Package 'qgcompint'

October 13, 2022

Title Quantile G-Computation Extensions for Effect Measure Modification

Version 0.7.0

Date 2022-03-22

Author Alexander Keil [aut, cre]

Maintainer Alexander Keil <akeil@unc.edu>

URL https://github.com/alexpkeil1/qgcomp/

BugReports https://github.com/alexpkeil1/qgcomp/issues

Description G-computation for a set of time-fixed exposures with quantile-based basis functions, possibly under linearity and homogeneity assumptions. Effect measure modification in this method is a way to assess how the effect of the mixture varies by a binary, categorical or continuous variable. Reference: Alexander P. Keil, Jessie P. Buckley, Katie M. OBrien, Kelly K. Ferguson, Shanshan Zhao, and Alexandra J. White (2019) A quantile-based g-computation approach to addressing the effects of exposure mixtures; <doi:10.1289/EHP5838>.

License GPL (>= 2)

Depends R (>= 3.5.0)

Imports qgcomp, arm, survival, future, future.apply, ggplot2, gridExtra

Suggests knitr, markdown, devtools

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

NeedsCompilation no

Repository CRAN

Date/Publication 2022-03-22 16:00:02 UTC

R topics documented:

`getjointeffects` *Calculate joint effect of mixture effect and modifier vs. common referent*

Description

A standard qgcomp fit with effect measure modification only estimates effects at the referent (0) level of the modifier (psi1). This function can be used to estimate a "common referent" parameter that estimates the effect of being in a non-referent category of the modifier and increasing exposure by one quantile, relative to no change in exposure in the referent category of the modifier. This is generally useful for binary exposures (for a mixture with a set of binary exposures, this would be the "effect" of being exposed and at the index level of the mediator, relative to being unexposed in the referent level of the mediator), but it may also be of interest with more general exposures.

Usage

```
getjointeffects(x, emmval = 1, ...)
```

Arguments

`x`	"qgcompemmfit" object from qgcomp.emm.noboot function
`emmval`	numerical: value of effect measure modifier at which weights are generated
`...`	unused

Value

An object of class "qgcompemmeffects", which inherits from "qgcompemmfit" and "list"

This class contains the `emmval`-stratum specific effect estimates of the mixture. By default, this prints a coefficient table, similar to objects of type "qgcompemmfit" which displays the stratum specific joint effects from a "qgcompemmfit" model.

See Also

qgcomp.emm.noboot getstrateffects

Examples

```
library(qgcompint)
n = 500
dat <- data.frame(y=rbinom(n,1,0.5), cd=runif(n), pb=runif(n),
                  raceth=factor(sample(c("WNH", "BNH", "AMIND"), n, replace=TRUE),
                         levels = c("BNH", "WNH", "AMIND")))
(qfit <- qgcomp.emm.noboot(f=y ~cd + pb, emmvar="raceth",
                           expnms = c('cd', 'pb'), data=dat, q=4,
                           family=binomial()))


# first level of the stratifying variable should be the referent category,
#  which you can set with the "levels" argument to "factor" when
#  cleaning/generating data
levels(dat$raceth)

# stratum specific mixture log-odds ratios
# this one comes straight from the model (psi 1)
getjointeffects(qfit, emmval = "BNH")
# this will coincide with joint effects, since it is in the referent category
getstrateffects(qfit, emmval = "BNH")

# the stratum specific effect for a non-referent category of the EMM
#  will not coincide with the joint effect
getjointeffects(qfit, emmval = "AMIND")
getstrateffects(qfit, emmval = "AMIND")
```

`getstrateffects` *Calculate mixture effect at a set value of effect measure modifier*

Description

A standard qgcomp fit with effect measure modification only estimates effects at the referent (0) level of the modifier (psi1). This function can be used to estimate effects at arbitrary levels of the modifier

Usage

```
getstrateffects(x, emmval = 1, ...)
```

Arguments

`x`	"qgcompemmfit" object from qgcomp.emm.noboot function
`emmval`	numerical: value of effect measure modifier at which weights are generated
`...`	unused

Value

An object of class "qgcompemmeffects", which inherits from "qgcompemmfit" and "list"

This class contains the emmval-stratum specific effect estimates of the mixture. By default, this prints a coefficient table, similar to objects of type "qgcompemmfit" which displays the stratum specific joint effects from a "qgcompemmfit" model.

See Also

qgcomp.emm.noboot getstratweights

Examples

```
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
  z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat, q=2, family=gaussian()))
getstrateffects(qfit, emmval = 0)
strateffects = getstrateffects(qfit, emmval = 1)
```

---

getstratweights *Calculate weights at a set value of effect measure modifier*

Description

A standard qgcomp fit with effect measure modification only estimates weights at the referent (0) level of the modifier. This function can be used to estimate weights at arbitrary levels of the modifier

Usage

```
getstratweights(x, emmval = 1, ...)
```

Arguments

x	"qgcompemmfit" object from qgcomp.emm.noboot function
emmval	numerical: value of effect measure modifier at which weights are generated
...	unused

Value

An object of class "qgcompemmweights", which is just a special R list

This class contains the emmval-stratum specific weights of components of the mixture. By default, this prints a list of "weights", similar to objects of type "qgcompemmfit" which displays the stratum specific weights from a "qgcompemmfit" model (if it is run without bootstrapping).

See Also

qgcomp.emm.noboot

Examples

```
set.seed(1231)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
  z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat, q=2, family=gaussian()))
getstratweights(qfit, emmval = 0)
weights1 = getstratweights(qfit, emmval = 1)
weights1$pos.weights
```

---

modelbound *Estimating qgcomp regression line confidence bounds*

Description

Calculates: expected outcome (on the link scale), and upper and lower confidence intervals (both pointwise and simultaneous)

Usage

```
modelbound(x, emmval = 0, alpha = 0.05, pwonly = FALSE, ...)
```

Arguments

x	"qgcompemmfit" object from qgcomp.emm.boot,
emmval	fixed value for effect measure modifier at which pointwise comparisons are calculated
alpha	alpha level for confidence intervals
pwonly	logical: return only pointwise estimates (suppress simultaneous estimates)
...	not used

Details

This method leverages the bootstrap distribution of qgcomp model coefficients to estimate pointwise regression line confidence bounds. These are defined as the bounds that, for each value of the independent variable X (here, X is the joint exposure quantiles) the 95% bounds (for example) for the model estimate of the regression line E(Y|X) are expected to include the true value of E(Y|X) in 95% of studies. The "simultaneous" bounds are also calculated, and the 95% simultaneous bounds contain the true value of E(Y|X) for all values of X in 95% of studies. The latter are more conservative and account for the multiple testing implied by the former. Pointwise bounds are calculated via the standard error for the estimates of E(Y|X), while the simultaneous bounds are estimated using the bootstrap method of Cheng (reference below). All bounds are large sample

bounds that assume normality and thus will be underconservative in small samples. These bounds may also inclue illogical values (e.g. values less than 0 for a dichotomous outcome) and should be interpreted cautiously in small samples.

Reference:

Cheng, Russell CH. "Bootstrapping simultaneous confidence bands." Proceedings of the Winter Simulation Conference, 2005.. IEEE, 2005.

Value

A data frame containing

linpred: The linear predictor from the marginal structural model

r/o/m: The canonical measure (risk/odds/mean) for the marginal structural model link

se....: the stndard error of linpred

ul..../ll....: Confidence bounds for the effect measure, and bounds centered at the canonical measure (for plotting purposes)

The confidence bounds are either "pointwise" (pw) and "simultaneous" (simul) confidence intervals at each each quantized value of all exposures.

See Also

qgcomp.emm.boot

Examples

```
## Not run:
set.seed(50)
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
                  z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
                           expnms = c('x1', 'x2'), data=dat, q=4, family=gaussian()))
(qfit2 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
                          degree = 1,
                          expnms = c('x1', 'x2'), data=dat, q=4, family=gaussian()))
# modelbound(qfit) # this will error (only works with bootstrapped objects)
 modelbound(qfit2)
# logistic model
set.seed(200)
dat2 <- data.frame(y=rbinom(200, 1, 0.3), x1=runif(200), x2=runif(200),
                  z=rbinom(200,1,0.5))
(qfit3 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
                          degree = 1,
                        expnms = c('x1', 'x2'), data=dat2, q=4, rr = FALSE, family=binomial()))
modelbound(qfit3)
# risk ratios instead (check for upper bound > 1.0, indicating implausible risk)
(qfit3b <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
                          degree = 1,
                         expnms = c('x1', 'x2'), data=dat2, q=4, rr = TRUE, family=binomial()))
modelbound(qfit3b)
```

```
# categorical modifier
set.seed(50)
dat3 <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
   z=sample(0:2, 50,replace=TRUE), r=rbinom(50,1,0.5))
dat3$z = as.factor(dat3$z)
(qfit4 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
                          degree = 1,
                          expnms = c('x1', 'x2'), data=dat3, q=4, family=gaussian()))
modelbound(qfit4, emmval=2)

## End(Not run)
```

`plot.qgcompemmfit` *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 'qgcompemmfit'
plot(x, emmval = 0, suppressprint = FALSE, ...)
```

Arguments

`x`	"qgcompfit" object from `qgcomp.noboot`, `qgcomp.boot`, `qgcomp.cox.noboot`, `qgcomp.cox.boot`, `qgcomp.zi.noboot` or `qgcomp.zi.boot` functions
`emmval`	fixed value for effect measure modifier at which pointwise comparisons are calculated
`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)
`...`	unused

Value

If suppressprint=FALSE, then this function prints a plot specific to a "qgcompemmfit" object. If no bootstrapping is used, it will print a butterfly plot of the weights at the specified value of the modifier (set via `emmval` parameter) If bootstrapping is used, it will print a joint regression line for all exposures at the specified value of the modifier (set via `emmval` parameter)

If suppressprint=TRUE, then this function returns a "gg" (regression line) or "gtable" (butterfly plot) object (from ggplot2 package or gtable/grid packages), which can be used to print a ggplot figure and modify either of the above figures (see example below)

See Also

qgcomp.noboot, qgcomp.boot, and qgcomp

Examples

```
set.seed(50)
# linear model, binary modifier
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
expnms = c('x1', 'x2'), data=dat, q=2, family=gaussian()))
plot(qfit, emmval = 1)
#
library(ggplot2)

# example with bootstrapping
dat2 <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
z=sample(0:2, 50,replace=TRUE), r=rbinom(50,1,0.5))
dat2$z = as.factor(dat2$z)
(qfit4 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
                          degree = 1, B = 20,
                         expnms = c('x1', 'x2'), data=dat2, q=4, family=gaussian()))
plot(qfit4)
pp0 = plot(qfit4, emmval=0, suppressprint=TRUE)
pp1 = plot(qfit4, emmval=1, suppressprint=TRUE)
pp2 = plot(qfit4, emmval=2, suppressprint=TRUE)
pp1 + theme_linedraw() # can use with other ggplot functions

# overlay (fussy to work with)
ppom <- ggplot_build(pp0 + pp1[2] + pp2[[2]] + scale_color_discrete(guide="none"))
ppom$data[[1]]$colour <- ppom$data[[2]]$colour <- "gray40" # emmval = 0 -> dark gray
ppom$data[[3]]$colour <- ppom$data[[4]]$colour <- "gray80" # emmval = 1 -> light gray
ppom$data[[5]]$colour <- ppom$data[[6]]$colour <- "black" # emmval = 2 -> black
xincrement = 0.025
ppom$data[[1]]$x <- ppom$data[[2]]$x <- ppom$data[[1]]$x - xincrement
ppom$data[[2]]$xmin <- ppom$data[[2]]$xmin - xincrement
ppom$data[[2]]$xmax <- ppom$data[[2]]$xmax - xincrement
ppom$data[[5]]$x <- ppom$data[[6]]$x <- ppom$data[[5]]$x + xincrement
ppom$data[[6]]$xmin <- ppom$data[[6]]$xmin + xincrement
ppom$data[[6]]$xmax <- ppom$data[[6]]$xmax + xincrement
plot(ggplot_gtable(ppom))

## Not run:
library(gtable) # may need to separately install gtable
# example with no bootstrapping, adding object from bootstrapped fit
pp2 <- plot(qfit, emmval = 1, suppressprint=TRUE)
grid.draw(pp2)
# insert row on top that is 1/2 height of existing plot
pp2b = gtable::gtable_add_rows(pp2, heights=unit(0.5, 'null') ,pos = 0)
# add plot to that row
pp3 = gtable::gtable_add_grob(pp2b, ggplot2::ggplotGrob(pp), t=1,l=1,r=2)
grid.draw(pp3)
```

```
## End(Not run)
```

`pointwisebound` *Estimating pointwise comparisons for qgcompint fits*

Description

Calculates: expected outcome (on the link scale), mean difference (link scale) and the standard error of the mean difference (link scale) for pointwise comparisons

Usage

```
pointwisebound(x, alpha = 0.05, pointwiseref = 1, emmval = 0, ...)
```

Arguments

`x`	qgcompemmfit object from qgcomp.emm.noboot
`alpha`	alpha level for confidence intervals
`pointwiseref`	referent quantile (e.g. 1 uses the lowest joint-exposure category as the referent category for calculating all mean differences/standard deviations)
`emmval`	fixed value for effect measure modifier at which pointwise comparisons are calculated
`...`	not used

Details

The comparison of interest following a qgcomp fit is often comparisons of model predictions at various values of the joint-exposures (e.g. expected outcome at all exposures at the 1st quartile vs. the 3rd quartile). The expected outcome at a given joint exposure and at a given level of non-exposure covariates (W) is given as E(Y|S,W=w), where S takes on integer values 0 to q-1. Thus, comparisons are of the type E(Y|S=s,W=w) - E(Y|S=s2,W=w) where s and s2 are two different values of the joint exposures (e.g. 0 and 2). This function yields E(Y|S,W=w) as well as E(Y|S=s,W=w) - E(Y|S=p,W=w) where s is any value of S and p is the value chosen via "pointwise ref" - e.g. for binomial variables this will equal the risk/ prevalence difference at all values of S, with the referent category S=p-1. For the non-boostrapped version of quantile g-computation (under a linear model) Note that function only works with standard "qgcompint" objects from qgcomp.emm.noboot (so it doesn't work with zero inflated, hurdle, or Cox models) Variance for the overall effect estimate is given by: $transpose(G)Cov(\beta)G$ Where the "gradient vector" G is given by

$$G = [\partial(f(\beta))/\partial\beta_1 = 1, ..., \partial(f(\beta))/\partial\beta_3 = 1]$$

$f(\beta) = \sum_i^p \beta_i, and \partial y/\partial x$ denotes the partial derivative/gradient. The vector G takes on values that equal the difference in quantiles of S for each pointwise comparison (e.g. for a comparison of the 3rd vs the 5th category, G is a vector of 2s) This variance is used to create pointwise confidence intervals via a normal approximation: (e.g. upper 95% CI = psi + variance*1.96)

Value

A data frame containing

hx: The "partial" linear predictor $\beta_0 + \psi \sum_j X_j^q w_j$, or the effect of the mixture + intercept after conditioning out any confounders. This is similar to the h(x) function in bkmr. This is not a full prediction of the outcome, but only the partial outcome due to the intercept and the confounders

rr/or/mean.diff: The canonical effect measure (risk ratio/odds ratio/mean difference) for the marginal structural model link

se....: the stndard error of the effect measure

ul..../ll....: Confidence bounds for the effect measure

See Also

qgcomp.emm.noboot, pointwisebound.noboot

Examples

```
set.seed(50)
# linear model, binary modifier
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat, q=4, family=gaussian()))
pointwisebound(qfit, pointwiseref = 2, emmval = 0.1)
# linear model, categorical modifier
dat3 <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
z=as.factor(sample(0:2, 50,replace=TRUE)), r=rbinom(50,1,0.5))
(qfit3 <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
expnms = c('x1', 'x2'), data=dat3, q=5, family=gaussian()))
pointwisebound(qfit3, pointwiseref = 2, emmval = 0)
pointwisebound(qfit3, pointwiseref = 2, emmval = 1)
pointwisebound(qfit3, pointwiseref = 2, emmval = 2)
# linear model, categorical modifier, bootstrapped
# set B larger for real examples
(qfit3b <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
expnms = c('x1', 'x2'), data=dat3, q=5, family=gaussian(), B=10))
pointwisebound(qfit3b, pointwiseref = 2, emmval = 0)
pointwisebound(qfit3b, pointwiseref = 2, emmval = 1)
pointwisebound(qfit3b, pointwiseref = 2, emmval = 2)
# logistic model, binary modifier
dat4 <- data.frame(y=rbinom(50, 1, 0.3), x1=runif(50), x2=runif(50),
  z=as.factor(sample(0:1, 50,replace=TRUE)), r=rbinom(50,1,0.5))
(qfit4 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
expnms = c('x1', 'x2'), data=dat4, q=5, family=binomial(), B=10))
pointwisebound(qfit4, pointwiseref = 2, emmval = 0) # reverts to odds ratio
```

print.qgcompemmfit *Default printing method for a qgcompemmfit object*

Description

Prints output depending for `qgcomp.emm.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).

Usage

```
## S3 method for class 'qgcompemmfit'
print(x, showweights = TRUE, ...)
```

Arguments

`x`	"qgcompemmfit" object from `qgcomp.emm.noboot` function
`showweights`	logical: should weights be printed, if estimated?
`...`	unused

Value

Invisibly returns x. Called primarily for side effects.

See Also

`qgcomp.emm.noboot`, `getstratweights`

qgcomp.emm.boot *EMM for Quantile g-computation for continuous, binary, and count outcomes under linearity/additivity*

Description

This function fits a quantile g-computation model, allowing effect measure modification by a binary or continuous covariate. This allows testing of statistical interaction as well as estimation of stratum specific effects. This particular implementation formally fits a marginal structural model using a Monte Carlo-based g-computation method, utilizing bootstrapping for variance estimates. Because this approach allows for non-linear/non-additive effects of exposures, it does not report weights nor EMM stratum specific effects.

Usage

```
qgcomp.emm.boot(
  f,
  data,
  expnms = NULL,
  emmvar = "",
  q = 4,
  breaks = NULL,
  id = NULL,
  weights,
  alpha = 0.05,
  B = 200,
  rr = TRUE,
  degree = 1,
  seed = NULL,
  bayes = FALSE,
  MCsize = nrow(data),
  parallel = FALSE,
  parplan = FALSE,
  errcheck = FALSE,
  ...
)
```

Arguments

f	R style formula
data	data frame
expnms	character vector of exposures of interest
emmvar	(character) name of effect measure modifier in dataset (if categorical, must be coded as a factor variable)
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). Note that qgcomp.emm.noboot will not produce cluster-appropriate standard errors (this parameter is essentially ignored in qgcomp.emm.noboot). Qgcomp.emm.boot can be used for this, which will use bootstrap sampling of clusters/individuals to estimate cluster-appropriate standard errors via bootstrapping.
weights	"case weights" - passed to the "weight" argument of glm or bayesglm
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 bases for marginal model (e.g. degree = 2 allows that the relationship between the whole exposure mixture and the outcome is quadratic (default = 1).
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 interpreted as frequentist.
MCsize	integer: sample size for simulation to approximate marginal zero inflated model parameters. This can be left small for testing, but should be as large as needed to reduce simulation error to an acceptable magnitude (can compare psi coefficients for linear fits with qgcomp.noboot to gain some intuition for the level of expected simulation error at a given value of MCsize). This likely won't matter much in linear models, but may be important with binary or count outcomes.
parallel	use (safe) parallel processing from the future and future.apply packages
parplan	(logical, default=FALSE) automatically set future::plan to plan(multisession) (and set to existing plan after bootstrapping)
errcheck	(logical, default=TRUE) include some basic error checking. Slightly faster if set to false (but be sure you understand risks)
...	arguments to glm (e.g. family)

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 (neg.weights) directions.

See Also

qgcomp.noboot

Examples

```
set.seed(50)
# linear model, binary modifier
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
  z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat, q=4, family=gaussian()))
# set B larger for real examples
(qfit2 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
  degree = 1,
```

```
  expnms = c('x1', 'x2'), data=dat, q=4, family=gaussian(), B=10))
# categorical modifier
dat2 <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
  z=sample(0:2, 50,replace=TRUE), r=rbinom(50,1,0.5))
dat2$z = as.factor(dat2$z)
(qfit3 <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat2, q=4, family=gaussian()))
# set B larger for real examples
(qfit4 <- qgcomp.emm.boot(f=y ~ z + x1 + x2, emmvar="z",
  degree = 1,
  expnms = c('x1', 'x2'), data=dat2, q=4, family=gaussian(), B=10))
```

`qgcomp.emm.cox.noboot` *EMM for Quantile g-computation with survival outcomes under linearity/additivity*

Description

This function performs quantile g-computation in a survival setting, , allowing effect measure modification by a binary, categorical or continuous covariate. This allows testing of statistical interaction as well as estimation of stratum specific effects.

Usage

```
qgcomp.emm.cox.noboot(
  f,
  data,
  expnms = NULL,
  emmvar = NULL,
  q = 4,
  breaks = NULL,
  id = NULL,
  weights,
  cluster = NULL,
  alpha = 0.05,
  errcheck = TRUE,
  ...
)
```

Arguments

`f`	R style survival formula, which includes `Surv` in the outcome definition. E.g. `Surv(time,event) ~ exposure`. Offset terms can be included via `Surv(time,event) ~ exposure + offset(z)`
`data`	data frame
`expnms`	character vector of exposures of interest

emmvar	(character) name of effect measure modifier in dataset (if categorical, must be coded as a factor variable)
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)
weights	"case weights" - passed to the "weight" argument of coxph
cluster	not yet implemented
alpha	alpha level for confidence limit calculation
errcheck	(logical, default=TRUE) include some basic error checking. Slightly faster if set to false (but be sure you understand risks)
...	arguments to glm (e.g. family)

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 (neg.weights) directions.

See Also

qgcomp.cox.noboot

Examples

```
set.seed(5)
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+z
(fit1 <- survival::coxph(f, data = dat))
(obj <- qgcomp.emm.cox.noboot(f, expnms = expnms, emmvar="z", data = dat))

#categorical emm
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=sample(0:2, N, replace=TRUE))
 dat$z = as.factor(dat$z)
expnms=paste0("x", 1:2)
f = survival::Surv(time, d)~x1 + x2+z
(obj2 <- qgcomp.emm.cox.noboot(f, expnms = expnms, emmvar="z", data = dat))
```

`qgcomp.emm.noboot` *EMM for Quantile g-computation for continuous, binary, and count outcomes under linearity/additivity*

Description

This function fits a quantile g-computation model, allowing effect measure modification by a binary or continuous covariate. This allows testing of statistical interaction as well as estimation of stratum specific effects.

Usage

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

Arguments

`f`	R style formula
`data`	data frame
`expnms`	character vector of exposures of interest
`emmvar`	(character) name of effect measure modifier in dataset (if categorical, must be coded as a factor variable)
`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). Note that qgcomp.noboot will not produce cluster-appropriate standard errors (this parameter is essentially ignored in qgcomp.noboot). Qgcomp.boot can be used

	for this, which will use bootstrap sampling of clusters/individuals to estimate cluster-appropriate standard errors via bootstrapping.
weights	"case weights" - passed to the "weight" argument of glm or bayesglm
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 interpreted as frequentist.
errcheck	(logical, default=TRUE) include some basic error checking. Slightly faster if set to false (but be sure you understand risks)
...	arguments to glm (e.g. family)

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 (neg.weights) directions.

See Also

qgcomp.noboot

Examples

```
set.seed(50)
# linear model, binary modifier
dat <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
  z=rbinom(50,1,0.5), r=rbinom(50,1,0.5))
(qfit <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat, q=2, family=gaussian()))
# logistic model, continuous modifier
dat2 <- data.frame(y=rbinom(50, 1,0.5), x1=runif(50), x2=runif(50),
  z=runif(50), r=rbinom(50,1,0.5))
(qfit2 <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat2, q=2, family=binomial()))
# get weights and stratum specific effects at specific value of Z
#  (note that when Z=0, the effect is equal to psi1)
qgcompint::getstratweights(qfit2,emmval=0)
qgcompint::getstrateffects(qfit2,emmval=0)
qgcompint::getstratweights(qfit2,emmval=0.5)
qgcompint::getstrateffects(qfit2,emmval=0.5)
# linear model, categorical modifier
dat3 <- data.frame(y=runif(50), x1=runif(50), x2=runif(50),
  z=as.factor(sample(0:2, 50,replace=TRUE)), r=rbinom(50,1,0.5))
(qfit3 <- qgcomp.emm.noboot(f=y ~ z + x1 + x2, emmvar="z",
  expnms = c('x1', 'x2'), data=dat3, q=2, family=gaussian()))
# get weights and stratum specific effects at each value of Z
#  (note that when Z=0, the effect is equal to psi1)
qgcompint::getstratweights(qfit3,emmval=0)
```

```
qgcompint::getstrateffects(qfit3,emmval=0)
qgcompint::getstratweights(qfit3,emmval=1)
qgcompint::getstrateffects(qfit3,emmval=1)
qgcompint::getstratweights(qfit3,emmval=2)
qgcompint::getstrateffects(qfit3,emmval=2)
```

`simdata_quantized_emm` *Simulate quantized exposures for testing methods*

Description

Simulate quantized exposures for testing methods

Usage

```
simdata_quantized_emm(
  outcometype = c("continuous", "logistic", "survival"),
  n = 100,
  corr = NULL,
  b0 = 0,
  mainterms = c(1, 0, 0, 0),
  prodterms = c(1, 0, 0, 0),
  ztype = "binary",
  q = 4,
  yscale = 1,
  shape0 = 3,
  scale0 = 5,
  censtime = 4,
  ncheck = TRUE,
  ...
)
```

Arguments

`outcometype`	Character variable that is one of c("continuous", "logistic", "survival"). Selects what type of outcome should be simulated (or how). continuous = normal, continous outcome, logistic= binary outcome from logistic model, survival = right censored survival outcome from Weibull model.
`n`	Sample size
`corr`	NULL, or vector of correlations between the first exposure and subsequent exposures (if length(corr) < (length(coef)-1), then this will be backfilled with zeros)
`b0`	(continuous, binary outcomes) model intercept
`mainterms`	beta coefficients for X in the outcome model at referent (0) level of interacting variable
`prodterms`	product term coefficients for interacting variable
`ztype`	type of interacting variable: "continuous", "binary", "categorical"

q	Number of levels or "quanta" of each exposure
yscale	(continuous outcomes) error scale (residual error) for normally distributed outcomes
shape0	(survival outcomes) baseline shape of weibull distribution rweibull
scale0	(survival outcomes) baseline scale of weibull distribution rweibull
censtime	(survival outcomes) administrative censoring time
ncheck	(logical, default=TRUE) adjust sample size if needed so that exposures are exactly evenly distributed (so that qgcomp::quantize(exposure) = exposure)
...	unused

Details

Simulate continuous (normally distributed errors), binary (logistic function), or event-time outcomes as a linear function

Value

a data frame

See Also

qgcomp.boot, and qgcomp.noboot

Examples

```
set.seed(50)
qdat = simdata_quantized_emm(
  outcomtype="continuous",
  n=10000, corr=c(.9,.3,0,0), mainterms=c(1,1,0,0), prodterms=c(1,1,0,0),
  q = 8
)
cor(qdat)
qdat = simdata_quantized_emm(
  outcomtype="continuous",
  n=10000, corr=c(-.9,.3,0,0), mainterms=c(1,2,0,0), prodterms=c(1,1,0,0),
  q = 4
)
cor(qdat)
table(qdat$x1)
qgcomp.emm.noboot(y~x1+x2+x3+x4,expnms = c("x1", "x2", "x3", "x4"), emmvar = "z", data=qdat)
```

Index