Package 'ipft'

February 21, 2017

Type Package

Title Indoor Positioning Fingerprinting Toolset

Depends R (>= 2.10)

Version 0.2.2

Author Emilio Sansano

Maintainer Emilio Sansano <esansano@uji.es>

Description Algorithms and utility functions for indoor positioning using fingerprinting techniques. These functions are designed for manipulation of RSSI (Received Signal Strength Intensity) data sets, estimation of positions,comparison of the performance of different models, and graphical visualization of data. Machine learning algorithms and methods such as k-nearest neighbors or probabilistic fingerprinting are implemented in this package to perform analysis and estimations over RSSI data sets.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

LinkingTo Rcpp

Imports Rcpp, methods, stats, apcluster, cluster, dplyr, ggplot2

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-02-21 16:51:47

R topics documented:

`ipfCluster` *Creates clusters using the specified method*

Description

This function creates clusters using the the specified method and assigns a cluster id to each cluster

Usage

```
ipfCluster(data, method = "k-means", k = NULL, ...)
```

Arguments

`data`	a data frame
`method`	the method to use to clusterize the data. Implemented methods are: 'k-means' for k-means algorithm. Requires parameter k. 'AP' for affinity propagation algorithm.
`k`	parameter k
`...`	additional parameters for apcluster and apclusterK

Value

A list with: clusters -> a numeric vector with the ids of the clusters centers -> a data frame with the centers of the clusters

Examples

```
clusters <- ipfCluster(head(ipftrain, 20)[, 169:170], k = 4)

clusters <- ipfCluster(head(ipftrain[, grep('^wap', names(ipftrain))], 20),
method = 'AP')$clusters
```

ipfDist *Distance function*

Description

This function computes the distance from every observation in the test set to every observation in the train test

Usage

```
ipfDist(train, test, method = "euclidean", subset = NULL, norm = 2,
  sd = 10, epsilon = 1e-30, alpha = 20, threshold = 20)
```

Arguments

train	a vector, matrix or data frame containing a set of training examples
test	a vector, matrix or data frame containing a set of test examples
method	The method to be used to calculate the distance. Implemented methods are: 'euclidean', 'manhattan', 'norm', 'LGD' and 'PLGD'
subset	columns to use to compute the distance.
norm	parameter for the 'norm' method
sd	parameter for 'LGD' and 'PLGD' methods
epsilon	parameter for 'LGD' and 'PLGD' methods
alpha	parameter for 'PLGD' method
threshold	parameter for 'PLGD' method

Value

This function returns a matrix with dimensions: nrow(test) x nrow(train), containing the distances from test observations to train observations

Examples

```
dist <- ipfDist(ipftrain[,1:168], ipftest[,1:168])

dist <- ipfDist(ipftrain, ipftest, subset = seq(1,168))

dist <- ipfDist(ipftrain, ipftest, subset = c('LONGITUDE', 'LATITUDE'), method = 'manhattan')
```

ipfEstimate *This function estimates the location of the test observations*

Description

This function estimates the location of the test observations

Usage

```
ipfEstimate(ipfmodel, locdata, loctest = NULL)
```

Arguments

ipfmodel	an ipfModel
locdata	a matrix or a data frame containing the position of the training set observations
loctest	a matrix or a data frame containing the position of the test set observations

Value

An S4 class object of type ipfEstimation, with the following slots: location -> a matrix with the predicted locations groupploc -> a matrix with the location data for each group errors -> a numeric vector with the errors

Examples

```
model <- ipfKnn(ipftrain[, 1:168], ipftest[, 1:168])
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])

groups <- ipfGroup(ipftrain, LONGITUDE, LATITUDE)
model <- ipfProb(ipftrain[, 1:168], ipftest[, 1:168], groups, k = 9, delta = 10)
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])
```

ipfGroup *Creates groups based on the specified parameters*

Description

This function groups the data based on the specified variables and assigns an id to each group

Usage

```
ipfGroup(data, ...)
```

Arguments

data	A data frame
...	Variables to group by. All variables (columns) will be used if no parameter is provided.

Value

A numeric vector with the ids of the groups, in the same order as they appear in the data provided.

Examples

```
group <- ipfGroup(mtcars, cyl)

group <- ipfGroup(mtcars, gear, carb)

group <- ipfGroup(ipftrain, LONGITUDE, LATITUDE)
```

---

ipfKnn *This function implements the k-nearest neighbors algorithm*

Description

This function implements the k-nearest neighbors algorithm

Usage

```
ipfKnn(train, test, k = 3, method = "euclidean", norm = 2, sd = 5,
  epsilon = 0.001, alpha = 1, threshold = 20, FUN = NULL, ...)
```

Arguments

train	a data frame containing the RSSI vectors of the training set
test	a data frame containing the RSSI vectors of the test set
k	the k parameter for knn algorithm (number of nearest neighbors)
method	the method to compute the distance between the RSSI vectors: 'euclidean', 'manhattan', 'norm', 'LGD' or 'PLGD'
norm	parameter for the 'norm' method
sd	parameter for 'LGD' and 'PLGD' methods
epsilon	parameter for 'LGD' and 'PLGD' methods
alpha	parameter for 'PLGD' method
threshold	parameter for 'PLGD' method
FUN	an alternative function provided to compute the distance. This function must return a matrix of dimensions: nrow(test) x nrow(train), containing the distances from test observations to train observations
...	additional parameters for provided function FUN

Value

An S4 class object of type ipfModel, with the following slots: neighbors -> a matrix with k columns and nrow(test) rows, with the k nearest neighbors for each test observation weights -> a matrix with k columns and nrow(test) rows, with the weight for each neighbour distances -> a matrix with k columns and nrow(test) rows, with the distances between test and each neighbour k -> k parameter groups -> the group index for each training observation

Examples

```
model <- ipfKnn(ipftrain[, 1:168], ipftest[, 1:168])

model <- ipfKnn(ipftrain[, 1:168], ipftest[, 1:168], k = 9, method = 'manhattan')
```

`ipfPlotEcdf` *Plots the cumulative distribution function of the estimated error*

Description

Plots the cumulative distribution function of the estimated error

Usage

```
ipfPlotEcdf(estimation, xlab = "error",
  ylab = "cumulative density of error", title = "")
```

Arguments

`estimation`	an ipfEstimation
`xlab`	x-axis label
`ylab`	y-axis label
`title`	plot title

Examples

```
model <- ipfKnn(ipftrain[, 1:168], ipftest[, 1:168])
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])
ipfPlotEcdf(estimation)

groups <- ipfGroup(ipftrain, LONGITUDE, LATITUDE)
model <- ipfProb(ipftrain[, 1:168], ipftest[, 1:168], groups, k = 9, delta = 10)
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])
ipfPlotEcdf(estimation, title = 'Error cumulative distribution function')
```

ipfPlotEst *Plots the estimated locations*

Description

Plots the estimated locations

Usage

```
ipfPlotEst(model, estimation, testloc = NULL, observations = c(1),
  reverseAxis = FALSE, showneighbors = FALSE, showLabels = FALSE,
  xlab = NULL, ylab = NULL, title = "")
```

Arguments

model	an ipfModel
estimation	an ipfEstimation
testloc	location of test observations
observations	a numeric vector with the indices of estimations to plot
reverseAxis	swaps axis
showneighbors	plot the k selected neighbors
showLabels	shows labels
xlab	x-axis label
ylab	y-axis label
title	plot title

Examples

```
model <- ipfKnn(ipftrain[, 1:168], ipftest[, 1:168])
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])
ipfPlotEst(model, estimation, ipftest[, 169:170], observations = seq(7,10),
           showneighbors = TRUE, reverseAxis = TRUE)
```

ipfPlotLoc *Plots the spatial location of the observations*

Description

Plots the spatial location of the observations

Usage

```
ipfPlotLoc(locdata, plabel = FALSE, reverseAxis = FALSE, xlab = NULL,
  ylab = NULL, title = "", pgrid = FALSE)
```

Arguments

locdata	a data frame or matrix with the positions
plabel	if TRUE, adds labels to groups / observations
reverseAxis	swaps axis
xlab	x-axis label
ylab	y-axis label
title	plot title
pgrid	plot grid (boolean)

Examples

```
ipfPlotLoc(ipftrain[, 169:170])

ipfPlotLoc(ipftrain[, 169:170], plabel = TRUE, reverseAxis = TRUE,
           title = 'Position of training set observations')
```

ipfPlotPdf *Plots the probability density function of the estimated error*

Description

Plots the probability density function of the estimated error

Usage

```
ipfPlotPdf(estimation, xlab = "error", ylab = "density", title = "")
```

Arguments

estimation	an ipfEstimation
xlab	x-axis label
ylab	y-axis label
title	plot title

Examples

```
model <- ipfKnn(ipftrain[, 1:168], ipftest[, 1:168])
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])
ipfPlotPdf(estimation)

groups <- ipfGroup(ipftrain, LONGITUDE, LATITUDE)
model <- ipfProb(ipftrain[, 1:168], ipftest[, 1:168], groups, k = 9, delta = 10)
estimation <- ipfEstimate(model, ipftrain[, 169:170], ipftest[, 169:170])
ipfPlotPdf(estimation, title = 'Probability density function')
```

---

ipfProb *This function implements a probabilistic algorithm*

Description

This function implements a probabilistic algorithm

Usage

```
ipfProb(train, test, groups, k = 3, FUN = sum, delta = 1, ...)
```

Arguments

train	a data frame containing the RSSI vectors of the training set
test	a data frame containing the RSSI vectors of the test set
groups	a numeric vector of length = nrow(train) containing the group index for the training vectors
k	the k parameter for the algorithm (number of similar neighbors)
FUN	function to compute the similarity measurement. Default is 'sum'
delta	parameter delta
...	additional parameters for provided function FUN

Value

An S4 class object of type ipfModel, with the following slots: neighbors -> a matrix with k columns and nrow(test) rows, with the k most similar training observation for each test observation weights -> a matrix with k columns and nrow(test) rows, with the weights distances -> a matrix with k columns and nrow(test) rows, with the distances k -> k parameter groups -> the group index for each training observation

Examples

```
groups <- ipfGroup(ipftrain, LONGITUDE, LATITUDE)
model <- ipfProb(ipftrain[, 1:168], ipftest[, 1:168], groups)

groups <- ipfGroup(ipftrain, LONGITUDE, LATITUDE)
model <- ipfProb(ipftrain[, 1:168], ipftest[, 1:168], groups, k = 9, delta = 10)
```

`ipftest` *Indoor localization test data set to test Indoor Positioning System that rely on WLAN/WiFifingerprint. It was created during the Fingerprinting-based Indoor Positioning tutorial of the seventh international conference on indoor Positioning and Indoor Navigation (IPIN2016).*

Description

Indoor localization test data set to test Indoor Positioning System that rely on WLAN/WiFifingerprint. It was created during the Fingerprinting-based Indoor Positioning tutorial of the seventh international conference on indoor Positioning and Indoor Navigation (IPIN2016).

Usage

```
ipftest
```

Format

A data frame with columns:

wap1, wap2, wap3, wap4, wap5, wap6, wap7, wap8, wap9, wap10, wap11, wap12, wap13, wap14, wap15, wap16, wap
Intensity value for WAPs. Negative integer values from -99 to 0. Value 0 is used if WAP was not detected.

LONGITUDE Longitude in meters relative to the origin of a predefined coordinate system. From -0.60 to 4.39

LATITUDE Latitude in meters relative to the origin of a predefined coordinate system. From 0.00 to 30.42

FLOOR All the records of this dataset have been captured in the same floor. Therefore, the floor attribute is 0 to all the records.

BUILDINGID All the records of this dataset have been captured in the same building. Therefore, the building attribute is 0 to all the records.

SPACEID Internal ID number to identify the position at where the capture was taken.

USERID User identifier. Students created the train dataset (UserID from 1 to 8), and professors the test one (UserID is 0 in this case).

PHONEID All the records have 0 in this attribute. This attribute is not used in this dataset.

TIMESTAMP UNIX Time when the capture was taken.

Source

UJI - Institute of New Imaging Technologies, Universitat Jaume I, Avda. Vicente Sos Baynat S/N, 12071, Castellón, Spain. http://www.init.uji.es/

Examples

```
## Not run:
 ipftest

## End(Not run)
```

`ipftrain` *Indoor localization training data set to test Indoor Positioning System that rely on WLAN/WiFifingerprint. It was created during the Fingerprinting-based Indoor Positioning tutorial of the seventh international conference on indoor Positioning and Indoor Navigation (IPIN2016).*

Description

Indoor localization training data set to test Indoor Positioning System that rely on WLAN/WiFifingerprint. It was created during the Fingerprinting-based Indoor Positioning tutorial of the seventh international conference on indoor Positioning and Indoor Navigation (IPIN2016).

Usage

```
ipftrain
```

Format

A data frame with columns:

wap1, wap2, wap3, wap4, wap5, wap6, wap7, wap8, wap9, wap10, wap11, wap12, wap13, wap14, wap15, wap16, wap
Intensity value for WAPs. Negative integer values from -99 to 0. Value 0 is used if WAP was not detected.

LONGITUDE Longitude in meters relative to the origin of a predefined coordinate system. From -0.60 to 4.39

LATITUDE Latitude in meters relative to the origin of a predefined coordinate system. From 0.00 to 30.42

FLOOR All the records of this dataset have been captured in the same floor. Therefore, the floor attribute is 0 to all the records.

BUILDINGID All the records of this dataset have been captured in the same building. Therefore, the building attribute is 0 to all the records.

SPACEID Internal ID number to identify the position at where the capture was taken.

USERID User identifier. Students created the train dataset (UserID from 1 to 8), and professors the test one (UserID is 0 in this case).

PHONEID All the records have 0 in this attribute. This attribute is not used in this dataset.

TIMESTAMP UNIX Time when the capture was taken.

Source

UJI - Institute of New Imaging Technologies, Universitat Jaume I, Avda. Vicente Sos Baynat S/N, 12071, Castellón, Spain. http://www.init.uji.es/

Examples

```
## Not run:
 ipftrain

## End(Not run)
```

---

ipfTransform *Transform function*

Description

This function transforms the RSSI (Received Signal Strength Intensity) data to positive or exponential values

Usage

```
ipfTransform(data, trans = "positive", minRSSI = -104, maxRSSI = 0,
  noRSSI = 0, alpha = 24)
```

Arguments

`data`	a vector, matrix or data frame containing the RSSI vectors
`trans`	the transformations to perform
`minRSSI`	the minimum value for RSSI to consider when transforming the RSSI to positive values.
`maxRSSI`	the maximum value for RSSI to consider when transforming the RSSI to exponential values.
`noRSSI`	value used in the RSSI data to represent a not detected AP.
`alpha`	parameter for exponential transformation

Value

This function returns a vector, matrix or data frame containing the transformed data

Examples

```
trainRSSI <- ipftrain[,1:168]
ipfTransform(trainRSSI, trans = 'positive')

trainRSSI <- ipftrain[,1:168]
posTrainRSSI <- ipfTransform(trainRSSI, trans = 'positive')
expTrainRSSI <- ipfTransform(posTrainRSSI, trans = 'exponential', maxRSSI = 104)
```

Index