Package ‘BayesGWQS’

June 12, 2020

Type Package

Title Bayesian Grouped Weighted Quantile Sum Regression

Version 0.0.1

Author David Wheeler, Matthew Carli

Maintainer Matthew Carli <carlimm@mymail.vcu.edu>


License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Depends R (>= 3.6.0)

Imports coda, stats

Suggests R2OpenBUGS, testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2020-06-12 08:30:07 UTC

R topics documented:

bgwqs.fit .................................................. 2
make.X .................................................... 3
make.x.s .................................................. 4
simdata .................................................... 4

Index 6
Bayesian Grouped WQS Regression

Description

This function fits a Bayesian grouped weighted quantile sum (BGWQS) regression model.

Usage

```r
bgwqs.fit(
  y,  # A vector containing outcomes.
  x,  # A matrix of component data.
  z = NULL,  # A vector or matrix of covariates.
  x.s,  # A vector of the number of components in each index.
  n.quantiles = 4,  # The number of quantiles to apply to the component data.
  working.dir,  # A file path to the directory.
  n.iter = 10000,  # The number of total iterations per chain, including burn in.
  n.burnin = 5000,  # The number of iterations to discard at the beginning.
  n.thin = 1,  # The thinning rate, which must be a positive integer.
  debug = FALSE  # False by default, when true OpenBUGS remains open for further investigation.
)
```

Arguments

- `y`: A vector containing outcomes.
- `x`: A matrix of component data.
- `z`: A vector or matrix of covariates.
- `x.s`: A vector of the number of components in each index.
- `n.quantiles`: The number of quantiles to apply to the component data.
- `working.dir`: A file path to the directory.
- `n.iter`: The number of total iterations per chain, including burn in.
- `n.burnin`: The number of iterations to discard at the beginning.
- `n.thin`: The thinning rate, which must be a positive integer.
- `debug`: False by default, when true OpenBUGS remains open for further investigation.

Value

A list of 3 which includes the BUGS output, coda output, and convergence test results.
**Examples**

```r
data("simdata")
group_list <- list(c("pcb_118", "pcb_138", "pcb_153", "pcb_180", "pcb_192"),
c("as", "cu", "pb", "sn"),
c("carbaryl", "propoxur", "methoxychlor", "diazinon", "chlorpyrifos"))
x.s <- make.x.s(simdata, 3, group_list)
X <- make.X(simdata, 3, group_list)
Y <- simdata$Y
work_dir <- tempdir()
results <- bgwqs.fit(y = Y, x = X, x.s = x.s, n.quantiles=4, working.dir = work_dir,
n.iter = 10000, n.burnin = 5000, n.thin = 1)
```

---

**make.X**

*Forms matrix of components*

**Description**

This function returns a matrix of component variables, X. The user can specify the desired chemicals and order by creating a list of string vectors, each vector containing the variable names of all desired elements of that group.

**Usage**

```r
make.X(df, num.groups, groups)
```

**Arguments**

- `df` A dataframe containing named component variables
- `num.groups` An integer representing the number of component groups desired
- `groups` A list, each item in the list being a string vector of variable names for one component group

**Value**

A matrix of component variables

**Examples**

```r
data("simdata")
group_list <- list(c("pcb_118", "pcb_138", "pcb_153", "pcb_180", "pcb_192"),
c("as", "cu", "pb", "sn"),
c("carbaryl", "propoxur", "methoxychlor", "diazinon", "chlorpyrifos"))
X <- make.X(simdata, 3, group_list)
X
```
make.x.s

*Forms component group ID vector of X*

**Description**

This function returns a vector which lets WQS.fit know the size and order of groups in X.

**Usage**

`make.x.s(df, num.groups, groups)`

**Arguments**

- `df`: A dataframe containing named component variables.
- `num.groups`: An integer representing the number of component groups desired.
- `groups`: A list, each item in the list being a string vector of variable names for one component group.

**Value**

A vector of integers, each integer relating how many columns are in each group.

**Examples**

```r
data("simdata")
group_list <- list(c("pcb_118", "pcb_138", "pcb_153", "pcb_180", "pcb_192"),
c("as", "cu", "pb", "sn"),
c("carbaryl", "propoxur", "methoxychlor", "diazinon", "chlorpyrifos"))
x.s <- make.x.s(simdata, 3, group_list)
x.s
```

**simdata**

*Simulated data of chemical concentrations and one binary outcome variable*

**Description**

Data were simulated to have 0.7 in-group correlation and 0.3 between-group correlation. There are three groups, with the third being significantly correlated to the outcome variable.

**Usage**

`simdata`
Format
A data frame with 1000 rows and 15 variables:

- `pcb_118` a numeric vector; part of group 1
- `pcb_138` a numeric vector; part of group 1
- `pcb_153` a numeric vector; part of group 1
- `pcb_180` a numeric vector; part of group 1
- `pcb_192` a numeric vector; part of group 1
- `as` a numeric vector; part of group 2
- `cu` a numeric vector; part of group 2
- `pb` a numeric vector; part of group 2
- `sn` a numeric vector; part of group 2
- `carbaryl` a numeric vector; part of group 3
- `propoxur` a numeric vector; part of group 3
- `methoxychlor` a numeric vector; part of group 3
- `diazinon` a numeric vector; part of group 3
- `chlorpyrifos` a numeric vector; part of group 3
- `Y` a numeric vector; the outcome variable
Index

*Topic datasets
  simdata, 4

bgwqs.fit, 2
make.X, 3
make.x.s, 4
simdata, 4