Package 'spm'

September 25, 2015

Type Package

Title Stochastic Process Modeling (SPM)

Version 1.0

Date 2015-09-24

Author I. Y. Zhbannikov, I. V. Akushevich, K. G. Arbeev, A. I. Yashin

Maintainer Ilya Y. Zhbannikov <i.zhbannikov@mail.ru>

Description Stochastic Process Modeling

License GPL

Imports Rcpp (>= 0.11.1), RcppArmadillo (>= 0.4.200.0)

LinkingTo Rcpp, RcppArmadillo

Depends deSolve,mice,sas7bdat,RcppArmadillo

R topics documented:

`spm-package` *Stochastic Process Modeling*

Description

Stochastic Process Modeling (SPM) perform estimation of parameters of some stochastic process of interest. This package is mainly for estimating parameters from clinical traits.

Details

Package:	spm
Type:	Package
Version:	1.0
Date:	2015-09-05
License:	GPL

Author(s)

Ilya Y. Zhbannikov

Maintainer: Ilya Y. Zhbannikov <i.zhbannikov@mail.ru>

References

Anatoli I. Yashin, Konstantin G. Arbeev, Igor Akushevich, Aliaksandr Kulminski, Lucy Akushevich, Svetlana V. Ukraintseva, Stochastic model for analysis of longitudinal data on aging and mortality, Mathematical Biosciences, Volume 208, Issue 2, August 2007, Pages 538-551, ISSN 0025-5564, http://dx.doi.org/10.1016/j.mbs.2006.11.006.

Anatoli I. Yashin , Konstantin G. Arbeev, Aliaksandr Kulminski, Igor Akushevich, Lucy Akushevich, Svetlana V. Ukraintseva, Health decline, aging and mortality: how are they related? Biogerontology June 2007, Volume 8, Issue 3, pp 291-302

Examples

```
library(spm)
# Reading longitude data:
longdat <- read.csv(system.file("data","longdat.csv",package="spm"))
# Prepare data for optimization:
vitstat <- read.csv(system.file("data","vitstat.csv",package="spm"))
# Remove unneeded NAs:
longdat.nonan <- longdat[which(is.na(longdat$Age) == F),]
vitstat.nonan <- vitstat[which(is.na(vitstat$BirthCohort) == F),]
data=prepare_data(longdat=longdat.nonan, vitstat=vitstat.nonan,interval=1, col.status="IsDead", col.id="ID"
# Parameters estimation:
pars=spm(data,k = 1)
pars
```

prepare_data *Output values include: 1). Database (data table), simulated for (slow) continuous optimization with arbitrary intervals between observations. 2). Database (data table), simulated for (quick) discrete optimization with fixed intervals between each observation.*

Description

Output values include: 1). Database (data table), simulated for (slow) continuous optimization with arbitrary intervals between observations. 2). Database (data table), simulated for (quick) discrete optimization with fixed intervals between each observation.

Usage

```
prepare_data(longdat, vitstat, interval = 1, col.status = "IsDead",
  col.id = "ID", col.age = "Age", col.age.event = "LSmort",
  covariates = c("DBP", "BMI", "DBP1", "DBP2", "Weight", "Height"),
  verbose = T)
```

Arguments

longdat	A table with longitude records.
vitstat	A table with vital statistics (mortality).
interval	A number of breaks between observations for discrete simulation. Default = 1 (no breaks).
col.status	A name of column containing status variable (0/1 which indicate alive/dead).
col.id	A name of column containing patient ID. This ID should be the same in both longdat and vitstat tables.
col.age	A name of age column.
col.age.event	A name of event column.
covariates	A list of covariates.
verbose	A verbosing output indicator. Default=TRUE.

Value

A list of two elements: first element contains a data table for continuous case, with arbitrary intervals between observations and second element contains a data table for a discrete case (fixed intervals between observations).

Examples

```
library(spm)
#Reading longitude data:
longdat <- read.csv(system.file("data","longdat.csv",package="spm"))
# Prepare data for optimization:
vitstat <- read.csv(system.file("data","vitstat.csv",package="spm"))
# Remove unneeded NAs:
longdat.nonan <- longdat[which(is.na(longdat$Age) == F),]
vitstat.nonan <- vitstat[which(is.na(vitstat$BirthCohort) == F),]
data=prepare_data(longdat=longdat.nonan, vitstat=vitstat.nonan,interval=1, col.status="IsDead", col.id="ID"
```

```
# Parameters estimation:
pars=spm(data,k = 1)
pars
```

simdata *Function that simulates data using u, R, epsilon, mu0, b, Q, theta*

Description

Function that simulates data using u, R, epsilon, mu0, b, Q, theta

Usage

```
simdata(N = 100, u = 8, R = 0.95, epsilon = 5, mu0 = 2e-05, b = 10,
  Q = 2e-08, theta = 0.08, tstart = 30, ystart = 80, dt = 1,
  tmax = 105, k = 1)
```

Arguments

N	Number of individuals
u	A drift vector with length of k.
R	A k by k regression matrix.
epsilon	A time-dependent normally distributed random vector (size=k).
mu0	mortality at start period of time.
b	A diffusion coefficient, k by k matrix.
Q	A matrix k by k, which is a non-negative-definite symmetric matrix.
theta	A displacement coefficient of the Gompertz function.
tstart	A number that defines starting time (30 by default).
ystart	A vector with length equal to number of dimensions used, defines starting values of covariates.
dt	A time step (1 by default).
k	Number of dimensions (k = 1 by default).
tend	A number, defines final time (105 by default).

Value

A table with simulated data.

Examples

```
library(spm)
data <- simdata(N=1000, ystart=c(75, 94), k=1)
head(data)
```

simdata_cont *One-dimensional simulation function for continuous trait.*

Description

One-dimensional simulation function for continuous trait.

Usage

```
simdata_cont(N = 10, aH = -0.05, f1H = 80, QH = 2e-07, fH = 80,
  bH = 5, mu0H = 2e-05, thetaH = 0.08, step = 0.05, tstart = 30,
  tend = 105, ystart = 80, sd0 = 4)
```

Arguments

N	Number of individuals.
tstart	A number that defines starting time (30 by default).
tend	A number, defines final time (105 by default).
ystart	A vector with length equal to number of dimensions used, defines starting values of covariates.
a	A k by k matrix, which characterize the rate of the adaptive response.
f1	A particular state, which if a deviation from the normal (or optimal). This is a vector with length of k.
Q	A matrix k by k, which is a non-negative-definite symmetric matrix.
f	A vector-function (with length k) of the normal (or optimal) state.
b	A diffusion coefficient, k by k matrix.
mu0	mortality at start period of time.
theta	A displacement coefficient of the Gompertz function.

Value

A table with simulated data.

Examples

```
library(spm)
dat <- simdata_cont(N=2500)
dat
```

simdata_cont_MD *Multi-dimensional simulation function for continuous trait.*

Description

Multi-dimensional simulation function for continuous trait.

Usage

```
simdata_cont_MD(N = 10, aH = -0.05, f1H = 80, QH = 2e-07, fH = 80,
  bH = 5, mu0H = 2e-05, thetaH = 0.08, step = 0.05, tstart = 30,
  tend = 105, ystart = 80, sd0 = 4, k = 1)
```

Arguments

N	Number of individuals.
tstart	A number that defines starting time (30 by default).
tend	A number, defines final time (105 by default).
ystart	A vector with length equal to number of dimensions used, defines starting values of covariates.
k	number of dimensions (k = 1 by default).
a	A k by k matrix, which characterize the rate of the adaptive response.
f1	A particular state, which if a deviation from the normal (or optimal). This is a vector with length of k.
Q	A matrix k by k, which is a non-negative-definite symmetric matrix.
f	A vector-function (with length k) of the normal (or optimal) state.
b	A diffusion coefficient, k by k matrix.
mu0	mortality at start period of time.
theta	A displacement coefficient of the Gompertz function.

Value

A table with simulated data.

Examples

```
library(spm)
dat <- simdata_cont(N=2500)
dat
```

simdata_time_dep — *Simulation function for continuous trait with time-dependant coefficients.*

Description

Simulation function for continuous trait with time-dependant coefficients.

Usage

```
simdata_time_dep(N = 10, formulas = list(at = "-0.05", f1t = "80", Qt =
  "2e-7*exp(0.08*t)", ft = "80", bt = "5", mu0t = "2e-5*exp(0.08*t)"),
  step = 0.05, tstart = 30, tend = 105, ystart = 80, sd0 = 4, k = 1)
```

Arguments

N	Number of individuals.
formulas	: a list of formulas that define age (time) - dependency. Default: list(at="a", f1t="f1", Qt="Q*exp(theta*t)", ft="f", bt="b", mu0t="mu0*exp(theta*t)")
tstart	A number that defines starting time (30 by default).
tend	A number, defines final time (105 by default).
ystart	A starting value of covariates.

Value

A table with simulated data.

Examples

```
library(spm)
dat <- simdata_time_dep(N=2500)
dat
```

sim_discrete — *Multi-dimension simulation function It uses a, f1, Q, f, b, mu0 and theta as input parameters.*

Description

Multi-dimension simulation function It uses a, f1, Q, f, b, mu0 and theta as input parameters.

Usage

```
sim_discrete(N = 100, a = -0.05, f1 = 80, Q = 2e-08, f = 80, b = 5,
  mu0 = 1e-05, theta = 0.08, ystart = c(80), tstart = 30, tend = 105,
  dt = 1, k = 1)
```

Arguments

N	Number of individuals
a	A k by k matrix, which characterize the rate of the adaptive response.
f1	A particular state, which if a deviation from the normal (or optimal). This is a vector with length of k.
Q	A matrix k by k, which is a non-negative-definite symmetric matrix.
f	A vector-function (with length k) of the normal (or optimal) state.
b	A diffusion coefficient, k by k matrix.
mu0	mortality at start period of time.
theta	A displacement coefficient of the Gompertz function.
ystart	A vector with length equal to number of dimensions used, defines starting values of covariates.
tstart	A number that defines starting time (30 by default).
tend	A number, defines final time (105 by default).
dt	A time step (1 by default).
k	number of dimensions (k = 1 by default).

Value

A table with simulated data.

Examples

```
library(spm)
data <- sim(N=1000, ystart=c(75, 94), k=1)
head(data)
```

---

spm *Stochastic Process Modelling (SPM) A main function that estimates parameters a, f1, Q, f, b, mu0, theta from given dataset.*

Description

Stochastic Process Modelling (SPM) A main function that estimates parameters a, f1, Q, f, b, mu0, theta from given dataset.

Usage

```
spm(dat, k = 2, verbose = F, tol = NULL)
```

Arguments

dat	A dataset.
k	Number of dimensions.
verbose	A verbosing output indicator.
tol	A tolerance threshold for matrix inversion.

Value

A list of (1) Estimated starting point (from quick discrete optimization) and (2) Estimated coefficients.

Examples

```
library(spm)
# Reading longitude data:
longdat <- read.csv(system.file("data","longdat.csv",package="spm"))
# Prepare data for optimization:
vitstat <- read.csv(system.file("data","vitstat.csv",package="spm"))
# Remove unneeded NAs:
longdat.nonan <- longdat[which(is.na(longdat$Age) == F),]
vitstat.nonan <- vitstat[which(is.na(vitstat$BirthCohort) == F),]
data=prepare_data(longdat=longdat.nonan, vitstat=vitstat.nonan,interval=1, col.status="IsDead", col.id="ID"
# Parameters estimation:
pars=spm(data,k = 1)
pars
```

`spm_integral_MD` *Continuous multi-dimensional optimization It is much slower that discrete but more precise and can handle time intervals with different lengths.*

Description

Continuous multi-dimensional optimization It is much slower that discrete but more precise and can handle time intervals with different lengths.

Usage

```
spm_integral_MD(dat, parameters, k, verbose = F)
```

Arguments

`dat`	A data table.
`parameters`	A starting pont (a vector).
`k`	A number of dimensions.
`verbose`	An indicator of verbosing output.
`tol`	A tolerance threshold for matrix inversion.

Value

A list of two elements: (1) parameters a, f1, Q, f, b, mu0, theta; (2) An output from "optim" function used for maximum likelihood estimation.

Examples

```
#'library(spm)
# Reading longitude data:
longdat <- read.csv(system.file("data","longdat.csv",package="spm"))
# Prepare data for optimization:
vitstat <- read.csv(system.file("data","vitstat.csv",package="spm"))
# Remove unneeded NAs:
longdat.nonan <- longdat[which(is.na(longdat$Age) == F),]
vitstat.nonan <- vitstat[which(is.na(vitstat$BirthCohort) == F),]
dat=prepare_data(longdat=longdat.nonan, vitstat=vitstat.nonan,interval=1, col.status="IsDead", col.id="ID",
# Parameters estimation:
dat<-[,1:6]
pars=spm_integral_MD(dat, parameters=c(-0.05, 80, 2e-8, 80, 5, 2e-5, 0.08), k = 1)
pars
```

---

spm_quick_MD — *Discrete multi-dimensional optimization It is way much faster that continuous (but less precise) and used mainly in estimation of starting point.*

Description

Discrete multi-dimensional optimization It is way much faster that continuous (but less precise) and used mainly in estimation of starting point.

Usage

```
spm_quick_MD(dat, k = 2, theta_range = seq(0.001, 0.09, by = 0.001),
  tol = NULL)
```

Arguments

`dat`	A data table.
`k`	A number of dimensions.
`theta_range`	A range of theta parameter (axe displacement of Gompertz function).
`tol`	A tolerance threshold for matrix inversion.

Value

A list of two elements: (1) parameters u, R, b, epsilon, Q, mu0, theta and (2) parameters a, f1, Q, f, b, mu0, theta. Note: b and mu0 from first list are different from b and mu0 from the second list.

Examples

```
#'library(spm)
# Reading longitude data:
longdat <- read.csv(system.file("data","longdat.csv",package="spm"))
# Prepare data for optimization:
vitstat <- read.csv(system.file("data","vitstat.csv",package="spm"))
# Remove unneeded NAs:
longdat.nonan <- longdat[which(is.na(longdat$Age) == F),]
vitstat.nonan <- vitstat[which(is.na(vitstat$BirthCohort) == F),]
```

```
data=prepare_data(longdat=longdat.nonan, vitstat=vitstat.nonan,interval=1, col.status="IsDead", col.id="ID"
# Parameters estimation:
pars=spm_quick_MD(data,k = 1)
pars
```

---

spm_time_dep *spm_time_dep : a function that can handle time-dependant coefficients:*

Description

spm_time_dep : a function that can handle time-dependant coefficients:

Usage

```
spm_time_dep(data, start = list(a = -0.5, f1 = 80, Q = 2e-08, f = 80, b = 5,
  mu0 = 1e-05, theta = 0.08), formulas = list(at = "a", f1t = "f1", Qt =
  "Q*exp(theta*t)", ft = "f", bt = "b", mu0t = "mu0*exp(theta*t)"))
```

Arguments

start	: a list of starting parameters, default: llist(a=-0.5, f1=80, Q=2e-8, f=80, b=5, mu0=1e-5, theta=0.08),
formulas	: a list of formulas that define age (time) - dependency. Default: list(at="a", f1t="f1", Qt="Q*exp(theta*t)", ft="f", bt="b", mu0t="mu0*exp(theta*t)")

Value

optimal coefficients

Examples

```
library(spm)
Data preparation:
N <- 1000
data <- simdata_cont(N=N, aH=-0.05, f1H=80, QH=2e-8, fH=80, bH=5, mu0H=2e-5, thetaH=0.08)
opt.par <- spm_time_dep(data[,2:6], formulas=list(at="a", f1t="f1", Qt="Q*exp(theta*t)", ft="f", bt="b", mu0
opt.par
```

Index