Package 'hillmakeR'

August 29, 2016

Type Package

Title Perform occupancy analysis

Version 0.2

Depends R (>= 2.10)

URL https://github.com/gilinson/hillmakeR

Date 2014-07-14

Author David Gilinson <dgilinson@reefpointgroup.com>

Maintainer David Gilinson <dgilinson@reefpointgroup.com>

Description Generate occupancy patterns based on arrival and departure timestamps

License MIT + file LICENSE

LazyLoad true

LazyData true

Suggests plyr

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-07-16 01:13:38

R topics documented:

hillmakeR-package *hillmakeR*

Description

Conduct arrival, departure and occupancy analysis

Details

Package:	hillmakeR
Type:	Package
Version:	0.2
Date:	2014-07-014
License:	MIT

The main function in this package is occupancy. See its help for more information.

Author(s)

David Gilinson dgilinson@reefpointgroup.com

References

Concept based on software written by MW Isken. See simular work at http://www.hselab.org/tags/hillmaker.

hflights *Houston flights data*

Description

Dataset based on data available from 'hflights' package. Data provides arrival and departure information from IAH airport.

Usage

```
hflights
```

See Also

For more information see the hflights package.

occupancy *occupancy*

Description

Calculates the number of items between time stamps using an efficient algorithm.

Usage

```
occupancy(startTimes, stopTimes, resolution = "min",
  initial = NULL, fillup = NULL, countlast = TRUE)
```

Arguments

startTimes	a vector of POSIXct objects representing the time that an item arrived
stopTimes	a vector of POSIXct objects representing the time that an item departed
resolution	size of buckets to calculate occupancy within. May be "sec" "min" "hour" or "day"
fillup	a percentile of the residence times of items to exclude from the return. Only either fillup or initial can be defined.
initial	the initial count of items in the system at time 0. Only either fillup or initial can be defined.
countlast	if true, then the last then items are treated as if they depart at the end of the depart time stamp. See more information in Details

Details

fillup should be used when the initial conditions of the system are unknown. For example a value of 0.95 indicates a 95% chance that any items that were in the system at time 0 have departed and will not be included in the occupancy count.

countlast determines how the stopTimes timestamps are treated. For example if an item has a startTime of 07/07/2014 and the endTime of 07/07/2014 with resolution = "day", then the item will be counted as in the system on 07/07/2014 with countlast = TRUE, but will not be counted as in the system with countlast = FALSE

Value

A data.frame with two coloumns

times	A datetime, class POSIXct
counts	The number of items in the system at current time

References

Concept based on software written by MW Isken. See similar work at http://www.hselab.org/tags/hillmaker.

Examples

```
library(plyr) # need ddply
attach(hflights) # use hflights dataset

# determine how many planes are at airport each minute
planeCount <- occupancy(startTimes=Arrivals,
                        stopTimes=Departures, resolution="min", fillup = 0.95)

# determine how many planes are at airport by hour of day using ddply for summary stats
planeCount$hour <- as.POSIXlt(planeCount$times)$hour
byHourOfDay <- ddply(planeCount, c("hour"),
  function(x) c(mean = mean(x$counts),
  median = median(x$counts),
  q90 = quantile(x$counts, 0.9, names = FALSE)))

# display output graphically
plot(byHourOfDay$mean, type = "o")

# Repeat by Carrier, wrapping ddply around call to hillmakeR functions
# This is the pattern to use whenever there are different types of items to count
planeCountbyCarrier <- ddply(hflights, "Carrier",
  function(x) occupancy(startTimes=x$Arrivals,
  stopTimes=x$Departures, resolution="min", fillup = 0.95))

# determine how many planes are at airport by hour of day
planeCountbyCarrier$hour <- as.POSIXlt(planeCountbyCarrier$times)$hour
byHourAndCarrier <- ddply(planeCountbyCarrier, c("Carrier", "hour"),
  function(x) c(mean = mean(x$counts),
    median = median(x$counts),
    q90 = quantile(x$counts, 0.9)))

plot(subset(byHourAndCarrier, Carrier == "AA")$mean, ylim = c(0, 12.5), type = "o")
for (i in levels(Carrier)){
 lines(subset(byHourAndCarrier, Carrier == i)$mean, col = sample(colours(), 1), type = "o" )
}

# example of using initial values instead of fillup option with several types of items

# make a dummy lookup table
initial.lookup <- data.frame(key = levels(Carrier),
                             value = c(380, 164, 131, 74, 114,
                             161, 334, 106, 127, 498, 485, 158,
                             68, 100, 60))


planeCountbyCarrier <- ddply(hflights, "Carrier",
                    function(x) occupancy(startTimes=x$Arrivals,
                    stopTimes=x$Departures,
                    resolution="min",
                  initial = initial.lookup[initial.lookup$key == x$Carrier[1], "value"]))
```

Index