Package: hydroEvents (via r-universe)

October 11, 2024

Title Extract Event Statistics in Hydrologic Time Series

Version 0.11

Author Conrad Wasko [aut, cre], Danlu Guo [aut]

Maintainer Conrad Wasko <conrad.wasko@gmail.com>

Description Events from individual hydrologic time series are extracted, and events from multiple time series can be matched to each other. Tang, W. & Carey, S. K. (2017) <doi:10.1002/hyp.11185>. Kaur, S., Horne, A., Stewardson, M.J., Nathan, R., Costa, A.M., Szemis, J.M., & Webb, J.A. (2017) <doi:10.1080/24705357.2016.1276418>. Ladson, A., Brown, R., Neal, B., & Nathan, R. J. (2013) <doi:10.7158/W12-028.2013.17.1>.

URL https://github.com/conradwasko/hydroEvents

License GPL-3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.2

Depends R (>= 2.10)

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

RemoteUrl https://github.com/conradwasko/hydroevents

RemoteRef HEAD

RemoteSha 8c67a0f45e14c2d6c316e6fce9345cb0fb8be54d

Contents

`baseflowA` *Baseflow removal (after Fuka et al. 2018)*

Description

This function caclualates baseflow using a recursive digital filter and is based on the implementation in the EcoHydRology package.

The formulation is orginally after Lyne and Hollick (1979) and described in Furey and Gupta (2001). Reccomended parameters are after Nathan and McMahon (1990).

Usage

```
baseflowA(q, alpha = 0.925, passes = 3)
```

Arguments

`q`	The vector series of streamflow
`alpha`	Filter parameter
`passes`	Number of passes

Value

A list of the baseflow and baseflow index at each timestep.

References

Fuka D. R., Walter, M.T., Archiblad, J.A., Steenhuis, T.S., & Easton, Z. M. (2018). A Community Modeling Foundation for Eco-Hydrology, R package version 0.4.12.1 Flow from Streamflow Time Series. Water Resources Research, 37(11), 2709–2722.

Furey, P., & Gupta, V. (2001). A Physically Based Filter for Spearating Base Flow from Streamflow Time Series. Water Resources Research, 37(11), 2709–2722.

Lyne, V., & Hollick, M. (1979). Stochastic time-variable rainfall-runoff modelling. Institute of Engineers Australia National Conference, 89-92.

Nathan, R. J., & McMahon, T. A. (1990). Evaluation of automated techniques for base flow and recession analyses. Water Resources Research, 26(7), 1465–1473.

Examples

```
library(hydroEvents)
data(dataBassRiver)
alpha.list = c(0, 0.9, 0.925, 0.95, 0.98, 0.987)
BFI.1 = numeric(length(alpha.list))
for (i in 1:length(alpha.list)) {
  bf.1 = baseflowA(dataBassRiver, alpha = alpha.list[i])
  BFI.1[i] = sum(bf.1$bf)/sum(dataBassRiver)
}
print(cbind(alpha.list, BFI.1))
```

`baseflowA` *Baseflow removal (after Fuka et al. 2018)*

Description

This function calculates baseflow using a recursive digital filter and is based on the implementation in the EcoHydRology package.

The formulation is originally after Lyne and Hollick (1979) and described in Furey and Gupta (2001). Recommended parameters are after Nathan and McMahon (1990).

Usage

```
baseflowA(q, alpha = 0.925, passes = 3)
```

Arguments

`q`	The vector series of streamflow
`alpha`	Filter parameter
`passes`	Number of passes

Value

A list of the baseflow and baseflow index at each timestep.

References

Fuka D. R., Walter, M.T., Archiblad, J.A., Steenhuis, T.S., & Easton, Z. M. (2018). A Community Modeling Foundation for Eco-Hydrology, R package version 0.4.12.1 Flow from Streamflow Time Series. Water Resources Research, 37(11), 2709–2722.

Furey, P., & Gupta, V. (2001). A Physically Based Filter for Spearating Base Flow from Streamflow Time Series. Water Resources Research, 37(11), 2709–2722.

Lyne, V., & Hollick, M. (1979). Stochastic time-variable rainfall-runoff modelling. Institute of Engineers Australia National Conference, 89-92.

Nathan, R. J., & McMahon, T. A. (1990). Evaluation of automated techniques for base flow and recession analyses. Water Resources Research, 26(7), 1465–1473.

Examples

```
library(hydroEvents)
data(dataBassRiver)
alpha.list = c(0, 0.9, 0.925, 0.95, 0.98, 0.987)
BFI.1 = numeric(length(alpha.list))
for (i in 1:length(alpha.list)) {
  bf.1 = baseflowA(dataBassRiver, alpha = alpha.list[i])
  BFI.1[i] = sum(bf.1$bf)/sum(dataBassRiver)
}
print(cbind(alpha.list, BFI.1))
```

---

baseflowB *Baseflow removal (after Ladson et al)*

Description

This function calculates baseflow using a recursive digital filter and is based on the implementation described in Ladson et al (2013).

Usage

```
baseflowB(q, alpha = 0.925, passes = 3, r = 30)
```

Arguments

`q`	The vector series of streamflow
`alpha`	Filter parameter
`passes`	Number of passes
`r`	number of points reflected at start and end of data set

Details

The reflected points act to resolve spin up issues and are removed before the baseflow is removed.

Value

A list of the baseflow and baseflow index at each timestep.

References

Ladson, A., Brown, R., Neal, B., & Nathan, R. (2013). A standard approach to baseflow separation using the Lyne and Hollick filter. Australian Journal of Water Resources, 17(1).

Examples

```
library(hydroEvents)
data(dataBassRiver)
alpha.list = c(0, 0.9, 0.925, 0.95, 0.98, 0.987)
BFI = numeric(length(alpha.list))
for (i in 1:length(alpha.list)) {
  bf = baseflowB(dataBassRiver, alpha = alpha.list[i])
  BFI[i] = sum(bf$bf)/sum(dataBassRiver)
}
print(cbind(alpha.list, BFI))
```

`calcStats` *Extract statistics from events*

Description

Given the start and end indices of events statistics are calculated for the values in between the start and end points inclusive.

Usage

```
calcStats(srt, end, data, f.vec = c("which.max", "max", "min"))
```

Arguments

`srt`	Vector of indices for the event start
`end`	Vector of indices for the event end
`data`	Vector of data
`f.vec`	c("which.max", "max", "min") Functions to be applied to the events

Value

Returns a dataframe where the row is each event and the column is each statistic. If `which.min` or `which.max` are called the indices returned are global, that is, relative to the start of `data`.

See Also

`eventPOT` `eventBaseflow` `eventMaxima` `eventMinima`

Examples

```
# Extract event statistics and plot the maxima
event.indices = eventPOT(dataLoch, out.style = "none")
event.stats = calcStats(event.indices$srt, event.indices$end, dataLoch)
print(event.stats)

plot(1:length(dataLoch), dataLoch, type = "h", lwd = 2, col = "steelblue",
     ylab = "Rainfall (mm)", xlab = "Time index", mgp = c(2, 0.6, 0))
points(event.stats$which.max, event.stats$max, col = "red", pch = 16, cex = 1.2)
legend("topright", legend = c("Rainfall", "Max"), cex = 0.8,
       lwd = c(2, NA), pch = c(NA, 16), col = c("steelblue", "red"), bty = "n")
```

dataBassRiver *Streamflow data*

Description

Streamflow data for Bass River at Loch (227219A) for 30/06/1974-04/09/1974

Usage

```
dataBassRiver
```

Format

A vector of 67 daily streamflow values in (ML/day)

Details

This data is obtained from Grayson et al (1996)

References

Grayson, R., Argent, R. M., Nathan, R. J., McMahon, T. A. & Mein, R. G. (1996) Hydrological Recipes, Cooperative Research Centre for Catchment Hydrology, Melbourne.

See Also

dataLoch

`dataCatchment` *Catchment data*

Description

Example data for five sites across Australia

Usage

```
dataCatchment
```

Format

A list with streamflow and catchment average precipitation and temperature for the following sites: 120301B, 602004, 235203, 410044, 105105A, corresponding to Arid, Mediterranean, Temperate, Subtropical, and Tropical climates. Catchment areas are 35326, 2433, 721, 1072, 297 km2 respectively. Streamflow is from the Australian Bureau of Meteorology Hydrologic Reference Station network and catchment average climate variables were extracted using AWAPer.

Source

http://www.bom.gov.au/water/hrs/

References

Peterson, T.J., Wasko, C., Saft., & Peel, M.C. (2020) AWAPer: An R package for area weighted catchment daily meteorological data anywhere within Australia, Hydrological Processes, 34, 1301-1306.

Jones, D., Wang, W., & Fawcett, R., 2009. High-quality spatial climate data-sets for Australia. Aust. Meteorol. Oceanogr. J. 58, 233–248.

`dataLoch` *Rainfall data*

Description

Rainfall data for Loch (Station ID 086067) for 30/06/1974-04/09/1974

Usage

```
dataLoch
```

Format

A vector of 67 daily rainfall values in (mm)

Source

http://www.bom.gov.au/climate/data/stations/

See Also

dataBassRiver

---

data_P_WL *Example sub-daily rainfall and tidal water level data*

Description

Hourly rainfall (P) and water level (WL) at Burnie, Tasmania for 1997-01-14 to 1997-02-14 (Pluvio ID: 091009; Tide gauge: IDO71005)

Usage

```
data_P_WL
```

Format

Each of P and WL data is a simple vector with no time stamp. The original data is in hourly time step.

Source

Sub-daily rainfall data are from Australian Bureau of Meteorology: http://www.bom.gov.au/climate/data/stations/ Sub-daily tidal water level data are from Australian Bureau of Meteorolgy Australian Baseline Sea Level Monitoring Project: http://www.bom.gov.au/oceanography/projects/abslmp/data/index.shtml

---

eventBaseflow *Event identification (using baseflow index)*

Description

Events are identified on the basis of the Baseflow Index (BFI).

Usage

```
eventBaseflow(
  data,
  BFI_Th = 0.5,
  bfi = baseflowB(data)$bfi,
  min.diff = 1,
  out.style = "summary"
)
```

Arguments

`data`	The data vector (e.g. a streamflow time series)
`BFI_Th`	Minimum BFI to identify baseflow
`bfi`	If no BFI is provided the BFI is calculated automatically using baseflowB
`min.diff`	Minimum length for an event
`out.style`	The type of output (currently either "summary" or "none")

Details

Any flow above the `BFI_Th` will be considered an event with a minimum event separation of `min.diff`.

Value

By default, the `out.style` returns the indices of the maximum in each event, as well as the value of the maximum and the sum of the `data` in each event, alongside the start and end of the events. Otherwise just the indices of start and end of events as a two column dataframe are returned.

References

Kaur, S., Horne, A., Stewardson, M.J., Nathan, R., Costa, A.M., Szemis, J.M., & Webb, J.A., (2017) Challenges for determining frequency of high flow spells for varying thresholds in environmental flows programmes. J. Ecohydraulics 2, 28–37.

See Also

`calcStats` `eventBaseflow` `eventMaxima` `eventPOT`

Examples

```
# Example
BFI_res = eventBaseflow(dataBassRiver, BFI_Th = 0.5, min.diff = 1)
```

`eventMaxima` *Event identification (using local maxima as a basis)*

Description

Events are identified on the basis of local maxima with an "event" considered to have occurred if the maxima is above a tolerable threshold of the neighbouring troughs/valleys.

Usage

```
eventMaxima(
  data,
  delta.y = 200,
  delta.x = 1,
  threshold = -1,
  out.style = "summary"
)
```

Arguments

`data`	The data vector
`delta.y`	Minimum allowable difference from a peak to a trough
`delta.x`	Minimum spacing between peaks
`threshold`	Value above which an event is considered to have occurred
`out.style`	The type of output (currently either "summary" or "none")

Details

If `delta.y` is negative it is applied a fractional decrease from the peak, otherwise it is treated as an absolute value. The `threshold` is applied after the event separation meaning that if a trough goes below the threshold but was originally considered one event it will continue to be considered one event. This makes this method distinct from the peaks over threshold algorithm in `eventPOT`. The `threshold` here should be thought of as a filter to remove trace amounts that are not part of an event rather than event separation metric.

Value

By default, the `out.style` returns the indices of the maximum in each event, as well as the value of the maximum and the sum of the `data` in each event, alongside the start and end of the events. Otherwise just the indices of start and end of events as a two column dataframe are returned.

See Also

`calcStats` `eventBaseflow` `eventMaxima` `eventPOT`

Examples

```
# Example extracting events from quickflow
bf = baseflowB(dataBassRiver, alpha = 0.925)
qf = dataBassRiver - bf$bf
events = eventMaxima(qf, delta.y = 200, delta.x = 1, threshold = 0)
print(events)
plotEvents(qf, dates = NULL, events = events, type = "lineover", main = "")
# Other examples to try
# delta.y = 200; delta.x = 1 # 5 events identified
# delta.y = 500; delta.x = 1 # 3 events identified
# delta.y = 10;  delta.x = 7 # 2 events identified
```

eventMinima *Event identification (using local minima as a basis)*

Description

Events are identified on the basis of local minima with an "event" considered to have occurred once the data has returned to within a threshold level of the start of the event.

Usage

```
eventMinima(
  data,
  delta.y = 20,
  delta.x = 5,
  threshold = -1,
  out.style = "summary"
)
```

Arguments

`data`	The data vector
`delta.y`	Maximum allowable difference between troughs
`delta.x`	Minimum length for an event
`threshold`	Value above which an event is considered to have occurred
`out.style`	The type of output (currently either "summary" or "none")

Details

The `threshold` is applied after the event separation meaning that if a trough goes below the threshold but was originally considered one event it will continue to be considered one event. This makes this method distinct from the peaks over threshold algorithm in `eventPOT`. The `threshold` here should be thought of as a filter to remove trace amounts that are not part of an event rather than event separation metric.

Value

By default, the `out.style` returns the indices of the maximum in each event, as well as the value of the maximum and the sum of the `data` in each event, alongside the start and end of the events. Otherwise just the indices of start and end of events as a two column dataframe are returned.

References

Tang, W., & Carey, S. K. (2017) HydRun: A MATLAB toolbox for rainfall-runoff analysis, Hydrological Processes (31) 2670-2682

See Also

calcStats eventBaseflow eventMaxima eventPOT

Examples

```
# Example extracting events from quickflow
bf = baseflowB(dataBassRiver, alpha = 0.925)
qf = dataBassRiver - bf$bf
events = eventMinima(qf, delta.x = 5, delta.y = 20)
print(events)
plotEvents(qf, dates = NULL, events = events, type = "lineover", main = "")
# delta.x = 5, delta.y = 20 # 5 events identified
# delta.x = 5, delta.y = 10 # 4 events identified
# delta.x = 1, delta.y = 20 # 6 events identified
```

eventPOT *Event identification (using a peak over threshold algorithm)*

Description

Identify events using a specified threshold value over which an event is considered to have occurred.

Usage

```
eventPOT(data, threshold = 0, min.diff = 1, out.style = "summary")
```

Arguments

data	A data vector
threshold	Value above which an event is considered to have occurred
min.diff	Spacing required for two events to be considered separate
out.style	The type of output (currently either "summary" or "none")

Details

The threshold can be thought of a value below which the data are considered to be "zero". The min.diff can be viewed as the minimum spacing for event independence.

Value

By default, the out.style returns the indices of the maximum in each event, as well as the value of the maximum and the sum of the data in each event, alongside the start and end of the events. Otherwise just the indices of start and end of events as a two column dataframe are returned.

See Also

calcStats eventBaseflow eventMaxima eventMinima

Examples

```
# Example using streamflow data
bf = baseflowB(dataBassRiver, alpha = 0.925)
qf = dataBassRiver - bf$bf
events = eventPOT(qf)
plotEvents(qf, dates = NULL, events = events, type = "lineover",
   main = "Events (plotted on quickflow)")
plotEvents(dataBassRiver, dates = NULL, events = events, type = "lineover",
   main = "Events (plotted on streamflow)")

# Examples using rainfall data
events = eventPOT(dataLoch, threshold = 0, min.diff = 1)
plotEvents(dataLoch, dates = NULL, events = events, type = "hyet",
   main = "Rainfall Events (threshold = 0, min.diff = 1)")

events = eventPOT(dataLoch, threshold = 2, min.diff = 2)
plotEvents(dataLoch, dates = NULL, events = events, type = "hyet",
   main = "Rainfall Events (threshold = 2, min.diff = 2)")
```

`limbs` *Extract rising/falling limbs*

Description

Identify the rising and falling limbs within each event (and optionally plot)

Usage

```
limbs(
  data,
  dates = NULL,
  events,
  to.plot = TRUE,
  ymin = min(data),
  ymax = max(data),
  xlab = "",
  ylab = "",
  main = ""
)
```

Arguments

`data`	The data vector (e.g. a streamflow time series)
`dates`	Date variable, default to NULL (inputting data as a simple vector)
`events`	Event extracted
`to.plot`	c(TRUE,FALSE) whether a plot is produced for the limbs
`ymin`	Minimum plot extend in vertical direction

ymax	Maximum plot extent in vertical direction
xlab	x-axis label
ylab	y-axis label
main	Plot title

Value

Returns indices of start and end of events and the rising/falling limbs within each event

Examples

```
# Example 1
library(hydroEvents)
qdata = WQ_Q$qdata[[1]]
BF_res = eventBaseflow(qdata$Q_cumecs)
limbs(data = qdata$Q_cumecs, dates = NULL, events = BF_res, main = ”with 'eventBaseflow'”)
BFI_res = eventBaseflow(dataBassRiver)

# Example 2
library(hydroEvents)
BFI_res = eventBaseflow(dataBassRiver)
d = as.Date(”1974-06-30”) + 0:(length(dataBassRiver)-1)
limbs(data = dataBassRiver, dates = NULL, events = BFI_res)
limbs(data = dataBassRiver, dates = d, events = BFI_res)
```

localMin *Local minima*

Description

Returns the index of local minima.

Usage

```
localMin(x)
```

Arguments

x	The data vector

Details

If values are repeated it returns the first index of occurrence. If the first value is repeated it is ignored as a local minima.

Value

Returns indices of local minima

Examples

```
# Find minima (with repeated values)
x = c(1, 2, 9, 9, 2, 1, 1, 5, 5, 1)
m = localMin(x)
plot(x, type = "l", lwd = 2, xlab = "", ylab = "", mgp = c(2, 0.6, 0))
points(m, x[m], pch = 16, col = "red")

# Find maxima (with repeated values)
x = c(1, 2, 9, 9, 2, 1, 1, 5, 5, 1)
m = localMin(-x)
plot(x, type = "l", lwd = 2, xlab = "", ylab = "", mgp = c(2, 0.6, 0))
points(m, x[m], pch = 16, col = "red")

# Minima in streamflow
m = localMin(dataBassRiver)
plot(dataBassRiver, type = "l", col = "steelblue", lwd = 2, ylab = "Flow (ML/d)",
   xlab = "Time index", mgp = c(2, 0.6, 0))
points(m, dataBassRiver[m], col = "red", pch = 16)

# Minima in quickflow
bf = baseflowA(dataBassRiver, alpha = 0.925)
qf = dataBassRiver - bf$bf
m = localMin(qf)
plot(qf, type = "l", lwd = 2, ylab = "Quickflow (ML/d)", xlab = "Time index", mgp = c(2, 0.6, 0))
points(m, qf[m], col = "red", pch = 16)

# Maxima in quickflow
bf = baseflowA(dataBassRiver, alpha = 0.925)
qf = dataBassRiver - bf$bf
m = localMin(-qf)
plot(qf, type = "l", lwd = 2, ylab = "Quickflow (ML/d)", xlab = "Time index", mgp = c(2, 0.6, 0))
points(m, qf[m], col = "red", pch = 16)
```

pairEvents *Pair Events*

Description

Pairing of events performed either forwards or backwards within specified lag times.

Usage

```
pairEvents(events.1, events.2, lag = 5, type = 1)
```

Arguments

`events.1`	Events of first data set
`events.2`	Events of second data set
`lag`	Maximum lag time (search radius) for pairing
`type`	Method used to pair events (see details)

Details

Pairing can be performed forwards and backwards and centrally.events.1 and events.2 need to be a dataframe with column names appropriate to the method type. That is, if pairing needs a time of maximum then "which.max" is expected (see examples). Column names are taken from the function event matching functions. The method types are:

- Type = 1: Search for the peak in events.2 within the start of event.1 to the end of event.1 + lag
- Type = 2: Search for an end in events.2 within the start of event.1 to the end of event.1 + lag
- Type = 3: Search for the peak in events.1 within the start of event.2 - lag to the peak in event.2
- Type = 4: Search for a start in events.1 within the start of event.2 - lag to the start of event.2
- Type = 5: Search for the peak in events.2 within the peak of event.1 - lag to the peak of event.1 + lag

It is appropriate to pick a lag time that is equivalent to the catchment time of concentration if matching rainfall to streamflow.

Value

Returns indices of start and end of events as well as the matched events as a four column dataframe.

See Also

calcStats eventBaseflow eventMaxima eventMinima eventPOT

Examples

```
# Load package
library(hydroEvents)
# Identify events
srt = as.Date("2015-02-05")
end = as.Date("2015-04-01")
idx = which(dataCatchment$`105105A`$Date >= srt & dataCatchment$`105105A`$Date <= end)
dat = dataCatchment$`105105A`[idx,]
events.P = eventPOT(dat$Precip_mm, threshold = 1, min.diff = 2)
events.Q = eventMaxima(dat$Flow_ML, delta.y = 2, delta.x = 1, thresh = 70)
# Plot events
oldpar <- par(mfrow = c(2, 1), mar = c(3, 2.7, 2, 1))
plotEvents(dat$Precip_mm, events = events.P, type = "hyet", colpnt = "#E41A1C",
   colline = "#E41A1C", ylab = "Precipitation (mm)", xlab = "Index", main = "2015")
plotEvents(dat$Flow_ML, events = events.Q, type = "lineover", colpnt = "#E41A1C",
   colline = "#377EB8", ylab = "Flow (ML/day)", xlab = "Index", main = "")
par(oldpar)
# Pair events
matched.1 = pairEvents(events.P, events.Q, lag = 5,  type = 1)
matched.2 = pairEvents(events.P, events.Q, lag = 5,  type = 2)
matched.3 = pairEvents(events.P, events.Q, lag = 3,  type = 3)
matched.4 = pairEvents(events.P, events.Q, lag = 7, type = 4)
matched.5 = pairEvents(events.P, events.Q, lag = 5, type = 5)
# Plot Pairs
oldpar <- par(mfrow = c(5, 1), mar = c(2, 3, 2, 3))
plotPairs(data.1 = dat$Precip_mm, data.2 = dat$Flow_ML, events = matched.1,
   col = rainbow(nrow(events.P)), ylab.1 = "P (mm)", ylab.2 = "Q (ML/day)", cex.2 = 0.66)
plotPairs(data.1 = dat$Precip_mm, data.2 = dat$Flow_ML, events = matched.2,
   col = rainbow(nrow(events.P)), ylab.1 = "P (mm)", ylab.2 = "Q (ML/day)", cex.2 = 0.66)
plotPairs(data.1 = dat$Precip_mm, data.2 = dat$Flow_ML, events = matched.3,
   col = rainbow(nrow(events.P)), ylab.1 = "Q (ML/day)", ylab.2 = "P (mm)", cex.2 = 0.66)
plotPairs(data.1 = dat$Precip_mm, data.2 = dat$Flow_ML, events = matched.4,
   col = rainbow(nrow(events.P)), ylab.1 = "Q (ML/day)", ylab.2 = "P (mm)", cex.2 = 0.66)
plotPairs(data.1 = dat$Precip_mm, data.2 = dat$Flow_ML, events = matched.5,
    col = rainbow(nrow(events.P)), ylab.1 = "P (mm)", ylab.2 = "Q ML/day)", cex.2 = 0.66)
par(oldpar)
```

plotEvents *Plot Events*

Description

Wrapper function for plotting identified events.

Usage

```
plotEvents(
  data,
  dates = NULL,
  events,
  type = "lineover",
  colline = "red",
  colpnt = "blue",
  colbound = "red",
  ymin = min(data),
  ymax = max(data),
  xlab = "",
  ylab = "",
  main = "events"
)
```

Arguments

`data`	The data vector
`dates`	Optional date vector
`events`	Events data frame
`type`	The type of plot (see details)

`colline`	Line colour
`colpnt`	Point colour
`colbound`	Background colour for plot type "bound"
`ymin`	Minimum plot extend in vertical direction
`ymax`	Maximum plot extent in vertical direction
`xlab`	x-axis label
`ylab`	y-axis label
`main`	Plot title

Details

Three plot types are implemented: `"lineover"`, `"bound"`, `"hyet"`. See examples. If events contains a column titled "which.max" the maxima are also plotted.

Value

No return value.

See Also

`eventBaseflow` `eventMaxima` `eventMinima` `eventPOT`

Examples

```
# Plot events
library(hydroEvents)
BFI_res = eventBaseflow(dataBassRiver)

oldpar <- par(mfrow = c(3, 1), mar = c(3, 2.7, 2, 1))
d = as.Date("1974-06-30") + 0:(length(dataBassRiver)-1)
plotEvents(data = dataBassRiver, dates = d, events = BFI_res,
   type = "lineover", xlab = "Date", ylab = "Flow (ML/day)", main = "lineover")
plotEvents(data = dataBassRiver, dates = d, events = BFI_res, type = "bound",
   xlab = "Date", ylab = "Flow (ML/day)", main = "bound")
plotEvents(data = dataBassRiver, dates = d, events = BFI_res, type = "hyet",
   xlab = "Date", ylab = "Flow (ML/day)", main = "hyet")
par(oldpar)
```

`plotPairs` *Plot Paired Events*

Description

Wrapper function for plotting paired events.

Usage

```
plotPairs(
  data.1,
  data.2,
  events,
  dates = NULL,
  type = "hyet",
  color.list = rainbow(nrow(events)),
  xlab = "",
  ylab.1 = "",
  ylab.2 = "",
  cex.2 = 1,
  main = ""
)
```

Arguments

`data.1`	The first data vector
`data.2`	The second data vector
`events`	The pairedeEvents data frame from pairEvents
`dates`	Optional date vector
`type`	The type of plot (see details)
`color.list`	Vector of colours used for plotting
`xlab`	x-axis label
`ylab.1`	primary y-axis label
`ylab.2`	secondary y-axis label
`cex.2`	cex for secondary y-axis label
`main`	Plot title

Details

If the type is `"hyet"` then `data.1` is plotted as a vertical lines and `data.2` as a line. If the type is `"lineover"` then all data is plotted as lines.

Value

No return value.

See Also

`pairEvents`

Examples

```
library(hydroEvents)
BFI_res = eventBaseflow(dataBassRiver)
POT_res = eventPOT(dataLoch)
pairs.1 = pairEvents(POT_res, BFI_res, type = 1, lag = 5)
pairs.3 = pairEvents(POT_res, BFI_res, type = 3, lag = 3)
d = as.Date("1974-06-30") + 0:(length(dataBassRiver)-1)
oldpar <- par(mar = c(3, 3.5, 2, 3.5), mfrow = c(2, 1))
plotPairs(dataLoch, dataBassRiver, pairs.1, dates = d, type = "hyet", xlab = "Date",
    ylab.1 = "Rain (mm)", ylab.2 = "Flow (ML/day)", main = "Matching Forward")
plotPairs(dataLoch, dataBassRiver, pairs.3, dates = d, type = "hyet", xlab = "Date",
    ylab.1 = "Flow (ML/day)", ylab.2 = "Rain (mm)", main = "Matching Backward")
par(oldpar)
```

WQ_Q *Example water quality and streamflow data*

Description

Data from 4 HRS (Hydrologic Reference Stations, Australian Bureau of Meteorology) catchments are included: catchment IDs: 410073, 424002, G8150018, A5020502.

Usage

```
WQ_Q
```

Format

Water quality (WQ) and streamflow (Q) data at matching time steps from 4 HRS catchments. Each dataset (qdata and wqdata) is a list of length 4, corresponding to the 4 catchments.

Source

HRS streamflow data: http://www.bom.gov.au/water/hrs/ water quality data: WaterNSW: https://realtimedata.waternsw.com.au/ Northern Territory Department of Environment, Parks and Water Security https://ntg.aquaticinformatics.net/AQWebportal/ South Australia Department for Environment and Water https://www.waterconnect.sa.gov.au/

Index