Package ‘Markovchart’

March 15, 2020

Type Package

Title Markov Chain-Based Cost-Optimal Control Charts

Version 1.0.6

Author Balazs Dobi & Andras Zempleni

Maintainer Balazs Dobi <dobib@cs.elte.hu>


License GPL

Encoding UTF-8

LazyData true

Imports stats, parallel, doParallel, optimParallel, foreach, ggplot2, metR

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-15 10:30:08 UTC

R topics documented:

Markovchart ................................................................. 2
Markovcontour ............................................................ 8
Markovsim ................................................................. 9

Index 12
Economic design for the X-bar control chart with random shift size, random repair and random sampling time.

Description

Wrapper for Markov chain-based cost optimal control charts. Includes methods for different shift size distributions and optimisation with respect to the average cost and cost standard deviation where the free parameters are the sampling interval (h) and the control limit (k).

Usage

```r
Markovchart(shiftfun = c("exp", "exp-geo", "deg"), h, k, sigma, s, delta, probmix = 0.5, probnbin = 0.5, disj = 1, RanRep = FALSE, alpha = NULL, beta = NULL, RanSam = FALSE, a = NULL, b = NULL, StateDep = TRUE, q = NULL, z = NULL, p = 1, Vd = 50, V, Qparam = 25, COST = c("no", "yes", "optim"), constantr = FALSE, cs = NULL, cofun = cofun_default, coparams = NULL, crfun = crfun_default, crparams = NULL, cf = crparams, vcofun = vcofun_default, vcoparams = c(0, 0), vcrfun = vcrfun_default, vcrparams = c(0, 0), method = c("L-BFGS-B", "Nelder-Mead", "BFGS", "CG", "SANN", "Brent"), parallel_opt = NULL, silent = TRUE, ...)
```

Arguments

- `shiftfun`: A string defining the shift size distribution to be used. Must be either "exp" (exponential), "exp-geo" (exponential-geometric mixture) or "deg" (degenerate). Use "deg" for fixed shift size with perfect repair and guaranteed sampling i.e. Duncan's traditional cycle model. (Random repair and random sampling is not implemented for the fixed shift size distribution.)
- `h`: The time between samplings. Must be a positive value, can be a numeric vector. For optimisation, this is the initial value.
- `k`: The control limit (critical value). Must be a positive value, can be a numeric vector. For optimisation, this is the initial value. Only one sided shifts are allowed, thus there is only one control limit.
- `sigma`: Process standard deviation (the distribution is normal).
- `s`: Expected number of shifts in an unit time interval.
- `delta`: Expected shift size. Used as the parameter of the exponential distribution (shiftfun="exp" or "exp-geo"), or simply as the size of the shift (shiftfun="deg").
- `probmix`: The weight of the geometric distribution in case of exponential-geometric mixture shift distribution and should be between 0 and 1.
- `probnbin`: The probability parameter of the geometric distribution in case of exponential-geometric mixture shift distribution and should be between 0 and 1.
### disj
The size of a discrete jump in case of exponential-geometric mixture shift distribution, must be a positive number.

### RanRep
Logical. Should the repair be random? Default is FALSE (no). The repair is always perfect (non-random) for `shiftfun="deg"`.

### alpha
First shape parameter for the random repair beta distribution.

### beta
Second shape parameter for the random repair beta distribution.

### RanSam
Logical. Should the sampling be random? Default is FALSE (no). The sampling is never random for `shiftfun="deg"`.

### a
First parameter $\alpha h$ for the random sampling time beta distribution. The first shape parameter is $\alpha/h$ to create dependency on the time between samplings.

### b
Second shape parameter for the random sampling time beta distribution.

### StateDep
Logical. Should the sampling probability also depend on the distance from the target value (state dependency)? (If TRUE, a beta distribution is used for the sampling probability, if FALSE then a logistic function.)

### q
The steepness of the curve of the random sampling time logistic function.

### z
The logistic sigmoid’s midpoint of the random sampling time logistic function.

### p
The weight of the cost expectation in the calculation of the G-value and should be between 0 and 1.

### Vd
Integer discretisation parameter: the number of states after the equidistant discretisation of the state space. Should be an integer value greater than 2.

### V
Numeric discretisation parameter: the maximum (positive) distance from the target value taken into account.

### Qparam
Integer discretisation parameter: the number of maximum events taken into account within a sampling interval.

### COST
A string of value of either "no","yes" or "optim". Default is no, in which case only the stationary distribution is returned. If yes, statistics related to costs are calculated and cost parameters must be provided. Optimisation (when the value is optim) is only possible with respect to costs.

### constantr
Logical. Should the repair cost be assumed to constantly occur over time (TRUE) or assumed to only occur when there is a repair due to an alarm (FALSE)? See 'Details'.

### cs
Sampling cost per sampling.

### cofun
A function describing the relationship between the distance from the target value and the resulting out-of-control costs. Default is calculated using a base and a distance-scaling out-of-control parameter. See 'Details'.

### coparams
Numeric vector. Parameters of cofun.

### crfun
A function describing the relationship between the distance from the target value and the resulting repair costs. For the default function see 'Details'.

### crparams
Numeric vector. Parameters of crfun.

### cf
Numeric. The false alarm cost. Only relevant when `shiftfun` is "deg".

### vcofun
A function describing the relationship between the distance from the target value and the resulting out-of-control cost variance. For the default function see 'Details'.

---
vcoparams  Numeric vector. Parameters of vcofun.

cvctrfun  A function describing the relationship between the distance from the target value and the resulting repair cost variance. For the default function see 'Details'.

vcvtparams  Numeric vector. Parameters of vcvcf

method  Method used for optimisation. Same as the argument of optim, but the default here is "L-BFGS-B", because it turned out to be more robust in testing.

parallel_opt  A list of parallel options. See e.g. the argument parallel in the documentation of optimParallel. If left empty then the number of cores (threads) is automatically detected and all but one is used. (Single-core computers simply use one core.)

silent  Should the call be returned? Default is FALSE.

...  Further arguments to be passed down to optimParallel.

Details

The constantr parameter is used for different repair assumptions. In traditional control chart theory, repair cost only occurs in case of an alarm signal. This is represented by constantr=FALSE, which is the default. In this case the repair is just a momentary cost, occurring at the time of the sampling. However this model is inappropriate in several cases in healthcare. For example there are chronic diseases that require constant medication (repair in the sense of the model). In this approach (constantr=TRUE) the repair cost still depends on the state of the process during sampling, but occurs even if there is no alarm and is divided by $h$ to represent the constant repair through the whole sampling interval. Thus the repair cost should be given in a way which corresponds to the model used.

The default cofun calculates the out-of-control (OOC) cost using a base and a distance-scaling OOC parameter:

$$ c_o = c_{ob} + c_{os} \Delta \mu^2_0, $$

where $c_o$ is the total OOC cost, $c_{ob}$ is the base OOC cost (even without shift), $c_{os}$ is the shift-scaling cost and $\Delta \mu^2_0$ is the squared distance from the target value. This latter part is defined like this because we are using a Taguchi-type loss function. Even if the user defines a custom cost function for the OOC cost this $\Delta \mu^2_0$ term must be included, as a closed form solution has been developed for the calculation of the squared distances in case of exponential shifts, considerably decreasing run times. Thus the arguments of the OOC cost function should look like this: function($\Delta \mu^2_0$, other parameters contained in a vector). $\Delta \mu^2_0$ is fed to the cost function as a vector, thus the function should vectorised with respect to this argument. The default function looks like this:

```r
cofun_default <- function(sqmudist,coparams)
{
  sqmudist=sqmudist
  cob=coparams[1]
  cos=coparams[2]
  co <- cob + cos*sqmudist
  return(co)
}
cofun_default <- Vectorize(cofun_default,"sqmudist")
```
The default vcofun also uses a Taguchi-type loss function and has identical parts and requirements as cofun. The final standard deviation itself is calculated using the Total Law of Variance. The default vcofun is:

```r
vcofun_default <-function(sqmudist,vcoparams)
{
  sqmudist=sqmudist
  vcob=vcoparams[1]
  vcos=vcoparams[2]
  vco <-vcob + vcos*sqmudist
  return(vco)
}
```

```r
cfunc_default <-Vectorize(vcofun_default,"sqmudist")
```

The defaults for the repair cost and cost variance are simple linear functions. For crfun it is

\[
c_r = c_{rb} + c_{rs}\Delta \mu_0,
\]

where the notation are the same as before and "r" denotes repair. A custom function can be defined more freely here, but the first argument should be \(\Delta \mu_0\) and the second a vector of further parameters.

The default function are:

```r
crfun_default <-function(mudist,crparams)
{
  mudist=mudist
  crb=crparams[1]
  crs=crparams[2]
  cr <-crb + crs*mudist
  return(cr)
}
```

```r
crfun_default <-Vectorize(crfun_default,"mudist")
```

```r
vcrfun_default <-function(mudist,vcrparams)
{
  mudist=mudist
  vcrb=vcrparams[1]
  vcrs=vcrparams[2]
  vcr <-vcrb + vcrs*mudist;
  return(vcr)
}
```

```r
vcrfun_default <-Vectorize(vcrfun_default,"mudist")
```

Value

The value depends on the parameters: If \(h\) and \(k\) are both of length 1, no costs are calculated (\(\text{COST} = \text{no}\)) and shiftfun is not "deg" then the value is the stationary distribution of the Markov chain which is a numeric vector of length \(Vd*2\). The length is double of \(Vd\) because each state has an alarm and a non-alarm (out-of-control) version. If shiftfun is "deg" then the stationary distribution...
is always of length 4. The probabilities in the stationary distribution are labeled. If shiftfun is not "deg" then there are multiple out-of-control and true alarm states. These are labeled with an index and the value the state represents. If shiftfun is "deg" then the out-of-control and true alarm states are at a distance $\delta$ from the target value. The in-control and the false alarm state are always at the target value.

If either h or k have length greater than 1, then the G-value (weighted average of average cost and cost standard deviation) is calculated for all given values without optimisation. The value of the function in this case is a data frame with $\text{length}(h) \times \text{length}(k)$ number of rows and three columns for h, k and the G-value.

If h and k are both of length 1 and costs are calculated (COST="yes" or "optim"), then the value of the function is a "Markov_chart" object, which is a list of length 4, detailing the properties of the control chart setup.

Results
Vector of G-value, expected cost, cost standard deviation and further process moments. Note that these further moments only take into account the process variation (i.e. the standard deviation), while the "Total cost std. dev." takes into account all sources of variance (e.g. the different costs that can occur due to being out-of-control).

Subcosts
Vector of sub-costs that are parts of the total expected cost.

Parameters
A vector that contains the time between samplings (h) and critical value (k) which was used in the control chart setup.

Stationary_distribution
The stationary distribution of the Markov chain. See above.

Author(s)
Balazs Dobi and Andras Zempleni

References


Examples

#Defining parallel_opt parallel settings
#parallel_opt can also be left empty to be defined automatically by the function
require(parallel)
num_workers <- min(c(detectCores(),2))
parall <- list(cl=makeCluster(num_workers), forward=FALSE, loginfo=TRUE)

#Fixed shift size (essentially Duncan's cycle model) - no optimisation
res <- Markovchart(h=1, k=1, shiftfun="deg", sigma=1, s=0.2, delta=2.5, crparams=20, coparams=50, COST="yes", lower = c(0.01,0.01), upper = c(5,5))
res

#Fixed shift size (essentially Duncan's cycle model) - with optimisation
res <- Markovchart(h=1, k=1, shiftfun="deg", sigma=1, s=0.2, delta=2.5, cs=1, crparams=20, coparams=50, COST="optim", lower = c(0.01,0.01), upper = c(5,5), parallel_opt=parall)
res

#Exponential shift - perfect repair - no optimisation - default cost functions - no cost calculation
res <- Markovchart(h=1, k=1, shiftfun="exp", sigma=1, s=0.2, delta=2, RanRep=FALSE, COST="no", cs=1, coparams=c(10,3), crparams=c(1,2), vcrparams=c(8000,100), vcoparams=c(50000,-600000,1.5), p=0.9, Vd=30, V=18)
res

#Notice how the In-control and the False-alarm states have non-zero probabilities
#If the repair is random (RanRep=TRUE), then these states have zero probability
head(res)

#Exponential shift - no optimisation - default cost functions - no cost calculation
res <- Markovchart(h=1, k=1, shiftfun="exp", sigma=1, s=0.2, delta=2, RanRep=TRUE, alpha=1, beta=3, COST="no", cs=1, coparams=c(10,3), crparams=c(1,2), vcrparams=c(50000,-600000,1.5), p=0.9, Vd=30, V=18)
res

#Exponential shift - no optimisation - default cost functions
res <- Markovchart(h=1, k=1, shiftfun="exp", sigma=1, s=0.2, delta=2, RanRep=TRUE, alpha=1, beta=3, COST="yes", cs=1, coparams=c(10,3), crparams=c(1,2), vcrparams=c(50000,-600000,1.5), p=0.9, Vd=30, V=18)
res

#Exponential shift - with optimisation - default cost functions
parall <- list(cl=makeCluster(num_workers), forward=FALSE, loginfo=TRUE)
res <- Markovchart(h=1, k=1, shiftfun="exp", sigma=1, s=0.2, delta=2, RanRep=TRUE, alpha=1, beta=3, COST="optim", cs=1, coparams=c(10,3), crparams=c(1,2), vcrparams=c(50000,-600000,1.5), p=0.9, Vd=30, V=18, parallel_opt=parall)
res

#Exponential-geometric mixture shift - no optimisation - custom repair variance function
vcrfun <-function(mudist,vcrparams)
{
mudist=mudist
vcrb=vcrparams[1]
vcrs=vcrparams[2]
vcrs2=vcrparams[3]

vcr <-vcrb + vcrs/(mudist + vcrs2)
return(vcr)
}
vcrfun <-Vectorize(vcrfun,"mudist")
res <- Markovchart(h=1, k=2, shiftfun="exp-geo", sigma=1, s=0.2, delta=2, probmix=0.3, probnbin=0.2, disj=1, RanRep=TRUE, alpha=1, beta=3, COST="yes", cs=1,
Markovcontour

Description

Convenience function for plotting G-values in a contour plot as the function of the time between samplings and the critical value.

Usage

Markovcontour(Gmtx, xlab = "Time between samplings", ylab = "Critical value", low = "white", mid = "#999999", high = "black", colour = "white", name = "G-value per \nunit time")

Arguments

Gmtx A data.frame with three columns (preferably created by the Markovchart function): time between samplings, critical value and the weighted mean of the expected cost and the cost standard deviation (G-values).

xlab A title for the x axis.

ylab A title for the x axis.

low Colour for the low end of the gradient.

mid Colour for mid point.

high Colour for the high end of the gradient.

colour Colour of the contour lines.

name The name of the scale.

Value

A plot object of class "gg" and "ggplot" produced using the ggplot2 package.

Note

The plot itself is made using the package ggplot2 by Hadley Wickham et al. The text on the contour lines is added with the geom_text_contour function from the package metR by Elio Campitelli.
Markovsim

Author(s)
Balazs Dobi and Andras Zempleni

References


See Also
Markovchart

Examples

```r
#Defining parallel_opt parallel settings
#parallel_opt can also be left empty to be defined automatically by the function
require(parallel)
num_workers <- min(c(detectCores(),2))
parall <-list(cl=makeCluster(num_workers), forward=FALSE, loginfo=TRUE)
Gmtx <-Markovchart(h=seq(1,10,by=(10-1)/5),k=seq(0.1,5,by=(5-0.1)/5), shiftfun="exp", sigma=1,s=0.2, delta=2, RanRep=TRUE, alpha=1, beta=3, COST="yes", cs=1, coparams=c(10,3), crparams=c(1,2), vcoparams=c(8000,100), vcrparams=c(50000,-600000,1.5), p=0.9, V=18, parallel_opt=parall)
Markovcontour(Gmtx)
```

---

### Markovsim

Progression and monitoring simulation of a process with random shift size, random repair and random sampling time.

**Description**

Wrapper for simulation of processes with a Markov chain-based control chart setup. Includes methods for different shift size distributions.

**Usage**

```r
Markovsim(num = 100, shiftfun = c("exp", "exp-geo"), h, k, sigma, s, delta, probmix = 1, probnbin = 0.5, RanRep = FALSE, alpha, beta, RanSam = FALSE, a = NULL, b = NULL, V)
```
Arguments

- `num` Integer. The number of sampling intervals simulated. This means that the time elapsed in the simulation is `num*h`.
- `shiftfun` A string defining the shift size distribution to be used. Must be either "exp", "exp-geo".
- `h` The time between samplings. Must be a positive value.
- `k` The control limit (critical value). Must be a positive value. Only one sided shifts are allowed, thus there is only one control limit.
- `sigma` Process standard deviation (the distribution is normal).
- `s` Expected number of shifts in an unit time interval.
- `delta` Expected shift size.
- `probmix` The weight of the geometric distribution in case of exponential-geometric mixture shift distribution and should be between 0 and 1.
- `probnbin` The probability parameter of the geometric distribution in case of exponential-geometric mixture shift distribution and should be between 0 and 1.
- `RanRep` Logical. Should the repair be random? Default is FALSE (no).
- `alpha` First shape parameter for the random repair beta distribution.
- `beta` Second shape parameter for the random repair beta distribution.
- `RanSam` Logical. Should the sampling be random? Default is FALSE (no).
- `a` First parameter*h for the random sampling time beta distribution. The first shape parameter is `a/h` to create dependency on the time between samplings.
- `b` Second shape parameter for the random sampling time beta distribution.
- `V` Numeric discretisation parameter: the maximum (positive) distance from the target value taken into account. This parameter is needed for the calculation of sampling probabilities in the case of random sampling.

Details

The simulation only includes the more complicated process and control chart cases and is meant for model checking and for situations when the exact calculation is problematic (such as low probabilities in the stationary distribution leading to rounding errors). The simulation creates 100 data points per sampling interval.

Value

A list of length 3.

- `Value_at_samplings` The process value at sampling.
- `Sampling_event` The event at sampling.
- `Simulation_data` The simulated data (distances from the target value).

The event at sampling can be success (there was a sampling but no alarm), alarm (sampling with alarm) and failure (no sampling occurred).
Author(s)

Balazs Dobi and Andras Zempleni

References


See Also

Markovchart

Examples

#Simulation using exponential shifts, random repair and random samling.
simres1 <- Markovsim(num=500, shiftfun="exp", h=1, k=1, sigma=1, s=0.2, delta=2,
                      RanRep=TRUE, alpha=1, beta=3, RanSam=TRUE, a=0.1, b=1, V=10)
simres1[[2]]
hist(simres1[[1]],20,freq=FALSE)

#Simulation using exponential-geometric mixture shifts, random repair and random samling.
simres2 <- Markovsim(num=500, shiftfun="exp-geo", h=1, k=1, sigma=1, s=0.2, delta=2,
                      probmix=0.9, probnbin=0.6, RanRep=TRUE, alpha=1, beta=3, RanSam=TRUE, a=0.1, b=1, V=10)
simres2[[2]]
hist(simres2[[1]],20,freq=FALSE)
Index

*Topic Markov chain
  Markovchart, 2
  Markovcontour, 8
  Markovsim, 9
*Topic contour
  Markovcontour, 8
*Topic control chart
  Markovchart, 2
  Markovcontour, 8
  Markovsim, 9
*Topic plot
  Markovcontour, 8
*Topic simulation
  Markovsim, 9

Markovchart, 2, 9, 11
Markovcontour, 8
Markovsim, 9
optimParallel, 4