Package ‘WEE’

November 15, 2016

Type Package

Title Weighted Estimated Equation (WEE) Approaches in Genetic Case-Control Studies

Version 1.0

Date 2016-11-05

Author Xiaoyu Song, Iuliana Ionita-Laza, Mengling Liu, Joan Reitman, Ying Wei

Maintainer Wodan Ling <wl2459@columbia.edu>

Description Secondary analysis of case-control studies using a weighted estimating equation (WEE) approach: logistic regression for binary secondary outcomes, linear regression and quantile regression for continuous secondary outcomes.

License GPL-2

Imports quantreg, doParallel, foreach, parallel

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-15 02:19:43

R topics documented:

WEE-package *Weighted Estimated Equation (WEE) Approaches in Genetic Case-Control Studies*

Description

Secondary analysis of case-control studies using a weighted estimating equation (WEE) approach: logistic regression for binary secondary outcomes, linear regression and quantile regression for continuous secondary outcomes.

Author(s)

Xiaoyu Song, Iuliana Ionita-Laza, Mengling Liu, Joan Reitman, Ying Wei

Maintainer: Wodan Ling <wl2459@columbia.edu>

References

[1] Ying Wei, Xiaoyu Song, Mengling Liu, Iuliana Ionita-Laza and Joan Reibman (2016). Quantile Regression in the Secondary Analysis of Case Control Data. *Journal of the American Statistical Association*, 111:513, 344-354; DOI: 10.1080/01621459.2015.1008101

[2] Xiaoyu Song, Iuliana Ionita-Laza, Mengling Liu, Joan Reibman, Ying Wei (2016). A General and Robust Framework for Secondary Traits Analysis. *Genetics*, vol. 202 no. 4 1329-1343; DOI: 10.1534/genetics.115.181073

Examples

```
#---------------------- WEE logistic regression ----------------------#
## Generate simulated data
# set population size as 500000
n = 500000

# set parameters
beta = c(0.2, 0.1) # P(Y|X,Z)
gamma = c(0.3, log(2), log(2)) #P(D|X,Y,Z)

# generate the genetic variant X
x = rbinom(n,size=2,prob=0.3)

# generate the standardized continuous covariate Z correlated with X
z = rnorm(n, mean=0.5*x-0.3, sd=1)

# generate the binary secondary trait Y
```

```
py = exp(-1+beta[1]*x+beta[2]*z)/
        (1+exp(-1+beta[1]*x+beta[2]*z))
y = rbinom(n,1, py)

# generate the primary disease D
# (alpha changes to make sure the disease prevalence = 0.1 )
alpha = -2.88
pd = exp(alpha+x*gamma[1]+y*log(2)+z*log(2))/
(1+exp(alpha+x*gamma[1]+y*log(2)+z*log(2)))
d = rbinom(n,size=1,prob=pd)

# form the population dataset
dat = as.data.frame(cbind(d, y, z, x))

# generate sample dataset with 200 cases and 200 controls
dat_cases = dat[which(dat$d==1),]
dat_controls= dat[which(dat$d==0),]
dat_cases_sample = dat_cases[sample(sum(dat$d==1),
                             200,replace=FALSE),]
dat_controls_sample = dat_controls[sample(sum(dat$d==0),
                                   200,replace=FALSE),]

dat_logistic = rbind(dat_cases_sample,dat_controls_sample)
colnames(dat_logistic) = c("D", "y", "z","x")
D = dat_logistic$D # Disease status
pD = sum(dat$d==1)/500000 # Population disease prevalence

## WEE logsitic regression
WEE.logistic(y ~ x + z, D,
             data = dat_logistic, pD)

WEE.logistic(y ~ x + z, D,
             data = dat_logistic, pD, boot = 500)



#---------------------- WEE linear regression ----------------------#
## Generate simulated data
# set population size as 500000
n = 500000

# set parameters
beta = c(0.2, 0.1) # P(Y|X,Z)
gamma = c(0.3, log(2), log(2)) #P(D|X,Y,Z)

# generate the genetic variant X
x = rbinom(n,size=2,prob=0.3)

# generate the standardized continuous covariate Z correlated with X
z = rnorm(n, mean=0.5*x-0.3, sd=1)

# generate the continuous secondary trait Y
y = 1+beta[1]*x+beta[2]*z+rnorm(n)
```

```
# generate the primary disease D
alpha = -3.62
pd = exp(alpha + x*gamma[1] + y*log(2) + z*log(2))/
(1 + exp(alpha + x*gamma[1] + y*log(2) + z*log(2)))
d = rbinom(n,size=1,prob=pd)

# form population data set
dat=as.data.frame(cbind(d, y, z, x))

# generate sample dataset with 200 cases and 200 controls
dat_cases = dat[which(dat$d==1),]
dat_controls= dat[which(dat$d==0),]
dat_cases_sample = dat_cases[sample(sum(dat$d==1),
                             200, replace=FALSE),]
dat_controls_sample = dat_controls[sample(sum(dat$d==0),
                                   200, replace=FALSE),]

dat_linear=rbind(dat_cases_sample,dat_controls_sample)
colnames(dat_linear)=c("D", "y", "z","x")
D = dat_linear$D # Disease status
pD = sum(dat$d == 1)/500000 # Population disease prevalence

## WEE linear regresssion
WEE.linear(y ~ x + z, D,
           data = dat_linear, pD)

WEE.linear(y ~ x + z, D,
          data = dat_linear, pD, boot = 500)



#---------------------- WEE quantile regression ----------------------#
## Generate simulated data
# set population size as 500000
n = 500000

# set parameters
beta = c(0.12, 0.1) # P(Y|X,Z)
gamma = c(-4, log(1.5), log(1.5), log(2) ) #P(D|X,Y,Z)

# generate the genetic variant X
x = rbinom(n,size=2,prob=0.3)

# generate the continuous covariate Z
z = rnorm(n)

# generate the continuous secondary trait Y
y= 1 + beta[1]*x + beta[2]*z + (1+0.02*x)*rnorm(n)

# generate disease status D
p = exp(gamma[1]+x*gamma[2]+z*gamma[3]+y*gamma[4])/
(1+exp(gamma[1]+x*gamma[2]+z*gamma[3]+y*gamma[4]))
```

```
d = rbinom(n,size=1,prob=p)

# form population data dataset
dat = as.data.frame(cbind(x,y,z,d))
colnames(dat) = c("x","y","z","d")

# Generate sample dataset with 200 cases and 200 controls
dat_cases = dat[which(dat$d==1),]
dat_controls= dat[which(dat$d==0),]
dat_cases_sample = dat_cases[sample(sum(dat$d==1),
                            200, replace=FALSE),]
dat_controls_sample = dat_controls[sample(sum(dat$d==0),
                                   200, replace=FALSE),]

dat_quantile = as.data.frame(rbind(dat_cases_sample,
                                   dat_controls_sample))
colnames(dat_quantile) = c("x","y","z","D")
D = dat_quantile$D # Disease status
pd = sum(d==1)/n # population disease prevalence

# WEE quantile regressions:
WEE.quantile(y ~ x, D, tau = 0.5,
             data = dat_quantile, pd_pop = pd)

WEE.quantile(y ~ x + z, D, tau = 1:9/10,
             data = dat_quantile, pd_pop = pd, boot = 500)
```

---

plot.predict.WEE.quantile

Plot predicted quantiles of WEE.quantile regression fit

Description

Plot the predicted quantiles and their point-wise confidence intervals of a WEE-quantile fit on new dataset.

Usage

```
## S3 method for class 'predict.WEE.quantile'
plot(x, CI = FALSE, level = 0.95, index = 1, ...)
```

Arguments

`x`	object produced by `predict.WEE.quantile`.
`CI`	logical flag indicating whether to plot confidence interval: default is FALSE; if TRUE the function not only plots point predictions for each of the 'newdata' points but also lower and upper confidence limits. Only set TRUE when boot > 0.

`level`	confidence level.
`index`	a vector to indicate the subset of newx to be plotted. Default is 1, i.e. the first combination of newx.
`...`	further graphical parameters passed to `plot`.

See Also

`predict.WEE.quantile`

Examples

```
## continued from predict.WEE.quantile
## Plot prediction without confidence interval
plot(p1,index = c(2,3))

## Plot prediction with confidence interval
plot(p2, CI = TRUE)
```

`plot.WEE.quantile` *Plot coefficients estimated from WEE.quantile*

Description

Plot the estimated quantile coefficients and their pointwise confidence intervals from WEE.quantile regression

Usage

```
## S3 method for class 'WEE.quantile'
plot(x, CI = FALSE, level = 0.95, index = 1, ...)
```

Arguments

`x`	object produced by `WEE.quantile`.
`CI`	logical flag indicating whether to plot confidence interval: default is FALSE; if TRUE the pointwise confidence interval is plotted. Only set TRUE when boots > 0 in the WEE.quantile fitting process.
`level`	confidence level.
`index`	a vector to indicate the subset of coefficients to be plotted (e.g., 2 indicates the coefficient of the first covariate, 3 indicates the coefficient of the second covaraite). Default is 1, i.e. the intercept.
`...`	further graphical parameters passed to `plot`.

See Also

`WEE.quantile`

Examples

```
## continued from WEE.quantile
## plot fitted model without pointwise confidence interval
plot(WEE.quantile(y ~ x, D, tau = 0.5,
                  data = dat_quantile, pd_pop = pd),index = c(2,3))

## plot fitted model with pointwise confidence interval
plot(WEE.quantile(y ~ x + z, D, tau = c(0.25,0.5),
                  data = dat_quantile, pd_pop = pd, boot = 500),
                  CI = TRUE)
```

`predict.WEE.linear` *WEE Linear Regression Prediction*

Description

Prediction on new dataset based on model fitted by WEE linear regression

Usage

```
## S3 method for class 'WEE.linear'
predict(object,newx, ...)
```

Arguments

`object`	Object produced by `WEE.linear`.
`newx`	A data matrix in which to look for variables with which to predict, newx cannot be omitted.
`...`	Further arguments passed to or from other methods.

Details

Produces predicted values, obtained by evaluating the WEE linear regression function on newx.

Value

If in the WEE.linear fitting procedure boot = 0, only point predictions are provided here. If in the WEE.linear fitting procedurep boot > 0, standard errors of prediction are also provided.

See Also

`WEE.linear`

Examples

```
## continued from WEE.linear
## predict outcome y based on newx
newx = dat[sample(500000,3, replace=F),][,c("x","z")]
predict(WEE.linear(y ~ x + z, D,
data = dat_sample, pD),newx)
predict(WEE.linear(y ~ x + z, D,
data = dat_sample, pD, boot = 500),newx)
```

`predict.WEE.logistic` *WEE logistic Regression Prediction*

Description

Prediction on new dataset based on model fitted by WEE logistic regression

Usage

```
## S3 method for class 'WEE.logistic'
predict(object,newx, ...)
```

Arguments

`object`	Object produced by `WEE.logistic`.
`newx`	A data matrix in which to look for variables with which to predict, newx cannot be omitted
`...`	Further arguments passed to or from other methods.

Details

Produces predicted values, obtained by evaluating the WEE logistic regression function on newx.

Value

If in the WEE.logistic fitting procedure boot = 0, linear predictor and predicted response of each newx are given. If in the WEE.logistic fitting procedure boot > 0, standard errors of linear predictor and predicted response are given.

See Also

`WEE.logistic`

Examples

```
## continued from WEE.logistic
## predict outcome y based on newx
newx = dat[sample(500000,3, replace=FALSE),][,c("x","z")]
predict(WEE.logistic(y ~ x + z, D,
        data = dat_sample, pD),newx)
predict(WEE.logistic(y ~ x + z, D,
        data = dat_sample, pD, boot = 500),newx)
```

---

predict.WEE.quantile *WEE quantile Regression Prediction*

Description

Prediction on new dataset based on model fitted by WEE quantile regression

Usage

```
## S3 method for class 'WEE.quantile'
predict(object,newx, ...)
```

Arguments

object	Object produced by WEE.quantile.
newx	A new data matrix in which to look for data with which to predict, newx cannot be omitted.
...	Further arguments passed to or from other methods.

Details

Produces predicted values, obtained by evaluating the WEE quantile regression function on newx.

Value

If in the WEE.quantile fitting procedure boot = 0, only point predictions are given. If in the WEE.quantile fitting procedure boot > 0, standard errors of prediction are also given.

See Also

WEE.quantile

Examples

```
## continued from WEE.quantile
## prediction based on newx
newx = dat[sample(500000,3, replace=F),][,c("x")]
p1 = predict(WEE.quantile(y ~ x, D, tau = 0.5,
     data = dat_quantile, pd_pop = pd),newx)
p1

newx = dat[sample(500000,3, replace=F),][,c("x","z")]
p2 = predict(WEE.quantile(y ~ x + z, D, tau = c(0.25,0.5),
     data = dat_quantile, pd_pop = pd, boot = 500),newx)
p2
```

`print.summary.WEE.linear`

Print WEE Linear Summary Object

Description

Print summary of WEE linear regression object

Usage

```
## S3 method for class 'summary.WEE.linear'
print(x, ...)
```

Arguments

`x`	An object of class "summary.WEE.linear" produced by a call to summary.WEE.quantile()
`...`	Optional arguments passed to printing function

See Also

`summary.WEE.linear`

print.summary.WEE.logistic

Print WEE logistic Summary Object

Description

Print summary of WEE logistic regression object

Usage

```
## S3 method for class 'summary.WEE.logistic'
print(x, ...)
```

Arguments

x	An object of class "summary.WEE.logistic" produced by a call to summary.WEE.quantile()
...	Optional arguments passed to printing function

See Also

summary.WEE.logistic

print.summary.WEE.quantile

Print WEE quantile Summary Object

Description

Print summary of WEE quantile regression object

Usage

```
## S3 method for class 'summary.WEE.quantile'
print(x, ...)
```

Arguments

x	An object of class "summary.WEE.quantile" produced by a call to summary.WEE.quantile()
...	Optional arguments passed to printing function

See Also

summary.WEE.quantile

`print.WEE.linear` *Print a WEE.linear object*

Description

Print an object generated by WEE.linear

Usage

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

Arguments

`x`	Object returned from WEE.linear representing the fit of the model
`...`	Optional arguments passed to printing function

See Also

`WEE.linear`

`print.WEE.logistic` *Print a WEE.linear object*

Description

Print an object generated by WEE.logistic

Usage

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

Arguments

`x`	Object returned from WEE.logistic representing the fit of the model
`...`	Optional arguments passed to printing function

See Also

`WEE.logistic`

`print.WEE.quantile` *Print a WEE.linar object*

Description

Print an object generated by WEE.quantile

Usage

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

Arguments

`x`	Object returned from WEE.quantile representing the fit of the model
`...`	Optional arguments passed to printing function

See Also

`WEE.quantile`

`summary.WEE.linear` *Summary methods for WEE linear Regression*

Description

Returns a summary list for a WEE linear regression fit.

Usage

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

Arguments

`object`	object produced by `WEE.linear`.
`...`	further arguments passed to or from other methods.

Value

a list is returned with the following components.

`Coefficients`	a vector of coefficients
`StdErr`	bootstrap standard errors of the coefficients, only returned when boot > 0
`Chisq`	Chi-squared test statistics of the coefficients, only returned when boot > 0
`p.value`	p-values of the chi-squared test statistics, only returned when boot > 0
`Covariance`	the estimated covariance matrix of the coefficients in the model, provided that boot > 0 in the called sequence.

See Also

`WEE.linear`

Examples

```
## continued from WEE.linear
## summary of WEE linear object
summary(WEE.linear(y ~ x + z, D,
        data = dat_sample, pd_pop = pD))
summary(WEE.linear(y ~ x + z, D,
        data = dat_sample, pd_pop = pD, boot=500))
```

`summary.WEE.logistic` *Summary methods for WEE logistic Regression*

Description

Returns a summary list for a WEE logistic regression fit.

Usage

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

Arguments

`object`	object produced by `WEE.logistic`.
`...`	further arguments passed to or from other methods.

Value

a list is returned with the following components.

`Coefficients`	a vector of coefficients
`Oddsratio`	the exponentiated coefficients, namely the odds ratio associated with the corresponding covariate
`StdErr`	bootstrap standard errors of the coefficients, only returned when boot > 0
`Wald`	Wald test statistics of the coefficients, only returned when boot > 0
`p.value`	p-values of the Wald test statistics, only returned when boot > 0
`Covariance`	the estimated covariance matrix for the coefficients in the model, provided that boot > 0 in the called sequence

See Also

`WEE.logistic`

Examples

```
## continued from WEE.logistic
## summary of WEE logistic object
summary(WEE.logistic(y ~ x + z, D,
        data = dat_sample, pd_pop = pD))
summary(WEE.logistic(y ~ x + z, D,
        data = dat_sample, pd_pop = pD, boot=500))
```

`summary.WEE.quantile` *Summary methods for WEE Quantile Regression*

Description

Returns a summary list for a WEE quantile regression fit.

Usage

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

Arguments

`object`	object produced by `WEE.quantile`.
`...`	further arguments passed to or from other methods.

Value

a list is returned with the following components.

`Coefficients`	a vector of coefficients
`StdErr`	bootstrap standard errors of the coefficients, only returned when boot > 0
`Wald`	Wald test statistics of the coefficients, only returned when boot > 0
`p.value`	p-values of the Wald test statistics, only returned when boot > 0
`Covariance`	the estimated covariance matrix for the coefficients in the model, provided that boot > 0 in the called sequence

See Also

`WEE.quantile`

Examples

```
## continued from WEE.quantile
summary(WEE.quantile(y ~ x, D, tau = 0.5,
                     data = dat_quantile, pd_pop = pd))
summary(WEE.quantile(y ~ x + z, D, tau = c(0.25,0.5),
                     data = dat_quantile, pd_pop = pd, boot=500))
```

`WEE.linear` *WEE linear regression*

Description

Returns an object of class "WEE.linear" that is generated by linear regression with WEE approach for continuous secondary traits in genetic case-control studies.

Usage

```
WEE.linear(formula, D, data, pd_pop, boot = 0, ...)
```

Arguments

`formula`	the secondary trait given SNPs and covariates. e.g. y~x+z
`D`	primary disease (case-control status)
`data`	dataset with real observation.
`pd_pop`	the population disease prevelance of primary disease.
`boot`	number of bootstrap samples. (boot=0 by default)
`...`	optional arguments to be passed through to lm.

Value

`Coefficients`	Point estimates
`StdErr`	Bootstrap standard errors, returned if boot > 0
`Chisq`	Chi-squared test statistics, returned if boot > 0
`p.value`	p-values, returned if boot > 0
`Covariance`	Covariance matrix, returned if boot > 0

Warning

If boot = 0, point estimates are plotted. If boot > 0, boostrap standard errors, chisquare test statistics, p-values, and covariance matrix are also returned. Optional arguments from lm can be passed to this function, but arguments 'subset' and 'weights' should be used with caution.

References

Xiaoyu Song, Iuliana Ionita-Laza, Mengling Liu, Joan Reibman, Ying Wei (2016). A General and Robust Framework for Secondary Traits Analysis. *Genetics*, vol. 202 no. 4 1329-1343; DOI: 10.1534/genetics.115.181073

Examples

```
## Generate simulated data
# set population size as 500000
n = 500000

# set parameters
beta = c(0.2, 0.1) # P(Y|X,Z)
gamma = c(0.3, log(2), log(2)) #P(D|X,Y,Z)

# generate the genetic variant X
x = rbinom(n,size=2,prob=0.3)

# generate the standardized continuous covariate Z correlated with X
z = rnorm(n, mean=0.5*x-0.3, sd=1)

# generate the continuous secondary trait Y
y = 1+beta[1]*x+beta[2]*z+rnorm(n)

# generate the primary disease D
alpha = -3.62
pd = exp(alpha + x*gamma[1] + y*log(2)+ z*log(2)) /
         (1+exp(alpha+ x*gamma[1] + y*log(2)+ z*log(2)))
d = rbinom(n,size=1,prob=pd)

# form population data set
dat=as.data.frame(cbind(d, y, z, x))

# generate sample dataset with 200 cases and 200 controls
dat_cases = dat[which(dat$d==1),]
dat_controls= dat[which(dat$d==0),]
dat_cases_sample = dat_cases[sample(sum(dat$d==1),
                             200, replace=FALSE),]
dat_controls_sample = dat_controls[sample(sum(dat$d==0),
                                  200, replace=FALSE),]

dat_linear=rbind(dat_cases_sample,dat_controls_sample)
colnames(dat_linear)=c("D", "y", "z","x")
D = dat_linear$D # Disease status
pD = sum(dat$d == 1)/500000 # Population disease prevalence

## WEE linear regresssion
WEE.linear(y ~ x + z, D,
           data = dat_linear, pD)

WEE.linear(y ~ x + z, D,
           data = dat_linear, pD, boot = 500)
```

WEE.logistic *WEE logistic regression*

Description

Returns an object of class "WEE.logistic" that is generated by logistic regression with WEE approach for binary secondary traits in genetic case-control studies.

Usage

```
WEE.logistic(formula, D, data, pd_pop, iter = 5, boot = 0, ...)
```

Arguments

`formula`	The secondary trait given SNPs and covariates. e.g. y~x+z
`D`	Primary disease (case-control status)
`data`	Dataset with real observation.
`pd_pop`	The population disease prevelance of primary disease.
`iter`	Number of generating pseudo observations. (iter=10 by default)
`boot`	Number of bootstrape samples. (boot=0 by default)
`...`	Optional arguments to be passed through to glm.

Value

`Coefficients`	Point estimates
`Oddsratio`	The exponentiated coefficients, namely the odds ratio associated with the corresponding covariate
`StdErr`	Bootstrap standard errors, returned if boot > 0
`Wald`	Wald test statistics, returned if boot > 0
`p.value`	p-values, returned if boot > 0
`Covariance`	Covariance matrix, returned if boot > 0

Warning

If boot = 0, point estimates are plotted. If boot > 0, boostrap standard errors, Wald test statistics, p-values, and covariance matrix are also returned. Optional arguments from glm can be passed to this function, but arguments 'subset' and 'weights' should be used with caution.

References

Xiaoyu Song, Iuliana Ionita-Laza, Mengling Liu, Joan Reibman, Ying Wei (2016). A General and Robust Framework for Secondary Traits Analysis. *Genetics*, vol. 202 no. 4 1329-1343; DOI: 10.1534/genetics.115.181073

Examples

```
## Generate simulated data
# set population size as 500000
n = 500000

# set parameters
beta = c(0.2, 0.1) # P(Y|X,Z)
gamma = c(0.3, log(2), log(2)) #P(D|X,Y,Z)

# generate the genetic variant X
x = rbinom(n,size=2,prob=0.3)

# generate the standardized continuous covariate Z correlated with X
z = rnorm(n, mean=0.5*x-0.3, sd=1)

# generate the binary secondary trait Y
py = exp(-1+beta[1]*x+beta[2]*z)/
         (1+exp(-1+beta[1]*x+beta[2]*z))
y = rbinom(n,1, py)

# generate the primary disease D
# (alpha changes to make sure the disease prevalence = 0.1 )
alpha = -2.88
pd = exp(alpha+x*gamma[1]+y*log(2)+z*log(2))/
        (1+exp(alpha+x*gamma[1]+y*log(2)+z*log(2)))
d = rbinom(n,size=1,prob=pd)

# form the population dataset
dat = as.data.frame(cbind(d, y, z, x))

# generate sample dataset with 200 cases and 200 controls
dat_cases = dat[which(dat$d==1),]
dat_controls= dat[which(dat$d==0),]
dat_cases_sample = dat_cases[sample(sum(dat$d==1),
                             200,replace=FALSE),]
dat_controls_sample = dat_controls[sample(sum(dat$d==0),
                                   200,replace=FALSE),]

dat_logistic = rbind(dat_cases_sample,dat_controls_sample)
colnames(dat_logistic) = c("D", "y", "z","x")
D = dat_logistic$D # Disease status
pD = sum(dat$d==1)/500000 # Population disease prevalence

## WEE logsitic regression
WEE.logistic(y ~ x + z, D,
             data = dat_logistic, pD)

WEE.logistic(y ~ x + z, D,
             data = dat_logistic, pD, boot = 500)
```

WEE.quantile — *WEE quantile regression*

Description

Returns an object of class "WEE.quantile" that is generated by quantile regression with WEE approach for continuous secondary traits in genetic case-control studies.

Usage

```
WEE.quantile(formula, D, data, pd_pop, tau, iter = 5, boot = 0, ...)
```

Arguments

`formula`	The secondary trait given SNPs and covariates. e.g. y~x+z
`D`	Primary disease (case-control status), must be specified.
`data`	Dataset with real observation.
`pd_pop`	The population disease prevelance of primary disease.
`tau`	The quantile level to be estimated. Multiple taus can be chosen.
`iter`	Number of generating pseudo observations. (iter=10 by default)
`boot`	Number of bootstrape samples. (boot=0 by default)
`...`	Optional arguments to be passed through to rq.

Details

The quantile regression package "quantreg" is required before calling this function

Value

`Coefficients`	Point estimates
`StdErr`	Bootstrap standard errors, returned if boot > 0
`Wald`	Wald test statistics, returned if boot > 0
`p.value`	p-values, returned if boot > 0
`Covariance`	Covariance matrix, returned if boot > 0

Warning

If boot = 0, point estimates are plotted. If boot > 0, boostrap standard errors, Wald test statistics, p-values, and covariance matrix are also returned. Optional arguments from rq can be passed to this function, but arguments 'subset' and 'weights' should be used with caution.

References

[1] Ying Wei, Xiaoyu Song, Mengling Liu, Iuliana Ionita-Laza and Joan Reibman (2016). Quantile Regression in the Secondary Analysis of Case Control Data. *Journal of the American Statistical Association*, 111:513, 344-354; DOI: 10.1080/01621459.2015.1008101

[2] Xiaoyu Song, Iuliana Ionita-Laza, Mengling Liu, Joan Reibman, Ying Wei (2016). A General and Robust Framework for Secondary Traits Analysis. *Genetics*, vol. 202 no. 4 1329-1343; DOI: 10.1534/genetics.115.181073

Examples

```
## Generate simulated data
# set population size as 500000
n = 500000

# set parameters
beta = c(0.12, 0.1) # P(Y|X,Z)
gamma = c(-4, log(1.5), log(1.5), log(2) ) #P(D|X,Y,Z)

# generate the genetic variant X
x = rbinom(n,size=2,prob=0.3)

# generate the continuous covariate Z
z = rnorm(n)

# generate the continuous secondary trait Y
y= 1 + beta[1]*x + beta[2]*z + (1+0.02*x)*rnorm(n)

# generate disease status D
p = exp(gamma[1]+x*gamma[2]+z*gamma[3]+y*gamma[4])/
(1+exp(gamma[1]+x*gamma[2]+z*gamma[3]+y*gamma[4]))
d = rbinom(n,size=1,prob=p)

# form population data dataset
dat = as.data.frame(cbind(x,y,z,d))
colnames(dat) = c("x","y","z","d")

# Generate sample dataset with 200 cases and 200 controls
dat_cases = dat[which(dat$d==1),]
dat_controls= dat[which(dat$d==0),]
dat_cases_sample = dat_cases[sample(sum(dat$d==1),
                             200, replace=FALSE),]
dat_controls_sample = dat_controls[sample(sum(dat$d==0),
                                   200, replace=FALSE),]

dat_quantile = as.data.frame(rbind(dat_cases_sample,
                                   dat_controls_sample))
colnames(dat_quantile) = c("x","y","z","D")
D = dat_quantile$D # Disease status
pd = sum(d==1)/n # population disease prevalence

# WEE quantile regressions:
```

```
WEE.quantile(y ~ x, D, tau = 0.5,
             data = dat_quantile, pd_pop = pd)

WEE.quantile(y ~ x + z, D, tau = 1:9/10,
             data = dat_quantile, pd_pop = pd, boot = 500)
```

Index