Package 'smnet'

January 27, 2015

Type Package

Title Smoothing For Stream Network Data

Version 1.0

Date 2014-12-09

Depends SSN, spam

Imports splines, dfoptim, RSQLite, igraph, DBI

Author Alastair Rushworth

Maintainer Alastair Rushworth <alastair.rushworth@glasgow.ac.uk>

Description Fits flexible additive models to data on stream networks, taking account of flow-connectivity of the network. Models are fit using penalised least squares.

License GPL-2

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-09 17:51:37

R topics documented:

get_adjacency *Construct an Adjacency Matrix*

Description

Builds a sparse adjacency matrix from a user specified SSN data directory, by extracting and processing the binaryID.db table. The resulting output of this function is typically a required input for fitting additive network models to SSN objects using the main smnet function.

Usage

```
get_adjacency(ssn_directory, net)
```

Arguments

ssn_directory	Required character string indicating the path to the location of the .ssn directory which contains the binaryID.db table
net	Integer specifying the particular stream network of interest within the SSN object. Defaults to 1.

Value

Square sparse matrix of class "spam" with row and column dimension equal to the number of stream segments. If the i^th column has non-zero elements j_1 and j_2 then this indicates that j_1 and j_2 are direct upstream neighbours of i.

If the i^{th} column has sum 1, then this indicates that i has only one upstream neighbour, and therefore no confluence lies between them; by default the spatial penalties treat these differently.

Author(s)

Alastair Rushworth

See Also

smnet

Examples

```
# Set up an SSN object - this part taken
# from the SSN:::SimulateOnSSN help file
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)
```

```
## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")

## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")

## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))

lmP<-smnet(formula = Sim_Values~1 +
          network(adjacency = adjacency, weight = "addfunccol", netID = 1),
          data.object = sim1.ssn)

plot(lmP, type = "segments")
plot(lmP, type = "nodes")
```

---

m *Specify Smooth Terms in Formulae*

Description

Function used to set up univariate or bivariate smooth terms based on P-splines, for use within a call to smnet.

Usage

```
m(..., k = -1)
```

Arguments

...	One or variables for which a P-spline smooth is desired
k	Integer, defines the number of evenly spaced B-spline basis functions to represent the smooth component, default is 10. For two-dimensional smooths, this is the marginal basis size.

Value

term	Character vector of the names of the variables involved in the smooth to be set up
bs.dim	Number of B-spline basis functions to be used in the smooth

Author(s)

Alastair Rushworth

References

Modified version of s originally from package mgcv, Simon Wood (2014).

See Also

smnet

Examples

```
# Set up an SSN object - this part taken
# from the SSN:::SimulateOnSSN help file
library(SSN)
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)

## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")
```

```
## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
  ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")

## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))

lmP<-smnet(formula = Sim_Values~ m(NEAR_X) +
          network(adjacency = adjacency, weight = "addfunccol",
          netID = 1), data.object = sim1.ssn)

plot(lmP, type = "covariates", res = TRUE)
plot(lmP, type = "segments")
plot(lmP, type = "nodes")
```

---

network *Specify Network Smoother in Formulae*

Description

This function specifies all of the information required to smooth parameters over the segments of a stream network using an adjacency matrix, and a vector of flow weights.

Usage

```
network(adjacency = NULL, weight = NULL, netID = 1, locs = NULL)
```

Arguments

adjacency	A sparse adjacency matrix describing the flow connections across the network. adjacency must be of class "spam", and is typically obtained from a call to get_adjacency.
weight	Either a numeric vector of flow weights or a character string indicating the column name of a numeric vector of flow weights contained in the data.object that has been passed to smnet. When a vector of flow weights is provided, this must have length equal to the number of rows of adjacency and respect the same ordering.
netID	Integer identifying the specific network of interest within the SSN object supplied to smnet. Ignored if the input data to smnet is a data.frame and not an SSN object.
locs	Either a character string referring to a column of stream segment locations inside the data.object that has been passed to smnet. Otherwise a numeric vector of stream segment locations, these should be arranged so that they correspond to the row and column ordering of the adjacency matrix. Ignored if the input data to smnet is a data.frame and not an SSN object.

Value

A list combining the processed input components above. For internal use within smnet.

adjacency	Sparse adjacency matrix
weight	Numeric vector of flow weights
netID	Integer identifying network of interest
locs	Vector of stream segments locations

Author(s)

Alastair Rushworth

See Also

smnet, get_adjacency

Examples

```
# Set up an SSN object - this part taken
# from the SSN:::SimulateOnSSN help file
library(SSN)
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)

## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")

## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
  ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")

## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))
```

```
lmP<-smnet(formula = Sim_Values~1 +
          network(adjacency = adjacency, weight = "addfunccol", netID = 1),
          data.object = sim1.ssn)

plot(lmP, type = "segments")
plot(lmP, type = "nodes")
```

`plot.smnet` *Plot a Stream Network Model*

Description

Plot linear, univariate and bivariate smooth effects and network smooth terms that resulting from a call to `smnet`.

Usage

```
  ## S3 method for class 'smnet'
plot(x, type = "covariates", se = FALSE, res = FALSE, coords = NULL, key = TRUE, ...)
```

Arguments

`x`	An object of class **smnet**
`type`	Character string identifying the type of plot to be produced. The default, "`covariates`", produces plots of all linear and smooth components (the latter corresponding to each appearance of `m` in the model formula). "node" plots the spatial network model component associated with a use of `network` in the model formula; the plot uses a set of nodes each associated with a stream stretch and plotted at their geographical midpoints. The colour of each node corresponds to the fitted value associated with that stream segment. "`segments`" plots the spatial network model component associated with a use of `network` in the model formula; segments are plotted using information within an **SSN** object (may be slower than "node" for larger networks).
`se`	Logical. When TRUE (the default), calculates and adds standard errors to plots of linear and smooth components the default is TRUE. Setting to FALSE may be be quicker for very large data sets.
`res`	Logical. Plots partial residuals on linear and smooth component plots when TRUE, the default. Ignored if cov = FALSE.
`coords`	A 2-column matrix of coordinates required to plot spatial smooths when input data was not an **SSN** object
`key`	Logical. Plots a colour legend for the node plot when set to TRUE (the default. Ignored when node = FALSE.)
`...`	Other arguments passed to plot

Author(s)

Alastair Rushworth

See Also

predict.smnet, summary.smnet

Examples

```
# Set up an SSN object - this part taken
# from the SSN:::SimulateOnSSN help file
library(SSN)
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)

## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")

## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
  ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")
```

```
## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))

lmP<-smnet(formula = Sim_Values~1 +
          network(adjacency = adjacency, weight = "addfunccol", netID = 1),
          data.object = sim1.ssn)

plot(lmP, type = "segments")
plot(lmP, type = "nodes")
```

`predict.smnet` *Predict From a Stream Network Model.*

Description

Get predictions and standard errors at a fixed set of spatial locations and covariate values from a model fitted by `smnet`.

Usage

```
## S3 method for class 'smnet'
predict(object, newdata =  NULL,...)
```

Arguments

`object`	Object of class `smnet`, usually the result of a call to smnet.
`newdata`	New design matrix at which to make predictions, not required if the input data to `smnet` was an `SSN` object with an associated set of prediction points.
`...`	other arguments passed to predict.smnet

Value

`predictions`	Vector of predictions corresponding to prediction points in the `SSN` input object
`predictions.se`	Vector of prediction standard errors

Author(s)

Alastair Rushworth

See Also

`smnet`

Examples

```
# Set up an SSN object - this part taken
# from the SSN:::SimulateOnSSN help file
library(SSN)
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)

## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")

## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
  ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")

## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))
```

```
# fit the model using smnet
lmP<-smnet(formula = Sim_Values~1 +
          network(adjacency = adjacency, weight = "addfunccol", netID = 1),
          data.object = sim1.ssn)

plot(lmP, type = "segments")
plot(lmP, type = "nodes")

# make some predictions
z<-predict(lmP)
```

`simulateNetwork` *Simulate river network topology and data*

Description

Simulates a graph, where nodes correspond to stream segments and an edge denotes that two stream segments are connected via a confluence. In addition data is simulated on the network, with one datum associated with each stream segment. Not of practical use, but good for checking that other utilities in SmoothNetwork work. Uses the igraph package to construct the graph.

Usage

```
simulateNetwork(n.segments, beta = NULL, lambda = 1, subs = F)
```

Arguments

`n.segments`	Number of nodes the simulated network should have
`lambda`	Positive real number: level of smoothness the simulated data should have. Small values mean less smoothness. Default = 1.
`beta`	Numeric vector of covariate effects. If supplied, independent random normal covariate data is generated on each stream segment with true coefficients as in `beta`.
`subs`	Number between 0 and 1. Determines whether data is simulated on a random subset of the network nodes of size `subs*n.segments`.

Value

`response`	Vector of simulated data of length `n.segments`, or if `subs` is supplied, of length `subs*n.segments`
`response.locs`	Vector indicating which node each value of `response`came from
`alpha`	Random intercept term, generated from U(0,1) draw
`z`	Vector of true spatial effects (`response` - `alpha` - random noise)
`Z`	If `subs` is supplied these are the true values for all nodes
`wgts`	Vector of (flow) weights, based on Shreve order
`adjacency`	`n.segments`*`n.segments` adjacency matrix of class `spam`
`coords`	`n.segments`*2 matrix of coordinates of node locations, useful for plotting

Author(s)

Alastair Rushworth

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. http://igraph.sf.net

See Also

plot.smnet, summary.smnet

Examples

```
# Generate some simulated data with 100 stream segments
x<-simulateNetwork(n.segments = 100, lambda = 1)

lmSim<-smnet(formula=response~1 +
               network(adjacency = x$adjacency, weight = x$wgts,
               locs = x$response.locs),
               data.object = x$obsdata,
               method = "AICC")

plot(lmSim, type = "nodes", coords = x$obsdata[,2:3])
```

---

smnet *Additive Modelling for Stream Networks*

Description

Fits (Gaussian) additive models to river network data based on the flexible modelling framework described in O'Donnell et al. (2014). Data must either be in the form of an object of class SpatialStreamNetwork as used by the R package SSN (Ver Hoef et al., 2012) or in the form of a data.frame. Smoothness of covariate effects is represented by transforming data onto a uniformly spaced B-spline basis; parameter estimates are obtained by penalized least-squares. Optimal smoothness is achieved by using a numerical optimization of AIC, GCV or AICC.

Care is taken to exploit the sparse matrix properties of model objects using the optimized storage and algebra routines in the R packages spam (Furrer and Sain, 2010). This allows faster fitting and lower memory footprint.

The formula interpreter used for penalised additive components is modelled on the code found in the package mgcv.

Usage

```
smnet(formula, data.object, control = NULL, method = "AICC")
```

Arguments

formula	A formula similar to those as used in by the gam function in the package mgcv. Smooth functions based on P-splines with m(..., k=20) function, up to 2-dimensional interactions currently supported. At present, only k the spline basis size can be specified using m.
data.object	Either an object of class "SpatialStreamNetwork" or a data.frame containing response variable and covariates
control	A list of options passed to the optimiser. maxit, default = 500, sets an upper limit of iterations made by the optimiser. approx = NULL, positive integer specifying the number samples to collect using a Monte-Carlo method to approximate the performance criterion surface when direct evaluation is too slow - this takes advantage of matrix sparsity and may be much faster if the network has a large number of segments or the data is large (approx = 100 often works well in these cases.
method	Character string determining the performance criterion for choosing optimal smoothness, options are "AICC" or "GCV".

Value

Object of class smnet with components

1	Original SSN object used provided as data.object, unchanged
2	List; model output including fitted values (fit), effective dimension (ED), residual variance (sigma.sq), sparse matrices involved in model fit etc.

Author(s)

Alastair Rushworth

References

Ver Hoef, J.M., Peterson, E.E., Clifford, D., Shah, R. (2012) SSN: An R Package for Spatial Statistical Modeling on Stream Networks

O' Donnell, D., Rushworth, A.M., Bowman, A.W., Scott, E.M., Hallard, M. (2014) Flexible regression models over river networks. Journal of the Royal Statistical Society: Series C (Applied Statistics). 63(1) 47–63.

Reinhard Furrer, Stephan R. Sain (2010). spam: A Sparse Matrix R Package with Emphasis on MCMC Methods for Gaussian Markov Random Fields. Journal of Statistical Software, 36(10), 1-25. URL: http://www.jstatsoft.org/v36/i10/

See Also

get_adjacency, plot.smnet

Examples

```
# Set up an SSN object; this part is taken from the SSN:::SimulateOnSSN help file
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)

## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")

## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")

## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))

lmP<-smnet(formula = Sim_Values~1 + m(NEAR_X)+
```

```
            network(adjacency = adjacency, weight = "addfunccol", netID = 1),
            data.object = sim1.ssn)

plot(lmP, type = "covariates")
plot(lmP, type = "nodes")
```

---

summary.smnet *Summarise Stream Network Model*

Description

Generate summaries of linear and smooth components of an smnet object.

Usage

```
## S3 method for class 'smnet'
summary(object, ...)
```

Arguments

object	An object of class smnet.
...	Other arguments passed to summary

Value

List object with components

1	smoothcomponents: a table containing the values of the smoothing parameters, and the partial degrees of freedom associated with each smooth component. Note: network components always have two smoothing parameters, where the second is a (usually small) ridge parameter.
2	fixed.effects: a table summarising the linear components of the fitted model

Author(s)

Alastair Rushworth

See Also

smnet

Examples

```
# Set up an SSN object - this part taken
# from the SSN:::SimulateOnSSN help file
library(SSN)
set.seed(101)
## simulate a SpatialStreamNetwork object
raw1.ssn <- createSSN(n = 100,
    obsDesign = binomialDesign(50), predDesign = binomialDesign(50),
  importToR = TRUE, path = paste(tempdir(),"/sim1", sep = ""))

## create distance matrices, including between predicted and observed
createDistMat(raw1.ssn, "preds", o.write=TRUE, amongpred = TRUE)

## extract the observed and predicted data frames
raw1DFobs <- getSSNdata.frame(raw1.ssn, "Obs")
raw1DFpred <- getSSNdata.frame(raw1.ssn, "preds")

## add a continuous covariate randomly
raw1DFobs[,"X1"] <- rnorm(length(raw1DFobs[,1]))
raw1DFpred[,"X1"] <- rnorm(length(raw1DFpred[,1]))

## add a categorical covariate randomly
raw1DFobs[,"F1"] <- as.factor(sample.int(3,length(raw1DFobs[,1]), replace = TRUE))
raw1DFpred[,"F1"] <- as.factor(sample.int(3,length(raw1DFpred[,1]), replace = TRUE))

## simulate Gaussian data
sim1.out <- SimulateOnSSN(raw1.ssn,
  ObsSimDF = raw1DFobs,
PredSimDF = raw1DFpred,
PredID = "preds",
formula = ~ X1 + F1,
coefficients = c(1, .5, -1, 1),
CorModels = c("Exponential.tailup", "Exponential.taildown"),
use.nugget = TRUE,
use.anisotropy = FALSE,
CorParms = c(2, 5, 2, 5, 0.1),
addfunccol = "addfunccol")

## extract the ssn.object
sim1.ssn <- sim1.out$ssn.object

## extract the observed and predicted data frames, now with simulated values
sim1DFobs <- getSSNdata.frame(sim1.ssn, "Obs")
sim1DFpred <- getSSNdata.frame(sim1.ssn, "preds")

## store simulated prediction values, and then create NAs in their place
sim1preds <- sim1DFpred[,"Sim_Values"]
sim1DFpred[,"Sim_Values"] <- NA
sim1.ssn <- putSSNdata.frame(sim1DFpred, sim1.ssn, "preds")

# create the adjacency matrix for use with SmoothNetwork
adjacency<-get_adjacency(paste(tempdir(),"/sim1", sep = ""))
```

```
lmP<-smnet(formula = Sim_Values~1 +
          network(adjacency = adjacency, weight = "addfunccol", netID = 1),
          data.object = sim1.ssn)

plot(lmP, type = "nodes")
plot(lmP, node = "segments")

summary(lmP)
```

Index