Package 'qgcomp'

January 17, 2020

Title Quantile G-Computation

Version 2.0.0

Date 2020-01-16

Author Alexander Keil [aut, cre]

Maintainer Alexander Keil <akeil@unc.edu>

Description G-computation for a set of time-fixed exposures with quantile-based basis functions, possibly under linearity and homogeneity assumptions. This approach estimates a regression line corresponding to the expected change in the outcome (on the link basis) given a simultaneous increase in the quantile-based category for all exposures. Works with continuous, binary, and right-censored time-to-event outcomes. Reference: Alexander P. Keil, Jessie P. Buckley, Katie M. OBrien, Kelly K. Ferguson, Shanshan Zhao Alexandra J. White (2019) A quantile-based g-computation approach to addressing the effects of exposure mixtures; <arXiv:1902.04200> [stat.ME].

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports arm, future, future.apply, ggplot2, grDevices, grid, gridExtra, pscl, stats, survival

Suggests knitr

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Repository CRAN

Date/Publication 2020-01-16 23:10:02 UTC

R topics documented:

checknames . . . 2
coxmsm.fit . . . 3
metals . . . 4
msm.fit . . . 6
msm.predict . . . 7
plot.qgcompfit . . . 8
predict.qgcompfit . . . 9
print.qgcompfit . . . 10
qgcomp . . . 11
qgcomp.boot . . . 13
qgcomp.cox.boot . . . 15
qgcomp.cox.noboot . . . 18
qgcomp.noboot . . . 19
qgcomp.zi.noboot . . . 21
quantize . . . 23
se_comb . . . 24

Index **25**

`checknames` *check for valid model terms if 'expnms' parameter not given*

Description

This is an internal function called by `qgcomp`, `qgcomp.boot`, and `qgcomp.noboot`, but is documented here for clarity. Generally, users will not need to call this function directly. This function tries to determine whether there are non-linear terms in the underlying model, which helps infer whether the appropriate function is called, and whether more explicit function calls are needed.

Usage

```
checknames(terms)
```

Arguments

`terms`	model terms from attr(terms(modelfunction, data), "term.labels")

coxmsm.fit *estimating the parameters of a marginal structural model (MSM) based on g-computation with quantized exposures*

Description

this is an internal function called by qgcomp.cox.noboot, qgcomp.cox.boot, and qgcomp.cox.noboot, but is documented here for clarity. Generally, users will not need to call this function directly.

Usage

```
coxmsm.fit(
  f,
  qdata,
  intvals,
  expnms,
  main = TRUE,
  degree = 1,
  id = NULL,
  MCsize = 10000,
  ...
)
```

Arguments

f	an r formula representing the conditional model for the outcome, given all exposures and covariates. Interaction terms that include exposure variables should be represented via the AsIs function
qdata	a data frame with quantized exposures (as well as outcome and other covariates)
intvals	sequence, the sequence of integer values that the joint exposure is 'set' to for estimating the msm. For quantile g-computation, this is just 0:(q-1), where q is the number of quantiles of exposure.
expnms	a character vector with the names of the columns in qdata that represent the exposures of interest (main terms only!)
main	logical, internal use: produce estimates of exposure effect (psi) and expected outcomes under g-computation and the MSM
degree	polynomial basis function for marginal model (e.g. degree = 2 allows that the relationship between the whole exposure mixture and the outcome is quadratic. Default=1)
id	(optional) NULL, or variable name indexing individual units of observation (only needed if analyzing data with multiple observations per id/cluster)
MCsize	integer: sample size for simulation to approximate marginal hazards ratios
...	arguments to coxph (e.g. ties)

Details

This function first computes expected outcomes under hypothetical interventions to simultaneously set all exposures to a specific quantile. These predictions are based on g-computation, where the exposures are 'quantized', meaning that they take on ordered integer values according to their ranks, and the integer values are determined by the number of quantile cutpoints used. The function then takes these expected outcomes and fits an additional model (a marginal structural model) with the expected outcomes as the outcome and the intervention value of the exposures (the quantile integer) as the exposure. Under causal identification assumptions and correct model specification, the MSM yields a causal exposure-response representing the incremental change in the expected outcome given a joint intervention on all exposures.

See Also

qgcomp.cox.boot, and qgcomp.cox.noboot

Examples

```
set.seed(50)
dat <- data.frame(time=(tmg <- pmin(.1,rweibull(50, 10, 0.1))), d=1.0*(tmg<0.1),
                  x1=runif(50), x2=runif(50), z=runif(50))
expnms=paste0("x", 1:2)
qdata  = qgcomp:::quantize(dat, expnms)$data
f = survival::Surv(time, d)~x1 + x2
fit <- survival::coxph(f, data = qdata, y=TRUE, x=TRUE)
r1 = qdata[1,,drop=FALSE]
times = survival::survfit(fit, newdata=r1, se.fit=FALSE)$time
(obj <- qgcomp:::coxmsm.fit(f, qdata, intvals=c(0,1,2,3), expnms, main=TRUE, degree=1,
   id=NULL, MCsize=100))
#dat2 <- data.frame(psi=seq(1,4, by=0.1))
#summary(predict(obj))
#summary(predict(obj, newdata=dat2))
```

---

metals

Simulated well water measurements in North Carolina: 16 metals, 6 water chemistry measures, and 2 health outcomes (y = continuous; disease_state = binary/time-to-event in combination with disease_time)

Description

A dataset containing well water measurements and health outcomes for 253 individuals. All continuous variables are standardized to have mean 0, standard deviation 1.

Usage

```
metals
```

Format

A data frame with 253 rows and 24 variables:

y continuous birth outcome

disease_state binary outcome

disease_time time-to-disease_state: survival outcome censored at approximately the median

arsenic metal

barium metal

cadmium metal

calcium metal

chloride metal

chromium metal

copper metal

iron metal

lead metal

magnesium metal

manganese metal

mercury metal

selenium metal

silver metal

sodium metal

zinc metal

mage35 Binary covariate: maternal age > 35

nitrate water chemistry measure

nitrite water chemistry measure

sulfate water chemistry measure

ph water chemistry measure

total_alkalinity water chemistry measure

total_hardness water chemistry measure

`msm.fit` *fitting marginal structural model (MSM) based on g-computation with quantized exposures*

Description

This is an internal function called by `qgcomp`, `qgcomp.boot`, and `qgcomp.noboot`, but is documented here for clarity. Generally, users will not need to call this function directly.

Usage

```
msm.fit(
  f,
  qdata,
  intvals,
  expnms,
  rr = TRUE,
  main = TRUE,
  degree = 1,
  id = NULL,
  bayes,
  ...
)
```

Arguments

`f`	an r formula representing the conditional model for the outcome, given all exposures and covariates. Interaction terms that include exposure variables should be represented via the `AsIs` function
`qdata`	a data frame with quantized exposures
`intvals`	sequence, the sequence of integer values that the joint exposure is 'set' to for estimating the msm. For quantile g-computation, this is just 0:(q-1), where q is the number of quantiles of exposure.
`expnms`	a character vector with the names of the columns in qdata that represent the exposures of interest (main terms only!)
`rr`	logical, estimate log(risk ratio) (family='binomial' only)
`main`	logical, internal use: produce estimates of exposure effect (psi) and expected outcomes under g-computation and the MSM
`degree`	polynomial basis function for marginal model (e.g. degree = 2 allows that the relationship between the whole exposure mixture and the outcome is quadratic. Default=1)
`id`	(optional) NULL, or variable name indexing individual units of observation (only needed if analyzing data with multiple observations per id/cluster)

bayes	use underlying Bayesian model (‘arm‘ package defaults). Results in penalized parameter estimation that can help with very highly correlated exposures. Note: this does not lead to fully Bayesian inference in general, so results should be interpereted as frequentist.
...	arguments to glm (e.g. family)

Details

This function first computes expected outcomes under hypothetical interventions to simultaneously set all exposures to a specific quantile. These predictions are based on g-computation, where the exposures are ‘quantized’, meaning that they take on ordered integer values according to their ranks, and the integer values are determined by the number of quantile cutpoints used. The function then takes these expected outcomes and fits an additional model (a marginal structural model) with the expected outcomes as the outcome and the intervention value of the exposures (the quantile integer) as the exposure. Under causal identification assumptions and correct model specification, the MSM yields a causal exposure-response representing the incremental change in the expected outcome given a joint intervention on all exposures.

See Also

qgcomp.boot, and qgcomp

Examples

```
set.seed(50)
dat <- data.frame(y=runif(200), x1=runif(200), x2=runif(200), z=runif(200))
X <- c('x1', 'x2')
qdat <- quantize(dat, X, q=4)$data
mod <- qgcomp:::msm.fit(f=y ~ z + x1 + x2 + I(x1*x2),
        expnms = c('x1', 'x2'), qdata=qdat, intvals=1:4, bayes=FALSE)
summary(mod$fit) # outcome regression model
summary(mod$msmfit) # msm fit (variance not valid - must be obtained via bootstrap)
```

---

msm.predict *msm.predict: secondary prediction method for the MSM within a qgcompfit object (non-survival outcomes only).*

Description

Makes predictions from the MSM (rather than the g-computation model) from a "qgcompfit" object. Generally, this should not be used in favor of the default predict.qgcompfit function. This function can only be used following the ‘qgcomp.boot‘ function. For the ‘qgcomp.noboot‘ function, predict.qgcompfit gives identical inference to predicting from an MSM.

get predicted values from a qgcompfit object from qgcomp.boot

Usage

```
msm.predict(object, newdata = NULL)
```

Arguments

object	"qgcompfit" object from ‘qgcomp.boot‘ function
newdata	(optional) new set of data (data frame) with a variable called ‘psi‘ representing the joint exposure level of all exposures under consideration

Examples

```
set.seed(50)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50), z=runif(50))
obj <- qgcomp.boot(y ~ z + x1 + x2 + I(z*x1), expnms = c('x1', 'x2'), data=dat, q=4, B=10, seed=125)
dat2 <- data.frame(psi=seq(1,4, by=0.1))
summary(msm.predict(obj))
summary(msm.predict(obj, newdata=dat2))
```

---

plot.qgcompfit *plot.qgcompfit: default plotting method for a qgcompfit object*

Description

Plot a quantile g-computation object. For qgcomp.noboot, this function will create a butterfly plot of weights. For qgcomp.boot, this function will create a box plot with smoothed line overlaying that represents a non-parametric fit of a model to the expected outcomes in the population at each quantile of the joint exposures (e.g. ’1’ represents ’at the first quantile for every exposure’)

Usage

```
## S3 method for class 'qgcompfit'
plot(
  x,
  suppressprint = FALSE,
  pointwisebars = TRUE,
  modelfitline = TRUE,
  modelband = TRUE,
  flexfit = TRUE,
  pointwiseref = 1,
  ...
)
```

Arguments

x	"qgcompfit" object from ‘qgcomp.noboot‘, ‘qgcomp.boot‘, ‘qgcomp.cox.noboot‘, ‘qgcomp.cox.boot‘, ‘qgcomp.zi.noboot‘ or ‘qgcomp.zi.boot‘ functions
suppressprint	If TRUE, suppresses the plot, rather than printing it by default (it can be saved as a ggplot2 object (or list of ggplot2 objects if x is from a zero- inflated model) and used programmatically) (default = FALSE)
pointwisebars	(boot.gcomp only) If TRUE, adds 95% error bars for pointwise comparisons of E(Y|joint exposure) to the smooth regression line plot

modelfitline	(boot.gcomp only) If TRUE, adds fitted (MSM) regression line of E(Y|joint exposure) to the smooth regression line plot
modelband	If TRUE, adds 95% prediction bands for E(Y|joint exposure) (the MSM fit)
flexfit	(boot.gcomp only) if TRUE, adds flexible interpolation of predictions from underlying (conditional) model
pointwiseref	(boot.gcomp only) integer: which category of exposure (from 1 to q) should serve as the referent category for pointwise comparisons? (default=1)
...	unused

See Also

qgcomp.noboot, qgcomp.boot, and qgcomp

Examples

```
set.seed(12)
dat <- data.frame(x1=(x1 <- runif(100)), x2=runif(100), x3=runif(100), z=runif(100),
                  y=runif(100)+x1+x1^2)
ft <- qgcomp.noboot(y ~ z + x1 + x2 + x3, expnms=c('x1','x2','x3'), data=dat, q=4)
ft
plot(ft)
# examining fit
plot(ft$fit, which=1) # residual vs. fitted is not straight line!

# using non-linear outcome model
ft2 <- qgcomp.boot(y ~ z + x1 + x2 + x3 + I(x1*x1), expnms=c('x1','x2','x3'),
data=dat, q=4, B=10)
ft2
plot(ft2$fit, which=1) # much better looking fit diagnostics suggests
# it is better to include interaction term for x
plot(ft2) # the msm predictions don't match up with a smooth estimate
# of the expected outcome, so we should consider a non-linear MSM
# using non-linear marginal structural model
ft3 <- qgcomp.boot(y ~ z + x1 + x2 + x3 + I(x1*x1), expnms=c('x1','x2','x3'),
data=dat, q=4, B=10, degree=2)
# plot(ft3$fit, which=1) - not run - this is identical to ft2 fit
plot(ft3) # the MSM estimates look much closer to the smoothed estimates
# suggesting the non-linear MSM fits the data better and should be used
# for inference about the effect of the exposure
```

---

predict.qgcompfit *predict.qgcompfit: default prediction method for a qgcompfit object (non-survival outcomes only)*

Description

get predicted values from a qgcompfit object, or make predictions in a new set of data based on the qgcomfit object. Note that when making predictions from an object from qgcomp.boot, the predictions are made from the g-computation model rather than the marginal structural model. Predictions from the marginal structural model can be obtained via msm.predict

Usage

```
## S3 method for class 'qgcompfit'
predict(object, expnms = NULL, newdata = NULL, type = "response", ...)
```

Arguments

object	"qgcompfit" object from ‘qgcomp.noboot‘ or ‘qgcomp.boot‘ functions
expnms	character vector of exposures of interest
newdata	(optional) new set of data with all predictors from "qgcompfit" object
type	(from predict.glm) the type of prediction required. The default is on the scale of the linear predictors; the alternative "response" is on the scale of the response variable. Thus for a default binomial model the default predictions are of log-odds (probabilities on logit scale) and type = "response" gives the predicted probabilities. The "terms" option returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.
...	arguments to predict.glm

Examples

```
set.seed(50)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50), z=runif(50))
obj1 <- qgcomp.noboot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2)
obj2 <- qgcomp.boot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, B=10, seed=125)
set.seed(52)
dat2 <- data.frame(y=runif(50), x1=runif(50), x2=runif(50), z=runif(50))
summary(predict(obj1, expnms = c('x1', 'x2'), newdata=dat2))
summary(predict(obj2, expnms = c('x1', 'x2'), newdata=dat2))
```

---

print.qgcompfit *default printing method for a qgcompfit object*

Description

Gives variable output depending on whether ‘qgcomp.noboot‘ or ‘qgcomp.boot‘ is called. For ‘qgcomp.noboot‘ will output final estimate of joint exposure effect (similar to the ’index’ effect in weighted quantile sums), as well as estimates of the ’weights’ (standardized coefficients). For ‘qgcomp.boot‘, the marginal effect is given, but no weights are reported since this approach generally incorporates non-linear models with interaction terms among exposures, which preclude weights with any useful interpretation.

Usage

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

Arguments

x	"qgcompfit" object from ‘qgcomp‘, ‘qgcomp.noboot‘ or ‘qgcomp.boot‘ function
...	unused

See Also

qgcomp.noboot, qgcomp.boot, and qgcomp

Examples

```
set.seed(50)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50), z=runif(50))
obj1 <- qgcomp.noboot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2)
obj2 <- qgcomp.boot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, B=10, seed=125)
# does not need to be explicitly called, but included here for clarity
print(obj1)
print(obj2)
```

---

qgcomp *estimation of quantile g-computation fit*

Description

This function automatically selects between qgcomp.noboot, qgcomp.boot, qgcomp.cox.noboot, and qgcomp.cox.boot for the most efficient approach to estimate the average expected change in the (log) outcome per quantile increase in the joint exposure to all exposures in ‘expnms’, given the underlying model. For example, if the underlying model (specified by the formula ‘f‘) is a linear model with all linear terms for exposure, then ‘qgcomp.noboot“ will be called to fit the model. Non-linear terms or requesting the risk ratio for binomial outcomes will result in the ‘qgcomp.boot‘ function being called. For a given linear model, boot and noboot versions will give identical inference, though when using survival outcomes, the ‘boot‘ version uses simulation based inference, which can vary from the ‘nonboot‘ version due to simulation error (which can be minimized via setting the MCsize parameter very large - see qgcomp.cox.boot for details).

Usage

```
qgcomp(f, data = data, family = gaussian(), rr = TRUE, ...)
```

Arguments

f	R style formula (may include survival outcome via Surv)
data	data frame
family	‘gaussian()‘, ‘binomial()‘, ‘cox()‘
rr	logical: if using binary outcome and rr=TRUE, qgcomp.boot will estimate risk ratio rather than odds ratio. Note, to get population average effect estimates for a binary outcome, set rr=TRUE (default: ORs are generally not of interest as population average effects, so if rr=FALSE then a conditional OR will be estimated, which cannot be interpreted as a population average effect

...	arguments to qgcomp.noboot or qgcomp.boot (e.g. q)

Value

a qgcompfit object, which contains information about the effect measure of interest (psi) and associated variance (var.psi), as well as information on the model fit (fit) and possibly information on the marginal structural model (msmfit) used to estimate the final effect estimates (qgcomp.boot, qgcomp.cox.boot only). If appropriate, weights are also reported, which represent the proportion of a directional (positive/negative) effect that is accounted for by each exposure.

See Also

qgcomp.noboot, qgcomp.boot, qgcomp.cox.noboot and qgcomp.cox.boot (qgcomp is just a wrapper for these functions)

Examples

```
set.seed(50)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50), z=runif(50))
qgcomp.noboot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2)
qgcomp.boot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, B=10, seed=125)
# automatically selects appropriate method
qgcomp(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2)
# note for binary outcome this will
dat <- data.frame(y=rbinom(50, 1, 0.5), x1=runif(50), x2=runif(50), z=runif(50))
qgcomp.noboot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, family=binomial())
qgcomp.boot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, B=10, seed=125,
  family=binomial())
# automatically selects appropriate method
qgcomp(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, family=binomial())
qgcomp(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2, family=binomial(), rr=TRUE)

#survival objects
set.seed(50)
N=200
dat <- data.frame(time=(tmg <- pmin(.1,rweibull(N, 10, 0.1))),
                d=1.0*(tmg<0.1), x1=runif(N), x2=runif(N), z=runif(N))
expnms=paste0("x", 1:2)
f = survival::Surv(time, d)~x1 + x2
qgcomp(f, expnms = expnms, data = dat)
# note that in the survival models, MCsize should be set to a large number
#  such that results are repeatable (within an error tolerance such as 2 significant digits)
# if you run them under different  seed values
f = survival::Surv(time, d)~x1 + x2 + x1:x2
qgcomp(f, expnms = expnms, data = dat, B=10, MCsize=100)
```

qgcomp.boot *estimating the parameters of a marginal structural model (MSM) based on g-computation with quantized exposures*

Description

This function yields population average effect estimates for both continuous and binary outcomes

Usage

```
qgcomp.boot(
  f,
  data,
  expnms = NULL,
  q = 4,
  breaks = NULL,
  id = NULL,
  alpha = 0.05,
  B = 200,
  rr = TRUE,
  degree = 1,
  seed = NULL,
  bayes = FALSE,
  parallel = FALSE,
  ...
)
```

Arguments

f	R style formula
data	data frame
expnms	character vector of exposures of interest
q	NULL or number of quantiles used to create quantile indicator variables representing the exposure variables. If NULL, then gcomp proceeds with untransformed version of exposures in the input datasets (useful if data are already transformed, or for performing standard g-computation)
breaks	(optional) NULL, or a list of (equal length) numeric vectors that characterize the minimum value of each category for which to break up the variables named in expnms. This is an alternative to using 'q' to define cutpoints.
id	(optional) NULL, or variable name indexing individual units of observation (only needed if analyzing data with multiple observations per id/cluster)
alpha	alpha level for confidence limit calculation
B	integer: number of bootstrap iterations (this should typically be >=200, though it is set lower in examples to improve run-time).

rr	logical: if using binary outcome and rr=TRUE, qgcomp.boot will estimate risk ratio rather than odds ratio
degree	polynomial basis function for marginal model (e.g. degree = 2 allows that the relationship between the whole exposure mixture and the outcome is quadratic.
seed	integer or NULL: random number seed for replicable bootstrap results
bayes	use underlying Bayesian model ('arm' package defaults). Results in penalized parameter estimation that can help with very highly correlated exposures. Note: this does not lead to fully Bayesian inference in general, so results should be interpereted as frequentist.
parallel	use (safe) parallel processing from the future and future.apply packages
...	arguments to glm (e.g. family)

Details

Estimates correspond to the average expected change in the (log) outcome per quantile increase in the joint exposure to all exposures in 'expnms'. Test statistics and confidence intervals are based on a non-parametric bootstrap, using the standard deviation of the bootstrap estimates to estimate the standard error. The bootstrap standard error is then used to estimate Wald-type confidence intervals. Note that no bootstrapping is done on estimated quantiles of exposure, so these are treated as fixed quantities

Value

a qgcompfit object, which contains information about the effect measure of interest (psi) and associated variance (var.psi), as well as information on the model fit (fit) and information on the marginal structural model (msmfit) used to estimate the final effect estimates.

See Also

qgcomp.noboot, and qgcomp

Examples

```
set.seed(30)
# continuous outcome
dat <- data.frame(y=rnorm(100), x1=runif(100), x2=runif(100), z=runif(100))
# Conditional linear slope
qgcomp.noboot(y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=4, family=gaussian())
# Marginal linear slope (population average slope, for a purely linear,
#  additive model this will equal the conditional)

qgcomp.boot(f=y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=4,
  family=gaussian(), B=200) # B should be at least 200 in actual examples

# Population average mixture slope which accounts for non-linearity and interactions
qgcomp.boot(y ~ z + x1 + x2 + I(x1^2) + I(x2*x1), family="gaussian",
 expnms = c('x1', 'x2'), data=dat, q=4, B=200)

# binary outcome
```

```
dat <- data.frame(y=rbinom(50,1,0.5), x1=runif(50), x2=runif(50), z=runif(50))

# Conditional mixture OR
qgcomp.noboot(y ~ z + x1 + x2, family="binomial", expnms = c('x1', 'x2'),
  data=dat, q=2)

#Marginal mixture OR (population average OR - in general, this will not equal the
# conditional mixture OR due to non-collapsibility of the OR)
qgcomp.boot(y ~ z + x1 + x2, family="binomial", expnms = c('x1', 'x2'),
  data=dat, q=2, B=3)

# Population average mixture RR
qgcomp.boot(y ~ z + x1 + x2, family="binomial", expnms = c('x1', 'x2'),
  data=dat, q=2, rr=TRUE, B=3)

# Population average mixture RR, indicator variable representation of x2
# note that I(x==...) operates on the quantile-based category of x,
# rather than the raw value
res = qgcomp.boot(y ~ z + x1 + I(x2==1) + I(x2==2) + I(x2==3),
  family="binomial", expnms = c('x1', 'x2'), data=dat, q=4, rr=TRUE, B=200)
res$fit
plot(res)

# now add in a non-linear MSM
res2 = qgcomp.boot(y ~ z + x1 + I(x2==1) + I(x2==2) + I(x2==3),
  family="binomial", expnms = c('x1', 'x2'), data=dat, q=4, rr=TRUE, B=200,
  degree=2)
res2$fit
res2$msmfit
plot(res2)
# Log risk ratio per one IQR change in all exposures (not on quantile basis)
dat$x1iqr <- dat$x1/with(dat, diff(quantile(x1, c(.25, .75))))
dat$x2iqr <- dat$x2/with(dat, diff(quantile(x2, c(.25, .75))))
# note that I(x>...) now operates on the untransformed value of x,
# rather than the quantized value
res2 = qgcomp.boot(y ~ z + x1iqr + I(x2iqr>0.1) + I(x2>0.4) + I(x2>0.9),
  family="binomial", expnms = c('x1iqr', 'x2iqr'), data=dat, q=NULL, rr=TRUE, B=200,
  degree=2)
res2
# using parallel processing

qgcomp.boot(y ~ z + x1iqr + I(x2iqr>0.1) + I(x2>0.4) + I(x2>0.9),
  family="binomial", expnms = c('x1iqr', 'x2iqr'), data=dat, q=NULL, rr=TRUE, B=200,
  degree=2, parallel=TRUE)
```

---

qgcomp.cox.boot *estimation of quantile g-computation fit for a survival outcome*

Description

This function yields population average effect estimates for (possibly right censored) time-to event outcomes

Usage

```
qgcomp.cox.boot(
  f,
  data,
  expnms = NULL,
  q = 4,
  breaks = NULL,
  id = NULL,
  alpha = 0.05,
  B = 200,
  MCsize = 10000,
  degree = 1,
  seed = NULL,
  parallel = FALSE,
  ...
)
```

Arguments

f	R style survival formula, which includes Surv in the outcome defintion. E.g. Surv(time,event) ~ exposure
data	data frame
expnms	character vector of exposures of interest
q	NULL or number of quantiles used to create quantile indicator variables representing the exposure variables. If NULL, then gcomp proceeds with untransformed version of exposures in the input datasets (useful if data are already transformed, or for performing standard g-computation)
breaks	(optional) NULL, or a list of (equal length) numeric vectors that characterize the minimum value of each category for which to break up the variables named in expnms. This is an alternative to using 'q' to define cutpoints.
id	(optional) NULL. Reserved for future use. This does nothing yet.
alpha	alpha level for confidence limit calculation
B	integer: number of bootstrap iterations (this should typically be >=200, though it is set lower in examples to improve run-time).
MCsize	integer: sample size for simulation to approximate marginal hazards ratios (if < sample size, then set to sample size). Note that large values will slow down the fitting, but will result in higher accuracy - if you run the function multiple times you will see that results vary due to simuation error. Ideally, MCsize would be set such that simulation error is negligible in the precision reported (e.g. if you report results to 2 decimal places, then MCsize should be set high enough that you consistenty get answers that are the same to 2 decimal places).

degree	polynomial basis function for marginal model (e.g. degree = 2 allows that the relationship between the whole exposure mixture and the outcome is quadratic.
seed	integer or NULL: random number seed for replicable bootstrap results
parallel	logical (default FALSE): use future package to speed up bootstrapping
...	arguments to glm (e.g. family)

Details

'qgcomp.cox.boot' estimates the log(hazard ratio) per quantile increase in the joint exposure to all exposures in 'expnms'. This function uses g-computation to estimate the parameters of a marginal structural model for the population average effect of increasing all expsoures in 'expnms' by a single quantile. This approach involves specifying an underlying conditional outcome model, given all exposures of interest (possibly with non-linear basis function representations such as splines or product terms) and confounders or covariates of interest. This model is fit first, which is used to generate expected outcomes at each quantile of all exposures, which is then used in a second model to estimate a population average dose-response curve that is linear or follows a simple polynomial function. See section on MCSize below

Test statistics and confidence intervals are based on a non-parametric bootstrap, using the standard deviation of the bootstrap estimates to estimate the standard error. The bootstrap standard error is then used to estimate Wald-type confidence intervals. Note that no bootstrapping is done on estimated quantiles of exposure, so these are treated as fixed quantities

MCSize is crucial to get accurate point estimates. In order to get marginal estimates of the population hazard under different values of the joint exposure at a given quantile for all exposures in 'expnmns', 'qgcomp.cox.boot' uses Monte Carlo simulation to generate outcomes implied by the underlying conditional model and then fit a separate (marginal structural) model to those outcomes. In order to get accurate results that don't vary much from run-to-run of this approach, MCsize must be set large enough so that results are stable across runs according to a pre-determined precision (e.g. 2 significant digits).

Value

a qgcompfit object, which contains information about the effect measure of interest (psi) and associated variance (var.psi), as well as information on the model fit (fit) and information on the marginal structural model (msmfit) used to estimate the final effect estimates.

See Also

qgcomp.cox.noboot, and qgcomp

Examples

```
set.seed(50)
N=200
dat <- data.frame(time=(tmg <- pmin(.1,rweibull(N, 10, 0.1))),
                d=1.0*(tmg<0.1), x1=runif(N), x2=runif(N), z=runif(N))
expnms=paste0("x", 1:2)
f = survival::Surv(time, d)~x1 + x2
(fit1 <- survival::coxph(f, data = dat))
```

```
(obj <- qgcomp.cox.noboot(f, expnms = expnms, data = dat))

# not run (slow when using boot version to proper precision)
(obj2 <- qgcomp.cox.boot(f, expnms = expnms, data = dat, B=10, MCsize=2000))
# using future package, marginalizing over confounder z
(obj3 <- qgcomp.cox.boot(survival::Surv(time, d)~x1 + x2 + z, expnms = expnms, data = dat,
                         B=1000, MCsize=20000, parallel=TRUE))
# non-constant hazard ratio, non-linear terms
(obj4 <- qgcomp.cox.boot(survival::Surv(time, d)~factor(x1) + splines::bs(x2) + z,
                         expnms = expnms, data = dat,
                         B=1000, MCsize=20000, parallel=FALSE, degree=1))
```

`qgcomp.cox.noboot` *estimation of quantile g-computation fit for a survival outcome*

Description

This function performs quantile g-computation in a survival setting. The approach estimates the covariate-conditional hazard ratio for a joint change of 1 quantile in each exposure variable specified in expnms parameter

Usage

```
qgcomp.cox.noboot(
  f,
  data,
  expnms = NULL,
  q = 4,
  breaks = NULL,
  id = NULL,
  alpha = 0.05,
  ...
)
```

Arguments

`f`	R style formula
`data`	data frame
`expnms`	character vector of exposures of interest
`q`	NULL or number of quantiles used to create quantile indicator variables representing the exposure variables. If NULL, then gcomp proceeds with untransformed version of exposures in the input datasets (useful if data are already transformed, or for performing standard g-computation)
`breaks`	(optional) NULL, or a list of (equal length) numeric vectors that characterize the minimum value of each category for which to break up the variables named in expnms. This is an alternative to using 'q' to define cutpoints.

id	(optional) NULL, or variable name indexing individual units of observation (only needed if analyzing data with multiple observations per id/cluster)
alpha	alpha level for confidence limit calculation
...	arguments to glm (e.g. family)

Details

For survival outcomes (as specified using methods from the survival package), this yields a conditional log hazard ratio representing a change in the expected conditional hazard (conditional on covariates) from increasing every exposure by 1 quantile. In general, this quantity quantity is not equivalent to g-computation estimates. Hypothesis test statistics and 95 estimate variance of a linear combination of random variables.

Value

a qgcompfit object, which contains information about the effect measure of interest (psi) and associated variance (var.psi), as well as information on the model fit (fit) and information on the weights/standardized coefficients in the positive (pweights) and negative (nweight) directions.

See Also

qgcomp.cox.boot, qgcomp.boot, and qgcomp

Examples

```
set.seed(50)
N=200
dat <- data.frame(time=(tmg <- pmin(.1,rweibull(N, 10, 0.1))),
                d=1.0*(tmg<0.1), x1=runif(N), x2=runif(N), z=runif(N))
expnms=paste0("x", 1:2)
f = survival::Surv(time, d)~x1 + x2
(fit1 <- survival::coxph(f, data = dat))
(obj <- qgcomp.cox.noboot(f, expnms = expnms, data = dat))

# not run: bootstrapped version is much slower
(obj2 <- qgcomp.cox.boot(f, expnms = expnms, data = dat, B=200, MCsize=20000))
```

---

qgcomp.noboot *estimating the parameters of a marginal structural model (MSM) based on g-computation with quantized exposures*

Description

This function mimics the output of a weighted quantile sums regression in large samples.

Usage

```
qgcomp.noboot(
  f,
  data,
  expnms = NULL,
  q = 4,
  breaks = NULL,
  id = NULL,
  alpha = 0.05,
  bayes = FALSE,
  ...
)
```

Arguments

f	R style formula
data	data frame
expnms	character vector of exposures of interest
q	NULL or number of quantiles used to create quantile indicator variables representing the exposure variables. If NULL, then gcomp proceeds with untransformed version of exposures in the input datasets (useful if data are already transformed, or for performing standard g-computation)
breaks	(optional) NULL, or a list of (equal length) numeric vectors that characterize the minimum value of each category for which to break up the variables named in expnms. This is an alternative to using 'q' to define cutpoints.
id	(optional) NULL, or variable name indexing individual units of observation (only needed if analyzing data with multiple observations per id/cluster)
alpha	alpha level for confidence limit calculation
bayes	use underlying Bayesian model ('arm' package defaults). Results in penalized parameter estimation that can help with very highly correlated exposures. Note: this does not lead to fully Bayesian inference in general, so results should be interpereted as frequentist.
...	arguments to glm (e.g. family)

Details

For continuous outcomes, under a linear model with no interaction terms, this is equivalent to g-computation of the effect of increasing every exposure by 1 quantile. For binary outcomes outcomes, this yields a conditional log odds ratio representing the change in the expected conditional odds (conditional on covariates) from increasing every exposure by 1 quantile. In general, the latter quantity is not equivalent to g-computation estimates. Hypothesis test statistics and 95 estimate variance of a linear combination of random variables.

Value

a qgcompfit object, which contains information about the effect measure of interest (psi) and associated variance (var.psi), as well as information on the model fit (fit) and information on the weights/standardized coefficients in the positive (pos.weights) and negative (nweight) directions.

See Also

qgcomp.boot, and qgcomp

Examples

```
set.seed(50)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50), z=runif(50))
qgcomp.noboot(f=y ~ z + x1 + x2, expnms = c('x1', 'x2'), data=dat, q=2)
```

---

qgcomp.zi.noboot *estimating the parameters of a zero-inflated marginal structural model (MSM) based on g-computation with quantized exposures*

Description

This function mimics the output of a weighted quantile sums regression in large samples.

Usage

```
qgcomp.zi.noboot(
  f,
  data,
  expnms = NULL,
  q = 4,
  breaks = NULL,
  id = NULL,
  alpha = 0.05,
  bayes = FALSE,
  ...
)
```

Arguments

f	R style formula using syntax from 'pscl' package: depvar ~ indvars_count | indvars_zero
data	data frame
expnms	character vector of exposures of interest
q	NULL or number of quantiles used to create quantile indicator variables representing the exposure variables. If NULL, then gcomp proceeds with untransformed version of exposures in the input datasets (useful if data are already transformed, or for performing standard g-computation)

breaks	(optional) NULL, or a list of (equal length) numeric vectors that characterize the minimum value of each category for which to break up the variables named in expnms. This is an alternative to using 'q' to define cutpoints.
id	(optional) NULL, or variable name indexing individual units of observation (only needed if analyzing data with multiple observations per id/cluster)
alpha	alpha level for confidence limit calculation
bayes	not yet implemented
...	arguments to zeroinf (e.g. dist)

Details

A zero-inflated version of quantile g-computation based on the implementation in the 'pscl' package. A zero-inflated distribution is a mixture distribution in which one of the distributions is a point mass at zero (with probability given by a logistoic model), and the other distribution is a discrete or continuous distribution. This estimates the effect of a joint increase in all exposures on 1) the odds of belonging to the "zero" vs. "count" portions of the distribution and/or 2) the rate parameter for the "count" portion of the distribution.

Value

a qgcompfit object, which contains information about the effect measure of interest (psi) and associated variance (var.psi), as well as information on the model fit (fit) and information on the weights/standardized coefficients in the positive (pos.weights) and negative (nweight) directions.

See Also

qgcomp.noboot, qgcomp.cox.noboot, and zeroinfl

Examples

```
set.seed(50)
n=100
dat <- data.frame(y=rbinom(n, 1, 0.5)*rpois(n, 1.2), x1=runif(n), x2=runif(n), z=runif(n))
# poisson count model, mixture in both portions
qgcomp.zi.noboot(f=y ~ z + x1 + x2 | x1 + x2, expnms = c('x1', 'x2'),
    data=dat, q=2, dist="poisson")
# negative binomial count model, mixture and covariate in both portions
qgcomp.zi.noboot(f=y ~ z + x1 + x2 | z + x1 + x2, expnms = c('x1', 'x2'),
   data=dat, q=2, dist="negbin")
qgcomp.zi.noboot(f=y ~ z + x1 + x2, expnms = c('x1', 'x2'),
   data=dat, q=2, dist="negbin") # equivalent
# negative binomial count model, mixture only in the 'count' portion of the model
qgcomp.zi.noboot(f=y ~ z + x1 + x2 | z, expnms = c('x1', 'x2'), data=dat, q=2, dist="negbin")
```

quantize *create variables representing indicator functions with cutpoints defined by quantiles*

Description

This function creates categorical variables in place of the exposure variables named in 'expnms.' For example, a continuous exposure 'x1' will be replaced in the output data by another 'x1' that takes on values 0:(q-1), where, for example, the value 1 indicates that the original x1 value falls between the first and the second quantile.

Usage

```
quantize(data, expnms, q = 4, breaks = NULL)
```

Arguments

data	a data frame
expnms	a character vector with the names of the columns to be quantized
q	integer, number of quantiles used in creating quantized variables
breaks	(optional) list of (equal length) numeric vectors that characterize the minimum value of each category for which to break up the variables named in expnms. This is an alternative to using 'q' to define cutpoints.

Details

This function is a vectorized version of 'quantile_f' from the 'gWQS' package that also allows the use of externally defined breaks

Examples

```
set.seed(1232)
dat = data.frame(y=runif(100), x1=runif(100), x2=runif(100), z=runif(100))
qdata = quantize(data=dat, expnms=c("x1", "x2"), q=4)
table(qdata$data$x1)
table(qdata$data$x2)
summary(dat[c("y", "z")]);summary(qdata$data[c("y", "z")]) # not touched
dat = data.frame(y=runif(100), x1=runif(100), x2=runif(100), z=runif(100))
# using 'breaks' requires specifying min and max (the qth quantile)
# example with theoretical quartiles (could be other relevant values)
qdata2 = quantize(data=dat, expnms=c("x1", "x2"),
   breaks=list(c(-1e64, .25, .5, .75, 1e64),
               c(-1e64, .25, .5, .75, 1e64)
               ))
table(qdata2$data$x1)
table(qdata2$data$x2)
```

se_comb *Calculate standard error of weighted linear combination of random variables*

Description

This function uses the Delta method to calculate standard errors of linear functions of variables (similar to 'lincom' in Stata). Generally, users will not need to call this function directly.

Usage

```
se_comb(expnms, covmat, grad = NULL)
```

Arguments

expnms	a character vector with the names of the columns to be of interest in the covariance matrix for a which a standard error will be calculated (e.g. same as expnms in qgcomp fit)
covmat	covariance matrix for parameters, e.g. from a model or bootstrap procedure
grad	the "weight" vector for calculating the contribution of each variable in expnms to the final standard error. For a linear combination, this is equal to a vector of ones (and is set automatically). Or can be calculated via the grad.poly procedure, in the case of coming up with proper weights when the combination of expnms derives from a polynomial function (as in qgcomp.boot with degree>1).

Details

This function takes inputs of a set of exposure names (character vector) and a covariance matrix (with colnames/rownames that contain the full set of exposure names), as well as a possible 'grad' parameter to calculate the variance of a weighted combination of the exposures in 'expnms', where the weights are based off of 'grad' (which defaults to 1, so that this function yields the variance of a sum of all variables in 'expnms')

Here is simple version of the delta method for a linear combination of three model coefficients:

$f(\beta) = \beta_1 + \beta_2 + \beta_3$ given gradient vector

$$G = [d(f(\beta))/d\beta_1 = 1, d(f(\beta))/d\beta_2 = 1, d(f(\beta))/d\beta_3 = 1]$$

$t(G)Cov(\beta)G$ = delta method variance, where t() is the transpose operator

Examples

```
vcov = rbind(c(1.2, .9),c(.9, 2.0))
colnames(vcov) <- rownames(vcov) <- expnms <- c("x1", "x2")
qgcomp:::se_comb(expnms, vcov, c(1, 0))^2 # returns the given variance
qgcomp:::se_comb(expnms, vcov, c(1, 1)) # default linear MSM fit: all exposures
# have equal weight
qgcomp:::se_comb(expnms, vcov, c(.3, .1)) # used when one exposure contributes
  # to the overall fit more than others  = d(msmeffect)/dx
```

Index

∗Topic **datasets**
metals, 4

AsIs, *3*, *6*

checknames, 2
coxmsm.fit, 3

metals, 4
msm.fit, 6
msm.predict, 7, *9*

plot.qgcompfit, 8
predict.qgcompfit, *7*, 9
print.qgcompfit, 10

qgcomp, *2*, *6*, *7*, *9*, *11*, 11, *12*, *14*, *17*, *19*, *21*
qgcomp.boot, *2*, *6*, *7*, *9*, *11*, *12*, 13, *19*, *21*
qgcomp.cox.boot, *3*, *4*, *11*, *12*, 15, *19*
qgcomp.cox.noboot, *3*, *4*, *12*, *17*, 18, *22*
qgcomp.noboot, *2*, *6*, *9*, *11*, *12*, *14*, 19, *22*
qgcomp.zi.noboot, 21
quantize, 23

se_comb, 24
Surv, *11*, *16*

zeroinfl, *22*