

# spm: an R-infrastructure package for Stochastic Process Modeling of survival trajectories from longitudinal studies

*Ilya Y. Zhbannikov*

*2016-01-24*

## Overview

The R-package `spm` (<https://github.com/izhbannikov/spm>) is developed for modeling trajectories from longitudinal data and it allows (1) data simulation and (2) estimating the process parameters using maximum likelihood estimation by optimizing parameters used in the model. Specifically, developed R-package `spm` allows (i) one-dimensional SPM; (ii) multiple dimensional SPM; (iii) data simulation for one- and multiple dimensions.

## Installation

```
require(devtools)
devtools::install_github("izhbannikov/spm")
```

If you experience errors during installation, please download a binary file from the following url: [https://github.com/izhbannikov/spm/blob/master/bin/win/spm\\_1.0.zip](https://github.com/izhbannikov/spm/blob/master/bin/win/spm_1.0.zip)

Then, execute this command (from R environment):

```
install.packages("<path to the downloaded r-package spm>", repos=NULL, type="binary")
```

## Data description

Data represents a typical longitudinal data in form of two datasets: longitudinal dataset (follow-up studies), in which one record represents a single observation, and vital (survival) statistics, where one record represents all information about the subject. Longitudinal dataset `cat` contain a subject ID (identification number), status (event(1)/no event(0)), time and measurements across the variables. The `spm` can handle an infinite number of variables but in practice, 5-7 variables is enough.

Below there is an example of clinical data that can be used in `spm` and we will discuss the field later. Longitudinal studies:

##	ID	Indicator	Death	Age	DBP	BMI
## 1	1		0	30	80.00000	25.00000
## 2	1		0	32	80.51659	26.61245
## 3	1		0	34	77.78412	29.16790
## 4	1		0	36	77.86665	32.40359
## 5	1		0	38	96.55673	31.92014
## 6	1		0	40	94.48616	32.89139

Vital statistics:

```
##   ID IsDead   LSmort
## 1  1      1 85.34578
## 2  2      1 80.55053
## 3  3      1 98.07315
## 4  4      1 81.29779
## 5  5      1 89.89829
## 6  6      1 72.47687
```

## Data fields description

### Longitude studies

- ID - subject unique identificatin number.
- IndicatorDeath - 0/1, indicates death of a subject.
- Age - current age of subjects.
- AgeNext - next age of subject he will attend to the survey/exam.
- DBP, BMI - covariates, here “DBP” represents a diastolic blood pressure, “BMI” a body-mass index.

### Survival statistics

- ID - subject’s unique ID.
- IsDead - death indicator, 0 - alive, 1 - dead.
- LSmort - age at death of stopping observations.

## Discrete and Continuous cases

There are two main SPM types in the package: discrete model and continuous model. Discrete model assumes equal intervals between follow-up observations. The example of discrete dataset is given below.

```
library(spm)
data <- simdata_discr(N=10, ystart=c(80), k=1)
head(data)
```

```
##      id xi t1 t2      y1 y1.next
## [1,]  1  0 30 31 80.00000 85.83903
## [2,]  1  0 31 32 85.83903 87.64522
## [3,]  1  0 32 33 87.64522 95.65693
## [4,]  1  0 33 34 95.65693 90.05915
## [5,]  1  0 34 35 90.05915 88.66307
## [6,]  1  0 35 36 88.66307 94.94620
```

In this case there are equal intervals between t1 and t2 (Age and Age.next).

The opposite is continuous case, in which intervals between observations are not equal. The example of continuous case dataset is shown below:

```
library(spm)
data <- simdata_cont(N=5, ystart = c(50))
head(data)
```

```

##      id xi      t1      t2      y1  y1.next
## 1    1  0 65.71834 68.36441 44.10038 43.37688
## 2    1  0 68.36441 69.66032 43.37688 53.13613
## 3    1  0 69.66032 71.74222 53.13613 57.40598
## 4    1  0 71.74222 74.40194 57.40598 43.57437
## 5    1  0 74.40194 75.44397 43.57437 41.46099
## 6    1  0 75.44397 76.50128 41.46099 49.42497

```

### Discrete case

In discrete case, we use the following assumptions:

$$\bar{y}(t+1) = \bar{u} + \bar{R} \times \bar{y}(t) + \bar{\epsilon}$$

(1)

$$\mu(t) = \mu_0(t) + \bar{b}(t) \times \bar{y}(t) + \bar{Q} \times \bar{y}(t)^2$$

(2)

Where:

$$\mu_0(t) = \mu_0 e^{\theta t}$$

$$\bar{b}(t) = \bar{b} e^{\theta t}$$

$$\bar{Q}(t) = \bar{Q} e^{\theta t}$$

### Continuous case

$$\mu(u) = \mu_0(u) + (\bar{m}(u) - \bar{f}(u)^* \times \bar{Q}(u) \times (\bar{m}(u) - \bar{f}(u)) + Tr(\bar{Q}(u) \times \bar{\gamma}(u))$$

(3)

$$dm(t)/dt = \bar{a}(t) \times (\bar{m}(t) - \bar{f}_1(t)) - 2\bar{\gamma}(t) \times \bar{Q}(t) \times (\bar{m}(t) - \bar{f}(t))$$

(4)

$$d\bar{\gamma}(t)/dt = \bar{a}(t) \times \bar{\gamma}(t) + \bar{\gamma}(t) \times \bar{a}(t)^* + \bar{b}(t) \times \bar{b}(t)^* - 2\bar{\gamma}t \times \bar{Q}(t) \times \bar{\gamma}(t)$$

(5)

### Coefficient conversion between continuous and discrete cases

$$Q = \bar{Q}$$

$$\bar{a} = \bar{R} - diag(k)$$

$$\bar{b} = \bar{\epsilon}$$

$$\bar{f}_1 = -1 \times \bar{u} \times a^{-1}$$

$$\bar{f} = -0.5 \times \bar{b} \times Q^{-1}$$

$$mu_0 = m u_0 - \bar{f} \times \bar{Q} \times t(\bar{f})$$

$$\theta = \bar{\theta}$$

## Case with time-dependent coefficients

In two previous cases, we assumed that coefficients is sort of time-dependant: we multiplied them on to

$$e^{\theta t}$$

. In general, this may not be the case. We extend this to a general case, i.e. (we consider one-dimensional case):

$$a(t) = par_1 t + par_2$$

- linear function.

The corresponding equations will be equivalent to one-dimensional continuous case described above.

## Simulation

We added one- and multi- dimensional simulation to be able to generate test data for hypothesis testing. Data, which can be simulated can be discrete (equal intervals between observations) and continuous (with arbitrary intervals).

### Discrete

The corresponding function is:

```
simdata_discr(N=100, a=-0.05, f1=80, Q=2e-8, f=80, b=5, mu0=1e-5, theta=0.08, ystart=80,
tstart=30, tend=105, dt=1, k=1)
```

Here:

**N** - Number of individuals

**a** - A matrix of **k** by **k**, 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 of **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 (baseline hazard)

**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 a start time (30 by default)

**tend** - A number, defines a final time (105 by default)

**dt** - A time interval between observations.

**k** - number of dimensions (1 by default)

This function returns a table with simulated data, as shown in example below:

```
library(spm)
data <- simdata_discr(N=10, ystart=c(75, 94), k=2)
head(data)
```

```
##      id xi t1 t2      y1 y1.next      y2 y2.next
## [1,]  1  0 30 31 75.00000 74.99694 94.00000 87.72651
## [2,]  1  0 31 32 74.99694 67.89194 87.72651 85.42609
## [3,]  1  0 32 33 67.89194 58.94846 85.42609 81.48598
## [4,]  1  0 33 34 58.94846 56.29565 81.48598 80.29210
## [5,]  1  0 34 35 56.29565 48.81120 80.29210 78.03831
## [6,]  1  0 35 36 48.81120 39.41164 78.03831 72.87558
```

## Continuous

The corresponding function is:

```
simdata_cont(N=100, a=-0.05, f1=80, Q=2e-07, f=80, b=5, mu0=2e-05, theta=0.08, ystart=80,
tstart=30, tend=105, k=1)
```

Here:

**N** - Number of individuals

**a** - A matrix of  $k \times k$ , which characterizes the rate of the adaptive response

**f1** - A particular state, which is a deviation from the normal (or optimal). This is a vector with length of  $k$

**Q** - A matrix of  $k$  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 (baseline hazard)

**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 a start time (30 by default)

**tend** - A number, defines a final time (105 by default)

**k** - number of dimensions (1 by default)

This function returns a table with simulated data, as shown in example below:

```
library(spm)
data <- simdata_cont(N=10)
head(data)
```

```
##      id xi      t1      t2      y1 y1.next
## 1  1  0 101.14598 103.99070 80.94372 74.57742
## 2  1  1 103.99070 106.88418 74.57742      NA
## 3  2  0  61.71377  63.92138 82.35719 79.72499
## 4  2  0  63.92138  65.20908 79.72499 74.74778
## 5  2  0  65.20908  68.02169 74.74778 65.87406
## 6  2  0  68.02169  69.65108 65.87406 73.30042
```

## Simulation strategies

R-package **spm** currently offers continuous- and discrete time simulations. Below we describe the simulations in details. In general, the input to each corresponding function: **simdata\_cont\_MD(...)** for continuous-time and **simdata\_discr\_MD(...)** for discrete-time simulations.

## Continuous-time simulation strategies

### Step 1

We model observations from a subject (which can be any system in general) and at first, we think that the subject is alive and compute the starting observation time `t1` and the next time `t2`:

```
t1 = runif(1, tstart, tend) t2 = t1 + 2*runif(1, 0, 1)
```

Here `runif()` a random number generator which returns uniformly distributed value. We assume that the `t1` as a random value, uniformly distributed from the start time (`tstart`) to end (`tend`).

### Step 2

Computing `y1` (an observed variable) from the previous observation:

```
if event = False:
    y1 = rnorm(1, ystart, sd0)
} else {
    y1 = y2
}
```

Here `rnorm(...)` is a random number generator which returns normally distributed values.

### Step 3

In order to compute `y2`, we need to compute a survival function `S` based on the equations 3, 4 and 5. We then compare the `S` to the random number, uniformly distributed. If `S` is larger than that number, then we assume that the event is happened (death of subject or system failure). Otherwise we compute `y2` and proceed to the next iteration:

```
if S > runif(1, 0, 1) :
    y2 = rnorm(1, m, sqrt(gamma))
    event = True
    new_subject = True
else if event = False:
    y2 = rnorm(1, m, sqrt(gamma))
    event = False
    new_record = True
```

## Discrete-time simulation strategies

In this case we use equal intervals `dt` between observations and survival function `S` is computed directly from  $\mu$  (2):

$$S = e^{-1\mu(t_1)}$$

The rest of the discrete simulation routine is the same as in continuous-time simulation case.