"Watersheds": Spatial watershed aggregation and spatial drainage network analysis

Jairo A. Torres
Institute for Geoinformatics, University of Münster, Münster, Germany
arturo.torres@uni-muenster.de

August 13, 2013

1 Introduction

The present document has the purpose of illustrating the package `Watersheds` implemented in the programming language `R` Project for Statistical Computing (Ihaka & Gentleman, 1996; R Development Core Team, 2013).

The package allows spatial analysis for watersheds aggregation and ordering accordingly to an outlet point and size of tributary watershed of the current watershed. Also, enables spatial drainage networks analysis inside the aggregated watersheds. It makes use of the functionalities of the spatial classes, functions and methods of the package `sp` (Pebesma & Bivand, 2005-2012).

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.

2 The data: river Weser basin, Germany

The package has an example dataset of the ECRINS database for the river Weser basin, Germany. 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). 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)), (European Environment Agency - EAA, 2012).

2.1 Subsets

The dataset contains the following subsets:

- `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.

- `rWiumme` an object `SpatialLinesDataFrame` as is defined in package `sp` that represents the basin of the river Wiumme, a major tributary of the river Weser. The `data` slot contains 74 variables as attributes of 18 observations.

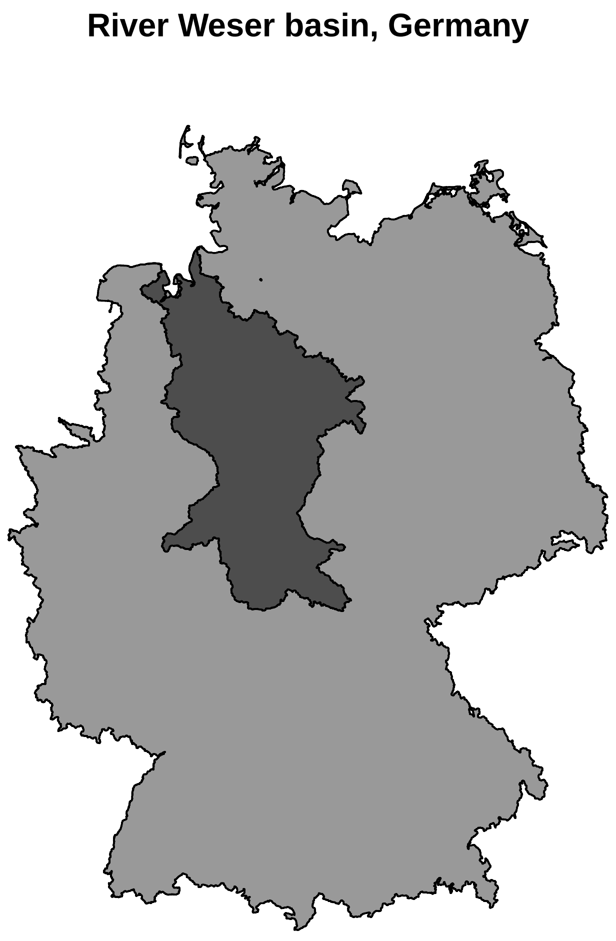
- `station` an object `SpatialPoints` as is defined in package `sp` that represents a point of interest for which the watershed will be aggregated an ordered. Could be a point with the coordinates of a measurement station.

- `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.

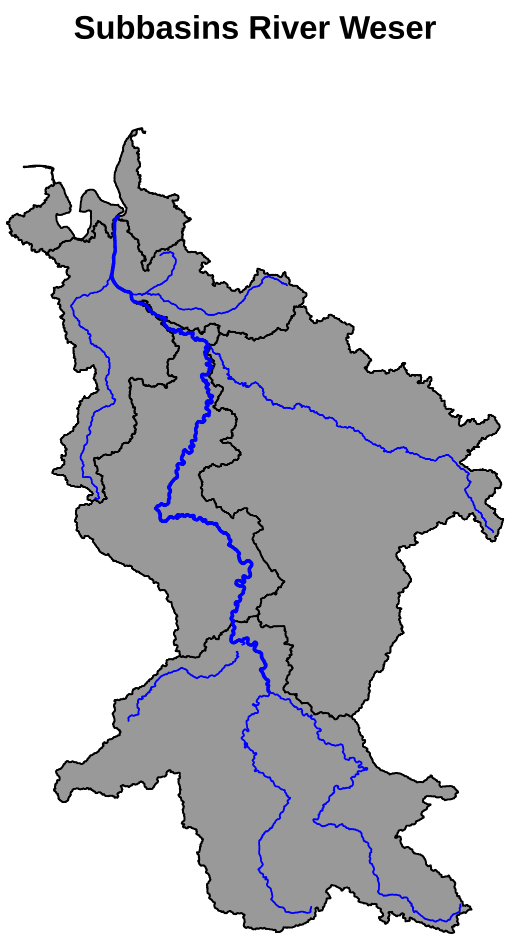
- `zhyd` an object `SpatialPolygonsDataFrame` as is defined in package `sp` that contains the primary hydrological units of the river Weser basin accordingly with ECRINS. The `data` slot contains 50 variables as attributes and 915 observations.

Some examples for visualising the dataset are presented as follows:

```
library(Watersheds)
data(WatershedsData)
# ls() str(WatershedsData) str(WatershedsData['basin'])


# plotting river Weser basin
plot(WatershedsData["ctry"][[1]], col = "gray60")
plot(WatershedsData["basin"][[1]], col = "gray30", add = TRUE)
title("River Weser basin, Germany")
```

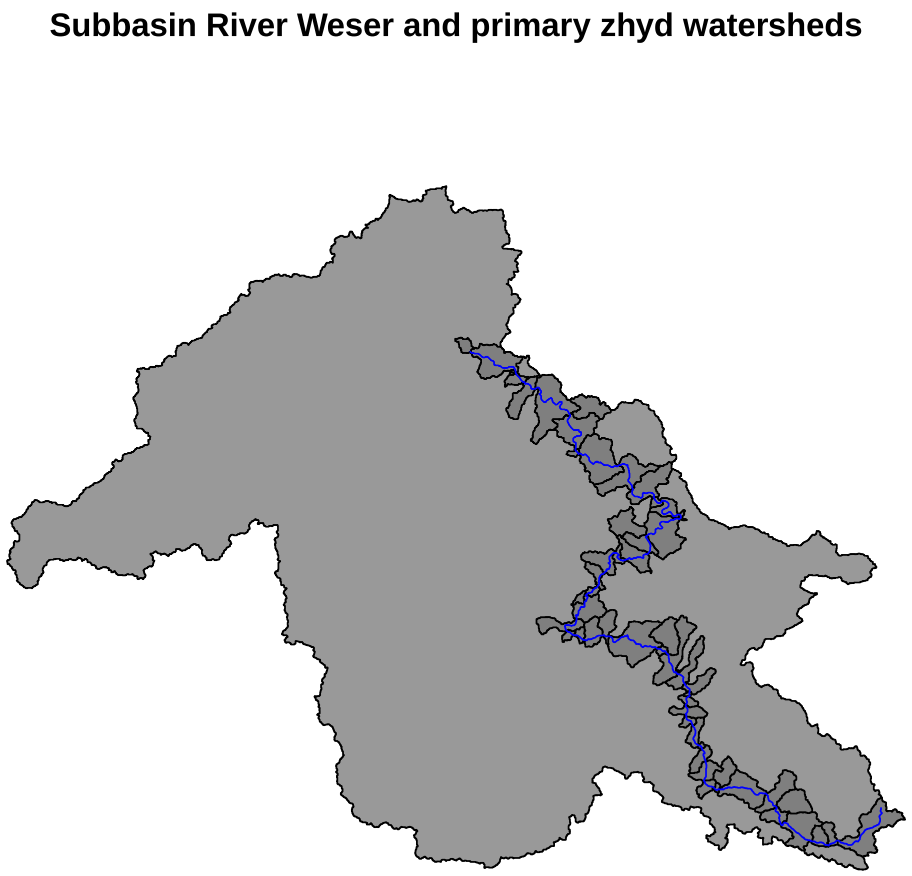


```r
# plotting subbasins river Weser basin
plot(WatershedsData["basin"][[1]])
plot(WatershedsData["subbasin"][[1]], col = "gray60", add = TRUE)
plot(WatershedsData["rWeser"][[1]], col = "blue", lwd = 2, add = TRUE)
plot(WatershedsData["rAller"][[1]], col = "blue", lwd = 1, add = TRUE)
plot(WatershedsData["rDiemel"][[1]], col = "blue", lwd = 1, add = TRUE)
plot(WatershedsData["rFulda"][[1]], col = "blue", lwd = 1, add = TRUE)
plot(WatershedsData["rHunte"][[1]], col = "blue", lwd = 1, add = TRUE)
plot(WatershedsData["rWerra"][[1]], col = "blue", lwd = 1, add = TRUE)
plot(WatershedsData["rWiumme"][[1]], col = "blue", lwd = 1, add = TRUE)
title("Subbasins River Weser")
```

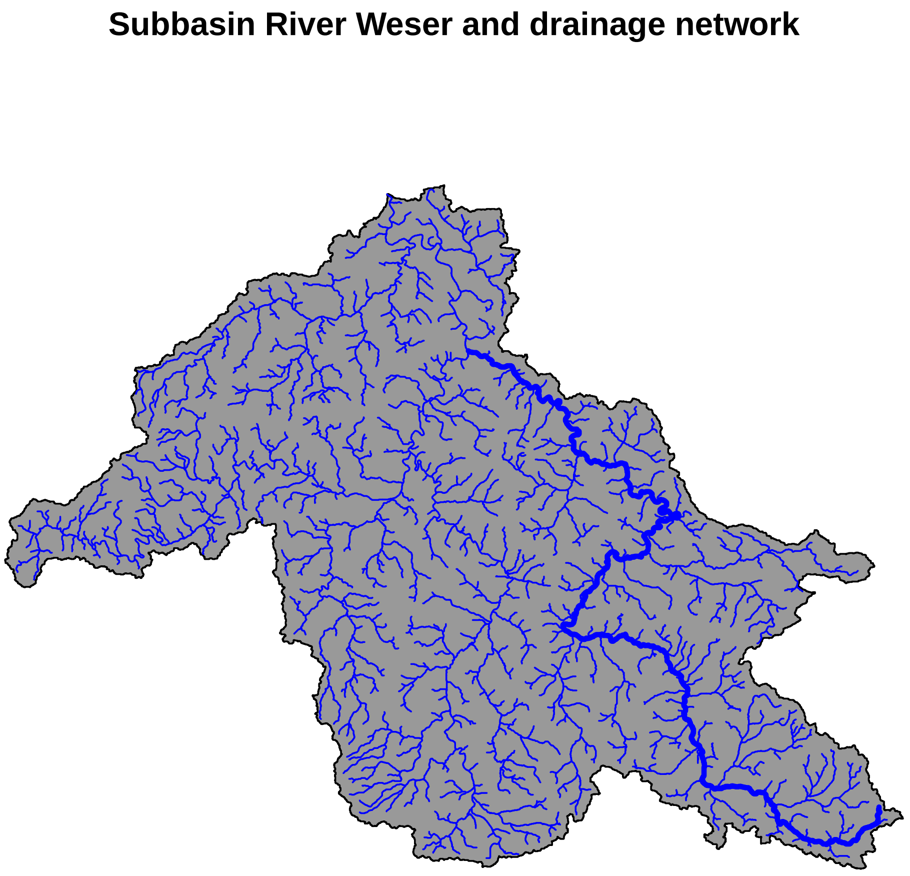


```r
# plotting primary zhyd watersheds and drainage network inside river Werra
# subbasin subsetting the river Werra subbasin
```

```r
id = list(gIntersects(WatershedsData["rWerra"][[1]], WatershedsData["subbasin"][[1]],
    byid = TRUE))
subbasin_rWerra = SpDF_Subset(id, WatershedsData["subbasin"][[1]])

# subsetting the river Werra zhyd watersheds
id = list(gIntersects(WatershedsData["rWerra"][[1]], WatershedsData["zhyd"][[1]],
    byid = TRUE))
zhyd_rWerra = SpDF_Subset(id, WatershedsData["zhyd"][[1]])
plot(subbasin_rWerra, col = "grey60")
plot(zhyd_rWerra, col = "grey50", add = TRUE)
plot(WatershedsData["rWerra"][[1]], col = "blue", lwd = 1, add = TRUE)
title("Subbasin River Weser and primary zhyd watersheds")
```



```
# subsetting the river Werra river drainage watersheds
id = list(gIntersects(subbasin_rWerra, WatershedsData["river"][[1]], byid = TRUE))
river_rWerra = SpDF_Subset(id, WatershedsData["river"][[1]])
plot(subbasin_rWerra, col = "grey60")
plot(WatershedsData["rWerra"][[1]], col = "blue", lwd = 3, add = TRUE)
plot(river_rWerra, col = "blue1", add = TRUE)
title("Subbasin River Weser and drainage network")
```



3 The `Watersheds` objects

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

```
station1 = WatershedsData["station"][[1]]
subbasin1 = WatershedsData["subbasin"][[1]]
zhyd1 = WatershedsData["zhyd"][[1]]
river1 = WatershedsData["river"][[1]]
node1 = WatershedsData["node"][[1]]

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

## [1] "Watershed"
## attr(,"package")
## [1] "Watersheds"
```

4 The `Watersheds.Order` method

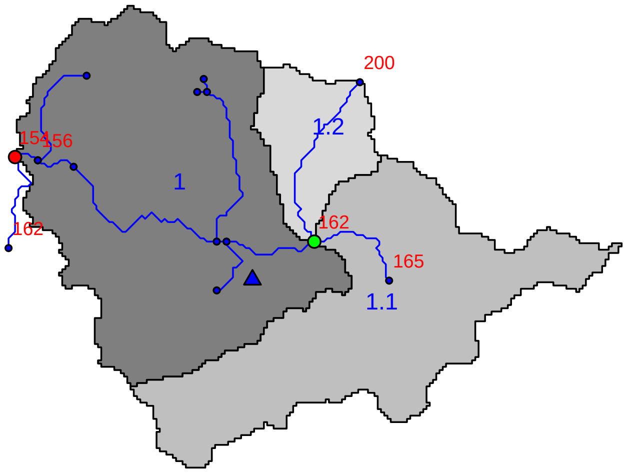
The Method for function `Watershed.Order` allows definition of the properties of the current `zhyd` watershed over `Watershed` objects.

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.

```
station1 = WatershedsData["station"][[1]]
subbasin1 = WatershedsData["subbasin"][[1]]
zhyd1 = WatershedsData["zhyd"][[1]]
river1 = WatershedsData["river"][[1]]
node1 = WatershedsData["node"][[1]]

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)

## [1] "length(riverIO) == 4"

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] * 0.998
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 = 0.5, add = TRUE)
plot(nodeIO, pch = 21, bg = "blue", cex = 0.5, 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)")
```

Current zhyd watershed (1)



First order tributary watersheds (1.1, 1.2)

5 The `Watersheds.Order2` method

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

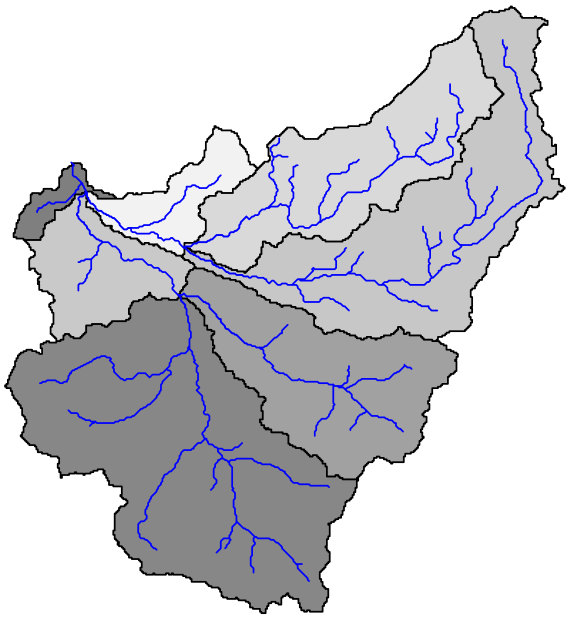
The method takes the objec of class `Watershed` when object `node_trib` is length 2. The method identifies the **`zhyd`** watershed that contaions the current `station` (class `SpatialPoints`) and apply the method `Watershed.Order` on each point of `node_trib` returning a list of objects `Watershed.Order`. The computation is done via parallel processes for optimizing and take advance of multicore functionalities.

```
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)
```

```r
a = Watershed.Order(watershed)

## [1] "end WatershedIO_R3"
## [1] "length(riverIO) == 3"

c1 = a[[1]]
node_trib = a[[6]]
c1_river = a[[10]]

watershed2 = new("Watershed", station = node_trib, subbasin = subbasin1, zhyd = zhyd1,
    river = river1, c1 = c1, node = node1)
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
```

```r
bbox1 = slot(c3.2, "bbox")
bbox = matrix(0, 2, 2)
bbox[, 1] = bbox1[, 1] * 0.995
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)

title(main = "Current zhyd watershed and \n 1st and 2nd order tributary watersheds")
```

Current zhyd watershed and 1st and 2nd order tributary watersheds

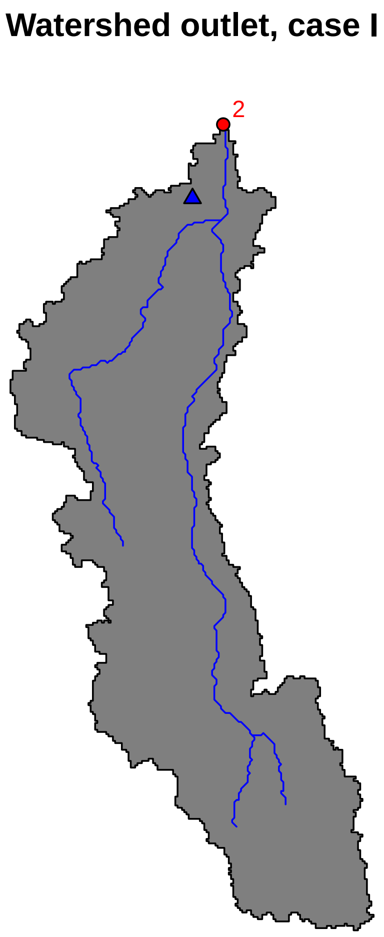


6 The `Watersheds.IOR1` function

The `Watersheds.IOR1` function means Watershed inlet and outlet nodes: case 1. 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 (length(riverIO)=1).

```
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)

## [1] "length(riverIO) == 1"
```

```
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"][[1]]
c1_inlet

## [1] 0

c1_outlet = a["outlet"][[1]]
c1_outlet

##              coordinates OBJECTID     ID WSO_ID SOURCE LEN_TOM NUM_SEG ELEV
## 143 (4234350, 3330950)    25485 587506      7      N   57980       9    2
##     WINDOW WXSOID      NodID Is_2Keep LENK_FRS LENK_TOM
## 143   2000   <NA> Y000587506       -1         0    57.98


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")
```



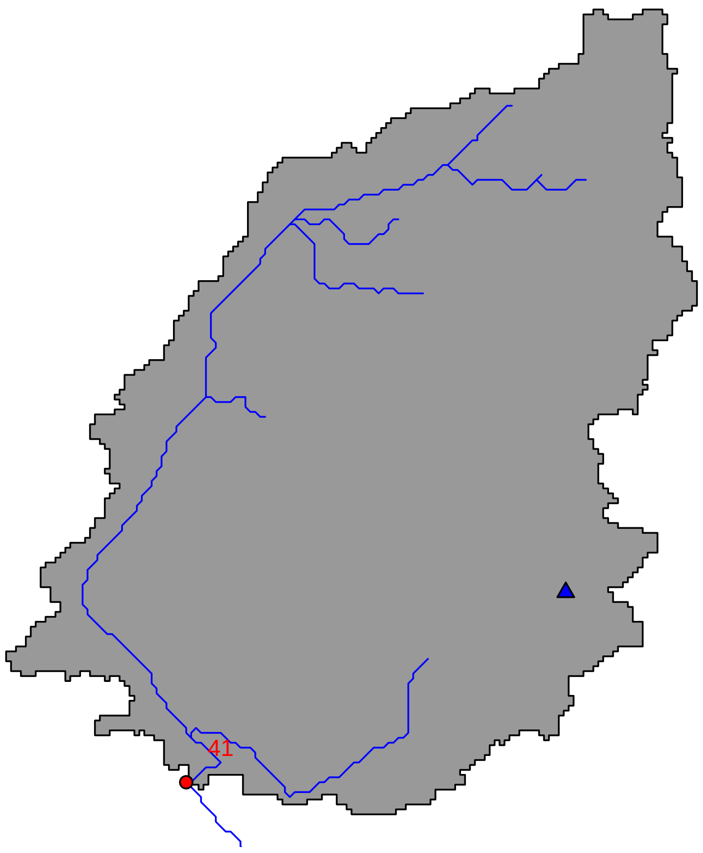
7 The `Watersheds.IOR2` function

The `Watersheds.IOR1` function means Watershed inlet and outlet nodes: case 2. 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).

```
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)

## [1] "length(riverIO) == 2"
```

```
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)
c1_inlet = a["inlet"][[1]]
c1_inlet

## [1] 0

c1_outlet = a["outlet"][[1]]
c1_outlet

##              coordinates OBJECTID     ID WSO_ID SOURCE LEN_TOM NUM_SEG ELEV
## 508 (4322650, 3280950)    29656 599477      7      N  210109      57   41
##     WINDOW WXSOID      NodID Is_2Keep LENK_FRS LENK_TOM
## 508   2000   <NA> Y000599477       -1        0    210.1


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 outlet, case II

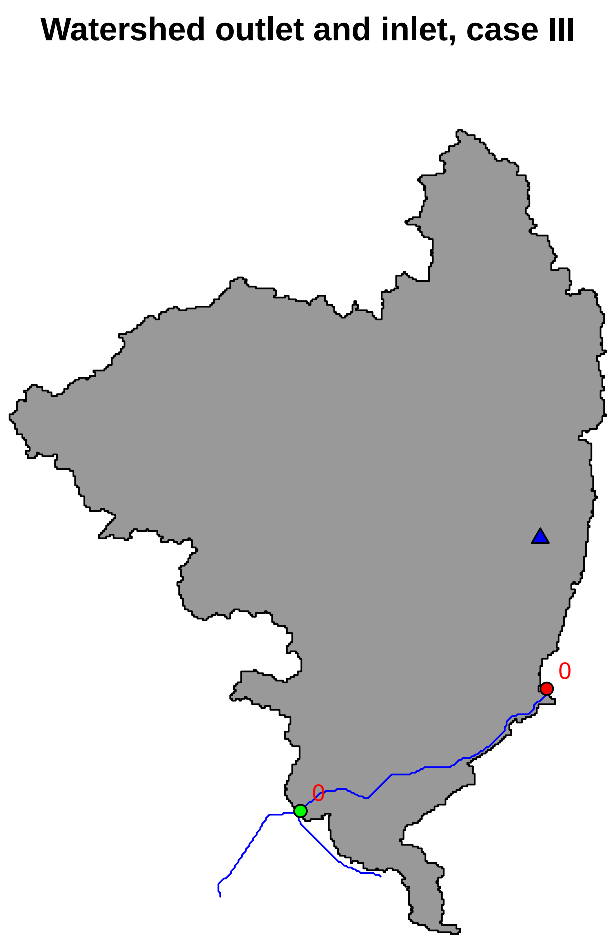


8 The `Watersheds.IOR3` function

The `Watersheds.IOR1` function means: Watershed inlet and outlet nodes: case 3. 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 (length(riverIO)=3).

```
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)

## [1] "end WatershedIO_R3"
## [1] "length(riverIO) == 3"
```

```
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)

## [1] "end WatershedIO_R3"

c1_inlet = a["inlet"][[1]]
c1_inlet

##              coordinates OBJECTID     ID WSO_ID SOURCE LEN_TOM NUM_SEG ELEV
## 122 (4204050, 3338550)    24935 585844      7      N   53043       5    0
##     WINDOW WXSOID      NodID Is_2Keep LENK_FRS LENK_TOM
## 122   2000   <NA> Y000585844       -1        0    53.04

c1_outlet = a["outlet"][[1]]
c1_outlet

##              coordinates OBJECTID     ID WSO_ID SOURCE LEN_TOM NUM_SEG ELEV
## 108 (4217550, 3345250)    24419 584384      7      N   35826       4    0
##     WINDOW WXSOID      NodID Is_2Keep LENK_FRS LENK_TOM
## 108   2000   <NA> Y000584384       -1        0    35.83


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")
```



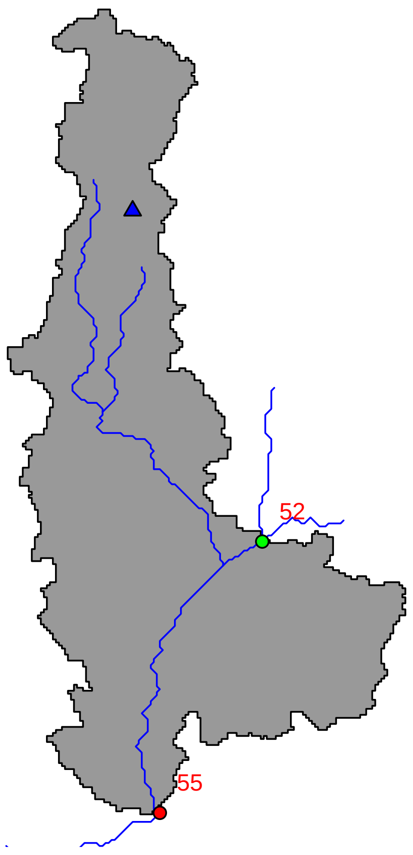
9 The `Watersheds.IOR4` function

The `Watersheds.IOR1` function means Watershed inlet and outlet nodes: case 4. 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 (length(riverIO)=4).

```
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)

## [1] "length(riverIO) == 4"
```

```
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"][[1]]
c1_inlet

##              coordinates OBJECTID     ID WSO_ID SOURCE LEN_TOM NUM_SEG ELEV
## 591 (4362250, 3273550)    30177 601229      7      N  273785      76   52
##     WINDOW WXSOID      NodID Is_2Keep LENK_FRS LENK_TOM
## 591   2000   <NA> Y000601229       -1        0    273.8

c1_outlet = a["outlet"][[1]]
c1_outlet

##              coordinates OBJECTID     ID WSO_ID SOURCE LEN_TOM NUM_SEG ELEV
## 672 (4358850, 3264550)    30739 603332      7      N  261866      74   55
##     WINDOW WXSOID      NodID Is_2Keep LENK_FRS LENK_TOM
## 672   2000   <NA> Y000603332       -1        0    261.9


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 outlet and inlet, case IV



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)

Ihaka, R., & Gentleman, R. (1996). R: a languague for data analysis and graphics. *Journal of Computational and Graphical Statistics*, *5*, 299-314.

Pebesma, E., & Bivand, R. (2005-2012, May). Package "sp": classes and methods for spatial data (.9-99 ed.) [Computer software manual].

R Development Core Team. (2013). *R: A Language and Environment for Statistical Computing.* Vienna, Austria. Retrieved from `http://www.R-project.org/`