Package 'conStruct'

May 8, 2026

Version 1.0.6

Date 2024-1-08

Title Models Spatially Continuous and Discrete Population Genetic Structure

Description A method for modeling genetic data as a combination of discrete layers, within each of which relatedness may decay continuously with geographic distance. This package contains code for running analyses (which are implemented in the modeling language 'rstan') and visualizing and interpreting output. See the paper for more details on the model and its utility.

License GPL-3

Encoding UTF-8

LazyData true

ByteCompile true

Depends R (>= 3.4.0), Rcpp (>= 0.12.0), methods

Imports rstan (>= 2.26.0), rstantools (>= 1.5.0), caroline, gtools, foreach, parallel, doParallel

LinkingTo StanHeaders (>= 2.26.0), rstan (>= 2.26.0), BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1)

SystemRequirements GNU make

NeedsCompilation yes

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, maps

VignetteBuilder knitr

Author Gideon Bradburd [aut, cre]

Maintainer Gideon Bradburd <bradburd@umich.edu>

Repository CRAN

Date/Publication 2024-01-08 22:00:06 UTC

Contents

`conStruct-package` *The 'conStruct' package.*

Description

A method for modeling genetic data as a combination of discrete layers, within each of which relatedness may decay continuously with geographic distance. This package contains code for running analyses (which are implemented in the modeling language 'rstan') and visualizing and interpreting output. See the associated paper for more details on the model and its utility.

References

G.S. Bradburd, G.M. Coop, and P.L. Ralph (2018) <doi: 10.1534/genetics.118.301333>.

Stan Development Team (2018). RStan: the R interface to Stan. R package version 2.17.3. http://mc-stan.org

`calculate.layer.contribution` *Calculate layer contribution*

Description

`calculate.layer.contribution`

Usage

```
calculate.layer.contribution(conStruct.results, data.block, layer.order = NULL)
```

Arguments

`conStruct.results`	The list output by a `conStruct` run for a given MCMC chain.
`data.block`	A `data.block` list saved during a `conStruct` run.
`layer.order`	An optional `vector` giving the order in which the layers of `conStruct.results` are read.

Details

This function takes the results of a `conStruct` analysis and calculates the relative contributions of each layer to total covariance.

This function calculates the contribution of each layer to total covariance by multiplying the within-layer covariance in a given layer by the admixture proportions samples draw from that layer. The relative contribution of that layer is this absolute contribution divided by the sum of those of all other layers. A layer can have a large contribution if many samples draw large amounts of admixture from it, or if it has a very large within-layer covariance parameter (phi), or some combination of the two. Layer contribution can be useful for evaluating an appropriate level of model complexity for the data (e.g., choosing a value of `K` or comparing the spatial and nonspatial models).

Value

This function returns a `vector` giving the relative contributions of the layers in the analysis.

`compare.two.runs` *Compare two conStruct runs*

Description

`compare.two.runs` makes figures comparing the output from two conStruct analyses.

Usage

```
compare.two.runs(
  conStruct.results1,
  data.block1,
  conStruct.results2,
  data.block2,
  prefix,
  layer.colors = NULL
)
```

Arguments

conStruct.results1	The list output by a conStruct run.
data.block1	A data.block list saved during a conStruct run.
conStruct.results2	The list output by a second conStruct run.
data.block2	A data.block list saved during a second conStruct run.
prefix	A character vector to be prepended to all figures.
layer.colors	A vector of colors to be used in plotting results for different layers. Users must specify one color per layer. If NULL, plots will use a pre-specified vector of colors.

Details

This function takes the outputs from two conStruct analyses and generates a number of plots for comparing results and diagnosing MCMC performance.

This function produces a variety of plots that can be useful for comparing results from two conStruct analyses. The runs must have the same number of independent MCMC chains, but may have different values of K. The spatial and nonspatial models can be compared. If the runs were executed with different values of K, the run with the smaller value of K should be specified in the first set of arguments (conStruct.results1 and data.block1).

The plots made are by no means an exhaustive, and users are encouraged to make further plots, or customize these plots as they see fit. For each plot, one file is generated for each MCMC chain in each analysis (specified with the n.chains argument in the function conStruct. For clarity, the layers in the second are matched to those in the first using the function match.clusters.x.runs The plots generated (as .pdf files) are:

- Structure plot - STRUCTURE-style plot, where each sample is represented as a stacked bar plot, and the length of the bar plot segments of each color represent that sample's admixture proportion in that layer. Described further in the help page for make.structure.plot.
- Admixture pie plot - A map of samples in which each sample's location is denoted with a pie chart, and the proportion of a pie chart of each color represents that sample's admixture in each layer. Described further in the help page for make.admix.pie.plot
- model.fit.CIs - A plot of the sample allelic covariance shown with the 95% credible interval of the parametric covariance for each entry in the matrix.

- layer.covariances - A plot of the layer-specific covariances overlain unto the sample allelic covariance.
- Trace plots - Plots of parameter values over the MCMC.
  - lpd - A plot of the log posterior probability over the MCMC.
  - nuggets - A plot of estimates of the nugget parameters over the MCMC.
  - gamma - A plot of estimates of the gamma parameter over the MCMC.
  - layer.cov.params - Plots of estimates of the layer-specific parameters over the MCMC.
  - admix.props - A plot of estimates of the admixture proportions over the MCMC.

Value

This function has only invisible return values.

`conStruct` *Run a conStruct analysis.*

Description

`conStruct` runs a conStruct analysis of genetic data.

Usage

```
conStruct(
  spatial = TRUE,
  K,
  freqs,
  geoDist = NULL,
  coords,
  prefix = "",
  n.chains = 1,
  n.iter = 1000,
  make.figs = TRUE,
  save.files = TRUE,
  ...
)
```

Arguments

`spatial`	A logical indicating whether to perform a spatial analysis. Default is `TRUE`.
`K`	An `integer` that indicates the number of layers to be included in the analysis.
`freqs`	A `matrix` of allele frequencies with one column per locus and one row per sample. Missing data should be indicated with `NA`.
`geoDist`	A full `matrix` of geographic distance between samples. If `NULL`, user can only run the nonspatial model.

coords	A matrix giving the longitude and latitude (or X and Y coordinates) of the samples.
prefix	A character vector giving the prefix to be attached to all output files.
n.chains	An integer indicating the number of MCMC chains to be run in the analysis. Default is 1.
n.iter	An integer giving the number of iterations each MCMC chain is run. Default is 1e3. If the number of iterations is greater than 500, the MCMC is thinned so that the number of retained iterations is 500 (before burn-in).
make.figs	A logical value indicating whether to automatically make figures once the analysis is complete. Default is TRUE.
save.files	A logical value indicating whether to automatically save output and intermediate files once the analysis is complete. Default is TRUE.
...	Further options to be passed to rstan::sampling (e.g., adapt_delta).

Details

This function initiates an analysis that uses geographic and genetic relationships between samples to estimate sample membership (admixture proportions) across a user-specified number of layers.

This function acts as a wrapper around a STAN model block determined by the user-specified model (e.g., a spatial model with 3 layers, or a nonspatial model with 5 layers). User-specified data are checked for appropriate format and consistent dimensions, then formatted into a data.block, which is then passed to the STAN model block. Along with the conStruct.results output described above, several objects are saved during the course of a conStruct call (if save.files=TRUE). These are the data.block, which contains all data passed to the STAN model block, model.fit, which is unprocessed results of the STAN run in stanfit format, and the conStruct.results, which are saved in the course of the function call in addition to being returned. If make.figs=TRUE, running conStruct will also generate many output figures, which are detailed in the function make.all.the.plots in this package.

Value

This function returns a list with one entry for each chain run (specified with n.chains). The entry for each chain is named "chain_X" for the Xth chain. The components of the entries for each are detailed below:

- posterior gives parameter estimates over the posterior distribution of the MCMC.
  - n.iter number of MCMC iterations retained for analysis (half of the n.iter argument specified in the function call).
  - lpd vector of log posterior density over the retained MCMC iterations.
  - nuggets matrix of estimated nugget parameters with one row per MCMC iteration and one column per sample.
  - par.cov array of estimated parametric covariance matrices, for which the first dimension is the number of MCMC iterations.
  - gamma vector of estimated gamma parameter.
  - layer.params list summarizing estimates of layer-specific parameters. There is one entry for each layer specified, and the entry for the kth layer is named "Layer_k".

    * alpha0 vector of estimated alpha0 parameter in the kth layer.
    * alphaD vector of estimated alphaD parameter in the kth layer.
    * alpha2 vector of estimated alpha2 parameter in the kth layer.
    * mu vector of estimated mu parameter in the kth layer.
    * layer.cov vector of estimated layer-specific covariance parameter in the kth layer.
  - admix.proportions array of estimated admixture proportions. The first dimension is the number of MCMC iterations, the second is the number of samples, and the third is the number of layers.
- MAP gives point estimates of the parameters listed in the posterior list described above. Values are indexed at the MCMC iteration with the greatest posterior probability.
  - index.iter the iteration of the MCMC with the highest posterior probability, which is used to index all parameters included in the MAP list
  - lpd the greatest value of the posterior probability
  - nuggets point estimate of nugget parameters
  - par.cov point estimate of parametric covariance
  - gamma point estimate of gamma parameter
  - layer.params point estimates of all layer-specific parameters
  - admix.proportions point estimates of admixture proportions.

Examples

```
# load example dataset
data(conStruct.data)

# run example spatial analysis with K=1
#
# for this example, make.figs and save.files
# are set to FALSE, but most users will want them
# set to TRUE
my.run <- conStruct(spatial = TRUE,
  K = 1,
  freqs = conStruct.data$allele.frequencies,
  geoDist = conStruct.data$geoDist,
  coords = conStruct.data$coords,
  prefix = "test",
  n.chains = 1,
  n.iter = 1e3,
  make.figs = FALSE,
  save.files = FALSE)
```

---

conStruct.data *Example dataset used in a* conStruct *analysis*

Description

A simulated dataset containing the allele frequency and sampling coordinate data necessary to run a conStruct analysis.

Usage

```
conStruct.data
```

Format

A list with two elements:

allele.frequencies a matrix with one row for each of the 16 samples and one column for each of 10,000 loci, giving the frequency of the counted allele at each locus in each sample

coords a matrix with one row for each of the 16 samples, in the same order as that of the allele frequency matrix, and two columns, the first giving the x-coordinate (or longitude), the second giving the y-coordinate (or latitude)

---

data.block *Example* data.block *generated by a* conStruct *analysis*

Description

An example data.block object generated in a conStruct analysis from the raw data supplied by the user. This object is automatically saved and is used in several subsequent plotting functions.

Usage

```
data.block
```

Format

A list with 7 elements:

N the number of samples included in the analysis

K the number of clusters/layers included in the model

spatial a boolean indicating whether the spatial model has been specified

L the number of loci included in the analysis

coords a matrix with one row for each of the N samples, in the same order as that of the obsCov matrix, and two columns, the first giving the x-coordinate (or longitude), the second giving the y-coordinate (or latitude)

obsCov the sample allelic covariance matrix, in the same order as that of the coords matrix, with N rows and columns

geoDist a matrix of pairwise geographic distance between , samples in the same order as that of the obsCov, with N rows and columns

sd.geoDist the standard deviation of the raw geographic distance matrix, used for normalizing geoDist within the stan model

varMeanFreqs the variance of the mean allele frequencies, averaged over choice of counted allele (passed to the model as a prior on the global covariance parameter)

make.admix.pie.plot *Make admixture pie plot*

Description

make.structure.plot makes a map of pie plots showing admixture proportions across layers.

Usage

```
make.admix.pie.plot(
  admix.proportions,
  coords,
  layer.colors = NULL,
  radii = 2.7,
  add = FALSE,
  x.lim = NULL,
  y.lim = NULL,
  mar = c(2, 2, 2, 2)
)
```

Arguments

admix.proportions	A matrix of admixture proportions, with one row per sample and one column per layer.
coords	matrix of sample coordinates, with one row per sample and two columns giving (respectively) the X and Y plotting coordinates.
layer.colors	A vector of colors to be used in plotting results for different layers. Users must specify one color per layer. If NULL, the plot will use a pre-specified vector of colors.
radii	A vector of numeric values giving the radii to be used in plotting admixture pie plots. If the number of values specified is smaller than the number of samples, radii values will be recycled across samples. The default is 2.7.
add	A logical value indicating whether to add the pie plots to an existing plot. Default is FALSE.
x.lim	A vector giving the x limits of the plot. The default value is NULL, which indicates that the range of values given in the first column of coords should be used.
y.lim	A vector giving the y limits of the plot. The default value is NULL, which indicates that the range of values given in the second column of coords should be used.
mar	A vector giving the number of lines of margin specified for the four sides of the plotting window (passed to par). Default value, which is only used if add=FALSE, is c(2,2,2,2).

Details

This function takes the output from a conStruct analysis and makes a map of pie plots showing admixture proportions across layers, where each sample is represented as a pie chart, and the proportion of the pie of each color represent that sample's admixture proportion in that layer.

Value

This function has only invisible return values.

Examples

```
# make admixture pie plot
make.admix.pie.plot(admix.proportions = admix.props,coords = coords)
```

`make.all.the.plots` *Make output plots*

Description

`make.all.the.plots` makes figures from the output from a conStruct analysis.

Usage

```
make.all.the.plots(conStruct.results, data.block, prefix, layer.colors = NULL)
```

Arguments

`conStruct.results`	The list output by a `conStruct` run.
`data.block`	A `data.block` list saved during a `conStruct` run.
`prefix`	A character vector to be prepended to all figures.
`layer.colors`	A `vector` of colors to be used in plotting results for different layers. Users must specify one color per layer. If `NULL`, plots will use a pre-specified vector of colors.

Details

This function takes the output from a conStruct analysis and generates a number of plots for visualizing results and diagnosing MCMC performance.

This function produces a variety of plots that can be useful for visualizing results or diagnosing MCMC performance. The plots made are by no means exhaustive, and users are encouraged to make further plots, or customize these plots as they see fit. For each plot, one file is generated for each MCMC chain (specified with the `n.chains` argument in the function `conStruct`. The plots generated (as .pdf files) are:

- Structure plot - STRUCTURE-style plot, where each sample is represented as a stacked bar plot, and the length of the bar plot segments of each color represent that sample's admixture proportion in that layer. Described further in the help page for `make.structure.plot`.
- Admixture pie plot - A map of samples in which each sample's location is denoted with a pie chart, and the proportion of a pie chart of each color represents that sample's admixture in each layer. Described further in the help page for `make.admix.pie.plot`
- model.fit.CIs - A plot of the sample allelic covariance shown with the 95% credible interval of the parametric covariance for each entry in the matrix.
- layer.covariances - A plot of the layer-specific covariances overlain unto the sample allelic covariance.
- Trace plots - Plots of parameter values over the MCMC.
  - lpd - A plot of the log posterior probability over the MCMC.
  - nuggets - A plot of estimates of the nugget parameters over the MCMC.
  - gamma - A plot of estimates of the gamma parameter over the MCMC.
  - layer.cov.params - Plots of estimates of the layer-specific parameters over the MCMC.
  - admix.props - A plot of estimates of the admixture proportions over the MCMC.

Value

This function has only invisible return values.

`make.structure.plot` *Make STRUCTURE output plot*

Description

`make.structure.plot` makes a STRUCTURE-style plot from the output from a conStruct analysis.

Usage

```
make.structure.plot(
  admix.proportions,
  mar = c(2, 4, 2, 2),
  sample.order = NULL,
  layer.order = NULL,
  sample.names = NULL,
  sort.by = NULL,
  layer.colors = NULL
)
```

Arguments

admix.proportions	A matrix of admixture proportions, with one row per sample and one column per layer.
mar	A vector of plotting margins passed to par. Default is c(2,4,2,2), which tends to look good.
sample.order	A vector giving the order in which sample admixture proportions are to be plotted, left to right. If NULL, samples are plotted in the order they occur in admix.proportions.
layer.order	A vector giving the order in which layers are plotted, bottom to top. If NULL, layers are plotted in the order they occur in admix.proportions.
sample.names	Vector of names to be plotted under each sample's admixture proportion bar plot. The index of a sample's name should be the same as the index of the sample's row in admix.proportions. If NULL, no names are printed.
sort.by	An integer giving the column index of the admix.proportions matrix to be used in determining sample plotting order. If specified, samples are plotted from left to right in increasing order of their membership in that layer. If NULL, samples are plotted in the order they occur in admix.proportions.
layer.colors	A vector of colors to be used in plotting results for different layers. Users must specify one color per layer. If NULL, the plot will use a pre-specified vector of colors.

Details

This function takes the output from a conStruct analysis and makes a STRUCTURE-style plot, where each sample is represented as a stacked bar plot, and the length of the bar plot segments of each color represent that sample's admixture proportion in that layer.

Value

This function has only invisible return values.

Examples

```
# make STRUCTURE-style plot
make.structure.plot(admix.proportions = admix.props)

# make STRUCTURE-style plot, sorted by membership in layer 1
make.structure.plot(admix.proportions = admix.props,sort.by=1)
```

`match.layers.x.runs` *Match layers up across independent conStruct runs*

Description

`match.layers.x.runs`

Usage

```
match.layers.x.runs(admix.mat1, admix.mat2, admix.mat1.order = NULL)
```

Arguments

`admix.mat1`	A `matrix` of estimated admixture proportions from the original `conStruct` analysis, with one row per sample and one column per layer.
`admix.mat2`	A `matrix` of estimated admixture proportions from a second `conStruct` analysis, with one row per sample and one column per layer, for which the layer order is desired. Must have equal or greater number of layers to `admix.mat1`.
`admix.mat1.order`	An optional `vector` giving the order in which the layers of `admix.mat1` are read.

Details

This function takes the results of two independent `conStruct` analyses and compares them to identify which layers in a new analysis correspond most closely to the layers from an original analysis.

This function compares admixture proportions in layers across independent `conStruct` runs, and compares between them to identify the layers in `admix.mat2` that correspond most closely to those in `admix.mat1`. It then returns a vector giving an ordering of `admix.mat2` that matches up the order of the layers that correspond to each other. This can be useful for:

1. Dealing with "label switching" across independent runs with the same number of layers;
2. Plotting results from independent runs with different numbers of layers using consistent colors (e.g., the "blue" layer shows up as blue even as `K` increases);
3. Examining results for multimodality (i.e., multiple distinct solutions with qualitatively different patterns of membership across layers).

The `admix.mat1.order` argument can be useful when running this function to sync up plotting colors/order across the output of more than two `conStruct` runs.

Value

This function returns a `vector` giving the ordering of the layers in `admix.mat2` that maximizes similarity between `admix.mat1` and re-ordered `admix.mat2`.

Examples

```
# compare the estimated admixture proportions from
# two different conStruct runs to determine which
# layers in one run correspond to those in the other
match.layers.x.runs(admix.props1,admix.props2)
```

`print.conStruct.results` *An S3 print method for class conStruct.results*

Description

An S3 print method for class conStruct.results

Usage

```
## S3 method for class 'conStruct.results'
print(x, ...)
```

Arguments

`x`	an object of class `conStruct.results`
`...`	further options to be passed to `print`

Value

prints a top-level summary of the conStruct.results, returns nothing

---

`print.data.block` *An S3 print method for class data.block*

Description

An S3 print method for class data.block

Usage

```
## S3 method for class 'data.block'
print(x, ...)
```

Arguments

`x`	an object of class `data.block`
`...`	further options to be passed to `print`

Value

prints a top-level summary of the data.block, returns nothing

---

print.freq.data *An S3 print method for class freq.data*

Description

An S3 print method for class freq.data

Usage

```
## S3 method for class 'freq.data'
print(x, ...)
```

Arguments

x	an object of class freq.data
...	further options to be passed to print

Value

prints a top-level summary of the freq.data, returns nothing

---

print.layer.params *An S3 print method for class layer.params*

Description

An S3 print method for class layer.params

Usage

```
## S3 method for class 'layer.params'
print(x, ...)
```

Arguments

x	an object of class layer.params
...	further options to be passed to print

Value

prints a top-level summary of the layer.params, returns nothing

`structure2conStruct` *Convert a dataset from STRUCTURE to conStruct format*

Description

`structure2conStruct` converts a STRUCTURE dataset to conStruct format

Usage

```
structure2conStruct(
  infile,
  onerowperind,
  start.loci,
  start.samples = 1,
  missing.datum,
  outfile
)
```

Arguments

`infile`	The name and path of the file in STRUCTURE format to be converted to `conStruct` format.
`onerowperind`	Indicates whether the file format has one row per individual (`TRUE`) or two rows per individual (`FALSE`).
`start.loci`	The index of the first column in the dataset that contains genotype data.
`start.samples`	The index of the first row in the dataset that contains genotype data (e.g., after any headers). Default value is 1.
`missing.datum`	The character or value used to denote missing data in the STRUCTURE dataset (often 0 or -9).
`outfile`	The name and path of the file containing the `conStruct` formatted dataset to be generated by this function.

Details

This function takes a population genetics dataset in STRUCTURE format and converts it to conStruct format. The STRUCTURE file can have one row per individual and two columns per locus, or one column and two rows per individual. It can only contain bi-allelic SNPs. Missing data is acceptable, but must be indicated with a single value throughout the dataset.

This function takes a STRUCTURE format data file and converts it to a `conStruct` format data file. This function can only be applied to diploid organisms. The STRUCTURE data file must be a plain text file. If there is extraneous text or column headers before the data starts, those extra lines should be deleted by hand or taken into account via the `start.samples` argument.

The STRUCTURE dataset can either be in the ONEROWPERIND=1 file format, with one row per individual and two columns per locus, or the ONEROWPERIND=0 format, with two rows and one column per individual. The first column of the STRUCTURE dataset should be individual names.

There may be any number of other columns that contain non-genotype information before the first column that contains genotype data, but there can be no extraneous columns at the end of the dataset, after the genotype data.

The genotype data must be bi-allelic single nucleotide polymorphisms (SNPs). Applying this function to datasets with more than two alleles per locus may result in cryptic failure. For more details, see the `format-data` vignette.

Value

This function returns an allele frequency data matrix that can be used as the `freqs` argument in a conStruct analysis run using `conStruct`. It also saves this object as an .RData file so that it can be used in future analyses.

`x.validation` *Run a conStruct cross-validation analysis*

Description

`x.validation` runs a conStruct cross-validation analysis

Usage

```
x.validation(
  train.prop = 0.9,
  n.reps,
  K,
  freqs = NULL,
  data.partitions = NULL,
  geoDist,
  coords,
  prefix,
  n.iter,
  make.figs = FALSE,
  save.files = FALSE,
  parallel = FALSE,
  n.nodes = NULL,
  ...
)
```

Arguments

`train.prop`	A numeric value between 0 and 1 that gives the proportions of the data to be used in the training partition of the analysis. Default is 0.9.
`n.reps`	An `integer` giving the number of cross- validation replicates to be run.
`K`	A numeric `vector` giving the numbers of layers to be tested in each cross-validation replicate. E.g., `K=1:7`.

freqs	A matrix of allele frequencies with one column per locus and one row per sample. Missing data should be indicated with NA.
data.partitions	A list with one element for each desired cross-validation replicate. This argument can be specified instead of the freqs argument if the user wants to provide their own data partitions for model training and testing. See the model comparison vignette for details on what this should look like.
geoDist	A matrix of geographic distance between samples. If NULL, user can only run the nonspatial model.
coords	A matrix giving the longitude and latitude (or X and Y coordinates) of the samples.
prefix	A character vector giving the prefix to be attached to all output files.
n.iter	An integer giving the number of iterations each MCMC chain is run. Default is 1e3. If the number of iterations is greater than 500, the MCMC is thinned so that the number of retained iterations is 500 (before burn-in).
make.figs	A logical value indicating whether to automatically make figures during the course of the cross-validation analysis. Default is FALSE.
save.files	A logical value indicating whether to automatically save output and intermediate files once the analysis is complete. Default is FALSE.
parallel	A logical value indicating whether or not to run the different cross-validation replicates in parallel. Default is FALSE. For more details on how to set up runs in parallel, see the model comparison vignette.
n.nodes	Number of nodes to run parallel analyses on. Default is NULL. Ignored if parallel is FALSE. For more details in how to set up runs in parallel, see the model comparison vignette.
...	Further options to be passed to rstan::sampling (e.g., adapt_delta).

Details

This function initiates a cross-validation analysis that uses Monte Carlo cross-validation to determine the statistical support for models with different numbers of layers or with and without a spatial component.

Value

This function returns (and also saves as a .Robj) a list containing the standardized results of the cross-validation analysis across replicates. For each replicate, the function returns a list with the following elements:

- sp - the mean of the standardized log likelihoods of the "testing" data partition of that replicate for the spatial model for each value of K specified in K.
- nsp - the mean of the standardized log likelihoods of the "testing" data partitions of that replicate for the nonspatial model for each value of K specified in K.

In addition, this function saves two text files containing the standardized cross-validation results for the spatial and nonspatial results (prefix_sp_xval_results.txt and prefix_nsp_xval_results.txt, respectively). These values are written as matrices for user convenience; each column is a cross-validation replicate, and each row gives the result for a value of K.

Index