Package 'eDITH'

March 28, 2024

Type Package

Title Model Transport of Environmental DNA in River Networks

Version 0.3.0

Description Runs the eDITH (environmental DNA Integrating Transport
and Hydrology) model, which implements a mass balance of environmental DNA (eDNA)
transport at a river network scale coupled with a species distribution model
to obtain maps of species distribution. eDITH can work with both eDNA concentration
(e.g., obtained via quantitative polymerase chain reaction) or metabarcoding
(read count) data. Parameter estimation can be performed via Bayesian techniques
(via the 'BayesianTools' package) or optimization algorithms. An interface to the
'DHARMa' package for posterior predictive checks is provided. See Carraro and
Altermatt (2024) <doi:10.1111/2041-210X.14317> for a package introduction;
Carraro et al. (2018) <doi:10.1073/pnas.1813843115> and Carraro et al. (2020)
<doi:10.1038/s41467-020-17337-8> for methodological details.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 3.6)

Suggests knitr, rmarkdown, bookdown

VignetteBuilder knitr

URL `https://lucarraro.github.io/eDITH/`,
`https://github.com/lucarraro/eDITH`

BugReports `https://github.com/lucarraro/eDITH/issues`

Imports Rcpp (>= 1.0.10), OCNet (>= 1.1.0), rivnet (>= 0.3.1),
BayesianTools, LaplacesDemon, DHARMa, terra, fields

LinkingTo Rcpp

NeedsCompilation yes

Author Luca Carraro [cre, aut],
Florian Altermatt [aut],
University of Zurich [cph, fnd]

Maintainer Luca Carraro <`luca.carraro@hotmail.it`>

Repository CRAN

Date/Publication 2024-03-28 15:50:06 UTC

R topics documented:

`eDITH-package` *Model Transport of Environmental DNA In River Networks*

Description

Runs the eDITH (eDNA Integrating Transport and Hydrology) model, which implements a mass balance of eDNA transport at a river network scale coupled with a species distribution model to obtain maps of species distribution. eDITH can work with both eDNA concentration (e.g., obtained via qPCR) or metabarcoding (read count) data. Parameter estimation can be performed via Bayesian techniques (via the `BayesianTools` package) or optimization algorithms. An interface to the `DHARMa` package for posterior predictive checks is provided.

Author(s)

Luca Carraro (<`luca.carraro@hotmail.it`>)

References

Carraro, L., Hartikainen, H., Jokela, J., Bertuzzo, E., and Rinaldo, A. (2018). Estimating species distribution and abundance in river networks using environmental DNA. Proceedings of the National Academy of Sciences of the United States of America, 115(46), 11724-11729. doi:10.1073/pnas.1813843115

Carraro, L., Maechler, E., Wuethrich, R., and Altermatt, F. (2020). Environmental DNA allows upscaling spatial patterns of biodiversity in freshwater ecosystems. Nature Communications, 11(1) doi:10.1038/s41467-020-17337-8

Carraro, L., Stauffer, J. B., and Altermatt, F. (2021). How to design optimal eDNA sampling strategies for biomonitoring in river networks. Environmental DNA, 3(1), 157-172. doi:10.1002/edn3.137

Carraro, L., Blackman, R. C., and Altermatt, F. (2023). Modelling environmental DNA transport in rivers reveals highly resolved spatio-temporal biodiversity patterns. Scientific Reports, 13(1) doi:10.1038/s41598-023-35614-6

`dataC` *eDNA concentration data*

Description

The dataset consists of triplicate eDNA measurements for each of the 24 sampling sites.

Usage

```
data(dataC)
```

Format

A data frame containing location of eDNA sampling sites for the river Wigger (`dataC$ID`) and eDNA concentration values (`dataC$values`) (in mol m-3) for a given taxon.

`dataRead` *eDNA read number data*

Description

The dataset consists of triplicate eDNA measurements for each of the 24 sampling sites.

Usage

```
data(dataRead)
```

Format

A data frame containing location of eDNA sampling sites for the river Wigger (`dataRead$ID`) and eDNA read number values (`dataRead$values`) for a given taxon.

`eval_posterior_eDITH` *Evaluate posterior predictions from an eDITH run*

Description

Function that evaluates relevant quantities from a posterior sample of the parameters of an eDITH model

Usage

```
eval_posterior_eDITH(x, river, quant = 0.5)
```

Arguments

`x`	List as produced by `run_eDITH_BT`.
`river`	A `river` object generated via `aggregate_river`.
`quant`	Vector of quantiles.

Details

Add details.

Value

The output list copies all objects of the input `x` list. The following objects are added:

`p_quantile`	Selected quantiles (along rows) of the posterior distribution of production rates.
`C_quantile`	Selected quantiles (along rows) of the posterior distribution of eDNA values (concentrations or read numbers).
`probDet_quantile`	Selected quantiles (along rows) of the posterior distribution of detection probability.
`p_mean`	Mean of the posterior distribution of production rates.
`C_mean`	Mean of the posterior distribution of eDNA values (concentrations or read numbers).
`probDet_mean`	Mean of the posterior distribution of detection probability.

All of these objects are vectors of length `river$AG$nNodes`. However, if a custom likelihood was used in `run_eDITH_BT`, then `probDet_quantile` and `probDet_mean` are not evaluated, and are replaced by a vector of zero length.

Examples

```
library(rivnet)
data(wigger)
data(outSample)
out <- eval_posterior_eDITH(outSample, wigger)
plot(wigger, out$p_mean)
```

`outSample` *Posterior sample from fitted eDITH model*

Description

It is produced via:

```
covariates <- data.frame(urban=wigger$SC$locCov$landcover_1, agriculture=wigger$SC$locCov$landcover_2
forest=wigger$SC$locCov$landcover_3, elev=wigger$AG$Z, log_drainageArea=log(wigger$AG$A))
```

```
set.seed(1)
```

```
outSample <- run_eDITH_BT(dataC, wigger, covariates, mcmc.settings=list(iterations=9e5,
burnin = 6e5, message = TRUE, thin = 30))
```

Usage

```
data(outSample)
```

Format

A list.

`posterior_pred_sim_eDITH`
Predictive posterior simulations from an eDITH run

Description

This function performs predictive posterior simulations from a run of the eDITH model (via `run_eDITH_BT`). These can be used for diagnostics purposes, in particular to assess scaled (quantile) residuals via the `DHARMa` package.

Usage

```
posterior_pred_sim_eDITH(x, river, nParamSets = 10000, nDrawsPerParamSet = 10,
verbose = FALSE)
```

Arguments

`x`	List as produced by `run_eDITH_BT`.
`river`	A `river` object generated via `aggregate_river`.
`nParamSets`	Number of unique parameter sets sampled from the posterior distribution.
`nDrawsPerParamSet`	Number of simulations run per parameter set.
`verbose`	Logical. Should updates be printed on the console?

Details

`nParamSets` can be higher than the number of unique parameter sets in the posterior distribution, since the sampling of posterior parameter sets is operated with replacement.

Value

A matrix with dimensions `length(x$data$ID)`-by-`nParamSets*nDrawsPerParamSet`. Each column is a predictive posterior simulation. Each row corresponds to a site where eDNA data were observed (corresponding to the entries of argument `data` in `run_eDITH_BT`. Matrix entries are eDNA values (either concentrations or read numbers) predicted by the model for a given predictive posterior simulation at a given observational site.

See Also

`DHARMa`.

Examples

```
library(DHARMa)
data(outSample)
data(wigger)
data(dataC)
pps <- posterior_pred_sim_eDITH(outSample, wigger, nParamSets = 1000)
# reduced nParamSets for illustrative purposes

sim.out <- createDHARMa(pps, dataC$values)
plot(sim.out)
```

---

run_eDITH_BT *Run eDITH with BayesianTools*

Description

Function that runs a Bayesian sampler estimating parameters of an eDITH model

Usage

```
run_eDITH_BT(data, river, covariates = NULL, Z.normalize = TRUE,
           use.AEM = FALSE, n.AEM = NULL, par.AEM = NULL,
           no.det = FALSE, ll.type = "norm", source.area = "AG",
           mcmc.settings = NULL, likelihood = NULL,
   prior = NULL, sampler.type = "DREAMzs",
           tau.prior = list(spec = "lnorm", a = 0, b = Inf,
meanlog = log(5), sd = sqrt(log(5) - log(4))),
           log_p0.prior = list(spec="unif",min=-20, max=0),
           beta.prior = list(spec="norm",sd=1),
       sigma.prior = list(spec="unif",min=0, max=max(data$values, na.rm = TRUE)),
       omega.prior = list(spec="unif",min=1, max=10*max(data$values, na.rm = TRUE)),
       Cstar.prior = list(spec="unif",min=0, max=max(data$values, na.rm = TRUE)),
   verbose = FALSE)
```

Arguments

data	eDNA data. Data frame containing columns ID (index of the AG node/reach where the eDNA sample was taken) and values (value of the eDNA measurement, expressed as concentration or number of reads).
river	A river object generated via aggregate_river.
covariates	Data frame containing covariate values for all river reaches. If NULL (default option), production rates are estimated via AEMs.
Z.normalize	Logical. Should covariates be Z-normalized?
use.AEM	Logical. Should eigenvectors based on AEMs be used as covariates? If covariates = NULL, it is set to TRUE. If TRUE and covariates are provided, AEM eigenvectors are appended to the covariates data frame.

n.AEM	Number of AEM eigenvectors (sorted by the decreasing respective eigenvalue) to be used as covariates. If par.AEM$moranI = TRUE, this parameter is not used. Instead, the eigenvectors with significantly positive spatial autocorrelation are used as AEM covariates.
par.AEM	List of additional parameters that are passed to river_to_AEM for calculation of AEMs. In particular, par.AEM$moranI = TRUE imposes the use of AEM covariates with significantly positive spatial autocorrelation based on Moran's I statistic.
no.det	Logical. Should a probability of non-detection be included in the model?
ll.type	Character. String defining the error distribution used in the log-likelihood formulation. Allowed values are norm (for normal distribution), lnorm (for lognormal distribution), nbinom (for negative binomial distribution) and geom (for geometric distribution). The two latter choices are suited when eDNA data are expressed as read numbers, while norm and lnorm are better suited to eDNA concentrations.
source.area	Defines the extent of the source area of a node. Possible values are "AG" (if the source area is the reach surface, i.e. length*width), "SC" (if the source area is the subcatchment area), or, alternatively, a vector with length river$AG$nodes.
mcmc.settings	List. It is passed as argument settings in runMCMC. Default is list(iterations = 2.7e6, burnin=1.8e6, message = TRUE, thin = 10).
likelihood	Likelihood function to be passed as likelihood argument to createBayesianSetup. If not specified, it is generated based on arguments no.det and ll.type. If a custom likelihood is specified, a custom prior must also be specified.
prior	Prior function to be passed as prior argument to createBayesianSetup. If not specified, it is generated based on the *.prior arguments provided. If a user-defined prior is provided, parameter names must be included in prior$lower, prior$upper (see example).
sampler.type	Character. It is passed as argument sampler in runMCMC.
tau.prior	List that defines the prior distribution for the decay time parameter tau. See details.
log_p0.prior	List that defines the prior distribution for the logarithm (in base 10) of the baseline production rate p0. See details. If covariates = NULL, this defines the prior distribution for the logarithm (in base 10) of production rates for all river reaches.
beta.prior	List that defines the prior distribution for the covariate effects beta. See details. If a single spec is provided, the same prior distribution is specified for all beta parameters. Alternatively, if spec (and the other arguments, if provided) is a vector with length equal to the number of covariates included, different prior distributions can be specified for the different beta parameters.
sigma.prior	List that defines the prior distribution for the standard deviation of the measurement error when ll.type is "norm" or "lnorm". It is not used if ll.type = "nbinom". See details.
omega.prior	List that defines the prior distribution for the overdispersion parameter omega of the measurement error when ll.type = "nbinom". It is not used if ll.type is "norm" or "lnorm". See details.

`Cstar.prior`	List that defines the prior distribution for the `Cstar` parameter controlling the probability of no detection. It is only used if `no.det = TRUE`. See details.
`verbose`	Logical. Should console output be displayed?

Details

The arguments of the type `*.prior` consist in the lists of arguments required by `dtrunc` (except the first argument `x`).

By default, AEMs are computed without attributing weights to the edges of the river network. Use e.g. `par.AEM = list(weight = "gravity")` to attribute weights.

Value

A list with objects:

`param_map`	Vector of named parameters corresponding to the maximum a posteriori estimate. It is the output of the call to `MAP`.
`p_map`	Vector of best-fit eDNA production rates corresponding to the maximum a posteriori parameter estimate `param_map`. It has length equal to `river$AG$nNodes`.
`C_map`	Vector of best-fit eDNA values (in the same unit as `data$values`, i.e. concentrations or read numbers) corresponding to the maximum a posteriori parameter estimate `param_map`. It has length equal to `river$AG$nNodes`.
`probDet_map`	Vector of best-fit detection probabilities corresponding to the maximum a posteriori parameter estimate `param_map`. It has length equal to `river$AG$nNodes`. If a custom `likelihood` is provided, this is a vector of null length (in which case the user should calculate the probability of detection independently, based on the chosen likelihood).
`cI`	Output of the call to `getCredibleIntervals`.
`gD`	Output of the call to `gelmanDiagnostics`.
`covariates`	Data frame containing input covariate values (possibly Z-normalized).
`source.area`	Vector of source area values.
`outMCMC`	Object of class `mcmcSampler` returned by the call to `runMCMC`.

Moreover, arguments `ll.type` (possibly changed to `"custom"` if a custom likelihood is specified), `no.det` and `data` are added to the list.

Examples

```
data(wigger)
data(dataC)
data(dataRead)

# reduce number of iterations for illustrative purposes
# (use default mcmc.settings to ensure convergence)
settings.short <- list(iterations = 1e3, thin = 10)
set.seed(1)
out <- run_eDITH_BT(dataC, wigger, mcmc.settings = settings.short)
```

```
library(rivnet)
# best-fit (maximum a posteriori) map of eDNA production rates
plot(wigger, out$p_map)

# best-fit map (maximum a posteriori) of detection probability
plot(wigger, out$probDet_map)


# compare best-fit vs observed eDNA concentrations
plot(out$C_map[dataC$ID], dataC$values,
xlab="Modelled (MAP) concentrations", ylab="Observed concentrations")
abline(a=0, b=1)

## fit eDNA read number data - use AEMs as covariates
out <- run_eDITH_BT(dataRead, wigger, ll.type = "nbinom",
par.AEM = list(weight = "gravity"),
mcmc.settings = settings.short) # use default mcmc.settings to ensure convergence

## use user-defined covariates
covariates <- data.frame(urban = wigger$SC$locCov$landcover_1,
                         agriculture = wigger$SC$locCov$landcover_2,
                         forest = wigger$SC$locCov$landcover_3,
                         elev = wigger$AG$Z,
                         log_drainageArea = log(wigger$AG$A))

out.cov <-  run_eDITH_BT(dataC, wigger, covariates,
mcmc.settings = settings.short) # use default mcmc.settings to ensure convergence

# use user-defined covariates and AEMs
out.covAEM <-  run_eDITH_BT(dataC, wigger, covariates,
use.AEM = TRUE, par.AEM = list(weight = "gravity"),
mcmc.settings = settings.short) # use default mcmc.settings to ensure convergence

# use AEMs with significantly positive spatial autocorrelation
out.AEM.moran <- run_eDITH_BT(dataC, wigger, use.AEM = TRUE,
par.AEM = list(weight = "gravity", moranI = TRUE),
mcmc.settings = settings.short) # use default mcmc.settings to ensure convergence

## use posterior sample to specify user-defined prior
library(BayesianTools)
data(outSample)
pp <- createPriorDensity(outSample$outMCMC)
# Important! add parameter names to objects lower, upper
names(pp$lower) <- names(pp$upper) <- colnames(outSample$outMCMC$chain[[1]])[1:8]
# the three last columns are for log-posterior, log-likelihood, log-prior
out.new <- run_eDITH_BT(dataC, wigger, covariates, prior = pp,
mcmc.settings = settings.short)
```

`run_eDITH_optim` *Optimize eDITH*

Description

Function that performs search of optimal parameters of an eDITH model

Usage

```
run_eDITH_optim(data, river, covariates = NULL, Z.normalize = TRUE,
           use.AEM = FALSE, n.AEM = NULL, par.AEM = NULL,
           no.det = FALSE, ll.type = "norm", source.area = "AG",
           likelihood = NULL, sampler = NULL, n.attempts = 100,
   n.restarts = round(n.attempts/10), par.optim = NULL,
   tau.prior = list(spec="lnorm",a=0,b=Inf,
   meanlog=log(5), sd=sqrt(log(5)-log(4))),
           log_p0.prior = list(spec="unif",min=-20, max=0),
           beta.prior = list(spec="norm",sd=1),
       sigma.prior = list(spec="unif",min=0, max=1*max(data$values, na.rm = TRUE)),
       omega.prior = list(spec="unif",min=1, max=10*max(data$values, na.rm = TRUE)),
       Cstar.prior = list(spec="unif",min=0, max=1*max(data$values, na.rm = TRUE)),
   verbose = FALSE)
```

Arguments

`data`	eDNA data. Data frame containing columns `ID` (index of the AG node/reach where the eDNA sample was taken) and `values` (value of the eDNA measurement, expressed as concentration or number of reads).
`river`	A `river` object generated via `aggregate_river`.
`covariates`	Data frame containing covariate values for all `river` reaches. If `NULL` (default option), production rates are estimated via AEMs.
`Z.normalize`	Logical. Should covariates be Z-normalized?
`use.AEM`	Logical. Should eigenvectors based on AEMs be used as covariates? If `covariates = NULL`, it is set to `TRUE`. If `TRUE` and `covariates` are provided, AEM eigenvectors are appended to the `covariates` data frame.
`n.AEM`	Number of AEM eigenvectors (sorted by the decreasing respective eigenvalue) to be used as covariates. If `par.AEM$moranI = TRUE`, this parameter is not used. Instead, the eigenvectors with significantly positive spatial autocorrelation are used as AEM covariates.
`par.AEM`	List of additional parameters that are passed to `river_to_AEM` for calculation of AEMs. In particular, `par.AEM$moranI = TRUE` imposes the use of AEM covariates with significantly positive spatial autocorrelation based on Moran's I statistic.
`no.det`	Logical. Should a probability of non-detection be included in the model?

`ll.type`	Character. String defining the error distribution used in the log-likelihood formulation. Allowed values are `norm` (for normal distribution), `lnorm` (for log-normal distribution), `nbinom` (for negative binomial distribution) and `geom` (for geometric distribution). The two latter choices are suited when eDNA data are expressed as read numbers, while `norm` and `lnorm` are better suited to eDNA concentrations.
`source.area`	Defines the extent of the source area of a node. Possible values are `"AG"` (if the source area is the reach surface, i.e. length*width), `"SC"` (if the source area is the subcatchment area), or, alternatively, a vector with length `river$AG$nodes`.
`likelihood`	Likelihood function. If not specified, it is generated based on arguments `no.det` and `ll.type`.
`sampler`	Function generating sets of initial parameter values for the optimization algorithm. If `NULL`, initial parameter values are drawn from the default prior distributions of `run_eDITH_BT`. See details.
`n.attempts`	Number of times the optimizing function `optim` is executed. Every time a "restart" happens (see `n.restarts`), `sampler` is used to draw an initial parameter set. If a "restart" does not happen, the optimal parameter set from the previous attempt is used as initial parameter set.
`n.restarts`	Number of times a random parameter set is drawn as initial condition for `optim`.
`par.optim`	List of parameters to be passed to `optim`. By default, the likelihood is maximized (i.e., `control$fnscale = -1`), and the maximum number of iterations is set to 1e6. The default optimization method is "Nelder-Mead" (same default as in `optim`).
`tau.prior, log_p0.prior,beta.prior,sigma.prior,omega.prior,Cstar.prior`	Prior distribution for the relevant parameters of the eDITH model.
`verbose`	Logical. Should console output be displayed?

Details

This function attempts to maximize the log-posterior (sum of log-likelihood and log-prior) via the non-linear optimization function `optim`.

If specified by the user, `sampler` must be a function that produces as output a "named num" vector of parameters. Parameter names must be same as in the `likelihood`. See example.

By default, AEMs are computed without attributing weights to the edges of the river network. Use e.g. `par.AEM = list(weight = "gravity")` to attribute weights.

Value

A list with objects:

`p`	Vector of best-fit eDNA production rates corresponding to the optimum parameter estimates `param`. It has length equal to `river$AG$nNodes`.
`C`	Vector of best-fit eDNA values (in the same unit as `data$values`, i.e. concentrations or read numbers) corresponding to the optimum parameter estimates `param`. It has length equal to `river$AG$nNodes`.

`probDet`	Vector of best-fit detection probabilities corresponding to the optimum parameter estimate `param_map`. It has length equal to `river$AG$nNodes`. If a custom `likelihood` is provided, this is a vector of null length (in which case the user should calculate the probability of detection independently, based on the chosen likelihood).
`param`	Vector of named parameters corresponding to the best-fit estimate.
`covariates`	Data frame containing input covariate values (possibly Z-normalized).
`source.area`	Vector of source area values.
`out_optim`	List as provided by `optim`. Only the result of the call to `optim` (out of `n.attempts`) yielding the highest likelihood is exported.
`attempts.stats`	List containing relevant output for the different optimization attempts. It contains `lp` (vector of maximized log-posterior values for each single attempt), `counts` (total function evaluations), `conv` (convergence flags as produced by `optim`), and `tau` (best-fit decay time values in h).

Moreover, arguments `ll.type` (possibly changed to `"custom"` if a custom likelihood is specified), `no.det` and `data` are added to the list.

Examples

```
data(wigger)
data(dataC)
data(dataRead)

## fit eDNA concentration data - use AEMs as covariates
set.seed(9)
out <- run_eDITH_optim(dataC, wigger, n.AEM = 10,
n.attempts = 1) # reduced n.AEM, n.attempts for illustrative purposes
# it is recommended to attempt optimization several times to ensure convergence

library(rivnet)
# best-fit map of eDNA production rates
plot(wigger, out$p)

# best-fit map of detection probability
plot(wigger, out$probDet)

# compare best-fit vs observed eDNA concentrations
plot(out$C[dataC$ID], dataC$values,
xlab = "Modelled concentrations", ylab = "Observed concentrations")
abline(a=0, b=1)

## fit eDNA read number data - use AEMs as covariates
set.seed(5)
out <- run_eDITH_optim(dataRead, wigger, ll.type = "nbinom",
par.AEM = list(weight = "gravity"),
n.attempts = 1) # reduced n.attempts for illustrative purposes

## use user-defined covariates
covariates <- data.frame(urban = wigger$SC$locCov$landcover_1,
                            agriculture = wigger$SC$locCov$landcover_2,
                            forest = wigger$SC$locCov$landcover_3,
                            elev = wigger$AG$Z,
                            log_drainageArea = log(wigger$AG$A))

set.seed(2)
out.cov <-  run_eDITH_optim(dataC, wigger, covariates, n.attempts = 1)
# reduced n.attempts for illustrative purposes

# use user-defined covariates and AEMs
set.seed(1)
out.covAEM <- run_eDITH_optim(dataC, wigger, covariates, use.AEM = TRUE,
 par.AEM = list(weight = "gravity"),
 n.attempts = 1) # reduced n.attempts for illustrative purposes

# use AEMs with significantly positive spatial autocorrelation
set.seed(1)
out.AEM.moran <- run_eDITH_optim(dataC, wigger, use.AEM = TRUE,
par.AEM = list(weight = "gravity", moranI = TRUE),
n.attempts = 1) # reduced n.attempts for illustrative purposes

# define sampler function when the first 10 AEMs are used as covariates
samp_fun <- function(n){ # input argument needed but not used
    mins = c(0, -20, rep(-5,10), 0)
    maxs = c(10, 0, rep(5,10), 5e-12)
    nams = c("tau", "log_p0", paste0("beta_AEM",1:10), "sigma")
    vec <- runif(numeric(13), min=mins, max=maxs)
    names(vec) <- nams
    return(vec)}
set.seed(1)
out.samp <- run_eDITH_optim(dataC, wigger, n.AEM = 10,
    sampler = samp_fun,
n.attempts = 1) # reduced n.attempts for illustrative purposes
```

run_eDITH_single *Run eDITH for a single parameter set*

Description

Function that runs the eDITH model for a given parameter set

Usage

```
run_eDITH_single(param, river, covariates,  Z.normalize = TRUE,
no.det = FALSE, ll.type = NULL,
data = NULL, source.area = "AG",
                  tau.prior = list(spec="lnorm",a=0,b=Inf,
meanlog=log(5), sd=sqrt(log(5)-log(4))),
                log_p0.prior = list(spec="unif",min=-20, max=0),
                    beta.prior = list(spec="norm",sd=1),
                sigma.prior = list(spec="unif",min=0,
max=max(data$values, na.rm = TRUE)),
                omega.prior = list(spec="unif",min=1,
max=10*max(data$values, na.rm = TRUE)),
                 Cstar.prior = list(spec="unif",min=0,
max=max(data$values, na.rm = TRUE)))
```

Arguments

`param`	Parameter set. It has to be a named vector, with names: `tau` Decay time (expressed in h). `log_p0` Natural logarithm of the baseline production rate. `beta_X` Effect size of covariate `X`. There must be as many `beta_X` as columns in `covariates`. `X` must be the name of the corresponding column in `covariates`. `omega`, `sigma`, `Cstar` Parameters for estimation of the log-likelihood and detection probability. Only required if `ll.type` is provided.
`river`	A `river` object generated via `aggregate_river`.
`covariates`	Data frame containing covariate values for all `river` reaches.
`Z.normalize`	Logical. Should covariates be Z-normalized?
`no.det`	Logical. Should a probability of non-detection be included in the model?
`ll.type`	Character. String defining the error distribution used in the log-likelihood formulation. Allowed values are `norm` (for normal distribution), `lnorm` (for lognormal distribution), `nbinom` (for negative binomial distribution) and `geom` (for geometric distribution). The two latter choices are suited when eDNA data are expressed as read numbers, while `norm` and `lnorm` are better suited to eDNA concentrations.
`data`	eDNA data. Data frame containing columns `ID` (index of the AG node/reach where the eDNA sample was taken) and `values` (value of the eDNA measurement, expressed as concentration or number of reads).
`source.area`	Defines the extent of the source area of a node. Possible values are `"AG"` (if the source area is the reach surface, i.e. length*width), `"SC"` (if the source area is the subcatchment area), or, alternatively, a vector with length `river$AG$nodes`.
`tau.prior, log_p0.prior,beta.prior,sigma.prior,omega.prior,Cstar.prior`	Prior distribution for the relevant parameters of the eDITH model. Only used if both `ll.type` and `data` are provided.

Value

A list with objects:

`p`	Vector of eDNA production rates corresponding to the parameter set `param`. It has length equal to `river$AG$nNodes`.

`C`	Vector of eDNA values (concentrations or read numbers) corresponding to the parameter set `param`. It has length equal to `river$AG$nNodes`.
`probDet`	Vector of detection probabilities corresponding to the parameter set `param`. It is only computed if `ll.type` is provided. It has length equal to `river$AG$nNodes`.
`logprior`	Value of the log-prior distribution (computed only if `ll.type` and `data` are provided).
`loglik`	Value of the log-likelihood distribution (computed only if `ll.type` and `data` are provided).
`logpost`	Value of the log-posterior distribution (computed only if `ll.type` and `data` are provided).

See Also

See `run_eDITH_BT`, `run_eDITH_optim` for details on parameters names and log-likelihood specification.

Examples

```
library(rivnet)
data(wigger)

# calculate AEMs and use the first 10 as covariates
ae <- river_to_AEM(wigger)
covariates <- data.frame(ae$vectors[,1:10])
names(covariates) <- paste0("AEM",1:10)
# covariates names must correspond to param names
set.seed(1); param <- c(3,-15, runif(10,-1,1))
names(param) <- c("tau", "log_p0", paste0("beta_AEM",1:10))
# param names must correspond to covariates names

out <- run_eDITH_single(param, wigger, covariates)

# add parameter sigma and compute detection probability
param <- c(param, 5e-12)
names(param)[length(param)] <- "sigma"
# note that the value of sigma has to be within the range indicated by sigma.prior
out2 <- run_eDITH_single(param, wigger, covariates, ll.type="norm")

# include data and compute logprior, loglikelihood, logposterior
data(dataC)
out3 <- run_eDITH_single(param, wigger, covariates,
ll.type="norm", data=dataC)
```

`wigger` *River Wigger*

Description

It is built via

```
wigger <- extract_river(outlet=c(637478,237413), EPSG=21781, ext=c(6.2e5,6.6e5,2e5,2.5e5),
z=9)
wigger <- aggregate_river(wigger, maxReachLength = 2500)
hydrodata <- data.frame(data=c(8, 15), type=c("w","Q"), node=wigger$AG$outlet*c(1,1))
wigger <- hydro_river(hydrodata, wigger)
r1 <- rast(system.file("extdata/landcover.tif", package="rivnet"))
wigger <- covariate_river(r1, wigger)
```

Usage

```
data(wigger)
```

Format

A `river` object. See `extract_river` documentation for details.

Index