Package ‘maq’

July 22, 2025

Title Multi-Armed Qini

Version 0.6.0

Description Policy evaluation using generalized Qini curves: Evaluate data-driven treatment targeting rules for one or more treatment arms over different budget constraints in experimental or observational settings under unconfoundedness.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp

Imports Rcpp

Depends R (>= 3.5.0)

Suggests grf (>= 2.3.0), ggplot2, testthat (>= 3.0.0)

SystemRequirements GNU make

URL `https://github.com/grf-labs/maq`

BugReports `https://github.com/grf-labs/maq/issues`

NeedsCompilation yes

Author Erik Sverdrup [aut, cre],
Han Wu [aut],
Susan Athey [aut],
Stefan Wager [aut]

Maintainer Erik Sverdrup <`erik.sverdrup@monash.edu`>

Repository CRAN

Date/Publication 2025-04-14 11:40:02 UTC

Contents

average_gain 2
difference_gain 2
get_aipw_scores 3

get_ipw_scores 4
integrated_difference 6
maq 6
plot.maq 10
predict.maq 11
print.maq 13
scale_maq 13
summary.maq 14

Index **15**

`average_gain` *Get estimate of gain given a spend level.*

Description

Get an estimate of Q(B).

Usage

```
average_gain(object, spend)
```

Arguments

`object`	A maq object.
`spend`	The spend level B.

Value

An estimate of Q(B) along with standard errors.

`difference_gain` *Get estimate of difference in gain given a spend level with paired standard errors.*

Description

Given two Qini curves, Q_a and Q_b, get an estimate of the difference $Q_a(B) - Q_b(B)$, at a spend level B.

Usage

```
difference_gain(object.lhs, object.rhs, spend)
```

Arguments

`object.lhs`	A maq object Q_a to subtract from.
`object.rhs`	A maq object Q_b to subtract with.
`spend`	The spend level B.

Value

An estimate of difference in gain along with standard errors.

`get_aipw_scores` *Construct evaluation scores via augmented inverse-propensity weighting.*

Description

A simple convenience function to construct an AIPW-based evaluation score given estimates of conditional means and treatment propensities.

Usage

```
get_aipw_scores(Y, W, mu.hat, W.hat = NULL)
```

Arguments

`Y`	The observed outcome.
`W`	The observed treatment assignment (must be a factor vector, where the first factor level is the control arm).
`mu.hat`	A matrix of conditional mean estimates for each arm, $E[Y_i \vert W_i = k, X_i]$.
`W.hat`	Optional treatment propensities. If these vary by unit and arm, then this should be a matrix with the treatment assignment probability of units to arms, with columns corresponding to the levels of `W`. If these only vary by arm, a vector can also be supplied. If W.hat is NULL (Default), then the assignment probabilities are assumed to be uniform and the same for each arm.

Value

An $n \cdot K$ matrix of evaluation scores (eqn (13) in the multi-armed Qini paper).

References

Robins, James M, Andrea Rotnitzky, and Lue Ping Zhao. "Estimation of regression coefficients when some regressors are not always observed." Journal of the American statistical Association, 89(427), 1994.

Sverdrup, Erik, Han Wu, Susan Athey, and Stefan Wager. "Qini Curves for Multi-Armed Treatment Rules". Journal of Computational and Graphical Statistics, forthcoming.

Examples

```
if (require("grf", quietly = TRUE)) {
# Simulate data with two treatment arms (k = 1, 2) and a control arm (k = 0).
n <- 3000
p <- 5
X <- matrix(runif(n * p), n, p)
W <- as.factor(sample(c("0", "1", "2"), n, replace = TRUE))
Y <- X[, 1] + X[, 2] * (W == "1") + 1.5 * X[, 3] * (W == "2") + rnorm(n)

# Fit a CATE estimator on a training sample.
train <- sample(1:n, n/2)
tau.forest <- grf::multi_arm_causal_forest(X[train, ], Y[train], W[train])

# Predict CATEs on held out evaluation data.
test <- -train
tau.hat <- predict(tau.forest, X[test, ], drop = TRUE)$predictions
# Form costs.
cost <- cbind(X[test, 4] / 4, X[test, 5])

# Estimate nuisance components for test set AIPW scores.
X.test <- X[test, ]
Y.test <- Y[test]
W.test <- W[test]

# Fit models for E[Y | W = k, X], k = 0, 1, 2, using for example separate random forests.
Y0.forest <- grf::regression_forest(X.test[W.test == 0, ], Y.test[W.test == 0])
Y1.forest <- grf::regression_forest(X.test[W.test == 1, ], Y.test[W.test == 1])
Y2.forest <- grf::regression_forest(X.test[W.test == 2, ], Y.test[W.test == 2])
mu.hat = cbind(
   mu0 = predict(Y0.forest, X.test)$predictions,
   mu1 = predict(Y1.forest, X.test)$predictions,
   mu2 = predict(Y2.forest, X.test)$predictions
)

# If unknown, estimate the propensity scores E[W = k | X].
W.hat <- predict(grf::probability_forest(X.test, W.test))$predictions

# Form doubly robust scores.
DR.scores <- get_aipw_scores(Y.test, W.test, mu.hat, W.hat)

# Fit a Qini curve estimated with forest-based AIPW.
qini <- maq(tau.hat, cost, DR.scores, R = 200)
plot(qini)
}
```

`get_ipw_scores` *Construct evaluation scores via inverse-propensity weighting.*

Description

A simple convenience function to construct an evaluation score matrix via IPW, where entry (i, k) equals

$$\frac{\mathbf{1}(W_i = k)Y_i}{P[W_i = k|X_i]} - \frac{\mathbf{1}(W_i = 0)Y_i}{P[W_i = 0|X_i]},$$

where W_i is the treatment assignment of unit i and Y_i the observed outcome. $k = 1 \ldots K$ are one of K treatment arms and k = 0 is the control arm.

Usage

```
get_ipw_scores(Y, W, W.hat = NULL)
```

Arguments

`Y`	The observed outcome.
`W`	The observed treatment assignment (must be a factor vector, where the first factor level is the control arm).
`W.hat`	Optional treatment propensities. If these vary by unit and arm, then this should be a matrix with the treatment assignment probability of units to arms, with columns corresponding to the levels of `W`. If these only vary by arm, a vector can also be supplied. If W.hat is NULL (Default), then the assignment probabilities are assumed to be uniform and the same for each arm.

Value

An $n \cdot K$ matrix of evaluation scores.

Examples

```
# Draw some equally likely samples from control arm A and treatment arms B and C.
n <- 5000
W <- as.factor(sample(c("A", "B", "C"), n, replace = TRUE))
Y <- 42 * (W == "B") - 42 * (W == "C") + rnorm(n)
IPW.scores <- get_ipw_scores(Y, W)
# An IPW-based estimate of E[Y(B) - Y(A)] and E[Y(C) - Y(A)]. Should be approx 42 and -42.
colMeans(IPW.scores)

# Draw non-uniformly from the different arms.
W.hat <- c(0.2, 0.2, 0.6)
W <- as.factor(sample(c("A", "B", "C"), n, replace = TRUE, prob = W.hat))
Y <- 42 * (W == "B") - 42 * (W == "C") + rnorm(n)
IPW.scores <- get_ipw_scores(Y, W, W.hat = W.hat)
# Should still be approx 42 and -42.
colMeans(IPW.scores)
```

`integrated_difference` *Get estimate of the area between two Qini curves with paired standard errors.*

Description

Given two Qini curves, Q_a and Q_b, and a maximum spend $\bar{B}$, get an estimate of the integrated difference $\int_0^{\bar{B}} (Q_a(B) - Q_b(B))dB$.

Usage

```
integrated_difference(object.lhs, object.rhs, spend)
```

Arguments

`object.lhs`	A maq object Q_a to subtract from.
`object.rhs`	A maq object Q_b to subtract with.
`spend`	The spend level $\bar{B}$.

Value

An estimate of the area between the two curves along with standard errors.

`maq` *Fit a multi-armed Qini curve.*

Description

Fit a curve that shows estimates of a policy value $Q(B)$ over increasing decision thresholds B. These may include constraints on the treatment allocation, such as the fraction treated or spending per unit. The policy uses estimated treatment effects, for example from one or more CATE functions, to optimize treatment allocation under the decision constraint B.

Usage

```
maq(
  reward,
  cost,
  DR.scores,
  budget = NULL,
  target.with.covariates = TRUE,
  R = 0,
  paired.inference = TRUE,
  sample.weights = NULL,
  clusters = NULL,
  tie.breaker = NULL,
  num.threads = NULL,
  seed = 42
)
```

Arguments

`reward`	A $n \cdot K$ matrix of test set treatment effect estimates $\hat{\tau}(X_i)$. (Note: the estimated function $\hat{\tau}(\cdot)$ should be constructed on a held-out training set)
`cost`	A $n \cdot K$ matrix of test set costs $C(X_i) > 0$, where entry (i, k) measures the cost of assigning the i-th unit the k-th treatment arm. If the costs does not vary by unit, only by arm, this can also be a K-length vector. (Note: these costs need not be denominated on the same scale as the treatment effect estimates).
`DR.scores`	An $n \cdot K$ matrix of test set evaluation scores used to form an estimate of Q(B). With known treatment propensities $P[W_i \| X_i]$, these scores can be constructed via inverse-propensity weighting, i.e, with entry (i, k) equal to $\frac{\mathbf{1}(W_i=k)Y_i}{P[W_i=k \| X_i]} - \frac{\mathbf{1}(W_i=0)Y_i}{P[W_i=0 \| X_i]}$. In observational settings where $P[W_i \| X_i]$ has to be estimated, then an alternative is to construct these scores via augmented inverse-propensity weighting (AIPW) - yielding a doubly robust estimate of the Qini curve (for details, see the paper).
`budget`	The maximum spend per unit, B_{max}, to fit the Qini curve on. Setting this to NULL (Default), will fit the path up to a maximum spend per unit where each unit that is expected to benefit (that is, $\hat{\tau}_k(X_i) > 0$) is treated.
`target.with.covariates`	If TRUE (Default), then the policy π_B takes covariates X_i into account. If FALSE, then the policy only takes the average reward $\bar{\tau} = E[\hat{\tau}(X_i)]$ and average costs $\bar{C} = E[C(X_i)]$ into account when allocating treatment. This can be used to construct a baseline Qini curve to assess the value of treatment targeting based on covariates.
`R`	Number of bootstrap replicates for computing standard errors. Default is 0 (only point estimates are computed).
`paired.inference`	Whether to allow for paired tests with other Qini curves fit on the same evaluation data. If TRUE (Default) then the path of bootstrap replicates are stored in order to perform paired comparisons that account for the correlation between curves evaluated on the same data. This takes memory on the order of $O(RnK)$ and requires the comparison objects to be fit with the same seed and R values as well as the same number of samples.
`sample.weights`	Weights given to an observation in estimation. If NULL, each observation is given the same weight. Default is NULL.
`clusters`	Vector of integers or factors specifying which cluster each observation corresponds to, which are used to construct clustered standard errors. Default is NULL (ignored).
`tie.breaker`	An optional permutation of the integers 1 to n used to break potential ties in the optimal treatment allocation (only relevant if the predictions $\hat{\tau}(X)$ are not continuous). If NULL, the ties are broken by the lowest sample id (i.e. the sample appearing first in the data). Default is NULL.

`num.threads`	Number of threads used in bootstrap replicates. By default, the number of threads is set to the maximum hardware concurrency.
`seed`	The seed of the C++ random number generator. Default is 42.

Details

Consider $k = 1, \ldots, K$ mutually exclusive and costly treatment arms, where k = 0 is a zero-cost control arm. Let $\hat{\tau}(\cdot)$ be an *estimated* multi-armed treatment effect function and $C(\cdot)$ a known cost function (where the k-th element of these vectors measures $E[Y_i(k) - Y_i(0)|X_i]$ and $E[C_i(k) - C_i(0)|X_i]$ where $Y_i(k)$ are potential outcomes corresponding to the k-th treatment state, $C_i(k)$ the cost of assigning unit i the k-th arm, and X_i a set of covariates). We provide estimates of the Qini curve:

$$Q(B) = E[\langle \pi_B(X_i), \tau(X_i) \rangle], B \in (0, B_{max}],$$

which is the expected gain, at any budget constraint B, when assigning treatment in accordance to π_B, the treatment policy that optimally selects which arm to assign to which unit while incurring a cost less than or equal to B in expectation when using the given functions $\hat{\tau}(\cdot)$ and $C(\cdot)$:

$$\pi_B = argmax_\pi \left\{ E[\langle \pi(X_i), \hat{\tau}(X_i) \rangle] : E[\langle \pi(X_i), C(X_i) \rangle] \le B \right\}.$$

At a budget B, the k-th element of $\pi_B(X_i)$ is 1 if assigning the k-th arm to the i-th unit is optimal, and 0 otherwise. The Qini curve can be used to quantify the value, as measured by the expected gain over assigning each unit the control arm when using the estimated function $\hat{\tau}(\cdot)$ with cost structure $C(\cdot)$ to allocate treatment, as we vary the available budget B.

Value

A fit maq object.

References

Sverdrup, Erik, Han Wu, Susan Athey, and Stefan Wager. "Qini Curves for Multi-Armed Treatment Rules". Journal of Computational and Graphical Statistics, forthcoming.

Examples

```
if (require("grf", quietly = TRUE)) {
# Fit a CATE estimator on a training sample.
n <- 3000
p <- 5
X <- matrix(runif(n * p), n, p)
W <- as.factor(sample(c("0", "1", "2"), n, replace = TRUE))
Y <- X[, 1] + X[, 2] * (W == "1") + 1.5 * X[, 3] * (W == "2") + rnorm(n)
train <- sample(1:n, n/2)

tau.forest <- grf::multi_arm_causal_forest(X[train, ], Y[train], W[train])

# Predict CATEs on held out evaluation data.
test <- -train
tau.hat <- predict(tau.forest, X[test, ], drop = TRUE)$predictions
# Assume costs equal a unit's pre-treatment covariate - the following is a toy example.
cost <- cbind(X[test, 4] / 4, X[test, 5])

# Fit an evaluation forest to compute doubly robust scores on the test set.
eval.forest <- grf::multi_arm_causal_forest(X[test, ], Y[test], W[test])
DR.scores <- grf::get_scores(eval.forest, drop = TRUE)

# Fit a Qini curve on evaluation data, using 200 bootstrap replicates for confidence intervals.
ma.qini <- maq(tau.hat, cost, DR.scores, R = 200)

# Plot the Qini curve.
plot(ma.qini)
legend("topleft", c("All arms", "95% CI"), lty = c(1, 3))

# Get an estimate of gain at a given spend per unit along with standard errors.
average_gain(ma.qini, spend = 0.2)

# Get the treatment allocation matrix at a given spend per unit.
pi.mat <- predict(ma.qini, spend = 0.2)

# If the treatment randomization probabilities are known, then an alternative to
# evaluation via AIPW scores is to use inverse-propensity weighting (IPW).
W.hat <- rep(1/3, 3)
IPW.scores <- get_ipw_scores(Y[test], W[test], W.hat)
mq.ipw <- maq(tau.hat, cost, IPW.scores)

plot(mq.ipw, add = TRUE, col = 2)
legend("topleft", c("All arms", "95% CI", "All arms (IPW)"), col = c(1, 1, 2), lty = c(1, 3, 1))

# Estimate some baseline policies.
# a) A policy that ignores covariates and only takes the average reward/cost into account.
qini.avg <- maq(tau.hat, cost, DR.scores, target.with.covariates = FALSE, R = 200)

# b) A policy that only use arm 1.
qini.arm1 <- maq(tau.hat[, 1], cost[, 1], DR.scores[, 1], R = 200)

# c) A policy that only use arm 2.
qini.arm2 <- maq(tau.hat[, 2], cost[, 2], DR.scores[, 2], R = 200)

plot(ma.qini, ci.args = NULL)
plot(qini.avg, col = 2, add = TRUE, ci.args = NULL)
plot(qini.arm1, col = 3, add = TRUE, ci.args = NULL)
plot(qini.arm2, col = 4, add = TRUE, ci.args = NULL)
legend("topleft", c("All arms (targeting)", "All arms (without targeting)", "Arm 1", "Arm 2"),
       col = 1:4, lty = 1)

# Estimate the value of employing all arms over a random allocation.
difference_gain(ma.qini, qini.avg, spend = 0.2)

# Estimate the value of targeting with both arms as opposed to targeting with only arm 1.
difference_gain(ma.qini, qini.arm1, spend = 0.2)

# Estimate the value of targeting with both arms as opposed to targeting with only arm 2.
```

```
difference_gain(ma.qini, qini.arm2, spend = 0.2)

# Compare targeting strategies over a range of budget values by estimating an area between
# two curves up to a given spend point.
integrated_difference(ma.qini, qini.arm1, spend = 0.3)
}
```

`plot.maq` *Plot the estimated Qini curve.*

Description

Plot the estimated curve $Q(B), B \in (0, B_{max}]$. If the underlying estimated policy π_B entails treating zero units (that is, all the estimated treatment effects are negative) then this function returns an empty value.

Usage

```
## S3 method for class 'maq'
plot(
  x,
  ...,
  add = FALSE,
  horizontal.line = TRUE,
  ci.args = list(),
  grid.step = NULL
)
```

Arguments

`x`	A maq object.
`...`	Additional arguments passed to plot.
`add`	Whether to add to an already existing plot. Default is FALSE.
`horizontal.line`	Whether to draw a horizontal line where the Qini curve plateaus. Only applies if the maq object is fit with a maximum `budget` that is sufficient to treat all units that are expected to benefit. Default is TRUE.
`ci.args`	A list of optional arguments to `lines()` for drawing 95 % confidence bars. Set to NULL to ignore CIs.
`grid.step`	The spend grid increment size to plot the curve on. Default is `max(floor(length(path.length) / 1000), 1)` where path.length is the size of the grid underlying the estimated Qini curve.

Value

A data.frame with the data making up the plot (point estimates and lower/upper 95% CIs)

Examples

```
if (require("ggplot2", quietly = TRUE)) {
# Generate toy data and customize plots.
n <- 500
K <- 1
reward <- matrix(1 + rnorm(n * K), n, K)
scores <- reward + matrix(rnorm(n * K), n, K)
cost <- 1

# Fit Qini curves.
qini.avg <- maq(reward, cost, scores, R = 200, target.with.covariates = FALSE)
qini <- maq(reward, cost, scores, R = 200)

# In some settings we may want to plot using one of R's many plot libraries.
# The plot method invisibly returns the plot data we can use for this purpose.
df.qini.baseline <- plot(qini.avg)
df.qini <- plot(qini, add = TRUE, col = 2)

# Make an alternate plot style, using, for example, ggplot.
ggplot(df.qini, aes(x = spend, y = gain)) +
  geom_ribbon(aes(ymin = gain - 1.96 * std.err,
                  ymax = gain + 1.96 * std.err),
              fill = "lightgray") +
  geom_line(linewidth = 2) +
  ylab("Policy value") +
  xlab("Fraction treated") +
  geom_line(data = df.qini.baseline, aes(x = spend, y = gain), lty = 2)
}
```

predict.maq *Predict treatment allocation.*

Description

Get an estimate of the policy $\pi_B(X_i)$ at a spend level B. $\pi_B(X_i)$ is a K-dimensional vector where the k-th element is 1 if assigning the k-th arm to unit i is optimal at a given spend B, and 0 otherwise (with all entries 0 if the control arm is assigned). Depending on the value of B, $\pi_B(X_j)$ might be fractional for at most one unit j. There are two such cases - the first one is when there is not sufficient budget left to assign j an initial arm. The second is if there is not sufficient budget to upgrade unit j from arm k to k'. In these cases $\pi_B(X_j)$ takes on one, or two fractional values, respectively, representing an assignment probability of a given arm.

Usage

```
## S3 method for class 'maq'
predict(object, spend, type = c("matrix", "vector"), ...)
```

Arguments

`object`	A maq object.
`spend`	The spend level B.
`type`	If "matrix" (Default), then return a matrix where the i-th entry equals $\pi_B(X_i)$ as described above. If "vector", then $\pi_B(X_i)$ is instead encoded taking values in the set {0, 1, ..., K}. If the allocation is fractional at the given B, this option returns the policy corresponding to the previous/lower value of the spend path, at which point the policy is integer-valued, but incurs a cost less than B in expectation.
`...`	Additional arguments (currently ignored).

Value

A matrix with row i equal to $\pi_B(X_i)$. If `type = "vector"` then an n-length vector with elements equal to the arm (from 0 to K) that is assigned at the given spend B (note: if the treatment allocation contains a fractional entry at the given B, then the returned vector is the policy at the nearest spend B' in the solution path where the allocation is integer-valued but incurs a cost B' < B).

Examples

```
# Generate some toy data and fit a solution path.
n <- 10
K <- 4
reward <- matrix(rnorm(n * K), n, K)
cost <- matrix(runif(n * K), n, K)
DR.scores <- reward + rnorm(n)
path <- maq(reward, cost, DR.scores)

# Get the treatment allocation matrix
pi.mat <- predict(path, 0.1)
pi.mat
# pi.mat might have fractional entries for a single unit but satisfies
# the budget in expectation exactly.
sum(cost * pi.mat) / n

# Get the treatment allocation instead encoded in the set {0, 1, ..., K}.
pi.vec <- predict(path, 0.1, type = "vector")
pi.vec
# If a unit has a fractional entry, then pi.vec will incur a cost slightly
# lower than 0.1.
sum(cost[cbind(1:n, pi.vec)]) / n

# Retrieve the underlying solution path.
data.path <- summary(path)
# If we predict at a spend level on this grid, say entry 5,
# then the policy is integer-valued:
spend <- data.path$spend[5]
predict(path, spend)
predict(path, spend, type = "vector")
```

print.maq *Print a maq object.*

Description

Print a maq object.

Usage

```
## S3 method for class 'maq'
print(x, ...)
```

Arguments

x	A maq object.
...	Additional arguments (currently ignored).

scale_maq *Scale a Qini curve.*

Description

Remaps the policy value and budget to application-specific units. This convenience function is typically useful for plots.

Usage

```
scale_maq(object, scale = 1)
```

Arguments

object	A maq object.
scale	A numeric value to scale by.

Value

A maq object with policy values and budget rescaled by the given factor.

Examples

```
# Generate some single-arm toy data.
n <- 1500
K <- 1
reward <- matrix(1 + runif(n * K), n, K)
scores <- reward + 5 * matrix(rnorm(n * K), n, K)
cost <- 1

# Fit a Qini curve.
qini <- maq(reward, cost, scores, R = 200)

# Plot the policy values as we vary the fraction treated.
plot(qini, xlab = "Fraction treated")

# Plot the policy values for a maximum allocation of, for example, 500 units.
plot(scale_maq(qini, 500), xlab = "Units treated")

# With R 4.1.0 or later, the native pipe can be used to chain scaling and plotting.
# scale_maq(qini, 500) |>
#   plot(xlab = "Units treated")
```

`summary.maq` *Qini curve summary.*

Description

Get a data.frame with columns equal to [B, Q(B), std.err(Q(B)), i, k], where i is the unit and k the treatment arm that is optimal to assign at a spend level B.

Usage

```
## S3 method for class 'maq'
summary(object, ...)
```

Arguments

`object`	A maq object.
`...`	Additional arguments (currently ignored).

Value

A data.frame making up the elements of the estimated Qini curve.

Index

average_gain, 2

difference_gain, 2

get_aipw_scores, 3
get_ipw_scores, 4

integrated_difference, 6

maq, 6

plot.maq, 10
predict.maq, 11
print.maq, 13

scale_maq, 13
summary.maq, 14