Package 'spBayesSurv'

July 2, 2014

Type Package

Title Spatial Copula Modelling to Bayesian Survival Analysis

Version 1.0.0

Date 2014-06-10

Author Haiming Zhou <zhouh@email.sc.edu> and Tim Hanson <hansont@stat.sc.edu>

Maintainer Haiming Zhou <zhouh@email.sc.edu>

Description This package provides Bayesian model fitting for several survival models including spatial copula linear dependent Dirichlet process mixture model, anova Dirichlet process mixture model,proportional hazards model and marginal spatial proportional hazards model.

License GPL (>= 2)

Depends R (>= 3.0.2)

Imports Rcpp (>= 0.11.1), survival

LinkingTo Rcpp, RcppArmadillo (>= 0.4.300.0)

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-06-12 07:40:12

R topics documented:

spBayesSurv-package *Spatial Copula Modelling to Bayesian Survival Analysis*

Description

This package provides Bayesian model fitting for several survival models including spatial copula linear dependent Dirichlet process mixture model, anova Dirichlet process mixture model, proportional hazards model and marginal spatial proportional hazards model.

Details

Package:	spBayesSurv
Type:	Package
Version:	1.0.0
Date:	2014-06-10
License:	GPL (>= 2)

This package provides functions for fitting several Bayesian survival models including `spCopulaDDP` for spatial copula linear dependent Dirichlet process mixture model, `anovaDDP` for anova Dirichlet process mixture, `indeptCoxph` for proportional hazards model, and `spCopulaCox` for marginal spatial proportional hazards model.

Author(s)

Haiming Zhou <zhouh@email.sc.edu> and Tim Hanson <hansont@stat.sc.edu>

References

De Iorio, M., Johnson, W. O., Mueller, P., and Rosner, G. L. (2009). Bayesian nonparametric nonproportional hazards survival modeling. Biometrics, 65(3), 762-771.

Zhou, H., Hanson, T. and Knapp, R. (2014+). A Spatial Copula Approach to Fully Bayesian Nonparametric Survival Analysis. In preparation for publication.

anovaDDP *Bayesian Nonparametric Nonproportional Hazards Survival Modeling*

Description

This function fits an ANOVA DDP model for survival analysis of right censored time-to-event data.

Usage

```
anovaDDP(y, delta, x=NULL, prediction, prior, mcmc, state,
         data=sys.frame(sys.parent()), na.action=na.fail, work.dir=NULL)
```

Arguments

y	an n by 1 vector giving the log survival times.
delta	an n by 1 vector indicating whether it is right censored (=0) or not (=1).
x	an n by p matrix of covariates without intercept. The default is NULL, indicating no covariates included.
prediction	a list giving the information used to obtain conditional inferences. The list includes the following elements: xpred giving the n by p covariates matrix, used for prediction.
prior	a list giving the prior information. The list includes the following parameter: N an integer giving the truncation of the Dirichlet process, a0 and b0 giving the hyperparameters for prior distribution of the precision parameter alpha. See Zhou, Hanson and Knapp (2014+) for more detailed hyperprior specifications.
mcmc	a list giving the MCMC parameters. The list must include the following elements: nburn an integer giving the number of burn-in scans, nskip an integer giving the thinning interval, nsave an integer giving the total number of scans to be saved, ndisplay an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out).
state	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
data	data frame.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes anovaDDP to print an error message and terminate if there are any incomplete observations.
work.dir	working directory.

Details

This function fits an ANOVA DDP model (De Iorio et al., 2009) for survival analysis of right censored time-to-event data.

Value

The results include the MCMC chains for the parameters discussed in De Iorio et al. (2009) and Zhou, Hanson and Knapp (2014+). Use names to find out what they are.

Author(s)

Haiming Zhou <<zhouh@email.sc.edu>> and Tim Hanson <<hansont@stat.sc.edu>>

References

De Iorio, M., Johnson, W. O., Mueller, P., and Rosner, G. L. (2009). Bayesian nonparametric nonproportional hazards survival modeling. Biometrics, 65(3), 762-771.

Zhou, H., Hanson, T. and Knapp, R. (2014+). A Spatial Copula Approach to Fully Bayesian Nonparametric Survival Analysis. In preparation for publication.

See Also

spCopulaDDP

Examples

```
## Not run:

#################################################################
# A simulated data: mixture of two normals
#################################################################
rm(list=ls())
library(MASS)
library(Rcpp)
library(RcppArmadillo)
library(coda)
library(survival)
library(spBayesSurv)

## True parameters
betaT = cbind(c(3.5, 0.5), c(2.5, -1));
wT = c(0.4, 0.6);
sig2T = c(1^2, 0.5^2);
theta1 = 0.98; theta2 = 100000;

## generate coordinates:
## npred is the # of locations for prediction
n = 300; npred = 30; ntot = n + npred;
ldist = 100; wdist = 40;
s1 = runif(ntot, 0, wdist); s2 = runif(ntot, 0, ldist);
s = rbind(s1,s2);
#plot(s[1,], s[2,]);

## Covariance matrix
corT = matrix(1, ntot, ntot);
for (i in 1:(ntot-1)){
  for (j in (i+1):ntot){
    dij = sqrt(sum( (s[,i]-s[,j])^2 ));
    corT[i,j] = theta1*exp(-theta2*dij);
    corT[j,i] = theta1*exp(-theta2*dij);
  }
}

## Generate x
x = runif(ntot,-1.5,1.5);
```

```
X = cbind(rep(1,ntot), x);
p = ncol(X); # number of covariates + 1
## Generate transformed log of survival times
z = mvrnorm(1, rep(0, ntot), corT);
## The pdf of Ti:
fi = function(y, xi, w=wT){
  nw = length(w);
  ny = length(y);
  res = matrix(0, ny, nw);
  Xi = c(1,xi);
  for (k in 1:nw){
    res[,k] = w[k]*dnorm(y, sum(Xi*betaT[,k]), sqrt(sig2T[k]) )
  }
  apply(res, 1, sum)
}
## true plot
xx = seq(-2, 7, 0.01)
plot(xx, fi(xx, -1), "l", lwd=2, col=2)
lines(xx, fi(xx, 1), "l", lwd=2, col=3)
## The CDF of Ti:
Fi = function(y, xi, w=wT){
  nw = length(w);
  ny = length(y);
  res = matrix(0, ny, nw);
  Xi = c(1,xi);
  for (k in 1:nw){
    res[,k] = w[k]*pnorm(y, sum(Xi*betaT[,k]), sqrt(sig2T[k]) )
  }
  apply(res, 1, sum)
}
## The inverse for CDF of Ti
Finvsingle = function(u, xi) {
  res = uniroot(function (x) Fi(x, xi)-u, lower=-500, upper=500);
  res$root
}
Finv = function(u, xi) {sapply(u, Finvsingle, xi)};
## Generate log of survival times y
u = pnorm(z);
y = rep(0, ntot);
for (i in 1:ntot){
  y[i] = Finv(u[i], x[i]);
}
#plot(x,y);
yTrue = y;

## Censoring scheme
Centime = runif(ntot, 3.5,5);
Centime = 10000;
delta = (y<=Centime) +0 ;
sum(delta)/ntot;
cen = which(delta==0);
y[cen] = Centime[cen];
```

```
## make a data frame
dtotal = data.frame(s1=s1, s2=s2, y=y, x=x, delta=delta, yTrue=yTrue);
## Hold out npred=30 for prediction purpose
predindex = sample(1:ntot, npred);
dpred = dtotal[predindex,];
dtrain = dtotal[-predindex,];

# rename the variables
d = dtrain;n=nrow(d); n;
s = cbind(d$s1, d$s2);
y = d$y;
x = d$x;
delta =d$delta;

# Prediction settings
xpred = dpred$x;
s0 = cbind( dpred$s1, dpred$s2 );
prediction = list(spred=s0, xpred=xpred);

###################################################################
# ANOVA DDP
###################################################################

# Prior information
prior = list(N = 10,
             m0 = rep(0,p),
             S0 = diag(rep(1e5,p), nrow=p, ncol=p),
             Sig0 = diag(rep(1e5,p), nrow=p, ncol=p),
             k0 = 7,
             nua = 2, nub = 1,
             a0 = 1, b0 = 1);

# current state values
state <- NULL

# MCMC parameters
nburn <- 1000
nsave <- 500
nskip <- 4
ndisplay <- 100
mcmc <- list(nburn=nburn,
             nsave=nsave,
             nskip=nskip,
             ndisplay=ndisplay)

# Fit the model
res1 = anovaDDP( y = y,
              delta =delta,
              x = x,
              prediction=prediction,
              prior=prior,
              mcmc=mcmc,
              state=state);
```

```
par(mfrow = c(2,2))
w.save = res1$w;
Kindex = which.max(rowMeans(w.save));
traceplot(mcmc(w.save[Kindex,]), main="w")
sig2.save = res1$sigma2;
traceplot(mcmc(sig2.save[Kindex,]), main="sig2")
beta.save = res1$beta;
traceplot(mcmc(beta.save[2,Kindex,]), main="beta")
alpha.save = res1$alpha;
traceplot(mcmc(alpha.save), main="alpha")

## LPML
#cpo=CPOanovaDDP(y, delta, X, beta.save, sig2.save, w.save )$cpo;
LPML1 = sum(log(res1$cpo)); LPML1;

## MSPE
mean((dpred$yTrue-apply(res1$Ypred, 1, median))^2);

## plots
par(mfrow = c(2,2))
xnew = c(-1, 1)
xpred = cbind(xnew);
nxpred = nrow(xpred);
ygrid = seq(0,6.0,0.05); tgrid = exp(ygrid);
ngrid = length(ygrid);
estimates = GetCurves(res1, xpred, ygrid, CI=c(0.05, 0.95));
fhat = estimates$fhat;
Shat = estimates$Shat;
## density in y
plot(ygrid, fi(ygrid, xnew[1]), "l", lwd=2, ylim=c(0, 0.8),
     xlim=c(0,6), main="density in y")
for(i in 1:nxpred){
  lines(ygrid, fi(ygrid, xnew[i]), lwd=2)
  lines(ygrid, fhat[,i], lty=2, lwd=2, col=4);
}
## survival in y
plot(ygrid, 1-Fi(ygrid, xnew[1]), "l", lwd=2, ylim=c(0, 1),
     xlim=c(0,6), main="survival in y")
for(i in 1:nxpred){
  lines(ygrid, 1-Fi(ygrid, xnew[i]), lwd=2)
  lines(ygrid, Shat[,i], lty=2, lwd=2, col=4);
}
## density in t
plot(tgrid, fi(ygrid, xnew[1])/tgrid, "l", lwd=2, ylim=c(0, 0.15),
     xlim=c(0,100), main="density in t")
for(i in 1:nxpred){
  lines(tgrid, fi(ygrid, xnew[i])/tgrid, lwd=2)
  lines(tgrid, fhat[,i]/tgrid, lty=2, lwd=2, col=4);
}
## survival in t
plot(tgrid, 1-Fi(ygrid, xnew[1]), "l", lwd=2, ylim=c(0, 1),
     xlim=c(0,100), main="survival in t")
```

```
for(i in 1:nxpred){
  lines(tgrid, 1-Fi(ygrid, xnew[i]), lwd=2)
  lines(tgrid, Shat[,i], lty=2, lwd=2, col=4);
}


## End(Not run)
```

GetCurves *Obtain estimated mariginal density, survival, and hazard functions given covariates*

Description

This function estimates mariginal density, survival, and hazard functions given covariates.

Usage

```
GetCurves(fit, xpred, ygrid, CI=c(0.05, 0.95))
```

Arguments

fit	an object obtained from functions including anovaDDP, spCopulaDDP, indeptCoxph and spCopulaCoxph.
xpred	an n by p vector of covariates matrix used for curve estimates.
ygrid	a vector of grid points (in log survival time scale) indicating where the curves will be estimated.
CI	a vector to indicate the level of credible interval, where first element indicate the lower probability.

Details

This function fits an ANOVA DDP model (De Iorio et al., 2009) for survival analysis of right censored time-to-event data.

Value

Use names to find out what they are, where fhat represents density, Shat represents survival and Hhat represents hazard.

Author(s)

Haiming Zhou <<zhouh@email.sc.edu>> and Tim Hanson <<hansont@stat.sc.edu>>

See Also

anovaDDP, spCopulaDDP, indeptCoxph, spCopulaCoxph

indeptCoxph *Bayesian Approach to Cox Proportional Hazard Model*

Description

This function fits a Cox proportional hazard model for right censored time-to-event data.

Usage

```
indeptCoxph(y, delta, x=NULL, prediction, prior, mcmc, state, RandomIntervals=F,
        data=sys.frame(sys.parent()), na.action=na.fail, work.dir=NULL)
```

Arguments

y	an n by 1 vector giving the survival times.
delta	an n by 1 vector indicating whether it is right censored (=0) or not (=1).
x	an n by p matrix of covariates without intercept. The default is NULL, indicating no covariates included.
prediction	a list giving the information used to obtain conditional inferences. The list includes the following elements: xpred giving the n by p covariates matrix, used for prediction.
prior	a list giving the prior information. The list includes the following parameter: M an integer giving the total number of cut points for baseline hazard. See Zhou, Hanson and Knapp (2014+) for more detailed hyperprior specifications.
mcmc	a list giving the MCMC parameters. The list must include the following elements: nburn an integer giving the number of burn-in scans, nskip an integer giving the thinning interval, nsave an integer giving the total number of scans to be saved, ndisplay an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out).
state	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
RandomIntervals	indicate if random cut ponts for baseline hazard is need. The default is TRUE.
data	data frame.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes indeptCoxph to print an error message and terminate if there are any incomplete observations.
work.dir	working directory.

Details

This generic function fits an independent Cox PH model (Zhou, Hanson and Knapp, 2014+), for right-censored data.

Value

The results include the MCMC chains for the parameters discussed in Zhou, Hanson and Knapp (2014+). Use `names` to find out what they are.

Author(s)

Haiming Zhou <<zhouh@email.sc.edu>> and Tim Hanson <<hansont@stat.sc.edu>>

References

Zhou, H., Hanson, T. and Knapp, R. (2014+). A Spatial Copula Approach to Fully Bayesian Nonparametric Survival Analysis. In preparation for publication.

See Also

spCopulaCoxph

Examples

```
## Not run:

#################################################################
# A simulated data: Cox PH
#################################################################
rm(list=ls())
library(MASS)
library(Rcpp)
library(RcppArmadillo)
library(coda)
library(survival)
library(spBayesSurv)
## True parameters
betaT = c(-1);
theta1 = 0.98; theta2 = 100000;

## generate coordinates:
## npred is the # of locations for prediction
n = 300; npred = 30; ntot = n + npred;
ldist = 100; wdist = 40;
s1 = runif(ntot, 0, wdist); s2 = runif(ntot, 0, ldist);
s = rbind(s1,s2); #plot(s[1,], s[2,]);

## Covariance matrix
corT = matrix(1, ntot, ntot);
for (i in 1:(ntot-1)){
  for (j in (i+1):ntot){
    dij = sqrt(sum( (s[,i]-s[,j])^2 ));
    corT[i,j] = theta1*exp(-theta2*dij);
    corT[j,i] = theta1*exp(-theta2*dij);
  }
}
```

```
## Generate x
x = runif(ntot,-1.5,1.5);
## Generate transformed log of survival times
z = mvrnorm(1, rep(0, ntot), corT);
## The CDF of Ti: Lambda(t) = t^2;
Fi = function(t, xi){
  res = 1-exp(-t^3*exp(sum(xi*betaT)));
  res[which(t<0)] = 0;
  res
}
## The pdf of Ti:
fi = function(t, xi){
  res=(1-Fi(t,xi))*3*t^2*exp(sum(xi*betaT));
  res[which(t<0)] = 0;
  res
}
#integrate(function(x) fi(x, 0), -Inf, Inf)
## true plot
xx = seq(0, 10, 0.1)
plot(xx, fi(xx, -1), "l", lwd=2, col=2)
lines(xx, fi(xx, 1), "l", lwd=2, col=3)

## The inverse for CDF of Ti
Finvsingle = function(u, xi) {
  res = uniroot(function (x) Fi(x, xi)-u, lower=0, upper=5000);
  res$root
}
Finv = function(u, xi) {sapply(u, Finvsingle, xi)};
## Generate survival times t
u = pnorm(z);
t = rep(0, ntot);
for (i in 1:ntot){
  t[i] = Finv(u[i], x[i]);
}
tTrue = t; #plot(x,t);

## Censoring scheme
Centime = runif(ntot, 1, 3);
#Centime = 10000;
delta = (t<=Centime) +0 ;
sum(delta)/ntot;
cen = which(delta==0);
t[cen] = Centime[cen];

## make a data frame
dtotal = data.frame(s1=s1, s2=s2, t=t, logt=log(t), x=x, delta=delta,
                    tTrue=tTrue, logtTrue=log(tTrue));
## Hold out npred=30 for prediction purpose
predindex = sample(1:ntot, npred);
dpred = dtotal[predindex,];
dtrain = dtotal[-predindex,];

# Prediction settings
```

```
xpred = dpred$x;
s0 = cbind( dpred$s1, dpred$s2 );
prediction = list(spred=s0, xpred=xpred);


#################################################################
# Independent Cox PH
#################################################################
# rename the variables
d = dtrain;n=nrow(d); n;
s = cbind(d$s1, d$s2);
t = d$t;
x = d$x;
X = cbind(x);
p = ncol(X); # number of covariates
delta =d$delta;

# Initial Exp Cox PH
fit0 = survreg( Surv(t, delta) ~ x, dist="exponential", data=d );
h0 = as.vector( exp( -fit0$coefficients[1] ) );
beta = as.vector( -fit0$coefficients[-1] );
S0 = as.matrix( fit0$var[-1,-1] );

# Prior information
prior = list(M = 10,
             h0 = h0,
             r0 = 1,
             V0 = 2*as.vector( exp( -2*fit0$coefficients[1] )*fit0$var[1,1] ),
             mu0 = as.vector( -fit0$coefficients[-1] ),
             Sig0 = diag(rep(1e5,p), nrow=p, ncol=p),
             adapter = (2.4)^2/p,
             hadapter = (2.4)^2,
             spadapter = (2.4)^2/2,
             theta0=c(1.0, 1.0, 0.5, 0.2));

# current state values
state <- list(theta = c(0.9, 1));

# MCMC parameters
nburn <- 1000
nsave <- 500
nskip <- 4
ndisplay <- 100
mcmc <- list(nburn=nburn,
             nsave=nsave,
             nskip=nskip,
             ndisplay=ndisplay)

# Fit the Cox PH model
res1 = indeptCoxph( y = t,
                    delta =delta,
                    x = x,
                    RandomIntervals=T,
```

```
                    prediction=prediction,
                    prior=prior,
                    mcmc=mcmc,
                    state=state);
par(mfrow = c(2,1))
h.save = res1$h; rowMeans(h.save);
traceplot(mcmc(h.save[2,]), main="h")
beta.save = res1$beta; rowMeans(beta.save);
traceplot(mcmc(beta.save[1,]), main="beta")
res1$ratebeta;
## LPML
LPML1 = sum(log(res1$cpo)); LPML1;
## MSPE
mean((dpred$tTrue-apply(res1$Tpred, 1, median))^2);

## plots
par(mfrow = c(2,1))
xnew = c(-1, 1)
xpred = cbind(xnew);
nxpred = nrow(xpred);
ygrid = seq(-5, 1.1, 0.05);tgrid = exp(ygrid);
ngrid = length(ygrid);
estimates = GetCurves(res1, xpred, ygrid, CI=c(0.05, 0.95));
fhat = estimates$fhat;
Shat = estimates$Shat;
## density in t
plot(tgrid, fi(tgrid, xnew[1]), "l", lwd=2,  xlim=c(0,6), main="density in y")
for(i in 1:nxpred){
  lines(tgrid, fi(tgrid, xnew[i]), lwd=2)
  lines(tgrid, fhat[,i], lty=2, lwd=2, col=4);
}
## survival in t
plot(tgrid, 1-Fi(tgrid, xnew[1]), "l", lwd=2, xlim=c(0,6), main="survival in y")
for(i in 1:nxpred){
  lines(tgrid, 1-Fi(tgrid, xnew[i]), lwd=2)
  lines(tgrid, Shat[,i], lty=2, lwd=2, col=4);
}


## End(Not run)
```

---

spCopulaCoxph *A Spatial Copula Approach to Cox Proportional Hazards Model*

Description

This function fits a spatial copula Cox PH model for survival analysis of point-referenced right censored time-to-event data.

Usage

```
spCopulaCoxph(y, delta, x=NULL, s, prediction, prior, mcmc, state, RandomIntervals=F,
      data=sys.frame(sys.parent()), na.action=na.fail, work.dir=NULL)
```

Arguments

y	an n by 1 vector giving the survival times.
delta	an n by 1 vector indicating whether it is right censored (=0) or not (=1).
x	an n by p matrix of covariates without intercept. The default is NULL, indicating no covariates included.
s	an n by d matrix of UMT coordinates, where d is the dimension of space.
prediction	a list giving the information used to obtain conditional inferences. The list includes the following elements: spred and xpred giving the n by 2 new locations and corresponding n by p covariates matrix, respectively, used for prediction .
prior	a list giving the prior information. The list includes the following parameter: M an integer giving the total number of cut points for baseline hazard. See Zhou, Hanson and Knapp (2014+) for more detailed hyperprior specifications.
mcmc	a list giving the MCMC parameters. The list must include the following elements: nburn an integer giving the number of burn-in scans, nskip an integer giving the thinning interval, nsave an integer giving the total number of scans to be saved, ndisplay an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out).
state	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
RandomIntervals	indicate if random cut ponts for baseline hazard is need. The default is TRUE.
data	data frame.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes spCopulaCoxph to print an error message and terminate if there are any incomplete observations.
work.dir	working directory.

Details

This generic function fits a spatial copula Cox PH model (Zhou, Hanson and Knapp, 2014+), for (potentially) point-referenced right-censored data.

Value

The results include the MCMC chains for the parameters discussed in Zhou, Hanson and Knapp (2014+). Use names to find out what they are.

Author(s)

Haiming Zhou <<zhouh@email.sc.edu>> and Tim Hanson <<hansont@stat.sc.edu>>

References

Zhou, H., Hanson, T. and Knapp, R. (2014+). A Spatial Copula Approach to Fully Bayesian Nonparametric Survival Analysis. In preparation for publication.

See Also

spCopulaDDP

Examples

```
## Not run:

#################################################################
# A simulated data: spatial Copula Cox PH
#################################################################
rm(list=ls())
library(MASS)
library(Rcpp)
library(RcppArmadillo)
library(coda)
library(survival)
library(spBayesSurv)
## True parameters
betaT = c(-1);
theta1 = 0.98; theta2 = 0.1;

## generate coordinates:
## npred is the # of locations for prediction
n = 300; npred = 30; ntot = n + npred;
ldist = 100; wdist = 40;
s1 = runif(ntot, 0, wdist); s2 = runif(ntot, 0, ldist);
s = rbind(s1,s2); #plot(s[1,], s[2,]);

## Covariance matrix
corT = matrix(1, ntot, ntot);
for (i in 1:(ntot-1)){
  for (j in (i+1):ntot){
    dij = sqrt(sum( (s[,i]-s[,j])^2 ));
    corT[i,j] = theta1*exp(-theta2*dij);
    corT[j,i] = theta1*exp(-theta2*dij);
  }
}

## Generate x
x = runif(ntot,-1.5,1.5);
## Generate transformed log of survival times
z = mvrnorm(1, rep(0, ntot), corT);
## The CDF of Ti: Lambda(t) = t^2;
Fi = function(t, xi){
  res = 1-exp(-t^3*exp(sum(xi*betaT)));
  res[which(t<0)] = 0;
  res
```

```
}
## The pdf of Ti:
fi = function(t, xi){
  res=(1-Fi(t,xi))*3*t^2*exp(sum(xi*betaT));
  res[which(t<0)] = 0;
  res
}
#integrate(function(x) fi(x, 0), -Inf, Inf)
## true plot
xx = seq(0, 10, 0.1)
plot(xx, fi(xx, -1), "l", lwd=2, col=2)
lines(xx, fi(xx, 1), "l", lwd=2, col=3)

## The inverse for CDF of Ti
Finvsingle = function(u, xi) {
  res = uniroot(function (x) Fi(x, xi)-u, lower=0, upper=5000);
  res$root
}
Finv = function(u, xi) {sapply(u, Finvsingle, xi)};
## Generate survival times t
u = pnorm(z);
t = rep(0, ntot);
for (i in 1:ntot){
  t[i] = Finv(u[i], x[i]);
}
tTrue = t; #plot(x,t);

## Censoring scheme
Centime = runif(ntot, 1, 5);
#Centime = 10000;
delta = (t<=Centime) +0 ;
sum(delta)/ntot;
cen = which(delta==0);
t[cen] = Centime[cen];

## make a data frame
dtotal = data.frame(s1=s1, s2=s2, t=t, logt=log(t), x=x, delta=delta,
                    tTrue=tTrue, logtTrue=log(tTrue));
## Hold out npred=30 for prediction purpose
predindex = sample(1:ntot, npred);
dpred = dtotal[predindex,];
dtrain = dtotal[-predindex,];

# Prediction settings
xpred = dpred$x;
s0 = cbind( dpred$s1, dpred$s2 );
prediction = list(spred=s0, xpred=xpred);

#################################################################
# spatial Copula Cox PH
#################################################################
# rename the variables
d = dtrain;n=nrow(d); n;
```

```
s = cbind(d$s1, d$s2);
t = d$t;
x = d$x;
X = cbind(x);
p = ncol(X); # number of covariates + 1
delta =d$delta;

# Initial Exp Cox PH
fit0 = survreg( Surv(t, delta) ~ x, dist="exponential", data=d );
h0 = as.vector( exp( -fit0$coefficients[1] ) );
beta = as.vector( -fit0$coefficients[-1] );
S0 = as.matrix( fit0$var[-1,-1] );

# Prior information
prior = list(M = 10,
             h0 = h0,
             r0 = 1,
             V0 = 2*as.vector( exp( -2*fit0$coefficients[1] )*fit0$var[1,1] ),
             mu0 = as.vector( -fit0$coefficients[-1] ),
             Sig0 = diag(rep(1e5,p), nrow=p, ncol=p),
             S0 = as.matrix( fit0$var[-1,-1] ),
             adapter = 2,
             hs0 = sqrt( as.vector( exp( -2*fit0$coefficients[1] )*fit0$var[1,1] ) ),
             hadapter = (2.4)^2,
             spS0 = diag(c(2,4)),
             spadapter = (2.4)^2/2,
             theta0=c(1.0, 1.0, 0.5, 0.2));

# current state values
state <- list(theta = c(0.9, 1));

# MCMC parameters
nburn <- 1000
nsave <- 500
nskip <- 4
ndisplay <- 100
mcmc <- list(nburn=nburn,
             nsave=nsave,
             nskip=nskip,
             ndisplay=ndisplay)

# Fit the spatial Cox PH model
res2 = spCopulaCoxph( y = t,
                      delta =delta,
                      s = s,
                      x = X,
                      RandomIntervals=T,
                      prediction=prediction,
                      prior=prior,
                      mcmc=mcmc,
                      state=state);
par(mfrow = c(2,1))
h.save = res2$h; rowMeans(h.save);
```

```
traceplot(mcmc(h.save[2,]), main="h")
beta.save = res2$beta; rowMeans(beta.save);
traceplot(mcmc(beta.save[1,]), main="beta")
res2$ratebeta;
res2$ratetheta;
res2$rateh;
## LPML
LPML2 = sum(log(res2$cpo)); LPML2;
## MSPE
mean((dpred$tTrue-apply(res2$Tpred, 1, median))^2);
## plots
par(mfrow = c(2,1))
xnew = c(-1, 1)
xpred = cbind(xnew);
nxpred = nrow(xpred);
ygrid = seq(-5, 1.1, 0.05);tgrid = exp(ygrid);
ngrid = length(ygrid);
estimates = GetCurves(res2, xpred, ygrid, CI=c(0.05, 0.95));
fhat = estimates$fhat;
Shat = estimates$Shat;
## density in t
plot(tgrid, fi(tgrid, xnew[1]), "l", lwd=2,  xlim=c(0,6), main="density in y")
for(i in 1:nxpred){
  lines(tgrid, fi(tgrid, xnew[i]), lwd=2)
  lines(tgrid, fhat[,i], lty=2, lwd=2, col=4);
}
## survival in t
plot(tgrid, 1-Fi(tgrid, xnew[1]), "l", lwd=2, xlim=c(0,6), main="survival in y")
for(i in 1:nxpred){
  lines(tgrid, 1-Fi(tgrid, xnew[i]), lwd=2)
  lines(tgrid, Shat[,i], lty=2, lwd=2, col=4);
}


## End(Not run)
```

`spCopulaDDP` *A Spatial Copula Approach to Fully Bayesian Nonparametric Survival Analysis*

Description

This function fits a spatial copula survival model for survival analysis of point-referenced right censored time-to-event data.

Usage

```
spCopulaDDP(y, delta, x=NULL, s, prediction, prior, mcmc, state, FSA = TRUE, knots,
         data=sys.frame(sys.parent()), na.action=na.fail, work.dir=NULL)
```

Arguments

y	an n by 1 vector giving the log survival times.
delta	an n by 1 vector indicating whether it is right censored (=0) or not (=1).
x	an n by p matrix of covariates without intercept. The default is NULL, indicating no covariates included.
s	an n by d matrix of UMT coordinates, where d is the dimension of space.
prediction	a list giving the information used to obtain conditional inferences. The list includes the following elements: spred and xpred giving the n by 2 new locations and corresponding n by p covariates matrix, respectively, used for prediction. In addition, predid needs to be specified if FSA=TRUE .
prior	a list giving the prior information. The list includes the following parameter: N an integer giving the truncation of the Dirichlet process, a0 and b0 giving the hyperparameters for prior distribution of the precision parameter alpha. See Zhou, Hanson and Knapp (2014+) for more detailed hyperprior specifications.
mcmc	a list giving the MCMC parameters. The list must include the following elements: nburn an integer giving the number of burn-in scans, nskip an integer giving the thinning interval, nsave an integer giving the total number of scans to be saved, ndisplay an integer giving the number of saved scans to be displayed on screen (the function reports on the screen when every ndisplay iterations have been carried out).
state	a list giving the current value of the parameters. This list is used if the current analysis is the continuation of a previous analysis.
FSA	indicate if the full scale approximation is need. The default is FALSE.
knots	a list giving the knots and block ids when FSA=TRUE. This list includes the following parameter: ss an m by d matrix of UMT coordinates, where d is the dimension of space, blockid an n by 1 id vector indicating which block that each observation is in.
data	data frame.
na.action	a function that indicates what should happen when the data contain NAs. The default action (na.fail) causes spCopulaDDP to print an error message and terminate if there are any incomplete observations.
work.dir	working directory.

Details

This generic function fits a spatial copula survival model (Zhou, Hanson and Knapp, 2014+), for (potentially) point-referenced right-censored data.

Value

The results include the MCMC chains for the parameters discussed in Zhou, Hanson and Knapp (2014+). Use names to find out what they are.

Author(s)

Haiming Zhou <<zhouh@email.sc.edu>> and Tim Hanson <<hansont@stat.sc.edu>>

References

Zhou, H., Hanson, T. and Knapp, R. (2014+). A Spatial Copula Approach to Fully Bayesian Nonparametric Survival Analysis. In preparation for publication.

See Also

anovaDDP

Examples

```
## Not run:

#################################################################
# A simulated data: mixture of two normals
#################################################################
rm(list=ls())
library(MASS)
library(Rcpp)
library(RcppArmadillo)
library(coda)
library(survival)
library(spBayesSurv)

## True parameters
betaT = cbind(c(3.5, 0.5), c(2.5, -1));
wT = c(0.4, 0.6);
sig2T = c(1^2, 0.5^2);
theta1 = 0.98; theta2 = 0.1;

## generate coordinates:
## npred is the # of locations for prediction
n = 300; npred = 30; ntot = n + npred;
ldist = 100; wdist = 40;
s1 = runif(ntot, 0, wdist); s2 = runif(ntot, 0, ldist);
s = rbind(s1,s2);
#plot(s[1,], s[2,]);
## divide them into blocks
nldist=10; nwdist=4;
nb=nldist*nwdist; nb; # number of blocks;
coor = matrix(0, nb, 4); ## four edges for each block;
tempindex=1; lstep=ldist/nldist; wstep=wdist/nwdist;
for(i in 1:nwdist){
  for(j in 1:nldist){
    coor[tempindex,] = c((i-1)*wstep, i*wstep, (j-1)*lstep, j*lstep );
    tempindex = tempindex + 1;
  }
}
## Assign block id for each location
blockid = rep(NA,ntot);
for(i in 1:nb){
  blockid[((s1>coor[i,1])*(s1<=coor[i,2])*(s2>coor[i,3])*(s2<=coor[i,4]))==1]=i;
}
```

```
## Choose knots S*
nldist=15; nwdist=6;
m=nldist*nwdist; m; # number of knots;
ss = matrix(0, m, 2);
tempindex=1; lstep=ldist/nldist; wstep=wdist/nwdist;
for(i in 1:nwdist){
  for(j in 1:nldist){
    ss[tempindex,] = c( (i-1)*wstep+wstep/2, (j-1)*lstep+lstep/2);
    tempindex = tempindex + 1;
  }
}
## Covariance matrix
dnn = .Call("DistMat", s, s, PACKAGE = "spBayesSurv");
corT = theta1*exp(-theta2*dnn)+(1-theta1)*diag(ntot);

## Generate x
x = runif(ntot,-1.5,1.5);
X = cbind(rep(1,ntot), x);
p = ncol(X); # number of covariates + 1
## Generate transformed log of survival times
z = mvrnorm(1, rep(0, ntot), corT);
## The pdf of Ti:
fi = function(y, xi, w=wT){
  nw = length(w);
  ny = length(y);
  res = matrix(0, ny, nw);
  Xi = c(1,xi);
  for (k in 1:nw){
    res[,k] = w[k]*dnorm(y, sum(Xi*betaT[,k]), sqrt(sig2T[k]) )
  }
  apply(res, 1, sum)
}
## true plot
xx = seq(-2, 7, 0.01)
plot(xx, fi(xx, -1), "l", lwd=2, col=2)
lines(xx, fi(xx, 1), "l", lwd=2, col=3)
## The CDF of Ti:
Fi = function(y, xi, w=wT){
  nw = length(w);
  ny = length(y);
  res = matrix(0, ny, nw);
  Xi = c(1,xi);
  for (k in 1:nw){
    res[,k] = w[k]*pnorm(y, sum(Xi*betaT[,k]), sqrt(sig2T[k]) )
  }
  apply(res, 1, sum)
}
## The inverse for CDF of Ti
Finvsingle = function(u, xi) {
  res = uniroot(function (x) Fi(x, xi)-u, lower=-500, upper=500);
  res$root
}
Finv = function(u, xi) {sapply(u, Finvsingle, xi)};
```

```
## Generate log of survival times y
u = pnorm(z);
y = rep(0, ntot);
for (i in 1:ntot){
  y[i] = Finv(u[i], x[i]);
}
#plot(x,y);
yTrue = y;

## Censoring scheme
Centime = runif(ntot, 3.5,5);
Centime = 10000;
delta = (y<=Centime) +0 ;
sum(delta)/ntot;
cen = which(delta==0);
y[cen] = Centime[cen];

## make a data frame
dtotal = data.frame(s1=s1, s2=s2, y=y, x=x, delta=delta, yTrue=yTrue, id=blockid);
## Hold out npred=30 for prediction purpose
predindex = sample(1:ntot, npred);
dpred = dtotal[predindex,];
dtrain = dtotal[-predindex,];

# rename the variables
d = dtrain; n=nrow(d); n;
s = cbind(d$s1, d$s2);
y = d$y;
x = d$x;
delta =d$delta;

# FSA settings
knots = list(ss=ss, blockid=d$id);

# Prediction settings
xpred = dpred$x;
s0 = cbind( dpred$s1, dpred$s2 );
prediction = list(spred=s0, xpred=xpred, predid=dpred$id);

#################################################################
# spatial copula DDP
#################################################################
# MCMC parameters
nburn <- 1000
nsave <- 500
nskip <- 4
ndisplay <- 100
mcmc <- list(nburn=nburn,
             nsave=nsave,
             nskip=nskip,
             ndisplay=ndisplay)

# Prior information
```

```
prior = list(N = 10,
             m0 = rep(0,p),
             S0 = diag(rep(1e5,p), nrow=p, ncol=p),
             Sig0 = diag(rep(1e5,p), nrow=p, ncol=p),
             k0 = 7,
             nua = 2, nub = 1,
             a0 = 1, b0 = 1,
             spS0 = diag(c(4,4)),
             spadapter = 1,
             theta0 = c(1.0, 1.0, 0.5, 0.2));

# current state values
state <- list(theta=c(0.95, 0.5));

# Fit the model
fitspDDP = spCopulaDDP( y = y,
              delta =delta,
              x = x,
              s = s,
              prediction=prediction,
              prior=prior,
              mcmc=mcmc,
              state=state,
              FSA=FALSE,
              knots=knots);
res = fitspDDP; remove(fitspDDP);
# trace plots
par(mfrow = c(3,2))
w.save2 = res$w;
Kindex = which.max(rowMeans(w.save2));
traceplot(mcmc(w.save2[Kindex,]), main="w")
sig2.save2 = res$sigma2;
traceplot(mcmc(sig2.save2[Kindex,]), main="sig2")
beta.save2 = res$beta;
alpha.save2 = res$alpha;
traceplot(mcmc(beta.save2[2,Kindex,]), main="beta")
traceplot(mcmc(alpha.save2), main="alpha")
theta1.save2 = res$theta1;
theta2.save2 = res$theta2
traceplot(mcmc(theta1.save2), main="theta1")
traceplot(mcmc(theta2.save2), main="theta2")

## LPML
LPML2 = sum(log(res$cpo)); LPML2;
## MSPE
mean((dpred$yTrue-apply(res$Ypred, 1, median))^2);

## plots
par(mfrow = c(2,2));
xnew = c(-1, 1);
xpred = cbind(xnew);
nxpred = nrow(xpred);
ygrid = seq(0,6.0,0.05); tgrid = exp(ygrid);
```

```
ngrid = length(ygrid);
estimates = GetCurves(res, xpred, ygrid, CI=c(0.05, 0.95));
fhat = estimates$fhat;
Shat = estimates$Shat;
## density in y
plot(ygrid, fi(ygrid, xnew[1]), "l", lwd=2, ylim=c(0, 0.8),
      xlim=c(0,6), main="density in y")
for(i in 1:nxpred){
  lines(ygrid, fi(ygrid, xnew[i]), lwd=2)
  lines(ygrid, fhat[,i], lty=2, lwd=2, col=4);
}
## survival in y
plot(ygrid, 1-Fi(ygrid, xnew[1]), "l", lwd=2, ylim=c(0, 1),
      xlim=c(0,6), main="survival in y")
for(i in 1:nxpred){
  lines(ygrid, 1-Fi(ygrid, xnew[i]), lwd=2)
  lines(ygrid, Shat[,i], lty=2, lwd=2, col=4);
}
## density in t
plot(tgrid, fi(ygrid, xnew[1])/tgrid, "l", lwd=2, ylim=c(0, 0.15),
      xlim=c(0,100), main="density in t")
for(i in 1:nxpred){
  lines(tgrid, fi(ygrid, xnew[i])/tgrid, lwd=2)
  lines(tgrid, fhat[,i]/tgrid, lty=2, lwd=2, col=4);
}
## survival in t
plot(tgrid, 1-Fi(ygrid, xnew[1]), "l", lwd=2, ylim=c(0, 1),
      xlim=c(0,100), main="survival in t")
for(i in 1:nxpred){
  lines(tgrid, 1-Fi(ygrid, xnew[i]), lwd=2)
  lines(tgrid, Shat[,i], lty=2, lwd=2, col=4);
}


## End(Not run)
```

Index