Package ‘ipft’
January 4, 2018

Type Package
Title Indoor Positioning Fingerprinting Toolset
Depends R (>= 2.10)
Version 0.7.2
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
Author Emilio Sansano [aut, cre], Raúl Montoliu [ctb]
Repository CRAN
Date/Publication 2018-01-04 09:36:52 UTC

R topics documented:

  ipfCluster ................................................................. 2
  ipfDistance ............................................................... 3
  ipfEstimate ............................................................... 4
  ipfEstimateBeaconPositions ........................................... 4
  ipfGroup ................................................................. 5
  ipfKnn ................................................................. 6
ipfCluster

Creates clusters using the specified method

Description

Creates clusters using the specified method and assigns a cluster id to each cluster.

Usage

ipfCluster(data, method = "k-means", k = NULL, grid = 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. 'grid' for clustering based on grid partition. Requires parameter grid. 'AP' for affinity propagation algorithm.
k parameter k
grid a vector with the grid size for the 'grid' method
... additional parameters for k-means, apcluster and apclusterK for 'k-means' method
for 'apcluster' AP method additional parameters see: https://cran.r-project.org/web/packages/apcluster/index.html

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

```r
clusters <- ipfCluster(head(ipftrain[, 169:170], k = 4))
clusters <- ipfCluster(head(ipftrain[, grep('^wap', names(ipftrain))], 20),
                      method = 'AP')$clusters
```

---

### ipfDistance

#### Distance function

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

#### Usage

```r
ipfDistance(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

```r
dist <- ipfDistance(ipftrain[, 1:168], ipfTest[, 1:168])
dist <- ipfDistance(ipftrain, ipfTest, subset = c('X', 'Y'), method = 'manhattan')
```
**ipfEstimate**

Estimates the location of the test observations

**Description**

Estimates the location of the test observations

**Usage**

```r
ipfEstimate(ipfmodel, test_fgp, test_pos = NULL)
```

**Arguments**

- `ipfmodel` an ipfModel
- `test_fgp` a matrix or a data frame containing the fingerprints of the test set
- `test_pos` a matrix or a data frame containing the position of the test set fingerprints

**Value**

An S3 object of class ipfEstimation, with the following properties: location -> a matrix with the predicted locations errors -> a numeric vector with the errors 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

**Examples**

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

## Not run:

```r
model <- ipfProbabilistic(ipftrain[, 1:168], ipftrain[, 169:170], k = 9, delta = 10)
estimation <- ipfEstimate(model, ipftest[, 1:170], ipftest[, 169:170])
```

## End(Not run)

**ipfEstimateBeaconPositions**

Estimates the positions of the emitter beacons

**Description**

Estimates the positions of the emitter beacons
ipfGroup

Usage

`ipfEstimateBeaconPositions(fingerprints, positions, method = "wcentroid", rssirange = c(-100, 0), norssi = NA)`

Arguments

- `fingerprints`: a data frame or a matrix with the RSSI fingerprints
- `positions`: a data frame or a matrix with the positions of the fingerprints
- `method`: method to use to estimate the position of the access points: 'centroid', 'wcentroid' or 'wip'
- `rssirange`: a numeric vector with the range of the RSSI data
- `norssi`: value used in dataRSSI when a beacon is not detected

Examples

```r
wapp <- ipfEstimateBeaconPositions(ipftrain[, 1:16], ipftrain[, 169:170], method = "wcentroid")
```

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

```r
group <- ipfGroup(mtcars, cyl)
group <- ipfGroup(mtcars, gear, carb)
group <- ipfGroup(ipftrain, X, Y)
```
ipfKnn

*Implements the k-nearest neighbors algorithm*

**Description**

Implements the k-nearest neighbors algorithm

**Usage**

```r
ipfKnn(train_fgp, train_pos, k = 3, method = "euclidean",
       weights = "distance", norm = 2, sd = 5, epsilon = 0.001, alpha = 1,
       threshold = 20, FUN = NULL, ...)
```

**Arguments**

- `train_fgp`: a data frame containing the fingerprint vectors of the training set
- `train_pos`: a data frame containing the positions of the training set observations
- `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'
- `weights`: the algorithm to compute the weights: 'distance' or 'uniform'
- `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. The two first parameters taken by the function must be train and test
- `...`: additional parameters for provided function FUN

**Value**

An S3 object of class ipfModel, with the following properties: `params` -> a list with the parameters passed to the function `data` -> a list with the fingerprints and locations

**Examples**

```r
model <- ipfKnn(ipftrain[, 1:168], ipftrain[, 169:170], 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 = "Empirical cumulative density function")

Arguments

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

Examples

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

ipfPlotEstimation

Plots the estimated locations

Description

Plots the estimated locations

Usage

ipfPlotEstimation(model, estimation, testpos = NULL, observations = c(1),
reverseAxis = FALSE, showneighbors = FALSE, showLabels = FALSE,
xlab = NULL, ylab = NULL, title = "")
Arguments

model     an ipfModel
estimation an ipfEstimation
testpos   position of the 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], ipftrain[, 169:170])
estimation <- ipfEstimate(model, ipftest[, 1:168], ipftest[, 169:170])
ipfPlotEstimation(model, estimation, ipftest[, 169:170],
observations = seq(7, 10), showneighbors = TRUE,
reverseAxis = TRUE)

ipfPlotLocation  
Plots the spatial location of the observations

Description

Plots the spatial location of the observations

Usage

ipfPlotLocation(positions, plabel = FALSE, reverseAxis = FALSE,
xlab = NULL, ylab = NULL, title = "", pgrid = FALSE)

Arguments

positions     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

```r
ipfPlotLocation(ipftrain[, 169:170])

ipfPlotLocation(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**

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

**Arguments**

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

**Examples**

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

---

**ipfProbabilistic**

This function implements a probabilistic algorithm

**Description**

This function implements a probabilistic algorithm

**Usage**

```r
ipfProbabilistic(train_fgp, train_pos, group_cols = NULL, groups = NULL,
    k = 3, FUN = sum, delta = 1, ...)
```
**ipfProximity**

Estimates the position of the observations from its fingerprints and the access point location using a logarithmic path loss model

### Arguments

- `train_fgp`: a data frame containing the fingerprint vectors of the training set
- `train_pos`: a data frame containing the positions of the training set observations
- `group_cols`: a character vector with the names of the columns to be used as the criteria to group the fingerprints. By default, the groups will be created using all the columns available in the `train_pos` data frame.
- `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 S3 object of class `ipfModel`, with the following properties:
- `params`: a list with the parameters passed to the function
- `data`: a list with the fingerprints probabilistic parameters (means and standard deviations) and its locations

### Examples

```r
groups <- ipfGroup(ipftrain[, X], Y)
model <- ipfProbabilistic(ipftrain[, 1:168], ipftrain[, 169:170], groups = groups)

## Not run:
model <- ipfProbabilistic(ipftrain[, 1:168], ipftrain[, 169:170], k = 9, delta = 10)

## End(NOT run)
```

### Description

Estimates the position of the observations from its fingerprints and the access point location using a logarithmic path loss model

### Usage

```r
ipfProximity(bpos, rssirange = c(-100, 0), norssi = NA, alpha = 5, wapPowl = -30)
```
### Arguments

- **bpos**: a matrix or a data frame containing the position of the beacons, in the same order as they appear in fingerprints.
- **rssirange**: range of the RSSI data.
- **norssi**: value used to represent a not detected AP.
- **alpha**: path loss exponent.
- **wapPow1**: detected RSSI at one meter range.

### Value

An S3 object of class ipfEstimation, with the following properties:
- **location**: a matrix with the predicted locations.
- **errors**: a numeric vector with the errors, if loctest has been provided.
- **neighbors**: NULL.
- **weights**: NULL.

### Examples

```r
ipfEst <- ipfProximity(ipftrain[1:10, 1:168], ipfpwap, ipftrain[1:10, 169:170], alpha = 4)
```

---

### Description

Indoor localization data set with the positions of the wireless access points present in the ipftrain and ipftest data sets. Unknown locations are stored as NAs. Data from the positioning tutorial of the seventh international conference on indoor Positioning and Indoor Navigation (IPIN2016).

---

### Usage

- **ipfpwap**

---

### Format

A data frame with columns:

- **X**: X coordinate in meters relative to the origin of a predefined Cartesian coordinate system. From -61 to 50.
- **Y**: Y coordinate in meters relative to the origin of a predefined Cartesian coordinate system. From 8.5 to 117.5.
Source


Examples

```r
## Not run:
ipfpwap

## End(Not run)
```

---

**ipfRMID**  
*Estimates the inherent difficulty of the radio map*

**Description**

Estimates the inherent difficulty of the radio map

**Usage**

```r
ipfRMID(fingerprints, positions, rangeRSSI = c(-100, 0), noRSSI = NA, gridSize = 5)
```

**Arguments**

- `fingerprints`: a matrix or a data frame containing the RSSI data (fingerprints) of the observations
- `positions`: a matrix or a data frame containing the positions of the fingerprints
- `rangeRSSI`: range of the RSSI data
- `noRSSI`: value used to represent a not detected AP
- `gridSize`: size of the grid to consider

**Value**

A numeric value representing the RMID value (Radio Map Inherent Difficulty)

**Examples**

```r
## Not run:
rmid <- ipfRMID(ipftrain[, 1:168], ipftrain[, 169:170], noRSSI = NA)

## End(Not run)
```
**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, wap17, wap18, ..., wap154, wap155, wap156, wap157, wap158, wap159, wap160, wap161, wap162, wap163, wap164, wap165, wap166, wap167, wap168**: Intensity value for WAPs. Negative integer values from -99 to 0. NA is used if WAP was not detected.
- **X**: X coordinate in meters relative to the origin of a predefined Cartesian coordinate system. From -0.60 to 4.39
- **Y**: Y coordinate in meters relative to the origin of a predefined Cartesian 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**

Examples

```r
## Not run:
ipftest

## End(Not run)
```

---

**ipftrain**

*Indoor localization training data set to test Indoor Positioning System that rely on WLAN/WiFi fingerprint. 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/WiFi fingerprint. 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, wap17, wap18, ... wap154, wap155, wap156, wap157, wap158, wap159, wap160, wap161, wap162, wap163, wap164, wap165, wap166, wap167, wap168`
  
  Intensity value for WAPs. Negative integer values from -99 to 0. NA is used if WAP was not detected.

- `X` X coordinate in meters relative to the origin of a predefined Cartesian coordinate system. From -0.60 to 4.39

- `Y` Y coordinate in meters relative to the origin of a predefined Cartesian 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.
ipfTransform

Source

Examples

```r
## Not run:
ipftrain

## End(Not run)
```

<table>
<thead>
<tr>
<th>ipfTransform</th>
<th>Transform function</th>
</tr>
</thead>
</table>

Description
Transforms the RSSI (Received Signal Strength Intensity) data to positive or exponential values.

Usage

```r
ipfTransform(data, outRange = c(0, 1), outNoRSSI = 0, inRange = NULL, inNoRSSI = 0, trans = "scale", base = exp(1), alpha = 24)
```

Arguments
- `data` a vector, matrix or data frame containing the RSSI vectors.
- `outRange` the desired range for the output RSSI data.
- `outNoRSSI` value desired in the RSSI output data to represent a not detected AP.
- `inRange` a vector containing the range of the RSSI value from the initial data.
- `inNoRSSI` value used in the RSSI data to represent a not detected AP.
- `trans` the transformation to perform, 'scale' or 'exponential'.
- `base` base for the 'exponential' transformation.
- `alpha` alpha parameter for the 'exponential' transformation.

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

Examples

```r
trainRSSI <- ipftrain[,1:168]
ipfTransform(trainRSSI, inRange = c(-100, 0), outRange = c(1, 100), inNoRSSI = NA, outNoRSSI = 0)
```
Index

*Topic datasets
  ipfpwap, 11
  ipftest, 13
  ipftrain, 14

ipfCluster, 2
ipfDistance, 3
ipfEstimate, 4
ipfEstimateBeaconPositions, 4
ipfGroup, 5
ipfKnn, 6
ipfPlotEcdf, 7
ipfPlotEstimation, 7
ipfPlotLocation, 8
ipfPlotPdf, 9
ipfProbabilistic, 9
ipfProximity, 10
ipfpwap, 11
ipfRmid, 12
ipftest, 13
ipftrain, 14
ipfTransform, 15