

# Package ‘WaMaSim’

May 7, 2026

**Type** Package

**Title** Simulate Rehabilitation Strategies for Water Distribution Systems

**Version** 1.0.0

**Date** 2019-05-02

**Author** Andreas Scheidegger [aut],  
Rachel Barrett [aut],  
Christian Foerster [cre]

**Maintainer** Christian Foerster <christian.foerster@eawag.ch>

**Description** The outcome of various rehabilitation strategies for water distribution systems can be modeled with the Water Management Simulator (WaMaSim). Pipe breaks and the corresponding damage and rehabilitation costs are simulated. It is mainly intended to be used as educational tool for the Water Infrastructure Experimental and Computer Laboratory at ETH Zurich, Switzerland.

**Encoding** UTF-8

**URL** <https://github.com/scheidan/WaMaSim>

**BugReports** <https://github.com/scheidan/WaMaSim/issues>

**License** GPL-3

**RoxygenNote** 6.1.1

**Depends** magrittr (>= 1.5)

**Suggests** testthat

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-05-03 15:50:13 UTC

## Contents

costs.per.year . . . . .	2
do.nothing . . . . .	3
expand . . . . .	3

fail . . . . .	4
failure.cost . . . . .	5
failures.per.year . . . . .	5
initiate.network . . . . .	6
pipes.built.per.year . . . . .	6
pipes.inservice.per.year . . . . .	7
replace.more.failures.than . . . . .	7
replace.n.highest.risk . . . . .	8
replace.n.oldest . . . . .	9
replace.n.random . . . . .	10
replace.older.than . . . . .	11
replacement.value . . . . .	12
sample.diameter . . . . .	13
simulate_network . . . . .	13
simulate_network.period . . . . .	15
WaMaSim . . . . .	18
\$.statelist . . . . .	18
<b>Index</b>	<b>19</b>

---

costs.per.year	<i>Calculate the total costs per year</i>
----------------	---

---

## Description

The annual total costs are calculated. The total costs consist of damage, failure and rehabilitation costs.

## Usage

```
costs.per.year(statelist, income)
```

## Arguments

statelist	a state list
income	the same values as passed to simulate_network. Either a scalar or vector.

## Value

a vector of the total cost per year

## Author(s)

Andreas Scheidegger

---

do.nothing	<i>Rehabilitation strategy: no pipe replacement, repairs only</i>
------------	---

---

**Description**

Dummy strategy to model no rehabilitation at all.

**Usage**

```
do.nothing(state)
```

**Arguments**

state            a state list

**Value**

a state list

**Author(s)**

Andreas Scheidegger

**See Also**

[replace.n.highest.risk](#), [replace.n.oldest](#), [replace.n.random](#), [replace.older.than](#), [replace.more.failures.than](#)

**Examples**

```
## define a strategy function that can be passed to simulate_network():  
mystrategy <- . %>% do.nothing
```

---

expand	<i>Model expansion of the network</i>
--------	---------------------------------------

---

**Description**

Expand the network with additional pipes. The diameter of these pipes is sampled.

**Usage**

```
expand(state, n.new, separate.budget = FALSE)
```



---

failure.cost	<i>Calculate the (random) cost of a failure</i>
--------------	---

---

**Description**

Calculate the costs caused by a failure according to Section 7.1 in "The Water Network Management Challenge", Max Maurer 2017.

**Usage**

```
failure.cost(diameter, mean = FALSE)
```

**Arguments**

diameter	diameter [mm]
mean	boolean. Should the expected cost be returned? Random otherwise.

**Value**

if mean=FALSE, the failure costs [CHF] are sampled randomly. If mean=TRUE, the expected average costs are returned.

**Author(s)**

Andreas Scheidegger

---

failures.per.year	<i>Calculate number of failures per year</i>
-------------------	--

---

**Description**

The number of failures per year is calculated from a state list produced by [simulate\\_network](#).

**Usage**

```
failures.per.year(statelist)
```

**Arguments**

statelist	a state list
-----------	--------------

**Value**

vector containing the number of failures per year

**Author(s)**

Andreas Scheidegger

---

<code>initiate.network</code>	<i>creates the initial network</i>
-------------------------------	------------------------------------

---

**Description**

Creates the initial network that can be used with [simulate\\_network](#).

**Usage**

```
initiate.network(inventory = 0, budget = Inf)
```

**Arguments**

<code>inventory</code>	if <code>inventory</code> is an integer it specifies the number of initial pipes, or alternatively it can be a <code>data.frame</code> containing the initial inventory of pipes.
<code>budget</code>	initial budget

**Value**

a state list

**Author(s)**

Andreas Scheidegger

---

<code>pipes.built.per.year</code>	<i>Calculate number of newly built pipes for each year</i>
-----------------------------------	--

---

**Description**

The number of newly built pipes per year is calculated from a state list produced by [simulate\\_network](#).

**Usage**

```
pipes.built.per.year(statelist)
```

**Arguments**

<code>statelist</code>	a state list
------------------------	--------------

**Value**

vector containing the number of newly built pipes for each year

**Author(s)**

Andreas Scheidegger

---

`pipes.inservice.per.year`

*Returns the number of pipes in service for each year*

---

### **Description**

The number of pipes in service is calculated for every year based on a state list produced by [simulate\\_network](#).

### **Usage**

```
pipes.inservice.per.year(statelist)
```

### **Arguments**

`statelist`      a state list

### **Value**

vector containing the number of pipes in service

### **Author(s)**

Andreas Scheidegger

---

`replace.more.failures.than`

*Rehabilitation strategy: replace pipes with too many failures*

---

### **Description**

Replace pipes with a high number of failures. Pipes are only replaced if the budget remains positive.

### **Usage**

```
replace.more.failures.than(state, failures, max.costs = Inf)
```

### **Arguments**

`state`            a state list

`failures`        maximal allowed number of failures

`max.costs`       maximal amount of money allowed to be spent on this strategy

### **Value**

a state list

**Author(s)**

Andreas Scheidegger

**See Also**

[replace.n.highest.risk](#), [replace.n.oldest](#), [replace.n.random](#), [replace.older.than](#), [do.nothing](#)

**Examples**

```
## define a strategy function that can be passed to simulate_network():
mystrategy <- . %>% replace.more.failures.than(max.failure=3, max.costs=20000)

## or define a more complex strategy by combining multiple strategies
## into a prioritized sequence:
mystrategy <- . %>%
  replace.more.failures.than(failures=2) %>%
  replace.n.oldest(n=3) %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.older.than(age=8) %>%
  replace.n.random(n=4)
```

---

replace.n.highest.risk

*Rehabilitation strategy: replace the n pipes with the highest risk*

---

**Description**

Strategy to prioritize pipes with the highest risk. Pipes are only replaced if the budget remains positive.

**Usage**

```
replace.n.highest.risk(state, n, prob.failure, max.costs = Inf)
```

**Arguments**

state	a state list
n	number of highest risk pipes to replace
prob.failure	failure rate function. Typically the same as passed to <a href="#">simulate_network</a> .
max.costs	maximal amount of money allowed to be spent on this strategy

**Details**

The risk is defined as the product of the failure probability in the next year and the expected failure costs.

**Value**

a state list

**Author(s)**

Andreas Scheidegger

**See Also**

[replace.n.oldest](#), [replace.n.random](#), [replace.older.than](#), [replace.more.failures.than](#), [do.nothing](#)

**Examples**

```
## define a strategy function that can be passed to simulate_network():
mystrategy <- . %>% replace.n.highest.risk(n=2, prob.failure=prob.failure.exp, max.costs=30000)

## or define a more complex strategy by combining multiple strategies
## into a prioritized sequence:
mystrategy <- . %>%
  replace.more.failures.than(failures=2) %>%
  replace.n.oldest(n=3) %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.older.than(age=8) %>%
  replace.n.random(n=4)
```

---

replace.n.oldest	<i>Rehabilitation strategy: replace the n oldest pipes</i>
------------------	--

---

**Description**

Prioritize the oldest pipes for replacement. Pipes are only replaced if the budget remains positive.

**Usage**

```
replace.n.oldest(state, n, max.costs = Inf)
```

**Arguments**

state	a state list
n	number of oldest pipes to replace
max.costs	maximal amount of money allowed to be spent on this strategy

**Value**

a state list

**Author(s)**

Andreas Scheidegger

**See Also**

[replace.n.highest.risk](#), [replace.n.random](#), [replace.older.than](#), [replace.more.failures.than](#), [do.nothing](#)

**Examples**

```
## define a strategy function that can be passed to simulate_network():
mystrategy <- . %>% replace.n.oldest(n=10)

## or define a more complex strategy by combining multiple strategies
## into a prioritized sequence:
mystrategy <- . %>%
  replace.more.failures.than(failures=2) %>%
  replace.n.oldest(n=3) %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.older.than(age=8) %>%
  replace.n.random(n=4)
```

---

replace.n.random	<i>Rehabilitation strategy: replace n randomly selected pipes</i>
------------------	---

---

**Description**

Replace a certain number of randomly chosen pipes. Pipes are only replaced if the budget remains positive.

**Usage**

```
replace.n.random(state, n, max.costs = Inf)
```

**Arguments**

state	a state list
n	number of random pipes to replace
max.costs	maximal amount of money allowed to be spent on this strategy

**Value**

a state list

**Author(s)**

Andreas Scheidegger

**See Also**

[replace.n.highest.risk](#), [replace.n.oldest](#), [replace.older.than](#), [replace.more.failures.than](#), [do.nothing](#)

**Examples**

```
## define a strategy function that can be passed to simulate_network():
mystrategy <- . %>% replace.n.random(n=10)

## or define a more complex strategy by combining multiple strategies
## into a prioritized sequence:
mystrategy <- . %>%
  replace.more.failures.than(failures=2) %>%
  replace.n.oldest(n=3) %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.older.than(age=8) %>%
  replace.n.random(n=4)
```

---

replace.older.than	<i>Rehabilitation strategy: replace pipes older than age</i>
--------------------	--

---

**Description**

Strategy to replace pipes older than a given age. Pipes are only replaced if the budget remains positive.

**Usage**

```
replace.older.than(state, age, max.costs = Inf)
```

**Arguments**

state	a state list
age	pipes older than age are replaced
max.costs	maximal amount of money allowed to be spent on this strategy

**Value**

a state list

**Author(s)**

Andreas Scheidegger

**See Also**

[replace.n.highest.risk](#), [replace.n.oldest](#), [replace.n.random](#), [replace.more.failures.than](#), [do.nothing](#)

## Examples

```
## define a strategy function that can be passed to simulate_network():
mystrategy <- . %>% replace.older.than(age=85, max.costs=20000)

## or define a more complex strategy by combining multiple strategies
## into a prioritized sequence:
mystrategy <- . %>%
  replace.more.failures.than(failures=2) %>%
  replace.n.oldest(n=3) %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.older.than(age=8) %>%
  replace.n.random(n=4)
```

---

replacement.value	<i>Calculate replacement value</i>
-------------------	------------------------------------

---

## Description

Based on Eq(14) of "The Water Network Management Challenge, Max Maurer 2017", assuming a pipe length of 100m.

## Usage

```
replacement.value(diameter)
```

## Arguments

diameter          diameter of the pipe [mm]

## Value

replacement value [CHF]

## Author(s)

Andreas Scheidegger

---

sample.diameter	<i>Sample a diameter of a new pipe.</i>
-----------------	---

---

**Description**

The diameter distribution is based on a real data set from Lisa Scholten (pipe\_data.csv). Refer to the old exercise on watermain break modelling in the ETH Infrastructure Systems course by Max Maurer.

**Usage**

```
sample.diameter(n = 1)
```

**Arguments**

n	number of samples
---	-------------------

**Value**

a vector of diameters

**Author(s)**

Andreas Scheidegger

---

simulate_network	<i>Simulate the failures, expansion, rehabilitation, and costs of a network</i>
------------------	---

---

**Description**

Simulates failures, expansion, rehabilitation, and costs of a water supply pipe network. The simulation is stochastic.

**Usage**

```
simulate_network(n.years, expansion, rehabilitation, prob.failure,  
  income = 0, initial.budget = Inf, initial.inventory = NULL,  
  free.expansion = TRUE)
```

**Arguments**

n.years	number of years to simulate
expansion	either a scalar describing the number of pipes added every year to expand the pipe network, or a vector of length n.years. Negative values are not allowed.
rehabilitation	a (combination of) rehabilitation strategy function(s). See details below.
prob.failure	a function describing the probability of a pipe failing in the next year given its age, number of previous failures, and the age at the last failure (if any).
income	either a scalar describing the annual income, or a vector of length n.years.
initial.budget	initial budget
initial.inventory	if it is an integer it specifies the number of initial pipes, or alternatively it can be a data.frame containing the initial inventory of pipes.
free.expansion	if TRUE costs for network expansion are not deducted from the budget.

**Details**

The rehabilitation is defined by combining different simple replacement strategies. See the example for how this can be done using the `mystrategy` function input. If the strategies vary over time, see [initiate.network](#) and [simulate\\_network.period](#).

The failure behavior is defined by the function `prob.failure`. It calculates the probability of a pipe failing within the next year based on pipe age, pipe age at the last failure, and the number of failures. Note, the model makes the assumption that a pipe cannot fail more than once per year.

The costs are calculated as a function of the pipe diameter, assuming all pipes have a length of 100 meters.

**Value**

an updated state list

**Author(s)**

Andreas Scheidegger

**See Also**

For more fine-grained control see [initiate.network](#) and [simulate\\_network.period](#). Different replacement strategies are implemented: [replace.n.highest.risk](#), [replace.n.oldest](#), [replace.n.random](#), [replace.older.than](#), [replace.more.failures.than](#), [do.nothing](#).

**Examples**

```
## -----
## define model parameters

## Define the annual probability of a failure
prob.failure.exp <- function(age, age.last.failure, n.failure) {
  if(n.failure==0){
```

```

    return(1/30)
  } else {
    return(1/10)
  }
}

## define a complicated (and probably useless) rehabilitation strategy
mystrategy <- . %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.more.failures.than(failures=5) %>%
  replace.older.than(age=70, max.cost=2e6) %>%
  replace.n.oldest(n=3) %>%
  replace.n.random(n=2)
## This means: every year (if we have enough budget!), replace first the 2 pipes
## with the highest risk, then all pipes with more than 5 failures,
## then all pipes older than 70 years (up to costs of 2e6), then the 3
## oldest pipes remaining, and finally replace 2 randomly selected pipes.

## -----
## run the simulation

result <- simulate_network(

  n.years = 100,           # run it for 100 years
  expansion = 10,         # build 10 pipes per year (if money is available)
  rehabilitation = mystrategy, # use the strategy defined above
  prob.failure = prob.failure.exp, # use the probability function defined above
  income = 1e6,          # the annual income
  initial.budget = 1e7,
  initial.inventory = 50, # start the simulation with 50 new pipes
  free.expansion = FALSE

)

## look at some results
## str(result)
## str(result$time.100)

```

---

```
simulate_network.period
```

*Simulate the network for a period of time*

---

### Description

Simulates failures, expansion, rehabilitation, and costs of a water supply pipe network. The simulation is stochastic.

**Usage**

```
simulate_network.period(statelist, n.years, expansion, rehabilitation,
  prob.failure, income = 0, free.expansion = TRUE)
```

**Arguments**

statelist	a state list
n.years	number of years to simulate
expansion	either a scalar describing the number of pipes added every year to expand the pipe network, or a vector of length n.years. Negative values are not allowed.
rehabilitation	a (combination of) rehabilitation strategy function(s). See details below.
prob.failure	a function describing the probability of a pipe failing in the next year given its age, number of previous failures, and the age at the last failure (if any).
income	either a scalar describing the annual income, or a vector of length n.years.
free.expansion	if TRUE costs for network expansion are not deducted from the budget.

**Details**

The rehabilitation is defined by combining different simple replacement strategies. See the example for how this can be done using the `mystrategy` function input.

The failure behavior is defined by the function `prob.failure`. It calculates the probability of a pipe failing within the next year based on pipe age, pipe age at the last failure, and the number of failures. Note, the model makes the assumption that a pipe cannot fail more than once per year.

The costs are calculated as a function of the pipe diameter, assuming all pipes have a length of 100 meters.

**Value**

an updated state list

**Author(s)**

Andreas Scheidegger

**See Also**

[simulate\\_network](#) provides a slightly more convenient interface.

**Examples**

```
## -----
## define model parameters

## Define the annual probability of a failure
prob.failure.exp <- function(age, age.last.failure, n.failure) {
  if(n.failure==0){
    return(1/30)
  }
}
```

```

    } else {
      return(1/10)
    }
  }

## define a complicated (and probably useless) rehabilitation strategy
mystrategy <- . %>%
  replace.n.highest.risk(n=2, prob.failure=prob.failure.exp) %>%
  replace.more.failures.than(failures=5) %>%
  replace.older.than(age=70, max.cost=2e6) %>%
  replace.n.oldest(n=3) %>%
  replace.n.random(n=2)
## This means: every year (if we have enough budget!), replace first the 2 pipes
## with the highest risk, then all pipes with more than 5 failures,
## then all pipes older than 70 years (up to costs of 2e6), then the 3
## oldest pipes remaining, and finally replace 2 randomly selected pipes.

## -----
## run the simulation in steps

statelist <- initiate.network(inventory = 50, budget = 1e7)

statelist <- simulate_network.period(

  statelist,          # state list to update
  n.years = 20,      # run it for 20 years
  expansion = 10,    # build 10 pipes per year (if money is available)
  rehabilitation = mystrategy, # use the strategy defined above
  prob.failure = prob.failure.exp, # use the probability function defined above
  income = 1e6       # the annual income

)

statelist <- simulate_network.period(

  statelist,          # state list to update
  n.years = 10,      # run it for 10 more years
  expansion = 2,      # now, build only 2 pipes per year (if money is available)
  rehabilitation = mystrategy, # use the strategy defined above
  prob.failure = prob.failure.exp, # use the probability function defined above
  income = 1e6       # the annual income

)

## look at some results
## str(statelist)
## str(statelist$time.30)

```

---

WaMaSim	<i>WaMaSim - Water Management Simulator</i>
---------	---

---

**Description**

WaMaSim is a package that simulates the effect of different rehabilitation strategies for water distribution systems. It is an education tool used for the Water Infrastructure Experimental and Computer Laboratory at ETH Zurich, Switzerland. See the documentation for [simulate\\_network](#) to get started.

**Author(s)**

Andreas Scheidegger

---

\$.statelist	<i>Extract time or budget as vectors</i>
--------------	--

---

**Description**

Convenient functions to extract the time or budget.

**Usage**

```
## S3 method for class 'statelist'
x$name
```

**Arguments**

x	a state list
name	name of the element to extract

**Value**

a vector of the time or budget, or a state

**Author(s)**

Andreas Scheidegger

**Examples**

```
## Not run:
str(result)      # result is a 'statelist' returned from simulate_network
result$budget    # vector of budget
result$time      # vector of time
result$time.22   # state list of time 22

## End(Not run)
```

# Index

`$.statelist`, 18

`costs.per.year`, 2

`do.nothing`, 3, 8–11, 14

`expand`, 3

`fail`, 4

`failure.cost`, 5

`failures.per.year`, 5

`initiate.network`, 6, 14

`pipes.built.per.year`, 6

`pipes.inservice.per.year`, 7

`replace.more.failures.than`, 3, 7, 9–11, 14

`replace.n.highest.risk`, 3, 8, 8, 10, 11, 14

`replace.n.oldest`, 3, 8, 9, 9, 11, 14

`replace.n.random`, 3, 8–10, 10, 11, 14

`replace.older.than`, 3, 8–11, 11, 14

`replacement.value`, 12

`sample.diameter`, 13

`simulate_network`, 5–8, 13, 16, 18

`simulate_network.period`, 14, 15

WaMaSim, 18

WaMaSim-package (WaMaSim), 18