Package ‘BayesBP’

October 2, 2019

Type   Package
Title  Bayesian Estimation using Bernstein Polynomial Fits Rate Matrix
Version 1.0
Author Li-Syuan Hong [aut, cre]
Maintainer Li-Syuan Hong <lisyuan@nhri.org.tw>
Description Smoothed lexis diagrams with Bayesian method specifically tailored to cancer incidence data. Providing to calculating slope and constructing credible interval.
License GPL (>= 2)
Encoding UTF-8
Date 2019-09-02
LazyLoad yes
LazyData yes
Depends R (>= 3.5.0),parallel,iterators,utils,stats,openxlsx
RoxygenNote 6.1.1
Suggests testthat (>= 2.1.0)
NeedsCompilation no
Repository CRAN
Date/Publication 2019-10-02 10:20:05 UTC

R topics documented:

BP2D ......................................................... 2
BP2D_coef .................................................. 4
BP2D_table ............................................... 5
BPbasis .................................................. 5
BPbasis_pd_ages ......................................... 6
BPbasis_pd_years ....................................... 7
BPFhat .................................................. 7
BP2D

Bayesian estimation using two dimensions Bernstein polynomial

Description

This function runs Metropolis-Hasting algorithm which is given setting prior and data. This algorithm starts storing coefficients when it runs halfway, so we use second halves of coefficients compute Rhat to check convergence.

Usage

BP2D(prior, input_data = input_data, ages = ages, years = years,
    Iterations = 2e+05, n_cluster = 1, n_chain = 5, RJC = 0.35,
    nn = 2, seed = TRUE, set = 1, interval = 100, double = 4)

Arguments

prior = (n0,alpha,L) where alpha is a Poisson parameter, n0 is upper bound of alpha L can be every number which is bigger than one.
input_data = It contain disease and population(ex: simulated_data_1).
age = Range of ages.
years = Range of years.
Iterations = Iterations of chain.
n_cluster = This parameter means number of cores, five cores is recommended.(default: n_cluster=1).
n_chain = Number of Markov chain.
RJC = Control parameter for transfer dimension.
nn = The parameter nn is lower bound of alpha.
seed = Set seed yes or not.
set = Choose seed.(defaults:set=1)
interval = Each hundreds save one coefficient.
double = If R.hat >1.1 then double the iterations of times.
**Value**

This function will return Bayesian estimate of incidence, stored parameters, posterior mean, posterior max and table.

- **Fhat**: Bayesian estimate of incidence.
- **chain**: Bayesian estimate of posterior p-value mean.
- **maxchain**: Bayesian estimate of posterior p-value max.
- **store_coefficients**: Two dimensional Bernstein coefficients.
- **output**: When M-H algorithm ends, construct the table which contains norm, mean of Fhat, maximum of Fhat, R.hat, iterations, P-value and elapsed time.

**References**


**See Also**

Other Bayesian estimate: **BP2D_coef**, **BP2D_table**

**Examples**

```r
library(BayesBP)
# simulated_data_1, simulated_data_2
# Ages 1-85, years 1988-2007
# Data are zero from 0 to 34
# Given one prior and simulated_data_1
data('simulated_data_1')
ages<-35:85
years<-'1988:2007'
prior<-c(10,5,2)
result<-BP2D(prior, simulated_data_1, ages, years, n_cluster=1)
result$Fhat
result$chain
result$maxchain
result$output
result$store_coefficients$chain_1
matplot(result$chain,type='l',main='Posterior mean trace plot')
matplot(result$maxchain,type='l',main='Posterior max trace plot')
BP2D_coef(result)
write.BP(result, filename = 'result.xlsx')
write.BP(result, filename = 'result.xlsx')

# Given four prior and simulated_data_2
data('simulated_data_2')
n0<-c(10,20,10,20)
alpha<-'c(5,10,5,10)
```


L<-c(2,2,4,4)
prior<-cbind(n0, alpha, L)
ages<-35:85
years<-1988:2007
results_list<-paste0('result_', letters[1:4])
for(i in 1:4){
    assign(results_list[i], BP2D(prior[i,], simulated_data_2, ages, years, n_cluster=1))
}
BPtable<-BP2D_table(results_list)
write.BPtable(BPtable, filename = 'BPtable.xlsx')
mapply(write.BP, results_list, paste0(results_list, '.xlsx'))

#Credible interval
CI<-Credible_interval(result, n_cluster = 1)
CI_pda<-Credible_interval_pd_ages(result, n_cluster = 1)
CI_pdy<-Credible_interval_pd_years(result, n_cluster = 1)
CI
CI_pda
CI_pdy

---

**BP2D_coef**

*Getting coefficients from BP2D result.*

**Description**

This function will return coefficients and length of each set of coefficient.

**Usage**

```r
BP2D_coef(result)
```

**Arguments**

- `result`: This is output of BP2D.

**Value**

Coefficients table.

**See Also**

Other Bayesian estimate: `BP2D_table, BP2D`
BP2D_table

Table and Criterion.

Description
If you give more groups of prior, you can use this function to get the table and T criterion.

Usage
BP2D_table(results_list)

Arguments
results_list
A vector of characters.

Value
Table and criterion T.

See Also
Other Bayesian estimate: BP2D_coef, BP2D

BPbasis

Bernstein polynomial basis.

Description
This function build two dimensional Bernstein polynomial basis.

Usage
BPbasis(n0, ages, years, nn = 1)

Arguments
n0
Upper bound of poission random variable.
ages
Range of ages.
years
Range of years.
nn
Lower bound of poission random variable.

Value
Bernstein basis.
See Also

Other Bernstein basis: **BPbasis_pd_ages**, **BPbasis_pd_years**

Examples

```r
ages<-35:85
list.basis<-BPbasis(10,ages,tyears)
list.basis
```

---

**BPbasis_pd_ages**

Partial differential Bernstein polynomial basis.

### Description

This function build two dimensional partial differential Bernstein polynomial basis by ages.

### Usage

```r
BPbasis_pd_ages(n0, ages, years, nn = 1)
```

### Arguments

- `n0`  
  Upper bound of possion random variable.
- `ages`  
  Range of ages.
- `years`  
  Range of years.
- `nn`  
  Lower bound of possion random variable.

### Value

Partial differential Bernstein basis by ages.

### See Also

Other Bernstein basis: **BPbasis_pd_years**, **BPbasis**

### Examples

```r
ages<-35:85
list.basis<-BPbasis_pd_ages(10,ages,year)
list.basis
```
**BPbasis_pd_years**

Partial differential Bernstein polynomial basis.

**Description**

This function builds a two-dimensional partial differential Bernstein polynomial basis by years.

**Usage**

```r
BPbasis_pd_years(n0, ages, years, nn = 1)
```

**Arguments**

- `n0`: Upper bound of Poisson random variable.
- `ages`: Range of ages.
- `years`: Range of years.
- `nn`: Lower bound of Poisson random variable.

**Value**

Partial differential Bernstein basis by years.

**See Also**

Other Bernstein basis: `BPbasis_pd_ages`, `BPbasis`

**Examples**

```r
ages<-35:85
years<-1988:2007
list.basis<-BPbasis_pd_years(10, ages, years)
list.basis
```

---

**BPFhat**

*Bernstein polynomial*

**Description**

Given Bernstein polynomial coefficients to compute Fhat.

**Usage**

```r
BPFhat(coef, ages, years)
```
Arguments

coef     Bernstein polynomial coefficients.
ages     Range of ages.
years    Range of years.

Value

This function return outer Bernstein polynomial using coefficients.

See Also

Other outer Bernstein polynomial: BPFhat_pd_ages, BPFhat_pd_years

Examples

coef<-runif(9)
ages<-35:85
years<-1988:2007
BPFhat(coef,ages,years)

Description

Given Bernstein polynomial coeffients to compute partial differential by ages Fhat

Usage

BPFhat_pd_ages(coef, ages, years)

Arguments

coef     Bernstein polynomial coefficients.
ages     Range of ages.
years    Range of years.

Value

This function return outer Bernstein polynomial using coefficients.

See Also

Other outer Bernstein polynomial: BPFhat_pd_years, BPFhat
Examples

```r
coeff <- runif(9)
ages <- 35:85
years <- 1988:2007
BPFhat_pd_ages(coeff, ages, years)
```

Description

Given Bernstein polynomial coefficients to compute partial differential by years $F^{\cdot\cdot}$.

Usage

```r
BPFhat_pd_years(coeff, ages, years)
```

Arguments

- `coeff`: Bernstein polynomial coefficients.
- `ages`: Range of ages.
- `years`: Range of years.

Value

This function returns the outer Bernstein polynomial using coefficients.

See Also

Other outer Bernstein polynomial: `BPFhat_pd_ages, BPFhat`

Examples

```r
coeff <- runif(9)
ages <- 35:85
years <- 1988:2007
BPFhat_pd_years(coeff, ages, years)
```
Credible_interval_pd_ages

Description
Builing two dimensional Bernstein polynomial credible interval.

Usage
```r
Credible_interval(result, n_cluster = 1, alpha = 0.05)
```

Arguments
- `result` This is output of BP2D.
- `n_cluster` Maticores is recommended.(default:n_cluster=1)
- `alpha` Level of significance.

Value
This function returns Bayesian credible interval with level of significance.

References

See Also
Other Credible interval: `Credible_interval_pd_ages`, `Credible_interval_pd_years`

Credible_interval_pd_ages

Description
Builing partial differential Bernstein polynomial credible interval.

Usage
```r
Credible_interval_pd_ages(result, n_cluster = 1, alpha = 0.05)
```
Arguments

- **result**: This is output of BP2D.
- **n_cluster**: Multi-cores is recommended (default: n_cluster=1)
- **alpha**: Level of significance.

Value

This function returns a Bayesian credible interval with a specified level of significance.

See Also

Other credible intervals: `Credible_interval_pd_years`, `Credible_interval`

---

**Credible_interval_pd_years**

*Credible interval.*

**Description**

Building partial differential Bernstein polynomial credible interval.

**Usage**

```r
credible_interval_pd_years(result, n_cluster = 1, alpha = 0.05)
```

**Arguments**

- **result**: This is output of BP2D.
- **n_cluster**: Multi-cores is recommended (default: n_cluster=1)
- **alpha**: Level of significance.

**Value**

This function returns a Bayesian credible interval with a specified level of significance.

**See Also**

Other credible intervals: `Credible_interval_pd_ages`, `Credible_interval`
## Mapping to $[0,1]$ (mapping_to_01)

### Description
Mapping to $[0,1]$

### Usage
```
mapping_to_01(x)
```

### Arguments
- **x**: Vector.

### Examples
```
mapping_to_01(35:85)
(35:85)/(85-35)
mapping_to_01(runif(10))
```

## Gelman Rubin statistics (Rhat)

### Description
Check Markov chains for convergence.

### Usage
```
Rhat(M, burn.in = 0.5)
```

### Arguments
- **M**: An n x m numeric matrix of Markov Chains.
- **burn.in**: The default value 0.5 means that the second halves of chains will be used to compute.

### Value
Gelman Rubin statistics.

### References
Generate simulated data

Description
It contains number of patients and risky population.

Usage
data(simulated_data_1)

Format
data frame

Examples
#Generate simulated data
#Given incidence rate function 1
FT1<-function(x,y){
  rate<-0.00148*sin(0.5*pi*x*y)+0.00002
  return(rate)
}
#Given risky population function
M<-function(x,y){
  r0=152040 ; r1=-285270; r2=110410; r3=173900
  r4=-49950 ; r5=-33630 ; r6=-19530; r7=-110330
  r8=88840 ; r9=-7990
  population<-r0+r1*x+r2*y+r3*x^2+
      r4*x*y+r5*x^2*y+r6*x^3+
      r7*x^2*y+r8*x*y^2+r9*y^3
  return(population)
}
ages<-35:85
years<-1988:2007
gen_data<-function(ages,years,FT,M){
  x<-mapping_to_01(ages)
y<-mapping_to_01(years)
disease<-outer(x,y,M)*outer(x,y,FT)
population<-outer(x,y,M)
zero<-matrix(0,ncol = 2*length(y),nrow = min(ages)-1)
simulated_data<-rbind(zero,cbind(disease,population))
colnames(simulated_data)<-rep(years,2)
row.names(simulated_data)<-1:max(ages)
return(simulated_data)
}
simulated_data_1<-gen_data(ages,years,FT1,M)
simulated_data_1
simulated_data_2 Generate simulated data

Description
It contains number of patients and risky population.

Usage
data(simulated_data_2)

Format
data frame

Examples
#Generate simulated data
#Given incidence rate function 2
FT2<-function(x,y){
  rate<-0.00148*sin(0.5*pi*x*(y+0.2))+0.00002
  return(rate)
}
#Given population function
M<-function(x,y){
  r0=152040 ;r1=-285270; r2=173900
  r4=-49950 ;r5=-33630 ; r6=-19530; r7=-110330
  r8=88840 ;r9=-7990
  population<-r0+r1*x+r2*y+r3*x^2+
    r4*x*y+r5*y^2+r6*x^3+
    r7*x^2*y+r8*x*y^2+r9*y^3
  return(population)
}
ages<-35:85
years<-1988:2007
gen_data<-function(ages,years,FT,M){
  x<-mapping_to_01(ages)
  y<-mapping_to_01(years)
  disease<-outer(x,y,M)*outer(x,y,FT)
  population<-outer(x,y,M)
  zero<-matrix(0,ncol = 2*length(y),nrow = min(ages)-1)
  simulated_data<-rbind(zero,cbind(disease,population))
  colnames(simulated_data)<-rep(years,2)
  row.names(simulated_data)<-1:max(ages)
  return(simulated_data)
}
simulated_data_2<-gen_data(ages,years,FT2,M)
simulated_data_2
**write.BP**

*Write xlsx file*

**Description**

This function will write result of BP2D to xlsx file.

**Usage**

`write.BP(writedata, filename)`

**Arguments**

- `writedata`: result of BP2D(character or list).
- `filename`: xlsx file name.

---

**write.BPtable**

*Write BPtable as xlsx file*

**Description**

If your environment has some result of BP2D, then you can use this function to store BPTable.

**Usage**

`write.BPtable(BPtable, filename)`

**Arguments**

- `BPtable`: output of BP2D_table.
- `filename`: xlsx file name.
Index

*Topic datasets
  simulated_data_1, 13
  simulated_data_2, 14

BP2D, 2, 4, 5
BP2D_coef, 3, 4, 5
BP2D_table, 3, 4, 5
BPbasis, 5, 6, 7
BPbasis_pd_ages, 6, 6, 7
BPbasis_pd_years, 6, 7
BPFhat, 7, 8, 9
BPFhat_pd_ages, 8, 8, 9
BPFhat_pd_years, 8, 9

Credible_interval, 10, 11
Credible_interval_pd_ages, 10, 10, 11
Credible_interval_pd_years, 10, 11, 11

mapping_to_01, 12

Rhat, 12

simulated_data_1, 13
simulated_data_2, 14

write.BP, 15
write.BPtable, 15