Package 'mmpp'

September 29, 2017

Type Package

Title Various Similarity and Distance Metrics for Marked Point Processes

Version 0.6

Date 2017-09-29

Author Hideitsu Hino, Ken Takano, Yuki Yoshikawa, and Noboru Murata

Maintainer Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>

Description Compute similarities and distances between marked point processes.

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-29 04:07:37 UTC

R topics documented:

mmpp-package *A package for Computing Similarity and Distance Metrics for Marked Point Process Data*

Description

This packages is used to calculate various similarity and distance measures for sample sequences of both simple and marked temporal point processes.

Details

A simple temporal point process (SPP) is an important class of time series, where the sample realization of the process is solely composed of the times at which events occur. Particular examples of point process data are neuronal spike patterns or spike trains, and a large number of distance and similarity metrics for those data have been proposed. A marked point process (MPP) is an extension of a simple temporal point process, in which a certain vector valued mark is associated with each of the temporal points in the SPP. Analysis of MPPs are of practical importance because instances of MPPs include recordings of natural disasters such as earthquakes and tornadoes. This package implements a number of distance and similarity metrics for SPP, and also extends those metrics for dealing with MPP. It provides a systematic and unified platform for calculating the similarities and distances between SPP, and support marked point process to offer a platform for performing metric-based analysis of earthquakes, tornados, epidemics, or stock exchange data.

The package has functions `coocmetric`, `fmetric`, `ieimetric`, and `iipmetric` for calculating similarity or distance between two sample sequences. A sample dataset `Miyagi20030626` is included in the package. It offers utility two functions: `splitMPP`, which splits a sample sequence into a list of partial sequences by using a sliding window, and `k2d`, which transforms a similarity matrix to a distance matrix and vice versa.

Author(s)

Author: Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>, Ken Takano, Yuki Yoshikawa, and Noboru Murata

References

R. Quian Quiroga, T. Kreuz, and P. Grassberger. Event synchronization: a simple and fast method to measure synchronicity and time delay patterns, Physical Review E, Vol. 66(4), 041904, 2002.

J. D. Hunter and G. Milton. Amplitude and frequency dependence of spike timing: implications for dynamic regulation, Journal of Neurophysiology, Vol. 90, pp. 387-94, 2003.

M. C. W. van Rossum. A Novel Spike Distance. Neural Computation, Vol. 13(4), pp. 751-763, 2001.

S. Schreiber, J.M. Fellous, P.H. Tiesinga, and T.J. Sejnowski. A new correlation-based measure of spike timing reliability, Neurocomputing, Vols. 52-54, pp. 925-931, 2003.

T. Kreuz, J.S. Haas, A. Morelli, H.D.I. Abarbanel, and A. Politi. Measuring spike train synchrony, Journal of Neuroscience Methods, Vol. 165(1), pp. 151-161, 2007.

A.R.C. Paiva, I. Park, and J.C. Principe. A reproducing kernel Hilbert space framework for spike train signal processing, Neural Computation, Vol. 21(2), pp. 424-449, 2009.

Examples

```
## An example to show that the prediction error of the magnitude based on
## 1-nearest neighbor predictor can be reduced by taking marks into account.
## It will take about 5 minutes if you run this example
## Not run:
library(mmpp)
data(Miyagi20030626)
## split the original MPP by using 3-hour time window
sMiyagi <- splitMPP(Miyagi20030626,h=60*60*3,scaleMarks=TRUE)$S

## target of the prediction is the maximum magnitude in the window
y <- NULL
for(i in 1:length(sMiyagi)){
  y <- c(y, max(sMiyagi[[i]]$magnitude))
}

y <- y[-1]
sMiyagi[[length(sMiyagi)]] <- NULL

## number of whole partial MPPs splitted by a 3-hour time window
N <- length(sMiyagi)
## training samples are past one week data
Ntr <- 24*7/3
## number of different prediction methods
Nd <- 10

err <- matrix(0, N-Ntr, Nd)
colnames(err) <- c("f SPP","iip SPP","cooc smooth SPP","cooc count SPP","iei SPP",
                   "f MPP","iip MPP","cooc smooth MPP","cooc count MPP","iei MPP")

## predict the max magnitude in the next 3-hour based on the similarity
## between the current partial point process and the 7-days past partial point process
cat("running prediction experiment")
for(t in 1:(N-Ntr)){
  cat(".")
  qid <- Ntr+t
  q <- sMiyagi[[qid]]

  ## simple PP
  ## fmetric with tau=1
  sim2query <- NULL
  for(i in 1:Ntr){
    sim2query <- c(sim2query,fmetric(q$time,sMiyagi[[qid-i]]$time))
  }
  err[t,1] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

  ## iipmetric with tau=1
  sim2query <- NULL
  for(i in 1:Ntr){
```

```
  sim2query <- c(sim2query,iipmetric(q$time,sMiyagi[[qid-i]]$time))
}
err[t,2] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## coocmetric (smooth) with tau=1
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,coocmetric(q$time,sMiyagi[[qid-i]]$time,type="smooth"))
}
err[t,3] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## coocmetric (count)
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,coocmetric(q$time,sMiyagi[[qid-i]]$time,type="count"))
}
err[t,4] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## iei metric
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,ieimetric(q$time,sMiyagi[[qid-i]]$time))
}
err[t,5] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## marked PP with latitude, longitude, depth, and magnitude
## fmetric with tau=1
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,fmetric(q,sMiyagi[[qid-i]]))
}
err[t,6] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## iipmetric with tau=1
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,iipmetric(q,sMiyagi[[qid-i]]))
}
err[t,7] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## coocmetric (smooth) with tau=1
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,coocmetric(q,sMiyagi[[qid-i]],type="smooth"))
}
err[t,8] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

## coocmetric (count)
sim2query <- NULL
for(i in 1:Ntr){
  sim2query <- c(sim2query,coocmetric(q,sMiyagi[[qid-i]],type="count"))
}
err[t,9] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])
```

```
  ## iei metric
  sim2query <- NULL
  for(i in 1:Ntr){
    sim2query <- c(sim2query,ieimetric(q,sMiyagi[[qid-i]]))
  }
  err[t,10] <- abs(y[qid]-y[t:(Ntr+t-1)][which.max(sim2query)])

}
cat("done\n")
print(colMeans(err))
##f SPP      iip SPP     cooc smooth SPP  cooc count SPP  iei SPP
##0.7002634  0.6839529   0.7263602        0.6632930       0.7905148
##f MPP      iip MPP     cooc smooth MPP  cooc count MPP  iei MPP
##0.6839529  0.6317594   0.6643804        0.6622056       0.7698548

## End(Not run)
```

---

coocmetric *Metrics for Point Process Realizations Based on Co-occurrence*

Description

For comparing two SPP realizations, it is natural to count the number of events which can be considered to be co-occurring. There are two metrics for SPP realizations based on the notion of co-occurrence. The first one proposed by Quian Quiroga et al. (2002) directly counts near-by events. The second counting metric co-occurrence is proposed by Hunter and Milton (2003), which is based on a smoothing function.

Usage

```
coocmetric(S1, S2, measure = "sim", type = "count", tau = 1, M = NULL)
```

Arguments

S1	marked point process data.
S2	marked point process data.
measure	"sim" for similarity and "dist" for distance. Default "sim".
type	if "count", counting near-by event measure by Quian is computed. If "smooth", smoothed counting co-occurrence measure by Hunter and Milton is computed. Default "count".
tau	a parameter for filtering function.
M	a precision matrix for filter of marks, i.e., exp(- r' M r) is used for filtering marks. It should be symmetric and positive semi-definite.

Details

coocmetric computes co-occurrence base metrics for two point process realizations. This function counts the number of events in S1 which is coincided with those in S2, and vice versa.

Value

Similarity or distance between two inputs (marked) point process S1 and S2.

Author(s)

Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>, Ken Takano, Yuki Yoshikawa, and Noboru Murata

References

R. Quian Quiroga, T. Kreuz, and P. Grassberger. Event synchronization: a simple and fast method to measure synchronicity and time delay patterns, Physical Review E, Vol. 66(4), 041904, 2002.

J. D. Hunter and G. Milton. Amplitude and frequency dependence of spike timing: implications for dynamic regulation, Journal of Neurophysiology, Vol. 90, pp. 387-94, 2003.

Examples

```
## The aftershock data of 26th July 2003 earthquake of M6.2 at the northern Miyagi-Ken Japan.
data(Miyagi20030626)
## time longitude latitude depth magnitude
## split events by 7-hour
sMiyagi <- splitMPP(Miyagi20030626,h=60*60*7,scaleMarks=TRUE)$S
N <- 10
sMat <- matrix(0,N,N)
tau<-0.2
  cat("calculating coocmetric(smooth)...")
 for(i in 1:(N)){
   cat(i," ")
   for(j in i:N){
     S1 <- sMiyagi[[i]]$time;S2 <- sMiyagi[[j]]$time
    sMat[i,j] <- coocmetric(S1,S2,type="smooth",tau=tau,M=diag(1,4))
   }
 }
 sMat <- sMat+t(sMat)
 tmpd <- diag(sMat) <- diag(sMat)/2
 sMat <- sMat/sqrt(outer(tmpd,tmpd))
image(sMat)
```

`fmetric` *Compute Filter-based Metrics in a Functional Space Between Marked Point Processes*

Description

The most commonly used and intensively studied metrics for spike trains, which is based on the continuation of event sequence to a real valued continuous function using a smoother function.

Usage

```
fmetric(S1, S2, measure = "sim", h = "laplacian", tau = 1, M = NULL,
  abs.tol = .Machine$double.eps^0.25)
```

Arguments

`S1`	marked point process data.
`S2`	marked point process data.
`measure`	`"sim"` for similarity and `"dist"` for distance. Default `"sim"`.
`h`	filtering function. Default `"laplacian"` offers significant computational advantage. A function can be specified here like `h=function(x,tau) exp(-x^2/tau)`. The function should be square integrable and non-negative (not checked in the code).
`tau`	parameter for filtering function.
`M`	a precision matrix for filter of marks, i.e., exp(- r' M r) is used for filtering marks. It should be symmetric and positive semi-definite.
`abs.tol`	absolute tolerance for numerical integration.

Details

`fmetric` computes filter-based measure between MPP realizations. Discrete event timings are transformed into a continuous function by using a kernel smoother, and usual l2 inner product is adopted for defining the similarity between two point process realizations.

Value

Similarity or distance between two inputs (marked) point process S1 and S2.

Author(s)

Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>, Ken Takano, Yuki Yoshikawa, and Noboru Murata

References

M. C. W. van Rossum. A Novel Spike Distance. Neural Computation, Vol. 13(4), pp. 751-763, 2001.

S. Schreiber, J.M. Fellous, P.H. Tiesinga, and T.J. Sejnowski. A new correlation-based measure of spike timing reliability, Neurocomputing, Vols. 52-54, pp. 925-931, 2003.

Examples

```
##The aftershock data of 26th July 2003 earthquake of M6.2 at the northern Miyagi-Ken Japan.
data(Miyagi20030626)
## time longitude latitude depth magnitude
## split events by 7-hour
sMiyagi <- splitMPP(Miyagi20030626,h=60*60*7,scaleMarks=TRUE)$S
N <- 10
tau <- 0.1
sMat <- matrix(0,N,N)
  cat("calculating fmetric with tau ",tau,"...")
 for(i in 1:(N)){
   cat(i," ")
   for(j in i:N){
     S1 <- sMiyagi[[i]]$time;S2 <- sMiyagi[[j]]$time
    sMat[i,j] <- fmetric(S1,S2,tau=tau,M=diag(1,4))
   }
 }
 sMat <- sMat+t(sMat)
 tmpd <- diag(sMat) <- diag(sMat)/2
 sMat <- sMat/sqrt(outer(tmpd,tmpd))
image(sMat)
```

`ieimetric` *Compute Inter Event Interval-based Metric Between Marked Point Processes*

Description

This metric considers inter event interval for point processes.

Usage

```
ieimetric(S1, S2, measure = "sim", M = NULL, window.length = NULL,
  variant = "spike", abs.tol = .Machine$double.eps^0.25)
```

Arguments

`S1`	marked point process data.
`S2`	marked point process data.
`measure`	`"sim"` for similarity and `"dist"` for distance. Default `"sim"`.

M	a precision matrix for filter of marks, i.e., exp(- r' M r) is used for filtering marks. It should be symmetric and positive semi-definite.
window.length	width of the window used for splitting the original MPP. If not provided, max(max(S1$time,S2$time) - min(S1$time,S2$time)) is used.
variant	choose from two variants "spike-weighted" or "time-weighted". Default "spike", which is computationally efficient than "time". See the reference for details.
abs.tol	absolute tolerance for numerical integration.

Details

iei computes inter event interval-based measure between MPP realizations. iei for simple point process does not have any tuning parameter, which can be a desirable property for data analysis. However, it's computational cost is relatively higher than other metrics.

Value

Similarity or distance between two inputs (marked) point process S1 and S2.

Author(s)

Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>, Ken Takano, Yuki Yoshikawa, and Noboru Murata

References

T. Kreuz, J.S. Haas, A. Morelli, H.D.I. Abarbanel, and A. Politi. Measuring spike train synchrony, Journal of Neuroscience Methods, Vol. 165(1), pp. 151-161, 2007.

Examples

```
##The aftershock data of 26th July 2003 earthquake of M6.2 at the northern Miyagi-Ken Japan.
data(Miyagi20030626)
## time longitude latitude depth magnitude
## split events by 7-hour
sMiyagi <- splitMPP(Miyagi20030626,h=60*60*7,scaleMarks=TRUE)$S
N <- 5
sMat <- matrix(0,N,N)
  cat("calculating intensity inner product...")
 for(i in 1:(N)){
   cat(i," ")
   for(j in i:N){
     S1 <- sMiyagi[[i]]$time;S2 <- sMiyagi[[j]]$time
    sMat[i,j] <- ieimetric(S1,S2,M=diag(1,4))
   }
 }
 sMat <- sMat+t(sMat)
 tmpd <- diag(sMat) <- diag(sMat)/2
 sMat <- sMat/sqrt(outer(tmpd,tmpd))
image(sMat)
```

`iipmetric` *Compute Intensity Inner Product Metrics*

Description

For the analysis of point process, intensity function plays a central roll. Paiva et al. (2009) proposed to use the intensity function for defining the inner product between point process realizations.

Usage

```
iipmetric(S1, S2, measure = "sim", tau = 1, M = NULL)
```

Arguments

`S1`	marked point process data.
`S2`	marked point process data.
`measure`	`"sim"` for similarity and `"dist"` for distance. Default `"sim"`.
`tau`	a parameter for filtering function.
`M`	a precision matrix for filter of marks, i.e., exp(- r' M r) is used for filtering marks. It should be symmetric and positive semi-definite.

Details

`iipmetric` computes intensity inner product metric. Intensity function for the point process realization is estimated by kernel density estimator. This function adopts Gaussian kernels for the sake of computational efficiency.

Value

Similarity or distance between two inputs (marked) point process S1 and S2.

Author(s)

Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>, Ken Takano, Yuki Yoshikawa, and Noboru Murata

References

A.R.C. Paiva, I. Park, and J.C. Principe. A reproducing kernel Hilbert space framework for spike train signal processing, Neural Computation, Vol. 21(2), pp. 424-449, 2009.

Examples

```
##The aftershock data of 26th July 2003 earthquake of M6.2 at the northern Miyagi-Ken Japan.
data(Miyagi20030626)
## time longitude latitude depth magnitude
## split events by 7-hour
sMiyagi <- splitMPP(Miyagi20030626,h=60*60*7,scaleMarks=TRUE)$S
N <- 10
tau <- 0.1
sMat <- matrix(0,N,N)
  cat("calculating intensity inner product...")
 for(i in 1:(N)){
   cat(i," ")
   for(j in i:N){
     S1 <- sMiyagi[[i]]$time;S2 <- sMiyagi[[j]]$time
    sMat[i,j] <- iipmetric(S1,S2,tau=tau,M=diag(1,4))
   }
 }
 sMat <- sMat+t(sMat)
 tmpd <- diag(sMat) <- diag(sMat)/2
 sMat <- sMat/sqrt(outer(tmpd,tmpd))
image(sMat)
```

---

k2d *Convert Kernel Matrix to Distance Matrix*

Description

k2d provides various methods for converting kernel matrix to distance matrix and vice versa.

Usage

```
k2d(Mat, direction = "k2d", method = "norm", scale = 1, pos = TRUE)
```

Arguments

Mat	matrix, either kernel matrix or distance matrix to be converted.
direction	a character string "k2d" or "d2k". The latter interpret the Mat as a distance matrix and convert it to a kernel matrix. Default "k2d".
method	a character string to specify how the matrix is converted. "Default "norm".
scale	a numeric parameter used to scale the matrix. Typically used in exp(- d(x,y)/scale).
pos	logical. If TRUE when direction="d2k", negative eigenvalues are round to zero to obtain positive semidefinite kernel matrix. Default TRUE.

Details

There are various ways to convert kernel function values to distance between two points. Normal-distance (when `method="norm"`) means a conversion d_ND(x,y) = sqrt k(x,x) - 2k(x,y) + k(y,y).

Cauchy-Schwarz-type conversion (`method="CS"`) is more principled way: d_CS(x,y) = arccos k^2(x,y)/k(x,x)k(y,y).

Other two simple ways are d_exp(x,y) = exp(- k(x,y)/scale), which is an exponential-type distance (`method="exp"`), and d_n(x,y) = 1 - k(x,y)/sqrt k(x,x)k(y,y), which we call naive (`method="naive"`).

For converting distance to kernel (`direction="d2k"`), it should be noted that we usually have distance between pairs of points only, and distances from "origin" are unknown. Double-centering (`method="DC"`) is the most popular and simple way to convert distance to kernel. However, it does not make positive definite kernel in general, and it sometimes require post-processing, e.g., cutting off negative eigenvalues (`pos=TRUE`). Another simple way is exponential map (`method="exp"`), i.e., k(x,y) = exp(- d(x,y)/scale).

Author(s)

Hideitsu Hino <hinohide@cs.tsukuba.ac.jp>, Ken Takano, Yuki Yoshikawa, and Noboru Murata

---

`Miyagi20030626` *The aftershock data of 26th July 2003 earthquake of M6.2 at the northern Miyagi-Ken Japan*

Description

A reparameterization of the `main2006JUL26` data frame from the `SAPP` package.

Usage

```
data(Miyagi20030626)
```

Format

A data object with 2305 seismic events.

Details

The variables are as follows:

- time. time in seconds from the main shock.
- longitude. longitude of the seismic center.
- latitude. latitude of the seismic center.
- depth. depth of the seismic center.
- magnitude. magnitude of the earthquake.

Source

SAPP R package available at https://cran.r-project.org

---

splitMPP *Split MPP Data by Sliding Time Window*

Description

This function splits a point process realization into a list of splitted point process realizations with length h.

Usage

```
splitMPP(mppdata, h = 60 * 60 * 48, ol = NULL, TimeOrigin = TRUE,
  scaleMarks = FALSE, scaleWindow = TRUE, MarkCenter = NULL,
  MarkCenterID = NULL)
```

Arguments

mppdata	marked point process in data.frame composed of "time, mark1, mark2, ...".
h	width of the time window. Default is set to h=60*60*48, which is two days when $time is recorded in second. This is suitable for a special seismic data only.
ol	length of overlap for the sliding window. Default 0.
TimeOrigin	logical. If TRUE, the beginning of the window is assumed to be the origin of time. Default TRUE.
scaleMarks	logical. If TRUE, marks (except time) are normalized to have unit variance in whole time series (not in individual windows). Default FALSE.
scaleWindow	logical. If TRUE, time interval (window.length) is normalized to one.
MarkCenter	vector for specifying the center of the mark. Use when there are relative center point such as the main shock of the earthquake. Default NULL.
MarkCenterID	vector for specifying the elements of center of the mark.

Details

splitMPP splits a point process realization into a list of splitted point process realizations with length h.

Examples

```
##The aftershock data of 26th July 2003 earthquake of M6.2 at the northern Miyagi-Ken Japan.
data(Miyagi20030626)
## time longitude latitude depth magnitude
## split events by 5-hours
sMiyagi <- splitMPP(Miyagi20030626,h=60*60*5,scaleMarks=TRUE)
```

Index