Package 'polyapost'

July 2, 2014

Version 1.1-6

Date 2014-04-21

Imports boot

Title Simulating from the Polya posterior

Author Glen Meeden <glen@stat.umn.edu> and Radu Lazar <lazar@stat.umn.edu>

Maintainer Glen Meeden <glen@stat.umn.edu>

Description Generate dependent samples from a non-full dimensional
polytope via a Markov Chain sampler

License GPL (>= 2)

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-04-22 07:25:57

R topics documented:

constrppmn *Estimating a population mean using the constrained Polya posterior.*

Description

Let p=(p1,...,pn) be a probability distribution defined on ysamp, the set of observed values, in a sample of size n from some population. p is assumed to belong to a polytope which is a lower dimensional subset of the n-dimensional simplex. The polytope is defined by a collection of linear equality and inequality constraints. A dependent sequence of values for p are generated by a Markov chain using the Metropolis-Hastings algorithm whose stationary distribution is the uniform distribution over the polytope. For each generated value of p the corresponding mean, sum(pi*yi) is found.

Usage

```
constrppmn(A1,A2,A3,b1,b2,b3,initsol,reps,ysamp,burnin)
```

Arguments

A1	The matrix for the equality constraints.This must always contain the constraint that the sum of the pi's is one.
A2	The matrix for the <= inequality constraints. This must always contain the constraints -pi <= 0, i.e. that the pi's must be nonnegative.
A3	The matrix for the >= inequality constraints. If there are no such constraints A3 must be set equal to NULL.
b1	The rhs vector for A1, each component must be nonnegative.
b2	The rhs vector for A2, each component must be nonnegative.
b3	The rhs vector for A3, each component must be nonnegative. If A3 is NULL then b3 must be NULL.
initsol	A vector which lies in the interior of the polytope.
reps	The total length of the chain that is generated.
ysamp	The observed sample from the population of interest.
burnin	The point in the chain at which the set of computed means begins.

Value

The returned value is a list whose first component is the chain of the means of length (reps - burnin -1), whose second component is the mean of the first component (i.e. the Polya estimate of the population mean) and whose third component is the 2.5th and 97.5th quantiles of the first component (i.e. an approximate 95 percent confidence interval of the population mean).

Examples

```
A1<-rbind(rep(1,6),1:6)
A2<-rbind(c(2,5,7,1,10,8),diag(-1,6))
b1<-c(1,3.5)
b2<-c(6,rep(0,6))
initsol<-rep(1/6,6)
rep<-1006
burnin<-1000
ysamp<-c(1,2.5,3.5,7,4.5,6)
out<-constrppmn(A1,A2,NULL,b1,b2,NULL,initsol,rep,ysamp,burnin)
out[[1]] # the Markov chain of the means.
out[[2]] # the average of out[[1]]
out[[3]] # the 2.5th and 97.5th quantiles of out[[1]]
```

---

constrpppprob *Dependent sampling from the unifrom distribution on a polytope.*

Description

Let p=(p1,...,pn) be a probability distribution which belongs to a lower dimensional polytope of the n-dimensional simplex. The polytope is defined by a collection of linear equality and inequality constraints. A dependent sequence of the p's are generated by a Markov chain using the Metropolis-Hastings algorithm whose stationary distribution is the uniform distribution over the polytope. This is done by generating k blocks of size step where the last member of each is returned.

Usage

```
constrpppprob(A1,A2,A3,b1,b2,b3,initsol,step,k)
```

Arguments

A1	The matrix for the equality constraints.This must always contain the constraint that the sum of the pi's is one.
A2	The matrix for the <= inequality constraints. This must always contain the constraints -pi <= 0, i.e. that the pi's must be nonnegative.
A3	The matrix for the >= inequality constraints. If there are no such constraints A3 must be set equal to NULL.
b1	The rhs vector for A1, each component must be nonnegative.
b2	The rhs vector for A2, each component must be nonnegative.
b3	The rhs vector for A3, each component must be nonnegative. If A3 is NULL then b3 must be NULL.
initsol	A vector which lies in the interior of the polytope.
step	The number of p's generated in a block before the last member of a block is returned.
k	The total number of blocks generated and hence the number of p's returned.

Value

The returned value is a k by n matrix of probability vectors.

Examples

```
A1<-rbind(rep(1,6),1:6)
A2<-rbind(c(2,5,7,1,10,8),diag(-1,6))
A3<-matrix(c(1,1,1,0,0,0),1,6)
b1<-c(1,3.5)
b2<-c(6,rep(0,6))
b3<-0.45
initsol<-rep(1/6,6)
constrppprob(A1,A2,A3,b1,b2,b3,initsol,2000,5)
```

`feasible` *Feasible solution for a probability distribution which must satisfy a system of linear equality and inequality constraints.*

Description

This function finds a feasible solution, p=(p1,...,pn), in the n-dimensional simplex of probability distributions which must satisfy A1p = b1, A2p <= b2 and A3p >= b3. All the components of the bi's must be nonnegative In addition each probability in the solution must be at least as big as eps, a small positive number.

Usage

```
feasible(A1,A2,A3,b1,b2,b3,eps)
```

Arguments

`A1`	The matrix for the equality constraints.This must always contain the constraint that the sum of the pi's is one.
`A2`	The matrix for the <= inequality constraints. This must always contain the constraints -pi <= 0, i.e. that the pi's must be nonnegative.
`A3`	The matrix for the >= inequality constraints. If there are no such constraints A3 must be set equal to NULL.
`b1`	The rhs vector for A1, each component must be nonnegative.
`b2`	The rhs vector for A2, each component must be nonnegative.
`b3`	The rhs vector for A3, each component must be nonnegative. If A3 is NULL then b3 must be NULL.
`eps`	A small positive number. Each member of the solution must be at least as large as eps. Care must be taken not to choose a value of eps which is too large.

Value

The function returns a vector. If the components of the vector are positive then the feasible solution is the vector returned, otherwise there is no feasible solution.

Examples

```
A1<-rbind(rep(1,7),1:7)
b1<-c(1,4)
A2<-rbind(c(1,1,1,1,0,0,0),c(.2,.4,.6,.8,1,1.2,1.4))
b2<-c(1,2)
A3<-rbind(c(1,3,5,7,9,10,11),c(1,1,1,0,0,0,1))
b3<-c(5,.5)
eps<-1/100
feasible(A1,A2,A3,b1,b2,b3,eps)
```

---

polyap *Polya sampling from an urn*

Description

Consider an urn containing a finite set of values. An item is selected at random from the urn. Then it is returned to the urn along with another item with the same value. Next a value is selected at random from the reconstituted urn and it and a copy our returned to the urn. This process is repeated until k additional items have been added to the original urn. The original composition of the urn along with the selected values, in order, are returned.

Usage

```
polyap(ysamp, k)
```

Arguments

ysamp	A vector of real numbers which make up the urn.
k	A positive integer which specifies the number of items added to the original composition of the urn.

Value

The returned value is a vector of length equal to the length of ysamp plus k

Examples

```
polyap(c(0,1),20)
```

`wtpolyap` *Polya sampling from an urn with possibly unequal weights*

Description

Consider an urn containing a finite set of values along with their respective positive weights. An item is selected at random from the urn with probability proportional to its weight. Then it is returned to the urn and its weight is increased by one. The process is repeated on the adjusted urn. We continue until the total weight in the urn has been increased by k. The original composition of the urn along with the k selected values, in order, are returned.

Usage

```
wtpolyap(ysamp, wts, k)
```

Arguments

`ysamp`	A vector of real numbers which make up the urn.
`wts`	A vector of positive weights which defines the initial probability of selection.
`k`	A positive integer which specifies the number of Polya samples taken from the urn where after each draw the weight of the selected item is increased by one.

Value

The returned value is a vector of length equal to the length of the sample plus k

Examples

```
wtpolyap(c(0,1,2),c(0.5,1,1.5),22)
```

Index