Package 'WRSS'

May 28, 2018

Type Package

Title Water Resources System Simulator

Depends R (>= 3.0.0), graphics, stats, Hmisc

Version 1.4

Date 2018-05-28

Author Rezgar Arabzadeh; Parisa Aberi; Kaveh Panaghi; Shahab Araghinejad; Majid Montaseri

Maintainer Rezgar Arabzadeh <rezgararabzadeh@ut.ac.ir>

Description Water resources system simulator is a tool for simulation and analysis of large-scale water resources systems. 'WRSS' proposes functions and methods for construction, simulation and analysis of primary water resources features (e.g. reservoirs, aquifers, and etc.) based on Standard Operating Policy (SOP).

License GPL-3

Imports ggplot2, GGally, network

Repository CRAN

NeedsCompilation no

Date/Publication 2018-05-28 14:38:28 UTC

R topics documented:

`WRSS-package` *Water Resources System Simulator*

Description

The WRSS is an object-oriented R package, which provides tools for simulation and analysis of large-scale supply hydrosystems. The package includes functions and methods for building, simulation, and visualization of water resources components.

Details

Package:	WRSS
Type:	Package
Version:	1.4
Date:	2018-05-28
License:	GPL-3

the package includes three major types of functions as follows:

1- functions for construction and manipulatation of water resources features:

a) `createArea`. constructor for basin/study area

b) `createJunction`. constructor for junction

c) `createRiver`. constructor for reach, rivers, and channels

d) `createReservoir`. constructor for reservoirs

e) `createDiversion`. constructor for diversions

f) `createAquifer`. constructor for aquifers

g) `createDemandSite`. constructor for demand sites

h) `addObjectToArea`. adds objects form mentioned above constructors to a basin inherited from class of `createBasin`

2- functions for analysis and operation of water resources objects using Standard Operating Policy (SOP):

a) `riverRouting`. river operation using

b) `reservoirRouting`. reservoir operation

c) `aquiferRouting`. aquifer operation

d) `diversionRouting`. diversion operation

e) `sim`. simulates an objects inherited from class of `createArea`

f) `rippl`. computes no-failure storage volume using the sequent peak algorithm(SPA)

3- functions for performance analysis and visualization.

a) `plot.sim`. plots the results of simulations for an object inherited from class of `sim`

b) `plot.createArea`. plots an object from class of `createArea`

c) `risk`. computes risk-based criateria for an object inherited from class of `sim`

Author(s)

Rezgar Arabzadeh; Parisa Aberi; Kaveh Panaghi; Shahab Araghinejad; Majid Montaseri

Maintainer: Rezgar Arabzadeh <rezgararabzadeh@ut.ac.ir>

References

Loucks, Daniel P., et al. Water resources systems planning and management: an introduction to methods, models and applications. Paris: Unesco, 2005.

See Also

`addObjectToArea`, `plot.sim`

addObjectToArea *Adds a feature to area*

Description

This function adds objects from the basin primary features to the object inherited from class of createArea.

Usage

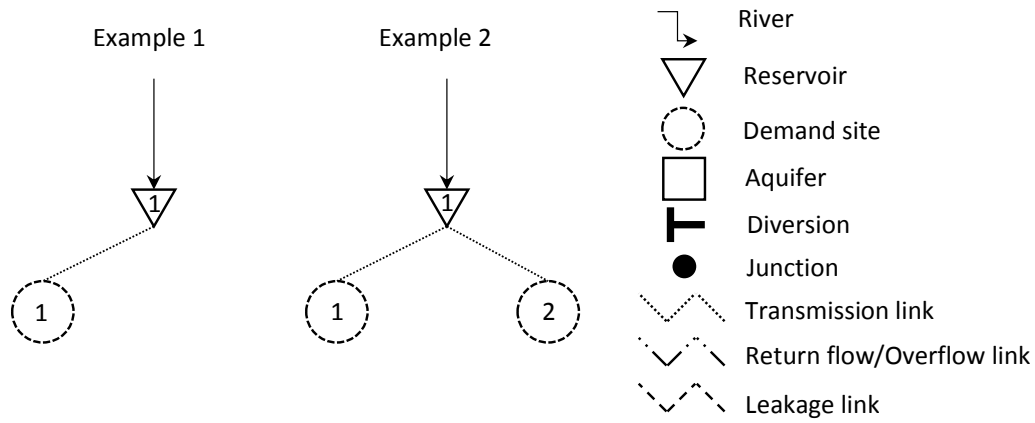
```
addObjectToArea(area, object)
```

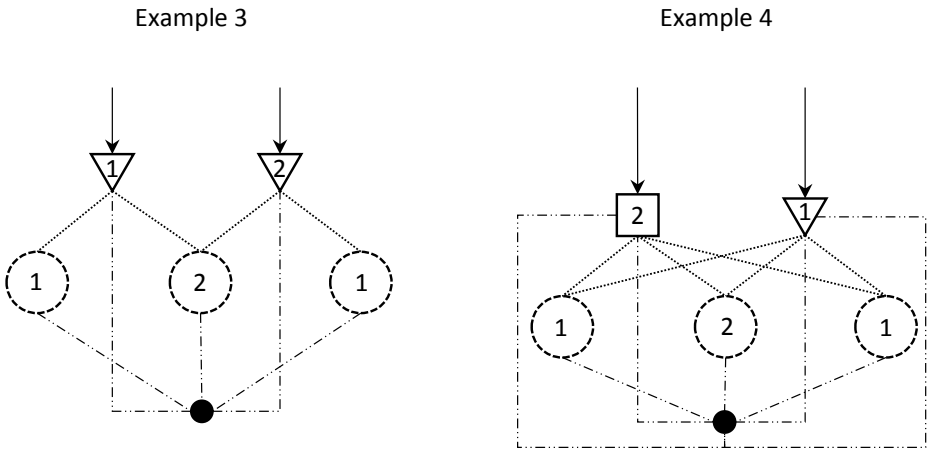
Arguments

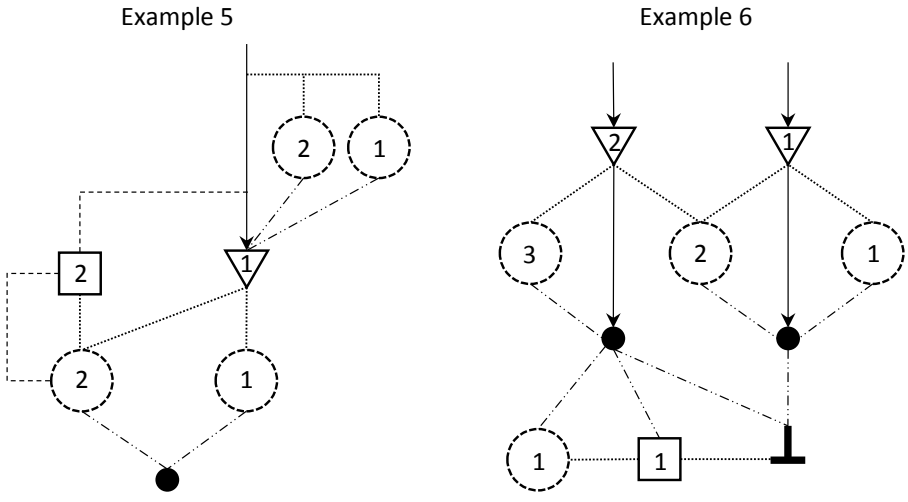
area	An object inherited from createArea
object	An objects inherited from any of the following constructors: createAquifer , createRiver, createReservoir, createJunction, createDiversion, and createDemandSite.

Details

The examples included in this documentation show construction and simulation of primary features of a water resources system using WRSS package. The Figure below presents schematic layouts attributed to the examples at the rest of the page:


Example 1
Example 2
River
Reservoir
Demand site
Aquifer
Diversion
Junction
Transmission link
Return flow/Overflow link
Leakage link


Example 3
Example 4


Example 5
Example 6

Value

an object from class of `createArea`

Author(s)

Rezgar Arabzadeh

References

Loucks, Daniel P., et al. Water resources systems planning and management: an introduction to methods, models and applications. Paris: Unesco, 2005.

See Also

`sim`

Examples

```
#--------------------1st Example--------------------
R<-createRiver(name="river1",label=1,downstream=2,discharge=rnorm(120,5,2))
Res<-createReservoir(name="res3",label=2,
                     priority=1,netEvaporation=rnorm(120,0.5,0.1),downstream =NA ,
                     geometry=list(deadStorage= 10 ,capacity= 90 ,
                     ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
D<-createDemandSite(name ="Agri5",label=5,demandTS=NA,
                    demandParams=list(annualUseRate=0.01,
                                      annualVariation=annualVariation,
                                      cropArea=5000),
                    returnFlowFraction =0.2,suppliers=2,
                    downstream=NA,priority=1)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start=c(1900,1),
                                 end  =c(1910,01)))
area<-addObjectToArea(area,R)
area<-addObjectToArea(area,Res)
area<-addObjectToArea(area,D)
## Not run:
plot(area)
simulated<-sim(area)
plot(simulated)

## End(Not run)


#--------------------2nd Example--------------------
R<-createRiver(name="Riv1",label=1,downstream=2,discharge=rnorm(120,4.5,1.5))
Res<-createReservoir(name="R1",label=2,priority=1,
                     netEvaporation=rnorm(120,0.5,0.1),downstream =NA,
                     geometry=list(deadStorage= 10,
```

```
                                  capacity= 20,
                                  ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="D1",label=5,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=3000),
                     suppliers=2,downstream=NA,priority=1)
D2<-createDemandSite(name ="D2",label=6,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=2000),
                     suppliers=2,downstream=NA,priority=2)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start=c(1900,1),end=c(1910,01)))
area<-addObjectToArea(area,R)
area<-addObjectToArea(area,Res)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
## Not run:
plot(area)
simulated<-sim(area)
plot(simulated)

## End(Not run)


#--------------------3rd Example--------------------
R1<-createRiver(name="Riv1",label=1,downstream=3,discharge=rnorm(120,4.5,1.5))
R2<-createRiver(name="Riv2",label=2,downstream=4,discharge=rnorm(120,6.5,2))
Res1<-createReservoir(name="R1",label=3,priority=1,
                      netEvaporation=rnorm(120,0.5,0.1),downstream =5,
                      geometry=list(deadStorage= 10,
                                    capacity= 20,
                                    ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
Res2<-createReservoir(name="R2",label=4,priority=2,
                      netEvaporation=rnorm(120,0.5,0.1),downstream =5,
                      geometry=list(deadStorage= 10 ,
                                    capacity= 30 ,
                                    ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
J1<-createJunction(name="j1",label=5,downstream=NA)
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="D1",label=6,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=3000),
                     returnFlowFraction=0.2,suppliers=3,
                     downstream=5,priority=1)
D2<-createDemandSite(name ="D2",label=7,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
```

```
                                    cropArea=3000),
                     returnFlowFraction=0.2,
                     suppliers=4,downstream=5,priority=1)
D3<-createDemandSite(name ="D3",label=8,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                    annualVariation=annualVariation,
                                    cropArea=4000),
                     returnFlowFraction=0.2,suppliers=c(3,4),
                     downstream=5,priority=2)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start=c(1900,1),end=c(1910,01)))
area<-addObjectToArea(area,R1)
area<-addObjectToArea(area,R2)
area<-addObjectToArea(area,Res1)
area<-addObjectToArea(area,Res2)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
area<-addObjectToArea(area,D3)
area<-addObjectToArea(area,J1)
## Not run:
plot(area)
simulated<-sim(area)
plot(simulated)

## End(Not run)


#--------------------4th Example--------------------
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
R1<-createRiver(name="Riv1",label=1,downstream=3,discharge=rnorm(120,4.5,1))
R2<-createRiver(name="Riv2",label=2,downstream=4,discharge=rnorm(120,2.5,0.25))
Res1<-createReservoir(name="R1",label=3,priority=1,
                      netEvaporation=rnorm(120,0.5,0.1),downstream =8,
                      seepageFraction=0.05, seepageCode=8,
                      geometry=list(deadStorage=10,
                                    capacity= 20,
                                    ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
Auq1<-createAquifer(name="Aquifer1",area=100,label=4,capacity=5000,
                    rechargeTS=rnorm(120,10,3),Sy=0.1,
                    leakageFraction=0.02,leakageCode=8,priority=2)
D1<-createDemandSite(name ="D1",label=5,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                    annualVariation=annualVariation,
                                    cropArea=3000),
                     returnFlowFraction=0.2,suppliers=c(3,4),
                     downstream=8,priority=1)
D2<-createDemandSite(name ="D2",label=6,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                    annualVariation=annualVariation,
                                    cropArea=3000),
                     returnFlowFraction=0.2,suppliers=c(3,4),
                     downstream=8,priority=1)
```

```
D3<-createDemandSite(name ="D3",label=7,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=4000),
                     returnFlowFraction=0.2,suppliers=c(3,4),
                     downstream=8,priority=2)
J1<-createJunction(name="j1",label=8,downstream=NA)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start=c(1900,1),end=c(1910,01)))
area<-addObjectToArea(area,R1)
area<-addObjectToArea(area,R2)
area<-addObjectToArea(area,Res1)
area<-addObjectToArea(area,Auq1)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
area<-addObjectToArea(area,D3)
area<-addObjectToArea(area,J1)
## Not run:
plot(area)
simulated<-sim(area)
plot(simulated)

## End(Not run)


#--------------------5th Example--------------------
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
R1<-createRiver(name="River1",label=1,
                downstream=2,discharge=rnorm(120,20,3),
                seepageFraction=0.1,seepageCode=3)
Res1<-createReservoir(name="Reservoir1",label=2,priority=1,
                      netEvaporation=rnorm(120,0.5,0.1),downstream=8,
                      seepageFraction =0.05,seepageCode=3,
                      geometry=list(deadStorage= 10 ,
                                    capacity= 20 ,
                                    ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
Auq1<-createAquifer(name="Aquifer1",area=50,label=3,
                    capacity=2000,rechargeTS=rnorm(120,10,3),
                    Sy=0.1,leakageFraction=0.02,leakageCode=6,priority=2)
D1<-createDemandSite(name ="Demand1",label=4,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=2000),
                     returnFlowFraction=0.2,suppliers=1,
                     downstream=2,priority=1)
D2<-createDemandSite(name ="Demand2",label=5,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=4000),
                     returnFlowFraction=0.2,suppliers=1,
                     downstream=2,priority=2)
D3<-createDemandSite(name ="Demand3",label=6,demandTS=NA,
```

```
                    demandParams=list(annualUseRate=0.01,
                                      annualVariation=annualVariation,
                                      cropArea=8000),
                    returnFlowFraction=0.2,suppliers=c(2,3),
                    downstream=8,priority=2)
D4<-createDemandSite(name ="Demand4",label=7,demandTS=NA,
                    demandParams=list(annualUseRate=0.01,
                                      annualVariation=annualVariation,
                                      cropArea=3000),
                    returnFlowFraction=0.2,suppliers=2,
                    downstream=8,priority=1)
J1<-createJunction(name="junction1",label=8,downstream=NA)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start=c(1900,1),end=c(1910,01)))
area<-addObjectToArea(area,R1)
area<-addObjectToArea(area,Res1)
area<-addObjectToArea(area,Auq1)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
area<-addObjectToArea(area,D3)
area<-addObjectToArea(area,D4)
area<-addObjectToArea(area,J1)
## Not run:
plot(area)
simulated<-sim(area)
plot(simulated)

## End(Not run)


#--------------------6th Example--------------------
R1<-createRiver(name="river1",label=1,
                downstream=3,discharge=rnorm(120,12,3))
R2<-createRiver(name="river2",label=2,
                downstream=4,discharge=rnorm(120,12,3))
R8<-createRiver(name="river8",label=8,
                downstream=10,discharge=rnorm(120,5,1))
R9<-createRiver(name="river9",label=9,
                downstream=11,discharge=rnorm(120,5,1))
Res3<-createReservoir(name="res3",label=3,priority=1,
                      netEvaporation=rnorm(120,0.5,0.1),downstream =8 ,
                      geometry=list(deadStorage= 10 ,
                      capacity= 90 ,
                      ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
Res4<-createReservoir(name="res4",label=4,priority=2,
                      netEvaporation=rnorm(120,0.5,0.1),downstream =9 ,
                      geometry=list(deadStorage= 10 ,
                                    capacity= 90 ,
                                    ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
D5<-createDemandSite(name ="Agri5",label=5,demandTS=NA,
                    demandParams=list(annualUseRate=0.01,
```

```
                                        annualVariation=annualVariation,
                                        cropArea=500),
                        returnFlowFraction =0.2,
                        suppliers=3,downstream=10,priority=1)
D6<-createDemandSite(name ="Agri6",label=6,demandTS=NA,
                        demandParams=list(annualUseRate=0.01,
                                        annualVariation=annualVariation,
                                        cropArea=700),
                        returnFlowFraction =0.2,suppliers=c(3,4),
                        downstream=10,priority=2)
D7<-createDemandSite(name ="Agri7",label=7,demandTS=NA,
                        demandParams=list(annualUseRate=0.01,
                                        annualVariation=annualVariation,
                                        cropArea=600),
                        returnFlowFraction =0.2,suppliers=4,
                        downstream=11,priority=3)
D13<-createDemandSite(name ="Agri13",label=13,demandTS=NA,
                         demandParams=list(annualUseRate=0.01,
                                          annualVariation=annualVariation,
                                          cropArea=300),
                         returnFlowFraction =0.2,suppliers=14,
                         downstream=10,priority=1)
J10<-createJunction(name="junc10",label=10,downstream=12)
J11<-createJunction(name="junc11",label=11,downstream=12)
Div12<-createDiversion(name="Div12",label=12,capacity=10,
                          divertTo=14,downstream=NA)
Auq14<-createAquifer(name="Aquifer14",area=100,label=14,
                        capacity=5000,rechargeTS=rnorm(120,10,3),
                        Sy=0.1,leakageFraction=0.02,leakageCode=10,priority=1)
area<-createArea(name="unknown",location="unknown",
                  simulation=list(start=c(1900,1),end=c(1910,01)))
area<-addObjectToArea(area,R1)
area<-addObjectToArea(area,R2)
area<-addObjectToArea(area,R8)
area<-addObjectToArea(area,R9)
area<-addObjectToArea(area,Res3)
area<-addObjectToArea(area,Res4)
area<-addObjectToArea(area,D5)
area<-addObjectToArea(area,D6)
area<-addObjectToArea(area,D7)
area<-addObjectToArea(area,D13)
area<-addObjectToArea(area,Div12)
area<-addObjectToArea(area,Auq14)
area<-addObjectToArea(area,J10)
area<-addObjectToArea(area,J11)
simulated<-sim(area)
## Not run:
plot(area)
plot(simulated)

## End(Not run)
```

aquiferRouting *base function for aquifer simulation*

Description

Given a sort of demand(s), aquiferRouting function simulates a lumped and simple model of an unconfined aquifer under an optional givn recharge time series, rechargeTS, and specific yield, Sy.

Usage

```
aquiferRouting(demand, priority = NA, area, capacity,
               rechargeTS = NA, leakageFraction = NA,
               initialStorage = NA, Sy, simulation)
```

Arguments

demand	A matrix: is column-wise matrix of demands, at which the rows presents demands for each monthly time step and columns are for different individual demand sites (MCM).
priority	A vector: is a vector of priorities associated to demand
area	The area of aquifer (Km^2)
capacity	The aquifer volume (MCM)
rechargeTS	A vector : a vector of water flowing into the aquifer (MCM)
leakageFraction	The leakage coeffcient of aquifer storage. The leakage is computed as the product of leakageFraction and aquifer storage
initialStorage	The initial volume of aquifer at the first step of the simulation (MCM). If missing, the function iterates to carry over the aquifer
Sy	Specific yield (default: 0.1)
simulation	A list: simulation is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

the aquiferRouting function returns a list of features discussed as folows: release: a matrix of release(s) equivalant to each demand (MCM) leakage: a vector of leakage time series (MCM) storage: a vector of storage time series (MCM)

Author(s)

Rezgar Arabzadeh

References

Mart nez-Santos, P., and J. M. Andreu. "Lumped and distributed approaches to model natural recharge in semiarid karst aquifers." Journal of hydrology 388.3 (2010): 389-398.

See Also

reservoirRouting

Examples

```
area           <-200
leakageFraction<-0.01
Sy             <-0.15
capacity       <-20000
priority       <-c(3,1,1,2)
rechargeTS     <-rnorm(120,60,8)
demand         <-matrix(rnorm(480,10,3),120)
simulation     <-list(start=c(2000,1),end=c(2010,1))

res<-
aquiferRouting(demand         =demand         ,
               priority       =priority       ,
               area           =area           ,
               capacity       =capacity       ,
               rechargeTS     =rechargeTS     ,
               leakageFraction=leakageFraction,
               Sy             =Sy             ,
               simulation     =simulation)

plot(ts(res$storage,start=simulation$start,frequency=12),ylab='Storage (MCM)')
```

`createAquifer` *Constructor for class of* `createAquifer`

Description

this function constructs an object from class of `createAquifer` that prescribes a simplified lupmped model of unconfined aquifer.

Usage

```
createAquifer(name, area, label, capacity,
              rechargeTS, Sy, leakageFraction,
              initialStorage, leakageCode, priority)
```

Arguments

`name`	(optional) A string: the name of the aquifer
`area`	The area of aquifer (Km^2)
`label`	An individual label assigned to the object as a reference code.
`capacity`	The aquifer volume (MCM)
`rechargeTS`	(optional) A vector : a vector of water flowing into the aquifer (MCM)
`Sy`	Specific yield (default: 0.1)
`leakageFraction`	(optional) The leakage coeffcient of aquifer storage. The leakage is computed as the product of `leakageFraction` and aquifer storage.
`initialStorage`	(optional) The initial volume of aquifer in the first step of the simulation (MCM). If missing, the function iterates to carry over the aquifer.
`leakageCode`	The code of an object which leakage volume pours to it.
`priority`	(optional) An integer: the supplying priority. `priority` is a value in [1, 99] interval. If missing, the `priority` is set to Inf.

Value

An object from class of `createAquifer`

Author(s)

Rezgar Arabzadeh

References

Mart nez-Santos, P., and J. M. Andreu. "Lumped and distributed approaches to model natural recharge in semiarid karst aquifers." Journal of hydrology 388.3 (2010): 389-398.

See Also

`addObjectToArea`

`createAquifer.base` *base function for class of* `createAquifer`

Description

this function constructs an object from class of `createAquifer` that prescribes a simplified lupmped model of unconfined aquifer.

Usage

```
## S3 method for class 'base'
createAquifer(name, area, label, capacity,
                rechargeTS, Sy,leakageFraction,
                initialStorage, leakageCode, priority)
```

Arguments

name	(optional) A string: the name of the aquifer
area	The area of aquifer (Km^2)
label	An individual label assigned to the object as a reference code.
capacity	The aquifer volume (MCM)
rechargeTS	(optional) A vector : a vector of water flowing into the aquifer (MCM)
Sy	Specific yield (default: 0.1)
leakageFraction	(optional) The leakage coeffcient of aquifer storage. The leakage is computed as the product of leakageFraction and aquifer storage.
initialStorage	(optional) The initial volume of aquifer in the first step of the simulation (MCM). If missing, the function iterates to carry over the aquifer.
leakageCode	The code of an object which leakage volume pours to it.
priority	(optional) An integer: the supplying priority. Is a value in [1, 99] interval. If missing, the priority is set to Inf.

Value

An object from class of list

See Also

createAquifer

---

createAquifer.default *default function for class of* createAquifer

---

Description

this function constructs an object from class of createAquifer that prescribes a simplified lupmped model of unconfined aquifer.

Usage

```
## Default S3 method:
createAquifer(name = "Aquifer1", area, label, capacity,
                rechargeTS = NA, Sy = 0.1, leakageFraction = NA,
                initialStorage = NA, leakageCode, priority=NA)
```

Arguments

name	(optional) A string: the name of the aquifer
area	The area of aquifer (Km^2)
label	An individual label assigned to the object as a reference code.
capacity	The aquifer volume (MCM)
rechargeTS	(optional) A vector : a vector of water flowing into the aquifer (MCM)
Sy	Specific yield (default: 0.1)
leakageFraction	(optional) The leakage coeffcient of aquifer storage. The leakage is computed as the product of leakageFraction and aquifer storage.
initialStorage	(optional) The initial volume of aquifer in the first step of the simulation (MCM). If missing, the function iterates to carry over the aquifer.
leakageCode	The code of an object which leakage volume pours to it.
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.

Value

An object from class of createAquifer

See Also

createAquifer

---

createArea *Constructor for class of* createArea

Description

this function constructs an object from class of createArea, supporting objects inherited from any of the folowing classes: createAquifer, createDemandSite, createDiversion, createJunction, createReservoir,and createRiver.

Usage

```
createArea(name, location, simulation)
```

Arguments

name	(optional) A string: the name of the aquifer
location	(optional) A string: the physical location of name
simulation	A list: simulation is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

An object from class of createArea

Author(s)

Rezgar Arabzadeh

See Also

addObjectToArea

---

createArea.base *base function for class of* createArea

Description

this function constructs an object from class of createArea, supporting objects inherited from any of the folowing classes: createAquifer, createDemandSite, createDiversion, createJunction, createReservoir,and createRiver.

Usage

```
## S3 method for class 'base'
createArea(name, location, simulation)
```

Arguments

name	(optional) A string: the name of the aquifer
location	(optional) A string: the physical location of name
simulation	A list: simulation is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

An object from class of list

See Also

createArea

createArea.default *default function for class of* createArea

Description

this function constructs an object from class of createArea, supporting objects inherited from the any of folowing classes: createAquifer, createDemandSite, createDiversion, createJunction, createReservoir,and createRiver.

Usage

```
## Default S3 method:
createArea(name = "unknown", location = "unknown",
               simulation = list(start = NULL, end = NULL))
```

Arguments

name	(optional) A string: the name of the aquifer
location	(optional) A string: the physical location of createArea
simulation	A list: simulation is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

An object from class of createArea

See Also

createArea

createDemandSite *Constructor for class of* createDemandSite

Description

this function constructs an object from class of createDemandSite, which represents a demand site such as domestic, agricultural, and etc, with a specified demand time series.

Usage

```
createDemandSite(name, label, demandTS, demandParams,
                 returnFlowFraction, suppliers,
                 downstream, priority)
```

Arguments

name	(optional) A string: the name of the demand site
label	An individual label assigned to the object as a reference code.
demandTS	A vector: a vector of demand time series (MCM). If demandParams is null, demandTS is compulsory, otherwise the demandTS is not needed!.
demandParams	A list: If demandTS is missing, demandParams should not be omitted. demandParams includes three parts as folows: annualUseRate: The annual water demand per hectare (MCM). annualVariation: the precentage of water demand distribution within a year (the precentages in each month) cropArea: the area of cropping farms (hectare).
returnFlowFraction	(optional) returnFlowFraction is fraction of total supplied water to the demand site. The return flow is computed as the product of returnFlowFraction and the amount of water the demand sites receives. returnFlowFraction must be in [0, 1] interval.
suppliers	the reference code number(s) of existing suppliers (objects inherited from the folowing classes: createAquifer, createRiver, createReservoir, createDiversion).
downstream	The code of an object which return flow volume pours to it.
priority	An integer: the priority to be supplied. A value in [1, 99] interval.

Value

An object from class of createDemandSite

Author(s)

Rezgar Arabzadeh

See Also

addObjectToArea

---

createDemandSite.base *base function for class of* createDemandSite

---

Description

this function constructs an object from class of createDemandSite, which represents a demand site such as domestic, agricultural, and etc, with a specified demand time series.

Usage

```
## S3 method for class 'base'
createDemandSite(name, label, demandTS, demandParams,
                  returnFlowFraction, suppliers,
                  downstream, priority)
```

Arguments

name	(optional) A string: the name of the demand site
label	An individual label assigned to the object as a reference code.
demandTS	A vector: a vector of demand time series (MCM). If demandParams is null, demandTS is compulsory, otherwise the demandTS is not needed!.
demandParams	A list: If demandTS is missing, demandParams should not be omitted. demandParams includes three parts as folows: annualUseRate: The annual water demand per hectare (MCM). annualVariation: the precentage of water demand distribution within a year (the precentages in each month) cropArea: the area of cropping farms (hectare).
returnFlowFraction	(optional) returnFlowFraction is fraction of total supplied water to the demand site. The return flow is computed as the product of returnFlowFraction and the amount of water the demand sites receives. returnFlowFraction must be in [0, 1] interval.
suppliers	the reference code number(s) of existing suppliers (objects inherited from the folowing classes: createAquifer, createRiver, createReservoir, createDiversion).
downstream	The code of an object which return flow volume pours to it.
priority	An integer: the priority to be supplied. A value in [1, 99] interval.

Value

An object from class of list

See Also

createDemandSite

---

createDemandSite.default

default function for class of createDemandSite

Description

this function constructs an object from class of createDemandSite, which represents a demand site such as domestic, agricultural, and etc, with a specified demand time series.

Usage

```
## Default S3 method:
createDemandSite(name  ="Unknown",
         label                                  ,
         demandTS          =NA                  ,
         demandParams=list(annualUseRate=NULL   ,
                           annualVariation=NULL,
```

```
                        cropArea=NULL)      ,
          returnFlowFraction =0.0           ,
          suppliers                         ,
          downstream        =NA             ,
          priority          =NA)
```

Arguments

name	(optional) A string: the name of the demand site
label	An individual label assigned to the object as a reference code.
demandTS	A vector: a vector of demand time series (MCM). If demandParams is null, demandTS is compulsory, otherwise the demandTS is not needed!.
demandParams	A list: If demandTS is missing, demandParams should not be omitted. demandParams includes three parts as folows: annualUseRate: The annual water demand per hectare (MCM). annualVariation: the precentage of water demand distribution within a year (the precentages in each month) cropArea: the area of cropping farms (hectare).
returnFlowFraction	(optional) returnFlowFraction is fraction of total supplied water to the demand site. The return flow is computed as the product of returnFlowFraction and the amount of water the demand sites receives. returnFlowFraction must be in [0, 1] interval.
suppliers	the reference code number(s) of existing suppliers (objects inherited from the folowing classes: createAquifer, createRiver, createReservoir, createDiversion).
downstream	The code of an object which return flow volume pours to it.
priority	An integer: the priority to be supplied. A value in [1, 99] interval.

Value

An object from class of createDemandSite

See Also

createDemandSite

---

createDiversion *Constructor for class of* createDiversion

Description

this function constructs an object from class of createDiversion, acting as a diversion dam which is able to divert water up to a specified capacity.

Usage

```
createDiversion(name, label, capacity,
                divertTo, downstream, priority)
```

Arguments

name	(optional) A string: the name of the diversion
label	An individual label assigned to the object as a reference code.
capacity	The maximum capacity of diversion dam (CMS).
divertTo	The code of an object which recieves the diverted water volume.
downstream	The code of an object which overflow volume pours to it.
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.

Value

An object from class of createDiversion

Author(s)

Rezgar Arabzadeh

See Also

addObjectToArea

---

createDiversion.base *base function for class of* createDiversion

---

Description

this function constructs an object from class of createDiversion, acting as a diversion dam which is able to divert water up to a specified capacity.

Usage

```
## S3 method for class 'base'
createDiversion(name, label, capacity,
                divertTo, downstream, priority)
```

Arguments

name	(optional) A string: the name of the diversion
label	An individual label assigned to the object as a reference code.
capacity	The maximum capacity of diversion dam (CMS).
divertTo	The code of an object which recieves the diverted water volume.
downstream	The code of an object which overflow volume pours to it.
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.

Value

An object from class of `list`

See Also

`createDiversion`

`createDiversion.default`

default function for class of `createDiversion`

Description

this function constructs an object from class of `createDiversion`, acting as a diversion dam which is able to divert water up to a specified capacity.

Usage

```
## Default S3 method:
createDiversion(name = "junc1", label, capacity,
                    divertTo, downstream = NA, priority=NA)
```

Arguments

`name`	(optional) A string: the name of the diversion
`label`	An individual label assigned to the object as a reference code.
`capacity`	The maximum capacity of diversion dam (CMS).
`divertTo`	The code of an object which recieves the diverted water volume.
`downstream`	The code of an object which overflow volume pours to it.
`priority`	(optional) An integer: the supplying priority. `priority` is a value in [1, 99] interval. If missing, the `priority` is set to `Inf`.

Value

An object from class of `createDiversion`

See Also

`createDiversion`

createJunction *Constructor for class of* createJunction

Description

this function constructs an object from class of `createDiversion`, acting as a junction in the basin which is able to aggregate outflow water from upper tributaries and/or objects in the upstream.

Usage

```
createJunction(name, label, downstream)
```

Arguments

`name`	(optional) A string: the name of the junction
`label`	An individual label assigned to the object as a reference code.
`downstream`	The code of an object which outflow volume pours to it.

Value

An object from class of `createJunction`

Author(s)

Rezgar Arabzadeh

See Also

`addObjectToArea`

createJunction.base *base function for class of* createJunction

Description

this function constructs an object from class of `createDiversion`, acting as a junction in the basin which is able to aggregate outflow water from upper tributaries and/or objects in the upstream.

Usage

```
## S3 method for class 'base'
createJunction(name, label, downstream)
```

Arguments

name	(optional) A string: the name of the junction
label	An individual label assigned to the object as a reference code.
downstream	The code of an object which outflow volume pours to it.

Value

An object from class of list

See Also

createJunction

---

createJunction.default

default function for class of createJunction

Description

this function constructs an object from class of createDiversion, acting as a junction in the basin which is able to aggregate outflow water from upper tributaries and/or objects in the upstream.

Usage

```
## Default S3 method:
createJunction(name = "junc1", label, downstream = NA)
```

Arguments

name	(optional) A string: the name of the junction
label	An individual label assigned to the object as a reference code.
downstream	The code of an object which outflow volume pours to it.

Value

An object from class of list

See Also

createJunction

createReservoir *Constructor for class of* createReservoir

Description

this function constructs an object from class of createReservoir, which is able to simulate a storage reservoir under given a sort of demand(s).

Usage

```
createReservoir(name, label, priority,
                netEvaporation, downstream,
                initialStorage, seepageFraction,
                seepageCode, geometry)
```

Arguments

name	(optional) A string: the name of the reservoir
label	An individual label assigned to the object as a reference code.
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.
netEvaporation	A vector: is a vector of net evaporation depth time series at the location of dam site (meter).
downstream	The code of an object which spilage volume pours to it.
initialStorage	(optional) The initial stored water at the reservoir in the first step of the simulation (MCM). If is missing the the function iterate to carry over the aquifer.
seepageFraction	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of seepageFraction and reservoir storage.
seepageCode	The code of an object which seepage volume pours to it.
geometry	A list of reservoir geometric specifications: deadStorage: refers to water in a reservoir that cannot be drained by gravity through a dam's outlet works (MCM) capacity: The maximum capacity of the reservoir ratingCurve: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (Km^2) corresponding to the first column.

Value

An object from class of createReservoir

Author(s)

Rezgar Arabzadeh

See Also

addObjectToArea

---

createReservoir.base *base function for class of* createReservoir

---

Description

this function constructs an object from class of createReservoir, which is able to simulate a storage reservoir under given a sort of demand(s).

Usage

```
## S3 method for class 'base'
createReservoir(name           ,
                               label          ,
                               priority       ,
                               netEvaporation ,
                               downstream     ,
                               initialStorage ,
                               seepageFraction,
                               seepageCode    ,
                               geometry)
```

Arguments

name	(optional) A string: the name of the reservoir
label	An individual label assigned to the object as a reference code.
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.
netEvaporation	A vector: is a vector of net evaporation depth time series at the location of dam site (meter).
downstream	The code of an object which spilage volume pours to it.
initialStorage	(optional) The initial stored water at the reservoir in the first step of the simulation (MCM). If is missing the the function iterate to carry over the aquifer.
seepageFraction	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of seepageFraction and reservoir storage.
seepageCode	The code of an object which seepage volume pours to it.
geometry	A list of reservoir geometric specifications: deadStorage: refers to water in a reservoir that cannot be drained by gravity through a dam's outlet works (MCM) capacity: The maximum capacity of the reservoir ratingCurve: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (Km^2) corresponding to the first column.

Value

An object from class of `list`

See Also

`createReservoir`

`createReservoir.default`

default function for class of `createReservoir`

Description

this function constructs an object from class of `createReservoir`, which is able to simulate a storage reservoir under given a sort of demand(s).

Usage

```
  ## Default S3 method:
createReservoir(name         ="resrvoir1",
         label                                ,
         priority        =NA                  ,
         netEvaporation                       ,
         downstream =NA                       ,
         initialStorage  =NA                  ,
         seepageFraction =NA                  ,
         seepageCode     =NA                  ,
         geometry   =list(deadStorage    = NULL ,
                          capacity       = NULL ,
                          ratingCurve    = NULL))
```

Arguments

`name`	(optional) A string: the name of the reservoir.
`label`	An individual label assigned to the object as a reference code.
`priority`	(optional) An integer: the supplying priority. `priority` is a value in [1, 99] interval. If missing, the `priority` is set to Inf.
`netEvaporation`	A vector: is a vector of net evaporation depth time series at the location of dam site (meter).
`downstream`	The code of an object which spilage volume pours to it.
`initialStorage`	(optional) The initial stored water at the reservoir in the first step of the simulation (MCM). If is missing the the function iterate to carry over the aquifer.
`seepageFraction`	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of `seepageFraction` and reservoir storage.

seepageCode	The code of an object which seepage volume pours to it.
geometry	A list of reservoir geometric specifications: deadStorage: refers to water in a reservoir that cannot be drained by gravity through the dam outlet works (MCM); capacity: The maximum capacity of the reservoir; ratingCurve: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (Km^2) corresponding to the first column.

Value

An object from class of createReservoir

See Also

createReservoir

---

createRiver *Constructor for class of* createRiver

Description

this function constructs an object from class of createRiver, which is able to act as a chanel or resource to supply a seort of demand(s).

Usage

```
createRiver(name, label, downstream, seepageFraction,
            seepageCode, discharge, priority)
```

Arguments

name	(optional) A string: the name of the river
label	An individual label assigned to the object as a reference code.
downstream	The code of an object which outflow volume pours to it.
seepageFraction	(optional) The seepage coeffcient of river discharge flow. The seepage is computed as the product of seepageFraction and river discharge.
seepageCode	The code of an object which seepage volume pours to it.
discharge	(optional) A vector: is a vector of river discharge time series (MCM).
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.

Value

An object from class of createRiver

Author(s)

Rezgar Arabzadeh

See Also

addObjectToArea

`createRiver.base` *base function for class of* `createRiver`

Description

this function constructs an object from class of `createRiver`, which is able to act as a chanel or resource to supply a seort of demand(s).

Usage

```
## S3 method for class 'base'
createRiver(name, label, downstream, seepageFraction,
                seepageCode, discharge, priority)
```

Arguments

`name`	(optional) A string: the name of the river
`label`	An individual label assigned to the object as a reference code.
`downstream`	The code of an object which outflow volume pours to it.
`seepageFraction`	(optional) The seepage coeffcient of river discharge flow. The seepage is computed as the product of `seepageFraction` and river discharge.
`seepageCode`	The code of an object which seepage volume pours to it.
`discharge`	(optional) A vector: is a vector of river discharge time series (MCM).
`priority`	(optional) An integer: the supplying priority. `priority` is a value in [1, 99] interval. If missing, the `priority` is set to `Inf`.

Value

An object from class of `list`

See Also

createRiver

createRiver.default *default function for class of* createRiver

Description

this function constructs an object from class of createRiver, which is able to act as a chanel or resource to supply a seort of demand(s).

Usage

```
## Default S3 method:
createRiver(name = "river1", label, downstream = NA,
               seepageFraction = NA, seepageCode = NA,
               discharge, priority = NA)
```

Arguments

name	(optional) A string: the name of the river
label	An individual label assigned to the object as a reference code.
downstream	The code of an object which outflow volume pours to it.
seepageFraction	(optional) The seepage coeffcient of river discharge flow. The seepage is computed as the product of seepageFraction and river discharge.
seepageCode	The code of an object which seepage volume pours to it.
discharge	(optional) A vector: is a vector of river discharge time series (MCM).
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.

Value

An object from class of createRiver

See Also

createRiver

`diversionRouting` *base function for diversion simulation*

Description

Given a sort of demand(s), `diversionRouting` function enable us to simulate the performance and effect of a diversion dam under a givn recharge time series, `inflow`, on the drainage network.

Usage

```
diversionRouting(demand, priority = NA,
                 capacity, inflow, simulation)
```

Arguments

`demand`	A matrix: is column-wise matrix of demands, at which the rows presents demands for each monthly time step and columns are for different individual demand sites (MCM).
`priority`	A vector: is a vector of priorities associated to `demand`
`capacity`	The maximum capacity of diversion dam (CMS).
`inflow`	A vector : a vector of water flowing into the diversion (MCM)
`simulation`	A list: `simulation` is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

the `diversionRouting` function returns a list of features discussed as folows: `release` : a matrix of release(s) equivalant to each `demand` (MCM) `diverted`: a vector of diverted volumes (MCM), release(s) are included `overflow`: a vector of overflow passing the diversion (MCM)

Author(s)

Rezgar Arabzadeh

See Also

`aquiferRouting`

Examples

```
demand         <-matrix(rnorm(480,10,3),120)
priority       <-sample(1:3,4,replace=TRUE)
capacity       <-12
inflow         <-rlnorm(120,log(50),log(4))
simulation     <-list(start=c(2000,1),end=c(2010,1))
diversionRouting(demand=demand,
```

```
                priority=priority,
                capacity=capacity,
                inflow=inflow,
                simulation=simulation)
```

`plot.createArea` *plot method for an object from class of* `createArea`

Description

plot method for objects inherited from class of `createArea`

Usage

```
## S3 method for class 'createArea'
plot(x,...)
```

Arguments

`x`	an object from class of `createArea`
`...`	other objects that can be passed to `plot` function

Author(s)

Rezgar Arabzadeh

See Also

`createArea`

`plot.sim` *plot method for an WRSS object*

Description

plot method for objects inherited from class of `sim`

Usage

```
## S3 method for class 'sim'
plot(x,...)
```

Arguments

`x`	an object from class of `sim`
`...`	other objects that can be passed to `plot` function

Author(s)

Rezgar Arabzadeh

See Also

sim

`reservoirRouting` *base function for reservoir simulation*

Description

Given a sort of demand(s), `reservoirRouting` function enable us to simulate the performance and effect of a dam under givn hydrometeorological time series, e.g. `inflow` and `netEvaporation`, on the drainage network.

Usage

```
reservoirRouting(demand, priority, inflow, netEvaporation,
                 geometry = list(deadStorage=NULL        ,
                                 capacity   =NULL        ,
                                 ratingCurve=NULL)       ,
                 initialStorage = NA                     ,
                 seepageFraction = NA, simulation)
```

Arguments

`demand`	A matrix: is column-wise matrix of demands, at which the rows presents demands for each monthly time steps and columns are for different individual demand sites (MCM).
`priority`	A vector: is a vector of priorities associated to `demand`
`inflow`	A vector : a vector of water flowing into the diversion (MCM)
`netEvaporation`	A vector: is a vector of net evaporation depth time series at the location of dam site (meter).
`geometry`	A list of reservoir geometric specifications: `deadStorage`: refers to water in a reservoir that cannot be drained by gravity through a dam's outlet works (MCM). `capacity`: The maximum capacity of the reservoir. `ratingCurve`: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (Km^2) corresponding to the first column.
`initialStorage`	(optional) The initial stored water at the reservoir in the first step of the simulation (MCM). If is missing the the function iterate to carry over the aquifer.
`seepageFraction`	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of `seepageFraction` and reservoir storage.

`simulation` A list: `simulation` is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

the `reservoirRouting` function returns a list of features given as folows: `release`: a matrix of release(s) equivalant to each `demand` (MCM) `spill` : a vector of spilage time series (MCM) `seepage`: a vector of steepage time series (MCM) `storage`: a vector of storage time series (MCM) `loss` : a vector of evaporation loss time series (MCM)

Author(s)

Rezgar Arabzadeh

References

Yeh, William WG. "Reservoir management and operations models: A state of the art review." Water resources research 21.12 (1985): 1797-1818.

See Also

`aquiferRouting`

Examples

```
demand         <-matrix(rnorm(480,10,3),120)
priority       <-sample(1:3,4,replace=TRUE)
inflow         <-rlnorm(120,log(50),log(4))
netEvaporation <-rnorm(120,0.4,0.1)
simulation     <-list(start=c(2000,1),end=c(2010,1))
seepageFraction<-0.05
geometry       <-list(deadStorage= 50,capacity= 100,
                      ratingCurve= cbind(seq(0,100,10),seq(0,10,1)))
reservoirRouting(demand=demand,
                 priority=priority,
                 inflow=inflow,
                 netEvaporation=netEvaporation,
                 geometry=geometry,
                 seepageFraction=seepageFraction,
                 simulation=simulation)
```

`rippl` *Rippl's method*

Description

Computes the Rippl-no-failure storage for given set of discharges and target.

Usage

```
rippl(discharge,target,plot=TRUE)
```

Arguments

discharge	a vector of natural discharge at the reservoir site.
target	a vector of demand time series with length equal that of discharge. If the time scale doesn't match, the target will be cycled or truncated.
plot	logical: whether plot the Rippl's method process or merely report the result.

Value

no-failure storage value for the given time series, discharge and target.

References

Rippl, Wengel. The capacity of storage reservoirs for water supply. Van Nostrand's Engineering Magazine (1879-1886) 29.175 (1883): 67.

See Also

sim

Examples

```
## Not run:
rippl(Nile,mean(Nile)*0.95)

## End(Not run)
```

---

risk *risk-based criteria*

Description

this function returns risk-based criteria for demand site(s) built-in the object inherited from class of sim.

Usage

```
risk(object , s.const = 0.95)
```

Arguments

object	an object from class of sim
s.const	satisfactory constant: a value in [0, 1] interval, which refers to the level at which if a demand is supplied over the s.const is considered fully supplied.

Details

This function computes the riks criteria based on the formulations proposed by Hashimoto et.al (1982).

Value

a matrix of criteria

Author(s)

Rezgar Arabzadeh

References

Hashimoto, Tsuyoshi, Jery R. Stedinger, and Daniel P. Loucks. "Reliability, resiliency, and vulnerability criteria for water resource system performance evaluation." Water resources research 18.1 (1982): 14-20.

See Also

sim

Examples

```
R<-createRiver(name="Riv1",label=1,downstream=2,discharge=rnorm(120,4.5,1.5))
Res<-createReservoir(name="R1",label=2,priority=1,
                     netEvaporation=rnorm(120,0.5,0.1),downstream =NA,
                     geometry=list(deadStorage= 10,
                                   capacity= 20,
                                   ratingCurve= cbind(seq(0,90,10),seq(0,9,1))))
annualVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="D1",label=5,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=3000),
                     suppliers=2,downstream=NA,priority=1)
D2<-createDemandSite(name ="D2",label=6,demandTS=NA,
                     demandParams=list(annualUseRate=0.01,
                                       annualVariation=annualVariation,
                                       cropArea=2000),
                     suppliers=2,downstream=NA,priority=2)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start=c(1900,1),end=c(1910,01)))
area<-addObjectToArea(area,R)
area<-addObjectToArea(area,Res)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
risk(sim(area))
```

`riverRouting` *base function for rivers and reachs simulation*

Description

Given a sort of demand(s), `riverRouting` function enable us to simulate rivers and channels under givn a hydrologic time series, `inflow`, and optional `demand`(s).

Usage

```
riverRouting(demand, priority = NA, discharge = NA, simulation)
```

Arguments

`demand`	A matrix: is column-wise matrix of demands, at which the rows presents demands for each monthly time steps and columns are for different individual demand sites (MCM).
`priority`	A vector: is a vector of priorities associated to `demand`
`discharge`	A vector : a vector of water flowing into the diversion (MCM)
`simulation`	A list: `simulation` is a list which includes two vectors of start and end of simulation interval respectively, at which each vector contains two number equivalant to year and month.

Value

the `riverRouting` returns a matrix of release(s) corresponding to each demand(s).

Author(s)

Rezgar Arabzadeh

See Also

`diversionRouting`

Examples

```
demand         <-matrix(rnorm(480,10,3),120)
priority       <-sample(1:3,4,replace=TRUE)
discharge      <-rlnorm(120,log(50),log(4))
simulation     <-list(start=c(2000,1),end=c(2010,1))

riverRouting(demand    = demand   ,
             priority  = priority ,
             discharge = discharge,
             simulation= simulation)
```

`sim` *Constructor for class of* `sim`

Description

`sim` simulates an object inherited from class of `createArea` using Standard Operating Policy (SOP).

Usage

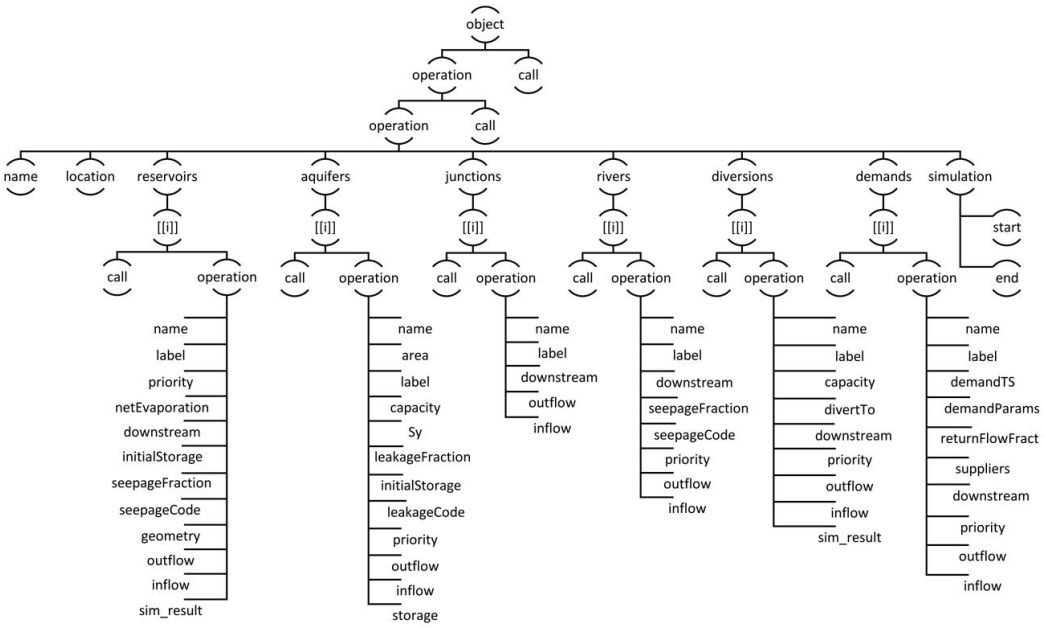
```
sim(object)
```

Arguments

`object`	an object inherited from class of `createArea`.

Value

an object inherited from class of `sim`. Address keys to access components built-in an object inherited from class of `sim` is as figure below:



Author(s)

Rezgar Arabzadeh

References

Loucks, Daniel P., et al. Water resources systems planning and management: an introduction to methods, models and applications. Paris: Unesco, 2005.

See Also

addObjectToArea

---

sim.base *base function for class of* sim

Description

sim simulates an object inherited from class of createArea using Standard Operating Policy (SOP).

Usage

```
## S3 method for class 'base'
sim(object)
```

Arguments

object	an object inherited from class of createArea.

Value

an object inherited from class of list and including features as list(s), which are accessable as follows:

reservoirs: operation$reservoirs rivers: operation$rivers junctions: operation$junctions aquifers: operation$aquifers diversions: operation$diversions demands: operation$demands

See Also

sim

---

sim.default *default function for class of* sim

Description

sim simulates an object inherited from class of createArea using Standard Operating Policy (SOP).

Usage

```
## Default S3 method:
sim(object)
```

Arguments

object	an object inherited from class of createArea.

Value

an object inherited from class of `sim` and including features as list(s), which are accessable as follows:

reservoirs: $operation$operation$reservoirs rivers: $operation$operation$rivers junctions: $operation$operation$junctions aquifers: $operation$operation$aquifers diversions: $operation$operation$diversions demands: $operation$operation$demands

See Also

`sim`

Index