raw` a numeric vector. Magnetometer y-axis value in bits

`mz_raw` a numeric vector. Magnetometer z-axis value in bits

`ax_raw` a numeric vector. Accelerometer x-axis value in bits

`ay_raw` a numeric vector. Accelerometer y-axis value in bits

`az_raw` a numeric vector. Accelerometer z-axis value in bits

`rotation` a character vector. One of `c("yaw1: clockwise", "yaw2: clockwise","pitch1: nose down", "pitch2: nose down", "roll1: starboard down","roll2: starboard down")`

Details

These data are raw bit values from a three-axis magnetometer and accelerometer. See the `$rotation` column for when each rotation starts and ends.

Source

These data were recorded by Bill Hagey and Randall Davis.

Examples

```
data(missionbay_calib2)
```

`pitch` *Calculate pitch angle.*

Description

calculate pitch angle using measurements from a 3-axis accelerometer.

Usage

```
pitch(x, y, z)
```

Arguments

x	a numeric vector of x-axis values.
y	a numeric vector of y-axis values.
z	a numeric vector of z-axis values.

Details

`pitch()` will return a pitch angle (in radians) representing the posterior-anterior axis position of the animal. This angle is the difference between the posterior-anterior axis (vector) of the animal and the horizon. This angle is calculated assuming the NED (north, east, down) frame of reference. Typically, this is measured with an accelerometer and is scaled from -1 to +1. Using NED means that when the accelerometer is measuring 1g, this is equal to -9.8 m/s^2. This also means that if the axis is aligned with the earth gravity vector (i.e. down, towards the center of the earth), it should have a value of +1. Pitch angle is calculated using

$$pitch = -atan(x/\sqrt{(y^2 + z^2)})$$

.

A positive pitch value indicates that the nose of the animal (x-axis) is pitched upward (toward the sky). Conversely, a negative pitch angle indicates that the animal is pitched downward. (Note: axis values must be converted to NED frame of reference prior to using this function)

Value

an object of pitch values (in radians) from $-\pi/2 \leq$ `pitch` $\leq \pi/2$

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Tuck, K. (2007), Tilt sensing using linear accelerometers. Freescale Semiconductor, AN3461 Rev. 2.

Smith, K. J. (1998), *Essentials of Trigonometry*. Pacific Grove, CA: Brooks/Cole.

See Also

pitch2 atan

Examples

```
## Import the yaw, pitch and roll simulated calibration dataset. For
## an explanation of the data use help(yprsim).
data(yprsim)

## Calculate pitch (the nose/x-axis is rotated downward through 2 full rotations)
phi <- pitch(yprsim$ax,yprsim$ay,yprsim$az)

## Plot
plot(phi*(180/pi),type='l',lty=1,lwd=2,xlab="time (s)",ylab="pitch (degrees)",
     main="Pitch Calculation (2 pitch rotations)")
abline(v=c(126,252),lty=3,lwd=2)
legend(-10,70,legend=c("Pitch","Change in \n Rotation"),col=c("black","black"),
       lty=c(1,3),bty="n")
text(50,-70,"Yaw");text(175,-70,"Pitch");text(320,-70,"Roll")
```

---

pitch2 *Calculate pitch angle.*

Description

calculate pitch angle using measurements from a three-axis accelerometer and roll angle.

Usage

```
pitch2(x, y, z, roll)
```

Arguments

x	a numeric vector of x-axis values.
y	a numeric vector of y-axis values.
z	a numeric vector of z-axis values.
roll	numeric vector of roll angles. ($-\pi \leq$ roll $\leq \pi$)

Details

`pitch2()` will return a pitch angle (in radians) representing the posterior-anterior axis position of the animal. This angle is the difference between the posterior-anterior axis (vector) of the animal and the horizon. This angle is calculated assuming the NED (north, east, down) frame of reference. Typically, this is measured with an accelerometer and is scaled from -1 to +1. Using the NED frame of reference means that when the accelerometer is measuring 1g, this is equal to -9.8 m/s^2. Also, if the axis is aligned with the earth gravity vector (i.e. down, towards the center of the earth), it should have a value of +1. Pitch angle is calculated by

$$pitch = atan(-x/(y * sin(roll) + (z * cos(roll))))$$

A positive pitch value indicates that the nose of the animal (x-axis) is pitched upward (toward the sky). Conversely, a negative pitch angle indicates that the animal is pitched downward. (Note: axis values must be converted to NED frame of reference prior to using this function)

Value

an object of pitch values (in radians) from $-\pi/2 \leq \texttt{pitch} \leq \pi/2$

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Ozyagcilar, T. (2012), Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. Freescale semiconductor, AN 4248, Rev. 3.

Smith, K. J. (1998), *Essentials of Trigonometry*. Pacific Grove, CA: Brooks/Cole.

See Also

`pitch` `atan`

Examples

```
## Import the yaw, pitch and roll simulated calibration dataset. For
## an explanation of the data use help(yprsim).
data(yprsim)

## Accelerometer axes
ax <- yprsim$ax
ay <- yprsim$ay
az <- yprsim$az

## Calculate roll
theta <- roll(ay, az)

## Calculate pitch (the nose/x-axis is rotated downward through 2 full rotations)
phi <- pitch2(ax, ay, az, theta)
```

```
## Plot
plot(phi*(180/pi),type='l',lty=1,lwd=2,xlab="time (s)",ylab="pitch (degrees)",
     main="Pitch2 Calculation (2 pitch rotations)")
abline(v=c(126,252),lty=3,lwd=2)
legend(-10,70,legend=c("Pitch","Change in \n Rotation"),
       col=c("black","black"),lty=c(1,3),bty="n")
text(50,-70,"Yaw");text(175,-70,"Pitch");text(320,-70,"Roll")
```

`roll` *Calculate roll angle.*

Description

calculate roll angle using measurements from a three-axis accelerometer. However, only two axes are required.

Usage

```
roll(y, z)
```

Arguments

y	a numeric vector of y-axis values.
z	a numeric vector of z-axis values.

Details

Roll angle is the angle between the left-right axis and the horizon. The angle is calculated assuming the NED (north, east, down) frame of reference, where x is north, y is east, and z is down. Typically, this is measured with an accelerometer and is scaled from -1 to +1. Using NED, this means that when the accelerometer is measuring 1g, this is equal to -9.8 m/s^2. This also means that if the axis is aligned with the earth gravity vector (i.e. down, towards the center of the earth), it should have a value of +1. Roll angle is calcualted by

$roll = atan2(y, z)$

A downward movement of the y-axis gives a positive roll angle.

Value

an object of roll values (in radians) from $-\pi \leq$ `roll` $\leq \pi$

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Ozyagcilar, T. (2012), Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. Freescale semiconductor, AN 4248, Rev. 3.

Smith, K. J. (1998), *Essentials of Trigonometry*. Pacific Grove, CA: Brooks/Cole.

Examples

```
## Import the yaw, pitch and roll simulated calibration dataset. For
## an explanation of the data use help(yprsim).
data(yprsim)

## Assign accelerometer variables
ay <- yprsim$ay
az <- yprsim$az

## Calculate roll
theta <- roll(ay, az)

plot(theta*(180/pi),type='l',lty=1,lwd=2,xlab="time (s)",ylab="pitch (degrees)",
     main="Roll Calculation (2 roll rotations)")
abline(v=c(126,252),lty=3,lwd=2)
legend(-10,110,legend=c("Roll","Change in \n Rotation"),
       col=c("black","black"),lty=c(1,3),bty="n")
text(50,-70,"Yaw");text(175,-70,"Pitch");text(320,-70,"Roll")
```

`tilt_compensate` *Tilt compensated compass.*

Description

calculate tilt compensated compass heading using three-axis magnetometer and accelerometer. Magnetometer and accelerometer must be in NED frame of reference.

Usage

```
tilt_compensate(x, y, z, pitch, roll, declination = 0, angle = "degree")
```

Arguments

`x`	a numeric vector of magnetic x-axis values (e.g. anterior axis of animal)
`y`	a numeric vector of magnetic y-axis values (e.g. starboard axis of animal)
`z`	a numeric vector of magnetic z-axis values (e.g. downward/ventral axis of animal)
`pitch`	a numeric vector of pitch angle values. values must range from $-\pi/2 \leq$ `pitch` $\leq \pi/2$. (NOTE: pitch must be calculated using a static acceleration signal.)

`roll`	a numeric vector of roll angle values. values must range from $-\pi \leq$ `roll` $\leq \pi$. (NOTE: pitch must be calculated using a static acceleration signal.)
`declination`	magnetic declination angle at animal track locality.
`angle`	unit of angular measurement of `pitch, roll, declination`

Details

Accurate compass heading is crucial when reconstructing an animal's 3D (or 2D) track through space, whether in the water or through the air. One of the goal's of deploying high frequency bio-logging tags on large fauna is to relate physiological parameters to geolocation. Tags that are equipped with a three-axis accelerometer and three-axis magnetometer can accurately measure the the attitude and movement of an animal. Each tag usually has a unique configuration for how the magnetometer and accelerometer are placed in the instrument, so it may be necessary to rotate the sensor outputs so that they are in the standard north-east-down (NED) frame of reference (also referred to as the "body frame", common in aviation).

The need for tilt compensation arises because free-ranging animals are not always oriented parallel to the horizon while moving or stationary. When a compass is tilted, the basic trigonometric functions for calculating heading using the x and y axes fail. In order to correct for this, the x and y values are re-projected back to the horizontal plane using Euler angles. The arctan(yh/xh) is then used to solve for the compass angle on the horizontal plane. For a proof of the tilt compensation equation see Ozyagcilar (2012) or Grygorenko (2011). Furthermore, a practical application of tilt compensated heading as well as a proof can be found in Johnson & Tyack (2003).

Value

Object of class `tiltcompensate`, which is a list with the following components:

`xh`	numeric vector of tilt compensated values of magnetic x-axis.
`yh`	numeric vector of tilt compensated values of magnetic y-axis.
`heading_mag`	numeric vector of compass heading values (in radians).
`heading_geo`	numeric vector of compass heading values corrected for declination angle (in radians). If no declination angle is given, this will be the same as `heading_mag`.

Note

Geomagnetic declination can be found at `http://www.ngdc.noaa.gov/`. Since the earth's magnetic field is dynamic, the declination value should be at the date and location of the study area.

Pitch and roll must also be calculated in the NED frame of reference.

Author(s)

Ed Farrell <edward.farrell27@gmail.com>

References

Cai, G., Chen, B. M., and T. H. Lee (2011). Coordinate systems and transformations. In *Unmanned Rotocraft Systems* (pp. 23-34). Springer, New York, NY.

Grygorenko, V. (2011), Sensing - magnetic compass with tilt compensation. Cypress Perform, AN2272, Document No. 001-32379 Rev. B.

Johnson, M. P. & P. L. Tyack (2003), A digital acoustic recording tag for measuring the response of wild animals to sound. IEEE Journal of Ocean Engineering, **28**, 3:12.

Ozyagcilar, T. (2012), Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. Freescale semiconductor, AN 4248, Rev. 3.

See Also

pitch, roll, calibrate.axis, calibrate.axes

Examples

```
## Import the yprsim dataset. This is a series of 2 yaws, 2 pitches,
## and 2 rolls. It is a simulated calibration Sequence. For details see
## help(yprsim)

data(yprsim)

## assign variables

mx <- yprsim$mx
my <- yprsim$my
mz <- yprsim$mz

ax <- yprsim$ax
ay <- yprsim$ay
az <- yprsim$az

## The magnetometer and accelerometer data are already scaled and centered in the
## NED frame. Also, there is no magnetic inclination simulated in this dataset.
## So, this calibration simulation would be valid at the magnetic equator.

pitch <- pitch(ax, ay, az)
roll <- roll(ay, az)

## calculate tilt compensated heading. with no declination.

tilt <- tilt_compensate(mx, my, mz, pitch, roll,
                        declination = 0, angle = "radian")

## Check the list of variables that is produced.
attributes(tilt)

## Plot
plot(tilt$heading_mag*(180/pi),col='red',type="l",lwd=2,ylim=c(-180,360),
```

```
     xlab="time (s)", ylab="degrees",main="Simulated Heading,
       Yaw, Pitch, and Roll")
lines(pitch*(180/pi),col='black',type='l',lty=1,lwd=2)
lines(roll*(180/pi),col='blue',lty=2,lwd=2)
abline(v=c(126,252),lty=3,lwd=2)
legend(0,-50,legend=c("Heading","Pitch","Roll"),col=c("red","black","blue"),
       lty=c(1,1,2),bty="n")

## Create a series of 3d plots, that can be saved to a
## file directory. These can then be used for animation.
## Not run:
  ## make sure that the "rgl" library is installed

  filename <- "~Rdir/"
  # Make an animated 3d plot of all the rotations

  for (i in 1:(length(yprsim[,1]))){
    ii <- roll[i]; jj <- pitch[i]; kk <- tilt$heading[i]
    animalplot3d(ii, jj, kk,angle = "radian")
    par3d(windowRect = c(50,50,700,700)) # make the window a bit larger.
    #save each plot as a png
    rgl.snapshot(filename=paste(filename,as.character(i),".png",sep=""),fmt="png")
  }

## End(Not run)
```

`yprsim` *Yaw, pitch, and roll simulation data.*

Description

Simulated calibration sequence with 2 yaw, 2 pitch, and 2 roll rotations. No magnetic inclination is assumed. The yaw rotation is clockwise, with the x-axis facing north at the beginning and end of the rotations. The pitch rotation is a downward pitch with the x-axis facing north at the beginning. The roll rotation is a downward rotation with the x-axis facing west at the beginning.

Usage

```
data(yprsim)
```

Format

A data frame with 378 observations on the following 11 variables.

`mx` a numeric vector of magnetometer X-axis values.

`my` a numeric vector of magnetometer Y-axis values.

`mz` a numeric vector of magnetometer Z-axis values.

`ax` a numeric vector of accelerometer X-axis values.

`ay` a numeric vector of accelerometer Y-axis values.

`az` a numeric vector of accelerometer Z-axis values.

`pitch` pitch values (in radians from $-\pi/2 \leq$ to $\leq \pi/2$)

`roll` roll values (in radians from $-\pi \leq$ to $\leq \pi$)

`heading` heading values (in radians from 0 to 2π)

`xh` tilt corrected X-axis magnetometer values (in radians)

`yh` tilt corrected Y-axis magnetometer values (in radians)

Examples

```
data(yprsim)
```

Index