

# Package ‘Watersheds’

October 12, 2022

**Type** Package

**Title** Spatial Watershed Aggregation and Spatial Drainage Network Analysis

**Version** 1.1

**Date** 2016-02-08

**Author** J.A. Torres-Matallana

**Maintainer** J. A. Torres-Matallana <arturo.torres@list.lu>

**Description** Methods for watersheds aggregation and spatial drainage network analysis.

**License** GPL (>= 2)

**Depends** R (>= 2.10), methods, sp, maptools, rgeos, lattice, splancs

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-02-09 14:36:29

## R topics documented:

Watersheds-package . . . . .	2
plot.PointAttribute-methods . . . . .	2
plot.PolyLineAttribute-methods . . . . .	3
RiverStation . . . . .	3
SpDF_Subset . . . . .	4
SpDF_Touch . . . . .	5
Watershed . . . . .	6
Watershed.IOR1 . . . . .	8
Watershed.IOR2 . . . . .	9
Watershed.IOR3 . . . . .	11
Watershed.IOR4 . . . . .	12
Watershed.Order-methods . . . . .	14
Watershed.Order2-methods . . . . .	16
Watershed.Tributary-methods . . . . .	19
WatershedsData . . . . .	20

<b>Index</b>	<b>23</b>
--------------	-----------

---

Watersheds-package      *Spatial watershed aggregation and spatial drainage network analysis*

---

### Description

Spatial analysis for watersheds aggregation and ordering accordingly to an outlet point and size of tributary watershed of the current watershed. Spatial drainage networks analysis inside the aggregated watersheds.

### Details

Package:    Watersheds  
Type:        Package  
Version:    1.1  
Date:        2016-02-08  
License:    GPL (>= 2)  
Depends:    R (>= 2.10), methods, sp, maptools, rgeos, lattice, splancs, multicore

Creation and handling of objects class Watershed for identifying the subbasin that contains the current station (class SpatialPoints) and subsets the zhyd object to subbasin and extract the current zhy object that contains station via the S4 method Watershed.Order. Identification of the inlet and outlet stretches and inlet and outlet nodes of the zhyd. Implementation of functions Watershed. , IOR1, IOR2, IOR3, and IOR4 for determining the actual inlet and outlet nodes. S4 methods Watershed.Order2 and Watershed.Tributary for defining tributary nodes and tributary catchments of the current zhyd watershed.

### Author(s)

J.A. Torres-Matallana

Maintainer: J.A. Torres-Matallana <arturo.torres@list.lu>

### See Also

See Also the class [Watershed](#) and the methods [Watershed.Order](#), [Watershed.Order2](#) and [Watershed.Tributary](#).

---

plot.PointAttribute-methods

*Plotting attributes of SpatialPointsDataFrame objects*

---

### Description

S4 Method for plotting attributes of SpatialPointsDataFrame objects.

**Methods**

```
signature(x = "SpatialPointsDataFrame", y = "character", dist = "numeric", cex = "numeric")
```

x A "SpatialPointsDataFrame" object from where the coordinates of the attribute will be retrieved.

y A "character" with the name of the attribute.

dist A "numeric" with the distance from the coordinate to plot the attribute text.

cex A "numeric" with the relative size to plot the attribute text.

plot.PolyLineAttribute-methods

*Plot attributes of Spatial-Lines, Polygons-DataFrame objects.*

**Description**

S4 Method for plotting attributes of SpatialLinesDataFrame and SpatialPolygonsDataFrame objects.

**Methods**

```
signature(x = "SpatialPolygonsDataFrame", y = "character", dist = "numeric", cex = "numeric")
```

x "SpatialPointsDataFrame" or "SpatialPolygonsDataFrame" object from where the coordinates of the attribute will be retrieved.

y "character" with the name of the attribute.

dist "numeric" with the distance from the coordinate to plot the attribute text.

cex "numeric" with the relative size to plot the attribute text.

RiverStation

*Intersection of SpatialPoints and SpatialLinesDataFrame*

**Description**

The function intersects objects SpatialPoints and SpatialLinesDataFrame. Identifies the closer stretch(es) to a station. The SpatialPoints must be length 1.

**Usage**

```
RiverStation(x, y, window = 100)
```

**Arguments**

x	An object of class SpatialPoints as is defined in package sp and length 1.
y	An object of class SpatialLinesDataFrame as is defined in package sp.
window	A numeric value that represents the size of the square (window) around the x object.

**Details**

window value magnifies the object x in order to certainly secure the intersection with the object y. The greater value the more intersection area is defined.

**Value**

An object SpatialLinesDataFrame that is a subset of th object x that represents the current intersection withe object x.

**Author(s)**

J.A. Torres

**Examples**

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
river1 = WatershedsData$river

tributary = RiverStation(station1, river1)
plot(tributary, col="blue")
plot(station1,pch=21,bg="red",cex=.8,add=TRUE)
plot.PolyLineAttribute(x=tributary, y="OBJECTID", dist=100, cex=.8)
title(main="Point station and tributary rivers")
```

---

SpDF\_Subset

*Subsetting spatial dataframe objects*

---

**Description**

Given and list x of logical values, the function subsets the object z accordingly the TRUE values of x.

**Usage**

```
SpDF_Subset(x, y)
```

**Arguments**

- x                    A list of logical values where TRUE values indicates the index of the subset.  
y                    A spatial object as is defined in package sp from extracting the subset.

**Value**

A spatial object of the same class of y.

**Author(s)**

J.A. Torres

**Examples**

```
library(Watersheds)
data(WatershedsData)

# subsetting the river Werra subbasin
id = list(gIntersects(WatershedsData$rWerra, WatershedsData$subbasin,byid=TRUE))
subbasin_rWerra = SpDF_Subset(id,WatershedsData$subbasin)
plot(subbasin_rWerra)

# subsetting the river Werra zhyd watersheds
id = list(gIntersects(WatershedsData$rWerra, WatershedsData$zhyd,byid=TRUE))
zhyd_rWerra = SpDF_Subset(id,WatershedsData$zhyd)
plot(WatershedsData$rWerra,col="blue",lwd=1,add=TRUE)
plot(zhyd_rWerra,col="green3",add=TRUE)
title("Subbasin River Weser and primary zhyd watersheds")

# subsetting the river Werra river drainage watersheds
id = list(gIntersects(subbasin_rWerra, WatershedsData$river,byid=TRUE))
river_rWerra = SpDF_Subset(id,WatershedsData$river)
plot(subbasin_rWerra)
plot(WatershedsData$rWerra,col="blue",lwd=3,add=TRUE)
plot(river_rWerra,col="blue1",add=TRUE)
title("Subbasin River Weser and drainage network")
```

---

SpDF\_Touch

*Touch function for spatial objects*

---

**Description**

The SpatialDataFrame Touch function. Identifies which nodes has touching lines and retrieves a list with two elements.

**Usage**

SpDF\_Touch(x, y)

**Arguments**

x	An spatial object as is described in package sp.
y	An spatial object as is described in package sp.

**Value**

A list with two elements:

comp1	A matrix with the OBJECTID of the node (column 1), the maximum number of lines that are touching the node (column 2), and the elevation of that node (column 3).
comp2	A matrix with the OBJECTID of the lines that are touching the node.

**Author(s)**

J.A. Torres

**Examples**

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4328448.74, 3118576.86),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)

a = Watershed.Order(watershed)
c1 = a[[1]]
riverIO = a[[8]]
nodeIO = a[[9]]

touch = SpDF_Touch(nodeIO, riverIO)
touch1 = touch[[1]]; touch1
```

---

Watershed

*Class "Watershed"*

---

**Description**

A S4 class "Watershed" for representing "Watershed" objects.

## Objects from the Class

Objects can be created by calls of the form `new("Watershed", ...)`.

## Slots

**station:** Object of class "SpatialPoints" of length 1. Represents a point from which aggregation fo watersheds will occur.

**subbasin:** Object of class "SpatialPolygonsDataFrame" of length 1. Represents the current boundary of the hydrological units or zhyd objects.

**zhyd:** Object of class "SpatialPolygonsDataFrame". Represents the current hydrological units (zhyd accordingly to ECRINS (EAA, 2012)) to be analyzed inside the subbasin boundary.

**river:** Object of class "SpatialLinesDataFrame" that represents the current river network to be analysed inside the subbasin boundary.

**c1:** Object of class "SpatialPolygonsDataFrame" of length 1. Represents the curren zhyd object of analysis.

**node:** Object of class "SpatialPointsDataFrame". Represents the current nodes of the river network to be analysed inside the subbasin boundary.

## Methods

**Watershed.Order** signature(x = "Watershed"): ...

**Watershed.Order2** signature(watershed = "Watershed"): ...

**Watershed.Tributary** signature(x = "SpatialPointsDataFrame", xo = "SpatialPointsDataFrame", y = "SpatialLinesDataFrame", z = "SpatialPointsDataFrame", zhyd = "SpatialPolygonsDataFrame", c1 = "SpatialPolygonsDataFrame")

## Author(s)

J.A. Torres-Matallana

## References

European Environment Agency - EAA. (2012). EEA catchments and rivers network system, ECRINS v1.1. rationales, building and improving for widening uses to Water Accounts and WISE applications (EEA Technical report No. 7/2012). (Luxembourg: Publications Office of the European Union).

## See Also

See Also as the functions [Watershed.IOR1](#), [Watershed.IOR2](#), [Watershed.IOR3](#), [Watershed.IOR4](#), or the S4 methods [Watershed.Order](#), [Watershed.Order2](#), [Watershed.Tributary](#)

**Examples**

```

data(WatershedsData)
station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=slot(station1,"coords"),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)

```

---

Watershed.IOR1

*Watershed inlet and outlet nodes: case 1*


---

**Description**

The function determines the inlet and outlet nodes for zhyd watershed objects. This case 1 is for those watersheds that its river inlet and outlet object is length 1 ( $\text{length}(\text{riverIO})=1$ ).

**Usage**

```
Watershed.IOR1(x, dist)
```

**Arguments**

x	An object "SpatialPointsDataFrame" as is described in package sp over the function will search the inlet and outlet nodes of the watershed.
dist	A vector with the distances of each point in x to the current zhyd boundary.

**Value**

A list of length 2:

inlet	A "SpatialPointsDataFrame" that represents the inlet node of the current zhyd.
outlet	A "SpatialPointsDataFrame" that represents the outlet node of the current zhyd.

**Note**

If there are not inlet or outlet node of the current zhyd, 0 is returned.

**Author(s)**

J.A. Torres



**See Also**

See Also the functions [Watershed.IOR2](#), [Watershed.IOR3](#), [Watershed.IOR4](#).

**Examples**

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4232972,3327634),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)

a = Watershed.Order(watershed)
c1 = a[[1]]
nodeIO = a[[9]]
c1_river = a[[10]]

# determining inlet and outlet watershed nodes
# determining distances of nodeIO to c1
boundary = gBoundary(c1)
dist = gDistance(nodeIO, boundary, byid =TRUE)
a = Watershed.IOR1(x=nodeIO, dist=dist)
c1_inlet = a$inlet; c1_inlet
c1_outlet = a$outlet; c1_outlet

plot(c1,col="gray50")
plot(station1,pch=24, bg="blue",add= TRUE)
plot(c1_river, col="blue", add=TRUE)
plot(c1_outlet,pch=21, bg="red",add= TRUE)
plot.PointAttribute(c1_outlet,"ELEV",700,0.8)
title(main="Watershed outlet, case I")
```

---

Watershed.IOR2

*Watershed inlet and outlet nodes: case 2*


---

**Description**

The function determines the inlet and outlet nodes for zhyd watershed objects. This case 2 is for those watersheds that its river inlet and outlet object is length 2 (length(riverIO)=2).

**Usage**

```
Watershed.IOR2(x, dist, node)
```

**Arguments**

x	An object "SpatialPointsDataFrame" or "SpatialPoints" as are described in package sp over the function will search the inlet and outlet nodes of the watershed.
dist	A vector with the distances of each point in x to the current zhyd boundary.
node	An object "SpatialPointsDataFrame" as are described in package sp over the function will search the inlet and outlet nodes of the watershed. It must be the entire node search object.

**Value**

A list of length 2:

inlet	A "SpatialPointsDataFrame" that represents the inlet node of the current zhyd.
outlet	A "SpatialPointsDataFrame" that represents the outlet node of the current zhyd.

**Note**

If there are not inlet or outlet node of the current zhyd is returned 0.

**Author(s)**

J.A. Torres

**See Also**

See Also the functions [Watershed.IOR1](#), [Watershed.IOR3](#), [Watershed.IOR4](#).

**Examples**

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4330341.36, 3284797.06),
proj4string=slot(subbasin1, "proj4string"))
watershed = new("Watershed", station=station1, subbasin=subbasin1,
zhyd=zhyd1, river=river1, c1=subbasin1, node=node1)

a = Watershed.Order(watershed)
c1 = a[[1]]
nodeIO = a[[9]]
c1_river = a[[10]]
```

```

c1_node = a[[11]]

# determining inlet and outlet watershed nodes
# determining distances of nodeIO to c1
boundary = gBoundary(c1)
dist = gDistance(nodeIO, boundary, byid =TRUE)
a = Watershed.IOR2(x=nodeIO, dist=dist, node=c1_node)
str(a)
c1_inlet = a$inlet; c1_inlet
c1_outlet = a$outlet; c1_outlet

plot(c1,col="gray60")
plot(station1,pch=24, bg="blue",add= TRUE)
plot(c1_river, col="blue", add=TRUE)
plot(c1_outlet,pch=21, bg="red",add= TRUE)
plot.PointAttribute(c1_outlet,"ELEV",700,0.8)
title(main="Watershed outlet, case II")

```

---

Watershed.IOR3

*Watershed inlet and outlet nodes: case 3*


---

### Description

The function determines the inlet and outlet nodes for zhyd watershed objects. This case 3 is for those watersheds that its river inlet and outlet object is length 3 ( $\text{length}(\text{riverIO})=3$ ).

### Usage

```
Watershed.IOR3(x, y, dist)
```

### Arguments

x	An object "SpatialPointsDataFrame" as is described in package sp over them the function will search the inlet and outlet nodes of the watershed.
y	An object "SpatialLinesDataFrame" as is described in package sp that represents the inlet and outlet rivers of the watershed.
dist	A vector with the distances of each point in x to the current zhyd boundary.

### Value

inlet	A "SpatialPointsDataFrame" that represents the inlet node of the current zhyd.
outlet	A "SpatialPointsDataFrame" that represents the outlet node of the current zhyd.

### Note

If there are not inlet or outlet node of the current zhyd is returned 0.

**Author(s)**

J.A. Torres

**See Also**See Also the functions [Watershed.IOR1](#), [Watershed.IOR2](#), [Watershed.IOR4](#).**Examples**

```

library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4217199.42, 3353511.83),
proj4string=slot(subbasin1, "proj4string"))
watershed = new("Watershed", station=station1, subbasin=subbasin1,
zhyd=zhyd1, river=river1, c1=subbasin1, node=node1)

a = Watershed.Order(watershed)
c1 = a[[1]]
riverIO = a[[8]]
nodeIO = a[[9]]
c1_river = a[[10]]

# determining inlet and outlet watershed nodes
# determining distances of nodeIO to c1
boundary = gBoundary(c1)
dist = gDistance(nodeIO, boundary, byid =TRUE)
a = Watershed.IOR3(x=nodeIO, y=riverIO, dist=dist)
c1_inlet = a$inlet; c1_inlet
c1_outlet = a$outlet; c1_outlet

plot(c1,col="gray60")
plot(station1,pch=24, bg="blue",add= TRUE)
plot(c1_river, col="blue", add=TRUE)
plot(c1_outlet,pch=21, bg="red",add= TRUE)
plot(c1_inlet,pch=21, bg="green",add= TRUE)
plot.PointAttribute(c1_outlet,"ELEV",1000,0.8)
plot.PointAttribute(c1_inlet,"ELEV",1000,0.8)
title(main="Watershed outlet and inlet, case III")

```

**Description**

The function determines the inlet and outlet nodes for zhyd watershed objects. This case 4 is for those watersheds that its river inlet and outlet object is length 4 ( $\text{length}(\text{riverIO})=4$ ).

**Usage**

```
Watershed.IOR4(x, y, dist)
```

**Arguments**

x	An object "SpatialPointsDataFrame" as is described in package sp over them the function will search the inlet and outlet nodes of the watershed.
y	An object "SpatialLinesDataFrame" as is described in package sp that represents the inlet and outlet rivers of the watershed.
dist	A vector with the distances of each point in x to the current zhyd boundary.

**Value**

inlet	A "SpatialPointsDataFrame" that represents the inlet node of the current zhyd.
outlet	A "SpatialPointsDataFrame" that represents the outlet node of the current zhyd.

**Note**

If there are not inlet or outlet node of the current zhyd is returned 0.

**Author(s)**

J.A. Torres

**See Also**

See Also the functions [Watershed.IOR1](#), [Watershed.IOR2](#), [Watershed.IOR3](#).

**Examples**

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4357947,3284525),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)
```

```

a = Watershed.Order(watershed)
c1 = a[[1]]
riverIO = a[[8]]
nodeIO = a[[9]]
c1_river = a[[10]]

# determining inlet and outlet watershed nodes
# determining distances of nodeIO to c1
boundary = gBoundary(c1)
dist = gDistance(nodeIO, boundary, byid =TRUE)
a = Watershed.IOR4(x=nodeIO, y=riverIO, dist=dist)
c1_inlet = a$inlet; c1_inlet
c1_outlet = a$outlet; c1_outlet

plot(c1,col="gray60")
plot(station1,pch=24, bg="blue",add= TRUE)
plot(c1_river, col="blue", add=TRUE)
plot(c1_outlet,pch=21, bg="red",add= TRUE)
plot(c1_inlet,pch=21, bg="green",add= TRUE)
plot.PointAttribute(c1_outlet,"ELEV",1000,0.8)
plot.PointAttribute(c1_inlet,"ELEV",1000,0.8)
title(main="Watershed outlet and inlet, case IV")

```

---

Watershed.Order-methods

*S4 Method for Function Watershed.Order*

---

### Description

S4 Method for function Watershed.Order. Definition of the properties of the current zhyd watershed.

### Value

The method returns a list of 11 objects:

c1	An object SpatialPolygonsDataFrame of length 1 that represents the current zhyd watershed object.
c1_inlet	An object SpatialPointsDataFrame of length 1 that represents the current inlet node of the zhyd watershed object.
c1_outlet	An object SpatialPointsDataFrame of length 1 that represents the current outlet node of the zhyd watershed object.
c2	An object SpatialPolygonsDataFrame of length 1 that represents the greater watershed tributary of the current zhyd watershed object.
c3	An object SpatialPolygonsDataFrame of length 1 that represents the second watershed tributary of the current zhyd watershed object.

node_trib	An object <code>SpatialPointsDataFrame</code> of length 2 that represents the station points of the tributary watershed objects.
sb1	An object <code>SpatialPointsDataFrame</code> of length 1 that represents the subbasin that contains the current zhyd watershed object.
riverIO	An object <code>SpatialLinesDataFrame</code> that represents the inlet (I) and outlet (O) rivers that crosses the boundary of the current zhyd watershed object.
nodeIO	An object <code>SpatialPointsDataFrame</code> that represents the nodes of the inlet (I) and outlet (O) rivers that crosses the boundary of the current zhyd watershed object.
c1_river	An object <code>SpatialLinesDataFrame</code> that represents the river network inside the current zhyd watershed object.
c1_node	An object <code>SpatialPointsDataFrame</code> that represents the node network inside the current zhyd watershed object.

## Methods

`signature(x = "Watershed")` The function takes the object of class `Watershed` and identifies the subbasin that contains the current station (class `SpatialPoints`) and subsets the `zhyd` object to subbasin and extract the current `zhy` object that contains station. Posteriorly, identifies the inlet and outlet stretches and probable inlet and outlet nodes of the `zhyd`. Then, runs the functions `Watershed. ,IOR1, IOR2, IOR3, or IOR4` for determining the actual inlet and outlet nodes. Finally, the method executes the S4 method `Watershed.Tributary` for defining tributary nodes and tributary catchments of the current `zhyd` watershed.

## See Also

See Also the class [Watershed](#) and the methods [Watershed.Order2](#) and [Watershed.Tributary](#).

## Examples

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4328448.74, 3118576.86),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)

a = Watershed.Order(watershed)
c1 = a[[1]]
c1_inlet = a[[2]]
c1_outlet = a[[3]]
```

```

c2 = a[[4]]
c3 = a[[5]]
node_trib = a[[6]]
sb1 = a[[7]]
riverIO = a[[8]]
nodeIO = a[[9]]
c1_river = a[[10]]
c1_node = a[[11]]

bbox1 = slot(c1, "bbox")
bbox = matrix(0,2,2)
bbox[,1] = bbox1[,1]*.998
bbox[,2] = bbox1[,2]*1.002

plot(c1, xlim=bbox[1,], ylim=bbox[2,],col="gray50")
plot(c2, col="gray75", add=TRUE)
plot(c3, col="gray85", add=TRUE)
plot(slot(watershed,"station"),pch=24, bg="blue",add= TRUE)
plot.PolyLineAttribute(c1, "order", 450, 0.8)
plot.PolyLineAttribute(c2, "order", 450, 0.8)
plot.PolyLineAttribute(c3, "order", 450, 0.8)
plot(c1_river, col="blue", add=TRUE)
plot(c1_node,pch=21,bg="blue",cex=.5,add=TRUE)
plot(nodeIO,pch=21,bg="blue",cex=.5,add=TRUE)
plot(c1_inlet, pch=21, bg="green",add= TRUE)
plot(c1_outlet,pch=21, bg="red",add= TRUE)
plot.PointAttribute(nodeIO,"ELEV",600,0.7)
title(main="Current zhyd watershed (1)",
sub="First order tributary watersheds (1.1, 1.2)")

```

---

Watershed.Order2-methods

*S4 Method for Function Watershed.Order2*

---

## Description

S4 Method for function Watershed.Order2. Definition of the tributary zhyd watersheds of the current zhyd watershed.

## Value

The method returns a list of 2 objects:

- |    |   |
|----|---|
| c2 | An object with the output of the method Watershed.Order of length 11 for one of the points of node_trib. The properties of the greater tributary watershed of the current zhyd watershed. |
| c3 | An object with the output of the method Watershed.Order of length 11 for the other points of node_trib. The properties of the second tributary watershed of the current zhyd watershed.   |



## Methods

signature(watershed = "Watershed") The method takes the object of class Watershed when object node\_trib is length 2. The method identifies the zhyd watershed that contains the current station (class SpatialPoints) and apply the method Watershed.Order on each point of node\_trib returning a list of objects Watershed.Order. The computation is done via parallel processes for optimizing and take advantage of multicore functionalities.

## See Also

See Also the class [Watershed](#) and the methods [Watershed.Order](#) and [Watershed.Tributary](#).

## Examples

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4328650,3174450),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)

a = Watershed.Order(watershed)
c1 = a[[1]]
node_trib = a[[6]]
c1_river = a[[10]]

nt = node_trib
sb = subbasin1
z1 = zhyd1
r1 = river1
n1 = node1

watershed2 = new("Watershed", station=nt, subbasin=sb, zhyd=z1, river=r1, c1=c1,node=n1)
c23 = Watershed.Order2(watershed2)
c2 = c23[[1]]
c3 = c23[[2]]

c2.0 = c2[[1]]
c2_inlet = c2[[2]]
c2_outlet = c2[[3]]
c2.1 = c2[[4]]
c2.2 = c2[[5]]
c2_node_trib = c2[[6]]
c2_sb1 = c2[[7]]
c2_riverIO = c2[[8]]
c2_nodeIO = c2[[9]]
```

```

c2_river = c2[[10]]
c2_node = c2[[11]]

c3.0 = c3[[1]]
c3_inlet = c3[[2]]
c3_outlet = c3[[3]]
c3.1 = c3[[4]]
c3.2 = c3[[5]]
c3_node_trib = c3[[6]]
c3_sb1 = c3[[7]]
c3_riverIO = c3[[8]]
c3_nodeIO = c3[[9]]
c3_river = c3[[10]]
c3_node = c3[[11]]

# subsetting river networks
id = list(gIntersects(c2.1, WatershedsData$river,byid=TRUE))
c21_river = SpDF_Subset(id,WatershedsData$river)

id = list(gIntersects(c2.2, WatershedsData$river,byid=TRUE))
c22_river = SpDF_Subset(id,WatershedsData$river)

id = list(gIntersects(c3.1, WatershedsData$river,byid=TRUE))
c31_river = SpDF_Subset(id,WatershedsData$river)

id = list(gIntersects(c3.2, WatershedsData$river,byid=TRUE))
c32_river = SpDF_Subset(id,WatershedsData$river)

# plots
bbox1 = slot(c3.2, "bbox")
bbox = matrix(0,2,2)
bbox[,1] = bbox1[,1]*.995
bbox[,2] = bbox1[,2]*1.005

plot(c1, col="gray50", xlim=bbox[1,], ylim=bbox[2,])
plot(c2.0, col = "gray95", add=TRUE)
plot(c3.0, col="gray79", add=TRUE)
plot(c2.1, col="gray78", add=TRUE)
plot(c2.2, col="gray85", add=TRUE)
plot(c3.1, col="gray53", add=TRUE)
plot(c3.2, col="gray63", add=TRUE)

plot(c1_river, col="blue",add=TRUE)
plot(c2_river, col="blue",add=TRUE)
plot(c3_river, col="blue",add=TRUE)
plot(c21_river, col="blue",add=TRUE)
plot(c22_river, col="blue",add=TRUE)
plot(c31_river, col="blue",add=TRUE)
plot(c32_river, col="blue",add=TRUE)

```

---

 Watershed.Tributary-methods

*S4 Method for Function Watershed.Tributary*


---

### Description

S4 Method for function Watershed.Tributary. Definition of the order of tributary zhyd watersheds of the current zhyd watershed.

### Value

The method returns a list of 4 objects:

c2c3	A list of length 2 with objects SpatialPolygonsDataFrame of length 1 ordered that represents the greater watershed and second tributary of the current zhyd watershed object.
c2	An object SpatialPolygonsDataFrame of length 1 ordered that represents the greater watershed tributary of the current zhyd watershed object.
c3	An object SpatialPolygonsDataFrame of length 1 ordered that represents the second watershed tributary of the current zhyd watershed object.
node_trib	An object SpatialPointsDataFrame of length 2 that represents the station points of the tributary watershed objects.

### Methods

signature(x = "SpatialPointsDataFrame", xo = "SpatialPointsDataFrame", y = "SpatialLinesDataFrame", z = '')

### See Also

See Also the class [Watershed](#) and the methods [Watershed.Order](#) and [Watershed.Order2](#).

### Examples

```
library(Watersheds)
data(WatershedsData)

station1 = WatershedsData$station
subbasin1 = WatershedsData$subbasin
zhyd1 = WatershedsData$zhyd
river1 = WatershedsData$river
node1 = WatershedsData$node

station1 = SpatialPoints(coords=cbind(4328448.74, 3118576.86),
proj4string=slot(subbasin1,"proj4string"))
watershed = new("Watershed",station=station1,subbasin=subbasin1,
zhyd=zhyd1,river=river1,c1=subbasin1,node=node1)
```

```

a = Watershed.Order(watershed)
c1 = a[[1]]
c1_inlet = a[[2]]
c1_outlet = a[[3]]
sb1 = a[[7]]
riverI0 = a[[8]]
nodeI0 = a[[9]]
c1_river = a[[10]]
c1_node = a[[11]]

a = Watershed.Tributary(x=c1_inlet,xo= c1_outlet,y=riverI0,z=nodeI0,zhyd=zhyd1, c1=c1)
c2c3 = a[[1]]
c2 = a[[2]]
c3 = a[[3]]
node_trib = a[[4]]

bbox1 = slot(c2c3, "bbox")
bbox = matrix(0,2,2)
bbox[,1] = bbox1[,1]*.998
bbox[,2] = bbox1[,2]*1.002

plot(c1, xlim=bbox[1,], ylim=bbox[2,],col="gray50")
plot(c2, col="gray75", add=TRUE)
plot(c3, col="gray85", add=TRUE)
plot(slot(watershed,"station"),pch=24, bg="blue",add= TRUE)
plot.PolyLineAttribute(c1, "order", 450, 0.8)
plot.PolyLineAttribute(c2, "order", 450, 0.8)
plot.PolyLineAttribute(c3, "order", 450, 0.8)
plot(c1_river, col="blue", add=TRUE)
plot(node_trib,pch=21,bg="red",cex=.8,add=TRUE)
plot.PointAttribute(node_trib,"ELEV",600,0.7)
title(main="Current zhyd watershed (1)",
sub="First order tributary nodes (1.1, 1.2)")

```

---

WatershedsData

*A dataset of the ECRINS database for the river Weser basin, Germany.*


---

### Description

The European Environment Agency (EEA) has been developed the Catchments and Rivers Network System (ECRINS) version 1.1. The ECRINS is the hydrographical system currently in use at the European level as well as widely serving as the reference system for the Water Information System (WISE) (EEA,2012). The current version of ECRINS is based on previous work carried out by the Joint Research Centre (JRC) Catchment Characterisation and Modelling (CCM) and the EEA (European Lakes, Dams and Reservoirs Database (Eldred2), European Rivers and Catchments (ERICA)), (EEA,2012).

### Usage

```
data(WatershedsData)
```

**format**

- basin:** an object `SpatialPolygonsDataFrame` as is defined in package `sp` that represents the river Weser basin. The data slot contains 6 variables as attributes of 1 observation.
- ctry:** an object `SpatialPolygonsDataFrame` as is defined in package `sp` that represents the administrative boundary of Germany. The data slot contains 6 variables as attributes of 1 observation.
- node:** an object `SpatialPointsDataFrame` as is defined in package `sp` that represents the nodes of the ECRINS river network of the river Weser basin. The data slot contains 13 variables as attributes of 3882 observations.
- rAller** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Aller, a major tributary of the river Weser. The data slot contains 74 variables as attributes of 88 observations.
- rDiemel** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Diemel, a major tributary of the river Weser. The data slot contains 74 variables as attributes of 39 observations.
- rFulda** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Fulda, a major tributary of the river Weser. The data slot contains 74 variables as attributes of 82 observations.
- rHunte** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Hunte, a major tributary of the river Weser. The data slot contains 74 variables as attributes of 34 observations.
- river** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the ECRINS river network of the river Weser basin. The data slot contains 52 variables as attributes of 3874 observations.
- rWerra** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Werra, a major tributary of the river Weser. The data slot contains 74 variables as attributes of 120 observations.
- rWeser** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Weser. The data slot contains 74 variables as attributes of 104 observations.
- rWumme** an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Wumme, a major tributary of the river Weser. The data slot contains 74 variables as attributes of 18 observations.
- station** an object `SpatialPoints` as is defined in package `sp` that represents a point of interest for which the watershed will be aggregated an ordered. Could be a point with the coordinates of a measurement station.
- subbasin** an object `SpatialPolygonsDataFrame` as is defined in package `sp` that represents the subbasins of the tributaries of the river Weser. The data slot contains 4 variables as attributes of 4 observations.
- zhyd** an object `SpatialPolygonsDataFrame` as is defined in package `sp` that contains the primary hydrological units of the river Weser basin accordingly with ECRINS. The data slot contains 50 variables as attributes and 915 observations.

## References

European Environment Agency - EEA. (2012). EEA catchments and rivers network system, ECRINS v1.1. rationales, building and improving for widening uses to Water Accounts and WISE applications (EEA Technical report No. 7/2012). (Luxembourg: Publications Office of the European Union).

## Examples

```
data(WatershedsData)

# plotting river Weser basin
plot(WatershedsData$ctry)
plot(WatershedsData$basin, col="green4", add=TRUE)
title("River Weser basin, Germany")

# plotting river Weser basin
plot(WatershedsData$ctry)
plot(WatershedsData$basin, col="green4", add=TRUE)
title("River Weser basin, Germany")

# plotting subbasins river Weser basin
plot(WatershedsData$basin)
plot(WatershedsData$subbasin, col="green3",add=TRUE)
plot(WatershedsData$rWeser,col="blue",lwd=2,add=TRUE)
plot(WatershedsData$rAller,col="blue",lwd=1,add=TRUE)
plot(WatershedsData$rDiemel,col="blue",lwd=1,add=TRUE)
plot(WatershedsData$rFulda,col="blue",lwd=1,add=TRUE)
plot(WatershedsData$rHunte,col="blue",lwd=1,add=TRUE)
plot(WatershedsData$rWerra,col="blue",lwd=1,add=TRUE)
plot(WatershedsData$rWiumme,col="blue",lwd=1,add=TRUE)
title("Subbasins River Weser")

# plotting primary zhyd watersheds and drainage network inside river Werra subbasin
# subsetting the river Werra subbasin
id = list(gIntersects(WatershedsData$rWerra, WatershedsData$subbasin,byid=TRUE))
subbasin_rWerra = SpDF_Subset(id,WatershedsData$subbasin)
plot(subbasin_rWerra)

# subsetting the river Werra zhyd watersheds
id = list(gIntersects(WatershedsData$rWerra, WatershedsData$zhyd,byid=TRUE))
zhyd_rWerra = SpDF_Subset(id,WatershedsData$zhyd)
plot(WatershedsData$rWerra,col="blue",lwd=1,add=TRUE)
plot(zhyd_rWerra,col="green3",add=TRUE)
title("Subbasin River Weser and primary zhyd watersheds")

# subsetting the river Werra river drainage watersheds
id = list(gIntersects(subbasin_rWerra, WatershedsData$river,byid=TRUE))
river_rWerra = SpDF_Subset(id,WatershedsData$river)
plot(subbasin_rWerra)
plot(WatershedsData$rWerra,col="blue",lwd=3,add=TRUE)
plot(river_rWerra,col="blue1",add=TRUE)
title("Subbasin River Weser and drainage network")
```

# Index

- \* **PointAttribute**
  - plot.PointAttribute-methods, [2](#)
- \* **PolyLineAttribute**
  - plot.PolyLineAttribute-methods, [3](#)
- \* **RiverStation**
  - RiverStation, [3](#)
- \* **SpDF\_Subset**
  - SpDF\_Subset, [4](#)
- \* **SpDF\_Touch**
  - SpDF\_Touch, [5](#)
- \* **Subset**
  - SpDF\_Subset, [4](#)
- \* **Touch**
  - SpDF\_Touch, [5](#)
- \* **Watershed.IOR1**
  - Watershed.IOR1, [8](#)
- \* **Watershed.IOR2**
  - Watershed.IOR2, [9](#)
- \* **Watershed.IOR3**
  - Watershed.IOR3, [11](#)
  - Watershed.IOR4, [12](#)
- \* **Watershed.IOR4**
  - Watershed.IOR4, [12](#)
- \* **Watershed.Order2**
  - Watershed.Order2-methods, [16](#)
- \* **Watershed.Order**
  - Watershed.Order-methods, [14](#)
- \* **Watershed.Tributary**
  - Watershed.Tributary-methods, [19](#)
- \* **Watershed**
  - Watershed, [6](#)
- \* **classes**
  - Watershed, [6](#)
- \* **datasets**
  - WatershedsData, [20](#)
- \* **methods**
  - plot.PointAttribute-methods, [2](#)
  - plot.PolyLineAttribute-methods, [3](#)
  - Watershed.Order-methods, [14](#)
  - Watershed.Order2-methods, [16](#)
  - Watershed.Tributary-methods, [19](#)

- \* **package**
  - Watersheds-package, [2](#)
- \* **plot.PointAttribute**
  - plot.PointAttribute-methods, [2](#)
- \* **plot.PolyLineAttribute**
  - plot.PolyLineAttribute-methods, [3](#)
- plot.PointAttribute
  - (plot.PointAttribute-methods), [2](#)
  - plot.PointAttribute, SpatialPointsDataFrame, character, numeric, (plot.PointAttribute-methods), [2](#)
  - plot.PointAttribute-methods, [2](#)
  - plot.PolyLineAttribute
    - (plot.PolyLineAttribute-methods), [3](#)
  - plot.PolyLineAttribute, SpatialLinesDataFrame, character, numeric, (plot.PolyLineAttribute-methods), [3](#)
  - plot.PolyLineAttribute, SpatialPolygonsDataFrame, character, numeric, (plot.PolyLineAttribute-methods), [3](#)
  - plot.PolyLineAttribute-methods, [3](#)
- RiverStation, [3](#)
- SpDF\_Subset, [4](#)
- SpDF\_Touch, [5](#)
- Watershed, [2](#), [6](#), [15](#), [17](#), [19](#)
- Watershed-class (Watershed), [6](#)
- Watershed.IOR1, [7](#), [8](#), [10](#), [12](#), [13](#)
- Watershed.IOR2, [7](#), [9](#), [9](#), [12](#), [13](#)
- Watershed.IOR3, [7](#), [9](#), [10](#), [11](#), [13](#)
- Watershed.IOR4, [7](#), [9](#), [10](#), [12](#), [12](#)
- Watershed.Order, [2](#), [7](#), [17](#), [19](#)
- Watershed.Order
  - (Watershed.Order-methods), [14](#)
- Watershed.Order, Watershed-method
  - (Watershed.Order-methods), [14](#)

- Watershed.Order-class
  - (Watershed.Order-methods), [14](#)
- Watershed.Order-methods, [14](#)
- Watershed.Order2, [2](#), [7](#), [15](#), [19](#)
- Watershed.Order2
  - (Watershed.Order2-methods), [16](#)
- Watershed.Order2,Watershed-method
  - (Watershed.Order2-methods), [16](#)
- Watershed.Order2-class
  - (Watershed.Order2-methods), [16](#)
- Watershed.Order2-methods, [16](#)
- Watershed.Tributary, [2](#), [7](#), [15](#), [17](#)
- Watershed.Tributary
  - (Watershed.Tributary-methods),  
[19](#)
- Watershed.Tributary, SpatialPointsDataFrame, SpatialPointsDataFrame, SpatialLinesDataFrame, SpatialPointsData
  - (Watershed.Tributary-methods),  
[19](#)
- Watershed.Tributary-class
  - (Watershed.Tributary-methods),  
[19](#)
- Watershed.Tributary-methods, [19](#)
- Watersheds (Watersheds-package), [2](#)
- Watersheds-package, [2](#)
- WatershedsData, [20](#)