Illustrating the eDITH package with two case studies

Luca Carraro*- Florian Altermatt†

Contents

1 Overview **1**

2 Main dependencies **2**

3 The workflow in a nutshell **3**

4 Initialization **4**
4.1 River data 4
4.2 Initialize variables 5

5 Case study 1: Sydenham river **5**
5.1 Watershed delineation 5
5.2 Hydrological characterization 8
5.3 Pinpoiting sampling sites to the river reaches 9
5.4 Apply the eDITH model 12

6 Case study 2: Koide river **17**
6.1 Watershed delineation and hydrological characterization 17
6.2 Pinpoiting sampling sites to the river reaches 20
6.3 Apply the eDITH model 20

7 Displaying species richness **22**

8 Displaying uncertainty in detection probability **23**

References **24**

1 Overview

This document exemplifies with case studies the application of the `eDITH` R package contained in "eDITH: an R-package to spatially project eDNA-based biodiversity across river networks with minimal prior information" by Luca Carraro and Florian Altermatt submitted to *Methods in Ecology and Evolution*.

The two case studies are based on two published spatially replicated fish eDNA metabarcoding datasets across two catchments, the Sydenham river (Canada, 1623 km^2, 30 sampling sites; (Balasingham et al. 2018)) and the Koide river (Japan, 39 km^2, 11 sampling sites; (Sakata et al. 2021)).

Thereby, we demonstrate how spatial extrapolations of the occurrence and density of organisms can be projected based on the eDITH model. We intentionally apply `eDITH` to independent published datasets that were not necessarily collected with such a spatial projection in mind, to showcase the generality of our

*University of Zurich and Eawag, Switzerland. luca.carraro@eawag.ch
†University of Zurich and Eawag, Switzerland.

approach. As exemplified by the two different case studies, from completely different regions and different spatial scales, the herein proposed approach can be applied to any riverine network and species or community eDNA samples when appropriately sampled. Please refer to the accompanying manuscript for a detailed discussion on the applicability of the eDITH model and its underlying assumptions.

2 Main dependencies

The full list of `eDITH`'s dependencies can be found on CRAN.

Among other packages, `eDITH` depends on `rivnet`, which in turn depends on the `traudem` package, which again in turn relies on the TauDEM library. `traudem` provides a guide for correct installation of TauDEM and its dependencies for different operating systems, and offers wrapper commands to call TauDEM methods from R. Please read the `traudem` documentation carefully. To check that `traudem`'s dependencies have been correctly installed, we can run the `traudem::taudem_sitrep` command:

```
traudem::taudem_sitrep()
#> v Found GDAL version GDAL 2.1.0, released 2016/04/25.
#> v Found Microsoft MPI Startup Program [Version 7.1.12437.25] (MPI).
#> v Found TauDEM path (C:/PROGRA~1/TauDEM/TAUDEM~1).
#> v Found TauDEM executables directory (C:/PROGRA~1/TauDEM/TAUDEM~1).
#> v Found all TauDEM executables.
#> i Testing TauDEM on an example file (please wait a bit)...
#> -- TauDEM output ------------------------------------------------------------
#> PitRemove version 5.3.7
#> Input file DEM.tif has projected coordinate system.
#> Processes: 1
#> Header read time: 0.002448
#> Data read time: 0.000580
#> Compute time: 0.003859
#> Write time: 0.003418
#> Total time: 0.010305
#> This run may take on the order of 1 minutes to complete.
#> This estimate is very approximate.
#> Run time is highly uncertain as it depends on the complexity of the input data
#> and speed and memory of the computer. This estimate is based on our testing on
#> a dual quad core Dell Xeon E5405 2.0GHz PC with 16GB RAM.
#> -- End of TauDEM output ----------------------------------------------------
#> v Was able to launch a TauDEM example!
#> ! Double-check above output for serious error messages.
```

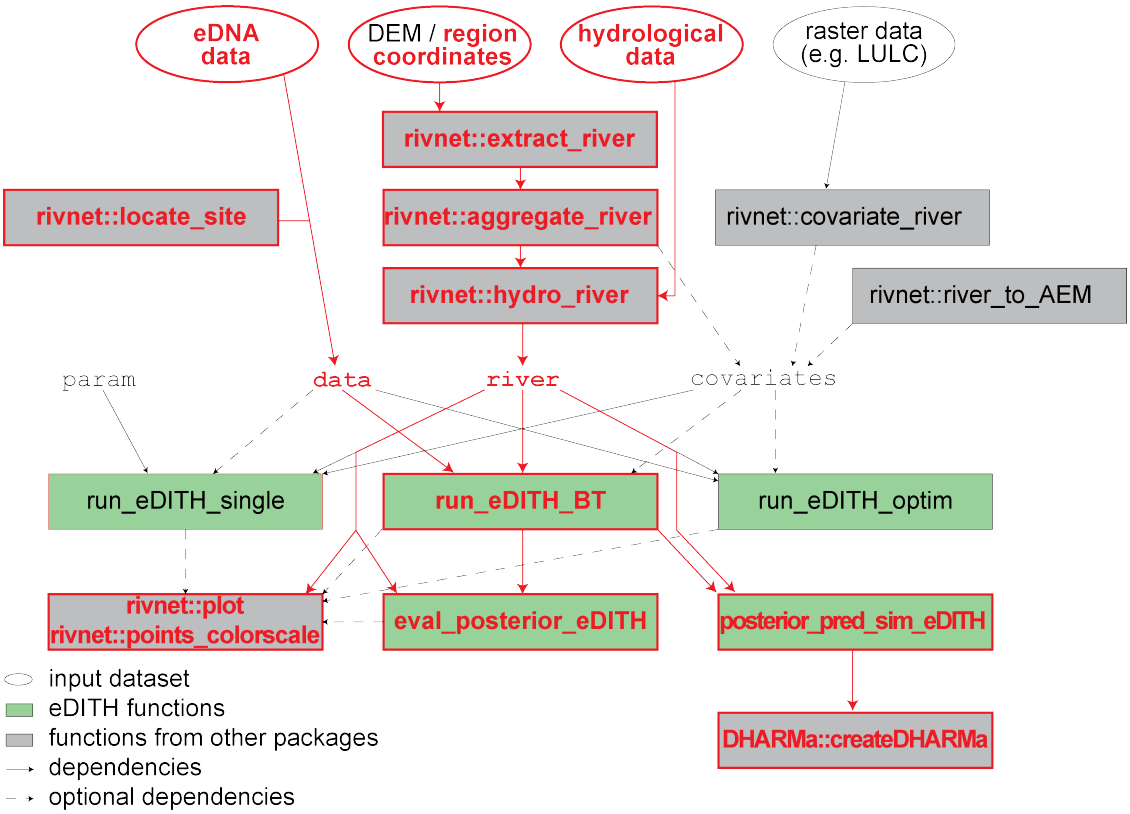
We can now load `eDITH` as well as other packages used in the following examples.

```
library(eDITH)          # main package
#> Registered S3 methods overwritten by 'adegraphics':
#>   method         from
#>   biplot.dudi    ade4
#>   kplot.foucart  ade4
#>   kplot.mcoa     ade4
#>   kplot.mfa      ade4
#>   kplot.pta      ade4
#>   kplot.sepan    ade4
#>   kplot.statis   ade4
#>   scatter.coa    ade4
#>   scatter.dudi   ade4
#>   scatter.nipals ade4
```

```
#>   scatter.pco    ade4
#>   score.acm      ade4
#>   score.mix      ade4
#>   score.pca      ade4
#>   screeplot.dudi ade4
#> Registered S3 method overwritten by 'spdep':
#>   method   from
#>   plot.mst ape
#> Registered S3 methods overwritten by 'adespatial':
#>   method             from
#>   plot.multispati    adegraphics
#>   print.multispati   ade4
#>   summary.multispati ade4
library(rivnet)          # preparation of river object
library(DHARMa)          # posterior predictive checks
#> This is DHARMa 0.4.6. For overview type '?DHARMa'. For recent changes, type news(package = 'DHARMa')
library(BayesianTools)   # tools for Bayesian inference
library(terra)           # tools for spatial analysis and visualization
#> Warning: package 'terra' was built under R version 4.3.2
#> terra 1.7.55
```

3 The workflow in a nutshell

The following flowchart outlines the workflow of `eDITH`. Functions and data types used in the following case studies are highlighted in red. Please refer to the accompanying manuscript (as well as the package documentation) for a detailed description of the package structure.



4 Initialization

4.1 River data

We stored key information on the river morphology and hydraulics in `riverData.csv`. Importantly, we do not need to delineate a watershed with external GIS software, nor provide a Digital Elevation Model (DEM) of the regions of interest. Rather, we use `rivnet` to automatically download open-source DEM data, for which we only need the coordinates of the basins' outlets and of the extent (i.e., lower left and upper right corners) of the DEMs to be downloaded. Internally, DEM data are accessed via a call to the `elevatr` package, on which `rivnet` depends.

```
riverData <- read.csv(file="../data/riverData.csv")
riverData
#>      river X.outlet Y.outlet EPSG   X.min   X.max   Y.min   Y.max  z  thrA
#> 1 Sydenham  2970788  8827521 3085 2960831 3039079 8817103 8907142 10 1e+06
#> 2    Koide  1402942  1143218 6684 1402699 1407461 1142792 1155735 10 1e+05
#>   Q.outlet w.outlet
#> 1     34.4       60
#> 2      1.0       15
```

- `river` provides a tag for the two case studies: `Sydenham` for (Balasingham et al. 2018) and `Koide` for (Sakata et al. 2021).
- `X.outlet`, `Y.outlet` are the outlet coordinates (longitude and latitude) in the projected coordinate system defined by `EPSG`. `EPSG = 3085` corresponds to the NAD83(HARN) / Texas Centric Albers Equal Area, while `EPSG = 6684` corresponds to the JGD2011 / Japan Plane Rectangular CS XVI system. Please refer to this website for an overview of EPSG codes. **It is fundamental that a projected coordinate system (i.e., with coordinate values expressed in meters) is chosen**. In this way, all distances and areas contained in the `river` object produced by `rivnet` will be in m and mˆ2, respectively, which will ensure a correct functioning of `eDITH` functions.
- `X.min`, `X.max`, `Y.min`, `Y.max` are the coordinates (in the system specified by `EPSG`) of the DEM that will be clipped and processed in order to delineate the watershed. **This region must be at least as wide as (or ideally minimally wider than) the extent of the watershed**, such that the complete watershed can be reconstructed. It is thus advised to provide a large range for these values, especially when the shape of the watershed is not exactly known a priori. Obviously, the drawback of providing too large of a range is that the watershed delineation will be slowed down. However, subsequent processes on the extracted watershed (such as the application of eDITH) are independent of the choice of the initial coordinate range, provided that the watershed has been correctly extracted.
- `z` is the zoom level at which the DEM will be downloaded. The exact cell size corresponding to a given zoom level depends on the latitude. Details are provided in the documentation of `elevatr`. `z = 10` corresponds to a cell size of about 50 m at the latitudes of the case study catchments. Increasing `z` by one unit will decrease the cell size by about twofold.
- `thrA` is the threshold area value (in m^2) at which a DEM cell is assumed to be part of the channel network. This value should be small enough such that all headwaters on which eDNA sampling sites were located are contained in the extracted river network; at the same time, too small of a value would generate a river network with too many nodes, which would slow down the execution of eDITH considerably. See also (Carraro and Altermatt 2022). We used a larger `thrA` value for the Sydenham river because it is a larger watershed.
- `Q.outlet` and `w.outlet` are the values of discharge (in $m^3\ s^{-1}$) and river width (in m) at the outlet, respectively. From these values, `rivnet`'s `hydro_river` can extrapolate hydrological variables needed for application of eDITH to all reaches of the river network. For this application, width values were inferred from aerial images (Google Maps). Mean annual discharge was reported in (Balasingham et al. 2018), while no discharge value was provided in (Sakata et al. 2021). Therefore, we inferred the outlet discharge value by assuming that each km^2 of drainage area would contribute a discharge of about 0.025 $m^3 s^{-1}$, which is reasonable in temperate climates.

4.2 Initialize variables

We will use Asymmetric Eigenvector Maps (AEMs) as covariates in the eDITH model. AEMs (Blanchet, Legendre, and Borcard 2008) are mutually orthogonal spatial variables obtained by a spatial filtering technique that considers space in an asymmetric way, and are thus suitable to model species distributions in river networks. The use of AEMs as covariates is convenient because it allows capturing spatial patterns in the data without the need to provide and process raster files to compute geomorphological and/or landcover covariates. See the documentation of `run_eDITH_BT` and `run_eDITH_optim` for examples where such covariates are used.

Specifically, we will only retain the first 10 AEM covariates (sorted in decreasing order by their respective eigenvalue). We will only consider species found in at least 5 sampling sites.

```
n_covariates <- 10 # number of covariates used in eDITH
thr_species <- 5   # exclude species with < thr_species detections from modelling

probDet_median <- probDet_025 <- probDet_975 <- vector("list",2)
signifCovariates <- gelmanDiag <- SST <- speciesID <- vector("list",2)
names(probDet_median) <- names(probDet_025) <- names(probDet_975) <- riverData$river
names(signifCovariates) <- names(gelmanDiag) <- names(SST) <- names(speciesID) <- riverData$river
```

All the following are lists with two objects named `Sydenham` and `Koide`.

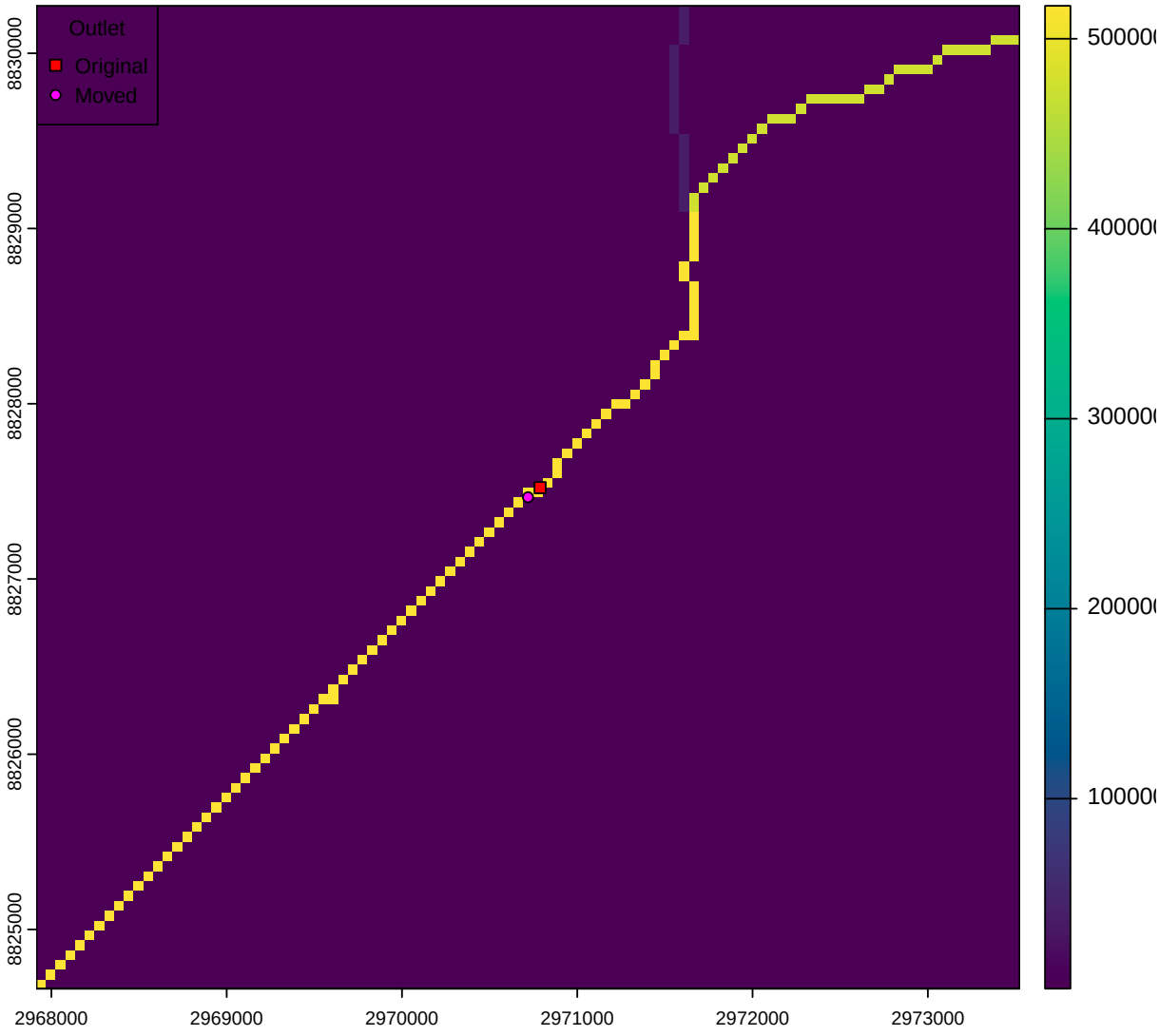
- `probDet_median`, `probDet_025`, `probDet_975`: modelled detection probabilities (median, 0.025- and 0.975-quantiles, respectively) for all species.
- `signifCovariates`: information on which covariates are significant for which species.
- `gelmanDiag`: values of the Gelman's diagnostic (provided by `BayesianTools`).
- `SST`: species-by-site tables as provided by (Balasingham et al. 2018) and (Sakata et al. 2021)
- `speciesID`: indices of species with more than `thr_species` occurrences, which will actually be modelled.

5 Case study 1: Sydenham river

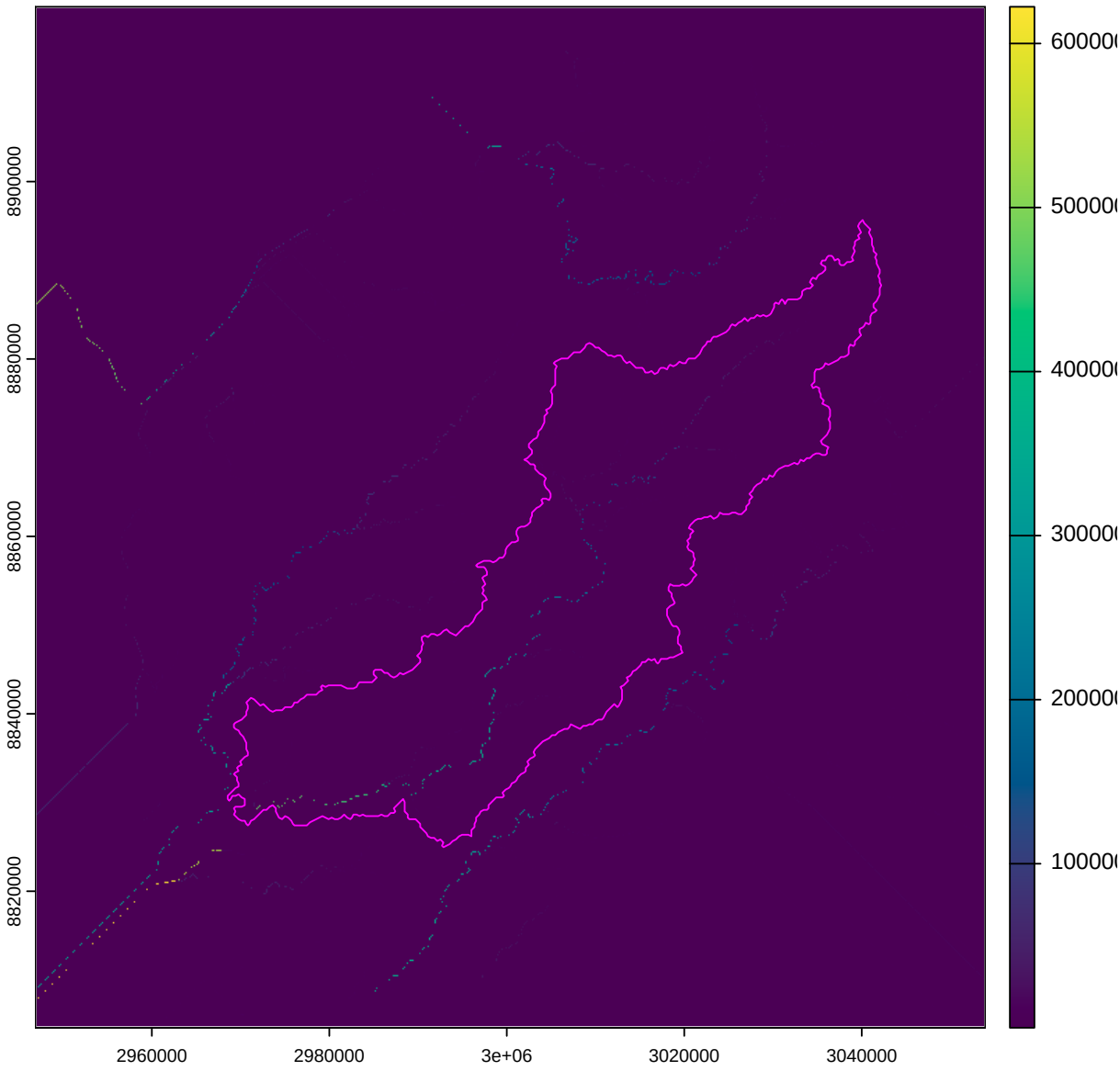
5.1 Watershed delineation

We can now use `rivnet`'s `extract_river` to delineate the Sydenham watershed:

```
Sydenham <- rivnet::extract_river(outlet = c(riverData$X.outlet[1],
                                             riverData$Y.outlet[1]),
                                  EPSG = riverData$EPSG[1],
                                  ext = c(riverData$X.min[1],
                                          riverData$X.max[1],
                                          riverData$Y.min[1],
                                          riverData$Y.max[1]),
                                  z = riverData$z[1],
                                  showPlot = T,
                                  threshold_parameter = 1000,
                                  displayUpdates = 1,
                                  n_processes = 8)
#> Remove pits...
#> D8 flow directions...
#> Contributing areas...
#> Stream definition by threshold...
#> Move outlet to stream...
#> Contributing area upstream of outlet...
```




```
#> Creation of river object...
#> Creation of river object... 100.0%
#> extract_river has finished.
#> Time for DEM download: 8.8 s
#> Time for TauDEM processing: 129.9 s
#> Time for creation of river object: 6.0 s
```




```
# showPlot = T: display figures
# threshold_parameter = 1000: minimum cell number to identify channels
#                             only used to snap the outlet to the river network
# displayUpdates = 1: display updates on console
# n_processes = 8: enable parallel computing with 8 processors
```

Using `showPlot = T` produced two figures, which we can use to identify possible issues in the delineation of a watershed. The first one shows a zoom-in in the proximity of the outlet. The coordinates of the original (i.e., provided by the user) and moved outlet (i.e., snapped to the river network) are shown. Values displayed with

colors are drainage areas (in number of cells). Blue cells are cells not belonging to the river network, while yellow cells identify the river network. **If the river appears white, it means that the corresponding cells have a NaN value of drainage area. This signals that something went wrong in the watershed delineation**. Most probably, the range provided in `ext` was not sufficient to contain the whole catchment. The second figure shows a drainage area map of the whole region (as specified in `ext`), with the derived watershed contour in magenta.

The so-obtained object of `river` class can be aggregated into reaches via `aggregate_river`:

```
Sydenham
#> Class         : river
#> Type          : Real river
#> No. FD nodes  : 516023
#> Dimensions    : 1909 x 2052
#> Cell size     : 56.09
#> Has elevation : TRUE
#> Aggregated    : FALSE
Sydenham <- rivnet::aggregate_river(Sydenham, thrA=riverData$thrA[1],
                                    maxReachLength=1000,
                                    equalizeLengths=TRUE)
Sydenham
#> Class         : river
#> Type          : Real river
#> No. FD nodes  : 516023
#> Dimensions    : 1909 x 2052
#> Cell size     : 56.09
#> Has elevation : TRUE
#> Aggregated    : TRUE
#>    Threshold area  : 1000000.00
#>    Max reach length: 1000.00
#>    No. RN nodes    : 18375
#>    No. AG nodes    : 1715
#>    Has paths       :  FALSE
#>    River geometry  :  FALSE
```

We imposed a maximum reach length of 1000 m, and forced the algorithm to split longer reaches in portions of (approximately) equal length.

```
Sydenham$AG$nNodes
#> [1] 1715
```

The extracted river consists of 1715 nodes at the AG level.

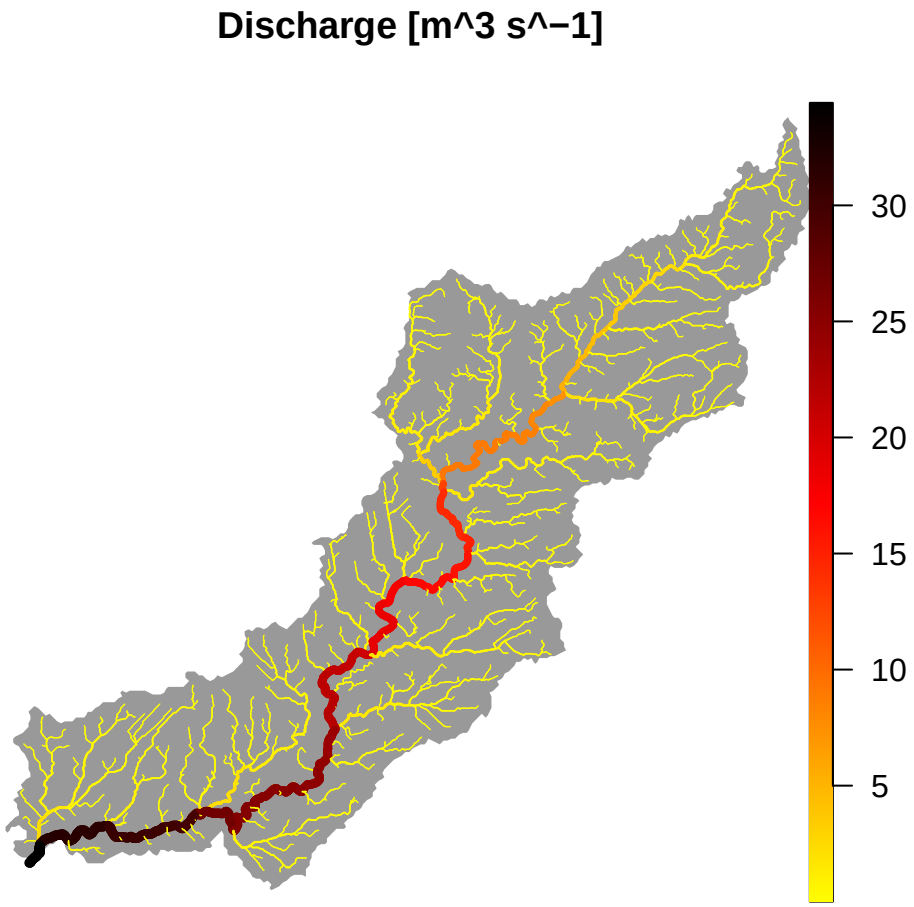
5.2 Hydrological characterization

Function `hydro_river` of `rivnet` can be used to extrapolate hydrological variables across the river network. We first define a data frame containing the necessary data (see `hydro_river`'s documentation for details), and then call `hydro_river`:

```
hd <- data.frame(data=c(riverData$Q.outlet[1], riverData$w.outlet[1]),
                 type=c("Q", "w"),
                 node=c(1,1)*Sydenham$AG$outlet)
Sydenham <- hydro_river(hd, Sydenham)
```

Extrapolated discharge values across the river network can be displayed via `plot()`:

```
plot(Sydenham$AG$discharge, Sydenham)
title('Discharge [m^3 s^-1]')
```



5.3 Pinpoiting sampling sites to the river reaches

Sampling sites' coordinates are contained in `samplingSitesSydenham.csv`. We can use `locate_site` from `rivnet` to identify the AG nodes (i.e., reaches) associated to the sampling sites.

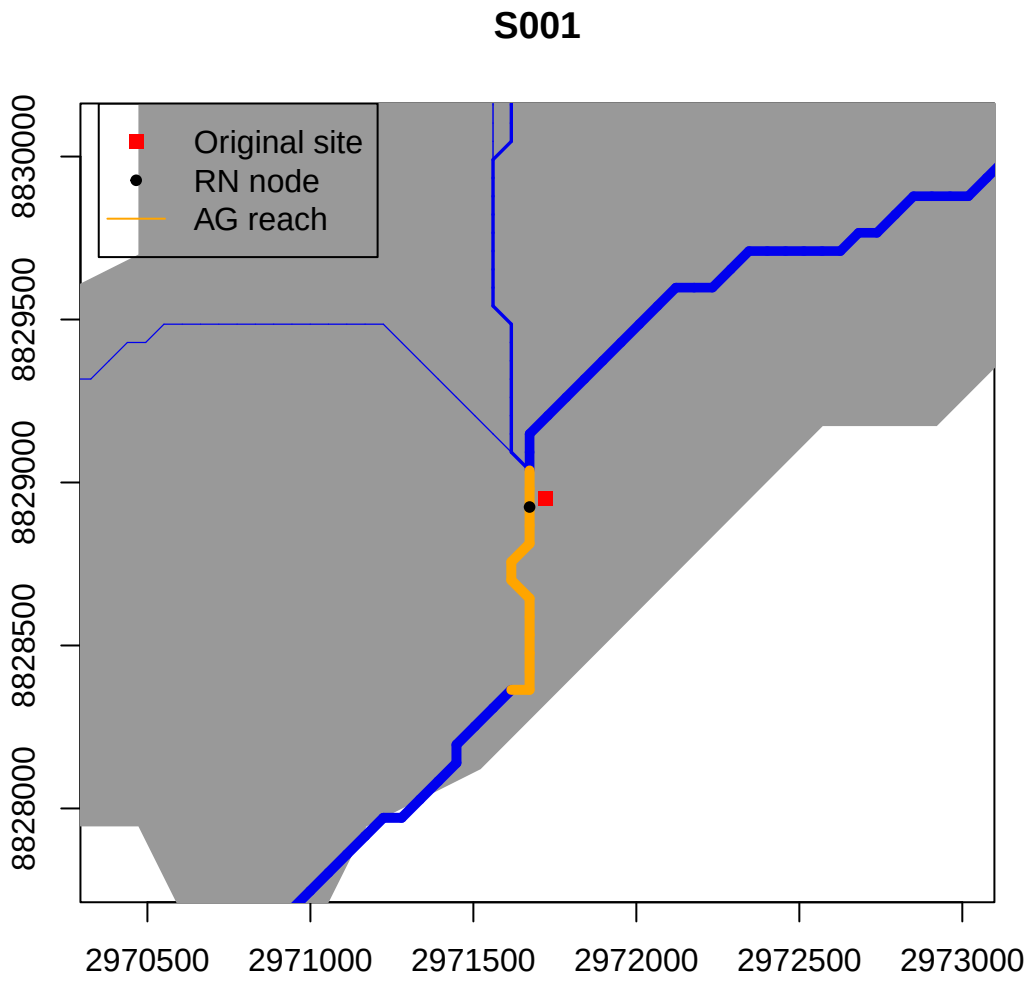
```
samplingSitesSydenham <- read.csv(file="../data/samplingSitesSydenham.csv")

AGnode <- numeric(length(samplingSitesSydenham$X))
for (j in 1: length(samplingSitesSydenham$X)){
  tmp <- rivnet::locate_site(samplingSitesSydenham$X[j],
                             samplingSitesSydenham$Y[j],
                             Sydenham,
                             showPlot = F)
  AGnode[j] <- tmp$AGnode
```

```
}
```

In this case, we automatically assigned the closest reach as the crow flies to each sampling site. However, it can happen that, due to discrepancies between the extracted river network and the actual river network, such an automatic assignment would pinpoint a sampling site to a wrong reach. To have a better control over this operation, one can use `showPlot = TRUE` in `locate_site`. For instance, for the first sampling site:

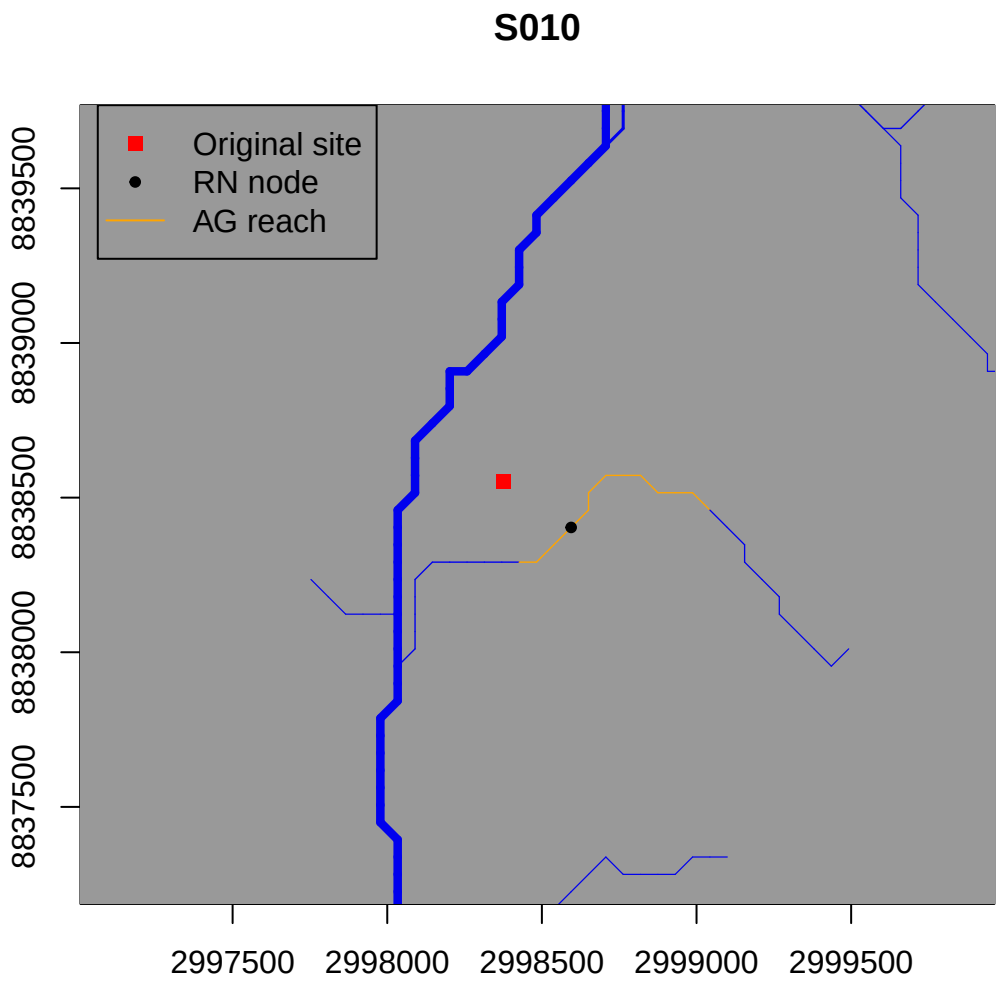
```
rivnet::locate_site(samplingSitesSydenham$X[1],
                    samplingSitesSydenham$Y[1],
                    Sydenham,
                    showPlot = T)
title(samplingSitesSydenham$siteID[1])
```



In this case, the reach assignation was correct. Conversely, site 10 is wrongly assigned to a tributary, instead of the main stem:

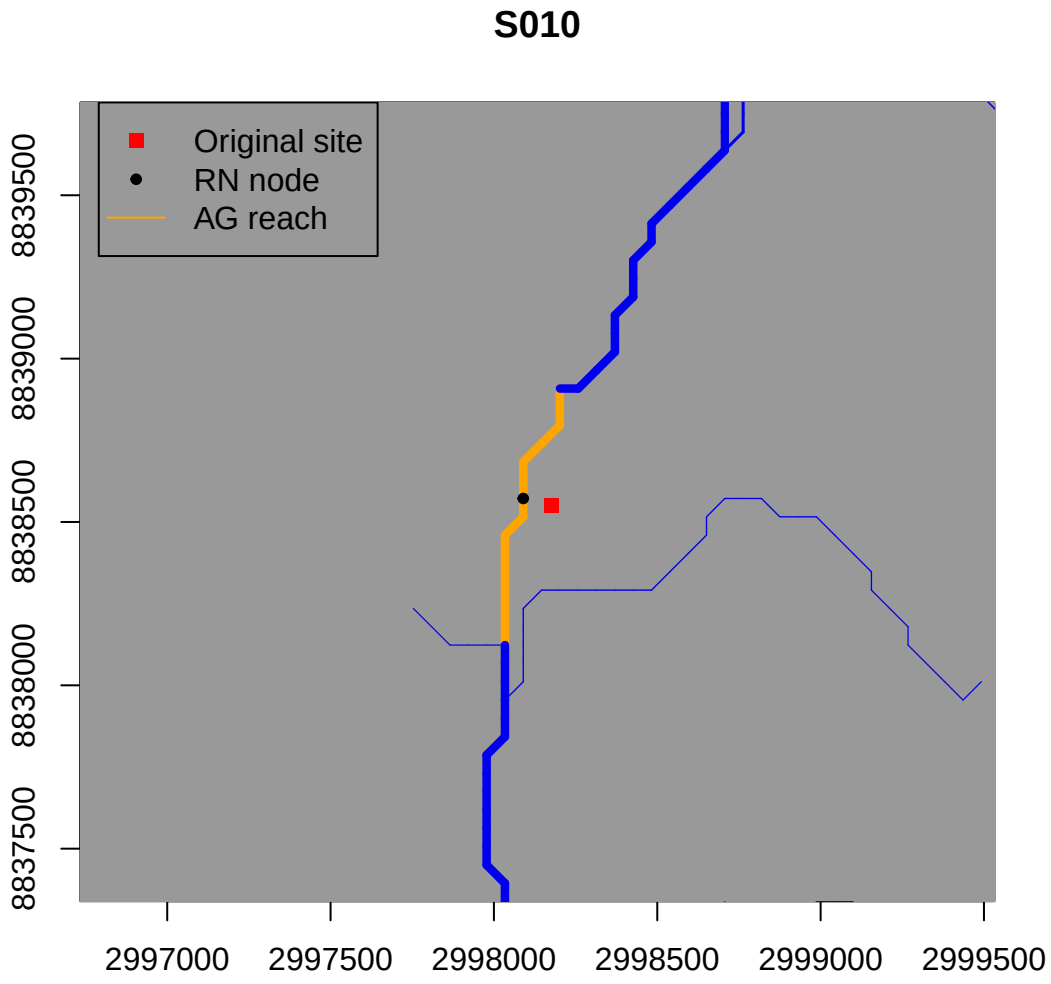
```
rivnet::locate_site(samplingSitesSydenham$X[10],
                    samplingSitesSydenham$Y[10],
```

```
                    Sydenham,
                    showPlot = T)
title(samplingSitesSydenham$siteID[10])
```



Since the topology of the extracted river network cannot be easily changed, the fastest alternative is to tweak the coordinates of the sampling sites so that they are attributed to the correct reach. In this case, we slightly shift site 10 westwards to make it snap to the main stem:

```
samplingSitesSydenham$X[10] <- samplingSitesSydenham$X[10] - 200
tmp <- rivnet::locate_site(samplingSitesSydenham$X[10],
                           samplingSitesSydenham$Y[10],
                           Sydenham,
                           showPlot = T)
title(samplingSitesSydenham$siteID[10])
```




```
AGnode[10] <- tmp$AGnode
```

All other sampling sites were snapped to the correct reach. We can now store these data:

```
samplingSitesSydenham[["AGnode"]] <- AGnode
```

5.4 Apply the eDITH model

5.4.1 Initialization

Read counts are contained in `siteSpeciesTableSydenham.csv`. The following are the species with at least `thr_species` detections:

```
# read species-by-site table
SST$Sydenham <- read.csv("../data/siteSpeciesTableSydenham.csv")
SST$Sydenham[is.na(SST$Sydenham)] <- 0 # change NAs to zeros
# choose species with at least thr_species presences
speciesID$Sydenham <- which(rowSums(SST$Sydenham[,-1]!=0) >= thr_species)
SST$Sydenham[speciesID$Sydenham,1]
```

```
#>  [1] "Esox lucius"               "Cottus cognatus"
#>  [3] "Cottus bairdii"            "Percina maculata"
#>  [5] "Noturus stigmosus"         "Noturus flavus"
#>  [7] "Nocomis biguttatus"        "Etheostoma blennioides"
#>  [9] "Semotilus atromaculatus"   "Oncorhynchus tshawytscha"
#> [11] "Pimephales notatus"        "Ambloplites rupestris"
#> [13] "Culaea inconstans"         "Etheostoma nigrum"
#> [15] "Ictalurus punctatus"       "Hypentelium nigricans"
#> [17] "Luxilus cornutus"          "Catostomus commersonii"
#> [19] "Moxostoma anisurum"        "Moxostoma erythrurum"
#> [21] "Cyprinus carpio"           "Neogobius melanostomus"
```

We now initialize the export variables for the Sydenham river.

```
# initialize variables for export
probDet_median$Sydenham <- data.frame(matrix(0, Sydenham$AG$nNodes,
                                      length(speciesID$Sydenham)))
probDet_025$Sydenham <- probDet_975$Sydenham <-probDet_median$Sydenham
names(probDet_median$Sydenham) <- SST$Sydenham[speciesID[[1]],1]
names(probDet_025$Sydenham) <- SST$Sydenham[speciesID[[1]],1]
names(probDet_975$Sydenham) <- SST$Sydenham[speciesID[[1]],1]

signifCovariates$Sydenham <- data.frame(matrix(0,n_covariates,
                                               length(speciesID$Sydenham)))
names(signifCovariates$Sydenham) <- SST$Sydenham[speciesID$Sydenham,1]

row.names(signifCovariates$Sydenham) <- paste0("beta_AEM",1:n_covariates)
gelmanDiag$Sydenham <- data.frame(matrix(0,n_covariates+3,
                                         length(speciesID$Sydenham)))

names(gelmanDiag$Sydenham) <- SST$Sydenham[speciesID$Sydenham,1]
row.names(gelmanDiag$Sydenham) <- c("tau","log_p0",
                                    paste0("beta_AEM",1:n_covariates),"omega")
```

In the last command, we manually included the names of the model parameters. The first two parameters `tau`and `log_p0` refer to the decay time and the logarithm in base 10 of the baseline production rate, respectively. Given that we will use the first 10 AEMs as covariates, parameters for the covariate effect sizes are termed `beta_AEM1`, ..., `beta_AEM10`. Finally, `omega` is the overdispersion parameter used to control the variance of the negative binomial distribution that we will use to model errors between observed and modelled read counts.

5.4.2 Run eDITH for a single species

We are now ready to run the eDITH model. As an example, let's select the 3rd species:

```
iS <- 3
indSp <- speciesID$Sydenham[iS]
nam <- SST$Sydenham[indSp,1]
nam
#> [1] "Cottus bairdii"
```

We create the `data` data frame containing IDs of the sampling sites and the respective eDNA values:

```
values <- as.numeric(SST$Sydenham[indSp,-1])
# final "-1" to get rid of the species name
# as.numeric() to only deal with numeric values
```

```
data <- data.frame(ID=samplingSitesSydenham$AGnode, values=values)
```

We can now call `run_eDITH_BT`. We do not specify any `covariates`, since we are going to use the first 10 AEMs for this purpose. Specifically, we will use exponential weights to define the AEMs (see documentation of `OCN_to_AEM` of package `OCNet` for details). We define a negative binomial error distribution (`ll.type = "nbinom"`). We use default settings with respect to prior distributions and options for the MCMC sampler. Depending on the processor used, this step should take about 1 hour to complete.

```
out <- run_eDITH_BT(data, Sydenham, ll.type="nbinom",
                    n.AEM=n_covariates, par.AEM=list(weight="exponential"),
                    verbose=T)
```

5.4.2.1 Analyze eDITH output

After `run_eDITH_BT` has completed, we now use `eval_posterior_eDITH` to evaluate relevant quantiles from the posterior distribution. Specifically, we are interested in the posterior median and in the 2.5th-97.5th percentiles:

```
out <- eval_posterior_eDITH(out, Sydenham, quant=c(0.025, 0.5, 0.975))
```

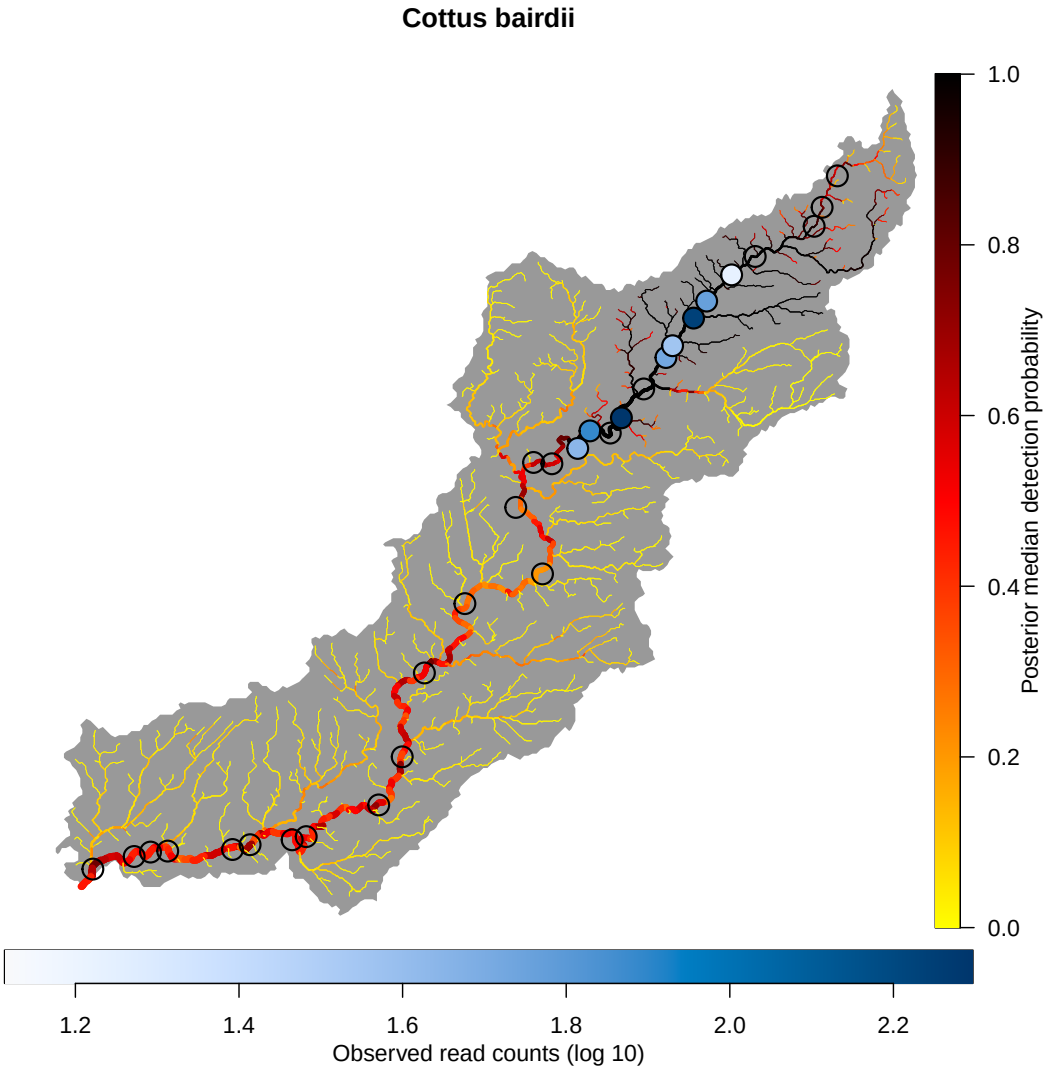
Let's now run posterior predictive simulations via `posterior_pred_sim_eDITH`:

```
pps <- posterior_pred_sim_eDITH(out, Sydenham)
```

We can display the posterior median detection probability with `plot`. This internally calls `draw_thematic_OCN` from `OCNet`, hence see this function's documentation for a more complete control of its functioning. For instance, we can use `args_imagePlot` to pass the colorbar's label.

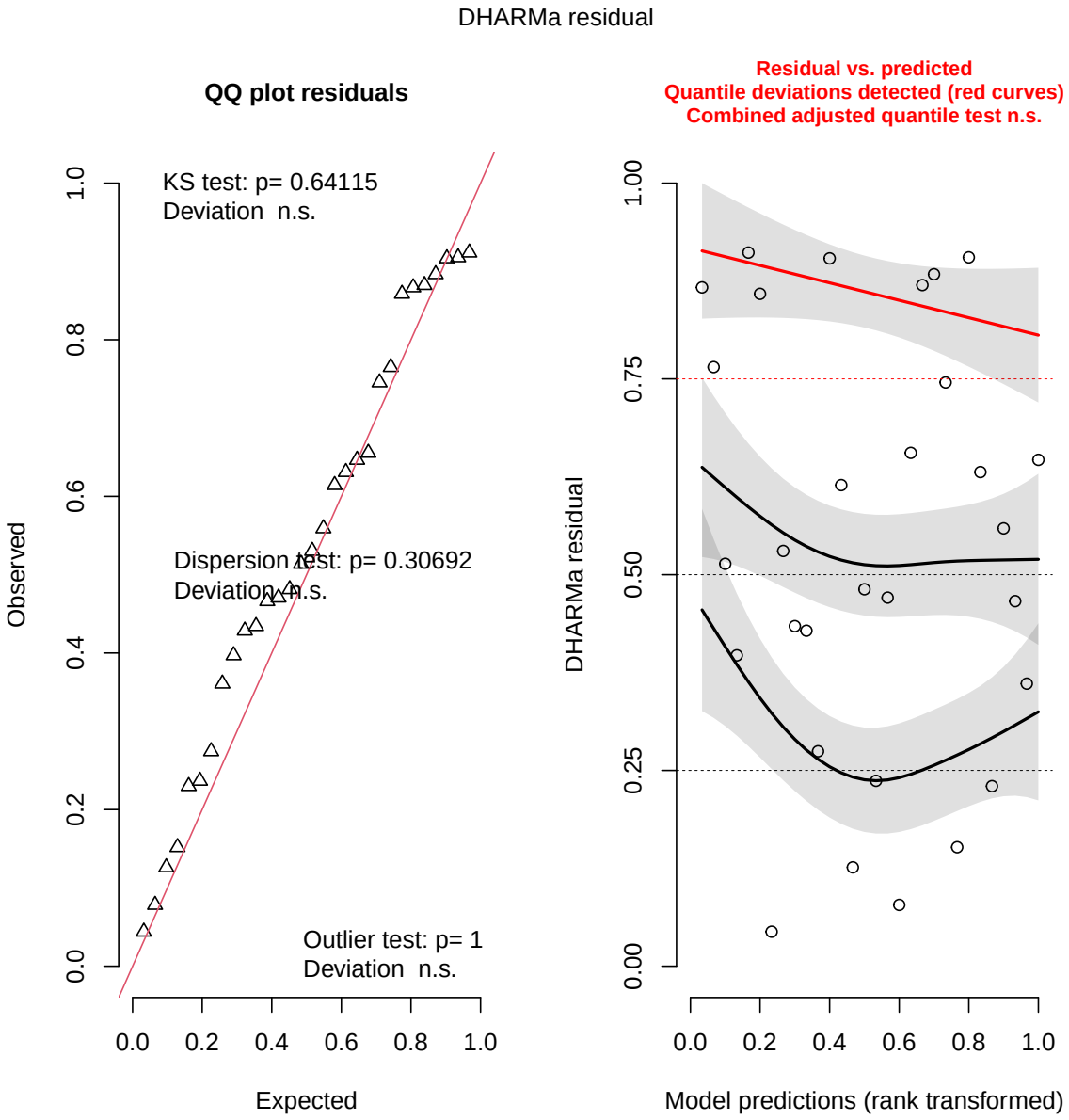
On top of the river map, we can draw the observed read counts in order to perform a visual check of the soundness of the model fitting. We can do so via `rivnet`'s `points_colorscale`. We can use option `force.range = FALSE` to show non-detections (i.e., `-Inf` values, when the logarithm of observed read numbers is taken) with a transparent background:

```
plot(out$probDetection_quantile[2,], Sydenham,
     args_imagePlot=list(legend.lab="Posterior median detection probability"))
title(nam)
rivnet::points_colorscale(samplingSitesSydenham$X, samplingSitesSydenham$Y,
                          log10(out$data$values),
                          force.range = FALSE,
                          bg.palette=hcl.colors(1000, "Blues 3", rev=T),
                          horizontal=T,
                          legend.lab = "Observed read counts (log 10)")
```



Finally, we can use the posterior predictive simulations `pps` as input in function `createDHARMa` from `DHARMa` to explore the goodness of fit of our model. Please refer to the `DHARMa` documentation for an interpretation of the diagnostics plot.

```
out.sim <- DHARMa::createDHARMa(pps, out$data$values)
#> No fitted predicted response provided, using the mean of the simulations
plot(out.sim)
```



5.4.3 Run eDITH for all species

The following code loops through all species found (i.e., those with at least `thr_species` detections) and runs the eDITH model for each of them. For each species, we export the posterior median detection probability (in `probDet`), Gelman diagnostics values (`gelmanDiag`) and information on significant covariates (`signifCovariates`).

- Gelman diagnostics are calculated within `BayesianTools`, and provide a metric to assess whether the Markov chains have converged (see the documentation of `BayesianTools`' `gelmanDiagnostics` for more details).
- Significant covariates are those for which the equal-tailed 95% of the corresponding posterior effect size does not overlap 0. If for a given species a certain covariate is significantly positive (i.e., the equal-tailed 95% of the posterior distribution of the corresponding effect size is positive), the corresponding entry of `signifCovariates` will contain a `1`. Negatively significant covariates are marked with `-1`.

Non-significant covariates are marked with 0.

```
for (iS in 1:length(speciesID$Sydenham)){ # loop over species
  indSp <- speciesID$Sydenham[iS]
  nam <- SST$Sydenham[indSp,1]
  fnam <- paste0('../resultsSydenham/',nam,'.rda')

  if(!file.exists(fnam)){ # if eDITH results were already generated, skip
  values <- as.numeric(SST$Sydenham[indSp,-1])
  data <- data.frame(ID=samplingSitesSydenham$AGnode, values=values)
  # for the species at hand
  # run eDITH model via BayesianTools
  out <- run_eDITH_BT(data, Sydenham, ll.type="nbinom",
                      n.AEM=n_covariates, par.AEM=list(weight="exponential"),
                      verbose=T)
  # evaluate quantiles from posterior distribution
  out <- eval_posterior_eDITH(out, Sydenham, quant=c(0.025,0.5, 0.975))
  # run posterior predictive simulations
  pps <- posterior_pred_sim_eDITH(out, Sydenham)

  } else {load(fnam)} # if results already generated, load them

  # store posterior detection probability (median, 0.025- and 0.975-quantiles),
  #   Gelman diagnostics and significance of covariates
  probDet_median$Sydenham[,iS] <- out$probDetection_quantile[2,]
  probDet_025$Sydenham[,iS] <- out$probDetection_quantile[1,]
  probDet_975$Sydenham[,iS] <- out$probDetection_quantile[3,]
  gelmanDiag$Sydenham[,iS] <- out$gD$psrf[,1]
  for (j in 1:length(out$covariates)){
    if (out$cI[1,j+2]>0) signifCovariates$Sydenham[j,iS] <- 1
    if (out$cI[2,j+2]<0) signifCovariates$Sydenham[j,iS] <- -1
  }
}
```

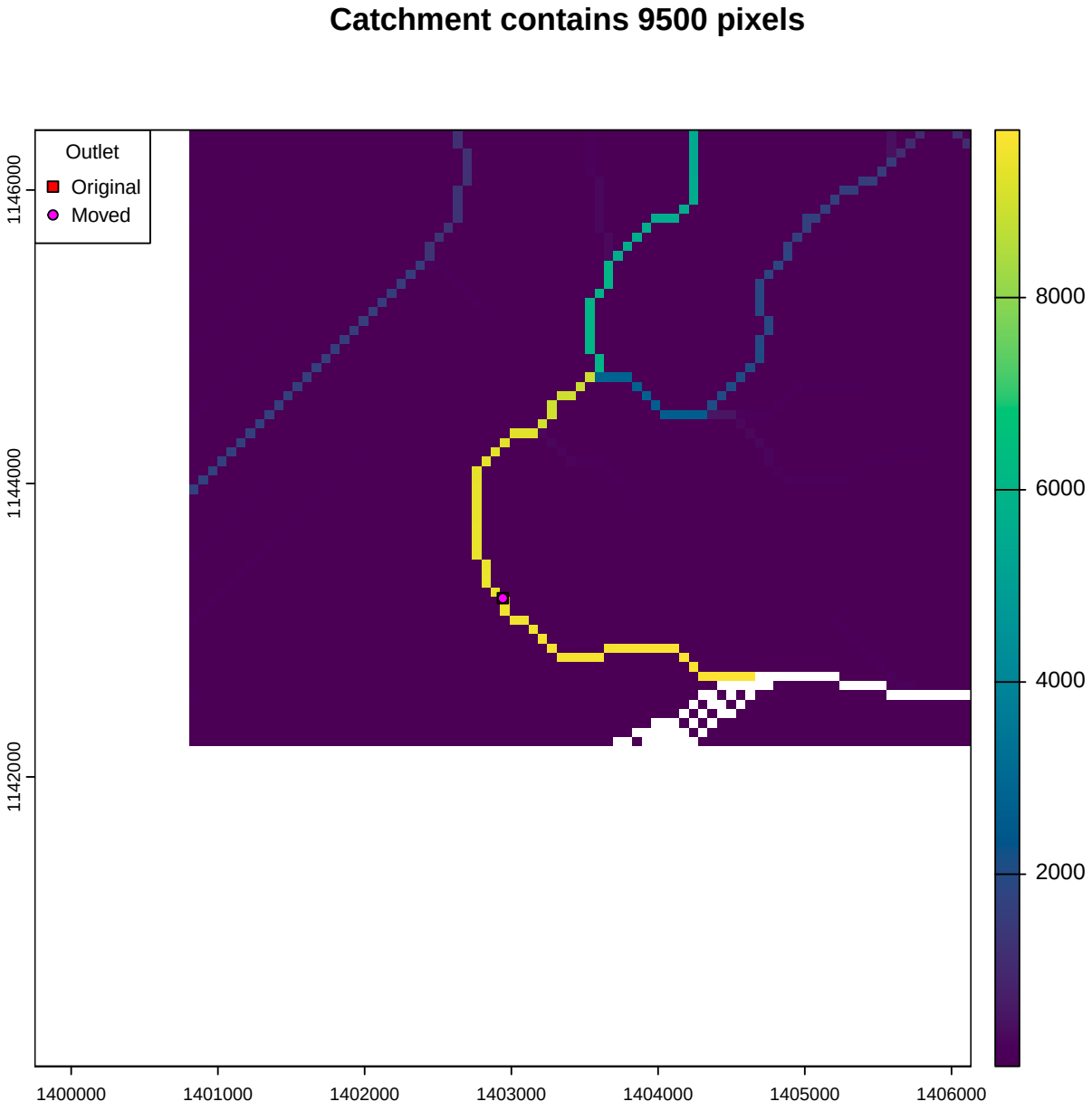
6 Case study 2: Koide river

In a similar fashion, we can apply the eDITH model to the dataset of (Sakata et al. 2021).

6.1 Watershed delineation and hydrological characterization

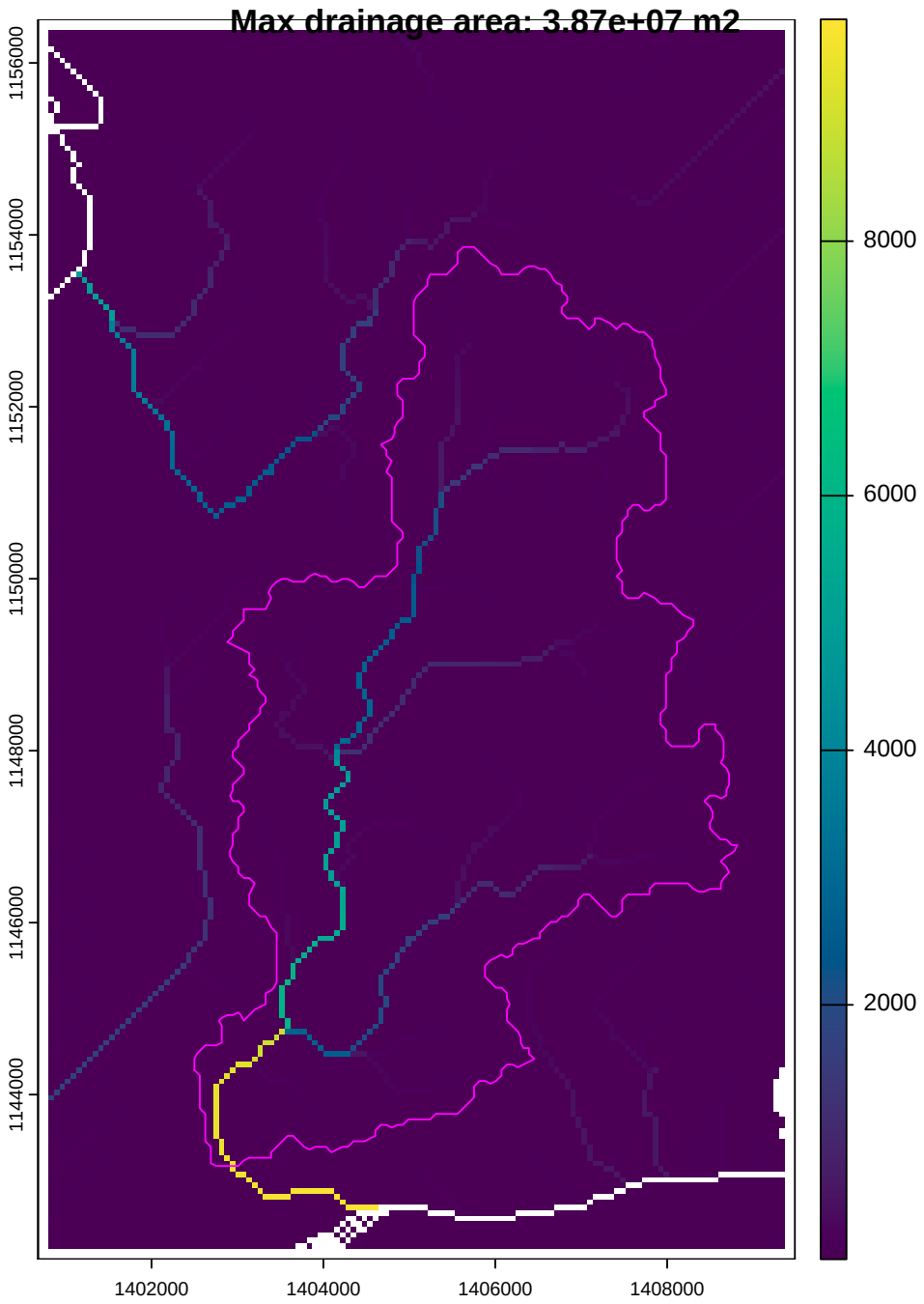
```
Koide <- rivnet::extract_river(outlet = c(riverData$X.outlet[2],
                                          riverData$Y.outlet[2]),
                               EPSG = riverData$EPSG[2],
                               ext = c(riverData$X.min[2],
                                       riverData$X.max[2],
                                       riverData$Y.min[2],
                                       riverData$Y.max[2]),
                               z = riverData$z[2],
                               showPlot = T,
                               threshold_parameter = 1000,
                               displayUpdates = 1,
                               n_processes = 8)
#> Remove pits...
```

```
#> D8 flow directions...
#> Contributing areas...
#> Stream definition by threshold...
#> Move outlet to stream...
#> Contributing area upstream of outlet...
```




```
#> Creation of river object...
#> Creation of river object... 100.0%
```

```
#> extract_river has finished.
#> Time for DEM download: 1.7 s
#> Time for TauDEM processing: 1.6 s
#> Time for creation of river object: 0.1 s
```




```
# aggregate river
Koide <- rivnet::aggregate_river(Koide, thrA=riverData$thrA[2],
                                 maxReachLength=1000,
                                 equalizeLengths=TRUE)
```

```
# create data frame with hydrological variables
hd <- data.frame(data=c(riverData$Q.outlet[2], riverData$w.outlet[2]),
                 type=c("Q", "w"),
                 node=c(1,1)*Koide$AG$outlet)
# extrapolate hydrological variables
Koide <- hydro_river(hd, Koide)
```

6.2 Pinpoiting sampling sites to the river reaches

In this case, all sampling sites are attributed to the correct reach by `locate_site`.

```
samplingSitesKoide <- read.csv(file="../data/samplingSitesKoide.csv")

AGnode <- numeric(length(samplingSitesKoide$X))
for (j in 1: length(samplingSitesKoide$X)){
  tmp <- rivnet::locate_site(samplingSitesKoide$X[j],
                             samplingSitesKoide$Y[j],
                             Koide,
                             showPlot = F)
  AGnode[j] <- tmp$AGnode
}
samplingSitesKoide[["AGnode"]] <- AGnode
```

6.3 Apply the eDITH model

6.3.1 Initialization

Read counts are contained in `siteSpeciesTableKoide.csv`. The following are the species with at least `thr_species` detections:

```
 # read species-by-site table
  SST$Koide <- read.csv("../data/siteSpeciesTableKoide.csv")
  SST$Koide[is.na(SST$Koide)] <- 0
  speciesID$Koide <- which(rowSums(SST$Koide[,-1]!=0) >= thr_species)
  SST$Koide[speciesID$Koide,1]
#>  [1] "Anguilla japonica"
#>  [2] "Carassius spp."
#>  [3] "Cyprinus carpio"
#>  [4] "Hemibarbus spp."
#>  [5] "Pseudogobio esocinus"
#>  [6] "Pseudorasbora parva"
#>  [7] "Rhynchocypris lagowskii steindachi neri"
#>  [8] "Tribolodon brandtii maruta"
#>  [9] "Tribolodon hakonensis"
#> [10] "Opsariichthys platypus"
#> [11] "Misgurnus anguillicaudatus"
#> [12] "Paramisgurnus dabryanus"
#> [13] "Silurus asotus"
#> [14] "Plecoglossus altivelis"
#> [15] "Mugil cephalus"
#> [16] "Oryzias latipes"
#> [17] "Lateolabrax japonicus"
#> [18] "Eleotris oxycephala"
#> [19] "Gymnogobius petschiliensis"
#> [20] "Rhinogobius giurinus"
```

```
#> [21] "Rhinogobius spp."
#> [22] "Tridentiger spp."
#> [23] "Channa argus"
```

```
  # initialize variables for export
  probDet_median$Koide <- data.frame(matrix(0,Koide$AG$nNodes,
                                    length(speciesID$Koide)))
  probDet_025$Koide <- probDet_975$Koide <- probDet_median$Koide
  names(probDet_median$Koide) <- SST$Koide[speciesID$Koide,1]
  names(probDet_025$Koide) <- SST$Koide[speciesID$Koide,1]
  names(probDet_975$Koide) <- SST$Koide[speciesID$Koide,1]

  signifCovariates$Koide <- data.frame(matrix(0,n_covariates,
                                              length(speciesID$Koide)))
  names(signifCovariates$Koide) <- SST$Koide[speciesID$Koide,1]

  row.names(signifCovariates$Koide) <- paste0("beta_AEM",1:n_covariates)
  gelmanDiag$Koide <- data.frame(matrix(0,n_covariates+3,
                                        length(speciesID$Koide)))

  names(gelmanDiag$Koide) <- SST$Koide[speciesID$Koide,1]
  row.names(gelmanDiag$Koide) <- c("tau","log_p0",
                                   paste0("beta_AEM",1:n_covariates),"omega")
```

6.3.2 Run eDITH for all species

```
for (iS in 1:length(speciesID$Koide)){ # loop over species
  indSp <- speciesID$Koide[iS]
  nam <- SST$Koide[indSp,1]
  fnam <- paste0('../resultsKoide/',nam,'.rda')

  if(!file.exists(fnam)){ # if eDITH results were already generated, skip
  values <- as.numeric(SST$Koide[indSp,-1])
  data <- data.frame(ID=samplingSitesKoide$AGnode, values=values)
  # for the species at hand
  # run eDITH model via BayesianTools
  out <- run_eDITH_BT(data, Koide, ll.type="nbinom",
                      n.AEM=n_covariates, par.AEM=list(weight="exponential"),
                      verbose=T)
  # evaluate quantiles from posterior distribution
  out <- eval_posterior_eDITH(out, Koide, quant=c(0.025,0.5, 0.975))
  # run posterior predictive simulations
  pps <- posterior_pred_sim_eDITH(out, Koide)

  } else {load(fnam)} # if results already generated, load them

  # store posterior median detection probability,
  #    Gelman diagnostics and significance of covariates
  probDet_median$Koide[,iS] <- out$probDetection_quantile[2,]
  probDet_025$Koide[,iS] <- out$probDetection_quantile[1,]
  probDet_975$Koide[,iS] <- out$probDetection_quantile[3,]
  gelmanDiag$Koide[,iS] <- out$gD$psrf[,1]
  for (j in 1:length(out$covariates)){
```

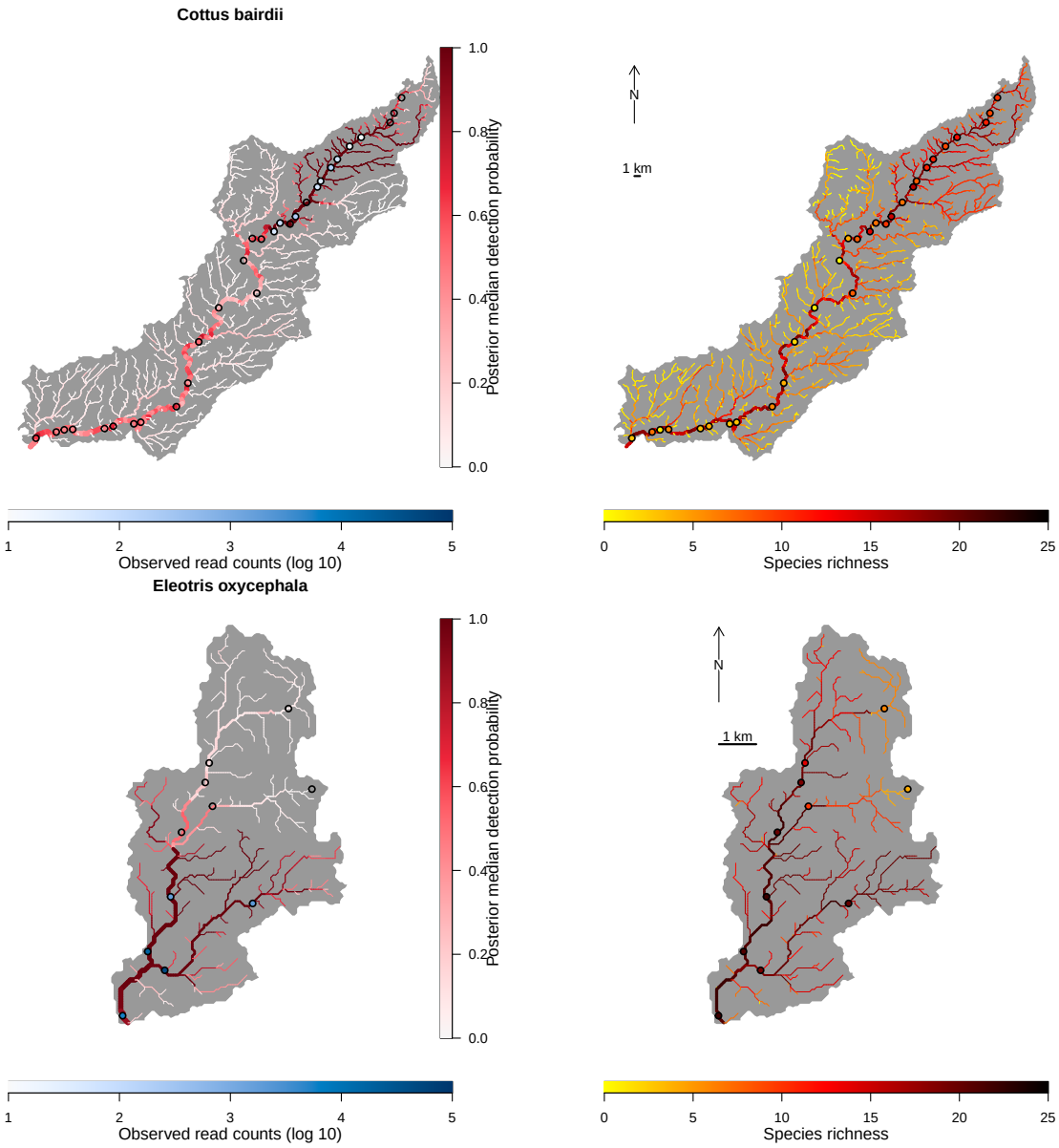
```
    if (out$cI[1,j+2]>0) signifCovariates$Koide[j,iS] <- 1
    if (out$cI[2,j+2]<0) signifCovariates$Koide[j,iS] <- -1
  }
}
```

7 Displaying species richness

Species richness predicted by eDITH can be computed by stacking up presence/absence predictions for all species contained in the datasets; in turn, these are derived by using a threhsold of 0.5 on the posterior median detection probability. We visually compare the predicted species richness with the observed one at the sampling sites, which we show as colored points in the river map. We also display the modelled detection probability for one characteristic species for each case study, together with the respective observed read counts. This figure corresponds to Fig. 4 shown in the manuscript.

```
sp_name <- c("Cottus bairdii","Eleotris oxycephala")
deu <- colorRampPalette(c("yellow","red","black"))
north_length <- c(5000,1000)
par(mfrow=c(2,2), oma=c(3,0,0,0))
for (i in 1:length(riverData$river)) {
  ID <- riverData$river[i]
  eval(parse(text=paste0('river <- ',ID)))
  eval(parse(text=paste0('samplingSites <- samplingSites',ID)))
  plot(probDet_median[[i]][[sp_name[i]]],  river,
       colPalette=hcl.colors(1000, "Reds 3", rev=T), colLevels = c(0,1),
       args_imagePlot=list(legend.lab="Posterior median detection probability"))
  values <- SST[[i]][which(SST[[i]][,1]==sp_name[i]),-1]
  points_colorscale(samplingSites$X, samplingSites$Y, log10(as.numeric(values)),
                    cex=1, force.range = FALSE,
                    bg.palette=hcl.colors(1000, "Blues 3", rev=T),
                    bg.range=c(1,5),
                    horizontal=T, legend.lab = "Observed read counts (log 10)")
  title(sp_name[i])

  plot(rowSums(probDet_median[[i]]>0.5), river, addLegend=F, max_lwd=3,
       colLevels=c(0,25))
  terra::sbar(d = 1000,
              xy = c(min(river$FD$X)+0.05*(max(river$FD$X)-min(river$FD$X)),
                     min(river$FD$Y)+0.7*(max(river$FD$Y)-min(river$FD$Y))),
              label = "1 km")
  terra::north(d = north_length[i],
               xy = c(min(river$FD$X)+0.05*(max(river$FD$X)-min(river$FD$X)),
                      min(river$FD$Y)+0.9*(max(river$FD$Y)-min(river$FD$Y))))
  rivnet::points_colorscale(samplingSites$X, samplingSites$Y,
                            colSums(SST[[i]][speciesID[[i]],-1]>0),
                    bg.palette = deu(1000),bg.range=c(0,25),
                    cex=1, horizontal = T, legend.lab = "Species richness")
}
```



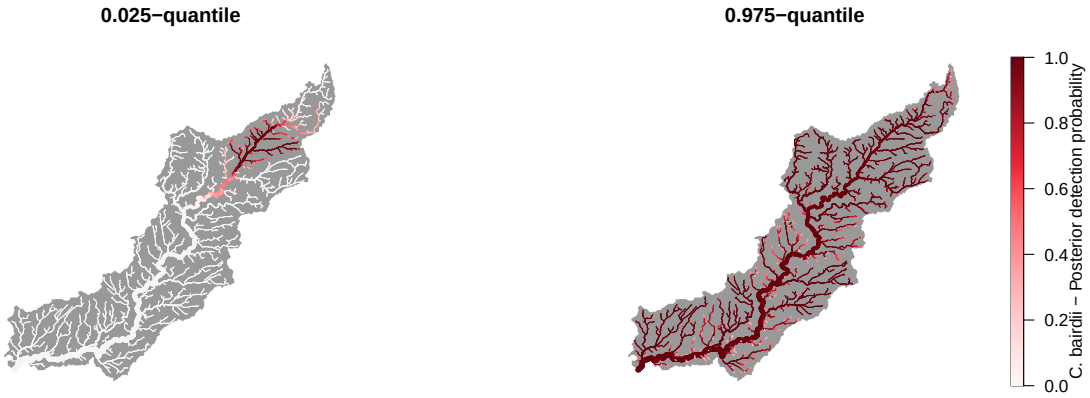
Note the use of `sbar` and `north` from the `terra` package in order to display the scale bar and north arrow, respectively. For both functions, argument `xy` is mandatory (see also documentation of `rivnet`'s `plot`). We here specify `xy` as a function of the range of the river's coordinates.

8 Displaying uncertainty in detection probability

Finally, we can use output from `eval_posterior_eDITH` to assess the uncertainty in the estimates of detection probabilities. By default, `eval_posterior_eDITH` evaluates the equal-tailed 95% interval of the posterior distributions, which we previously stored in `probDet_025` and `probDet_975`. Here follows an example for *Cottus bairdii* in the Sydenham river.

```
par(mfrow=c(1,2))
plot(probDet_025[[1]][[sp_name[1]]],  Sydenham,
      colPalette=hcl.colors(1000, "Reds 3", rev=T), colLevels = c(0,1),
```

```
      addLegend = FALSE)
title("0.025-quantile")
plot(probDet_975[[1]][[sp_name[1]]],  Sydenham,
     colPalette=hcl.colors(1000, "Reds 3", rev=T), colLevels = c(0,1),
     args_imagePlot=list(legend.lab="C. bairdii - Posterior detection probability"))
title("0.975-quantile")
```



References

Balasingham, K. D., R. P. Walter, N. E. Mandra, and D. D. Heath. 2018. "Environmental DNA Detection of Rare and Invasive Fish Species in Two Great Lakes Tributaries." *Molecular Ecology*. https://doi.org/10.1111/mec.14395.

Blanchet, F. G., P. Legendre, and D. Borcard. 2008. "Modelling Directional Spatial Processes in Ecological Data." *Ecological Modelling*. https://doi.org/10.1016/j.ecolmodel.2008.04.001.

Carraro, L., and F. Altermatt. 2022. "Optimal Channel Networks Accurately Model Ecologically-Relevant Geomorphological Features of Branching River Networks." *Communications Earth and Environment*. https://doi.org/10.1038/s43247-022-00454-1.

Sakata, M. K., T. Watanabe, N. Maki, K. Ikeda, T. Kosuge, H. Okada, H. Yamanaka, T. Sado, M. Miya, and T. Minamoto. 2021. "Determining an Effective Sampling Method for eDNA Metabarcoding; a Case Study for Fish Biodiversity Monitoring in a Small, Natural River." *Limnology*. https://doi.org/10.1007/s10201-020-00645-9.