

# Package ‘spatsurv’

May 9, 2026

**Type** Package

**Title** Bayesian Spatial Survival Analysis with Parametric Proportional Hazards Models

**Version** 2.0-1

**Date** 2023-10-18

**Author** Benjamin M. Taylor and Barry S. Rowlingson  
Additional contributions  
Ziyu Zheng

**Maintainer** Benjamin M. Taylor <benjamin.taylor.software@gmail.com>

**Description** Bayesian inference for parametric proportional hazards spatial survival models; flexible spatial survival models. See Benjamin M. Taylor, Barry S. Rowlingson (2017) <[doi:10.18637/jss.v077.i04](https://doi.org/10.18637/jss.v077.i04)>.

**License** GPL-3

**Imports** survival, sp, spatstat.explore, spatstat.geom,  
spatstat.random, raster, iterators, fields, Matrix, stringr,  
sf, RColorBrewer, methods, lubridate

**Suggests** rgl

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Depends** R (>= 2.10)

**Repository** CRAN

**Date/Publication** 2023-10-19 08:20:02 UTC

## Contents

spatsurv-package . . . . .	5
.onAttach . . . . .	10
allocate . . . . .	11
alpha . . . . .	12
B . . . . .	12

basehazard . . . . .	13
basehazard.basehazardspec . . . . .	13
baseHazST . . . . .	14
baselinehazard . . . . .	15
baselinehazard_multiWay . . . . .	16
betapriorGauss . . . . .	17
blockDiag . . . . .	17
boxplotRisk . . . . .	18
Bspline.construct . . . . .	18
BsplineHaz . . . . .	19
checkSurvivalData . . . . .	20
circulant . . . . .	21
circulant.matrix . . . . .	21
circulant.numeric . . . . .	22
circulantij . . . . .	22
covmodel . . . . .	23
CSplot . . . . .	23
cumbasehazard . . . . .	24
cumbasehazard.basehazardspec . . . . .	24
cumulativeBspline.construct . . . . .	25
densityquantile . . . . .	25
densityquantile.basehazardspec . . . . .	26
densityquantile_PP . . . . .	26
density_PP . . . . .	27
derivindepGaussianprior . . . . .	27
derivindepGaussianpriorST . . . . .	28
derivpsplineprior . . . . .	28
distinfo . . . . .	29
distinfo.basehazardspec . . . . .	30
estimateY . . . . .	30
etapriorGauss . . . . .	31
Et_PP . . . . .	31
EvalCov . . . . .	32
ExponentialCovFct . . . . .	32
exponentialHaz . . . . .	33
FFTgrid . . . . .	34
fixedpars . . . . .	35
fixmatrix . . . . .	35
fixParHaz . . . . .	36
frailtylag1 . . . . .	36
fs . . . . .	37
fstimes . . . . .	37
gamma2risk . . . . .	38
GammafromY . . . . .	38
GammaFromY_SPDE . . . . .	39
gencens . . . . .	40
getbb . . . . .	40
getBbasis . . . . .	41

getcov . . . . .	41
getgrd . . . . .	42
getGrid . . . . .	42
getleneta . . . . .	43
getOptCellwidth . . . . .	43
getparranges . . . . .	44
getsurvdata . . . . .	45
gompertzHaz . . . . .	45
gradbasehazard . . . . .	46
gradbasehazard.basehazardspec . . . . .	47
gradcumbasehazard . . . . .	48
gradcumbasehazard.basehazardspec . . . . .	48
grid2spdf . . . . .	49
grid2spix . . . . .	49
grid2spts . . . . .	50
gridY . . . . .	50
gridY_polygonal . . . . .	51
guess_t . . . . .	51
hasNext . . . . .	52
hasNext.iter . . . . .	52
hazardexceedance . . . . .	52
hazardpars . . . . .	53
hazard_PP . . . . .	54
hessbasehazard . . . . .	54
hessbasehazard.basehazardspec . . . . .	55
hesscumbasehazard . . . . .	55
hesscumbasehazard.basehazardspec . . . . .	56
imputationModel . . . . .	56
Independent . . . . .	57
indepGaussianprior . . . . .	57
indepGaussianpriorST . . . . .	58
inference.control . . . . .	59
insert . . . . .	60
invtransformweibull . . . . .	61
is.burnin . . . . .	61
is.retain . . . . .	62
iteration . . . . .	62
logPosterior . . . . .	63
logPosterior_gridded . . . . .	64
logPosterior_polygonal . . . . .	65
logPosterior_SPDE . . . . .	66
loop.mcmc . . . . .	67
makehamHaz . . . . .	68
maxlikparamPHsurv . . . . .	69
MCE . . . . .	70
mcmcLoop . . . . .	70
mcmcpars . . . . .	71
mcmcPriors . . . . .	71

mcmcProgressNone . . . . .	72
mcmcProgressPrint . . . . .	73
mcmcProgressTextBar . . . . .	73
midpts . . . . .	74
multiWayHaz . . . . .	74
neighLocs . . . . .	75
neighOrder . . . . .	75
nextStep . . . . .	76
NonSpatialLogLikelihood_or_gradient . . . . .	76
omegapriorGauss . . . . .	77
omegapriorGaussST . . . . .	78
optifix . . . . .	78
plot.FFTgrid . . . . .	79
plotsurv . . . . .	80
polyadd . . . . .	81
polymult . . . . .	82
posteriorcov . . . . .	82
predict.mcmcspsurv . . . . .	83
print.mcmc . . . . .	84
print.mcmcspsurv . . . . .	85
print.mlspatsurv . . . . .	85
print.textSummary . . . . .	86
priorposterior . . . . .	87
proposalVariance . . . . .	88
proposalVariance_gridded . . . . .	89
proposalVariance_polygonal . . . . .	90
proposalVariance_SPDE . . . . .	91
PsplineHaz . . . . .	92
psplineprior . . . . .	93
psplineRWprior . . . . .	94
QuadApprox . . . . .	95
quantile.mcmcspsurv . . . . .	95
quantile.mlspatsurv . . . . .	96
randompars . . . . .	97
reconstruct.bs . . . . .	97
reconstruct.bs.coxph . . . . .	98
reconstruct.bs.mcmcspsurv . . . . .	99
resetLoop . . . . .	100
residuals.mcmcspsurv . . . . .	100
rootWeibullHaz . . . . .	101
setTxtProgressBar2 . . . . .	102
setupHazard . . . . .	102
setupPrecMatStruct . . . . .	103
showGrid . . . . .	104
simsurv . . . . .	104
spatialpars . . . . .	105
spatsurvVignette . . . . .	106
SPDE . . . . .	106

SPDEprec . . . . .	107
SpikedExponentialCovFct . . . . .	107
Summarise . . . . .	108
summary.mcmc . . . . .	109
summary.mcmcsratsurv . . . . .	109
surv3d . . . . .	110
survival_PP . . . . .	111
survspat . . . . .	112
survspatNS . . . . .	113
textSummary . . . . .	114
timevaryingPL . . . . .	115
tpowHaz . . . . .	116
transformweibull . . . . .	117
TwoWayHazAdditive . . . . .	118
txtProgressBar2 . . . . .	118
vcov.mcmcsratsurv . . . . .	119
vcov.mlspatsurv . . . . .	120
weibullHaz . . . . .	120
YfromGamma . . . . .	122
YFromGamma_SPDE . . . . .	122

<b>Index</b>	<b>124</b>
--------------	------------

---

spatsurv-package	<i>spatsurv</i>
------------------	-----------------

---

## Description

An R package for spatially correlated parametric proportional hazards survival analysis.

## Usage

spatsurv

## Format

An object of class `logical` of length 1.

## Details

Package:	spatsurv
Type:	Package
Title:	Bayesian Spatial Survival Analysis with Parametric Proportional Hazards Models
Version:	2.0-1
Date:	2023-10-18
Author:	Benjamin M. Taylor and Barry S. Rowlingson Additional contributions Ziyu Zheng
Maintainer:	Benjamin M. Taylor <benjamin.taylor.software@gmail.com>
Description:	Bayesian inference for parametric proportional hazards spatial survival models; flexible spatial survival

```

License:          GPL-3
Imports:          survival, sp, spatstat.explore, spatstat.geom, spatstat.random, raster, iterators, fields, Matrix, stringr, sf, R6
Suggests:         rgl
Encoding:         UTF-8
RoxygenNote:      7.2.3
NeedsCompilation: no
Packaged:         2022-11-22 14:11:17 UTC; taylorb7
Depends:          R (>= 2.10)
Repository:       CRAN
Date/Publication: 2022-11-22 14:30:02 UTC

```

### Index of help topics:

```

.onAttach          .onAttach function
B                  B function
Bspline.construct  Bspline.construct function
BsplineHaz         BsplineHaz function
CSplot            CSplot function
Et_PP             Et_PP function
EvalCov           EvalCov function
ExponentialCovFct ExponentialCovFct function
FFTgrid           FFTgrid function
GammaFromY_SPDE   GammaFromY_SPDE function
GammafromY        GammafromY function
Independent        Independent function
MCE               MCE function
NonSpatialLogLikelihood_or_gradient
                  NonSpatialLogLikelihood_or_gradient function
PsplineHaz        PsplineHaz function
QuadApprox        QuadApprox function
SPDE              SPDE function
SPDEprec          SPDEprec function
SpikedExponentialCovFct
                  SpikedExponentialCovFct function
Summarise         Summarise function
TwoWayHazAdditive TwoWayHazAdditive function
YFromGamma_SPDE   YFromGamma_SPDE function
YfromGamma        YfromGamma function
allocate          allocate function
alpha             alpha function
baseHazST         baseHazST function
basehazard        basehazard function
basehazard.basehazardspec
                  basehazard.basehazardspec function
baselinehazard    baselinehazard function
baselinehazard_multiWay
                  baselinehazard_multiWay function

```

betapriorGauss	betapriorGauss function
blockDiag	A function to
boxplotRisk	boxplotRisk function
checkSurvivalData	checkSurvivalData function
circulant	circulant function
circulant.matrix	circulant.matrix function
circulant.numeric	circulant.numeric function
circulantij	circulantij function
covmodel	covmodel function
cumbasehazard	cumbasehazard function
cumbasehazard.basehazardspec	cumbasehazard.basehazardspec function
cumulativeBspline.construct	cumulativeBspline.construct function
density_PP	density_PP function
densityquantile	densityquantile function
densityquantile.basehazardspec	densityquantile.basehazardspec function
densityquantile_PP	densityquantile_PP function
derivindepGaussianprior	derivindepGaussianprior function
derivindepGaussianpriorST	derivindepGaussianpriorST function
derivpsplineprior	derivpsplineprior function
distinfo	distinfo function
distinfo.basehazardspec	distinfo.basehazardspec function
estimateY	estimateY function
etapriorGauss	etapriorGauss function
exponentialHaz	exponentialHaz function
fixParHaz	fixParHaz function
fixedpars	fixedpars function
fixmatrix	fixmatrix function
frailtylag1	frailtylag1 function
fs	London Fire Brigade property
fstimes	London Fire Brigade response times to dwelling fires, 2009
gamma2risk	gamma2risk function
gencens	gencens function
getBbasis	getBbasis function
getGrid	getGrid function
getOptCellwidth	getOptCellwidth function
getbb	getbb function
getcov	getcov function
getgrd	getgrd function
getleneta	getleneta function
getparranges	getparranges function
getsurvdata	getsurvdata function

gompertzHaz	gompertzHaz function
gradbasehazard	gradbasehazard function
gradbasehazard.basehazardspec	gradbasehazard.basehazardspec function
gradcumbasehazard	gradcumbasehazard function
gradcumbasehazard.basehazardspec	gradcumbasehazard.basehazardspec function
grid2spdf	grid2spdf function
grid2spix	grid2spix function
grid2spts	grid2spts function
gridY	gridY function
gridY_polygonal	gridY_polygonal function
guess_t	guess_t function
hasNext	generic hasNext method
hasNext.iter	hasNext.iter function
hazard_PP	hazard_PP function
hazardexceedance	hazardexceedance function
hazardpars	hazardpars function
hessbasehazard	hessbasehazard function
hessbasehazard.basehazardspec	hessbasehazard.basehazardspec function
hesscumbasehazard	hesscumbasehazard function
hesscumbasehazard.basehazardspec	hesscumbasehazard.basehazardspec function
imputationModel	imputationModel function
indepGaussianprior	indepGaussianprior function
indepGaussianpriorST	indepGaussianpriorST function
inference.control	inference.control function
insert	insert function
invtransformweibull	invtransformweibull function
is.burnin	is this a burn-in iteration?
is.retain	do we retain this iteration?
iteration	iteration number
logPosterior	logPosterior function
logPosterior_SPDE	logPosterior_SPDE function
logPosterior_gridded	logPosterior_gridded function
logPosterior_polygonal	logPosterior_polygonal function
loop.mcmc	loop over an iterator
makehamHaz	makehamHaz function
maxlikparamPHsurv	maxlikparamPHsurv function
mcmcLoop	iterator for MCMC loops
mcmcPriors	mcmcPriors function
mcmcProgressNone	null progress monitor
mcmcProgressPrint	printing progress monitor
mcmcProgressTextBar	text bar progress monitor
mcmcpars	mcmcpars function
midpts	midpts function

multiWayHaz	multiWayHaz function
neighLocs	neighLocs function
neighOrder	neighOrder function
nextStep	next step of an MCMC chain
omegapriorGauss	omegapriorGauss function
omegapriorGaussST	omegapriorGaussST function
optifix	optifix function
plot.FFTgrid	plot.FFTgrid function
plotsurv	plotsurv function
polyadd	polyadd function
polymult	polymult function
posteriorcov	posteriorcov function
predict.mcmcspatsurv	predict.mcmcspatsurv function
print.mcmc	print.mcmc function
print.mcmcspatsurv	print.mcmcspatsurv function
print.mlspatsurv	print.mlspatsurv function
print.textSummary	print.textSummary function
priorposterior	priorposterior function
proposalVariance	proposalVariance function
proposalVariance_SPDE	proposalVariance_SPDE function
proposalVariance_gridded	proposalVariance_gridded function
proposalVariance_polygonal	proposalVariance_polygonal function
psplineRWprior	psplineRWprior function
psplineprior	psplineprior function
quantile.mcmcspatsurv	quantile.mcmcspatsurv function
quantile.mlspatsurv	quantile.mlspatsurv function
randompars	randompars function
reconstruct.bs	reconstruct.bs function
reconstruct.bs.coxph	reconstruct.bs.coxph function
reconstruct.bs.mcmcspatsurv	reconstruct.bs.mcmcspatsurv function
resetLoop	reset iterator
residuals.mcmcspatsurv	residuals.mcmcspatsurv function
rootWeibullHaz	rootWeibullHaz function
setTxtProgressBar	set the progress bar
setupHazard	setupHazard function
setupPrecMatStruct	setupPrecMatStruct function
showGrid	showGrid function
simsurv	simsurv function
spatialpars	spatialpars function
spatsurv-package	spatsurv
spatsurvVignette	spatsurvVignette function
summary.mcmc	summary.mcmc function
summary.mcmcspatsurv	summary.mcmcspatsurv function
surv3d	Spatial Survival Plot in 3D

survival_PP	survival_PP function
survspat	survspat function
survspatNS	survspatNS function
textSummary	textSummary function
timevaryingPL	timevaryingPL function
tpowHaz	tpowHaz function
transformweibull	transformweibull function
txtProgressBar2	A text progress bar with label
vcov.mcmcspatsurv	vcov.mcmcspatsurv function
vcov.mlspatsurv	vcov.mlspatsurv function
weibullHaz	weibullHaz function

#### Dependencies

The package `spatsurv` depends upon some other important contributions to CRAN in order to operate; their uses here are indicated:

`survival`, `sp`, `spatstat`, `raster`, `iterators`, `RandomFields`, `fields`, `rgl`, `Matrix`, `stringr`, `RColorBrewer`, `geostatsp`.

#### Citation

To cite use of `spatsurv`, the user may refer to the following work:

Benjamin M. Taylor and Barry S. Rowlingson (2017).  
`spatsurv`: An R Package for Bayesian Inference with Spatial Survival Models.  
 Journal of Statistical Software, 77(4), 1-32, doi:10.18637/jss.v077.i04.

references

X

#### Author(s)

Benjamin Taylor, Health and Medicine, Lancaster University, Barry Rowlingson, Health and Medicine, Lancaster University

---

`.onAttach`

*.onAttach function*

---

#### Description

A function to print a welcome message on loading package

#### Usage

```
.onAttach(libname, pkgname)
```

#### Arguments

<code>libname</code>	libname argument
<code>pkgname</code>	pkgname argument

**Value**

...

---

allocate	<i>allocate function</i>
----------	--------------------------

---

**Description**

A function to allocate coordinates to an observation whose spatial location is known to the regional level

**Usage**

```
allocate(poly, popden, survdat, pid, sid, n = 2, wid = 2000)
```

**Arguments**

poly	a SpatialPolygonsDataFrame, on which the survival data exist in aggregate form
popden	a sub-polygon raster image of population density
survdat	data.frame containing the survival data
pid	name of the variable in the survival data that gives the region identifier in poly
sid	the name of the variable in poly to match the region identifier in survdat to
n	the number of different allocations to make. e.g. if n is 2 (the default) two candidate sets of locations are available.
wid	The default is 2000, interpreted in metres ie 2Km. size of buffer to add to window for raster cropping purposes: this ensures that for each polygon, the cropped raster covers it completely.

**Value**

matrices x and y, both of size (number of observations in survdat x n) giving n potential candidate locations of points in the columns of x and y.

---

alpha	<i>alpha function</i>
-------	-----------------------

---

**Description**

A function used in calculating the coefficients of a B-spline curve

**Usage**

alpha(i, j, knots, knotidx)

**Arguments**

i	index i
j	index j
knots	knot vector
knotidx	knot index

**Value**

a vector

---

B	<i>B function</i>
---	-------------------

---

**Description**

A recursive function used in calculating the coefficients of a B-spline curve

**Usage**

B(x, i, j, knots)

**Arguments**

x	locations at which to evaluate the B-spline
i	index i
j	index j
knots	a knot vector

**Value**

a vector of polynomial coefficients

---

basehazard	<i>basehazard function</i>
------------	----------------------------

---

**Description**

Generic function for computing the baseline hazard

**Usage**

```
basehazard(obj, ...)
```

**Arguments**

obj	an object
...	additional arguments – currently there are none, but this is for extensibility

**Value**

method basehazard

**See Also**

[basehazard.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

basehazard.basehazardspec	<i>basehazard.basehazardspec function</i>
---------------------------	---

---

**Description**

A function to retrieve the baseline hazard function

**Usage**

```
## S3 method for class 'basehazardspec'  
basehazard(obj, ...)
```

**Arguments**

obj	an object of class basehazardspec
...	additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the baseline hazard

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

baseHazST

*baseHazST function*

---

**Description**

A function to

**Usage**

```
baseHazST(  
  bh1 = NULL,  
  survobj,  
  t0,  
  nbreaks = 5,  
  breakmethod = "quantile",  
  MLinits = NULL  
)
```

**Arguments**

bh1	X
survobj	X
t0	X
nbreaks	X
breakmethod	X
MLinits	X

**Value**

...

---

baselinehazard	<i>baselinehazard function</i>
----------------	--------------------------------

---

**Description**

A function to compute quantiles of the posterior baseline hazard or cumulative baseline hazard.

**Usage**

```
baselinehazard(
  x,
  t = NULL,
  n = 100,
  probs = c(0.025, 0.5, 0.975),
  cumulative = FALSE,
  plot = TRUE,
  bw = FALSE,
  ...
)
```

**Arguments**

<code>x</code>	an object inheriting class <code>mcmcspatsurv</code>
<code>t</code>	optional vector of times at which to compute the quantiles, Default is <code>NULL</code> , in which case a uniformly spaced vector of length <code>n</code> from 0 to the maximum time is used
<code>n</code>	the number of points at which to compute the quantiles if <code>t</code> is <code>NULL</code>
<code>probs</code>	vector of probabilities
<code>cumulative</code>	logical, whether to return the baseline hazard (default i.e. <code>FALSE</code> ) or cumulative baseline hazard
<code>plot</code>	whether to plot the result
<code>bw</code>	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
<code>...</code>	additional arguments to be passed to <code>plot</code>

**Value**

the vector of times and quantiles of the baseline or cumulative baseline hazard at those times

**See Also**

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [predict.mcmcspatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

baselinehazard\_multiWay

*baselinehazard\_multiWay function*

---

### Description

A function to

### Usage

```
baselinehazard_multiWay(  
  x,  
  probs = c(0.025, 0.5, 0.975),  
  cumulative = FALSE,  
  plot = TRUE,  
  joint = FALSE,  
  xlims = NULL,  
  ylims = NULL,  
  ...  
)
```

### Arguments

x	X
probs	X
cumulative	X
plot	X
joint	X
xlims	X
ylims	X
...	X

### Value

...

---

betapriorGauss	<i>betapriorGauss function</i>
----------------	--------------------------------

---

**Description**

A function to define Gaussian priors for beta. This function simply stores a vector of means and standard deviations to be passed to the main MCMC function, survspat.

**Usage**

```
betapriorGauss(mean, sd)
```

**Arguments**

mean	the prior mean, a vector of length 1 or more. 1 implies a common mean.
sd	the prior standard deviation, a vector of length 1 or more. 1 implies a common standard deviation.

**Value**

an object of class "betapriorGauss"

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

blockDiag	<i>A function to</i>
-----------	----------------------

---

**Description**

A function to

**Usage**

```
blockDiag(matlist)
```

**Arguments**

matlist	X
---------	---

**Value**

...

---

boxplotRisk	<i>boxplotRisk function</i>
-------------	-----------------------------

---

**Description**

A function to

**Usage**

```
boxplotRisk(g2r)
```

**Arguments**

g2r	X
-----	---

**Value**

...

---

Bspline.construct	<i>Bspline.construct function</i>
-------------------	-----------------------------------

---

**Description**

A function to construct a B-spline basis matrix for given data and basis coefficients. Used in evaluating the baseline hazard.

**Usage**

```
Bspline.construct(x, basis)
```

**Arguments**

x	a vector, the data
basis	an object created by the getBbasis function

**Value**

a basis matrix

BsplineHaz

*BsplineHaz function***Description**

A function to define a parametric proportional hazards model where the baseline hazard is modelled by a basis spline. This function returns an object inheriting class 'basehazardspec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'gradcumbasehazard', 'hesscumbasehazard' and 'densityquantile'

**Usage**

```
BsplineHaz(times, knots = quantile(times), degree = 3, MLinits = NULL)
```

**Arguments**

times	vector of survival times (both censored and uncensored)
knots	vector of knots in ascending order, must include minimum and maximum values of 'times'
degree	degree of the spline basis, default is 3
MLinits	optional starting values for the non-spatial maximisation routine using optim. Note that we are working with the log of the parameters. Default is -10 for each parameter.

**Details**

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is `'densityquantile'`. In the case of the Weibull model for the baseline hazard, it can be shown that the `q`-th quantile is:

### Value

an object inheriting class `'basehazardspec'`

### See Also

[exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#)

---

checkSurvivalData      *checkSurvivalData function*

---

### Description

A function to check whether the survival data to be passed to `survspat` is in the correct format

### Usage

```
checkSurvivalData(s)
```

### Arguments

`s`                      an object of class `Surv`, from the `survival` package

### Value

if there are any issues with data format, these are returned with the data an error message explaining any issues with the data

---

circulant	<i>circulant function</i>
-----------	---------------------------

---

**Description**

generic function for constructing circulant matrices

**Usage**

```
circulant(x, ...)
```

**Arguments**

x	an object
...	additional arguments

**Value**

method circulant

---

circulant.matrix	<i>circulant.matrix function</i>
------------------	----------------------------------

---

**Description**

If x is a matrix whose columns are the bases of the sub-blocks of a block circulant matrix, then this function returns the block circulant matrix of interest.

**Usage**

```
## S3 method for class 'matrix'
circulant(x, ...)
```

**Arguments**

x	a matrix object
...	additional arguments

**Value**

If x is a matrix whose columns are the bases of the sub-blocks of a block circulant matrix, then this function returns the block circulant matrix of interest.

---

circulant.numeric      *circulant.numeric function*

---

**Description**

returns a circulant matrix with base x

**Usage**

```
## S3 method for class 'numeric'
circulant(x, ...)
```

**Arguments**

x                      an numeric object  
 ...                    additional arguments

**Value**

a circulant matrix with base x

---

circulantij              *circulantij function*

---

**Description**

A function to return the "idx" i.e. c(i,j) element of a circulant matrix with base "base".

**Usage**

```
circulantij(idx, base)
```

**Arguments**

idx                    vector of length 2 th (i,j) (row,column) index to return  
 base                   the base matrix of a circulant matrix

**Value**

the ij element of the full circulant

---

covmodel	<i>covmodel function</i>
----------	--------------------------

---

**Description**

A function to define the spatial covariance model, see also ?CovarianceFct. Note that the parameters defined by the 'pars' argument are fixed, i.e. not estimated by the MCMC algorithm. To have spatsurv estimate these parameters, the user must construct a new covariance function to do so, stop("") see the spatsurv vignette.

**Usage**

```
covmodel(model, pars)
```

**Arguments**

model	correlation type, a string see ?CovarianceFct
pars	vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in ?CovarianceFct and are not estimated

**Value**

an object of class covmodel

---

CSplot	<i>CSplot function</i>
--------	------------------------

---

**Description**

A function to produce a diagnostic plot for model fit using the Cox-Snell residuals.

**Usage**

```
CSplot(mod, plot = TRUE, bw = FALSE, ...)
```

**Arguments**

mod	an object produced by the function survspat
plot	whether to plot the result, default is TRUE
bw	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
...	other arguments to pass to plot

**Value**

the x and y values used in the plot

cumbasehazard                    *cumbasehazard function*

---

**Description**

Generic function for computing the cumulative baseline hazard

**Usage**

```
cumbasehazard(obj, ...)
```

**Arguments**

obj                    an object  
...                    additional arguments – currently there are none, but this is for extensibility

**Value**

method cumbasehazard

**See Also**

[cumbasehazard.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpow-Haz](#)

---

cumbasehazard.basehazardspec  
                                  *cumbasehazard.basehazardspec function*

---

**Description**

A function to retrieve the cumulative baseline hazard function

**Usage**

```
## S3 method for class 'basehazardspec'  
cumbasehazard(obj, ...)
```

**Arguments**

obj                    an object of class basehazardspec  
...                    additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the cumulative baseline hazard

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

cumulativeBspline.construct

*cumulativeBspline.construct function*

---

**Description**

A function to construct the integral of a B-spline curve given data and basis coefficients. Used in evaluating the cumulative baseline hazard.

**Usage**

```
cumulativeBspline.construct(x, basis)
```

**Arguments**

x	a vector, the data
basis	an object created by the getBbasis function

**Value**

an object that allows the integral of a given B-spline curve to be computed

---

densityquantile

*densityquantile function*

---

**Description**

Generic function for computing quantiles of the density function for a given baseline hazard. This may not be analytically tractable.

**Usage**

```
densityquantile(obj, ...)
```

**Arguments**

obj	an object
...	additional arguments – currently there are none, but this is for extensibility

**Value**

method densityquantile

**See Also**

[densityquantile.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

densityquantile.basehazardspec  
*densityquantile.basehazardspec function*

---

**Description**

A function to retrieve the quantiles of the density function

**Usage**

```
## S3 method for class 'basehazardspec'
densityquantile(obj, ...)
```

**Arguments**

obj            an object of class basehazardspec  
...            additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the density quantiles

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

densityquantile\_PP    *densityquantile\_PP function*

---

**Description**

A function to compute quantiles of the density function

**Usage**

```
densityquantile_PP(inputs)
```

**Arguments**

inputs            inputs for the function including the model matrix, frailties, fixed effects and the parameters of the baseline hazard derived from this model

**Value**

quantiles of the density function for the individual

---

density\_PP

*density\_PP function*

---

**Description**

A function to compute an individual's density function

**Usage**

```
density_PP(inputs)
```

**Arguments**

inputs            inputs for the function including the model matrix, frailties, fixed effects and the parameters of the baseline hazard derived from this model

**Value**

the density function for the individual

---

derivindepGaussianprior

*derivindepGaussianprior function*

---

**Description**

A function for evaluating the first and second derivatives of the log of an independent Gaussian prior

**Usage**

```
derivindepGaussianprior(beta = NULL, omega = NULL, eta = NULL, priors)
```

**Arguments**

beta            a vector, the parameter beta  
 omega          a vector, the parameter omega  
 eta            a vector, the parameter eta  
 priors        an object of class 'mcmcPrior', see ?mcmcPrior

**Value**

returns the first and second derivatives of the prior

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

derivindepGaussianpriorST

*derivindepGaussianpriorST function*

---

**Description**

A function to

**Usage**

```
derivindepGaussianpriorST(beta = NULL, omega = NULL, eta = NULL, priors)
```

**Arguments**

beta	X
omega	X
eta	X
priors	X

**Value**

...

---

derivpsplineprior

*derivpsplineprior function*

---

**Description**

A function for evaluating the first and second derivatives of the log of an independent Gaussian prior

**Usage**

```
derivpsplineprior(beta = NULL, omega = NULL, eta = NULL, priors)
```

**Arguments**

beta	a vector, the parameter beta
omega	a vector, the parameter omega
eta	a vector, the parameter eta
priors	an object of class 'mcmcPrior', see ?mcmcPrior

**Value**

returns the first and second derivatives of the prior

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

distinfo	<i>distinfo function</i>
----------	--------------------------

---

**Description**

Generic function for returning information about the class of baseline hazard functions employed.

**Usage**

```
distinfo(obj, ...)
```

**Arguments**

obj	an object
...	additional argument – currently there are none, but this is for extensibility

**Value**

method distinfo

**See Also**

[distinfo.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

```
distinfo.basehazardspec
      distinfo.basehazardspec function
```

---

**Description**

A function to retrieve information on the baseline hazard distribution of choice

**Usage**

```
## S3 method for class 'basehazardspec'
distinfo(obj, ...)
```

**Arguments**

```
obj          an object of class basehazardspec
...          additional arguments – currently there are none, but this is for extensibility
```

**Value**

a function returning information on the baseline hazard distribution of choice

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

```
estimateY      estimateY function
```

---

**Description**

A function to get an initial estimate of Y, to be used in calibrating the MCMC. Not for general use

**Usage**

```
estimateY(X, betahat, omegahat, surv, control)
```

**Arguments**

```
X          the design matrix containing covariate information
betahat    an estimate of beta
omegahat   an estimate of omega
surv       an object of class Surv
control    a list containing various control parameters for the MCMC and post-processing routines
```

**Value**

an estimate of  $Y$ , to be used in calibrating the MCMC

---

etapriorGauss	<i>etapriorGauss function</i>
---------------	-------------------------------

---

**Description**

A function to define Gaussian priors for eta. This function simply stores a vector of means and standard deviations to be passed to the main MCMC function, survspat.

**Usage**

```
etapriorGauss(mean, sd)
```

**Arguments**

mean	the prior mean, a vector of length 1 or more. 1 implies a common mean.
sd	the prior standard deviation, a vector of length 1 or more. 1 implies a common standard deviation.

**Value**

an object of class "etapriorGauss"

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

Et_PP	<i>Et_PP function</i>
-------	-----------------------

---

**Description**

A function to compute an individual's approximate expected survival time using numerical integration. Note this appears to be unstable; the function is based on R's integrate function. Not intended for general use (yet!).

**Usage**

```
Et_PP(inputs)
```

**Arguments**

inputs            inputs for the function including the model matrix, frailties, fixed effects and the parameters of the baseline hazard derived from this model

**Value**

the expected survival time for the individual, obtained by numerical integration of the density function.

---

EvalCov            *EvalCov function*

---

**Description**

This function is used to evaluate the covariance function within the MCMC run. Not intended for general use.

**Usage**

```
EvalCov(cov.model, u, parameters)
```

**Arguments**

cov.model        an object of class covmodel  
u                vector of distances  
parameters      vector of parameters

**Value**

method EvalCov

---

ExponentialCovFct    *ExponentialCovFct function*

---

**Description**

A function to declare and also evaluate an exponential covariance function.

**Usage**

```
ExponentialCovFct()
```

**Value**

the exponential covariance function

**See Also**

[SpikedExponentialCovFct](#), [covmodel](#)

---

exponentialHaz

*exponentialHaz* function

---

**Description**

A function to define a parametric proportional hazards model where the baseline hazard is taken from the exponential model. This function returns an object inheriting class 'basehazardspec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'grad-cumbasehazard', 'hesscumbasehazard' and 'densityquantile'

**Usage**

```
exponentialHaz()
```

**Details**

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is `'densityquantile'`. In the case of the Weibull model for the baseline hazard, it can be shown that the  $q$ -th quantile is:

### Value

an object inheriting class `'basehazardspec'`

### See Also

[tpowHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#)

---

FFTgrid

*FFTgrid function*

---

### Description

A function to generate an FFT grid and associated quantities including cell dimensions, size of extended grid, centroids,

### Usage

```
FFTgrid(spatialdata, cellwidth, ext, boundingbox = NULL)
```

### Arguments

<code>spatialdata</code>	a <code>SpatialPixelsDataFrame</code> object
<code>cellwidth</code>	width of computational cells
<code>ext</code>	multiplying constant: the size of the extended grid: $ext * M$ by $ext * N$
<code>boundingbox</code>	optional bounding box over which to construct computational grid, supplied as an object on which the function <code>'bbox'</code> returns the bounding box

### Value

a list

---

fixedpars	<i>fixedpars function</i>
-----------	---------------------------

---

**Description**

A function to return the mcmc chains for the covariate effects

**Usage**

```
fixedpars(x)
```

**Arguments**

x                    an object of class `mcmcpatsurv`

**Value**

the beta mcmc chains

**See Also**

[print.mcmcpatsurv](#), [quantile.mcmcpatsurv](#), [summary.mcmcpatsurv](#), [vcov.mcmcpatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcpatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

fixmatrix	<i>fixmatrix function</i>
-----------	---------------------------

---

**Description**

!! THIS FUNCTION IS NOT INTENDED FOR GENERAL USE !!

**Usage**

```
fixmatrix(mat)
```

**Arguments**

mat                    a matrix

**Details**

A function to fix up an estimated covariance matrix using a VERY ad-hoc method.

**Value**

the fixed matrix

---

fixParHaz	<i>fixParHaz function</i>
-----------	---------------------------

---

**Description**

A function to

**Usage**

```
fixParHaz(bh, idx, fixval)
```

**Arguments**

bh	X
idx	X
fixval	X

**Value**

...

---

frailtylag1	<i>frailtylag1 function</i>
-------------	-----------------------------

---

**Description**

A function to produce a plot of, and return, the lag 1 (or higher, see argument 'lag') autocorrelation for each of the spatially correlated frailty chains

**Usage**

```
frailtylag1(object, plot = TRUE, lag = 1, ...)
```

**Arguments**

object	an object inheriting class <code>mcmcsurv</code>
plot	logical whether to plot the result, default is TRUE
lag	the lag to plot, the default is 1
...	other arguments to be passed to the plot function

**Value**

the lag 1 autocorrelation for each of the spatially correlated frailty chains

**See Also**

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

fs

*London Fire Brigade property*

---

**Description**

London Fire Brigade property

**Usage**

```
data(fs)
```

**Format**

data.frame

**Source**

<https://data.london.gov.uk/>

**References**

<https://data.london.gov.uk/>, <https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/>

**Examples**

```
fire <- data(fs)
```

---

fstimes

*London Fire Brigade response times to dwelling fires, 2009*

---

**Description**

London Fire Brigade response times to dwelling fires, 2009

**Usage**

```
data(fstimes)
```

**Format**

data.frame

**Source**

<https://data.london.gov.uk/>

**References**

<https://data.london.gov.uk/>, <https://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/>

**Examples**

```
firetimes <- data(fstimes)
```

---

gamma2risk

*gamma2risk function*

---

**Description**

A function to

**Usage**

```
gamma2risk(mod)
```

**Arguments**

mod                    X

**Value**

...

---

GammafromY

*GammafromY function*

---

**Description**

A function to change Ys (spatially correlated noise) into Gammas (white noise). Used in the MALA algorithm.

**Usage**

```
GammafromY(Y, rootQeigs, mu)
```

**Arguments**

Y	Y matrix
rootQeigs	square root of the eigenvectors of the precision matrix
mu	parameter of the latent Gaussian field

**Value**

Gamma

---

GammaFromY_SPDE	<i>GammaFromY_SPDE function</i>
-----------------	---------------------------------

---

**Description**

A function to go from Y to Gamma

**Usage**

```
GammaFromY_SPDE(Y, U, mu)
```

**Arguments**

Y	Y
U	upper Cholesky matrix
mu	the mean

**Value**

the value of Gamma for the given Y

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

gencens	<i>gencens function</i>
---------	-------------------------

---

**Description**

A function to generate observed times given a vector of true survival times and a vector of censoring times. Used in the simulation of survival data.

**Usage**

```
gencens(survtimes, centimes, type = "right")
```

**Arguments**

survtimes	a vector of survival times
centimes	a vector of censoring times for left or right censored data, 2-column matrix of censoring times for interval censoring (number of rows equal to the number of observations).
type	the type of censoring to generate can be 'right' (default), 'left' or 'interval'

**Value**

an object of class 'Surv', the censoring indicator is equal to 1 if the event is uncensored and 0 otherwise for right/left censored data, or for interval censored data, the indicator is 0 uncensored, 1 right censored, 2 left censored, or 3 interval censored.

---

getbb	<i>getbb function</i>
-------	-----------------------

---

**Description**

A function to get the bounding box of a Spatial object

**Usage**

```
getbb(obj)
```

**Arguments**

obj	a spatial object e.g. a SpatialPolygonsDataFrame, SpatialPolygons, etc ... anything with a bounding box that can be computed with bbox(obj)
-----	---

**Value**

a SpatialPolygons object: the bounding box

---

getBbasis	<i>getBbasis function</i>
-----------	---------------------------

---

**Description**

A function returning the piecewise polynomial coefficients for a B-spline basis function i.e. the basis functions.

**Usage**

```
getBbasis(x, knots, degree, force = FALSE)
```

**Arguments**

x	a vector of data
knots	a vector of knots in ascending order. The first and last knots must be respectively the minimum and maximum of x.
degree	the degree of the spline
force	logical: skip check on knots? (not recommended!)

**Value**

the knots and the piecewise polynomial coefficients for a B-spline basis function i.e. the basis functions.

---

getcov	<i>getcov function</i>
--------	------------------------

---

**Description**

A function to return the covariance from a model based on the randomFields covariance functions. Not intended for general use.

**Usage**

```
getcov(u, sigma, phi, model, pars)
```

**Arguments**

u	distance
sigma	variance parameter
phi	scale parameter
model	correlation type, see ?CovarianceFct
pars	vector of additional parameters for certain classes of covariance function (eg Matern), these must be supplied in the order given in ?CovarianceFct and are not estimated

**Value**

this is just a wrapper for CovarianceFct

---

getgrd	<i>getgrd function</i>
--------	------------------------

---

**Description**

A function to create a regular grid over an observation window in order to model the spatial random effects as a Gaussian Markov random field.

**Usage**

```
getgrd(shape, cellwidth)
```

**Arguments**

shape	an object of class SpatialPolygons or SpatialPolygonsDataFrame
cellwidth	a scalar, the width of the grid cells

**Value**

a SpatialPolygons object: the grid on which prediction of the spatial effects will occur

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

getGrid	<i>getGrid function</i>
---------	-------------------------

---

**Description**

A function to extract and return the computational grid from a gridded analysis.

**Usage**

```
getGrid(mod, returnclass = "SpatialPolygonsDataFrame")
```

**Arguments**

mod                    an object of class `mcmcspatsurv`, returned by the function `survspat`

returnclass          the class of object to return, default is a `'SpatialPolygonsDataFrame'`. Other options are `'raster'`, which returns a raster brick; or `'SpatialPixelsDataFrame'`

**Value**

a `SpatialPolygonsDataFrame` in which Monte Carlo expectations can be stored and later plotted.

---

getleneta	<i>getleneta function</i>
-----------	---------------------------

---

**Description**

A function to compute the length of eta

**Usage**

```
getleneta(cov.model)
```

**Arguments**

cov.model            a covariance model

**Value**

the length of eta

---

getOptCellwidth	<i>getOptCellwidth function</i>
-----------------	---------------------------------

---

**Description**

A function to compute an optimal cellwidth close to an initial suggestion. This maximises the efficiency of the MCMC algorithm when in the control argument of the function `survspat`, the option `gridded` is set to `TRUE`

**Usage**

```
getOptCellwidth(dat, cellwidth, ext = 2, plot = TRUE, boundingbox = NULL)
```

**Arguments**

<code>dat</code>	any spatial data object whose bounding box can be computed using the function <code>bbox</code> .
<code>cellwidth</code>	an initial suggested cellwidth
<code>ext</code>	the extension parameter for the FFT transform, set to 2 by default
<code>plot</code>	whether to plot the grid and data to illustrate the optimal grid
<code>boundingbox</code>	optional bounding box over which to construct computational grid, supplied as an object on which the function <code>'bbox'</code> returns the bounding box

**Value**

the optimum cell width

---

`getparranges`                      *getparranges function*

---

**Description**

A function to extract parameter ranges for creating a grid on which to evaluate the log-posterior, used in calibrating the MCMC. This function is not intended for general use.

**Usage**

```
getparranges(priors, leneta, mult = 1.96)
```

**Arguments**

<code>priors</code>	an object of class <code>mcmcPriors</code>
<code>leneta</code>	the length of <code>eta</code> passed to the function
<code>mult</code>	defaults to 1.96 so the grid formed will be mean plus/minus 1.96 times the standard deviation

**Value**

an appropriate range used to calibrate the MCMC: the mean of the prior for `eta` plus/minus 1.96 times the standard deviation

---

getsurvdata	<i>getsurvdata function</i>
-------------	-----------------------------

---

**Description**

A function to return the survival data from an object of class `mcmcspsurv`. This function is not intended for general use.

**Usage**

```
getsurvdata(x)
```

**Arguments**

`x` an object of class `mcmcspsurv`

**Value**

the survival data from an object of class `mcmcspsurv`

---

gompertzHaz	<i>gompertzHaz function</i>
-------------	-----------------------------

---

**Description**

A function to define a parametric proportional hazards model where the baseline hazard is taken from a Gompertz model. This function returns an object inheriting class `'basehazardspec'`, list of functions `'distinfo'`, `'basehazard'`, `'gradbasehazard'`, `'hessbasehazard'`, `'cumbasehazard'`, `'gradcumbasehazard'`, `'hesscumbasehazard'` and `'densityquantile'`

**Usage**

```
gompertzHaz()
```

**Details**

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The basehazard function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times,  $t$  and returns a vector.

The gradbasehazard function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times,  $t$ , and returns a matrix.

The hessbasehazard function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times,  $t$  and returns a list of hessian matrices corresponding to each  $t$ .

The cumbasehazard function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times,  $t$  and returns a vector.

The gradcumbasehazard function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times,  $t$ , and returns a matrix.

The hesscumbasehazard function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times,  $t$  and returns a list of hessian matrices corresponding to each  $t$ .

The densityquantile function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the predict function where type is 'densityquantile'. In the case of the Weibull model for the baseline hazard, it can be shown that the  $q$ -th quantile is:

### Value

an object inheriting class 'basehazardspec'

### See Also

[tpowHaz](#), [exponentialHaz](#), [makehamHaz](#), [weibullHaz](#)

---

gradbasehazard	<i>gradbasehazard function</i>
----------------	--------------------------------

---

### Description

Generic function for computing the gradient of the baseline hazard

### Usage

```
gradbasehazard(obj, ...)
```

### Arguments

obj	an object
...	additional arguments – currently there are none, but this is for extensibility

**Value**

method `gradbasehazard`

**See Also**

[gradbasehazard.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

`gradbasehazard.basehazardspec`  
*gradbasehazard.basehazardspec function*

---

**Description**

A function to retrieve the gradient of the baseline hazard function

**Usage**

```
## S3 method for class 'basehazardspec'  
gradbasehazard(obj, ...)
```

**Arguments**

`obj` an object of class `basehazardspec`  
`...` additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the gradient of the baseline hazard

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

gradcumbasehazard      *gradcumbasehazard function*

---

**Description**

Generic function for computing the gradient of the cumulative baseline hazard

**Usage**

```
gradcumbasehazard(obj, ...)
```

**Arguments**

obj                    an object  
 ...                    additional arguments – currently there are none, but this is for extensibility

**Value**

method gradcumbasehazard

**See Also**

[gradcumbasehazard.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpow-Haz](#)

---

gradcumbasehazard.basehazardspec  
                                  *gradcumbasehazard.basehazardspec function*

---

**Description**

A function to retrieve the gradient of the cumulative baseline hazard function

**Usage**

```
## S3 method for class 'basehazardspec'  
gradcumbasehazard(obj, ...)
```

**Arguments**

obj                    an object of class basehazardspec  
 ...                    additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the gradient of the cumulative baseline hazard

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

grid2spdf	<i>grid2spdf function</i>
-----------	---------------------------

---

**Description**

A function to convert a regular (x,y) grid of centroids into a SpatialPoints object

**Usage**

```
grid2spdf(xgrid, ygrid, proj4string = CRS(as.character(NA)))
```

**Arguments**

xgrid	vector of x centroids (equally spaced)
ygrid	vector of x centroids (equally spaced)
proj4string	an optional proj4string, projection string for the grid, set using the function CRS

**Value**

a SpatialPolygonsDataFrame

---

grid2spix	<i>grid2spix function</i>
-----------	---------------------------

---

**Description**

A function to convert a regular (x,y) grid of centroids into a SpatialPixels object

**Usage**

```
grid2spix(xgrid, ygrid, proj4string = CRS(as.character(NA)))
```

**Arguments**

xgrid	vector of x centroids (equally spaced)
ygrid	vector of x centroids (equally spaced)
proj4string	an optional proj4string, projection string for the grid, set using the function CRS

**Value**

a SpatialPixels object

---

`grid2spts`*grid2spts function*

---

**Description**

A function to convert a regular (x,y) grid of centroids into a SpatialPoints object

**Usage**

```
grid2spts(xgrid, ygrid, proj4string = CRS(as.character(NA)))
```

**Arguments**

<code>xgrid</code>	vector of x centroids (equally spaced)
<code>ygrid</code>	vector of x centroids (equally spaced)
<code>proj4string</code>	an optional proj4string, projection string for the grid, set using the function CRS

**Value**

a SpatialPoints object

---

`gridY`*gridY function*

---

**Description**

A function to put estimated individual Y's onto a grid

**Usage**

```
gridY(Y, control)
```

**Arguments**

<code>Y</code>	estimate of Y
<code>control</code>	control parameters

**Value**

...

---

`gridY_polygonal`      *gridY\_polygonal function*

---

**Description**

A function to put estimated individual Y's onto a grid

**Usage**

```
gridY_polygonal(Y, control)
```

**Arguments**

Y	estimate of Y
control	control parameters

**Value**

...

---

`guess_t`      *guess\_t function*

---

**Description**

A function to get an initial guess of the failure time t, to be used in calibrating the MCMC. Not for general use

**Usage**

```
guess_t(surv)
```

**Arguments**

surv	an object of class Surv
------	-------------------------

**Value**

a guess at the failure times

---

hasNext	<i>generic hasNext method</i>
---------	-------------------------------

---

**Description**

test if an iterator has any more values to go

**Usage**

```
hasNext(obj)
```

**Arguments**

obj	an iterator
-----	-------------

---

hasNext.iter	<i>hasNext.iter function</i>
--------------	------------------------------

---

**Description**

method for iter objects test if an iterator has any more values to go

**Usage**

```
## S3 method for class 'iter'
hasNext(obj)
```

**Arguments**

obj	an iterator
-----	-------------

---

hazardexceedance	<i>hazardexceedance function</i>
------------------	----------------------------------

---

**Description**

A function to compute exceedance probabilities for the spatially correlated frailties.

**Usage**

```
hazardexceedance(threshold, direction = "upper")
```

**Arguments**

threshold	vector of thresholds
direction	default is "upper" which will calculate $P(Y > \text{threshold})$ , alternative is "lower", which will calculate $P(Y < \text{threshold})$

**Value**

a function that can be passed to the function MCE in order to compute the exceedance probabilities

**See Also**

[print.mcmcpatsurv](#), [quantile.mcmcpatsurv](#), [summary.mcmcpatsurv](#), [vcov.mcmcpatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcpatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#),

---

hazardpars	<i>hazardpars function</i>
------------	----------------------------

---

**Description**

A function to return the mcmc chains for the hazard function parameters

**Usage**

```
hazardpars(x)
```

**Arguments**

x an object of class mcmcpatsurv

**Value**

the omega mcmc chains

**See Also**

[print.mcmcpatsurv](#), [quantile.mcmcpatsurv](#), [summary.mcmcpatsurv](#), [vcov.mcmcpatsurv](#), [frailty-lag1](#), [spatialpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcpatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

hazard_PP	<i>hazard_PP function</i>
-----------	---------------------------

---

**Description**

A function to compute an individual's hazard function.

**Usage**

```
hazard_PP(inputs)
```

**Arguments**

inputs	inputs for the function including the model matrix, frailties, fixed effects and the parameters of the baseline hazard derived from this model
--------	--

**Value**

the hazard function for the individual

---

hessbasehazard	<i>hessbasehazard function</i>
----------------	--------------------------------

---

**Description**

Generic function for computing the hessian of the baseline hazard

**Usage**

```
hessbasehazard(obj, ...)
```

**Arguments**

obj	an object
...	additional arguments – currently there are none, but this is for extensibility

**Value**

method hessbasehazard

**See Also**

[hessbasehazard.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpow-Haz](#)

---

hessbasehazard.basehazardspec  
*hessbasehazard.basehazardspec function*

---

**Description**

A function to retrieve the Hessian of the baseline hazard function

**Usage**

```
## S3 method for class 'basehazardspec'  
hessbasehazard(obj, ...)
```

**Arguments**

obj                    an object of class basehazardspec  
...                    additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the Hessian of the baseline hazard

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

hesscumbasehazard      *hesscumbasehazard function*

---

**Description**

Generic function for computing the Hessian of the cumulative baseline hazard

**Usage**

```
hesscumbasehazard(obj, ...)
```

**Arguments**

obj                    an object  
...                    additional arguments – currently there are none, but this is for extensibility

**Value**

method hesscumbasehazard

**See Also**

[hesscumbasehazard.basehazardspec](#), [exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

`hesscumbasehazard.basehazardspec`  
*hesscumbasehazard.basehazardspec function*

---

**Description**

A function to retrieve the hessian of the cumulative baseline hazard function

**Usage**

```
## S3 method for class 'basehazardspec'
hesscumbasehazard(obj, ...)
```

**Arguments**

`obj`                    an object of class `basehazardspec`  
`...`                    additional arguments – currently there are none, but this is for extensibility

**Value**

a function returning the hessian of the cumulative baseline hazard

**See Also**

[exponentialHaz](#), [weibullHaz](#), [gompertzHaz](#), [makehamHaz](#), [tpowHaz](#)

---

`imputationModel`            *imputationModel function*

---

**Description**

A function to

**Usage**

```
imputationModel(formula, offset, covariateData, priors)
```

**Arguments**

formula	X
offset	X
covariateData	X
priors	X

**Value**

...

---

Independent	<i>Independent function</i>
-------------	-----------------------------

---

**Description**

A function to declare and also evaluate an exponential covariance function.

**Usage**

Independent()

**Value**

the exponential covariance function

**See Also**

[SpikedExponentialCovFct](#), [covmodel](#)

---

indepGaussianprior	<i>indepGaussianprior function</i>
--------------------	------------------------------------

---

**Description**

A function for evaluating the log of an independent Gaussian prior for a given set of parameter values.

**Usage**

indepGaussianprior(beta = NULL, omega = NULL, eta = NULL, priors)

**Arguments**

beta	parameter beta at which prior is to be evaluated
omega	parameter omega at which prior is to be evaluated
eta	parameter eta at which prior is to be evaluated
priors	an object of class <code>mcmcPriors</code> , see <code>?mcmcPriors</code>

**Value**

the log of the prior evaluated at the given parameter values

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

`indepGaussianpriorST` *indepGaussianpriorST function*

---

**Description**

A function to

**Usage**

```
indepGaussianpriorST(beta = NULL, omega = NULL, eta = NULL, priors)
```

**Arguments**

beta	X
omega	X
eta	X
priors	X

**Value**

...

---

inference.control      *inference.control function*


---

### Description

A function to control inferential settings. This function is used to set parameters for more advanced use of spatsurv.

### Usage

```
inference.control(
  gridded = FALSE,
  cellwidth = NULL,
  ext = 2,
  imputation = NULL,
  optimcontrol = NULL,
  hessian = FALSE,
  plotcal = FALSE,
  timeonlyMCMC = FALSE,
  nugget = FALSE,
  savenugget = FALSE,
  split = 0.5,
  logUsigma_priormean = 0,
  logUsigma_priorsd = 0.5,
  nis = NULL,
  olinfo = NULL
)
```

### Arguments

gridded	logical. Whether to perform computation on a grid. Default is FALSE.
cellwidth	the width of computational cells to use
ext	integer the number of times to extend the computational grid by in order to perform computation. The default is 2.
imputation	for polygonal data, an optional model for inference at the sub-polygonal level, see function <code>imputationModel</code>
optimcontrol	a list of optional arguments to be passed to <code>optim</code> for non-spatial models
hessian	whether to return a numerical hessian. Set this to TRUE for non-spatial models. equal to the number of parameters of the baseline hazard
plotcal	logical, whether to produce plots of the MCMC calibration process, this is a technical option and should onyl be set to TRUE if poor mixing is evident (the printed h is low), then it is also useful to use a graphics device with multiple plotting windows.
timeonlyMCMC	logical, whether to only time the MCMC part of the algorithm, or whether to include in the reported running time the time taken to calibrate the method (default)

nugget	whether to include a nugget effect in the estimation. Note that only the mean and variance of the nugget effect is returned.
savenugget	whether to save the MCMC chain for the nugget effect
split	how to split the spatial and nugget proposal variance as a the proportion of variance assigned to the spatial effect apriori. Default is 0.5
logUsigma_priormean	prior mean for log standard deviation of nugget effect
logUsigma_priorsd	prior sd for log standard deviation of nugget effect
nis	list of cell counts, each element being a matrix, with attributes "x" and "y" giving grid centroids in x and y directions. Used to impute locations of aggregated data.
olinfo	to be supplied with nis, if continuous inference from aggregated data is required

**Value**

returns parameters to be used in the function `survspat`

**See Also**

[survspat](#)

---

insert

*insert function*

---

**Description**

A function to

**Usage**

```
insert(pars, idx, val)
```

**Arguments**

pars	X
idx	X
val	X

**Value**

...

---

invtransformweibull    *invtransformweibull function*

---

**Description**

A function to transform estimates of the (alpha, lambda) parameters of the weibull baseline hazard function, so they are commensurate with R's inbuilt density functions, (shape, scale).

**Usage**

```
invtransformweibull(x)
```

**Arguments**

x                    a vector of paramters

**Value**

the transformed parameters. For the weibull model, this transforms 'shape' 'scale' (see ?dweibull) to 'alpha' and 'lambda' for the MCMC

---

is.burnin            *is this a burn-in iteration?*

---

**Description**

if this mcmc iteration is in the burn-in period, return TRUE

**Usage**

```
is.burnin(obj)
```

**Arguments**

obj                    an mcmc iterator

**Value**

TRUE or FALSE

---

<code>is.retain</code>	<i>do we retain this iteration?</i>
------------------------	-------------------------------------

---

**Description**

if this mcmc iteration is one not thinned out, this is true

**Usage**

```
is.retain(obj)
```

**Arguments**

`obj` an mcmc iterator

**Value**

TRUE or FALSE

---

<code>iteration</code>	<i>iteration number</i>
------------------------	-------------------------

---

**Description**

within a loop, this is the iteration number we are currently doing.

**Usage**

```
iteration(obj)
```

**Arguments**

`obj` an mcmc iterator

**Details**

get the iteration number

**Value**

integer iteration number, starting from 1.

---

logPosterior	<i>logPosterior function</i>
--------------	------------------------------

---

**Description**

A function to evaluate the log-posterior of a spatial parametric proportional hazards model. Not intended for general use.

**Usage**

```
logPosterior(
  surv,
  X,
  beta,
  omega,
  eta,
  gamma,
  priors,
  cov.model,
  u,
  control,
  gradient = FALSE,
  hessian = FALSE
)
```

**Arguments**

surv	an object of class Surv
X	the design matrix, containing covariate information
beta	parameter beta
omega	parameter omega
eta	parameter eta
gamma	parameter gamma
priors	the priors, an object of class 'mcmcPriors'
cov.model	the spatial covariance model
u	vector of interpoint distances
control	a list containing various control parameters for the MCMC and post-processing routines
gradient	logical whether to evaluate the gradient
hessian	logical whether to evaluate the Hessian

**Value**

evaluates the log-posterior and the gradient and hessian, if required.

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). *spatsurv: An R Package for Bayesian Inference with Spatial Survival Models*. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.

---

`logPosterior_gridded` *logPosterior\_gridded function*

---

**Description**

A function to evaluate the log-posterior of a spatial parametric proportional hazards model using gridded Y. Not intended for general use.

**Usage**

```
logPosterior_gridded(
  surv,
  X,
  beta,
  omega,
  eta,
  gamma,
  priors,
  cov.model,
  u,
  control,
  gradient = FALSE,
  hessian = FALSE
)
```

**Arguments**

<code>surv</code>	an object of class <code>Surv</code>
<code>X</code>	the design matrix, containing covariate information
<code>beta</code>	parameter <code>beta</code>
<code>omega</code>	parameter <code>omega</code>
<code>eta</code>	parameter <code>eta</code>
<code>gamma</code>	parameter <code>gamma</code>
<code>priors</code>	the priors, an object of class <code>'mcmcPriors'</code>
<code>cov.model</code>	the spatial covariance model
<code>u</code>	vector of interpoint distances
<code>control</code>	a list containing various control parameters for the MCMC and post-processing routines
<code>gradient</code>	logical whether to evaluate the gradient
<code>hessian</code>	logical whether to evaluate the Hessian

**Value**

evaluates the log-posterior and the gradient and hessian, if required.

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.

---

logPosterior\_polygonal

*logPosterior\_polygonal function*

---

**Description**

A function to evaluate the log-posterior of a spatial parametric proportional hazards model. Not intended for general use.

**Usage**

```
logPosterior_polygonal(
  surv,
  X,
  beta,
  omega,
  eta,
  gamma,
  priors,
  cov.model,
  u,
  control,
  gradient = FALSE,
  hessian = FALSE
)
```

**Arguments**

surv	an object of class Surv
X	the design matrix, containing covariate information
beta	parameter beta
omega	parameter omega
eta	parameter eta
gamma	parameter gamma
priors	the priors, an object of class 'mcmcPriors'
cov.model	the spatial covariance model

u	vector of interpoint distances
control	a list containing various control parameters for the MCMC and post-processing routines
gradient	logical whether to evaluate the gradient
hessian	logical whether to evaluate the Hessian

**Value**

evaluates the log-posterior and the gradient and hessian, if required.

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.

---

logPosterior\_SPDE      *logPosterior\_SPDE function*

---

**Description**

A function to evaluate the log-posterior of a spatial parametric proportional hazards model. Not intended for general use.

**Usage**

```
logPosterior_SPDE(
  surv,
  X,
  beta,
  omega,
  eta,
  gamma,
  priors,
  cov.model,
  u,
  control,
  gradient = FALSE,
  hessian = FALSE
)
```

**Arguments**

surv	an object of class Surv
X	the design matrix, containing covariate information
beta	parameter beta
omega	parameter omega

eta	parameter eta
gamma	parameter gamma
priors	the priors, an object of class 'mcmcPriors'
cov.model	the spatial covariance model
u	vector of interpoint distances
control	a list containing various control parameters for the MCMC and post-processing routines
gradient	logical whether to evaluate the gradient
hessian	logical whether to evaluate the Hessian

**Value**

evaluates the log-posterior and the gradient and hessian, if required.

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

loop.mcmc	<i>loop over an iterator</i>
-----------	------------------------------

---

**Description**

useful for testing progress bars

**Usage**

```
loop.mcmc(object, sleep = 1)
```

**Arguments**

object	an mcmc iterator
sleep	pause between iterations in seconds

---

`makehamHaz`*makehamHaz function*

---

### Description

A function to define a parametric proportional hazards model where the baseline hazard is taken from the Gompertz-Makeham model. This function returns an object inheriting class 'basehazard-spec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'gradcumbasehazard', 'hesscumbasehazard' and 'densityquantile'

### Usage

```
makehamHaz()
```

### Details

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is 'densityquantile'. In the case of the Weibull model for the baseline hazard, it can be shown that the `q`-th quantile is:

**Value**

an object inheriting class 'basehazardspec'

**See Also**

[tpowHaz](#), [exponentialHaz](#), [gompertzHaz](#), [weibullHaz](#)

---

maxlikparamPHsurv      *maxlikparamPHsurv function*

---

**Description**

A function to get initial estimates of model parameters using maximum likelihood. Not intended for general purpose use.

**Usage**

```
maxlikparamPHsurv(surv, X, control)
```

**Arguments**

surv	an object of class Surv
X	the design matrix, containing covariate information
control	a list containing various control parameters for the MCMC and post-processing routines

**Value**

initial estimates of the parameters

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.

---

MCE	<i>MCE function</i>
-----	---------------------

---

**Description**

A function to compute Monte Carlo expectations from an object inheriting class memcspatsurv

**Usage**

```
MCE(object, fun)
```

**Arguments**

object	an object inheriting class memcspatsurv
fun	a function with arguments beta, omega, eta and Y

**Value**

the Monte Carlo mean of the function over the posterior.

**See Also**

[print.memcspatsurv](#), [quantile.memcspatsurv](#), [summary.memcspatsurv](#), [vcov.memcspatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.memcspatsurv](#), [prior-posterior](#), [posteriorcov](#), [hazardexceedance](#)

---

mcmcLoop	<i>iterator for MCMC loops</i>
----------	--------------------------------

---

**Description**

control an MCMC loop with this iterator

**Usage**

```
mcmcLoop(N, burnin, thin, trim = TRUE, progressor = mcmcProgressPrint)
```

**Arguments**

N	number of iterations
burnin	length of burn-in
thin	frequency of thinning
trim	whether to cut off iterations after the last retained iteration
progressor	a function that returns a progress object

---

mcmcparams	<i>mcmcparams function</i>
------------	----------------------------

---

**Description**

A function for setting MCMC options.

**Usage**

```
mcmcparams(nits, burn, thin, inits = NULL, adaptivescheme = NULL)
```

**Arguments**

nits	numer of iterations,
burn	length of burnin
thin	thinning parameter eg operated on chain every 'thin' iteration (eg store output or compute some posterior functional)
inits	NOT CURRENTLY IN USE
adaptivescheme	NOT CURRENTLY IN USE

**Value**

mcmc parameters

---

mcmcPriors	<i>mcmcPriors function</i>
------------	----------------------------

---

**Description**

A function to define priors for the MCMC.

**Usage**

```
mcmcPriors(  
  betaprior = NULL,  
  omegaprior = NULL,  
  etaprior = NULL,  
  call = NULL,  
  derivative = NULL  
)
```

**Arguments**

betaprior	prior for beta, the covariate effects
omegaprior	prior for omega, the parameters of the baseline hazard
etaprior	prior for eta, the parameters of the latent field
call	function to evaluate the log-prior e.g. <code>logindepGaussianprior</code>
derivative	function to evaluate the first and second derivatives of the prior

**Details**

The package `spatsurv` only provides functionality for the built-in Gaussian priors. However, the choice of prior is extensible by the user by creating functions similar to the functions `betapriorGauss`, `omegapriorGauss`, `etapriorGauss`, `indepGaussianprior` and `derivindepGaussianprior`: the first three of which provide a mechanism for storing and retrieving the parameters of the priors; the fourth, a function for evaluating the log of the prior for a given set of parameter values; and the fifth, a function for evaluating the first and second derivatives of the log of the prior. It is assumed that parameters are a priori independent. The user interested in using other priors is encouraged to look at the structure of the five functions mentioned above.

**Value**

an object of class `mcmcPriors`

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

<code>mcmcProgressNone</code>	<i>null progress monitor</i>
-------------------------------	------------------------------

---

**Description**

a progress monitor that does nothing

**Usage**

```
mcmcProgressNone(mcmcloop)
```

**Arguments**

<code>mcmcloop</code>	an mcmc loop iterator
-----------------------	-----------------------

**Value**

a progress monitor

---

`mcmcProgressPrint`     *printing progress monitor*

---

**Description**

a progress monitor that prints each iteration

**Usage**

`mcmcProgressPrint(mcmcloop)`

**Arguments**

`mcmcloop`     an mcmc loop iterator

**Value**

a progress monitor

---

`mcmcProgressTextBar`     *text bar progress monitor*

---

**Description**

a progress monitor that uses a text progress bar

**Usage**

`mcmcProgressTextBar(mcmcloop)`

**Arguments**

`mcmcloop`     an mcmc loop iterator

**Value**

a progress monitor

---

midpts	<i>midpts function</i>
--------	------------------------

---

**Description**

A function to compute the midpoints of a vector

**Usage**

```
midpts(x)
```

**Arguments**

x	a vector
---	----------

**Value**

the midpoints, a vector of length length(x)-1

---

multiWayHaz	<i>multiWayHaz function</i>
-------------	-----------------------------

---

**Description**

A function to

**Usage**

```
multiWayHaz(bhlist, bhtime, bhfix, MLimits = NULL)
```

**Arguments**

bhlist	X
bhtime	X
bhfix	X
MLimits	X

**Value**

...

---

neighLocs	<i>neighLocs function</i>
-----------	---------------------------

---

**Description**

A function used in the computation of neighbours on non-rectangular grids. Not intended for general use.

**Usage**

```
neighLocs(coord, cellwidth, order)
```

**Arguments**

coord	coordinate of interest
cellwidth	a scalar, the width of the grid cells
order	the order of the SPDE approximation: see Lindgren et al 2011 for details

**Value**

coordinates of centroids of neighbours

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

neighOrder	<i>neighOrder function</i>
------------	----------------------------

---

**Description**

A function to compute the order of a set of neighbours. Not intended for general use.

**Usage**

```
neighOrder(neighlocs)
```

**Arguments**

neighlocs	an object created by the function neighLocs
-----------	---

**Value**

the neighbour orders

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

nextStep	<i>next step of an MCMC chain</i>
----------	-----------------------------------

---

**Description**

just a wrapper for nextElem really.

**Usage**

```
nextStep(object)
```

**Arguments**

object	an mcmc loop object
--------	---------------------

---

NonSpatialLogLikelihood_or_gradient	<i>NonSpatialLogLikelihood_or_gradient function</i>
-------------------------------------	---

---

**Description**

A function to evaluate the log-likelihood of a non-spatial parametric proportional hazards model. Not intended for general use.

**Usage**

```
NonSpatialLogLikelihood_or_gradient(
  surv,
  X,
  beta,
  omega,
  control,
  loglikelihood,
  gradient
)
```

**Arguments**

surv	an object of class Surv
X	the design matrix, containing covariate information
beta	parameter beta
omega	parameter omega
control	a list containg various control parameters for the MCMC and post-processing routines
loglikelihood	logical whether to evaluate the log-likelihood
gradient	logical whether to evaluate the gradient

**Value**

...

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.

---

omegapriorGauss      *omegapriorGauss function*

---

**Description**

A function to define Gaussian priors for omega. This function simply stores a vector of means and standard deviations to be passed to the main MCMC function, survspat.

**Usage**

```
omegapriorGauss(mean, sd)
```

**Arguments**

mean	the prior mean, a vector of length 1 or more. 1 implies a common mean.
sd	the prior standard deviation, a vector of length 1 or more. 1 implies a common standard deviation.

**Value**

an object of class "omegapriorGauss"

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

omegapriorGaussST	<i>omegapriorGaussST function</i>
-------------------	-----------------------------------

---

**Description**

A function to

**Usage**

```
omegapriorGaussST(basehaz, fmean, fsd, taumean, tausd, thetamean, thetasd)
```

**Arguments**

basehaz	X
fmean	X
fsd	X
taumean	X
tausd	X
thetamean	X
thetasd	X

**Value**

...

---

optifix	<i>optifix function</i>
---------	-------------------------

---

**Description**

optifix. Optimise with fixed parameters

**Usage**

```
optifix(
  par,
  fixed,
  fn,
  gr = NULL,
  ...,
  method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN"),
  lower = -Inf,
  upper = Inf,
  control = list(),
  hessian = FALSE
)
```

**Arguments**

par	X
fixed	X
fn	X
gr	X
...	X
method	X
lower	X
upper	X
control	X
hessian	X

**Details**

its like optim, but with fixed parameters.

specify a second argument 'fixed', a vector of TRUE/FALSE values. If TRUE, the corresponding parameter in fn() is fixed. Otherwise its variable and optimised over.

The return thing is the return thing from optim() but with a couple of extra bits - a vector of all the parameters and a vector copy of the 'fixed' argument.

Written by Barry Rowlingson <b.rowlingson@lancaster.ac.uk> October 2011

This file released under a CC By-SA license: <http://creativecommons.org/licenses/by-sa/3.0/>

and must retain the text: "Originally written by Barry Rowlingson" in comments.

**Value**

...

---

plot.FFTgrid	<i>plot.FFTgrid function</i>
--------------	------------------------------

---

**Description**

A function to

**Usage**

```
## S3 method for class 'FFTgrid'
plot(x, y = NULL, ...)
```

**Arguments**

x	X
y	X
...	X

**Value**

...

plotsurv

*plotsurv function***Description**

A function to produce a 2-D plot of right censored spatial survival data.

**Usage**

```
plotsurv(
  spp,
  ss,
  maxcex = 1,
  transform = identity,
  background = NULL,
  eventpt = 19,
  eventcol = "red",
  censpt = "+",
  censcol = "black",
  xlim = NULL,
  ylim = NULL,
  xlab = NULL,
  ylab = NULL,
  add = FALSE,
  ...
)
```

**Arguments**

spp	A spatial points data frame
ss	A Surv object (with right-censoring)
maxcex	maximum size of dots default is equivalent to setting cex equal to 1
transform	optional transformation to apply to the data, a function, for example 'sqrt'
background	a background object to plot default is null, which gives a blank background note that if non-null, the parameters xlim and ylim will be derived from this object.
eventpt	The type of point to illustrate events, default is 19 (see ?pch)
eventcol	the colour of events, default is black
censpt	The type of point to illustrate events, default is "+" (see ?pch)
censcol	the colour of censored observations, default is red
xlim	optional x-limits of plot, default is to choose this automatically

<i>yylim</i>	optional y-limits of plot, default is to choose this automatically
<i>xlab</i>	label for x-axis
<i>ylab</i>	label for y-axis
<i>add</i>	logical, whether to add the survival plot on top of an existing plot, default is FALSE, which produces a plot in a new device
<i>...</i>	other arguments to pass to plot

**Value**

Plots the survival data non-censored observations appear as dots and censored observations as crosses. The size of the dot is proportional to the observed time.

---

<i>polyadd</i>	<i>polyadd function</i>
----------------	-------------------------

---

**Description**

A function to add two polynomials in the form of vectors of coefficients. The first element of the vector being the constant (order 0) term

**Usage**

```
polyadd(poly1, poly2)
```

**Arguments**

<i>poly1</i>	a vector of coefficients for the first polynomial of length degree plus 1
<i>poly2</i>	a vector of coefficients for the second polynomial of length degree plus 1

**Value**

the coefficients of the sum of *poly1* and *poly2*

---

polymult	<i>polymult function</i>
----------	--------------------------

---

**Description**

A function to multiply two polynomials in the form of vectors of coefficients. The first element of the vector being the constant (order 0) term

**Usage**

```
polymult(poly1, poly2)
```

**Arguments**

poly1	a vector of coefficients for the first polynomial of length degree plus 1
poly2	a vector of coefficients for the second polynomial of length degree plus 1

**Value**

the coefficients of the product of poly1 and poly2

---

posteriorcov	<i>posteriorcov function</i>
--------------	------------------------------

---

**Description**

A function to produce a plot of the posterior covariance function with upper and lower quantiles.

**Usage**

```
posteriorcov(  
  x,  
  probs = c(0.025, 0.5, 0.975),  
  rmax = NULL,  
  n = 100,  
  plot = TRUE,  
  bw = FALSE,  
  corr = FALSE,  
  ...  
)
```

**Arguments**

x	an object of class mcmcspatsurv
probs	vector of probabilities to be fed to quantile function
rmax	maximum distance in space to compute this distance up to
n	the number of points at which to evaluate the posterior covariance.
plot	whether to plot the result
bw	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
corr	logical whether to return the correlation function, default is FALSE i.e. returns the covariance function
...	other arguments to be passed to matplot function

**Value**

produces a plot of the posterior spatial covariance function.

**See Also**

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [prior-posterior](#), [MCE](#), [hazardexceedance](#)

---

predict.mcmcspatsurv *predict.mcmcspatsurv function*

---

**Description**

A function to produce predictions from MCMC output. These could include quantiles of the individual density, survival or hazard functions or quantiles of the density function (if available analytically).

**Usage**

```
## S3 method for class 'mcmcspatsurv'
predict(
  object,
  type = "density",
  t = NULL,
  n = 110,
  indx = NULL,
  probs = c(0.025, 0.5, 0.975),
  plot = TRUE,
  pause = TRUE,
  bw = FALSE,
  ...
)
```

**Arguments**

object	an object of class <code>mcmcspatsurv</code>
type	can be "density", "hazard", "survival" or "densityquantile". Default is "density". Note that "densityquantile" is not always analytically tractable for some choices of baseline hazard function.
t	optional vector of times at which to compute the quantiles, Default is NULL, in which case a uniformly spaced vector of length n from 0 to the maximum time is used
n	the number of points at which to compute the quantiles if t is NULL
indx	the index number of a particular individual or vector of indices of individuals for which the quantiles should be produced
probs	vector of probabilities
plot	whether to plot the result
pause	logical whether to pause between plots, the default is TRUE
bw	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
...	other arguments, not used here

**Value**

the required predictions

**See Also**

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

`print.mcmc`

*print.mcmc function*

---

**Description**

print method print an mcmc iterator's details

**Usage**

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

**Arguments**

x	a mcmc iterator
...	other args

---

print.mcmcspatsurv     *print.mcmcspatsurv function*

---

**Description**

A function to print summary tables from an MCMC run

**Usage**

```
## S3 method for class 'mcmcspatsurv'  
print(x, probs = c(0.5, 0.025, 0.975), digits = 3, scientific = -3, ...)
```

**Arguments**

x	an object inheriting class mcmcspatsurv
probs	vector of quantiles to return
digits	see help file ?format
scientific	see help file ?format
...	additional arguments, not used here

**Value**

prints summary tables to the console

**See Also**

[quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

print.mlspatsurv     *print.mlspatsurv function*

---

**Description**

A function to print summary tables from an MCMC run

**Usage**

```
## S3 method for class 'mlspatsurv'  
print(x, probs = c(0.5, 0.025, 0.975), digits = 3, scientific = -3, ...)
```

**Arguments**

<code>x</code>	an object inheriting class <code>mcmcspatsurv</code>
<code>probs</code>	vector of quantiles to return
<code>digits</code>	see help file <code>?format</code>
<code>scientific</code>	see help file <code>?format</code>
<code>...</code>	additional arguments, not used here

**Value**

prints summary tables to the console

**See Also**

[quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

`print.textSummary`      *print.textSummary function*

---

**Description**

A function to print summary tables from an MCMC run

**Usage**

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

**Arguments**

<code>x</code>	an object inheriting class <code>textSummary</code>
<code>...</code>	additional arguments, not used here

**Value**

prints a text summary of 'x' to the console

---

priorposterior      *priorposterior function*

---

## Description

A function to produce plots of the prior (which shows as a red line) and posterior (showing as a histogram)

## Usage

```
priorposterior(  
  x,  
  breaks = 30,  
  ylab = "Density",  
  main = "",  
  pause = TRUE,  
  bw = FALSE,  
  ...  
)
```

## Arguments

x	an object inheriting class <code>mcmcspatsurv</code>
breaks	see <code>?hist</code>
ylab	optional y label
main	optional title
pause	logical whether to pause between plots, the default is <code>TRUE</code>
bw	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
...	other arguments passed to the <code>hist</code> function

## Value

plots of the prior (red line) and posterior (histogram).

## See Also

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

`proposalVariance`      *proposalVariance function*

---

**Description**

A function to compute an approximate scaling matrix for the MCMC algorithm. Not intended for general use.

**Usage**

```
proposalVariance(
  X,
  surv,
  betahat,
  omegahat,
  Yhat,
  priors,
  cov.model,
  u,
  control
)
```

**Arguments**

<code>X</code>	the design matrix, containing covariate information
<code>surv</code>	an object of class <code>Surv</code>
<code>betahat</code>	an estimate of beta
<code>omegahat</code>	an estimate of omega
<code>Yhat</code>	an estimate of Y
<code>priors</code>	the priors
<code>cov.model</code>	the spatial covariance model
<code>u</code>	a vector of pairwise distances
<code>control</code>	a list containing various control parameters for the MCMC and post-processing routines

**Value**

an estimate of eta and also an approximate scaling matrix for the MCMC

---

`proposalVariance_gridded`*proposalVariance\_gridded function*

---

**Description**

A function to compute an approximate scaling matrix for the MCMC algorithm. Not intended for general use.

**Usage**

```
proposalVariance_gridded(  
  X,  
  surv,  
  betahat,  
  omegahat,  
  Yhat,  
  priors,  
  cov.model,  
  u,  
  control  
)
```

**Arguments**

<code>X</code>	the design matrix, containing covariate information
<code>surv</code>	an object of class <code>Surv</code>
<code>betahat</code>	an estimate of beta
<code>omegahat</code>	an estimate of omega
<code>Yhat</code>	an estimate of Y
<code>priors</code>	the priors
<code>cov.model</code>	the spatial covariance model
<code>u</code>	a vector of pairwise distances
<code>control</code>	a list containing various control parameters for the MCMC and post-processing routines

**Value**

an estimate of eta and also an approximate scaling matrix for the MCMC

---

```
proposalVariance_polygonal
      proposalVariance_polygonal function
```

---

**Description**

A function to compute an approximate scaling matrix for the MCMC algorithm. Not intended for general use.

**Usage**

```
proposalVariance_polygonal(
  X,
  surv,
  betahat,
  omegahat,
  Yhat,
  priors,
  cov.model,
  u,
  control
)
```

**Arguments**

<code>X</code>	the design matrix, containing covariate information
<code>surv</code>	an object of class <code>Surv</code>
<code>betahat</code>	an estimate of $\beta$
<code>omegahat</code>	an estimate of $\omega$
<code>Yhat</code>	an estimate of $Y$
<code>priors</code>	the priors
<code>cov.model</code>	the spatial covariance model
<code>u</code>	a vector of pairwise distances
<code>control</code>	a list containing various control parameters for the MCMC and post-processing routines

**Value**

an estimate of  $\eta$  and also an approximate scaling matrix for the MCMC

---

proposalVariance\_SPDE *proposalVariance\_SPDE function*

---

### Description

A function to compute an approximate scaling matrix for the MCMC algorithm. Not intended for general use.

### Usage

```
proposalVariance_SPDE(
  X,
  surv,
  betahat,
  omegahat,
  Yhat,
  priors,
  cov.model,
  u,
  control
)
```

### Arguments

X	the design matrix, containing covariate information
surv	an object of class Surv
betahat	an estimate of beta
omegahat	an estimate of omega
Yhat	an estimate of Y
priors	the priors
cov.model	the spatial covariance model
u	a vector of pairwise distances
control	a list containing various control parameters for the MCMC and post-processing routines

### Value

an estimate of eta and also an approximate scaling matrix for the MCMC

PsplineHaz

*PsplineHaz* function**Description**

A function to define a parametric proportional hazards model where the baseline hazard is modelled by a basis spline and where the coefficients of the model follow a partially improper random walk prior. This function returns an object inheriting class 'basehazardspec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'gradcumbasehazard', 'hesscumbasehazard' and 'densityquantile'

**Usage**

```
PsplineHaz(times, knots = quantile(times), degree = 3, MLinits = NULL)
```

**Arguments**

times	vector of survival times (both censored and uncensored)
knots	vector of knots in ascending order, must include minimum and maximum values of 'times'
degree	degree of the spline basis, default is 3
MLinits	optional starting values for the non-spatial maximisation routine using optim. Note that we are working with the log of the parameters. Default is -10 for each parameter.

**Details**

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `nparams`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is `'densityquantile'`. In the case of the Weibull model for the baseline hazard, it can be shown that the `q`-th quantile is:

### Value

an object inheriting class `'basehazardspec'`

### See Also

[exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#)

---

psplineprior

*psplineprior function*

---

### Description

A function for evaluating the log of an independent Gaussian prior for a given set of parameter values.

### Usage

```
psplineprior(beta = NULL, omega = NULL, eta = NULL, priors)
```

### Arguments

<code>beta</code>	parameter <code>beta</code> at which prior is to be evaluated
<code>omega</code>	parameter <code>omega</code> at which prior is to be evaluated
<code>eta</code>	parameter <code>eta</code> at which prior is to be evaluated
<code>priors</code>	an object of class <code>mcmcPriors</code> , see <code>?mcmcPriors</code>

### Value

the log of the prior evaluated at the given parameter values

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

psplineRWprior      *psplineRWprior function*

---

**Description**

A function to define Gaussian priors for omega. This function simply stores a vector of means and standard deviations to be passed to the main MCMC function, `survspat`.

**Usage**

```
psplineRWprior(taumean, tausd, basehaz, order = 2)
```

**Arguments**

taumean	the prior mean, a vector of length 1 or more. 1 implies a common mean.
tausd	the prior standard deviation, a vector of length 1 or more. 1 implies a common standard deviation.
basehaz	an object inheriting class "basehazardspec", specifically, this function was used for such objects created by a call to the function <code>PsplineHaz</code>
order	the order of the random walk, default is 2

**Value**

an object of class "omegapriorGauss"

**See Also**

[survspat](#), [betapriorGauss](#), [omegapriorGauss](#), [etapriorGauss](#), [indepGaussianprior](#), [derivindepGaussianprior](#)

---

QuadApprox                      *QuadApprox function*

---

### Description

A function to compute the second derivative of a function (of several real variables) using a quadratic approximation on a grid of points defined by the list `argRanges`. Also returns the local maximum.

### Usage

```
QuadApprox(fun, npts, argRanges, plot = FALSE, ...)
```

### Arguments

<code>fun</code>	a function
<code>npts</code>	integer number of points in each direction
<code>argRanges</code>	a list of ranges on which to construct the grid for each parameter
<code>plot</code>	whether to plot the quadratic approximation of the posterior (for two-dimensional parameters only)
<code>...</code>	other arguments to be passed to <code>fun</code>

### Value

a 2 by 2 matrix containing the curvature at the maximum and the (x,y) value at which the maximum occurs

---

`quantile.mcmcspatsurv`    *quantile.mcmcspatsurv function*

---

### Description

A function to extract quantiles of the parameters from an mcmc run

### Usage

```
## S3 method for class 'mcmcspatsurv'
quantile(x, probs = c(0.025, 0.5, 0.975), ...)
```

### Arguments

<code>x</code>	an object inheriting class <code>mcmcspatsurv</code>
<code>probs</code>	vector of probabilities
<code>...</code>	other arguments to be passed to the function, not used here

**Value**

quantiles of model parameters

**See Also**

[print.mcmcspsurv](#), [summary.mcmcspsurv](#), [vcov.mcmcspsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

quantile.mlspatsurv    *quantile.mlspatsurv function*

---

**Description**

A function to extract quantiles of the parameters from an mcmc run

**Usage**

```
## S3 method for class 'mlspatsurv'  
quantile(x, probs = c(0.025, 0.5, 0.975), ...)
```

**Arguments**

x	an object inheriting class mcmcspsurv
probs	vector of probabilities
...	other arguments to be passed to the function, not used here

**Value**

quantiles of model parameters

**See Also**

[print.mcmcspsurv](#), [summary.mcmcspsurv](#), [vcov.mcmcspsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

randompars	<i>randompars function</i>
------------	----------------------------

---

**Description**

A function to return the mcmc chains for the spatially correlated frailties

**Usage**

```
randompars(x)
```

**Arguments**

x                    an object of class `mcmcspatsurv`

**Value**

the Y mcmc chains

**See Also**

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [vcov.mcmcspatsurv](#), [frailty-lag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

<code>reconstruct.bs</code>	<i>reconstruct.bs function</i>
-----------------------------	--------------------------------

---

**Description**

Generic function for reconstructing B-spline covariate effects. See `?reconstruct.bs.mcmcspatsurv` and `?reconstruct.bs.coxph`

**Usage**

```
reconstruct.bs(mod, ...)
```

**Arguments**

mod                    an object  
...                    additional arguments

**Value**

method `reconstruct.bs`

---

reconstruct.bs.coxph *reconstruct.bs.coxph function*


---

## Description

When `bs(varname)` has been used in the formula of a coxph model, this function can be used to reconstruct the predicted relative risk of that parameter over time.

## Usage

```
## S3 method for class 'coxph'
reconstruct.bs(
  mod,
  varname,
  fun = NULL,
  probs = c(0.025, 0.975),
  bw = FALSE,
  xlab = NULL,
  ylab = NULL,
  plot = TRUE,
  ...
)
```

## Arguments

<code>mod</code>	model output, created by function <code>survspat</code>
<code>varname</code>	name of the variable modelled by a B-spline
<code>fun</code>	optional function to feed in. Default is to plot relative risk against the covariate of interest. Useful choices include "identity" (but with no quotes), which plots the non-linear effect on the scale of the linear predictor.
<code>probs</code>	upper and lower quantiles for confidence regions to plot> The default is <code>c(0.025,0.975)</code> .
<code>bw</code>	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
<code>xlab</code>	label for x axis, there is a sensible default
<code>ylab</code>	label for y axis, there is a sensible default
<code>plot</code>	logical, whether to plot the effect of <code>varname</code> over time
<code>...</code>	other arguments to be passed to the plotting function.

## Value

median, upper and lower confidence bands for the effect of `varname` over time; the function also produces a plot.

---

```
reconstruct.bs.mcmcspatsurv  
      reconstruct.bs.mcmcspatsurv function
```

---

## Description

When `bs(varname)` has been used in the formula of a model, this function can be used to reconstruct the posterior relative risk of that parameter over time.

## Usage

```
## S3 method for class 'mcmcspatsurv'  
reconstruct.bs(  
  mod,  
  varname,  
  probs = c(0.025, 0.975),  
  bw = FALSE,  
  xlab = NULL,  
  ylab = NULL,  
  plot = TRUE,  
  ...  
)
```

## Arguments

<code>mod</code>	model output, created by function <code>survspat</code>
<code>varname</code>	name of the variable modelled by a B-spline
<code>probs</code>	upper and lower quantiles for confidence regions to plot> The default is <code>c(0.025,0.975)</code> .
<code>bw</code>	Logical. Plot in black/white/greyscale? Default is to produce a colour plot. Useful for producing plots for journals that do not accept colour plots.
<code>xlab</code>	label for x axis, there is a sensible default
<code>ylab</code>	label for y axis, there is a sensible default
<code>plot</code>	logical, whether to plot the effect of <code>varname</code> over time
<code>...</code>	other arguments to be passed to the plotting function.

## Value

median, upper and lower confidence bands for the effect of `varname` over time; the function also produces a plot.

resetLoop                    *reset iterator*

---

**Description**

call this to reset an iterator's state to the initial

**Usage**

```
resetLoop(obj)
```

**Arguments**

obj                    an mcmc iterator

---

residuals.mcmcspatsurv  
*residuals.mcmcspatsurv function*

---

**Description**

A function to compute Cox-Snell / modeified Cox-Snell / Martingale or Deviance residuals

**Usage**

```
## S3 method for class 'mcmcspatsurv'  
residuals(object, type = "Cox-Snell", ...)
```

**Arguments**

object                an object produced by the function survspat  
type                  type of residuals to return. Possible choices are 'Cox-Snell', 'modified-Cox-Snell', 'Martingale' or 'deviance'.  
...                    other arguments (not used here)

**Value**

the residuals

---

rootWeibullHaz	<i>rootWeibullHaz function</i>
----------------	--------------------------------

---

### Description

A function to define a parametric proportional hazards model where the baseline hazard is taken from the Weibull model. This function returns an object inheriting class 'basehazardspec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'gradcumbasehazard', 'hesscumbasehazard' and 'densityquantile'

### Usage

```
rootWeibullHaz(MLinits = NULL)
```

### Arguments

MLinits            initial values for optim, default is NULL

### Details

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is `'densityquantile'`. In the case of the Weibull model for the baseline hazard, it can be shown that the  $q$ -th quantile is:

### Value

an object inheriting class `'basehazardspec'`

### See Also

[tpwHaz](#), [exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#)

---

`setTxtProgressBar2`      *set the progress bar*

---

### Description

update a text progress bar. See `help(txtProgressBar)` for more info.

### Usage

```
setTxtProgressBar2(pb, value, title = NULL, label = NULL)
```

### Arguments

<code>pb</code>	text progress bar object
<code>value</code>	new value
<code>title</code>	ignored
<code>label</code>	text for end of progress bar

---

`setupHazard`      *setupHazard function*

---

### Description

A function to set up the baseline hazard, cumulative hazard and derivative functions for use in evaluating the log posterior. This function is not intended for general use.

### Usage

```
setupHazard(dist, pars, grad = FALSE, hess = FALSE)
```

**Arguments**

dist	an object of class 'basehazardspec'
pars	parameters with which to create the functions necessary to evaluate the log posterior
grad	logical, whether to create gradient functions for the baseline hazard and cumulative hazard
hess	logical, whether to create hessian functions for the baseline hazard and cumulative hazard

**Value**

a list of functions used in evaluating the log posterior

---

setupPrecMatStruct     *setupPrecMatStruct function*

---

**Description**

A function to set up the computational grid and precision matrix structure for SPDE models.

**Usage**

```
setupPrecMatStruct(shape, cellwidth, no)
```

**Arguments**

shape	an object of class SpatialPolygons or SpatialPolygonsDataFrame
cellwidth	a scalar, the width of the grid cells
no	the order of the SPDE approximation: see Lindgren et al 2011 for details

**Value**

the computational grid and a function for constructing the precision matrix

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

showGrid	<i>showGrid function</i>
----------	--------------------------

---

**Description**

A function to show the grid that will be used for a given cellwidth

**Usage**

```
showGrid(dat, cellwidth, ext = 2, boundingbox = NULL)
```

**Arguments**

dat	any spatial data object whose bounding box can be computed using the function <code>bbox</code> .
cellwidth	an initial suggested cellwidth
ext	the extension parameter for the FFT transform, set to 2 by default
boundingbox	optional bounding box over which to construct computational grid, supplied as an object on which the function <code>'bbox'</code> returns the bounding box

**Value**

a plot showing the grid and the data. Ideally the data should only just fit inside the grid.

---

simsurv	<i>simsurv function</i>
---------	-------------------------

---

**Description**

A function to simulate spatial parametric proportional hazards model. The function works by simulating candidate survival times using MCMC in parallel for each individual based on each individual's covariates and the common parameter effects, beta.

**Usage**

```
simsurv(
  X = cbind(age = runif(100, 5, 50), sex = rbinom(100, 1, 0.5), cancer = rbinom(100, 1,
    0.2)),
  beta = c(0.0296, 0.0261, 0.035),
  omega = 1,
  dist = exponentialHaz(),
  coords = matrix(runif(2 * nrow(X)), nrow(X), 2),
  cov.parameters = c(1, 0.1),
  cov.model = ExponentialCovFct(),
  mcmc.control = mcmcpars(nits = 1e+05, burn = 10000, thin = 90),
  savechains = TRUE
)
```

**Arguments**

X	a matrix of covariate information
beta	the parameter effects
omega	vector of parameters for the baseline hazard model
dist	the distribution choice: exp or weibull at present
coords	matrix with 2 columns giving the coordinates at which to simulate data
cov.parameters	a vector: the parameters for the covariance function
cov.model	an object of class covmodel, see ?covmodel
mcmc.control	mcmc control paramters, see ?mcmcpars
savechains	save all chains? runs faster if set to FALSE, but then you'll be unable to conduct convergence/mixing diagnostics

**Value**

in list element 'survtimes', a vector of simulated survival times (the last simulated value from the MCMC chains) in list element 'T' the MCMC chains

**See Also**

[covmodel](#), [survspat](#), [tpowHaz](#), [exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#)

---

spatialpars

*spatialpars function*

---

**Description**

A function to return the mcmc chains for the spatial covariance function parameters

**Usage**

```
spatialpars(x)
```

**Arguments**

x	an object of class mcmcparsurv
---	--------------------------------

**Value**

the eta mcmc chains

**See Also**

[print.mcmcparsurv](#), [quantile.mcmcparsurv](#), [summary.mcmcparsurv](#), [vcov.mcmcparsurv](#), [frailty-lag1](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcparsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

spatsurvVignette      *spatsurvVignette function*

---

**Description**

Display the introductory vignette for the spatsurv package.

**Usage**

```
spatsurvVignette()
```

**Value**

displays the vignette by calling browseURL

---

SPDE      *SPDE function*

---

**Description**

A function to declare and evaluate an SPDE covariance function.

**Usage**

```
SPDE(ord)
```

**Arguments**

ord      the order of the model to be used, currently an integer between 1 and 3. See Lindgren 2011 paper.

**Value**

an covariance function based on the SPDE model

**See Also**

[ExponentialCovFct](#), [covmodel](#)

---

SPDEprec

*SPDEprec function*

---

### Description

A function to used in entering elements into the precision matrix of an SPDE model. Not intended for general use.

### Usage

```
SPDEprec(a, ord)
```

### Arguments

a                    parameter a, see Lindgren et al 2011.  
ord                  the order of the SPDE model, see Lindgren et al 2011.

### Value

a function used for creating the precision matrix

### References

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

---

SpikedExponentialCovFct

*SpikedExponentialCovFct function*

---

### Description

A function to declare and also evaluate a spiked exponential covariance function. This is an exponential covariance function with a nugget.

### Usage

```
SpikedExponentialCovFct()
```

### Value

the spiked exponential covariance function

**See Also**

[ExponentialCovFct](#), [covmodel](#)

---

Summarise

*Summarise function*

---

**Description**

A function to completely summarise the output of an object of class `mcmcspatsurv`.

**Usage**

```
Summarise(
  obj,
  digits = 3,
  scientific = -3,
  inclIntercept = FALSE,
  printmode = "LaTeX",
  displaymode = "console",
  ...
)
```

**Arguments**

<code>obj</code>	an object produced by a call to <code>lgcpPredictSpatialPlusPars</code> , <code>lgcpPredictAggregateSpatialPlusPars</code> , <code>lgcpPredictSpatioTemporalPlusPars</code> or <code>lgcpPredictMultitypeSpatialPlusPars</code>
<code>digits</code>	see the option "digits" in <code>?format</code>
<code>scientific</code>	see the option "scientific" in <code>?format</code>
<code>inclIntercept</code>	logical: whether to summarise the intercept term, default is <code>FALSE</code> .
<code>printmode</code>	the format of the text to return, can be 'LaTeX' (the default) or 'text' for plain text.
<code>displaymode</code>	default is 'console' alternative is 'rstudio'
<code>...</code>	other arguments passed to the function "format"

**Value**

A text summary, that can be pasted into a LaTeX document and later edited.

---

summary.mcmc	<i>summary.mcmc function</i>
--------------	------------------------------

---

**Description**

summary of an mcmc iterator print out values of an iterator and reset it. DONT call this in a loop that uses this iterator - it will reset it. And break.

**Usage**

```
## S3 method for class 'mcmc'
summary(object, ...)
```

**Arguments**

object	an mcmc iterator
...	other args

---

summary.mcmcpatsurv	<i>summary.mcmcpatsurv function</i>
---------------------	-------------------------------------

---

**Description**

A function to return summary tables from an MCMC run

**Usage**

```
## S3 method for class 'mcmcpatsurv'
summary(object, probs = c(0.5, 0.025, 0.975), ...)
```

**Arguments**

object	an object inheriting class mcmcpatsurv
probs	vector of quantiles to return
...	additional arguments

**Value**

summary tables to the console

**See Also**

[print.mcmcpatsurv](#), [quantile.mcmcpatsurv](#), [vcov.mcmcpatsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcpatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

surv3d

*Spatial Survival Plot in 3D***Description**

Do a 3d plot of spatial survival data

**Usage**

```
surv3d(
  spp,
  ss,
  lwd = 2,
  lcol = "black",
  lalpha = 1,
  pstyle = c("point", "text"),
  psize = c(20, 10),
  pcol = c("red", "black"),
  ptext = c("X", ""),
  palpha = 1,
  title = "Spatial Survival",
  basegrid = TRUE,
  baseplane = TRUE
)
```

**Arguments**

spp	A spatial points data frame
ss	A Surv object (with right-censoring)
lwd	Line width for stems
lcol	Line colour for stems
lalpha	Opacity for stems
pstyle	Point style "point" or "text"
psize	Vector of length 2 for uncensored/censored points size
pcol	Vector of length 2 for uncensored/censored points colours
ptext	Vector of length 2 for uncensored/censored text characters
palpha	Opacity for points/text
title	Main title for plot
basegrid	add a grid at t=0
baseplane	add a plane at t=0

**Details**

Uses rgl graphics to make a spinny zoomy plot

**Value**

nothing

**Author(s)**

Barry S Rowlingson

**Examples**

```
## Not run:
require(sp)
require(survival)
d = data.frame(
  x=runif(40)*1.5,
  y = runif(40),
  age=as.integer(20+30*runif(40)),
  sex = sample(c("M", "F"), 40, TRUE)
)
coordinates(d)~x+y
d$surv = Surv(as.integer(5+20*runif(40)), runif(40)>.9)
clear3d(); surv3d(d, d$surv, baseplane=TRUE, basegrid=TRUE)
clear3d(); surv3d(d, d$surv, baseplane=TRUE, basegrid=TRUE, pstyle="t", lalpha=0.5, lwd=3, palpha=1)

## End(Not run)
```

---

survival\_PP

*survival\_PP function*

---

**Description**

A function to compute an individual's survival function

**Usage**

```
survival_PP(inputs)
```

**Arguments**

inputs            inputs for the function including the model matrix, frailties, fixed effects and the parameters of the baseline hazard derived from this model

**Value**

the survival function for the individual

---

survspat                      *survspat function*

---

### Description

A function to run a Bayesian analysis on censored spatial survival data assuming a proportional hazards model using an adaptive Metropolis-adjusted Langevin algorithm.

### Usage

```
survspat(
  formula,
  data,
  dist,
  cov.model,
  mcmc.control,
  priors,
  shape = NULL,
  ids = list(shpid = NULL, dataid = NULL),
  control = inference.control(grided = FALSE),
  boundingbox = NULL
)
```

### Arguments

formula	the model formula in a format compatible with the function <code>flexsurvreg</code> from the <code>flexsurv</code> package
data	a <code>SpatialPointsDataFrame</code> object containing the survival data as one of the columns OR for polygonal data a <code>data.frame</code> , in which case, the argument <code>shape</code> must also be supplied
dist	choice of distribution function for baseline hazard. Current options are: <code>exponentialHaz</code> , <code>weibullHaz</code> , <code>gompertzHaz</code> , <code>makehamHaz</code> , <code>tpowHaz</code>
cov.model	an object of class <code>covmodel</code> , see <code>?covmodel</code> <code>?ExponentialCovFct</code> or <code>?SpikedExponentialCovFct</code>
mcmc.control	mcmc control parameters, see <code>?mcmcpars</code>
priors	an object of class <code>Priors</code> , see <code>?mcmcPriors</code>
shape	when data is a <code>data.frame</code> , this can be a <code>SpatialPolygonsDataFrame</code> , or a <code>SpatialPointsDataFrame</code> , used to model spatial variation at the small region level. The regions are the polygons, or they represent the (possibly weighted) centroids of the polygons.
ids	named list entry <code>shpid</code> character string giving name of variable in <code>shape</code> to be matched to variable <code>dataid</code> in <code>data</code> . <code>dataid</code> is the second entry of the named list.
control	additional control parameters, see <code>?inference.control</code>
boundingbox	optional bounding box over which to construct computational grid, supplied as an object on which the function <code>'bbox'</code> returns the bounding box

**Value**

an object inheriting class 'mcmcsurv' for which there exist methods for printing, summarising and making inference from.

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. Journal of Statistical Software, 77(4), 1-32, doi:10.18637/jss.v077.i04.

**See Also**

[tpowHaz](#), [exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#), [covmodel](#), [ExponentialCovFct](#), [SpikedExponentialCovFct](#), [mcmcPars](#), [mcmcPriors](#), [inference.control](#)

---

survspatNS

*survspatNS function*

---

**Description**

A function to perform maximum likelihood inference for non-spatial survival data.

**Usage**

```
survspatNS(formula, data, dist, control = inference.control())
```

**Arguments**

formula	the model formula in a format compatible with the function flexsurvreg from the flexsurv package
data	a SpatialPointsDataFrame object containing the survival data as one of the columns
dist	choice of distribution function for baseline hazard. Current options are: exponentialHaz, weibullHaz, gompertzHaz, makehamHaz, tpowHaz
control	additional control parameters, see ?inference.control

**Value**

an object inheriting class 'mcmcsurv' for which there exist methods for printing, summarising and making inference from.

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. Journal of Statistical Software, 77(4), 1-32, doi:10.18637/jss.v077.i04.

**See Also**

[tpowHaz](#), [exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#), [covmodel](#), [ExponentialCovFct](#), [SpikedExponentialCovFct](#), [mcmcPars](#), [mcmcPriors](#), [inference.control](#)

---

textSummary	<i>textSummary function</i>
-------------	-----------------------------

---

### Description

A function to print a text description of the inferred parameters beta and eta from a call to the function `lgcpPredictSpatialPlusPars`, `lgcpPredictAggregateSpatialPlusPars`, `lgcpPredictSpatioTemporalPlusPars` or `lgcpPredictMultitypeSpatialPlusPars`

### Usage

```
textSummary(
  obj,
  digits = 3,
  scientific = -3,
  inclIntercept = FALSE,
  printmode = "LaTeX",
  ...
)
```

### Arguments

<code>obj</code>	an object produced by a call to <code>lgcpPredictSpatialPlusPars</code> , <code>lgcpPredictAggregateSpatialPlusPars</code> , <code>lgcpPredictSpatioTemporalPlusPars</code> or <code>lgcpPredictMultitypeSpatialPlusPars</code>
<code>digits</code>	see the option "digits" in <code>?format</code>
<code>scientific</code>	see the option "scientific" in <code>?format</code>
<code>inclIntercept</code>	logical: whether to summarise the intercept term, default is FALSE.
<code>printmode</code>	the format of the text to return, can be 'LaTeX' (the default) or 'text' for plain text.
<code>...</code>	other arguments passed to the function "format"

### Value

A text summary, that can be pasted into a LaTeX document and later edited.

---

timevaryingPL	<i>timevaryingPL function</i>
---------------	-------------------------------

---

**Description**

A function to

**Usage**

```
timevaryingPL(  
  formula,  
  t0,  
  t,  
  delta,  
  dist,  
  data,  
  ties = "Efron",  
  optimcontrol = NULL  
)
```

**Arguments**

formula	a formula of the form 'S ~ coef1 + coef2' etc the object S will be created
t0	X
t	X
delta	censoring indicator a vector of 1 for an event and 0 for censoring
dist	X
data	X
ties	X default is Efron
optimcontrol	X

**Value**

...

---

 tpowHaz
 

---



---

 tpowHaz function
 

---

### Description

A function to define a parametric proportional hazards model where the baseline hazard is taken from the 'powers of t' model. This function returns an object inheriting class 'basehazardspec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'gradcumbasehazard', 'hesscumbasehazard' and 'densityquantile'

### Usage

```
tpowHaz(powers)
```

### Arguments

**powers** a vector of powers of t. These are powers are treated as fixed in estimation routines and it is assumed that the log cumulative baseline hazard is a linear combination of these powers of t

### Details

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is `'densityquantile'`. In the case of the Weibull model for the baseline hazard, it can be shown that the `q`-th quantile is:

### Value

an object inheriting class `'basehazardspec'`

### See Also

[exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#), [weibullHaz](#)

---

<code>transformweibull</code>	<i>transformweibull function</i>
-------------------------------	----------------------------------

---

### Description

A function to back-transform estimates of the parameters of the weibull baseline hazard function, so they are commensurate with R's inbuilt density functions. Transforms from (shape, scale) to (alpha, lambda)

### Usage

```
transformweibull(x)
```

### Arguments

`x` a vector of paramters

### Value

the transformed parameters. For the weibull model, this is the back-transform from `'alpha'` and `'lambda'` to `'shape'` `'scale'` (see `?dweibull`).

---

TwoWayHazAdditive      *TwoWayHazAdditive function*

---

**Description**

A function to

**Usage**

```
TwoWayHazAdditive(bhlist, bhtime, bhfix, MLimits = NULL)
```

**Arguments**

bhlist	X
bhtime	X
bhfix	X
MLimits	X

**Value**

...

---

txtProgressBar2      *A text progress bar with label*

---

**Description**

This is the base txtProgressBar but with a little modification to implement the label parameter for style=3. For full info see txtProgressBar

**Usage**

```
txtProgressBar2(
  min = 0,
  max = 1,
  initial = 0,
  char = "=",
  width = NA,
  title = "",
  label = "",
  style = 1
)
```

**Arguments**

min	min value for bar
max	max value for bar
initial	initial value for bar
char	the character (or character string) to form the progress bar.
width	progress bar width
title	ignored
label	text to put at the end of the bar
style	bar style

---

vcov.mcmcspatsurv      *vcov.mcmcspatsurv function*

---

**Description**

A function to return the variance covariance matrix of the parameters beta, omega and eta

**Usage**

```
## S3 method for class 'mcmcspatsurv'
vcov(object, ...)
```

**Arguments**

object	an object inheriting class mcmcspatsurv
...	other arguments, not used here

**Value**

the variance covariance matrix of the parameters beta, omega and eta

**See Also**

[print.mcmcspatsurv](#), [quantile.mcmcspatsurv](#), [summary.mcmcspatsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcspatsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

vcov.mlspatsurv	<i>vcov.mlspatsurv function</i>
-----------------	---------------------------------

---

**Description**

A function to return the variance covariance matrix of the parameters beta, omega and eta

**Usage**

```
## S3 method for class 'mlspatsurv'
vcov(object, ...)
```

**Arguments**

object	an object inheriting class mcmcsurv
...	other arguments, not used here

**Value**

the variance covariance matrix of the parameters beta, omega and eta

**See Also**

[print.mcmcsurv](#), [quantile.mcmcsurv](#), [summary.mcmcsurv](#), [frailtylag1](#), [spatialpars](#), [hazardpars](#), [fixedpars](#), [randompars](#), [baselinehazard](#), [predict.mcmcsurv](#), [priorposterior](#), [posteriorcov](#), [MCE](#), [hazardexceedance](#)

---

weibullHaz	<i>weibullHaz function</i>
------------	----------------------------

---

**Description**

A function to define a parametric proportional hazards model where the baseline hazard is taken from the Weibull model. This function returns an object inheriting class 'basehazardspec', list of functions 'distinfo', 'basehazard', 'gradbasehazard', 'hessbasehazard', 'cumbasehazard', 'gradcumbasehazard', 'hesscumbasehazard' and 'densityquantile'

**Usage**

```
weibullHaz(MLinits = NULL)
```

**Arguments**

MLinits	initial values for optim, default is NULL
---------	---

## Details

The `distinfo` function is used to provide basic distribution specific information to other `spatsurv` functions. The user is required to provide the following information in the returned list: `npars`, the number of parameters in this distribution; `parnames`, the names of the parameters; `trans`, the transformation scale on which the priors will be provided; `itrans`, the inverse transformation function that will be applied to the parameters before the hazard, and other functions are evaluated; `jacobian`, the derivative of the inverse transformation function with respect to each of the parameters; and `hessian`, the second derivatives of the inverse transformation function with respect to each of the parameters – note that currently the package `spatsurv` only allows the use of functions where the parameters are transformed independently.

The `basehazard` function is used to evaluate the baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradbasehazard` function is used to evaluate the gradient of the baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hessbasehazard` function is used to evaluate the Hessian of the baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `cumbasehazard` function is used to evaluate the cumulative baseline hazard function for the distribution of interest. It returns a function that accepts as input a vector of times, `t` and returns a vector.

The `gradcumbasehazard` function is used to evaluate the gradient of the cumulative baseline hazard function with respect to the parameters, this typically returns a vector. It returns a function that accepts as input a vector of times, `t`, and returns a matrix.

The `hesscumbasehazard` function is used to evaluate the Hessian of the cumulative baseline hazard function. It returns a function that accepts as input a vector of times, `t` and returns a list of hessian matrices corresponding to each `t`.

The `densityquantile` function is used to return quantiles of the density function. This is NOT REQUIRED for running the MCMC, merely for us in post-processing with the `predict` function where `type` is `'densityquantile'`. In the case of the Weibull model for the baseline hazard, it can be shown that the `q`-th quantile is:

## Value

an object inheriting class `'basehazardspec'`

## See Also

[tpowHaz](#), [exponentialHaz](#), [gompertzHaz](#), [makehamHaz](#)

---

YFromGamma

*YfromGamma function*


---

**Description**

A function to change Gammas (white noise) into Ys (spatially correlated noise). Used in the MALA algorithm.

**Usage**

```
YFromGamma(Gamma, invrootQeigs, mu)
```

**Arguments**

Gamma	Gamma matrix
invrootQeigs	inverse square root of the eigenvectors of the precision matrix
mu	parameter of the latent Gaussian field

**Value**

Y

---

YFromGamma\_SPDE

*YFromGamma\_SPDE function*


---

**Description**

A function to go from Gamma to Y

**Usage**

```
YFromGamma_SPDE(gamma, U, mu)
```

**Arguments**

gamma	Gamma
U	upper Cholesky matrix
mu	the mean

**Value**

the value of Y for the given Gamma

**References**

1. Benjamin M. Taylor and Barry S. Rowlingson (2017). spatsurv: An R Package for Bayesian Inference with Spatial Survival Models. *Journal of Statistical Software*, 77(4), 1-32, doi:10.18637/jss.v077.i04.
2. Finn Lindgren, Havard Rue, Johan Lindstrom. An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B* 73(4)

# Index

- \* **datasets**
  - fs, [37](#)
  - fstimes, [37](#)
- \* **package**
  - spatsurv-package, [5](#)
  - .onAttach, [10](#)
- allocate, [11](#)
- alpha, [12](#)
- B, [12](#)
- basehazard, [13](#)
- basehazard.basehazardspec, [13](#), [13](#)
- baseHazST, [14](#)
- baselinehazard, [15](#), [35](#), [37](#), [53](#), [70](#), [83–87](#),  
[96](#), [97](#), [105](#), [109](#), [119](#), [120](#)
- baselinehazard\_multiWay, [16](#)
- betapriorGauss, [17](#), [17](#), [28](#), [29](#), [31](#), [58](#), [72](#),  
[77](#), [94](#)
- blockDiag, [17](#)
- boxplotRisk, [18](#)
- Bspline.construct, [18](#)
- BsplineHaz, [19](#)
- checkSurvivalData, [20](#)
- circulant, [21](#)
- circulant.matrix, [21](#)
- circulant.numeric, [22](#)
- circulantij, [22](#)
- covmodel, [23](#), [33](#), [57](#), [105](#), [106](#), [108](#), [113](#)
- CSplot, [23](#)
- cumbasehazard, [24](#)
- cumbasehazard.basehazardspec, [24](#), [24](#)
- cumulativeBspline.construct, [25](#)
- density\_PP, [27](#)
- densityquantile, [25](#)
- densityquantile.basehazardspec, [26](#), [26](#)
- densityquantile\_PP, [26](#)
- derivindepGaussianprior, [17](#), [27](#), [28](#), [29](#),  
[31](#), [58](#), [72](#), [77](#), [94](#)
- derivindepGaussianpriorST, [28](#)
- derivpsplineprior, [28](#)
- distinfo, [29](#)
- distinfo.basehazardspec, [29](#), [30](#)
- estimateY, [30](#)
- Et\_PP, [31](#)
- etapriorGauss, [17](#), [28](#), [29](#), [31](#), [31](#), [58](#), [72](#), [77](#),  
[94](#)
- EvalCov, [32](#)
- ExponentialCovFct, [32](#), [106](#), [108](#), [113](#)
- exponentialHaz, [13](#), [14](#), [20](#), [24–26](#), [29](#), [30](#),  
[33](#), [46–49](#), [54–56](#), [69](#), [93](#), [102](#), [105](#),  
[113](#), [117](#), [121](#)
- FFTgrid, [34](#)
- fixedpars, [15](#), [35](#), [37](#), [53](#), [70](#), [83–87](#), [96](#), [97](#),  
[105](#), [109](#), [119](#), [120](#)
- fixmatrix, [35](#)
- fixParHaz, [36](#)
- frailtylag1, [15](#), [35](#), [36](#), [53](#), [70](#), [83–87](#), [96](#),  
[97](#), [105](#), [109](#), [119](#), [120](#)
- fs, [37](#)
- fstimes, [37](#)
- gamma2risk, [38](#)
- GammafromY, [38](#)
- GammaFromY\_SPDE, [39](#)
- gencens, [40](#)
- getbb, [40](#)
- getBbasis, [41](#)
- getcov, [41](#)
- getgrd, [42](#)
- getGrid, [42](#)
- getleneta, [43](#)
- getOptCellwidth, [43](#)
- getparranges, [44](#)
- getsurvdata, [45](#)
- gompertzHaz, [13](#), [14](#), [20](#), [24–26](#), [29](#), [30](#), [34](#),  
[45](#), [47–49](#), [54–56](#), [69](#), [93](#), [102](#), [105](#),  
[113](#), [117](#), [121](#)

- gradbasehazard, 46
- gradbasehazard.basehazardspec, 47, 47
- gradcumbasehazard, 48
- gradcumbasehazard.basehazardspec, 48, 48
- grid2spdf, 49
- grid2spix, 49
- grid2spts, 50
- gridY, 50
- gridY\_polygonal, 51
- guess\_t, 51
  
- hasNext, 52
- hasNext.iter, 52
- hazard\_PP, 54
- hazardexceedance, 15, 35, 37, 52, 53, 70, 83–87, 96, 97, 105, 109, 119, 120
- hazardpars, 15, 35, 37, 53, 53, 70, 83–87, 96, 97, 105, 109, 119, 120
- hessbasehazard, 54
- hessbasehazard.basehazardspec, 54, 55
- hesscumbasehazard, 55
- hesscumbasehazard.basehazardspec, 56, 56
  
- imputationModel, 56
- Independent, 57
- indepGaussianprior, 17, 28, 29, 31, 57, 58, 72, 77, 94
- indepGaussianpriorST, 58
- inference.control, 59, 113
- insert, 60
- invtransformweibull, 61
- is.burnin, 61
- is.retain, 62
- iteration, 62
  
- logPosterior, 63
- logPosterior\_gridded, 64
- logPosterior\_polygonal, 65
- logPosterior\_SPDE, 66
- loop.mcmc, 67
  
- makehamHaz, 13, 14, 20, 24–26, 29, 30, 34, 46–49, 54–56, 68, 93, 102, 105, 113, 117, 121
- maxlikparamPHsurv, 69
- MCE, 15, 35, 37, 53, 70, 83–87, 96, 97, 105, 109, 119, 120
  
- mcmcLoop, 70
- mcmcpars, 71, 113
- mcmcPriors, 71, 113
- mcmcProgressNone, 72
- mcmcProgressPrint, 73
- mcmcProgressTextBar, 73
- midpts, 74
- multiWayHaz, 74
  
- neighLocs, 75
- neighOrder, 75
- nextStep, 76
- NonSpatialLogLikelihood\_or\_gradient, 76
  
- omegapriorGauss, 17, 28, 29, 31, 58, 72, 77, 77, 94
- omegapriorGaussST, 78
- optifix, 78
  
- plot.FFTgrid, 79
- plotsurv, 80
- polyadd, 81
- polymult, 82
- posteriorcov, 15, 35, 37, 53, 70, 82, 84–87, 96, 97, 105, 109, 119, 120
- predict.mcmcspsurv, 15, 35, 37, 53, 70, 83, 83, 85–87, 96, 97, 105, 109, 119, 120
- print.mcmc, 84
- print.mcmcspsurv, 15, 35, 37, 53, 70, 83, 84, 85, 87, 96, 97, 105, 109, 119, 120
- print.mlspatsurv, 85
- print.textSummary, 86
- priorposterior, 15, 35, 37, 53, 70, 83–86, 87, 96, 97, 105, 109, 119, 120
- proposalVariance, 88
- proposalVariance\_gridded, 89
- proposalVariance\_polygonal, 90
- proposalVariance\_SPDE, 91
- PsplineHaz, 92
- psplineprior, 93
- psplineRWPrior, 94
  
- QuadApprox, 95
- quantile.mcmcspsurv, 15, 35, 37, 53, 70, 83–87, 95, 97, 105, 109, 119, 120
- quantile.mlspatsurv, 96

randompars, [15](#), [35](#), [37](#), [53](#), [70](#), [83–87](#), [96](#), [97](#),  
[105](#), [109](#), [119](#), [120](#)  
reconstruct.bs, [97](#)  
reconstruct.bs.coxph, [98](#)  
reconstruct.bs.mcmcspatsurv, [99](#)  
resetLoop, [100](#)  
residuals.mcmcspatsurv, [100](#)  
rootWeibullHaz, [101](#)

setTxtProgressBar2, [102](#)  
setupHazard, [102](#)  
setupPrecMatStruct, [103](#)  
showGrid, [104](#)  
simsurv, [104](#)  
spatialpars, [15](#), [35](#), [37](#), [53](#), [70](#), [83–87](#), [96](#),  
[97](#), [105](#), [109](#), [119](#), [120](#)  
spatsurv (spatsurv-package), [5](#)  
spatsurv-package, [5](#)  
spatsurvVignette, [106](#)  
SPDE, [106](#)  
SPDEprec, [107](#)  
SpikedExponentialCovFct, [33](#), [57](#), [107](#)  
Summarise, [108](#)  
summary.mcmc, [109](#)  
summary.mcmcspatsurv, [15](#), [35](#), [37](#), [53](#), [70](#),  
[83–87](#), [96](#), [97](#), [105](#), [109](#), [119](#), [120](#)  
surv3d, [110](#)  
survival\_PP, [111](#)  
survspat, [17](#), [28](#), [29](#), [31](#), [58](#), [60](#), [72](#), [77](#), [94](#),  
[105](#), [112](#)  
survspatNS, [113](#)

textSummary, [114](#)  
timevaryingPL, [115](#)  
tpowHaz, [13](#), [14](#), [24–26](#), [29](#), [30](#), [34](#), [46–49](#),  
[54–56](#), [69](#), [102](#), [105](#), [113](#), [116](#), [121](#)  
transformweibull, [117](#)  
TwoWayHazAdditive, [118](#)  
txtProgressBar2, [118](#)

vcov.mcmcspatsurv, [15](#), [35](#), [37](#), [53](#), [70](#),  
[83–87](#), [96](#), [97](#), [105](#), [109](#), [119](#)  
vcov.mlspatsurv, [120](#)

weibullHaz, [13](#), [14](#), [20](#), [24–26](#), [29](#), [30](#), [34](#),  
[46–49](#), [54–56](#), [69](#), [93](#), [105](#), [113](#), [117](#),  
[120](#)

YfromGamma, [122](#)  
YFromGamma\_SPDE, [122](#)