Package ‘myPackage’

August 28, 2011

Type Package

Title Estimation and Model Checking for GLSM

Version 1.1

Date 2011-05-05

Author Liang Jing

Maintainer Liang Jing <`ljing918@gmail.com`>

Description This package mainly deals with generalized linear spatial models. 1)It performs posterior sampling for parameter estimation, prediction, and model checking in hierarchical models with correlated latent variables; 2) C++ programs are seamlessly embedded to handle heavy computational tasks of Markov chain generation and large matrices computation; 3) Parallel computing techniques are implemented to further speed up estimation and prediction; 4) results are displayed by a combination of numerical and graphical summaries.

License GPL (>= 2)

LazyLoad yes

Depends R (>= 2.12.0), Rcpp (>= 0.9.4)

LinkingTo Rcpp, RcppGSL, RcppArmadillo

Suggests coda, multicore, distrEx

R topics documented:

baseline.dist 2
BMCT 3
cdfU 4
cutChain 5
d.base 6

e2dist 6
Earthquakes 7
findMode 8
locCircle 9
locSquad 10
MCMCinput 11
mixChain 12
myPackage_hello_world 13
plotACF 13
plotData 14
plotDataBD 15
plot_baseline 16
plot_etran 17
plot_pRPS 17
pOne 18
predY 19
pRPS 20
repYeb 21
repYpost 22
rongelap 23
runMCMC 24
runMCMC.multiChain 25
simData 26
TexasCounty.boundary 28
TexasCounty.center 28
TexasCounty.population 29
tranR 30
Weed 31

Index **32**

`baseline.dist` *Generate Baseline Samples for Standard Normal*

Description

This function generates baseline samples for standard normal distribution.

Usage

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

Arguments

`n`	the number of residuals.
`iter`	the number of baseline samples to generate.

Details

...

Value

A matrix of distances.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`tranR`, `e2dist`

Examples

```
## Not run:
# (time-consuming! only need to run once for date sets with same size)
d.base <- baseline.dist(length(etran), iter=1000)
## End(Not run)
```

`BMCT` *Bayesian Model Checking*

Description

This function conducts Bayesian model checking by comparing observed and replicated data sets and plots the distribution of dignostic statistic.

Usage

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

Arguments

`Y.obs`	the observed data set.
`Y.rep`	the replicated data sets.
`funcT`	the dignostic statistic.
`ifplot`	whether plot the distribution of dignostic statistic.

Details

...

Value

The checking result: 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 — *CDF Value for Observed Data*

Description

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

Usage

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

Arguments

Y.obs	the observed data set.
Y.rep	the replicated data sets.
discrete	if the random variable is discrete.

Details

...

Value

A vector of CDF values.

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

tranR

Examples

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

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

Description

This function takes the results from running MCMC algorithms and modifies the chains with burn-in and thining.

Usage

```
cutChain(res, chain.ind, burnin, thining)
```

Arguments

`res`	the posterior samples of latent variables and parameters.
`chain.ind`	the index of chains.
`burnin`	the number of samples for burn-in.
`thining`	the number of samples for thining.

Details

...

Value

A list of modified chains.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`runMCMC`, `runMCMC.multiChain`.

Examples

```
## Not run:
res <- runMCMC(Y, L=0, loc=loc, X=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

...

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`baseline.dist`, `plot_baseline`

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 different distances between the empirical distribution of transformed residuals and standard normal.

Usage

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

Arguments

`e.tran`	the transformed residuals.

Details

...

Value

A matrix of distances.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`tranR`, `plot_etran`

Examples

```
## Not run:
Yrep <- repYeb(N.sim=2000, loc, L, beta = 5, sigma = 1, phi = 0.1)
etran <- tranR(Y.obs, Y.rep, discrete = TRUE)
require(distrEx)
e2dist(etran)
## 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 *Find Mode for a Given Sample*

Description

This function finds the mode of empirical density functior for given sample.

Usage

```
findMode(x)
```

Arguments

x	a vector of samples.

Details

...

Value

The value of mode.

Author(s)

Liang Jing <ljing918@gmail.com>

Examples

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

## End(Not run)
```

`locCircle` *Simulate Circlular Locations*

Description

This function simulates a given number of locations equally distributed on the 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*2 matrix indicates the x-y coordinates of the locations.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

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

Examples

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

## End(Not run)
```

`locSquad` *Simulate Squared Locations*

Description

This function simulates a given number of locations equally distributed on the 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*np-4)*2 matrix indicates the x-y coordinates of the locations.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

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

Examples

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

## End(Not run)
```

MCMCinput *Settings for GLSM MCMC Algorithm*

Description

This function sets up the parameters used for GLSM MCMC algorithm.

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 latern variables.
`rho.family`	`"rhoPowerExp"` indicates powered exponential correlation function.
`Y.family`	`"Poisson"` indicates Poisson distribution for response variables
`ifkappa`	indicates whether κ should be sampled.
`scales`	the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$.
`phi.bound`	the upper and lower bound for ϕ.
`initials`	the initials values.

Details

...

Value

A list of setting parameters.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`runMCMC`, `runMCMC.multiChain`.

Examples

```
## Not run:
  input <- 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) )
  res <- runMCMC(Y, L=0, loc=loc, X=loc, MCMCinput = input )

## End(Not run)
```

`mixChain` *Mix Multiple MCMC Chains*

Description

This function mix multiple chains into one chain.

Usage

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

Arguments

`res.m.prl`	multiple chains of the posterior samples.

Details

...

Value

A list of mixed chains.

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.rong, 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)
```

`myPackage_hello_world` *Simple function using Rcpp*

Description

Simple function using Rcpp

Usage

```
myPackage_hello_world()
```

Examples

```
## Not run:
myPackage_hello_world()

## End(Not run)
```

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

Description

This function plots auto-correlation for latent variables.

Usage

```
plotACF(S.mcmc)
```

Arguments

`S.mcmc`	the posterior samples of latent variables.

Details

...

Value

No return value. A plot of auto-correlation.

Author(s)

Liang Jing <`ljing918@gmail.com`>

Examples

```
## Not run:
require(coda)
S.mcmc <- mcmc(t(res.m$S))
plotACF(S.mcmc)

## 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 response variables at different locations.
`loc, locp, loct`	$n \times 2$ matrix that indicates the coordinates of locations.
`col, colt`	the colors for three types of response variables.
`pch`	the shape.
`size`	the range of the size.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`simData`.

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)
```

`plotDataBD` *Plot Geostatistical Data*

Description

This function plots geostatistical data for up to three data sets and the given boundaries.

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 of boundaries.
`Y, Yp, Yt`	the response variables at different locations.
`loc, locp, loct`	$n \times 2$ matrix that indicates the coordinates of locations.
`col, colt`	the colors for three types of response variables.
`pch`	the shape.
`size`	the range of the size.

Author(s)

Liang Jing <`ljing918@gmail.com`>

See Also

`simData`, `plotData`.

Examples

```
## Not run:

data(TexasCounty_boundary)
plotDataBD(TexasCounty.boundary)

## End(Not run)
```

plot_baseline *Plot Baseline Samples*

Description

This function plots the baseline samples.

Usage

```
plot_baseline(res.in, dist.name)
```

Arguments

`res.in`	the baseline samples.
`dist.name`	the name of distance.

Details

...

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

`e2dist`, `baseline.dist`

Examples

```
## Not run:
# (time-consuming! only need to run once for date sets with same size)
d.base <- baseline.dist(length(etran), iter=1000)
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` *Calculate Transformed Residuals for Observed Data*

Description

This function plots transformed residuals in different type of plots.

Usage

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

Arguments

`e.tran`	the transformed residuals.
`fig`	indicates which plot(s) to plot.

Details

...

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`tranR`

Examples

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

`plot_pRPS` *Plot for Observed and Replicated Statistics*

Description

This function plots the observed statistic and the empirical density of replicated statistics.

Usage

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

Arguments

`T.obs`	the observed statistic.
`T.rep`	the replicated statistics.
`nm`	the label of statistics.

Details

...

Value

A plot.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`BMCT`, `pRPS`

Examples

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

`pOne` *Calculate One-side P-value*

Description

This function calculates one-side p-value for observed data based on the baseline samples of distances.

Usage

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

Arguments

`d.obs`	the distance for observed data.
`d.base`	the baseline samples of distances.

Details

...

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

`e2dist`, `baseline.dist`

Examples

```
## Not run:
pOne(d.obs, d.base)
## End(Not run)
```

`predY` *Predict for Given Locations*

Description

This function generates posterior predictive samples of latent and reponse variables for predicted locations.

Usage

```
predY(res.m, loc, locp, X = NULL, Xp = NULL, Lp = 0, k = 1,
      rho.family = "rhoPowerExp", Y.family = "Poisson")
```

Arguments

`res.m`	the posterior samples of latent variables and parameters at observed locations.
`loc`	the observed locations.
`locp`	the predicted locations.
`X`	the covariate matrix for observed locations.
`Xp`	the covariate matrix for predicted locations.
`Lp`	the time duration for predicted locations.
`k`	κ for correlation function.
`rho.family`	`"rhoPowerExp"` indicates powered exponential correlation function.
`Y.family`	the distribution for response variables.

Details

...

Value

A list of posterior predictive samples.

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

runMCMC, runMCMC.multiChain.

Examples

```
## Not run:
locp <- matrix(runif(200),,2)
Ypred <- predY(res.m, loc, locp, X=loc, Xp=locp, k=1, rho.family="rhoPowerExp")
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 of RPS by comparing observed and replicated statistics.

Usage

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

Arguments

T.obs	the observed statistic.
T.rep	the replicated statistics.

Details

...

Value

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 *Generation of Replicated Data Sets*

Description

This function generats replicated data sets based on estimated parameters.

Usage

```
repYeb(N.sim, loc, L, X = NULL, rho.family = "rhoPowerExp",
    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`	$n \times 2$ matrix that indicates the coordinates of locations.
`L`	the time duration for all the locations.
`X`	the covariate matrix.
`rho.family`	the correlation function to be used for latent variables.
`res.m`	the posterior samples to be used for parameter estimation.
`est`	the way to estimate.
`beta`	alternative estimation for β
`sigma`	alternative estimation for σ
`phi`	alternative estimation for ϕ
`k`	alternative estimation for κ

Details

...

Value

A matrix of replicated data sets.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

repYpost

Examples

```
## Not run:
# Estimation from posterior samples
Yrep.eb <- repYeb(N.sim=2000, loc, L, res.m, est = "mode")
# Given estimated parameters
Yrep.eb <- repYeb(N.sim=2000, loc, L, beta = 5, sigma = 1, phi = 0.1,
    k = 1)

## End(Not run)
```

---

repYpost *Generation of Replicated Data Sets*

Description

This function generats replicated data sets based on posterior samples of latern variables.

Usage

```
repYpost(res.m, L)
```

Arguments

`res.m`	the posterior samples of latern variables to be used for generation.
`L`	the time duration for all the locations.

Details

...

Value

A matrix of replicated data sets.

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

repYeb

Examples

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

`rongelap` *Data Set of Rongelap Island*

Description

This data set contains the Rongelap data.

Usage

```
data(datRongelap)
```

Details

...

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`plotData`.

Examples

```
## Not run:

data(datRongelap)
str(rongelap)
plotData(rongelap$data, rongelap$coords)
lines(rongelap$borders)

## End(Not run)
```

runMCMC *Generate Posterior Samples for GLSM with MCMC Algorithm*

Description

This function generates posterior samples for GLSM.

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`	the response variables at given locations.
`L`	the time duration for each location.
`loc`	$n \times 2$ matrix that indicates the coordinates of locations.
`X`	a $n \times p$ covariate matrix.
`run`	the number of iterations.
`run.S`	the number of internal iterations for latern variables.
`rho.family`	`"rhoPowerExp"` indicates powered exponential correlation function.
`Y.family`	`"Poisson"` indicates Poisson distribution for response variables
`ifkappa`	indicates whether κ should be sampled.
`scales`	the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$.
`phi.bound`	the upper and lower bound for ϕ.
`initials`	the initials values.
`MCMCinput`	the list of alternative settings.
`partial`	indicats whether partial posterior sampling should be used.
`famT`	indicate the type of posterior sampleing.

Details

...

Value

A list containing the posterior samples for $(S, \beta, \sigma, \phi, \kappa)$ and the acceptance rates.

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

MCMCinput, runMCMC.multiChain.

Examples

```
## Not run:
  input <- 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) )
  res <- runMCMC(Y, L=0, loc=loc, X=loc, MCMCinput = input )

## End(Not run)
```

runMCMC.multiChain *Generate Posterior Samples for GLSM with MCMC Algorithm*

Description

This function generates multiple Markov chains of posterior samples for GLSM.

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)
```

Arguments

Y	the response variables at given locations.
L	the time duration for each location.
loc	$n \times 2$ matrix that indicates the coordinates of locations.
X	a $n \times p$ covariate matrix.
run	the number of iterations.
run.S	the number of internal iterations for latern variables.
rho.family	"rhoPowerExp" indicates powered exponential correlation function.
Y.family	"Poisson" indicates Poisson distribution for response variables
ifkappa	indicates whether κ should be sampled.

`scales`	the tuning parameters for $(S, \beta, \sigma, \phi, \kappa)$.
`phi.bound`	the upper and lower bound for ϕ.
`initials`	the initials values.
`MCMCinput`	the list of alternative settings.
`partial`	indicats whether partial posterior sampling should be used.
`famT`	indicate the type of posterior sampleing.
`n.chn`	the number of running chains.

Details

...

Value

A list containing the posterior samples for $(S, \beta, \sigma, \phi, \kappa)$ and the acceptance rates.

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`MCMCinput`, `runMCMC`.

Examples

```
## Not run:
  require(multicore)
  options(cores=5)
  input <- 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) )
  res.prl <- runMCMC.multiChain(Y, L=0, loc=loc, X=loc, MCMCinput = input, n.chn = 5 )

## End(Not run)
```

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

Description

This function simulates a data set from Generalized Linear Spatial Model on given locations.

Usage

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

Arguments

`loc`	$n \times 2$ matrix that indicates the coordinates of given locations.
`L`	a vector of length n that indicates the time duration for each location.
`X`	a n*p covariate matrix.
`beta`	a vector of length (p+1) that indicates the coefficients
`cov.par`	a vector of length 3 that indicates (σ, ϕ, κ)
`rho.family`	`"rhoPowerExp"` indicates powered exponential correlation function.
`Y.family`	`"Poisson"` indicates Poisson distribution for response variables

Details

...

Value

A list with two elements:

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

Author(s)

Liang Jing `<ljing918@gmail.com>`

See Also

`locCircle`, `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)
```

Details

...

Author(s)

Liang Jing <ljing918@gmail.com>

See Also

plotDataBD, TexasCounty.center, TexasCounty.population.

Examples

```
## Not run:

data(TexasCounty_boundary)
length(TexasCounty.boundary)
plotDataBD(TexasCounty.boundary)

## End(Not run)
```

TexasCounty.center *Data Set of Texas County Centers*

Description

This data set contains the center locations for all Texas countries.

Usage

```
data(TexasCounty_center)
```

Details

...

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

...

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 replicated data sets.

Usage

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

Arguments

`Y.obs`	the observed data set.
`Y.rep`	the replicated data sets.
`discrete`	if the random variable is discrete.

Details

...

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, discrete = TRUE)
## End(Not run)
```

`Weed` *Data Set of Weed*

Description

This data set contains the Weed data.

Usage

```
data(datWeed)
```

Details

...

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])

## End(Not run)
```

Index

∗Topic **ACF**
    plotACF, 13
∗Topic **Data**
    Earthquakes, 7
    plotDataBD, 15
    rongelap, 23
    TexasCounty.boundary, 28
    TexasCounty.center, 28
    TexasCounty.population, 29
    Weed, 31
∗Topic **MCMC**
    cutChain, 5
    MCMCinput, 11
    mixChain, 12
    runMCMC, 24
    runMCMC.multiChain, 25
∗Topic **Model Checking**
    baseline.dist, 2
    BMCT, 3
    cdfU, 4
    d.base, 6
    e2dist, 6
    plot_baseline, 16
    plot_etran, 17
    plot_pRPS, 17
    pOne, 18
    pRPS, 20
    tranR, 30
∗Topic **Replicated Data**
    repYeb, 21
    repYpost, 22
∗Topic **location**
    locCircle, 9
    locSquad, 10
∗Topic **mode**
    findMode, 8
∗Topic **plot**
    plotData, 14
∗Topic **prediction**
    predY, 19
∗Topic **spatial**
    simData, 26

baseline.dist, 2, *6*, *16*, *19*
BMCT, 3, *18*, *20*

cdfU, 4, *30*
cutChain, 5, *12*

d.base, 6

e2dist, *3*, 6, *16*, *19*, *30*
Earthquakes, 7

findMode, 8

locCircle, 9, *10*, *27*
locSquad, *9*, 10, *27*

MCMCinput, 11, *25, 26*
mixChain, 12
myPackage_hello_world, 13

plot_baseline, *6*, 16
plot_etran, *7*, 17, *30*
plot_pRPS, *4*, 17, *20*
plotACF, 13
plotData, *8–10*, 14, *15*, *23*, *27*, *31*
plotDataBD, 15, *28*
pOne, 18
predY, 19
pRPS, *4*, *18*, 20

repYeb, *4*, 21, *22*
repYpost, *4*, 22, *22*
rongelap, 23
runMCMC, *5*, *11*, *20*, 24, *26*
runMCMC.multiChain, *5*, *11, 12*, *20*, 25, *25*

simData, *9, 10*, *14, 15*, 26, *27*

TexasCounty.boundary, 28, *29*
TexasCounty.center, 28, *28, 29*
TexasCounty.population, *28*, 29, *29*
tranR, *3, 4*, *7*, *17*, 30

Weed, 31