

# Package ‘KnapsackSampling’

July 21, 2025

**Title** Generate Feasible Samples of a Knapsack Problem

**Version** 0.1.1

**Date** 2024-01-31

**Author** Chin Soon Lim [aut]

**Maintainer** Chin Soon Lim <chinsoon12@hotmail.com>

**Description** The `sampl.mcmc` function creates samples of the feasible region of a knapsack problem with both equalities and inequalities constraints.

**Depends** R (>= 3.3.0)

**Imports** lpSolve, stats

**License** GPL (>= 2) | file LICENSE

**RoxygenNote** 7.2.3

**URL** <https://github.com/chinsoon12/KnapsackSampling>

**BugReports** <https://github.com/chinsoon12/KnapsackSampling>

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2024-01-31 08:30:08 UTC

## Contents

<code>flip01</code> . . . . .	2
<code>initState</code> . . . . .	2
<code>sampl.mcmc</code> . . . . .	3

<b>Index</b>	5
--------------	---

---

flip01	<i>Flip a 1 and a 0 simultaneously</i>
--------	--

---

**Description**

Flip a 1 and a 0 simultaneously

**Usage**

```
flip01(x)
```

**Arguments**

x                    an integer or logical vector

**Value**

x an integer vector

---

initState	<i>Generate an initial feasible solution by solving a linear programming with binary variables</i>
-----------	--

---

**Description**

Generate an initial feasible solution by solving a linear programming with binary variables

**Usage**

```
initState(numVar, objVec = runif(numVar), constraints = NULL)
```

**Arguments**

numVar                - number of variables  
objVec                - objective function as a numeric vector  
constraints            - a list of list of constraints with constr.mat, constr.dir, constr.rhs in each sublist

**Value**

a binary vector containing a feasible solution

**Examples**

```
#see documentation for sampl.mcmc
```

---

sampl.mcmc	<i>Generate feasible solutions to a knapsack problem using Markov Chain Monte Carlo</i>
------------	---

---

**Description**

Generate feasible solutions to a knapsack problem using Markov Chain Monte Carlo

**Usage**

```
sampl.mcmc(init, numSampl, maxIter = 2 * numSampl, constraints = NULL)
```

**Arguments**

<code>init</code>	- an initial feasible solution
<code>numSampl</code>	- number of samples to be generated
<code>maxIter</code>	- maximum number of iterations to be run to prevent infinite loop
<code>constraints</code>	- a list of list of constraints with <code>constr.mat</code> , <code>constr.dir</code> , <code>constr.rhs</code> in each sublist. Please see example for an example of constraints.

**Value**

a matrix of {0, 1} with each row representing a sample

**Examples**

```
#number of variables
N <- 100

#number of variables in each group
grpLen <- 10

#equality matrix
A <- matrix(c(rep(1, N)), ncol=N, byrow=TRUE)

#inequality matrix
G <- matrix(c(rep(1, grpLen), rep(0, N - grpLen),
             rep(c(0,1), each=grpLen), rep(0, N - 2*grpLen)), ncol=N, byrow=TRUE)

#construct a list of list of constraints
constraints <- list(
  list(constr.mat=A, constr.dir=rep("==", nrow(A)), constr.rhs=c(20)),
  list(constr.mat=G, constr.dir=rep("<=", nrow(G)), constr.rhs=c(5, 5)),
  list(constr.mat=G, constr.dir=rep(">=", nrow(G)), constr.rhs=c(1, 2))
)

#generate an initial feasible solution
init <- initState(N, constraints=constraints)
```

```
#create feasible solutions to knapsack problems subject to constraints  
samples <- sampl.mcmc(init, 50, constraints=constraints)
```

# Index

`flip01`, 2

`initState`, 2

`sampl.mcmc`, 3