Package 'reservoir'

July 23, 2015

Type Package

Title Tools for Analysis, Design, and Operation of Water Supply Storages

Version 1.0.0

Date 2015-07-23

URL https://github.com/swd-turner/reservoir

Description Measure single-storage water supply system performance using resilience, reliability, and vulnerability metrics; assess storage-yield-reliability relationships; determine no-fail storage with sequent peak analysis; optimize release decisions using deterministic and stochastic dynamic programming; evaluate inflow persistence using the Hurst coefficient.

License GPL (>= 2)

LazyData yes

Imports gtools, stats

NeedsCompilation no

Author Sean Turner [aut, cre], Stefano Galelli [aut]

Maintainer Sean Turner <swd.turner@gmail.com>

Repository CRAN

Date/Publication 2015-07-23 07:16:21

R topics documented:

dp *Dynamic Programming*

Description

Determines the optimal sequence of releases from the reservoir to minimise a penalty cost function based on water supply defict.

Usage

```
dp(Q, capacity, target, S_disc = 1000, R_disc = 10, loss_exp = 2,
  S_initial = 1, plot = TRUE, rep_rrv = FALSE)
```

Arguments

`Q`	vector or time series object. Net inflows to the reservoir.
`capacity`	numerical. The reservoir storage capacity (must be the same volumetric unit as Q and the target release).
`target`	numerical. The target release constant.
`S_disc`	integer. Storage discretization–the number of equally-sized storage states. Default = 1000.
`R_disc`	integer. Release discretization. Default = 10 divisions.
`loss_exp`	numeric. The exponent of the penalty cost function–i.e., Cost[t] <- ((target - release[t]) / target) ^ **loss_exp**). Default value is 2.
`S_initial`	numeric. The initial storage as a ratio of capacity (0 <= S_initial <= 1). The default value is 1.
`plot`	logical. If TRUE (the default) the storage behavior diagram and release time series are plotted.
`rep_rrv`	logical. If TRUE then reliability, resilience and vulnerability metrics are computed and returned.

Value

Returns the time series of optimal releases and, if requested, the reliability, resilience and vulnerability of the system.

References

Loucks, D.P., van Beek, E., Stedinger, J.R., Dijkman, J.P.M. and Villars, M.T. (2005) Water resources systems planning and management: An introduction to methods, models and applications. Unesco publishing, Paris, France.

See Also

sdp for Stochastic Dynamic Programming

Examples

```
storage_cap <- 4 * mean(aggregate(ResX_inflow.ts)) # set storage ratio of 4 years
demand <- 0.8 * mean(ResX_inflow.ts) # set draft ratio of 0.8
optimal.releases <- dp(ResX_inflow.ts, capacity = storage_cap, target = demand)
```

Hurst *Hurst coefficient estimation*

Description

Hurst coefficient estimation.

Usage

```
Hurst(Q)
```

Arguments

Q	vector or annualized time series object. Net inflows or streamflow totals.

Value

Returns an estimate of the Hurst coefficient, H ($0.5 < H < 1$).

References

H.E.Hurst (1951) Long-term storage capacity of reservoirs, Transactions of the American Society of Civil Engineers 116, 770-808.

Pfaff, B. (2008) Analysis of integrated and cointegrated time series with R, Springer, New York. [p.68]

Examples

```
Q_annual <- aggregate(ResX_inflow.ts) #convert monthly to annual data
Hurst(Q_annual)
```

reservoir *reservoir: Tools for Analysis, Design, and Operation of Water Supply Storages*

Description

Measure single reservoir performance using resilience, reliability, and vulnerability metrics; compute storage-yield-reliability relationships; determine no-fail Rippl storage with sequent peak analysis; optimize release decisions using determinisitc and stochastic dynamic programming; evaluate inflow characteristics.

Analysis and design functions

The `Rippl` function executes the sequent peak algorithm to determine the no-fail storage for given inflow and release time series. The `storage` function gives the design storage for a specified time-based reliability and yield. Similarly, the `yield` function computes yield given the storage capacity. The `rrv` function returns three reliability measures, relilience, and dimensionless vulnerability for given storage, inflow time series, and target release. Users can assume Standard Operating Policy, or can apply the output of `sdp` analysis to determine the RRV metrics under different operating objectives. The `Hurst` function estimates the Hurst coefficient for an annualized inflow time series.

Optimization functions

Users may specify a loss exponent parameter for supply deficits and then optimize reservoir release decisions using Dynamic Programming (`dp`) or Stochastic Dynamic Programming (`sdp`). There is also an option to simulate the output of `sdp` using the `rrv` function to validate the policy under alternative inflows or analyze reservoir performance under different operating objectives.

Examples

```
# 1. Express the distribution of Rippl storage for a known inflow process...

# a) Assume the inflow process follows a lognormal distribution
# (meanlog = 0, sdlog = 1):
x <- rlnorm(1200)

# b) Convert to a monthly time series object beginning Jan 1900
# (ts objects are not required for most functions in "reservoir")
x <- ts(x, start = c(1900, 1), frequency = 12)

# c) Begin reservoir analysis... e.g., compute the Rippl storage
x_Rippl <- Rippl(x, R = rep(mean(x) * 0.7, length = length(x)))
no_fail_storage <- x_Rippl$Rippl_storage

# d) Resample x and loop the procedure multiple times to get the
# distribution of no-failure storage for the inflow process assuming
# constant release (R) equal to 70 percent of the mean inflow.
no_fail_storage <- vector("numeric", 500)
for (i in 1:length(no_fail_storage)){
```

```
  x <- ts(rlnorm(1200), start = c(1900, 1), frequency = 12)
  no_fail_storage[i] <- Rippl(x, R = rep(mean(x) * 0.7,
  length = length(x)),plot = FALSE)$Rippl_storage
}
hist(no_fail_storage)


# 2. Trade off between annual reliability and vulnerability for a given system...

# a) Define the system: inflow time series, storage, and target release.
inflow_ts <- ResX_inflow.ts
storage_cap <- 2 * mean(aggregate(inflow_ts)) #Storage ratio = 2
demand <- 0.8 * mean(inflow_ts)

# b) define range of loss exponents to control preference of high reliability
# (low loss exponent) or low vulnerability (high loss exponent).
loss_exponents <- c(0.5, 0.75, 0.9, 1.0, 1.1, 1.25, 1.5, 1.75, 2)

# c) set up results table
pareto_results <- data.frame(matrix(ncol = 2, nrow = length(loss_exponents)))
names(pareto_results) <- c("reliability", "vulnerability")
row.names(pareto_results) <- loss_exponents

# d) loop the sdp function through all loss exponents and plot results
for (loss_f in loss_exponents){
 sdp_temp <- sdp(inflow_ts, capacity = storage_cap, target = demand, rep_rrv = TRUE,
 S_disc = 200, R_disc = 20, plot = FALSE, loss_exp = loss_f)
 pareto_results$reliability[which(row.names(pareto_results)==loss_f)] <- sdp_temp$annual_reliability
 pareto_results$vulnerability[which(row.names(pareto_results)==loss_f)] <- sdp_temp$vulnerability
 }
plot (pareto_results$reliability,pareto_results$vulnerability, type = "b", lty = 3)
```

ResX_inflow.ts *Reservoir X inflow time series 1964 - 2014 (monthly)*

Description

Reservoir X inflow time series 1964 - 2014 (monthly)

Format

time series object

Author(s)

Sean Turner

Source

www.bom.gov.au/water/hrs/

Examples

```
plot(ResX_inflow.ts)
```

`Rippl` *Rippl analysis*

Description

Computes the Rippl no-failure storage for given time series of inflows and releases using the sequent peak algorith.

Usage

```
Rippl(Q, R, double_cycle = FALSE, plot = TRUE)
```

Arguments

`Q`	a time series or vector of net inflows to the reservoir (volumetric).
`R`	a time series or vector of target releases (volumetric). Must be the same length as Q.
`double_cycle`	logical. If TRUE the Q and R time series will be replicated and placed end-to-end to double the simulation. Recommended if the critical period occurs at the end of the sequence.
`plot`	logical. If TRUE (the default) the storage behavior diagram is plotted.

Value

Returns the no-fail storage capacity and corresponding storage behaviour time series.

References

Rippl, W. (1883) The capacity of storage reservoirs for water supply, In Proceedings of the Institute of Civil Engineers, 71, 270-278.

Thomas H.A., Burden R.P. (1963) Operations research in water quality management. Harvard Water Resources Group, Cambridge

Loucks, D.P., van Beek, E., Stedinger, J.R., Dijkman, J.P.M. and Villars, M.T. (2005) Water resources systems planning and management: An introduction to methods, models and applications. Unesco publishing, Paris, France.

Examples

```
# define a release vector for a constant release equal to 70 % of the mean inflow
release <- rep(mean(ResX_inflow.ts) * 0.7, length(ResX_inflow.ts))
no_fail_storage <- Rippl(ResX_inflow.ts,release)
```

rrv *Reliability, resilience, and vulnerability analysis*

Description

Computes time-based, annual, and volumetric reliability, as well as resilience and dimensionless vulnerability for a single reservoir.

Usage

```
rrv(Q, R_target, capacity, double_cycle = FALSE, plot = TRUE,
  S_initial = 1, policy = NULL)
```

Arguments

Q	time series or vector. The net inflows to the reservoir.
R_target	time series or vector. The target release. Must be the same length as Q.
capacity	numerical. The reservoir capacity. Should be same volumetric unit as Q and R.
double_cycle	logical. If TRUE the input series will be replicated and placed end-to-end to double the simulation. (Recommended if the critical period occurs at the end of the recorded inflow time series)
plot	logical. If TRUE (the default) the storage behavior diagram and release time series are plotted.
S_initial	numeric. The initial storage as a ratio of capacity (0 <= S_initial <= 1). The default value is 1.
policy	list. The output of the SDP function. The default is (NULL) is Standard Operating Policy.

Value

Returns reliability, resilience and vulnerability metrics based on supply deficits.

References

McMahon, T.A., Adeloye, A.J., Zhou, S.L. (2006) Understanding performance measures of reservoirs, Journal of Hydrology 324 (359-382)

Examples

```
# Determine the reliability, resilience and vulnerability for reservoir on Holland Creek
demand <- rep(0.8 * mean(ResX_inflow.ts), length = length(ResX_inflow.ts))
storage_cap <- 2*mean(aggregate(ResX_inflow.ts))  # 2 years' storage
rrv(ResX_inflow.ts, R_target = demand, capacity = storage_cap)
```

sdp *Stochastic Dynamic Programming*

Description

Derives the optimal release policy based on storage state, inflow class and within-year period.

Usage

```
sdp(Q, capacity, target, S_disc = 1000, R_disc = 10, Q_disc = c(0, 0.2375,
  0.475, 0.7125, 0.95, 1), loss_exp = 2, S_initial = 1, plot = TRUE,
  tol = 0.99, rep_rrv = FALSE)
```

Arguments

Q	time series object. Net inflows to the reservoir.
capacity	numerical. The reservoir storage capacity (must be the same volumetric unit as Q and the target release).
target	numerical. The target release constant.
S_disc	integer. Storage discretization–the number of equally-sized storage states. Default = 1000.
R_disc	integer. Release discretization. Default = 10 divisions.
Q_disc	vector. Inflow discretization bounding quantiles. Defaults to five inflow classes bounded by quantile vector c(0.0, 0.2375, 0.4750, 0.7125, 0.95, 1.0).
loss_exp	numeric. The exponent of the penalty cost function–i.e., Cost[t] <- ((target - release[t]) / target) ^ **loss_exp**). Default value is 2.
S_initial	numeric. The initial storage as a ratio of capacity (0 <= S_initial <= 1). The default value is 1.
plot	logical. If TRUE (the default) the storage behavior diagram and release time series are plotted.
tol	numerical. The tolerance for policy convergence. The default value is 0.990.
rep_rrv	logical. If TRUE then reliability, resilience and vulnerability metrics are computed and returned.

Value

Returns a list that includes: the optimal policy as an array of release decisions dependent on storage state, month/season, and current-period inflow class; the Bellman cost function based on storage state, month/season, and inflow class; the optimized release and storage time series through the training inflow data; the flow discretization (which is required if the output is to be implemented in the rrv function); and, if requested, the reliability, resilience, and vulnerability of the system under the optimized policy.

References

Loucks, D.P., van Beek, E., Stedinger, J.R., Dijkman, J.P.M. and Villars, M.T. (2005) Water resources systems planning and management: An introduction to methods, models and applications. Unesco publishing, Paris, France.

Gregory R. Warnes, Ben Bolker and Thomas Lumley (2014). gtools: Various R programming tools. R package version 3.4.1. http://CRAN.R-project.org/package=gtools

See Also

sdp for deterministic Dynamic Programming

Examples

```
storage_cap <- 4 * mean(aggregate(ResX_inflow.ts)) # set storage ratio of 4 years
demand <- 0.8 * mean(ResX_inflow.ts) # set draft ratio of 0.8
optimal.releases <- sdp(ResX_inflow.ts, capacity = storage_cap, target = demand)
```

---

storage *Storage-Reliability-Yield (SRY) relationships: Storage computation*

Description

Returns the required storage for given inflow time series, yield, and target time-based reliability. Assumes standard operating policy. Storage is computed iteratively using the bi-section method.

Usage

```
storage(Q, yield, reliability, profile = rep(1, frequency(Q)), plot = TRUE,
  S_initial = 1, max.iterations = 50, double_cycle = FALSE)
```

Arguments

Q	the net inflows to the reservoir. This must be a time series object or vector of the net inflow volumes.
yield	numerical. (must be same volumetric unit as Q and R).
reliability	numerical. (must be same volumetric unit as Q and R).
profile	a vector of factors with length = frequency(Q). Represents within-year demand profile. Defaults to constant release if left blank.
plot	logical. If TRUE (the default) the storage behavior diagram and release time series are plotted.
S_initial	numeric. The initial storage as a ratio of capacity (0 <= S_initial <= 1). The default value is 1.
max.iterations	Maximum number of iterations for yield computation.
double_cycle	logical. If TRUE the input series will be replicated and placed end-to-end to double the simulation. (Recommended if the critical period occurs at the end of the recorded inflow time series)

Value

Returns the required storage capacity necessary to supply specified yield with specified reliability.

Examples

```
# Determine the required storage for 95 % reliability and yield equal to 80 % of the mean inflow.
storage(ResX_inflow.ts, yield = 0.8*mean(ResX_inflow.ts), reliability = 0.95)
```

`yield` *Storage-Reliability-Yield (SRY) relationships: Yield computation*

Description

Returns the yield for given inflow time series, reservoir capacity, and required time-based reliability. Assumes standard operating policy. Yield is computed iteratively using the bi-section method.

Usage

```
yield(Q, capacity, reliability, profile = rep(1, frequency(Q)), plot = TRUE,
  S_initial = 1, max.iterations = 50, double_cycle = FALSE)
```

Arguments

`Q`	a time series or vector of net inflows to the reservoir.
`capacity`	numerical. (must be same volumetric unit as Q and R).
`reliability`	numerical. (must be same volumetric unit as Q and R).
`profile`	a vector of factors with length = frequency(Q). Represents within-year demand profile. Defaults to constant release if left blank.
`plot`	logical. If TRUE (the default) the storage behavior diagram and release time series are plotted.
`S_initial`	numeric. The initial storage as a ratio of capacity (0 <= S_initial <= 1). The default value is 1.
`max.iterations`	Maximum number of iterations for yield computation.
`double_cycle`	logical. If TRUE the input series will be replicated and placed end-to-end to double the simulation. (Recommended if the critical period occurs at the end of the recorded inflow time series)

Value

Returns the storage behaviour time series for the no-failure (Rippl) reservoir given net inflows Q and target release R.

Examples

```
# Compute yield for 0.95 reliability
yield_ResX <- yield(ResX_inflow.ts, capacity = 100000, reliability = 0.95)
# Compute yield for quarterly time series with seasonal demand profile

quarterly.ts <- aggregate(ResX_inflow.ts, nfrequency = 4)
yield_ResX.quart <- yield(quarterly.ts,
capacity = 100000, reliability = 0.9, profile = c(0.8, 1.2, 1.2, 0.8))
```

Index