Package 'WRSS'

March 6, 2026

Type Package

Title Water Resources System Simulator

Version 3.2

Author Rezgar Arabzadeh [aut, cre],
Parisa Aberi [aut],
Kaveh Panaghi [aut],
Shahab Araghinejad [aut],
Majid Montaseri [aut]

Maintainer Rezgar Arabzadeh <rarabzad@uwaterloo.ca>

Description Tools for constructing, simulating, and analyzing large-scale
water resources systems. The package provides functions to represent
system components such as reservoirs, aquifers, rivers, diversions,
and demand sites, and to simulate system behavior under Standard
Operating Policy. It also supports the development and evaluation
of water allocation strategies and hydropower operations within
integrated water resources systems.

License GPL-3

Encoding UTF-8

Depends R (>= 3.5.0)

Imports ggplot2, GGally, network, Hmisc, nloptr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-03-06 05:00:20 UTC

Contents

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 and hydropower water resources systems. The package includes functions and
methods for building, simulation, and visualization of water resources components.

Details

Package:	WRSS
Type:	Package
Version:	3.2
Date:	2026-03-05
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 objects

b) createJunction constructor for junction objects

c) createRiver constructor for reach, river, and channel objects

d) createReservoir constructor for reservoir objects

e) createDiversion constructor for diversion objects

f) createAquifer constructor for aquifer objects

g) createDemandSite constructor for demand site objects

h) set.as WRSS objects connector

i) 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)

g) cap_design computes RRV measures for a range of design parameters

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

d) GOF Goodness of fit function

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. Arabzadeh, R.; Aberi, P.; Hesarkazzazi, S.; Hajibabaei, M.; Rauch, W.; Nikmehr, S.; Sitzenfrei, R. WRSS: An Object-Oriented R Package for Large-Scale Water Resources Operation. Water 2021, 13, 3037. https://doi.org/10.3390/w13213037

See Also

addObjectToArea, plot.sim

Examples

```
###---------- loading data
data(zarrineh)

###---------- Constructing main features of Zerrineh river basin
Area<-createArea(name='Zerrineh',location='Kurdistan',
                 simulation=list(start='1900-01-01',
                                    end='1909-12-01',
                                    interval='month'))

  ###---------- Bukan dam
Q<-zarrineh$bukan$timeSeries[,1]
E<-zarrineh$bukan$timeSeries[,2]
R<-zarrineh$bukan$timeSeries[,3]
D<-zarrineh$bukan$timeSeries[,4]
A<-zarrineh$bukan$timeSeries[,5]
RC<-zarrineh$bukan$ratingCurve
min<-zarrineh$bukan$capacity[1]$min
max<-zarrineh$bukan$capacity[2]$max
bukan<-createReservoir(name='bukan',netEvaporation=E,
                       initialStorage=max,
                       geometry=list(deadStorage=min,
                                        capacity=max,
                                        storageAreaTable=RC),
                       latlon=c(36.422518, 46.528040))
Zerrineh<-createRiver(name='Zerrineh-River',downstream=bukan,
                      discharge=Q,latlon=c(36.195921, 46.421897))
R<-createDemandSite(name='E1',demandTS=R,suppliers=list(bukan),
                    priority=1,latlon=c(36.454432, 46.510824))
D<-createDemandSite(name='U1',demandTS=D,suppliers=list(bukan),
                    priority=2,latlon=c(38.070567, 46.274579))
A<-createDemandSite(name='A1',demandTS=A,suppliers=list(bukan),
                    priority=3,latlon=c(37.035751, 45.999025))
Area<-addObjectToArea(Area,Zerrineh)
Area<-addObjectToArea(Area,bukan)
Area<-addObjectToArea(Area,R)
Area<-addObjectToArea(Area,D)
Area<-addObjectToArea(Area,A)

  ###---------- a junction located in Bukan dam upstream
J<-createJunction(name='J1', downstream=Zerrineh, latlon=c(36.267078, 46.438414))
```

```
Area<-addObjectToArea(Area,J)


   ###---------- Markhuz dam
Q<-zarrineh$Markhuz$timeSeries[,1]
E<-zarrineh$Markhuz$timeSeries[,2]
A<-zarrineh$Markhuz$timeSeries[,3]
RC<-zarrineh$Markhuz$ratingCurve
min<-zarrineh$Markhuz$capacity[1]$min
max<-zarrineh$Markhuz$capacity[2]$max
Markhuz<-createReservoir(name='Markhuz',netEvaporation=E,
                         downstream=J,initialStorage=max,
                         geometry=list(deadStorage=min,
                         capacity=max,
                         storageAreaTable=RC), latlon=c(36.297520, 46.165240))
River<-createRiver(name='Markhuz-River',downstream=Markhuz,
                   discharge=Q,latlon=c(36.287369, 46.131472))
A<-createDemandSite(name='A3',demandTS=A,returnFlowFraction=0.3,
                    suppliers=list(Markhuz),downstream=J,
                    priority=1,latlon=c(36.311133, 46.238820))
Area<-addObjectToArea(Area, River)
Area<-addObjectToArea(Area, Markhuz)
Area<-addObjectToArea(Area, A)


   ###---------- Cheragh Veys dam
Q<-zarrineh$cheraghVeys$timeSeries[,1]
E<-zarrineh$cheraghVeys$timeSeries[,2]
R<-zarrineh$cheraghVeys$timeSeries[,3]
D<-zarrineh$cheraghVeys$timeSeries[,4]
A<-zarrineh$cheraghVeys$timeSeries[,5]
RC<-zarrineh$cheraghVeys$ratingCurve
min<-zarrineh$cheraghVeys$capacity[1]$min
max<-zarrineh$cheraghVeys$capacity[2]$max
cheraghVeys<-createReservoir(name='cheraghVeys',netEvaporation=E,
                             downstream=J,initialStorage=max,
                             geometry=list(deadStorage=min,
                                           capacity=max,
                                           storageAreaTable=RC),
                             latlon=c(36.178180, 46.099005))
River<-createRiver(name='Cheragh Veys-River',downstream=cheraghVeys,
                   discharge=Q,latlon=c(36.130474, 46.039753))
R<-createDemandSite(name='E2',demandTS=R,returnFlowFraction=1.0,
                    suppliers=list(cheraghVeys),downstream=J,
                    priority=1,latlon=c(36.187573, 46.140781))
D<-createDemandSite(name='U2',demandTS=D,returnFlowFraction=0.7,
                    suppliers=list(cheraghVeys),downstream=J,
                    priority=2,latlon=c(36.241318, 46.283620))
A<-createDemandSite(name='A2',demandTS=A,returnFlowFraction=0.3,
                    suppliers=list(cheraghVeys),downstream=J,
                    priority=3,latlon=c(36.209470, 46.244950))
Area<-addObjectToArea(Area, River)
Area<-addObjectToArea(Area, cheraghVeys)
Area<-addObjectToArea(Area, R)
Area<-addObjectToArea(Area, D)
```

```
Area<-addObjectToArea(Area, A)

   ###---------- Sonata dam
Q<-zarrineh$Sonata$timeSeries[,1]
E<-zarrineh$Sonata$timeSeries[,2]
R<-zarrineh$Sonata$timeSeries[,3]
A<-zarrineh$Sonata$timeSeries[,4]
RC<-zarrineh$Sonata$ratingCurve
min<-zarrineh$Sonata$capacity[1]$min
max<-zarrineh$Sonata$capacity[2]$max
Sonata<-createReservoir(name='Sonata',netEvaporation=E,downstream=J,
                        initialStorage=max,
                        geometry=list(deadStorage=min,
                                      capacity=max,
                                      storageAreaTable=RC),
                        latlon=c(36.177088, 46.570563))
River<-createRiver(name='Sonata-River',downstream=Sonata,
                   discharge=Q,latlon=c(36.156056, 46.555296))
R<-createDemandSite(name='E3',demandTS=R,returnFlowFraction=1.0,
                    suppliers=list(Sonata),downstream=J,
                    priority=1,latlon=c(36.189166, 46.580790))
A<-createDemandSite(name='A4',demandTS=A,returnFlowFraction=0.3,
                    suppliers=list(Sonata),downstream=J,
                    priority=2,latlon=c(36.202963, 46.592099))
Area<-addObjectToArea(Area, River)
Area<-addObjectToArea(Area, Sonata)
Area<-addObjectToArea(Area, R)
Area<-addObjectToArea(Area, A)

   ###---------- Sarogh dam
Q<-zarrineh$Sarogh$timeSeries[,1]
E<-zarrineh$Sarogh$timeSeries[,2]
D<-zarrineh$Sarogh$timeSeries[,3]
A<-zarrineh$Sarogh$timeSeries[,4]
RC<-zarrineh$Sarogh$ratingCurve
min<-zarrineh$Sarogh$capacity[1]$min
max<-zarrineh$Sarogh$capacity[2]$max
Sarogh<-createReservoir(name='Sarogh',netEvaporation=E,downstream=J,
                        initialStorage=max,
                        geometry=list(deadStorage=min,
                                      capacity=max,
                                      storageAreaTable=RC),
                        latlon=c(36.515682, 47.110757))
River<-createRiver(name='Sarogh-River',downstream=Sarogh,
                   discharge=Q,latlon=c(36.531021, 47.192899))
D<-createDemandSite(name='U3',demandTS=D,returnFlowFraction=0.7,
                    suppliers=list(Sarogh),downstream=J,
                    priority=1,latlon=c(36.403821, 47.114230))
A<-createDemandSite(name='A5',demandTS=A,returnFlowFraction=0.3,
                    suppliers=list(Sarogh),downstream=J,
                    priority=2,latlon=c(36.414478, 46.946946))
Area<-addObjectToArea(Area, River)
Area<-addObjectToArea(Area, Sarogh)
```

```
Area<-addObjectToArea(Area, D)
Area<-addObjectToArea(Area, A)
## Not run:
plot(Area)

## End(Not run)
plot(sim(Area))
```

---

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:

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",discharge=rnorm(120,5,1.5))
Res<-createReservoir(name="res3",type='storage',
                     priority=1,netEvaporation=rnorm(120,0.5,0.1),
                     geometry=list(deadStorage= 10 ,capacity= 90 ,
                     storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                      100/sum(sin(seq(0,pi,length.out=12))))
D<-createDemandSite(name ="Agri1",
                    demandParams=list(waterUseRate=1,
                                      waterVariation=waterVariation,
                                      cropArea=1000))
R<-set.as(Res,R,'downstream')
D<-set.as(Res,D,'supplier')

area<-createArea(name="unknown",location="unknown",
                 simulation=list(start='2000-01-01',
                                 end  ='2000-04-29',
                                 interval='day'))
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-------------------
Res<-createReservoir(name="res3",type='storage',
                     priority=1,netEvaporation=rnorm(120,0.5,0.1),
                     geometry=list(deadStorage= 10 ,capacity= 90 ,
```

```
                            storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
R<-createRiver(name="river1",discharge=rnorm(120,5,1.5))
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                          100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="Agri1",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=1000),
                     returnFlowFraction =0.2,priority=1)
D2<-createDemandSite(name ="Agri2",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=1000),
                     returnFlowFraction =0.2,priority=1)

R<-set.as(Res,R,'downstream')
D1<-set.as(Res,D1,'supplier')
D2<-set.as(Res,D2,'supplier')

area<-createArea(name="unknown",location="unknown",
                 simulation=list(start='2000-01-01',
                                 end  ='2000-04-29',
                                 interval='day'))
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-------------------
J1<-createJunction(name="j1")
Res1<-createReservoir(name="res1",type='storage',
                      priority=1,netEvaporation=rnorm(120,0.5,0.1),
                      geometry=list(deadStorage= 10 ,capacity= 90 ,
                      storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
Res2<-createReservoir(name="res2",type='storage',
                      priority=2,netEvaporation=rnorm(120,0.5,0.1),
                      geometry=list(deadStorage= 10 ,capacity= 90 ,
                      storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
R1<-createRiver(name="river1",discharge=rnorm(120,5,1.5))
R2<-createRiver(name="river2",discharge=rnorm(120,5,1.5))
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                          100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="Agri1",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=1000),
                     returnFlowFraction =0.2,priority=1)
```

```
D2<-createDemandSite(name ="Agri2",
                      demandParams=list(waterUseRate=1,
                                          waterVariation=waterVariation,
                                          cropArea=1000),
                      returnFlowFraction =0.2,priority=2)
D3<-createDemandSite(name ="Agri3",
                      demandParams=list(waterUseRate=1,
                                          waterVariation=waterVariation,
                                          cropArea=1000),
                      returnFlowFraction =0.2,priority=1)
area<-createArea(name="unknown",location="unknown",
                simulation=list(start='2000-01-01',
                                 end  ='2000-04-29',
                                 interval='day'))

R1<-set.as(Res1,R1,'downstream')
R2<-set.as(Res2,R2,'downstream')
Res1<-set.as(J1,Res1,'downstream')
Res2<-set.as(J1,Res2,'downstream')
D1<-set.as(J1,D1,'downstream')
D2<-set.as(J1,D2,'downstream')
D3<-set.as(J1,D3,'downstream')
D1<-set.as(Res1,D1,'supplier')
D2<-set.as(Res1,D2,'supplier')
D2<-set.as(Res2,D2,'supplier')
D3<-set.as(Res2,D3,'supplier')

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-------------------
J1<-createJunction(name="j1")
Res1<-createReservoir(name="res1",type='storage',
                       priority=1,netEvaporation=rnorm(120,0.5,0.1),downstream =J1 ,
                       geometry=list(deadStorage= 10 ,capacity= 90 ,
                       storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
Auq1<-createAquifer(name="Aquifer1",area=100,volume=5000,
                     rechargeTS=rnorm(120,10,3),Sy=0.1,
                     leakageFraction=0.02,leakageObject=J1,priority=2)
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                       100/sum(sin(seq(0,pi,length.out=12))))
```

```
R1<-createRiver(name="river1",downstream=Res1,discharge=rnorm(120,5,1.5))
R2<-createRiver(name="river2",downstream=Auq1,discharge=rnorm(120,5,1.5))
D1<-createDemandSite(name ="Agri1",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=1000),
                     returnFlowFraction =0.2,suppliers=list(Res1,Auq1),
                     downstream=J1,priority=1)
D2<-createDemandSite(name ="Agri2",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=1000),
                     returnFlowFraction =0.2,suppliers=list(Res1,Auq1),
                     downstream=J1,priority=2)
D3<-createDemandSite(name ="Agri3",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=1000),
                     returnFlowFraction =0.2,suppliers=list(Res1,Auq1),
                     downstream=J1,priority=1)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start='2000-01-01',
                                 end  ='2000-04-29',
                                 interval='day'))
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-------------------
J1<-createJunction(name="junction1")
Res1<-createReservoir(name="res1",type='storage',
                      priority=1,netEvaporation=rnorm(120,0.5,0.1),
                      geometry=list(deadStorage= 10 ,capacity= 90 ,
                      storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
Auq1<-createAquifer(name="Aquifer1",area=100,volume=5000,
                    rechargeTS=rnorm(120,10,3),Sy=0.1,priority=2)
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                      100/sum(sin(seq(0,pi,length.out=12))))
R1<-createRiver(name="River1",
                downstream=Res1,discharge=rnorm(120,20,3),
                seepageFraction=0.1,seepageObject=Auq1)
D1<-createDemandSite(name ="Agri1",
```

```
                              demandParams=list(waterUseRate=1,
                                                waterVariation=waterVariation,
                                                cropArea=1000),
                          returnFlowFraction =0.2,suppliers=list(Res1),
                          downstream=J1,priority=1)
D2<-createDemandSite(name ="Agri2",
                          demandParams=list(waterUseRate=1,
                                                waterVariation=waterVariation,
                                                cropArea=1000),
                          returnFlowFraction =0.2,suppliers=list(Res1,Auq1),
                          downstream=J1,priority=2)
D3<-createDemandSite(name ="Agri3",
                          demandParams=list(waterUseRate=1,
                                                waterVariation=waterVariation,
                                                cropArea=1000),
                          returnFlowFraction =0.2,suppliers=list(R1),
                          downstream=Res1,priority=2)
D4<-createDemandSite(name ="Agri4",
                          demandParams=list(waterUseRate=1,
                                                waterVariation=waterVariation,
                                                cropArea=1000),
                          returnFlowFraction =0.2,suppliers=list(R1),
                          downstream=Res1,priority=1)
area<-createArea(name="unknown",location="unknown",
                    simulation=list(start='2000-01-01',
                                      end  ='2000-04-29',
                                      interval='day'))
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-------------------
Auq1<-createAquifer(name="Aquifer1",area=100,volume=5000,
                        rechargeTS=rnorm(120,10,3),Sy=0.1)
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                        100/sum(sin(seq(0,pi,length.out=12))))
D0<-createDemandSite(name ="Agri0",
                          demandParams=list(waterUseRate=1,
                                                waterVariation=waterVariation,
                                                cropArea=1000),priority=1)
Div1<-createDiversion(name="Div1",capacity=10)
```

```
J2<-createJunction(name="junc2")

Res2<-createReservoir(name="res2",type='storage',
                       priority=1,netEvaporation=rnorm(120,0.5,0.1),
                       geometry=list(deadStorage= 10 ,capacity= 90 ,
                       storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
R2<-createRiver(name="river2",discharge=rnorm(120,12,3))
D3<-createDemandSite(name ="Agri3",
                     demandParams=list(waterUseRate=1,
                                        waterVariation=waterVariation,
                                        cropArea=1000),
                     returnFlowFraction =0.2,priority=2)
J1<-createJunction(name="junc1")
Res1<-createReservoir(name="res1",type='storage',
                       priority=1,netEvaporation=rnorm(120,0.5,0.1),
                       geometry=list(deadStorage= 10 ,capacity= 90 ,
                       storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
R1<-createRiver(name="river1",discharge=rnorm(120,5,1))
D2<-createDemandSite(name ="Agri2",
                     demandParams=list(waterUseRate=1,
                                        waterVariation=waterVariation,
                                        cropArea=1000),
                     returnFlowFraction =0.2,priority=2)
D1<-createDemandSite(name ="Agri1",
                     demandParams=list(waterUseRate=1,
                                        waterVariation=waterVariation,
                                        cropArea=1000),
                     returnFlowFraction =0.2,priority=1)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start='2000-01-01',
                                  end  ='2000-04-29',
                                  interval='day'))

R1<-set.as(Res1,R1,'downstream')
R2<-set.as(Res2,R2,'downstream')
Res1<-set.as(J1,Res1,'downstream')
Res2<-set.as(J2,Res2,'downstream')
J1<-set.as(Div1,J1,'downstream')
J2<-set.as(Auq1,J2,'downstream')
Div1<-set.as(Auq1,Div1,'divertObject')
D1<-set.as(J1,D1,'downstream')
D2<-set.as(J1,D2,'downstream')
D3<-set.as(J2,D3,'downstream')
D1<-set.as(Res1,D1,'supplier')
D2<-set.as(Res1,D2,'supplier')
D2<-set.as(Res2,D2,'supplier')
D3<-set.as(Res2,D3,'supplier')
D0<-set.as(Auq1,D0,'supplier')



area<-addObjectToArea(area,R1)
area<-addObjectToArea(area,R2)
```

```
area<-addObjectToArea(area,Res1)
area<-addObjectToArea(area,Res2)
area<-addObjectToArea(area,D0)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
area<-addObjectToArea(area,D3)
area<-addObjectToArea(area,Div1)
area<-addObjectToArea(area,Auq1)
area<-addObjectToArea(area,J1)
area<-addObjectToArea(area,J2)

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, volume,
               rechargeTS = NA, leakageFraction = NA,
               initialStorage = NA, Sy, simulation)
```

Arguments

demand	(optional) A matrix: is column-wise matrix of demands, at which the rows present demands for each monthly time step and columns are for different individual demand sites (MCM).
priority	(optional) A vector: is a vector of priorities associated to demand
area	The area of aquifer (Km^2)
volume	The aquifer volume (MCM)
rechargeTS	(optional) A vector : a vector of water flowing into the aquifer (MCM)
leakageFraction	
	(optional) The leakage coeffcient of aquifer storage. The leakage is computed as the product of leakageFraction and aquifer storage. It is in [0, 1] interval
initialStorage	(optional) 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 of three vectors: start, end, and interval. the start and end components must be in 'YYYY-MM-DD' format. the interval component can takes either of 'day','week', or 'month'.

Value

the aquiferRouting function returns a list of objects as bellow:

- 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
volume          <-20000
priority        <-c(3,1,1,2)
rechargeTS      <-rnorm(120,60,8)
demand          <-matrix(rnorm(480,10,3),120)
simulation      <-list(start='2000-01-01',end='2009-12-29',interval='month')

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

plot(res$storage$storage,ylab='Storage (MCM)',xlab='time steps(month)',type='o')
```

cap_design *Constructor for class of* cap_design

Description

Calculates the RRV measures for multiple design candidates.

Usage

```
cap_design(area,params,w,plot)
```

Arguments

area	An object from class of `'createArea'`
params	A list of list(s), which each sub-list can contains an object from either of classes `'createDemandSite'` or `'createReservoir'` and a vector of scale factors multiplied to the set design parameters. For reservoirs the scale factor will be multiplied to the capacity for the and for demand site, it will be multiplied to the demand time series
w	(optional) A vector of weights of sustainability indices summing 1 with length of equal with the number of demand site objects built-in `'params'` argument or equal with number of demand sites supplied by the reservoirs built-in `'params'`. If missing the weights will be assumed equall
plot	(optional) logical: plot the resault or not. The default is TRUE

Value

A matrix of RRV and sustainability index proposed by Hashemitto et al. (1982) and Loucks (1997).

Author(s)

Rezgar Arabzadeh

References

Hashimoto, T., Stedinger, J. R., & Loucks, D. P. (1982). Reliability, resiliency, and vulnerability criteria for water resource system performance evaluation. Water resources research, 18(1), 14-20. Loucks, D. P. (1997). Quantifying trends in system sustainability. Hydrological Sciences Journal, 42(4), 513-530.

See Also

[addObjectToArea](#)

Examples

```
Res1<-createReservoir(name="res1",type='storage',
                      priority=1,netEvaporation=rnorm(120,0.5,0.1),
                      geometry=list(deadStorage= 10 ,capacity= 50 ,
                                    storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
R1<-createRiver(name="river1",discharge=rnorm(120,25,1.5))
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                      100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="Agri1",
                     demandParams=list(waterUseRate=1,
                                       waterVariation=waterVariation,
                                       cropArea=500),
                     returnFlowFraction =0.2,priority=2)
area<-createArea(name="unknown",location="unknown",
                 simulation=list(start='2000-01-01',
                                 end  ='2000-04-29',
                                 interval='day'))
R1<-set.as(Res1,R1,'downstream')
D1<-set.as(Res1,D1,'supplier')
area<-addObjectToArea(area,R1)
area<-addObjectToArea(area,Res1)
area<-addObjectToArea(area,D1)
params<-list(
  list(Res1,seq(0.5,1.5,0.1))
)
cap_design(area,params)
```

---

cap_design.base   *base function for class of* cap_design

Description

Calculates the RRV measures for multiple design candidates.

Usage

```
## S3 method for class 'base'
cap_design(area,params,w,plot)
```

Arguments

area   An object from class of 'createArea'

params   A list of list(s), which each sub-list can contains an object from either of classes
                'createDemandSite' or 'createReservoir' and a vector of scale factors mul-
                tiplied to the set design parameters. For reservoirs the scale factor will be mul-
                tiplied to the capacity for the and for demand site, it will be multiplied to the
                demand time series

w	(optional) A vector of weights of sustainability indices summing 1 with length of equal with the number of demand site objects built-in `'params'` argument or equal with number of demand sites supplied by the reservoirs built-in `'params'`. If missing the weights will be assumed equall
plot	(optional) logical: plot the resault or not. The default is TRUE

Value

A matrix of RRV and sustainability index proposed by Hashemitto et al. (1982) and Loucks (1997).

Author(s)

Rezgar Arabzadeh

References

Hashimoto, T., Stedinger, J. R., & Loucks, D. P. (1982). Reliability, resiliency, and vulnerability criteria for water resource system performance evaluation. Water resources research, 18(1), 14-20. Loucks, D. P. (1997). Quantifying trends in system sustainability. Hydrological Sciences Journal, 42(4), 513-530.

See Also

cap_design

---

cap_design.default *default function for class of* cap_design

---

Description

Calculates the RRV measures for multiple design candidates.

Usage

```
## Default S3 method:
cap_design(area,params,w=NA,plot=TRUE)
```

Arguments

area	An object from class of `'createArea'`
params	A list of list(s), which each sub-list can contains an object from either of classes `'createDemandSite'` or `'createReservoir'` and a vector of scale factors multiplied to the set design parameters. For reservoirs the scale factor will be multiplied to the capacity for the and for demand site, it will be multiplied to the demand time series

w	(optional) A vector of weights of sustainability indices summing 1 with length of equal with the number of demand site objects built-in `'params'` argument or equal with number of demand sites supplied by the reservoirs built-in `'params'`. If missing the weights will be assumed equall
plot	(optional) logical: plot the resault or not. The default is `TRUE`

Value

A matrix of RRV and sustainability index proposed by Hashemitto et al. (1982) and Loucks (1997).

Author(s)

Rezgar Arabzadeh

References

Hashimoto, T., Stedinger, J. R., & Loucks, D. P. (1982). Reliability, resiliency, and vulnerability criteria for water resource system performance evaluation. Water resources research, 18(1), 14-20. Loucks, D. P. (1997). Quantifying trends in system sustainability. Hydrological Sciences Journal, 42(4), 513-530.

See Also

cap_design

---

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, volume,
              rechargeTS, Sy, leakageFraction,
              initialStorage, leakageObject, priority, latlon)
```

Arguments

name	(optional) A string: the name of the aquifer
area	The area of aquifer (Km^2)
volume	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. It is in `[0, 1]` interval

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.

leakageObject (optional) an object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; 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.

latlon (optional) A vector of two elements: `lat`, `lon` in decimal degree representing the location of the object.

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, volume,
                 rechargeTS, Sy,leakageFraction,
                 initialStorage, leakageObject, priority, latlon)
```

Arguments

name	(optional) A string: the name of the aquifer
area	The area of aquifer (Km^2)
volume	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. It is in [0, 1] interval
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.
leakageObject	(optional) an object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); 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.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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            ,
              volume          ,
              rechargeTS      = NA ,
              Sy              = 0.1,
              leakageFraction = NA ,
```

```
                       initialStorage  = NA ,
                       leakageObject   = NA ,
                       priority        = NA,
    latlon             = NULL)
```

Arguments

name	(optional) A string: the name of the aquifer
area	The area of aquifer (Km^2)
volume	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. It is in [0, 1] interval
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.
leakageObject	(optional) an object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); 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.
latlon	(optional) A vector of two elements: `lat`, `lon` in decimal degree representing the location of the object.

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 following 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 of three vectors: start, end, and interval. the start and end components must be in 'YYYY-MM-DD' format. The interval component can takes either of 'day','week', or '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 of three vectors: start, end, and interval. the start and end components must be in 'YYYY-MM-DD' format and the interval component is a string that can takes either of 'day','week', or '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, interval=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 of three vectors: start, end, and interval. the start and end components must be in 'YYYY-MM-DD' format and the interval component can takes either of 'day','week', or '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, demandTS, demandParams,
                 returnFlowFraction, suppliers,
                 downstream, priority,latlon)
```

Arguments

name	(optional) A string: the name of the demand site
demandTS	A vector: a vector of demand time series (MCM). If demandParams is null, providing the demandTS is compulsory.
demandParams	A list: If demandTS is missing, the demandParams must be provided to establish demandTS. The demandParams includes three parts as follows:

- waterUseRate: The total water demand per hectare (MCM) per a given water cycle.
- waterVariation: A vector of the precentages for water demand distribution within a water cycle (the precentages in each interval). For instance, if the cycle is annually and the interval is 'month'ly, the waterVariation could be a vector of length of 12, for which its indices signify the monthly portion of water demand, in precentage, by the total water demand required for the whole cycle.
- cropArea: the area of cropping farms (in 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	(optional) A list of object(s) inherited from the following classes: createAquifer, createRiver, createReservoir, createDiversion.
downstream	(optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which return flow volume pours to it.
priority	(optional) An integer: the priority to be supplied. A value in [1, 99] interval.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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, demandTS, demandParams,
                 returnFlowFraction, suppliers,
                 downstream, priority,latlon)
```

Arguments

name   (optional) A string: the name of the demand site

demandTS   A vector: a vector of demand time series (MCM). If demandParams is null, providing the demandTS is compulsory.

demandParams   A list: If demandTS is missing, the demandParams must be provided to establish demandTS. The demandParams includes three parts as follows:

- waterUseRate: The total water demand per hectare (MCM) per a given water cycle
- waterVariation: A vector of the precentages for water demand distribution within a water cycle (the precentages in each interval). For instance, if the cycle is annually and the interval is 'month'ly, the waterVariation could be a vector of length of 12, for which its indices signify the monthly portion of water demand, in precentage, by the total water demand required for the whole cycle
- cropArea: the area of cropping farms (in 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   (optional) A list of object(s) inherited from the folowing classes: createAquifer, createRiver, createReservoir, createDiversion.

downstream   (optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which return flow volume pours to it.

priority   (optional) An integer: the priority to be supplied. A value in [1, 99] interval.

latlon   (optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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",
        demandTS          =NA                      ,
        demandParams=list(waterUseRate=NULL        ,
                          waterVariation=NULL      ,
                          cropArea=NULL)           ,
        returnFlowFraction =0.0                    ,
        suppliers          = NA                    ,
        downstream          =NA                    ,
        priority            =NA                    ,
  latlon            =NULL)
```

Arguments

name	(optional) A string: the name of the demand site
demandTS	A vector: a vector of demand time series (MCM). If demandParams is null, providing the demandTS is compulsory.
demandParams	A list: If demandTS is missing, the demandParams must be provided to establish demandTS. The demandParams includes three parts as follows:

- waterUseRate: The total water demand per hectare (MCM) per a given water cycle.
- waterVariation: A vector of the precentages for water demand distribution within a water cycle (the precentages in each `interval`). For instance, if the cycle is annually and the interval is `'month'`ly, the waterVariation could be a vector of length of 12, for which its indices signify the monthly portion of water demand, in precentage, by the total water demand required for the whole cycle.

- cropArea: the area of cropping farms (in 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   (optional) A list of object(s) inherited from the folowing classes: createAquifer, createRiver, createReservoir, createDiversion.

downstream   (optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which return flow volume pours to it.

priority   (optional) An integer: the priority to be supplied. A value in [1, 99] interval.

latlon   (optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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, capacity,
                divertObject, downstream, priority,latlon)
```

Arguments

name   (optional) A string: the name of the diversion

capacity   The maximum capacity of diversion dam (CMS).

divertObject   (optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which recieves the diverted water volume.

downstream   (optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; 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. |
| latlon | (optional) A vector of two elements: `lat`, `lon` in decimal degree representing the location of the object. |

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, capacity,
                divertObject, downstream, priority, latlon)
```

Arguments

name	(optional) A string: the name of the diversion
capacity	The maximum capacity of diversion dam (CMS).
divertObject	(optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which recieves the diverted water volume.
downstream	(optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; 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.
latlon	(optional) A vector of two elements: `lat`, `lon` in decimal degree representing the location of the object.

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 = "Div1",
                  capacity          ,
                  divertObject = NA,
                  downstream   = NA,
                  priority     = NA,
latlon       = NULL)
```

Arguments

name	(optional) A string: the name of the diversion
capacity	The maximum capacity of diversion dam (CMS).
divertObject	(optional) An object from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which recieves the diverted water volume.
downstream	(optional) An object from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); 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`.
latlon	(optional) A vector of two elements: `lat, lon` in decimal degree representing the location of the object.

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, downstream, latlon)
```

Arguments

name	(optional) A string: the name of the junction
downstream	(optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which outflow volume pours to it.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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, downstream, latlon)
```

Arguments

name	(optional) A string: the name of the junction
downstream	(optional) An object from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which outflow volume pours to it.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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", downstream = NA, latlon =NULL)
```

Arguments

name	(optional) A string: the name of the junction
downstream	(optional) An object from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which outflow volume pours to it.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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(type,
                name,
                priority,
                downstream,
                netEvaporation,
                seepageFraction,
                seepageObject,
                geometry,
                plant,
                penstock,
                initialStorage,
latlon)
```

Arguments

type	A string: the type of the reservoir being instantiated: by default 'storage', however, it can be 'hydropower'
name	(optional) A string: the name of the reservoir.
priority	(optional) An integer: the supplying priority. priority is a value in [1, 99] interval. If missing, the priority is set to Inf.

downstream (optional) An object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which spillage volume pours to it.

netEvaporation A vector: is a vector of net evaporation depth time series at the location of dam site (meter). If omitted, the evaporation is assumed to be zero.

seepageFraction

(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of seepageFraction and reservoir storage. It is in [0, 1] interval

seepageObject (optional) An object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which seepage volume pours to it.

geometry A list of reservoir geometric specifications:

- storageAreaTable: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (in Km^2) corresponding to the first column
- storageElevationTable: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains elevation (in meter) corresponding to the first column
- dischargeElevationTable: is a matrix whose first column includes the capacity of reservoir tailwater discharge rate (in cms) for different elevation levels and the second column contains elevation levels corresponding to the first column, required if the type = 'hydropower' and the item submerged = TRUE
- 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

plant A list of power plant specifications. It is provided if type = 'hydropower':

- installedCapacity: the plant installed capacity (MW)
- efficiency: is a matrix whose first column includes discharge rate (in cms) and the second column turbine effeciency, in [0 1] interval, corresponding to the first column
- designHead: A vector of length of two, containing the minimum and maximum design water head (in meter) of the turbine respecively, that the it is in active state
- designFlow: A vector of length of two, containing the minimum and maximum design flow rate (in cms) of the turbine respecively, that the it is in active state
- turbineAxisElevation: The elevation of axis of the installed turbine (in meter)
- submerged: logical: if the turbine is of type of submeged on, TRUE, otherwise, FALSE
- loss: losses associated with the turbine (in meter)

penstock (optional) A list of penstock specifications. It is provided if type = 'hydropower':

- diameter: The diameter of the penstock (in meter)

- length: The length of the penstock (in meter)
- roughness: pipe roughness coefficient used for Hazen-Williams formulation

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 reservoir.

latlon   (optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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(type,
                              name,
                              priority,
                              downstream,
                              netEvaporation,
                              seepageFraction,
                              seepageObject,
                              geometry,
                              plant,
                              penstock,
                              initialStorage,
   latlon)
```

Arguments

`type`	A string: the type of the reservoir being instantiated: by default 'storage', however, it can be 'hydropower'
`name`	(optional) A string: the name of the reservoir.
`priority`	(optional) An integer: the supplying priority. `priority` is a value in [1, 99] interval. If missing, the `priority` is set to Inf.
`downstream`	(optional) An object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which spillage volume pours to it.
`netEvaporation`	A vector: is a vector of net evaporation depth time series at the location of dam site (meter). If omitted, the evaporation is assumed to be zero.
`seepageFraction`	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of `seepageFraction` and reservoir storage. It is in [0, 1] interval
`seepageObject`	(optional) An object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; which seepage volume pours to it.
`geometry`	A list of reservoir geometric specifications:

- `storageAreaTable`: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (in Km^2) corresponding to the first column
- `storageElevationTable`: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains elevation (in meter) corresponding to the first column
- `dischargeElevationTable`: is a matrix whose first column includes the capacity of reservoir tailwater discharge rate (in cms) for different elevation levels and the second column contains elevation levels corresponding to the first column, required if the `type = 'hydropower'` and the item `submerged = TRUE`
- `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

`plant`	A list of power plant specifications. It is provided if `type = 'hydropower'`:

- `installedCapacity`: the plant installed capacity (MW)
- `efficiency`: is a matrix whose first column includes discharge rate (in cms) and the second column turbine effeciency, in [0 1] interval, corresponding to the first column
- `designHead`: A vector of length of two, containing the minimum and maximum design water head (in meter) of the turbine respecively, that the it is in active state
- `designFlow`: A vector of length of two, containing the minimum and maximum design flow rate (in cms) of the turbine respecively, that the it is in active state

- turbineAxisElevation: The elevation of axis of the installed turbine (in meter)
- submerged: logical: if the turbine is of type of submeged on, TRUE, otherwise, FALSE
- loss: losses associated with the turbine (in meter)

penstock   (optional) A list of penstock specifications. It is provided if type = 'hydropower':

- diameter: The diameter of the penstock (in meter)
- length: The length of the penstock (in meter)
- roughness: pipe roughness coefficient used for Hazen-Williams formulation

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 reservoir.

latlon   (optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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(type='storage',
                        name='unknown',
                        priority=NA,
                        downstream=NA,
                        netEvaporation=NA,
                        seepageFraction=NA,
                        seepageObject=NA,
                        geometry=list(storageAreaTable=NULL,
                                      storageElevationTable=NULL,
                                      dischargeElevationTable=NULL,
```

```
                                deadStorage=NULL,
                                capacity=NULL),
                    plant=list(installedCapacity=NULL,
                                efficiency=NULL,
                                designHead=NULL,
                                designFlow=NULL,
                                turbineAxisElevation=NULL,
                                submerged=FALSE,
                                loss=0),
                    penstock=list(diameter=NULL,
                                length=NULL,
                                roughness=110),
                    initialStorage=NA,
    latlon=NULL)
```

Arguments

type	A string: the type of the reservoir being instantiated: by default 'storage', however, it can be 'hydropower'
name	(optional) A string: the name of the reservoir.
priority	(optional) An integer: the supplying priority. `priority` is a value in [1, 99] interval. If missing, the `priority` is set to Inf.
downstream	(optional) An object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which spillage volume pours to it.
netEvaporation	A vector: is a vector of net evaporation depth time series at the location of dam site (meter). If omitted, the evaporation is assumed to be zero.
seepageFraction	
	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of `seepageFraction` and reservoir storage. It is in [0, 1] interval
seepageObject	(optional) An object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which seepage volume pours to it.
geometry	A list of reservoir geometric specifications:

- storageAreaTable: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (in Km^2) corresponding to the first column
- storageElevationTable: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains elevation (in meter) corresponding to the first column
- dischargeElevationTable: is a matrix whose first column includes the capacity of reservoir tailwater discharge rate (in cms) for different elevation levels and the second column contains elevation levels corresponding to the first column, required if the type = 'hydropower' and the item submerged = TRUE

- 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

plant   A list of power plant specifications. It is provided if type = 'hydropower':

- installedCapacity: the plant installed capacity (MW)
- efficiency: is a matrix whose first column includes discharge rate (in cms) and the second column turbine effeciency, in [0 1] interval, corresponding to the first column
- designHead: A vector of length of two, containing the minimum and maximum design water head (in meter) of the turbine respecively, that the it is in active state
- designFlow: A vector of length of two, containing the minimum and maximum design flow rate (in cms) of the turbine respecively, that the it is in active state
- turbineAxisElevation: The elevation of axis of the installed turbine (in meter)
- submerged: logical: if the turbine is of type of submeged on, TRUE, otherwise, FALSE
- loss: losses associated with the turbine (in meter)

penstock   (optional) A list of penstock specifications. It is provided if type = 'hydropower'

- diameter: The diameter of the penstock (in meter)
- length: The length of the penstock (in meter)
- roughness: pipe roughness coefficient used for Hazen-Williams formulation

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 reservoir.

latlon   (optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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, downstream, seepageFraction,
            seepageObject, discharge, priority, latlon)
```

Arguments

name	(optional) A string: the name of the river
downstream	(optional) An object; from either of classes of [createAquifer](), [createRiver](), [createReservoir](), [createJunction](), [createDiversion]() , or [createDemandSite](); 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. It is in [0, 1] interval
seepageObject	(optional) An object; from either of classes of [createAquifer](), [createRiver](), [createReservoir](), [createJunction](), [createDiversion]() , or [createDemandSite](); 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.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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, downstream, seepageFraction,
              seepageObject, discharge, priority, latlon)
```

Arguments

name	(optional) A string: the name of the river
downstream	(optional) An object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; 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. It is in [0, 1] interval
seepageObject	(optional) An object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite; 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.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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",
                    downstream      = NA,
                    seepageFraction = NA,
                    seepageObject   = NA,
                    discharge       = NA,
                    priority        = NA,
latlon          = NULL)
```

Arguments

name	(optional) A string: the name of the river
downstream	(optional) An object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); 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. It is in [0, 1] interval
seepageObject	(optional) An object; from either of classes of [createAquifer](#) , [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); 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.
latlon	(optional) A vector of two elements: lat, lon in decimal degree representing the location of the object.

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

Arguments

demand	A matrix: is column-wise matrix of demands, at which the rows presents demands for each 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 of three vectors: start, end, and interval. the start and end components must be in 'YYYY-MM-DD' format. the interval component can takes either of 'day', 'week', or 'month'.

Value

the diversionRouting function returns a list of features given as below:

- 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 through 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='2000-01-01',end='2009-12-29',interval='month')
res<-diversionRouting(demand=demand,
                      priority=priority,
                      capacity=capacity,
                      inflow=inflow,
                      simulation=simulation)
plot(ecdf(res$diverted$diverted),xlab='cms',ylab='exceedance probability')
```

GOF *Goodness of fit*

Description

this function calculates the goodness of fit (gof) using chi-squared test.

Usage

```
GOF(basin,object,observed)
```

Arguments

basin	An object from class of sim.
object	An object from either of classes of [createAquifer](), [createRiver](), [createReservoir](), [createJunction](), [createDiversion](), or [createDemandSite](); which is associated with observed time series and exists in the basin.
observed	A vector of observed time series.

Value

A list with class "htest".

Author(s)

Rezgar Arabzadeh

See Also

[sim]()

Examples

```
J1<-createJunction(name="j1")
Res1<-createReservoir(name="res1",type='storage',
                      priority=1,netEvaporation=rnorm(120,0.5,0.1),
                      geometry=list(deadStorage= 10 ,capacity= 90 ,
                                    storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
Res2<-createReservoir(name="res2",type='storage',
                      priority=2,netEvaporation=rnorm(120,0.5,0.1),
                      geometry=list(deadStorage= 10 ,capacity= 90 ,
                                    storageAreaTable= cbind(seq(0,90,10),seq(0,9,1))))
R1<-createRiver(name="river1",discharge=rnorm(120,5,1.5))
R2<-createRiver(name="river2",discharge=rnorm(120,5,1.5))
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                      100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="Agri1",
                     demandParams=list(waterUseRate=1,
```

```
                                              waterVariation=waterVariation,
                                              cropArea=1000),
                          returnFlowFraction =0.2,priority=1)
    D2<-createDemandSite(name ="Agri2",
                          demandParams=list(waterUseRate=1,
                                              waterVariation=waterVariation,
                                              cropArea=1000),
                          returnFlowFraction =0.2,priority=2)
    D3<-createDemandSite(name ="Agri3",
                          demandParams=list(waterUseRate=1,
                                              waterVariation=waterVariation,
                                              cropArea=1000),
                          returnFlowFraction =0.2,priority=1)
    area<-createArea(name="unknown",location="unknown",
                    simulation=list(start='2000-01-01',
                                      end  ='2000-04-29',
                                      interval='day'))

    R1<-set.as(Res1,R1,'downstream')
    R2<-set.as(Res2,R2,'downstream')
    Res1<-set.as(J1,Res1,'downstream')
    Res2<-set.as(J1,Res2,'downstream')
    D1<-set.as(J1,D1,'downstream')
    D2<-set.as(J1,D2,'downstream')
    D3<-set.as(J1,D3,'downstream')
    D1<-set.as(Res1,D1,'supplier')
    D2<-set.as(Res1,D2,'supplier')
    D2<-set.as(Res2,D2,'supplier')
    D3<-set.as(Res2,D3,'supplier')

    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)

  ## End(Not run)
  simulated<-sim(area)
  observed<-apply(simulated$operation$operation$junctions[[1]]$operation$outflow,1,sum)
  observed<-observed+rnorm(length(observed),mean(observed)*0.2,sd(observed)*0.1)
  GOF(simulated,J1,observed)
```

GOF.base   *base function for class of* GOF

Description

this function calculates the goodness of fit (gof) using chi-squared test.

Usage

```
## S3 method for class 'base'
GOF(basin,object,observed)
```

Arguments

basin	An object from class of sim.
object	An object from either of classes of [createAquifer](#), [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which is associated with observed time series and exists in the basin.
observed	A vector of observed time series.

Value

A list with class "htest".

Author(s)

Rezgar Arabzadeh

See Also

[GOF](#)

GOF.default   *default function for class of* GOF

Description

this function calculates the goodness of fit (gof) using chi-squared test.

Usage

```
## Default S3 method:
GOF(basin,object,observed)
```

Arguments

basin	An object from class of sim.
object	An object from either of classes of [createAquifer](#), [createRiver](#), [createReservoir](#), [createJunction](#), [createDiversion](#), or [createDemandSite](#); which is associated with observed time series and exists in the basin.
observed	A vector of observed time series.

Value

A list with class "htest".

Author(s)

Rezgar Arabzadeh

See Also

[GOF](#)

---

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 simulates the effect of a dam under givn hydrometeorological time series, e.g. `inflow` and `netEvaporation`, on the drainage network.

Usage

```
reservoirRouting(type='storage',
                 inflow,
                 netEvaporation=NA,
                 demand=NA,
                 priority=NA,
                 seepageFraction=NA,
                 geometry=list(storageAreaTable=NULL,
                               storageElevationTable=NULL,
                               dischargeElevationTable=NULL,
                               deadStorage=0,
```

```
                                capacity=NULL),
                plant=list(installedCapacity=NULL,
                           efficiency=NULL,
                           designHead=NULL,
                           designFlow=NULL,
                           turbineAxisElevation=NULL,
                           submerged=FALSE,
                           loss=0),
                penstock=list(diameter=NULL,
                              length=0,
                              roughness=110),
                initialStorage=NA,
                simulation)
```

Arguments

type	A string: the type of the reservoir being instantiated: by default 'storage', however, it can be 'hydropower'
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). If omitted, the evaporation is assumed to be zero.
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	(optional) A vector: is a vector of priorities associated to demand
seepageFraction	(optional) The seepage coeffcient of reservoir storage. The seepage is computed as the product of seepageFraction and reservoir storage.
geometry	A list of reservoir geometric specifications:

- storageAreaTable: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains reservoir area (in Km^2) corresponding to the first column
- storageElevationTable: is a matrix whose first column includes reservoir volume (MCM) for different elevation levels and the second column contains elevation (in meter) corresponding to the first column
- dischargeElevationTable: is a matrix whose first column includes the capacity of reservoir tailwater discharge rate (in cms) for different elevation levels and the second column contains elevation levels corresponding to the first column, required if the type = 'hydropower' and the item submerged = TRUE
- 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

plant	A list of power plant specifications. It is provided if type = 'hydropower':

- efficiency: is a matrix whose first column includes discharge rate (in cms) and the second column turbine effeciency, in [0 1] interval, corresponding to the first column

- designHead: A vector of length of two, containing the minimum and maximum design water head (in meter) of the turbine respecively, that the it is in active state
- designFlow: A vector of length of two, containing the minimum and maximum design flow rate (in cms) of the turbine respecively, that the it is in active state
- turbineAxisElevation: The elevation of axis of the installed turbine (in meter)
- submerged: logical: if the turbine is of type of submeged on, TRUE, otherwise, FALSE
- loss: losses associated with the turbine (in meter)

penstock   (optional) A list of penstock specifications. It is provided if type = 'hydropower'

- diameter: The diameter of the penstock (in meter)
- length: The length of the penstock (in meter)
- roughness: pipe roughness coefficient used for Hazen-Williams formulation

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 reservoir.

simulation   A list: simulation is a list of three vectors: start, end, and interval. the start and end components must be in 'YYYY-MM-DD' format. the interval component can takes either of 'day','week', or '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

```
type          <-c('storage','hydropower')
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='2000-01-01',end='2009-12-29',interval='month')
seepageFraction<-0.05
geometry      <-list(storageAreaTable=cbind(seq(0,100,10),seq(0,10,1)),
                     storageElevationTable=cbind(seq(0,100,10),seq(0,200,20)),
                     dischargeElevationTable=cbind(seq(0,50,10),seq(0,10,2)),
                     deadStorage=50,
                     capacity=100)
plant         <-list(installedCapacity=50,
                     efficiency=cbind(c(5,25,45),c(0.5,0.9,0.7)),
                     designHead=c(100,200),
                     designFlow=c(10,40),
                     turbineAxisElevation=5,
                     submerged=TRUE,
                     loss=2)
penstock      <-list(diameter=2,
                     length=50,
                     roughness=110)

#-----Storage Reservoir----------
reservoirRouting(type=type[1],
                 inflow=inflow,
                 netEvaporation=netEvaporation,
                 demand=demand,
                 priority=priority,
                 seepageFraction=seepageFraction,
                 geometry=geometry,
                 plant=plant,
                 penstock=penstock,
                 simulation=simulation)
## Not run:
##-----Takes Several Minutes----------

#-----Hydropower Reservoir with demand----------
reservoirRouting(type=type[2],
                 inflow=inflow,
                 netEvaporation=netEvaporation,
                 demand=demand,
                 priority=priority,
                 seepageFraction=seepageFraction,
                 geometry=geometry,
                 plant=plant,
                 penstock=penstock,
                 simulation=simulation)
#-----Hydropower Reservoir----------
reservoirRouting(type=type[2],
                 inflow=inflow,
```

```
                        netEvaporation=netEvaporation,
                        priority=priority,
                        seepageFraction=seepageFraction,
                        geometry=geometry,
                        plant=plant,
                        penstock=penstock,
                        simulation=simulation)


## End(Not run)
```

---

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

```
Res<-createReservoir(name="R1",type='storage',
                     netEvaporation=rnorm(120,0.5,0.1),
                     geometry=list(deadStorage= 10,
                                   capacity= 700,
                                   storageAreaTable= cbind(seq(0,900,100),seq(0,9,1))))
R<-createRiver(name="Riv1",downstream=Res,discharge=rnorm(120,500,4))
waterVariation<-round(sin(seq(0,pi,length.out=12))*
                      100/sum(sin(seq(0,pi,length.out=12))))
D1<-createDemandSite(name ="D1",
                     demandParams=list(waterUseRate=5,
                                       waterVariation=waterVariation,
                                       cropArea=500),
                     suppliers=list(Res),priority=1)
D2<-createDemandSite(name ="D2",
                     demandParams=list(waterUseRate=5,
                                       waterVariation=waterVariation,
                                       cropArea=500),
                     suppliers=list(Res),priority=2)
D3<-createDemandSite(name ="D3",
                     demandParams=list(waterUseRate=5,
                                       waterVariation=waterVariation,
                                       cropArea=500),
                     suppliers=list(Res),priority=3)
area<-createArea(simulation=list(start='2000-01-01',end='2009-12-29',interval='month'))
area<-addObjectToArea(area,R)
area<-addObjectToArea(area,Res)
area<-addObjectToArea(area,D1)
area<-addObjectToArea(area,D2)
area<-addObjectToArea(area,D3)
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=NA, priority = NA, discharge, seepageFraction=NA, simulation)
```

Arguments

demand   (optional) A matrix: is column-wise matrix of demands, at which the rows presents demands for each time step and columns are for different individual demand sites (MCM).

priority   (optional) A vector: is a vector of priorities associated to demand

discharge   (optional) A vector : a vector of water flowing into the diversion (MCM)

seepageFraction

                    (optional) The seepage coeffcient of river discharge flow. The seepage is com-
                    puted as the product of seepageFraction and river discharge. It is in [0, 1]
                    interval

simulation   A list: `simulation` is a list of three vectors: `start`, `end`, and `interval`. the
                    `start` and `end` components must be in `'YYYY-MM-DD'` format. the `interval`
                    component can takes either of `'day'`,`'week'`, or `'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(480,15,3),120)
priority        <-sample(1:3,4,replace=TRUE)
discharge       <-rlnorm(120,log(50),log(4))
simulation      <-list(start='2000-01-01',end='2000-04-29',interval='day')

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

---

set.as   *WRSS objects connector*

Description

this function connects a base object as a either of: `'downstream'`, `'supplier'`, `'leakageObject'`,
`'seepageObject'`, or `'divertObject'` to a `target` object, which are both instantiated by WRSS
constructors.

Usage

```
set.as(base,target,type='downstream')
```

Arguments

base	An object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite
target	An object; from either of classes of createAquifer , createRiver, createReservoir, createJunction, createDiversion, or createDemandSite
type	the type of base object to be set as to the target object: 'downstream', 'supplier', 'leakageObject', 'seepageObject', or 'divertObject'

Value

an object from class of target object.

Author(s)

Rezgar Arabzadeh

See Also

addObjectToArea

---

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:
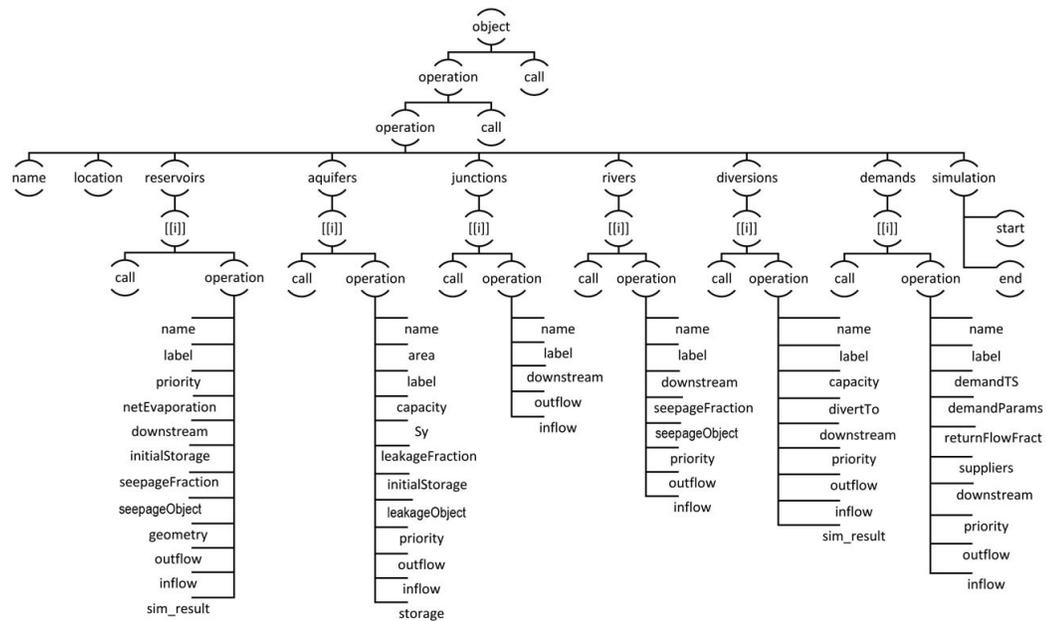
Author(s)

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](#)

zarrineh   *data of Zarrineh-rud river basin*

Description

The zarrineh object, is a list of objects including time series and detail a five-reservoir systen in the Zarrineh-rud river basin.

Format

list object

References

Iran Water Resources Management Company, 2016.

Index