Package: eventInterval (via r-universe)

October 29, 2024

Version 1.3

Title Sequential Event Interval Analysis

Date 2015-10-20

Author Jim Lemon <`drjimlemon@gmail.com`>,

Maintainer Jim Lemon <`drjimlemon@gmail.com`>

Imports MASS, graphics, stats

Description Functions for analysis of rate changes in sequential events.

License GPL (>= 2)

NeedsCompilation no

Date/Publication 2015-10-20 07:37:06

Repository https://jimlemon.r-universe.dev

RemoteUrl https://github.com/cran/eventInterval

RemoteRef HEAD

RemoteSha b1d63df5571ec035cc913712a57093a42b7c093e

Contents

eventInterval-package *Sequential event interval analysis*

Description

Functions for analysis of rate changes in sequential events. Appropriate data are the times of observation of well defined events, such as equipment failures or deaths.

Details

Package:	eventInterval
Version:	1.0
Date:	2013-05-28
Depends:	MASS
License:	GPL (>=2)
Packaged:	2013-05-28 11:12:16 UTC; root
Built:	R 3.0.0; ; 2013-05-28 11:12:19 UTC; unix

Index:

```
EIglm  Perform a GLM with intervals as response variable
EIdensity Plot the empirical and theoretical densities of intervals
EIplot  Plot the intervals against the occurrence of the events
rescale  scale numeric values into a new range
```

The basic model is of events that are independent and uniformly distributed in time. This assumption underlies the use of the exponential distribution in the generalized linear models produced by 'EIglm'. In general, it is expected that deviations from uniformly distributed intervals follow a function that is expressed in one or more of the predictors. A predictor that is included by default is the times of the occurrence of the events. This will reveal linear changes in the rate. Other predictors may test for non-linear or cyclic changes in rate.

Event interval analysis can be a useful alternative to Poisson based analysis when few events occur in the counting intervals. Its advantage in power to detect changes in rate diminishes as the number of events increases.

This type of analysis was known as event history analysis, but the phrase is now usually applied to the analysis of the antecedents of events, whether sequential or not.

Author(s)

Jim Lemon <jim@bitwrit.com.au>, Maintainer: Jim Lemon <jim@bitwrit.com.au>

References

Maguire, B.A., Pearson, E.S. & Wynn, A.H.A. (1952) The time intervals between industrial accidents. Biometrika, 39(1/2): 168-180.

Whitworth, W.A. (1901) Choice and chance. (3rd ed.) Cambridge: Deighton Bell.

Lemon, J. (2014) The analysis of event rates using intervals. The Quantitative Methods for Psychology, 10 (1), 68-76.

`EIdist` *Plot the empirical and exponential distributions of values*

Description

Plot the empirical distribution of a set of values.

Usage

```
EIdist(event_times,nbreaks=10,main="",xlab="",ylab="",
xaxticks=NA,xaxlabs=NA)
```

Arguments

`event_times`	A numeric vector of times of occurrence of events.
`nbreaks`	The number of breaks desired (but not always delivered).
`main`	Title for the plot.
`xlab`	X axis label for the plot.
`ylab`	Y axis label for the plot.
`xaxticks`	Optional X tick mark positions.
`xaxlabs`	Optional X tick labels.

Details

'`EIdist`' calculates the intervals between the times passed in '`event_times`' and plots a histogram of their distribution. It adds a smoothed line based on the '`mids`' and '`counts`'. It then calls '`fitdistr`' from **MASS** to get an estimated rate for a model exponential distribution. A second line is added to the plot for comparison with the smoothed empirical distribution.

Value

nil

Author(s)

Jim Lemon

Examples

```
 ## Not run:
  # won't pass check due to the call to fitdistr
  data(florida_hurr20)
  oldloc<-Sys.setlocale("LC_TIME", "C")
  fh_dates<-
   as.Date(paste(florida_hurr20$day,florida_hurr20$month,
   florida_hurr20$year,sep="-"),"%d-%B-%Y")
  fh_days<-as.numeric(fh_dates)
  fh_ints<-diff(fh_days)
  fh_counts<-tabulate(florida_hurr20$year[-1]-1899,nbins=100)
  require(MASS)
  EIdist(fh_days,main="Empirical distribution of hurricanes",
   xlab="Intervals in days",ylab="Frequencies")
  Sys.setlocale("LC_TIME", oldloc)

## End(Not run)
```

`EIglm` *Perform a GLM on times of events*

Description

Perform a GLM with the intervals between events as the response and the times of the events as a predictor.

Usage

```
 EIglm(event_times,event_vars=list(),formula=NULL)
```

Arguments

`event_times`	A numeric vector of times of occurrence of events.
`event_vars`	Optional predictors to include in the GLM.
`formula`	A optional string that can be coerced to a valid formula.

Details

'`EIglm`' calculates the intervals between the times passed in '`event_times`' and calls '`glm`' with '`family="Gamma"`'. If the times are not in increasing order, they will be sorted. If there are any zero length intervals, the events that occur in the same time increment are spread out over that time increment.

If more predictors are passed in '`event_vars`' these will be added to the formula passed to '`glm`'. When passing predictors based on the events, remember to discard the first value as the intervals are calculated between events. This means that the interval is unknown for the first event.

'`EIglm`' only tests the effects of the variables specified and does not test interactions. If a more complex model is to be tested, the user must directly call '`glm`' and compose a formula for the model (see the last example). Note that because there is no '`data`' argument in EIglm, each term in the model must be specified explicitly.

Value

An object of class 'glm'.

Author(s)

Jim Lemon

References

Blake, E.S., Rappaport, E.N. & Landsea, C.W. (2007) The deadliest, costliest, and most intense United States tropical cyclones from 1851 to 2006 (and other frequently requested hurricane facts). NOAA Technical Memorandum NWS TPC-5, Miami: National Weather Service.

Florida Climate Center (2013) Florida statewide averaged temperature data (in degrees Fahrenheit). URL: http://climatecenter.fsu.edu/products-services/data/statewide-averages/temperature accessed 28/5/2013

Examples

```
 ## Not run: 
  # Florida hurricane analysis (20th century)
  data(florida_hurr20)
  oldloc<-Sys.setlocale("LC_TIME", "C")
  fh_dates<-
   as.Date(paste(florida_hurr20$day,florida_hurr20$month,
   florida_hurr20$year,sep="-"),"%d-%B-%Y")
  fh_days<-as.numeric(fh_dates)
  fh_ints<-diff(fh_days)
  fh_counts<-tabulate(florida_hurr20$year[-1]-1899,nbins=100)
  # run a Poisson analysis on the counts
  print(summary(glm(fh_counts~I(1900:1999),family="poisson")))
  EIglm(fh_days)
  data(florida_temp20)
  plot(florida_temp20$year,florida_temp20$Annual,type="b",
    main="Average temperature in Florida",xlab="Year",ylab="degrees F")
  lines(supsmu(florida_temp20$year,florida_temp20$Annual),lwd=2)
  # define a function to match values
  findval<-
   function(x,set) return(which(set %in% x))
  ftemp_years<-florida_temp20$Annual[unlist(sapply(florida_hurr20$year,
   FUN=findval,florida_temp20$year))]
  EIglm(fh_days,list(ft=ftemp_years[-1]))
  Sys.setlocale("LC_TIME", oldloc)

## End(Not run)
 # manually performing an event interval GLM
 # get the Florida hurricane data
 data(florida_hurr20)
 # create date values for the events
 florida_hurr20$date<-
  as.Date(paste(florida_hurr20$day,florida_hurr20$month,
  florida_hurr20$year,sep="-"),"%d-%B-%Y")
```

```
# convert the dates to numeric values (offsets from 1/1/1970)
florida_hurr20$days<-as.numeric(florida_hurr20$date)
# calculate the intervals, adding NA for the first unknown value
florida_hurr20$ints<-c(NA,diff(florida_hurr20$days))
# first test the default model in EIglm, dropping the first event
summary(glm(ints~days,florida_hurr20[-1,],family="Gamma"))
# test a model predicting intervals with times of occurrence,
# the intensities of the hurricanes and their interaction,
# again dropping the first event
summary(glm(ints~days*category,florida_hurr20[-1,],family="Gamma"))
```

`EIplot` *Plot the intervals between times against the times*

Description

Plot the intervals between times against the times.

Usage

```
EIplot(event_times,main="",xlab="",ylab="",xaxticks=NA,xaxlabs=NA)
```

Arguments

`event_times`	A numeric vector of times of occurrence of events.
`main`	Title for the plot.
`xlab`	X axis label for the plot.
`ylab`	Y axis label for the plot.
`xaxticks`	Optional X tick mark positions.
`xaxlabs`	Optional X tick labels.

Details

'`EIplot`' calculates the intervals between the times passed in '`event_times`' and plots a scattergram of intervals against times. It adds a smoothed line and a "rug" of the event times to allow visual inspection for trend.

Value

nil

Author(s)

Jim Lemon

Examples

```
# Florida hurricane analysis (20th century)
data(florida_hurr20)
oldloc<-Sys.setlocale("LC_TIME", "C")
fh_dates<-
 as.Date(paste(florida_hurr20$day,florida_hurr20$month,
 florida_hurr20$year,sep="-"),"%d-%B-%Y")
fh_days<-as.numeric(fh_dates)
fh_ints<-diff(fh_days)
fh_counts<-tabulate(florida_hurr20$year[-1]-1899,nbins=100)
EIplot(fh_days,main="Florida hurricanes, 1900-1999",
 xlab="Year",ylab="Interval between hurricanes (days)",
 xaxticks=as.Date(as.character(seq(1900,2000,by=20)),"%Y"),
 xaxlabs=seq(1900,2000,by=20))
Sys.setlocale("LC_TIME", oldloc)
```

`florida_hurr20` *Hurricanes affecting Florida, USA from 1900-1999*

Description

Atlantic hurricanes recorded as affecting Florida, USA during the 20th century.

Usage

```
data(florida_hurr20)
```

Source

Blake, E.S., Rappaport, E.N. & Landsea, C.W. (2007) The deadliest, costliest, and most intense United States tropical cyclones from 1851 to 2006 (and other frequently requested hurricane facts). NOAA Technical Memorandum NWS TPC-5, Miami: National Weather Service.

`florida_temp20` *Average temperatures in Florida, USA from 1900-1999*

Description

Average statewide temperatures in Florida, USA from 1900-1999.

Usage

```
data(florida_temp20)
```

Source

Florida Climate Center (2013) Florida statewide averaged temperature data (in degrees Fahrenheit). URL: http://climatecenter.fsu.edu/products-services/data/statewide-averages/temperature accessed 28/5/2013 `get_intervals` *Calculate the intervals between events*

Description

Calculate the intervals between a series of times, spreading out equal times in one unit of time.

Usage

```
get_intervals(event_times)
```

Arguments

`event_times`	A numeric vector of times of occurrence of events.

Details

'`get_intervals`' calculates the intervals between the times passed in '`event_times`', adding increments to spread out times that are exactly the same. This can be a problem when the times are integers, as in days. The spreading out of events avoids zero intervals. The rationale for this is that even if more than event occurred on the same day, it is extremely unlikely that the events occurred at exactly the same time of day. The best estimate of the times of day is a uniform distribution over one day.

Value

a vector of intervals between the vector of times passed

Author(s)

Jim Lemon

Examples

```
event_times<-c(1,3,5,5,7,9,10,10,10,12)
get_intervals(event_times)
```

`rescale` *Scale numbers into a new range*

Description

Scale a vector or matrix of numbers into a new range.

Usage

```
rescale(x,newrange)
```

Arguments

`x`	A numeric vector, matrix or data frame.
`newrange`	The minimum and maximum value of the range into which '`x`' will be scaled.

Details

'`rescale`' performs a simple linear conversion of '`x`' into the range specified by '`newrange`'. Only numeric vectors, matrices or data frames with some variation will be accepted. NAs are now preserved - formerly the function would fail.

Value

On success, the rescaled object, otherwise the original object.

Author(s)

Jim Lemon

Examples

```
# scale one vector into the range of another
normal.counts<-rnorm(100)
normal.tab<-tabulate(cut(normal.counts,breaks=seq(-3,3,by=1)))
normal.density<-rescale(dnorm(seq(-3,3,length=100)),range(normal.tab))
# now plot them
plot(c(-2.5,-1.5,-0.5,0.5,1.5,2.5),normal.tab,xlab="X values",
 type="h",col="green")
lines(seq(-3,3,length=100),normal.density,col="blue")
```

Index