Package 'miscF'

June 14, 2013

Title Miscellaneous Functions

Version 0.1-2

Author Dai Feng

Description Various functions for random number generation, density estimation, classification, curve fitting, and spatial data analysis.

Maintainer Dai Feng <dai_feng@merck.com>

Depends R (>= 2.15.0), MCMCpack (>= 1.2-4), mvtnorm(>= 0.9-9992), Rcpp (>= 0.10.3), RcppArmadillo (>= 0.3.810.2)

Suggests mixAK (>= 2.6)

LinkingTo Rcpp, RcppArmadillo

License GPL

NeedsCompilation yes

Repository CRAN

Date/Publication 2013-06-14 07:00:23

R topics documented:

curve.polynomial.rjmcmc *Curve Fitting Using Piecewise Polynomials with Unknown Number and Location of Knots*

Description

Fit a variety of curves by a sequence of piecewise polynomials. The number and location of knots are determined by the Reversible Jump MCMC method.

Usage

```
curve.polynomial.rjmcmc(y, x, lambda, l, l0, c=0.4,
                        gamma.shape=1e-3, gamma.rate=1e-3,
                        maxit=10000, err=1e-8, verbose=TRUE)
```

Arguments

y	a vector of values of a response variable.
x	a vector of values of the corresponding explanatory variable.
lambda	the parameter of the Poisson prior for the number of knots.
l	the order of polynomials.
l0	the degree of continuity at the knots.
c	the parameter controlling the rate of changing dimension. The default value is 0.4.
gamma.shape	the parameter shape of the gamma prior for the error precision. The default value is 1e-3.
gamma.rate	the parameter rate of the gamma prior for the error precision. The default value is 1e-3.
maxit	the maximum number of iterations. The default value is 10000.
err	the iteration stops when consecutive difference in percentage of mean-squared error reaches this bound. The default value is 1e-8.
verbose	logical. If TRUE, then indicate the level of output after every 1000 iterations. The default is TRUE.

Details

The method is based on Denison et al. (1998). It can be used to fit both smooth and unsmooth curves. When both l0 and l are set to 3, it fits curves using cubic spline.

Value

knots.save	a list of location of knots. One component per iteration.
betas.save	a list of estimates of regression parameters βs. One component per iteration.
fitted.save	a matrix of fitted values. One column per iteration.
sigma2.save	a vector of estimate of error variance. One component per iteration.

Note

The factor $\frac{1}{\sqrt{n}}$ was added in the likelihood ratio to penalize the ratio for dimensionality as suggested in Dimatteo et al. (2001).

References

Denison, D. G. T., Mallick, B. K., Smith, A. F. M.(1998) Automatic Bayesian Curve Fitting *JRSSB* **vol. 60, no. 2** 333-350

Dimatteo, I., Genovese, C. R., Kass, R. E.(2001) Bayesian Curve-fitting with Free-knot Splines *Biometrika* **vol. 88, no. 4** 1055-1071

See Also

sm.spline

Examples

```
## Not run:
   #Example 1: smooth curve
   #example 3.1. (b) in Denison et al.(1998)

   x <- runif(200)
   xx <- -2 + 4*x
   y.truth <- sin(2*xx) + 2*exp(-16*xx^2)
   y <- y.truth + rnorm(200, mean=0, sd=0.3)

   results <- curve.polynomial.rjmcmc(y, x, lambda=1, l=2, l0=1)

   plot(sort(x), y.truth[order(x)], type="l")
   lines(sort(x), rowMeans(results$fitted.save), col="red")

   #Example 2: unsmooth curve
   #blocks in Denison et al.(1998)
   tj <- c(0.1, 0.13, 0.15, 0.23, 0.25, 0.4, 0.44, 0.65, 0.76, 0.78, 0.81)
   hj <- c(4, -5, 3, -4, 5, -4.2, 2.1, 4.3, -3.1, 2.1, -4.2)

   t <- seq(0, 1, len=2048)
   Ktmtj <- outer(t, tj, function(a,b) ifelse(a-b > 0, 1, -1))
   ft <- rowSums(Ktmtj %*% diag(hj))
   x <- t
   y <- ft + rnorm(2048, 0, 1)

   results <- curve.polynomial.rjmcmc(y, x, lambda=5, l=2, l0=1)
```

```
  plot(x, ft, type="s")
  lines(x, rowMeans(results$fitted.save), col="red")

  #Example 3: unsmooth curve
  #bumps in Denison et al.(1998)
  tj <- c(0.1, 0.13, 0.15, 0.23, 0.25, 0.4, 0.44, 0.65, 0.76, 0.78, 0.81)
  hj <- c(4, 5, 3, 4, 5, 4.2, 2.1, 4.3, 3.1, 5.1, 4.2)*10
  wj <- c(0.005, 0.005, 0.006, 0.01, 0.01, 0.03, 0.01, 0.01, 0.005, 0.008, 0.005)

  t <- seq(0, 1, len=2048)
  ft <- rowSums((1 + abs(outer(t, tj ,"-") %*% diag(1/wj)))^(-4) %*% diag(hj))
  y <- ft + rnorm(2048, 0, 1)

  results <- curve.polynomial.rjmcmc(y, t, lambda=5, l=2, l0=1)

  plot(t, ft, type="s")
  lines(t, rowMeans(results$fitted.save), col="red")

## End(Not run)
```

`mvn.bayes` *Estimate the Parameters of a Multivariate Normal Model by the Bayesian Methods*

Description

Estimate the parameters of a multivariate normal model under different priors.

Usage

```
mvn.bayes(X, nsim, prior=c("Jeffreys", "Conjugate"))
```

Arguments

`X`	a matrix of observations with one subject per row.
`nsim`	number of simulations.
`prior`	a character string specifying the prior distribution. It must be either "Jeffreys" or "Conjugate" and may be abbreviated. The default is "Jeffreys".

Details

Both the Jeffreys prior and normal-inverse-Wishart conjugate prior are available. The conjugate prior of variance covariance matrix is inverse-Wishart. To use a noninformative proper prior, the degree of freedom of Wishart prior was set as the number of dimensions and the scale matrix was chosen based on the unbiased estimate. The number of prior measurements was taken as one and the prior mean was set as its unbiased estimate.

Value

Mu.save	a matrix of mean vector of the model, one row per iteration.
Sigma.save	a three dimensional array of variance of the model. Sigma.save[,,i] is the result from the ith iteration.

Note

When the number of dimensions is two, under Jeffreys prior, it is straightforward to obtain independent samples from the posteriors.

References

Berger, J. O., Sun, D. (2008) Objective Priors for the Bivariate Normal Model. *The Annals of Statistics* **36** 963-982.

Gelman, A., Carlin, J. B., Stern, H. S., Rubin, D. B. (2003) Bayesian Data Analysis. 2nd ed. London: Chapman and Hall

Sun, D., Berger, J. O. (2009) Objective Priors for the Multivariate Normal Model. In Bayesian Statistics 8, Ed. J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith and M. West. Oxford: Oxford University Press.

Examples

```
Sigma <- matrix(c(100, 0.99*sqrt(100*100),
                  0.99*sqrt(100*100), 100),
                  nrow=2)
X <- mvrnorm(1000, c(100, 100), Sigma)
result <- mvn.bayes(X, 10000)
Mu <- colMeans(result$Mu.save)
Sigma <- apply(result$Sigma.save, c(1,2), mean)
```

---

mvn.ub *Unbiased Estimate of Parameters of a Multivariate Normal Distribution*

Description

Obtain the Unbiased Estimate of Parameters of a Multivariate Normal Distribution.

Usage

```
mvn.ub(X)
```

Arguments

X	a matrix of observations with one subject per row.

Value

hat.Mu	unbiased estimate of mean.
hat.Sigma	unbiased estimate of variance.

Examples

```
Sigma <- matrix(c(100, 0.99*sqrt(100*100),
                  0.99*sqrt(100*100), 100),
                  nrow=2)
X <- mvrnorm(1000, c(100, 100), Sigma)
result <- mvn.ub(X)
```

---

mvt.ecme *Estimate Parameters of a Multivariate t Distribution Using the ECME Algorithm*

Description

Use the Expectation/Conditional Maximization Either (ECME) algorithm to obtain estimate of parameters of a multivariate t distribution.

Usage

```
mvt.ecme(X, lower.v, upper.v, err=1e-4)
```

Arguments

X	a matrix of observations with one subject per row.
lower.v	lower bound of degrees of freedom (df).
upper.v	upper bound of df.
err	the iteration stops when consecutive difference in percentage of df reaches this bound. The default value is 1e-4.

Details

They are number of forms of the generalization of the univariate student-t distribution to multivariate cases. This function adopts the widely used representation as a scale mixture of normal distributions.

To obtain the estimate, the algorithm adopted is the Expectation/Conditional Maximization Either (ECME), which extends the Expectation/Conditional Maximization (ECM) algorithm by allowing CM-steps to maximize either the constrained expected complete-data log-likelihood, as with ECM, or the correspondingly constrained actual log-likelihood function.

Value

Mu	estimate of location.
Sigma	estimate of scale matrix.
v	estimate of df.

References

Chuanhai Liu (1994) Statistical Analysis Using the Multivariate t Distribution *Ph. D. Dissertation, Harvard University*

Examples

```
mu1 <- mu2 <- sigma12 <- sigma22 <- 100
rho12 <- 0.7
Sigma <- matrix(c(sigma12, rho12*sqrt(sigma12*sigma22),
                  rho12*sqrt(sigma12*sigma22), sigma22),
                nrow=2)
k <- 5
N <- 100
require(mvtnorm)
X <- rmvt(N, sigma=Sigma, df=k, delta=c(mu1, mu2))

result <- mvt.ecme(X, 3, 300)
result$Mu
result$Sigma
result$v
```

---

mvt.mcmc *Estimate Parameters of a Multivariate t Distribution Using the MCMC*

Description

Use the MCMC to obtain estimate of parameters of a multivariate t distribution.

Usage

```
mvt.mcmc(X, niter, prior.lower.v, prior.upper.v,
         prior.Mu0=rep(0, ncol(X)), prior.Sigma0=diag(10000, ncol(X)),
         prior.p=ncol(X), prior.V=diag(1, ncol(X)),
         initial.v=NULL, initial.Sigma=NULL)
```

Arguments

X	a matrix of observations with one subject per row.
niter	number of iterations.
prior.lower.v	lower bound of degrees of freedom (df).
prior.upper.v	upper bound of df.

prior.Mu0	mean vector of multivariate normal prior of the location. The default value is 0.
prior.Sigma0	variance matrix of multivariate normal prior of the location. The default value is a diagonal matrix with diagonal entries equal to 10000.
prior.p	the df of wishart prior of inverse of the scale matrix. The default value is dimensions of observation.
prior.V	the scale matrix of wishart prior of inverse of the scale matrix. The default value is identity matrix.
initial.v	the initial value of the df. The default is NULL. For the default, the value will be generated by using the ECME Algorithm.
initial.Sigma	the initial value of the scale matrix. The default is NULL. For the default, the value will be generated by using the ECME Algorithm.

Details

To generate samples from the full conditional distribution of df, the slice sampling was used and the code was adapted from http://www.cs.toronto.edu/~radford/ftp/slice-R-prog.

Value

Mu.save	a matrix of locations of the distribution, one row per iteration.
Sigma.save	a three dimensional array of scale matrix of the distribution. Sigma.save[,,i] is the result from the ith iteration.
v.save	a vector of df of the distribution, one component per iteration.

See Also

mvt.ecme

Examples

```
## Not run:
  mu1 <- mu2 <- sigma12 <- sigma22 <- 100
  rho12 <- 0.9
  Sigma <- matrix(c(sigma12, rho12*sqrt(sigma12*sigma22),
                    rho12*sqrt(sigma12*sigma22), sigma22),
                  nrow=2)
  k <- 8
  N <- 100
  X <- rmvt(N, sigma=Sigma, df=k, delta=c(mu1, mu2))

  result <- mvt.mcmc(X, 10000, 4, 25)
  colMeans(result$Mu.save)
  apply(result$Sigma.save, c(1,2), mean)
  mean(result$v.save)

## End(Not run)
```

rMultinom *Generate Random Samples from Different Multinomial Distributions with the Same Number of Classes*

Description

Generate random samples from multinomial distributions with the same number of classes but different event probabilities.

Usage

```
rMultinom(p)
```

Arguments

p	matrix with each row specifying the probabilities for different classes of each sample.

Details

This function vectorizes the generation of random samples from different multinomial distributions by the inversion of CDF method.

Value

Random samples from multinomial distributions.

See Also

rmultinom

Examples

```
#Example 1: Generate 100 random samples from multinomial distributions
#           with 3 classes and different event probabilities.
p1 <- runif(100)
p2  <- runif(100, 0, 1-p1)
p3 <- 1-p1-p2
x <- rMultinom(p=cbind(p1, p2, p3))
```

`spatail.lme.mcmc` *Spatial Modeling by a Bayesian Hierarchical Linear Mixed-effects Model*

Description

A linear mixed-effects model that combines unstructured variance/covariance matrix for inter-regional (long-range) correlations and an exchangeable correlation structure for intra-regional (short-range) correlations. Estimation is performed using the Gibbs sampling.

Usage

```
spatial.lme.mcmc(spatialMat, nlr, nsweep, verbose=TRUE)
```

Arguments

`spatialMat`	a matrix of observations with one subject per column and one location per row. For each subject, the observations are arranged one location after another.
`nlr`	a vector of number of locations within each region. One component per region.
`nsweep`	the number of iterations.
`verbose`	logical. If `TRUE`, then indicate the level of output after every 1000 iterations. The default is `TRUE`.

Details

The function was proposed to study the fMRI data. The original MATLAB code written by DuBois Bowman and Brian Caffo can be found at: http://www.biostat.jhsph.edu/~bcaffo/downloads/clusterBayes.m. Instead of stacking the data from two conditions, the R version fits the model for one condition and the user needs to use the function multiple times for separate conditions.

The initial values are obtained based on sample moments. The hyper-parameters for the prior distributions of the intra-regional variances and variances of locations' means are set up in the way that the mean is equal to the sample mean and the variance is large.

Value

`mu.save`	a matrix of means at every location. One column per iteration.
`sigma2.save`	a matrix of intra-regional variances. One column per iteration.
`lambda2.save`	a matrix of variances of locations' means within regions. One column per iteration.
`Gamma.save`	a matrix of inter-regional variance/covariance matrix. One column per iteration and within each column the elements of the variance/covariance matrix are arranged column-wise.

Note

There seemed to be no easy way to use lmer or lme to fit the variance/covariance structure in this model and *SAS proc mixed* failed for certain cases.

References

Brian Caffo, DuBois Bowman, Lynn Eberly and Susan Spear Bassett (2009) A Markov Chain Monte Carlo Based Analysis of a Multilevel Model for Functional MRI Data *Handbook of Markov Chain Monte Carlo*

F. DuBois Bowman, Brian Caffo, Susan Spear Bassett, and Clinton Kilts (2008) A Bayesian Hierarchical Framework for Spatial Modeling of fMRI Data *Neuroimage* **vol. 39, no. 1** 146-156

Examples

```
## Not run:
    #simulate the data
    ns=100; nr=2; nlr <- c(20, 20)
    mu0 <- c(0, 0)
    sigma2 <- c(1., 1.)
    Gamma <- matrix(c(3, 0, 0, 3), nrow=2)

    sample <- matrix(0, nrow=sum(nlr), ncol=ns)
    for(i in 1:ns){
        alpha <- mvrnorm(1, rep(0, nr), Gamma)
        sampleR <- NULL
        for(g in 1:nr){
            beta <- rnorm(nlr[g], mean=alpha[g] + mu0[g], sd=sqrt(sigma2[g]))
            sampleR <- c(sampleR, beta)
        }
        sample[,i] <- sampleR
    }

    #run mcmc
    mcmc.result <- spatial.lme.mcmc(sample, nlr, 10000)

    #check the results
    Gamma <- mcmc.result$Gamma.save
    sigma2 <- mcmc.result$sigma2.save
    mu <- mcmc.result$mu.save
    matrix(rowMeans(Gamma), nr, nr)
    apply(sigma2, 1, function(x) quantile(x, prob=c(0.025, 0.5, 0.975)))
    summary(rowMeans(mu[1:nlr[1],]))
    summary(rowMeans(mu[(nlr[1]+1):sum(nlr),]))


## End(Not run)
```

`uvnm.rjmcmc` *Univariate Normal Mixture (UVNM) Model with Unknown Number of Components*

Description

Estimate the parameters of an univariate normal mixture model including the number of components using the Reversible Jump MCMC method. It can be used for density estimation and/or classification.

Usage

```
uvnm.rjmcmc(y, nsweep, kmax, k, w, mu, sigma2, Z,
            delta=1, xi=NULL, kappa=NULL, alpha=2,
            beta=NULL, g=0.2, h=NULL, verbose=TRUE)
```

Arguments

`y`	a vector of observations.
`nsweep`	number of sweeps. One sweep has six moves to update parameters including the number of components.
`kmax`	the maximum number of components.
`k`	initial value of number of components.
`w`	initial values of weights of different components.
`mu`	initial values of means of different components.
`sigma2`	initial values of variances of different components.
`Z`	initial values of allocations of all observations.
`delta`	the parameter of the prior distribution of `w`, the symmetric Dirichlet distribution. The default value is one.
`xi`	the mean of the prior distribution of means of components. By taking the default value `NULL`, it is set as the median of the data `y` internally.
`kappa`	the precision of the prior distribution of means of components. By taking the default value `NULL`, it is set as 1/R^2 internally, where R is the range of `y`.
`alpha`	the parameter shape of the prior distribution of precision of components. The default values is 2.
`beta`	the parameter rate of the prior distribution of precision of components. By taking the default value `NULL`, it is set as a number generated randomly from a gamma distribution with shape `g` and rate `h`.
`g`	the parameter shape of the gamma distribution for `beta`. The default value is 0.2.
`h`	the parameter rate of the gamma distribution for `beta`. By taking the default value `NULL`, it is set as 10/R^2 internally, where R is the range of `y`.
`verbose`	logical. If `TRUE`, then indicate the level of output after every 1000 sweeps. The default is `TRUE`.

Details

Estimate the parameters of a univariate normal mixture model with flexible number of components using the Reversible Jump MCMC method in Richardson and Green (1997).

Value

`k.save`	a vector of number of components.
`w.save`	weights of the UVNM, one component of the list per sweep.
`mu.save`	means of the UVNM, one component of the list per sweep.
`sigma2.save`	variances of the UVNM, one component of the list per sweep.
`Z.save`	a matrix of allocation, one column per sweep.

Note

The error in equation (12) of Richardson and Green (1997) was corrected based on Richardson and Green (1998).

References

Sylvia Richardson and Peter J. Green (1997) On Bayesian Analysis of Mixtures with an Unknown Number of Components *JRSSB* **vol. 59, no. 4** 731-792

Sylvia Richardson and Peter J. Green (1998) Corrigendum: On Bayesian Analysis of Mixtures with an Unknown Number of Components *JRSSB* **vol. 60, no. 3** 661

See Also

NMixMCMC

Examples

```
## Not run:
    require(mixAK)
    data(Acidity)
    y <- Acidity
    w <- c(0.50, 0.17, 0.33)
    mu <- c(4, 5, 6)
    sigma2 <- c(0.08, 0.10, 0.14)
    Z <- do.call(cbind, lapply(1:3, function(i)
                                    w[i]*dnorm(y, mu[i], sqrt(sigma2[i]))))
    Z <- apply(Z, 1, function(x) which(x==max(x))[1])

    result <- uvnm.rjmcmc(y, nsweep=200000, kmax=30, k=3,
                          w, mu, sigma2, Z)

    ksave <- result$k.save
    round(table(ksave[-(1:100000)])/100000,4)

    #conditional density estimation
```

```
    focus.k <- 3
    pick.k <- which(ksave==focus.k)
    w <- unlist(result$w.save[pick.k])
    mu <- unlist(result$mu.save[pick.k])
    sigma2 <- unlist(result$sigma2.save[pick.k])
    den.estimate <- rep(w, each=length(y)) *
                    dnorm(rep(y, length(w)), mean=rep(mu, each=length(y)),
                          sd=rep(sqrt(sigma2), each=length(y)))
    den.estimate <- rowMeans(matrix(den.estimate, nrow=length(y)))*focus.k

    #within-sample classification
    class <- apply(result$Z.save[,pick.k], 1,
                                function(x) c(sum(x==1), sum(x==2), sum(x==3)))
    class <- max.col(t(class))

    #visualize the results
    hist(y, freq=FALSE, breaks=20, axes=FALSE, ylim=c(-0.3, 1),
         main="Density Estimation and Classification", ylab="")
    axis(2, at=c(-(3:1)/10, seq(0,10,2)/10), labels=c(3:1, seq(0,10,2)/10),
         font=2)
    lines(sort(y), den.estimate[order(y)], col="red")
    for(i in 1:3){
        points(y[class==i], rep(-i/10, length(y[class==i])), col=i, pch=i)
    }
    mtext("Density", 2, at=0.5, line=2)
    mtext("Classification", 2, at=-0.15, line=2)

## End(Not run)
```

Index