Package ‘geoCount’

January 10, 2012

Type Package

Title Analysis and Modeling for Geostatistical Count Data

Version 1.1

Date 2011-12-17

Author Liang Jing

Maintainer Liang Jing <ljing918@gmail.com>

Description This package provides a variety of functions to analyze and model geostatistical count data with generalized linear spatial models, including
1) simulate and visualize the data;
2) posterior sampling with robust MCMC algorithms (in serial or parallel way);
3) perform prediction for unsampled locations;
4) conduct Bayesian model checking procedure to evaluate the goodness of fitting;
5) conduct transformed residual checking procedure.
In the package, seamlessly embedded C++ programs and parallel computing techniques are implemented to speed up the computing processes.

License GPL (>= 2)

LazyLoad Yes

Depends R (>= 2.12.0), Rcpp (>= 0.9.7), RcppArmadillo (>= 0.2.29)

LinkingTo Rcpp, RcppArmadillo

Suggests coda, distrEx, reldist, snow, snowfall, multicore

R topics documented:

`baseline.dist` *Generate Distance Samples to Build Baseline Distribution*

Description

This function generates the samples of distance to build the baseline distribution for standard normal.

Usage

```
baseline.dist(n, iter)
```

Arguments

`n`	the number of residuals
`iter`	the number of distance samples to generate

Details

`HellingerDist` and `KolmogorovDist` functions in `{distrEx}` are used to compute the distances. See `?HellingerDist` and `?KolmogorovDist` for details about how the distances are computed.

Value

A $iter \times 3$ matrix for three types of distance: "Discrete Hellinger", "Smooth Hellinger" and "Kolmogorov".

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`d.base`, `baseline.parallel`, `plot_baseline`, `pOne`.

Examples

```
## Not run:
# Time-consuming! Run once with large "iter" and
# save the results for future use
d.base <- baseline.dist(50, iter=100)
## End(Not run)
```

baseline.parallel *Generate Distance Samples to Build Baseline Distribution (Parallel Version)*

Description

This function generates distance samples in parallel to build the baseline distribution for standard normal.

Usage

```
baseline.parallel(n, iter, n.cores = getOption("cores"))
```

Arguments

`n`	the number of residuals
`iter`	the number of distance samples to generate
`n.cores`	the number of CPUs that will be used for parallel computing

Details

`HellingerDist` and `KolmogorovDist` functions in `{distrEx}` are used to compute the distances. See `?HellingerDist` and `?KolmogorovDist` for details about how the distances are computed.

This function performs parallel computing with the help of `{multicore}` package. Be aware that `{multicore}` package currently is not available in Windows.

Value

A $iter \times 3$ matrix for three types of distance: "Discrete Hellinger", "Smooth Hellinger" and "Kolmogorov".

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`d.base`, `baseline.dist`, `plot_baseline`, `pOne`.

Examples

```
## Not run:
# Time-consuming! Run once with large "iter" and
# save the results for future use
require(multicore)
d.base <- baseline.parallel(50, iter=100, n.cores = 4)
## End(Not run)
```

BMCT *Perform Bayesian Model Checking*

Description

This function conducts Bayesian model checking by comparing observed and reference data sets and reveals the result via "p-value" and "RPS" (as well as the plot).

Usage

```
BMCT(Y.obs, Y.rep, funcT, ifplot = FALSE)
```

Arguments

`Y.obs`	a vector which indicates the observed data set
`Y.rep`	a matrix which indicates the reference data sets
`funcT`	a function which defines the dignostic statistic
`ifplot`	a logical value which indicates whether plot the dignostic statistics

Value

A vector of p-value and RPS.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`repYeb`, `repYpost`, `pRPS`, `plot_pRPS`

Examples

```
## Not run:
Yrep.eb <- repYeb(N.sim=2000, loc, L, res.m, est = "mode")
funcT <- function(Y){ max(Y)-min(Y) }
BMCT(Y, Yrep.eb, funcT, ifplot=TRUE)
## End(Not run)
```

cdfU *Approximate the CDF Value from Reference Samples*

Description

This function approximates the CDF value for the observed data by using reference data.

Usage

```
cdfU(Y.obs, Y.rep, discrete = FALSE)
```

Arguments

`Y.obs`	a vector which indicates the observed data set
`Y.rep`	a matrix which indicates the reference data sets
`discrete`	a logical value which indicates if the variable is discrete

Value

A vector of CDF values.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`tranR`

Examples

```
## Not run:
Y.obs <- 11:20
res <- matrix(0, 10, 50)
for(i in 1:50){
Y.rep <- matrix(rpois(10*5000, 15), 10, )
res[, i] <- cdfU(Y.obs, Y.rep)
}
matplot(t(res), type="l")
abline(h = ppois(11:20, 15))

## End(Not run)
```

`cutChain` *Modify Markov Chains with Burn-in and Thining*

Description

This function takes the results from `runMCMC` and modifies the chains of posterior samples for burn-in and thinning.

Usage

```
cutChain(res, chain.ind=2:4, burnin, thinning)
```

Arguments

`res`	a list with elements containing the posterior samples of latent variables and parameters; usually the output from `runMCMC`
`chain.ind`	the index of elements in "res" that will be modified
`burnin`	the number for burn-in
`thinning`	the number for thining

Value

A list with elements containing the modified posterior samples.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`runMCMC`, `runMCMC.multiChain`, `mixChain`.

Examples

```
## Not run:
res <- runMCMC(Y, L=0, loc=loc, MCMCinput = input )
res.m <- cutChain(res, chain.ind=1:4, burnin=100, thining=10)

## End(Not run)
```

d.base *Data Set of Baseline Samples*

Description

This data set contains baseline samples for 100 residuals with 5000 iterations.

Usage

```
data(Dbase_n100N5000)
```

Details

A data.frame "d.base" with 5000 observations and 3 variables will be loaded.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`baseline.dist`, `plot_baseline`, `pOne`.

Examples

```
## Not run:

data(Dbase_n100N5000)
str(d.base)
plot_baseline(d.base[,1], colnames(d.base)[1])

## End(Not run)
```

e2dist *Calculate Distances between Transformed Residuals and Standard Normal*

Description

This function calcualtes three types of distance between the empirical distribution of transformed residuals and standard normal.

Usage

```
e2dist(e.tran)
```

Arguments

`e.tran`	a vector which indicates the transformed residuals

Details

`HellingerDist` and `KolmogorovDist` functions in `{distrEx}` are used to compute the distances. See `?HellingerDist` and `?KolmogorovDist` for details about how the distances are computed.

Value

A vector with length 3 containing "Discrete Hellinger", "Smooth Hellinger" and "Kolmogorov" distances.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`tranR`, `baseline.dist`, `pOne`.

Examples

```
## Not run:
require(distrEx)
e2dist(rnorm(200))
## End(Not run)
```

`Earthquakes` *Data Set of Earthquakes*

Description

This data set contains the informations of earthquakes.

Usage

```
data(datEarthquake)
```

Details

...

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

plotData.

Examples

```
## Not run:

data(datEarthquake)
str(Earthquakes)
plotData(Earthquakes$Magnitude, Earthquakes[,c("Lat","Lon")])

## End(Not run)
```

---

findMode *Estimate Mode of the Posterior Samples*

Description

This function estimates the mode of empirical density function for a given posterior samples.

Usage

```
findMode(x, ...)
```

Arguments

`x`	a vector of posterior samples
`...`	other parameters used when estimating then empirical density function; see `?density`

Details

This function uses `density` function to estimate the empirical density function.

Value

The value of mode.

Author(s)

Liang Jing <ljing918@gmail.com>

Examples

```
## Not run:
findMode(rnorm(1000))

## End(Not run)
```

`helloWorld` *Hello*

Description

Test package loading.

Usage

```
helloWorld()
```

Examples

```
## Not run:
helloWorld()

## End(Not run)
```

`loc2U` *Calculate the Distance Matrix among Given Locations*

Description

This function calculates the distance matrix among the given locations.

Usage

```
loc2U(loc)
```

Arguments

`loc`	a matrix of $n \times 2$ which indicates the x-y coordinates of the original locations

Details

This function calls the underlying C++ program to do the computation.

Value

A $n \times n$ matrix with the element e_{ij} indicating the distance between the ith and jth locations.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

loc2U_R, locCircle, locGrid, locSquad.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  U <- loc2U(loc)

## End(Not run)
```

loc2U_R *Calculate the Distance Matrix among Given Locations*

Description

This function calculates the distance matrix among the given locations.

Usage

```
loc2U_R(loc)
```

Arguments

loc	a matrix of $n \times 2$ which indicates the x-y coordinates of the original locations

Details

This function performs the computation in R.

Value

A $n \times n$ matrix with the element e_{ij} indicating the distance between the ith and jth locations.

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

loc2U, locCircle, locGrid, locSquad.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  U <- loc2U_R(loc)

## End(Not run)
```

`locCircle` *Simulate Circlular Locations*

Description

This function simulates a given number of locations equally distributed on a circle.

Usage

```
locCircle(r, np)
```

Arguments

`r`	the radius of the circle
`np`	the number of locations on the circle

Details

The center of the circle is (0, 0).

Value

A $np \times 2$ matrix indicates the x-y coordinates of the locations.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`locGrid`, `locSquad`, `simData`, `plotData`.

Examples

```
## Not run:
  loc <- locCircle(1, 40)

## End(Not run)
```

`locGrid` *Simulate Locations on Grid*

Description

This function simulates a given number of locations distributed on a grid.

Usage

```
locGrid(x, y, nx, ny)
```

Arguments

`x`	the length of x edge
`y`	the length of y edge
`nx`	the number of locations in x direction
`ny`	the number of locations in y direction

Details

The grid lies in the range of $(0, x) \times (0, y)$.

Value

A $(nx \times ny) \times 2$ matrix indicates the x-y coordinates of the locations.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`locCircle`, `locSquad`, `simData`, `plotData`.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  plot(loc, xlab="x", ylab="y")

## End(Not run)
```

`locSquad` *Simulate Squared Locations*

Description

This function simulates a given number of locations equally distributed on a square.

Usage

```
locSquad(a, np)
```

Arguments

`a`	half length of the edge
`np`	the number of locations on each edge

Details

The center of the square is (0, 0).

Value

A $(4 \times np - 4) \times 2$ matrix indicates the x-y coordinates of the locations.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`locCircle`, `locGrid`, `simData`, `plotData`.

Examples

```
## Not run:
  plot(locSquad(0.5, 4))

## End(Not run)
```

`locUloc` *Calculate the Distance Matrix Between Observed and Predicting Locations*

Description

This function calculates the distance matrix between observed and predicting locations.

Usage

```
locUloc(loc, locp)
```

Arguments

`loc`	a matrix of $n \times 2$ which indicates the x-y coordinates of the observed locations; if a vector is used, it will be converted to matrix automatically
`locp`	a matrix of $m \times 2$ which indicates the x-y coordinates of the predicting locations; if a vector is used, it will be converted to matrix automatically

Details

This function calls the underlying C++ program to do the computation.

Value

A $m \times n$ matrix with the element e_{ij} indicating the distance between the ith predicting location and the jth observed locations.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`loc2U`, `locCircle`, `locGrid`, `locSquad`.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  locp <- c(0.5, 0.5)
  U <- locUloc(loc, locp)

## End(Not run)
```

`locUloc_R` *Calculate the Distance Matrix Between Observed and Predicting Locations*

Description

This function calculates the distance matrix between observed and predicting locations.

Usage

```
locUloc_R(loc, locp)
```

Arguments

`loc`	a matrix of $n \times 2$ which indicates the x-y coordinates of the observed locations; if a vector is used, it will be converted to matrix automatically
`locp`	a matrix of $m \times 2$ which indicates the x-y coordinates of the predicting locations; if a vector is used, it will be converted to matrix automatically

Details

This function performs the computation in R.

Value

A $m \times n$ matrix with the element e_{ij} indicating the distance between the ith predicting location and the jth observed locations.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`locUloc`, `loc2U`, `locCircle`, `locGrid`, `locSquad`.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  locp <- c(0.5, 0.5)
  U <- locUloc_R(loc, locp)

## End(Not run)
```

`MCMCinput` *Settings for the MCMC Algorithm*

Description

This function sets up the parameters and initial values used for the MCMC algorithms.

Usage

```
MCMCinput(run = 200, run.S = 1, rho.family = "rhoPowerExp",
        Y.family = "Poisson", ifkappa = 0,
        scales = c(0.5, 1.65^2 + 0.8, 0.8, 0.7, 0.15),
        phi.bound = c(0.005, 1),
        initials = list(c(1), 1.5, 0.2, 1))
```

Arguments

`run`	the number of iterations
`run.S`	the number of internal iterations for latent variables
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables
`ifkappa`	take zero or non-zero value which indicates whether κ should be sampled
`scales`	a vector which indicates the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$ respectively
`phi.bound`	the upper and lower bound for ϕ
`initials`	a list which indicates the initial values for $(\beta, \sigma, \phi, \kappa)$ respectively

Details

During each iteration of Gibbs sampling process, the group of latent variables is updated "run.S" times to improve accuracy and reduce autocorrelations.

Value

A list of setting parameters.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`runMCMC`, `runMCMC.multiChain`, `runMCMC.sf`.

Examples

```
## Not run:
  input <- MCMCinput( run = 10000, run.S = 10,
          rho.family = "rhoPowerExp",
          Y.family = "Poisson", ifkappa=0,
          scales=c(0.5, 1.5, 0.9, 0.6, 0.5),
          phi.bound=c(0.005, 1),
          initials=list(c(-1, 2, 1), 1, 0.1, 1) )
  res <- runMCMC(Y, L=0, loc=loc, X=loc, MCMCinput = input )

## End(Not run)
```

`mixChain` *Mix Parallel Markov Chains*

Description

This function mix parallel chains into one chain.

Usage

```
mixChain(res.m.prl)
```

Arguments

`res.m.prl`	a list with each element containing the result of posterior samples from one CPU; the elements shoul only contain the Markov chains of posterior samples (while "AccRate" is eliminated when using `cutChain`)

Value

A list with elements containing the mixed posterior samples.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`runMCMC.multiChain`, `cutChain`.

Examples

```
## Not run:
res.prl <- runMCMC.multiChain(Y, L=0, loc=loc, X=loc,
            MCMCinput = input, n.chn = 5)
res.m.prl <- lapply(res.prl, cutChain, chain.ind=1:4, burnin=200, thining=20)
res.mix <- mixChain(res.m.prl)

## End(Not run)
```

`plotACF` *Auto-correlation Plot for Latent Variables*

Description

This function plots auto-correlation curves for latent variables.

Usage

```
plotACF(S.mcmc, lags = NULL)
```

Arguments

`S.mcmc`	a matrix (or data.frame) with each row containing the posterior samples of one latent variable
`lags`	the maximum number of lags; the default "NULL" will result in $10*\log_{10}(N/m)$ where N is the number of observations and m the number of series

Details

This function uses `acf` function to compute the estimates of auto-correlation.

Value

No return value. A plot of auto-correlation curves.

Author(s)

Liang Jing `<ljing918@gmail.com>`

Examples

```
## Not run:
plotACF(res.m$S.posterior)

## End(Not run)
```

`plotData` *Plot Geostatistical Data*

Description

This function plots geostatistical data for up to three data sets.

Usage

```
plotData(Y, loc, Yp = NULL, locp = NULL, Yt = NULL, loct = NULL,
         col = 1:2, colt = 3, pch = 1, size = c(0.3, 2.7), ...)
```

Arguments

`Y, Yp, Yt`	the vector of response variables
`loc, locp, loct`	$n \times 2$ matrix that indicates the coordinates of locations
`col, colt`	the colors used for different sets of response variables
`pch`	the shape
`size`	the minimum and maximum of the sizes
`...`	other parameters that control the plotting

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`locCircle`, `locGrid`, `locSquad`, `plotDataBD`, `simData`.

Examples

```
## Not run:
  loc <- rbind(locCircle(1, 60),
               locCircle(0.667, 40),
               locCircle(0.333, 20)
               )
  dat <- simData(loc, cov.par = c(1, 0.1, 1))
  Y <- dat$data
  plotData(Y[1:60], loc[1:60, ], Y[61:100], loc[61:100, ],
           Y[101:120], loc[101:120, ], pch = 16
           )

## End(Not run)
```

plotDataBD *Plot Geostatistical Data*

Description

This function plots the given boundaries and geostatistical data sets.

Usage

```
plotDataBD(bdry, Y = NULL, loc = NULL,
        Yp = NULL, locp = NULL, Yt = NULL, loct = NULL,
        col = 1:2, colt = 3, pch = 1, size = c(0.3, 2.7), ...)
```

Arguments

`bdry`	a list containing the coordinates of boundaries
`Y, Yp, Yt`	the vector of response variables
`loc, locp, loct`	$n \times 2$ matrix that indicates the coordinates of locations
`col, colt`	the colors used for different sets of response variables
`pch`	the shape
`size`	the minimum and maximum of the sizes
`...`	other parameters that control the plotting

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`simData`, `plotData`.

Examples

```
## Not run:
data(TexasCounty_boundary)
plotDataBD(TexasCounty.boundary, xlab = "Longitude", ylab = "Latitude")

## End(Not run)
```

`plot_baseline` *Plot Baseline Samples*

Description

This function plots the baseline samples.

Usage

```
plot_baseline(d.samples, dist.name)
```

Arguments

`d.samples`	the baseline samples
`dist.name`	the name of distance

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`e2dist`, `baseline.dist` `d.base`.

Examples

```
## Not run:
plot_baseline(d.base[,1], colnames(d.base)[1])
plot_baseline(d.base[,2], colnames(d.base)[2])
plot_baseline(d.base[,3], colnames(d.base)[3])
## End(Not run)
```

`plot_etran` *Plot Transformed Residuals*

Description

This function plots transformed residuals in different types.

Usage

```
plot_etran(e.tran, fig = 1:4)
```

Arguments

`e.tran`	a vector which indicates the transformed residuals
`fig`	a vector which indicates which types to plot: 1 indicates scatter plot, 2 indicates QQ-plot, 3 indicates density plot, and 4 indicates relative density plot (with standard normal distribution served as the base)

Details

`density` function is used to compute the empirical density.

`reldist` function in `{reldist}` is used to compute the relative density.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`tranR`

Examples

```
## Not run:
require(reldist)
plot_etran(rnorm(200), fig=c(1,4))
## End(Not run)
```

`plot_pRPS` *Plot Observed vs. Reference Diagnostic Statistics*

Description

This function plots the observed statistic vs. the empirical density of reference statistics.

Usage

```
plot_pRPS(T.obs, T.rep, nm = "x")
```

Arguments

`T.obs`	a value which indicates the observed statistic
`T.rep`	a vector which indicates the reference statistics
`nm`	the name of the diagnostic statistics

Details

`density` function is used to compute the empirical density of reference statistics.

Value

A plot.

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

BMCT, pRPS.

Examples

```
## Not run:
plot_pRPS(1, rnorm(1000))
## End(Not run)
```

---

pOne *Calculate One-side P-value*

Description

This function calculates one-side p-value(s) for observed distance(s) with respect to the samples of baseline distances.

Usage

```
pOne(d.obs, d.base)
```

Arguments

d.obs	a value (or a vector) which indicates the distance for observed data
d.base	a vector (or a matrix) which indicates the samples of baseline distances

Value

A p-value (or a vector of p-values).

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

e2dist, baseline.dist, pRPS, plot_pRPS.

Examples

```
## Not run:
# For single value
pOne(2, rnorm(10000))
# Visualize with plot_pRPS()
plot_pRPS(2, rnorm(10000), nm="d")
# For vector
pOne(1:3, matrix(rnorm(30000),,3))

## End(Not run)
```

---

predY *Predict for Unsampled Locations*

Description

This function generates posterior predictive samples of latent and response variables for predicting locations.

Usage

```
predY(res.m, loc, locp, X = NULL, Xp = NULL, Lp = 0, k = 1,
      rho.family = "rhoPowerExp", Y.family = "Poisson",
      parallel = NULL, n.cores = getOption("cores"),
      cluster.type = "SOCK")
```

Arguments

`res.m`	a list with elements containing the posterior samples of latent variables and parameters for observed locations
`loc`	a matrix which indicates the coordinates of the observed locations
`locp`	a matrix which indicates the coordinates of the predicting locations
`X`	the covariate matrix for observed locations
`Xp`	the covariate matrix for predicting locations
`Lp`	a vector which indicates the time duration during which the Poisson counts are accumulated or the total number of trials for Binomial response; if 0 is found in the vector, 1 will be used to replace all the values in the vector
`k`	a value for fixed κ; ignored if there are posterior samples for κ in "res.m"
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables
`parallel`	the default input `NULL` indicates no parallel computing will be applied; the input value `"multicore"` or `"snowfall"` indicates parallel computing with the help of `{multicore}` or `{snowfall}` package will be applied a logical value which indicates if the parallel computing should be used for prediction

`n.cores`	the number of CPUs that will be used for parallel computing; used only if `parallel` isn't `NULL`
`cluster.type`	type of cluster to be used for parallel computing; can be "SOCK", "MPI", "PVM", or "NWS"; used only if `parallel="snowfall"`

Details

This function performs parallel computing with the help of `{multicore}` or `{snowfall}` package. Be aware that `{multicore}` package currently is not available in Windows (so set `parallel="snowfall"` if you want to do parallel prediction in Windows).

Value

A list with elements:

`S.predict`	a matrix containing the posterior predictive samples for latent variables
`Y.predict`	a matrix containing the posterior predictive samples for response variables

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`runMCMC`, `runMCMC.multiChain`.

Examples

```
## Not run:
Ypred <- predY(res.m, loc, locp, X=loc, Xp=locp, k=1,
               rho.family = "rhoPowerExp", Y.family = "Poisson")
# require(multicore)
# Ypred <- predY(res.m, loc, locp, X=loc, Xp=locp,
#                parallel="multicore", n.cores = 4)
Ypred.avg <- rowMeans(Ypred$Y); EYpred.avg <- rowMeans(exp(Ypred$Sp))

## End(Not run)
```

`pRPS` *Calculate P-value and RPS*

Description

This function calculates p-value and relative predictive surprise (RPS) by comparing observed and reference statistics.

Usage

```
pRPS(T.obs, T.rep)
```

Arguments

`T.obs`	a value which indicates the observed statistic
`T.rep`	a vector which indicates the reference statistics

Details

`density` function is used to compute the empirical density of reference statistics.

Value

A vector of p-value and RPS.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`BMCT`, `plot_pRPS`.

Examples

```
## Not run:
pRPS(2, rnorm(1000))
## End(Not run)
```

`repYeb` *Generate Replicated Data with Estimated Parameters*

Description

This function generates replicated data sets based on estimated parameters (given or from posterior samples).

Usage

```
repYeb(N.sim, loc, L, X = NULL, rho.family = "rhoPowerExp",
       Y.family="Poisson", res.m = NULL, est = "mode",
       beta = NULL, sigma = NULL, phi = NULL, k = 1)
```

Arguments

`N.sim`	the number of replicated data sets to be simulated
`loc`	a $n \times 2$ matrix which indicates the coordinates of observed locations
`L`	a vector of length n; it indicates the time duration during which the Poisson counts are accumulated, or the total number of trials for Binomial response
`X`	a $n \times p$ covariate matrix; the default value "NULL" indicates no covariate
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables
`res.m`	a list with elements containing the posterior samples of latent variables and parameters for observed locations
`est`	take the value of `"mode"` which indicates the mode of posterior samples will be used as the parameter estimate; otherwise, the mean will be used
`beta`	a value which indicates the estimation for β; ignored if "res.m" is given
`sigma`	a value which indicates the estimation for σ; ignored if "res.m" is given
`phi`	a value which indicates the estimation for ϕ; ignored if "res.m" is given
`k`	a value which indicates the estimation for κ; ignored if "res.m" is given and contains the posterior samples for κ

Value

A $n \times N.sim$ matrix of replicated data sets.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`repYpost`, `simData`.

Examples

```
## Not run:
# Estimate parameters from posterior samples
Yrep.eb <- repYeb(N.sim=2000, loc, L, res.m, est = "mode")
# Pre-determined parameters (also an efficient way to simulate massive data sets)
Yrep.eb <- repYeb(N.sim=2000, loc, L, beta = 5, sigma = 1, phi = 0.1,
                  k = 1)

## End(Not run)
```

`repYpost` *Generate Replicated Data with Posterior Samples of Latent Variables*

Description

This function generates replicated data sets based on posterior samples of latent variables.

Usage

```
repYpost(res.m, L, Y.family="Poisson")
```

Arguments

`res.m`	a list with elements containing the posterior samples of latent variables and parameters for observed locations
`L`	a vector of length n; it indicates the time duration during which the Poisson counts are accumulated, or the total number of trials for Binomial response
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables

Value

A matrix of replicated data sets.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`repYeb`

Examples

```
## Not run:
Yrep.post <- repYpost(res.m, L)
## End(Not run)
```

`rhoMatern` *Matern Correlation Function*

Description

This function calculates the Matern correlation.

Usage

```
rhoMatern(u, a, k)
```

Arguments

`u`	a value which indicates the distance
`a`	a value which indicates the scale parameter, ϕ
`k`	a value which indicates the shape parameter, κ

Details

The function is $\rho(u) = [2]^{\kappa-1}(-u/\phi)^{\kappa} K_{\kappa}(-u/\phi)$ where $K_{\kappa}(\cdot)$ denotes a modified Bessel function of order κ.

Value

A value of the correlation.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`rhoPowerExp`, `U2Z`, `loc2U`.

Examples

```
## Not run:
  rhoMatern(0.3, a=0.1, k=1)

## End(Not run)
```

rhoPowerExp *Powered Exponential Correlation Function*

Description

This function calculates the powered exponential correlation.

Usage

```
rhoPowerExp(u, a, k)
```

Arguments

`u`	a value which indicates the distance
`a`	a value which indicates the scale parameter, ϕ
`k`	a value which indicates the shape parameter, κ

Details

The function is $\rho(u) = \exp((-u/\phi)^{\kappa})$.

When using the powered exponential correlation function, note that $0 < \kappa <= 2$.

Value

A value of the correlation.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`rhoMatern`, `U2Z`, `loc2U`.

Examples

```
## Not run:
  rhoPowerExp(0.3, a=0.1, k=1)

## End(Not run)
```

`Rongelap` *Data Set of Rongelap Island*

Description

This data set contains the Rongelap data.

Usage

```
data(datRongelap)
```

Details

The data were collected from Rongelap Island, the principal island of Rongelap Atoll in the South Pacific, which forms part of the Marshall Islands. U.S. nuclear weapon testing program generated heavy fallout over the island in the 1950s and it has been uninhabited since 1985. Diggle, P. J., Tawn, J. A. and Moyeed, R. A. (1998). Model based geostatistics (with discussion). Applied Statistics, 47, 299-350.

Value

A list with 4 elements:

`coords`	a 157×2 matrix which indicates the coordinates of 157 sampled locations
`data`	a vector of length 157 indicates the counts of photo emission for 157 sampled locations
`units.m`	a vector of length 157 indicates the time (in seconds) over which the counts was accumulated
`borders`	a matrix containing the boundary information of Rongelap island

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`plotData`.

Examples

```
## Not run:
data(datRongelap)
str(Rongelap)
plotDataBD(Rongelap$borders, Rongelap$data, Rongelap$coords)

## End(Not run)
```

runMCMC *Perform Robust MCMC Algorithms for GLSM*

Description

This function performs robust MCMC algorithms for generalized linear spatial models and generates posterior samples for latent variables and hyper-parameters.

Usage

```
runMCMC(Y, L = 0, loc, X = NULL, run = 200, run.S = 1,
        rho.family = "rhoPowerExp", Y.family = "Poisson", ifkappa = 0,
        scales = c(0.5, 1.65^2 + 0.8, 0.8, 0.7, 0.15),
        phi.bound = c(0.005, 1),
        initials = list(c(1), 1.5, 0.2, 1),
        MCMCinput = NULL, partial = FALSE, famT = 1)
```

Arguments

`Y`	a vector of length n which indicates the response variables
`L`	a vector of length n; it indicates the time duration during which the Poisson counts are accumulated, or the total number of trials for Binomial response; if 0 is found in the vector, 1 will be used to replace all the values in the vector
`loc`	a $n \times 2$ matrix which indicates the coordinates of locations
`X`	a $n \times p$ covariate matrix; the default value "NULL" indicates no covariate
`run`	the number of iterations
`run.S`	the number of internal iterations for latern variables
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables
`ifkappa`	take zero or non-zero value which indicates whether κ should be sampled
`scales`	a vector which indicates the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$ respectively
`phi.bound`	the upper and lower bound for ϕ
`initials`	a list which indicates the initial values for $(\beta, \sigma, \phi, \kappa)$ respectively
`MCMCinput`	a list of alternative settings; usually the result from `MCMCinput` function
`partial`	a logical input which indicates whether partial posterior sampling should be used; only works for Y.family = "Poisson"
`famT`	take the value of 1, 2, or 3 which indicates the type of partial posterior sampling: 1 means "mean" diagnostic statistic is used, 2 means "maximum", and 3 means "minimum"; ignored if `partial=FALSE`

Details

Group updating scheme, Langevin algorithms, and Data-based parameterization are applied to improve the robustness and efficiency of MCMC algorithms. The flat priors are used to guarantee an appropriate posterior. See my dissertation for more details.

During each iteration of Gibbs sampling process, the group of latent variables is updated "run.S" times to improve accuracy and reduce autocorrelations.

Value

A list with elements:

`S.posterior`	a $n \times run$ matrix containing the posterior samples for latent variables
`m.posterior`	a $(p+1) \times run$ matrix (in case of p covariate variables) or a vector with length "run" (no covariate case), containing the posterior samples for β
`s.posterior`	a vector with length "run" containing the posterior samples for σ
`a.posterior`	a vector with length "run" containing the posterior samples for ϕ
`k.posterior`	a vector with length "run" containing the posterior samples for κ in the case that "ifkappa" is set to non-zero value
`AccRate`	a vector which indicates the acceptance rates

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`MCMCinput`, `runMCMC.multiChain`, `runMCMC.sf`.

Examples

```
## Not run:
  data(datWeed)
  Y <- Weed[,3]
  loc <- unifLoc(Weed[,1:2])
  L <- rep(1, length(Y))
  input.Weed <- MCMCinput( run=1000, run.S=10, rho.family="rhoPowerExp",
          Y.family = "Poisson", ifkappa=0,
          scales=c(0.5, 3.5, 0.9, 0.6, 0.5),
          phi.bound=c(0.005, 1),
          initials=list(c(-1), 1, 0.1, 1) )
  res <- runMCMC(Y, L=L, loc=loc, X=NULL, MCMCinput = input.Weed )

## End(Not run)
```

runMCMC.multiChain *Perform Robust MCMC Algorithms for GLSM in Parallel*

Description

This function performs robust MCMC algorithms in a parallel way for generalized linear spatial models and generates posterior samples for latent variables and hyper-parameters.

Usage

```
runMCMC.multiChain(Y, L = 0, loc, X = NULL, run = 200, run.S = 1,
        rho.family = "rhoPowerExp", Y.family = "Poisson", ifkappa = 0,
        scales = c(0.5, 1.65^2 + 0.8, 0.8, 0.7, 0.15),
        phi.bound = c(0.005, 1), initials = list(c(1), 1.5, 0.2, 1),
        MCMCinput = NULL, partial = FALSE, famT = 1,
        n.chn = 2, n.cores = getOption("cores"))
```

Arguments

`Y`	a vector of length n which indicates the response variables
`L`	a vector of length n; it indicates the time duration during which the Poisson counts are accumulated, or the total number of trials for Binomial response; if 0 is found in the vector, 1 will be used to replace all the values in the vector
`loc`	a $n \times 2$ matrix which indicates the coordinates of locations
`X`	a $n \times p$ covariate matrix; the default value "NULL" indicates no covariate
`run`	the number of iterations
`run.S`	the number of internal iterations for latern variables
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables
`ifkappa`	take zero or non-zero value which indicates whether κ should be sampled
`scales`	a vector which indicates the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$ respectively
`phi.bound`	the upper and lower bound for ϕ
`initials`	a list which indicates the initial values for $(\beta, \sigma, \phi, \kappa)$ respectively
`MCMCinput`	a list of alternative settings
`partial`	a logical input which indicats whether partial posterior sampling should be used; only works for Y.family = "Poisson"
`famT`	take the value of 1, 2, or 3 which indicates the type of partial posterior sampling: 1 means "mean" diagnostic statistic is used, 2 means "maximum", and 3 means "minimum"; ignored if partial=FALSE
`n.chn`	the number of Markov chain sets that will be generated in parallel
`n.cores`	the number of CPUs that will be used to generate parallel Markov chains

Details

Essentially, this function runs `runMCMC` function simultaneously on different CPUs (if there are more than one CPU available) with different initial values. In the case the number of available CPUs is less than "n.chn", Markov chains will be put in a queue.

This function performs parallel computing with the help of `{multicore}` package. Be aware that `{multicore}` package currently is not available in Windows (so use `runMCMC.sf` instead).

Value

A list of length "n.chn" containing the result of each Markov chain. Each element is a list with elements:

`S.posterior`	a $n \times run$ matrix containing the posterior samples for latent variables
`m.posterior`	a $(p+1) \times run$ matrix (in case of p covariate variables) or a vector with length "run" (no covariate case), containing the posterior samples for β
`s.posterior`	a vector with length "run" containing the posterior samples for σ
`a.posterior`	a vector with length "run" containing the posterior samples for ϕ
`k.posterior`	a vector with length "run" containing the posterior samples for κ in the case that "ifkappa" is set to non-zero value
`AccRate`	a vector which indicates the acceptance rates

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`MCMCinput`, `runMCMC`, `runMCMC.sf`.

Examples

```
## Not run:
  require(multicore)
  data(datWeed)
  Y <- Weed[,3]
  loc <- unifLoc(Weed[,1:2])
  L <- rep(1, length(Y))
  input.Weed <- MCMCinput( run=1000, run.S=10, rho.family="rhoPowerExp",
          Y.family = "Poisson", ifkappa=0,
          scales=c(0.5, 3.5, 0.9, 0.6, 0.5),
          phi.bound=c(0.005, 1),
          initials=list(c(-1), 1, 0.1, 1) )
  res.prl <- runMCMC.multiChain(Y, L=L, loc=loc, X=NULL,
          MCMCinput = input.Weed, n.chn = 4, n.cores = 4)

## End(Not run)
```

runMCMC.sf *Perform Robust MCMC Algorithms for GLSM in Parallel*

Description

This function performs robust MCMC algorithms in a parallel way for generalized linear spatial models and generates posterior samples for latent variables and hyper-parameters.

Usage

```
runMCMC.sf(Y, L = 0, loc, X = NULL, run = 200, run.S = 1,
        rho.family = "rhoPowerExp", Y.family = "Poisson", ifkappa = 0,
        scales = c(0.5, 1.65^2 + 0.8, 0.8, 0.7, 0.15),
        phi.bound = c(0.005, 1), initials = list(c(1), 1.5, 0.2, 1),
        MCMCinput = NULL, partial = FALSE, famT = 1,
        n.chn = 2, n.cores = getOption("cores"), cluster.type="SOCK")
```

Arguments

`Y`	a vector of length n which indicates the response variables
`L`	a vector of length n; it indicates the time duration during which the Poisson counts are accumulated, or the total number of trials for Binomial response; if 0 is found in the vector, 1 will be used to replace all the values in the vector
`loc`	a $n \times 2$ matrix which indicates the coordinates of locations
`X`	a $n \times p$ covariate matrix; the default value "NULL" indicates no covariate
`run`	the number of iterations
`run.S`	the number of internal iterations for latern variables
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables
`ifkappa`	take zero or non-zero value which indicates whether κ should be sampled
`scales`	a vector which indicates the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$ respectively
`phi.bound`	the upper and lower bound for ϕ
`initials`	a list which indicates the initial values for $(\beta, \sigma, \phi, \kappa)$ respectively
`MCMCinput`	a list of alternative settings
`partial`	a logical input which indicats whether partial posterior sampling should be used; only works for Y.family = "Poisson"
`famT`	take the value of 1, 2, or 3 which indicates the type of partial posterior sampling: 1 means "mean" diagnostic statistic is used, 2 means "maximum", and 3 means "minimum"; ignored if partial=FALSE
`n.chn`	the number of Markov chain sets that will be generated in parallel
`n.cores`	the number of CPUs that will be used to generate parallel Markov chains
`cluster.type`	type of cluster to be used for parallel computing; can be "SOCK", "MPI", "PVM", or "NWS"

Details

Essentially, this function runs runMCMC function simultaneously on different CPUs (if there are more than one CPU available) with different initial values. In the case the number of available CPUs is less than "n.chn", Markov chains will be put in a queue.

This function performs parallel computing with the help of `{snow}` and `{snowfall}` packages.

Value

A list of length "n.chn" containing the result of each Markov chain. Each element is a list with elements:

`S.posterior`	a $n \times run$ matrix containing the posterior samples for latent variables
`m.posterior`	a $(p+1) \times run$ matrix (in case of p covariate variables) or a vector with length "run" (no covariate case), containing the posterior samples for β
`s.posterior`	a vector with length "run" containing the posterior samples for σ
`a.posterior`	a vector with length "run" containing the posterior samples for ϕ
`k.posterior`	a vector with length "run" containing the posterior samples for κ in the case that "ifkappa" is set to non-zero value
`AccRate`	a vector which indicates the acceptance rates

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`MCMCinput`, `runMCMC`, `runMCMC.multiChain`.

Examples

```
## Not run:
  require(snowfall)
  data(datWeed)
  Y <- Weed[,3]
  loc <- unifLoc(Weed[,1:2])
  L <- rep(1, length(Y))
  input.Weed <- MCMCinput( run=1000, run.S=10, rho.family="rhoPowerExp",
          Y.family = "Poisson", ifkappa=0,
          scales=c(0.5, 3.5, 0.9, 0.6, 0.5),
          phi.bound=c(0.005, 1),
          initials=list(c(-1), 1, 0.1, 1) )
  res.prl <- runMCMC.sf(Y, L=L, loc=loc, X=NULL,
          MCMCinput = input.Weed, n.chn = 4, n.cores = 4, cluster.type="SOCK")

## End(Not run)
```

simData — *Simulate Data Set from Generalized Linear Spatial Model on Given Locations*

Description

This function simulates a data set on given locations for Poisson Log-normal spatial model or Binomial Logistic-normal spatial model.

Usage

```
simData(loc, L = 0, X = NULL, beta = 0, cov.par,
        rho.family = "rhoPowerExp", Y.family = "Poisson")
```

Arguments

`loc`	a $n \times 2$ matrix which indicates the coordinates of given locations
`L`	a vector of length n; it indicates the time duration during which the Poisson counts are accumulated, or the total number of trials for Binomial response; if 0 is found in the vector, 1 will be used to replace all the values in the vector
`X`	a $n \times p$ covariate matrix; the default value "NULL" indicates no covariate
`beta`	a vector of length $(p+1)$ that indicates the coefficients
`cov.par`	a vector of length 3 that indicates the value of (σ, ϕ, κ)
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used
`Y.family`	take the value of `"Poisson"` or `"Binomial"` which indicates Poisson or Binomial distribution for response variables

Details

When using the powered exponential correlation function, note that $0 < \kappa <= 2$.

Value

A list with two elements:

`data`	a vector indicates the response variables
`latent`	a vector indicates the latent variables

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`locCircle`, `locGrid`, `locSquad`, `simData`, `plotData`.

Examples

```
## Not run:
  loc <- rbind(locCircle(0.3, 10),
               locCircle(0.6, 30),
               locCircle(1.0, 50)
               )
  dat <- simData(loc, cov.par = c(1, 0.1, 1))
  plotData(dat$data, loc)

## End(Not run)
```

`TexasCounty.boundary` *Data Set of Texas County Boundries*

Description

This data set contains the boundary information for all Texas countries.

Usage

```
data(TexasCounty_boundary)
```

Value

A list with 254 elements each of which contains the boundary information for one county.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`plotDataBD`, `TexasCounty.center`, `TexasCounty.population`.

Examples

```
## Not run:
data(TexasCounty_boundary)
length(TexasCounty.boundary); names(TexasCounty.boundary)
plotDataBD(TexasCounty.boundary, xlab = "Longitude", ylab = "Latitude")
text(TexasCounty.center[,2:3], names(TexasCounty.boundary), cex=0.4)

## End(Not run)
```

`TexasCounty.center` *Data Set of Texas County Centers*

Description

This data set contains the locations of centers for all Texas countries.

Usage

```
data(TexasCounty_center)
```

Value

A data.frame with 254 observations and 3 variables.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`TexasCounty.boundary`, `TexasCounty.population`.

Examples

```
## Not run:
data(TexasCounty_center)
str(TexasCounty.center)
plotDataBD(TexasCounty.boundary)
points(TexasCounty.center[,2:3], col=2, pch=3)

## End(Not run)
```

`TexasCounty.population` *Data Set of Texas County Population*

Description

This data set contains the population information for all Texas countries.

Usage

```
data(TexasCounty_population)
```

Details

Year: 2009

Source: U.S. Census Bureau, Small Area Estimates Branch, Poverty and Median Income Estimates

Value

A data.frame with 254 observations and 3 variables.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`TexasCounty.boundary`, `TexasCounty.center`.

Examples

```
## Not run:
data(TexasCounty_population)
str(TexasCounty.population)

## End(Not run)
```

`tranR` *Calculate Transformed Residuals for Observed Data*

Description

This function approximates transformed residuals for the observed data by using reference data.

Usage

```
tranR(Y.obs, Y.rep, discrete = FALSE)
```

Arguments

`Y.obs`	a vector which indicates the observed data set
`Y.rep`	a matrix which indicates the reference data sets
`discrete`	a logical value which indicates if the distribution of response variable is discrete

Value

A vector of transformed residuals.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`cdfU`, `plot_etran`, `e2dist`.

Examples

```
## Not run:
Yrep <- repYeb(N.sim=2000, loc, L, beta = 5, sigma = 1, phi = 0.1)
tranR(Y.obs, Y.rep)
## End(Not run)
```

U2Z *Convert Distance Matrix to Correlation Matrix*

Description

This function converts the distance matrix to correlation matrix.

Usage

```
U2Z(U, cov.par, rho.family = "rhoPowerExp")
```

Arguments

`U`	a $n \times n$ matrix which indicates the distance between locations
`cov.par`	a vector of length 3 that indicates the value of (σ, ϕ, κ)
`rho.family`	take the value of `"rhoPowerExp"` or `"rhoMatern"` which indicates the powered exponential or Matern correlation function is used

Details

When using the powered exponential correlation function, note that $0 < \kappa <= 2$.

Value

A $n \times n$ matrix with the element e_{ij} indicating the correlation between variables on the ith and jth locations.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`loc2U`, `rhoPowerExp`, `rhoMatern`.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  U <- loc2U(loc)
  Z <- U2Z(U, cov.par=c(0.5, 0.1, 1))

## End(Not run)
```

---

unifLoc *Scale Locations into A Unit Square*

Description

This function scales the coordinates of original locations to fit into a unit square.

Usage

```
unifLoc(loc, length=1)
```

Arguments

`loc`	a matrix of $n \times 2$ which indicates the x-y coordinates of the original locations
`length`	the edge length of the square

Value

A matrix of $n \times 2$ which indicates the x-y coordinates of scaled locations.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`locCircle`, `locGrid`, `locSquad`.

Examples

```
## Not run:
  loc <- locGrid(1, 2, 10, 5)
  plot(unifLoc(loc, scale=1))

## End(Not run)
```

`Weed` *Weed Data*

Description

This data set contains the Weed data set.

Usage

```
data(datWeed)
```

Details

The Weed data were collected at the Bjertorp farm in the south-west of Sweden. Weed counts of non-crop plants were observed at different locations, and camera recorded images were used to estimate the counts with the help of certain image analysis algorithm. Guillot, G., Loren, N., and Rudemo, M. (2009). Spatial prediction of weed intensities from exact count data and image-based estimates. Journal of Applied Statistics, 58, Part 4, 525-542.

Value

A data.frame with 100 observations and 4 variables.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`plotData`.

Examples

```
## Not run:
data(datWeed)
str(Weed)
plotData(Weed[,3], Weed[,1:2], Weed[,4], Weed[,1:2],
         xlab="Eastings", ylab="Northings")

## End(Not run)
```

Index