Package ‘Markovchart’

May 8, 2021

Type Package

Title Markov Chain-Based Cost-Optimal Control Charts

Version 2.0.1

Author Balazs Dobi & Andras Zempleni

Maintainer Balazs Dobi <dobibalazs@inf.elte.hu>


License GPL

Encoding UTF-8

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

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-08 15:50:02 UTC

R topics documented:

Markovchart ................................................................. 2
Markovsim ................................................................. 7
Markovstat ................................................................. 9
plot.Markov_grid ......................................................... 12

Index 15
Markovchart

Cost-efficient X-bar control charts with fixed/random shift size, random repair and random sampling time.

Description

Wrapper for Markov chain-based cost optimal control charts. Includes cost calculation 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/critical value (k).

Usage

```r
Markovchart(statdist, h = NULL, k = NULL,
OPTIM = FALSE, p = 1, constantr = FALSE,
ooc_rep = 0, 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

- **statdist**: The stationary distribution of the Markov chain. Must be an object of class `Markov_stationary`, preferably created by `Markovstat`.
- **h**: The time between samplings. Must be a positive value, can be a numeric vector. For optimisation, this is the initial value. Inherited from `statdist` if not given.
- **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. Inherited from `statdist` if not given.
- **OPTIM**: Logical. Should the resulting G-value (weighted average of the expected cost and cost standard deviation) be optimised by finding the adequate value of h and k.
- **p**: The weight of the cost expectation in the calculation of the G-value; should be between 0 and 1.
- **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, default)? If TRUE, then the repair cost should be given per unit time.
- **ooc_rep**: Numeric value between 0 and 1. The percentage of repair cost occurring during out-of-control operation. Default is 0. If a value greater than 0 is set, then `constantr` should be TRUE, but it is not forced.
- **cs**: Sampling cost per sampling.
Markovchart

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. The default function assumes a linear relationship between the repair cost and the distance, and uses a base and a distance-scaling repair cost parameter. See "Details".

crparams Numeric vector. Parameters of crfun.

cf Numeric. The false alarm cost. Only relevant when statdist was created using shiftfun="deg".

cofun 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".

coparams Numeric vector. Parameters of cofun.

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

crparams Numeric vector. Parameters of vcrfun.

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. Can be left empty, in this case 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:

\[ c_o = c_{ob} + c_{os} A^2(v), \]

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 \( A^2(v) \) is the squared distance from the target value. This latter part is defined like
this because a Taguchi-type loss function is used. This $A^2(v)$ incorporates the distances (the base of the losses) incurred not just at the time of the sampling, but also between samplings (hence it depends on $h$). Even if the user defines a custom cost function for the OOC cost, this $A^2(v)$ 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($A^2(v)$, other parameters contained in a vector). $A^2(v)$ 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)
}
```

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 law of total 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)
}
```

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

$$c_r = c_{rb} + c_{rs}v,$$

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

Value

The value depends on the parameters:

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 of class codeMarkov_grid with length(h) x length(k) number of rows and three columns for h, k and the G-value.

If h and k are both of length 1 (they may be inherited from statdist), 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 of the process itself), 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). The "Total cost std. dev." is only relevant and calculated for non-degenerate distributions.

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. Further information about the stationary distribution can be calculated using the Markovstat function.

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.
stat_deg <- Markovstat(shiftfun="deg", h=1, k=1, sigma=1, s=0.2, delta=2.5)
res1 <- Markovchart(statdist=stat_deg, cs=1, crparams=20, coparams=50)
res1

# Fixed shift size (essentially Duncan’s cycle model) - with optimisation.
res2 <- Markovchart(statdist=stat_deg, OPTIM=TRUE, cs=1, crparams=20, coparams=50,
        lower = c(0.01,0.01), upper = c(5,5),
        parallel_opt=parall)
res2

# Exponential shift - no optimisation - default cost functions.
stat_exp <- Markovstat(shiftfun="exp", h=0.5, k=2, sigma=1, s=0.2, delta=2,
        RanRep=FALSE, Vd=30, V=18)
res3 <- Markovchart(stat_exp, p=0.9, cs=1, coparams=c(10,3), crparams=c(1,2))
res3

# Exponential shift - with optimisation - default cost functions.
stat_exp2 <- Markovstat(shiftfun="exp", h=1, k=1, sigma=1, s=0.2, delta=2,
        RanRep=TRUE, alpha=1, beta=3, Vd=30, V=18)
parall <- list(cl=makeCluster(num_workers), forward=FALSE, loginfo=TRUE)
res4 <- Markovchart(statdist=stat_exp2, OPTIM=TRUE, p=0.9, cs=1,
        coparams=c(10,3), crparams=c(1,2), vcoparams=c(8,1.5),
        vcrparams=c(5,2), parallel_opt=parall)
res4

# Exponential-geometric mixture shift - no optimisation -
# random sampling - custom repair variance function.
stat_exprgeo <- Markovstat(shiftfun="exp-geo",h=1.5, k=2, sigma=1,
        s=0.2, delta=1.2, probmix=0.7, probbin=0.8,
        disj=2, RanRep=TRUE, alpha=1, beta=3, RanSam=TRUE,
        StateDep=TRUE, a=1, b=15, Vd=100, V=8)

vcrfun_new <- function(mudist,vcparams)
{
    mudist=mudist
    vcrb=vcparams[1]
    vcrs=vcparams[2]
    vcrs2=vcparams[3]
    vcr <- vcrb + vcrs/(mudist + vcrs2)
    return(vcr)
}
```r
res5 <- Markovchart(statdist=stat_expgeo, p=0.9, cs=1,
                     coparams=c(10,6), crparams=c(20,3),
                     vcoparams=c(10000,100), vcrfun=vcrfun_new,
                     vcrparams=c(50000,-600000,1.5))
res5

#Exponential shift - no optimisation - vectorised.
parall <- list(cl=makeCluster(num_workers), forward=FALSE, loginfo=TRUE)
Gmtx <- Markovchart(statdist=stat_exp2, h=seq(1,10,by=(10-1)/5),
                    k=seq(0.1,5,by=(5-0.1)/5), p=0.9, cs=1,
                    coparams=c(10,3), crparams=c(1,2),
                    vcoparams=c(8,1.5), vcrparams=c(5,2),
                    V=18, parallel_opt=parall)
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(shiftfun = c("exp", "exp-geo"), num = 100, h, k, sigma,
          s, delta, probmix = 1, probnbin = 0.5, disj=1,
          RanRep = FALSE, alpha = NULL, beta = NULL, RanSam = FALSE,
          StateDep = FALSE, a = NULL, b = NULL, q = NULL,
          z = NULL, detail = 100, Vd = 50, V, burnin = 1)
```

**Arguments**

- `shiftfun` A string defining the shift size distribution to be used. Must be either "exp", "exp-geo".
- `num` Integer. The number of sampling intervals simulated. This means that the time elapsed in the simulation is num*h.
- `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.
probNb\ri

disj

RanRep

alpha

beta

RanSam

StateDep

a

b

q

z

detail

V_d

V

burnin

**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).

**Value**

A Markov_sim object which is a list of length 4.

Value_at_samplings

The process value at sampling.

Sampling_event

The event at sampling, each can either be success (there was a sampling but no alarm), alarm (sampling with alarm) or failure (no sampling occurred).
Simulation_data
The simulated data (distances from the target value).

Stationary_distribution
The stationary distribution of the Markov chain, created by discretising the simulated data. See the documentation of the Markovchart function.

Author(s)
Balazs Dobi and Andras Zempleni

References


See Also
Markovstat

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

#Simulation using exponential-geometric mixture shifts, random repair and random samling.
simres2 <-Markovsim(shiftfun="exp-geo", num=500, 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,
StateDep=TRUE, a=0.1, b=1, V=10)
simres2
hist(simres2[[1]], 20, freq=FALSE)
Description

Calculates the stationary distribution of a process described by a discrete state, discrete time Markov chain. The process is described by a degradation-repair cycle type model. The user must give parameters describing both the degradation and the repair. The process is not repaired until the problem is discovered by sampling, hence the control chart setup. The same, single element is monitored (i.e. the sample size is always 1).

Usage

Markovstat(shiftfun = c("exp", "exp-geo", "deg"), h, k, sigma, s, delta, probmix = 1, probnbin = 0.5, disj = 1, RanRep = FALSE, alpha = NULL, beta = NULL, RanSam = FALSE, StateDep = FALSE, a = NULL, b = NULL, q = NULL, z = NULL, Vd = 100, V, Qparam = 30)

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.

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 assumed to be 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").

probbmix The weight of the geometric distribution in case of exponential-geometric mixture shift distribution; should be between 0 and 1.

probnbin The probability parameter of the geometric distribution in case of exponential-geometric mixture shift distribution; 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 (the repair is perfect, the process is always repaired to the target value). 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".

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.)
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 as described at the StateDep parameter.

Second shape parameter for the random sampling time beta distribution.

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

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

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

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

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

The function returns a list object of class Markov_stationary. The list is of length 3:

Stationary_distribution

Stationary distribution of the Markov chain. The probabilities in the stationary distribution are labeled. If shiftfun is "deg" then the stationary distribution is always of length 4. If shiftfun is not "deg" then there are multiple out-of-control (OOC) 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, and the in-control and the false alarm state are always at the target value.

Transition_matrix

The transition matrix of the Markov chain. Not printed.

Param_list

Parameters given to the function and various technical results used by the Markovchart function. Not printed.

Balazs Dobi and Andras Zempleni


Markovchart
Examples

```r
# Fixed shift size (essentially Duncan's cycle model).
res1 <- Markovstat(shiftfun="deg", h=1, k=1, sigma=1, s=0.2, delta=2.5)
res1

# Exponential shift - perfect repair - deterministic sampling
res2 <- Markovstat(shiftfun="exp", h=1, k=1, sigma=1, s=0.2, delta=2, Vd=30, V=18)
res2
# Notice how the In-control and the False-alarm states have non-zero probabilities.
# If the repair would be random (RanRep=TRUE), then these states would have zero probability.

# Exponential-geometric mixture shift - random repair - random sampling.
res3 <- Markovstat(shiftfun="exp-geo", h=1.5, k=2, sigma=1, s=0.2,
                   delta=1.2, probmix=0.7, probnbin=0.8, disj=2,
                   RanRep=TRUE, alpha=1, beta=3, RanSam=TRUE,
                   StateDep=TRUE, a=1, b=15, Vd=40, V=8)
res3
```

---

### plot.Markov_grid

**Contour plot for Markov_grid control chart results.**

**Description**

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

**Usage**

```r
## S3 method for class 'Markov_grid'
plot(x, y = expression(atop(italic("G")*-value~per, unit~time)),
     xlab = "Time between samplings", ylab = "Critical value",
     low = "white", mid = "#999999", high = "black",
     colour = "white", nbreaks = 16, ...)
```

**Arguments**

- `x` A *Markov_grid* 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).
- `y` The name of the scale.
- `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 the midpoint.
- `high` Colour for the high end of the gradient.
**plot.Markov_grid**

- **colour**: Colour of the contour lines.
- **nbreaks**: Number of contour breaks. Uses `pretty()`, thus actual, plotted number of breaks may differ.
- **...**: Further arguments to be passed down to `plot`. Mostly kept due to S3 method compatibility.

**Value**

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

**Note**

The plot itself is made using the package `ggplot` 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.

**Author(s)**

Balazs Dobi and Andras Zempleni

**References**


**See Also**

- `Markovchart`
- `Markovstat`

**Examples**

```r
#Defining parallel_opt parallel settings.
#parallel_opt can also be left empty to be defined automatically by the function.
require(parallel)
um_workers <- min(c(detectCores(),2))

#Exponential shift - default cost functions.
stat_exp <- Markovstat(shiftfun="exp", h=1, k=1, sigma=1, s=0.2, delta=2,
                   RanRep=TRUE, alpha=1, beta=3, Vd=30, V=18)
parall <- list(cl=makeCluster(num_workers), forward=FALSE, loginfo=TRUE)
Gmtx <-Markovchart(statdist=stat_exp, h=seq(1,10,by=(10-1)/5),
                 k=seq(0.1,5,by=(5-0.1)/5), p=0.9, cs=1,
                 coparams=c(10,3), crparams=c(1,2),
                 vcoparams=c(8,1.5), vcrparams=c(5,2),
```
plot(Markov_grid)

plot(Gmtx)

V=18, parallel_opt=parall)
Index

* Markov chain
  Markovchart, 2
  Markovsim, 7
  Markovstat, 9
  plot.Markov_grid, 12
* contour
  plot.Markov_grid, 12
* control chart
  Markovchart, 2
  Markovsim, 7
  Markovstat, 9
  plot.Markov_grid, 12
* plot
  plot.Markov_grid, 12
* simulation
  Markovsim, 7
* stationary distribution
  Markovstat, 9

geom_text_contour, 13
ggplot, 13

Markovchart, 2, 9, 11, 13
Markovsim, 7
Markovstat, 5, 9, 9, 13

optimParallel, 3

plot.Markov_grid, 12