Package ‘RegParallel’

June 27, 2024

Type Package
Title Standard regression functions in R enabled for parallel processing over large data-frames
Version 1.22.0
Maintainer Kevin Blighe <kevin@clinicalbioinformatics.co.uk>
Description In many analyses, a large amount of variables have to be tested independently against the trait/endpoint of interest, and also adjusted for covariates and confounding factors at the same time. The major bottleneck in these is the amount of time that it takes to complete these analyses. With RegParallel, a large number of tests can be performed simultaneously. On a 12-core system, 144 variables can be tested simultaneously, with 1000s of variables processed in a matter of seconds via 'nested' parallel processing. Works for logistic regression, linear regression, conditional logistic regression, Cox proportional hazards and survival models, and Bayesian logistic regression. Also caters for generalised linear models that utilise survey weights created by the 'survey' CRAN package and that utilise 'survey::svyglm'.
License GPL-3
Depends doParallel, foreach, parallel, iterators, data.table, stringr, survival, arm, stats, utils, methods
Suggests RUnit, BiocGenerics, knitr, DESeq2, airway, magrittr, Biobase, GEOquery, biomaRt, survminer, survey, markdown
URL https://github.com/kevinblighe/RegParallel
biocViews DiseaseModel
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/RegParallel
git_branch RELEASE_3_19
git_last_commit b73243e
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-06-27
Author Kevin Blighe [aut, cre], Sarega Gurudas [ctb], Jessica Lasky-Su [aut]
Contents

RegParallel-package .................................................. 2
bayesglmParallel .................................................... 2
clogitParallel ....................................................... 5
coxphParallel ....................................................... 7
glmParallel ........................................................ 9
lmParallel .......................................................... 12
RegParallel .......................................................... 14
svyglmParallel ..................................................... 20

Index 23

RegParallel-package

RegParallel: Standard regression functions in R enabled for parallel processing over large data-frames.

Description

In many analyses, a large amount of variables have to be tested independently against the trait/endpoint of interest, and also adjusted for covariates and confounding factors at the same time. The major bottleneck in these is the amount of time that it takes to complete these analyses. With RegParallel, a large number of tests can be performed simultaneously. On a 12-core system, 144 variables can be tested simultaneously, with 1000s of variables processed in a matter of seconds via ‘nested’ parallel processing. Works for logistic regression, linear regression, conditional logistic regression, Cox proportional hazards and survival models, and Bayesian logistic regression. Also caters for generalised linear models that utilise survey weights created by the ‘survey’ CRAN package and that utilise ‘survey::svyglm’.

bayesglmParallel

Standard regression functions in R enabled for parallel processing over large data-frames - Bayesian logistic regression

Description

This is a non-user function that is managed by RegParallel, the primary function.

Usage

bayesglmParallel(
  data,
  formula.list,
  FUN,
  variables,
  terms,
  startIndex,
bayesglmParallel

blocksize, blocks, APPLYFUN, conflevel, excludeTerms, excludeIntercept)

Arguments

data A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.
formulalist A list containing formulae that can be coerced to formula class via as.formula(). REQUIRED.
FUN Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family = binomial, data = data). REQUIRED.
variables Vector of variable names in data to be tested independently. Each variable will have its own formula in formula.list. REQUIRED.
terms Vector of terms used in the formulae in formula.list, excluding the primary variable of interest. REQUIRED.
startIndex Starting column index in data object from which processing can commence. REQUIRED.
blocksize Number of variables to test in each foreach loop. REQUIRED.
blocks Total number of blocks required to complete analysis. REQUIRED.
APPLYFUN The apply function to be used within each block during processing. Will be one of: 'mclapply(...)', system=linux/mac and nestedParallel=TRUE; 'parLapply(cl, ...)', system=windows and nestedParallel=TRUE; 'lapply(...)', nestedParallel=FALSE. REQUIRED.
conflevel Confidence level for calculating odds or hazard ratios. REQUIRED.
excludeTerms Remove these terms from the final output. These will simply be grepped out. REQUIRED.
excludeIntercept Remove intercept terms from the final output. REQUIRED.

Details

This is a non-user function that is managed by RegParallel, the primary function.

Value

A data.table object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>
Examples

```r
options(scipen=10)
options(digits=6)

col <- 20000
row <- 20
mat <- matrix(
  rexp(col*row, rate = .1),
  ncol = col)
colnames(mat) <- paste0('gene', 1:ncol(mat))
rownames(mat) <- paste0('sample', 1:nrow(mat))

modelling <- data.frame(
  cell = rep(c('B', 'T'), nrow(mat) / 2),
group = c(rep(c('treatment'), nrow(mat) / 2), rep(c('control'), nrow(mat) / 2)),
dosage = t(data.frame(matrix(rexp(row, rate = 1), ncol = row))),
mat,
row.names = rownames(mat))

data <- modelling[,1:5000]
variables <- colnames(data)[4:ncol(data)]
res6 <- RegParallel(
  data = data,
  formula = 'as.numeric(factor(cell)) ~ [*]:dosage',
  FUN = function(formula, data)
    bayesglm(formula = formula,
      data = data,
      prior.mean = 2),
  FUNtype = 'bayesglm',
  variables = variables,
  blocksize = 500,
  cores = 2,
  nestedParallel = FALSE,
  p.adjust = "none",
  conflevel = 99,
  excludeTerms = NULL,
  excludeIntercept = FALSE)

# spot checks
m <- bayesglm(formula = as.numeric(factor(cell)) ~ gene1645:dosage, data = data, prior.mean = 2)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res6[which(res6$Variable == 'gene1645'),]

m <- bayesglm(formula = as.numeric(factor(cell)) ~ gene3664:dosage, data = data, prior.mean = 2)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res6[which(res6$Variable == 'gene3664'),]
```
clogitParallel

Standard regression functions in R enabled for parallel processing over large data-frames - conditional logistic regression.

Description

This is a non-user function that is managed by RegParallel, the primary function.

Usage

clogitParallel(
  data,
  formula.list,
  FUN,
  variables,
  terms,
  startIndex,
  blocksize,
  blocks,
  APPLYFUN,
  conflevel,
  excludeTerms)

Arguments

data A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.
formula.list A list containing formulae that can be coerced to formula class via as.formula(). REQUIRED.
FUN Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family = binomial, data = data). REQUIRED.
variables Vector of variable names in data to be tested independently. Each variable will have its own formula in formula.list. REQUIRED.
terms Vector of terms used in the formulae in formula.list, excluding the primary variable of interest. REQUIRED.
startIndex Starting column index in data object from which processing can commence. REQUIRED.
blocksize Number of variables to test in each foreach loop. REQUIRED.
blocks Total number of blocks required to complete analysis. REQUIRED.
APPLYFUN The apply function to be used within each block during processing. Will be one of: 'mclapply(...)', system=linux/mac and nestedParallel=TRUE; 'parLapply(cl, ...)', system=windows and nestedParallel=TRUE; 'lapply(...)', nestedParallel=FALSE. REQUIRED.
conflevel Confidence level for calculating odds or hazard ratios. REQUIRED.
excludeTerms Remove these terms from the final output. These will simply be grepped out. REQUIRED.
Details

This is a non-user function that is managed by RegParallel, the primary function.

Value

A data.table object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```r
options(scipen=10)
options(digits=6)

col <- 20000
row <- 20
mat <- matrix(
  rexp(col*row, rate = .1),
  ncol = col)
colnames(mat) <- paste0('gene', 1:ncol(mat))ownames(mat) <- paste0('sample', 1:nrow(mat))

modelling <- data.frame(
  cell = rep(c('B', 'T'), nrow(mat) / 2),
  group = c(rep(c('treatment'), nrow(mat) / 2), rep(c('control'), nrow(mat) / 2)),
  dosage = t(data.frame(matrix(rexp(row, rate = 1), ncol = row))),
  mat,
  row.names = rownames(mat))
data <- modelling[,1:500]
variables <- colnames(data)[4:ncol(data)]
res5 <- RegParallel(
  data = data,
  formula = 'as.integer(factor(group)) - [*] * strata(cell) + dosage',
  FUN = function(formula, data)
    clogit(formula = formula,
       data = data,
       ties = 'breslow',
       singular.ok = TRUE),
  FUNtype = 'clogit',
  variables = variables,
  blocksize = 200,
  cores = 2,
  nestedParallel = FALSE,
  p.adjust = "none",
  conflevel = 50,
  excludeTerms = 'non-existent term',
  excludeIntercept = FALSE
)
```
coxphParallel

Standard regression functions in R enabled for parallel processing over large data-frames - Cox proportional hazards regression.

Description

This is a non-user function that is managed by RegParallel, the primary function.

Usage

```r
coxphParallel(
data, formula.list, FUN, variables, terms, startIndex, blocksize, blocks, APPLYFUN, conflevel, excludeTerms)
```

Arguments

- **data**: A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.
- **formula.list**: A list containing formulae that can be coerced to formula class via as.formula(). REQUIRED.
- **FUN**: Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family = binomial, data = data). REQUIRED.
- **variables**: Vector of variable names in data to be tested independently. Each variable will have its own formula in formula.list. REQUIRED.
- **terms**: Vector of terms used in the formulae in formula.list, excluding the primary variable of interest. REQUIRED.
startIndex   Starting column index in data object from which processing can commence. REQUIRED.
blocksize    Number of variables to test in each foreach loop. REQUIRED.
blocks       Total number of blocks required to complete analysis. REQUIRED.
APPLYFUN     The apply function to be used within each block during processing. Will be one of: 'mclapply(...)', system=linux/mac and nestedParallel=TRUE; 'parLapply(cl, ...)', system=windows and nestedParallel=TRUE; 'lapply(...)', nestedParallel=FALSE. REQUIRED.
conflevel    Confidence level for calculating odds or hazard ratios. REQUIRED.
excludeTerms Remove these terms from the final output. These will simply be GREpped out. REQUIRED.

Details
This is a non-user function that is managed by RegParallel, the primary function.

Value
A data.table object.

Author(s)
Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples
options(scipen=10)
options(digits=6)

col <- 20000
row <- 20
mat <- matrix(
  rexp(col*row, rate = .1),
  ncol = col)

colnames(mat) <- paste0('gene', 1:ncol(mat))
rownames(mat) <- paste0('sample', 1:nrow(mat))

modelling <- data.frame(
  cell = rep(c('B', 'T'), nrow(mat) / 2),
  group = c(rep(c('treatment'), nrow(mat) / 2), rep(c('control'), nrow(mat) / 2)),
  dosage = t(data.frame(matrix(rexp(row, rate = 1), ncol = row))),
  mat,
  row.names = rownames(mat))

require(survival)
data <- modelling[,1:800]
variables <- colnames(data)[4:ncol(data)]
data$time <- c(100,200,400,300,200,250,600,1000,886,450,
             c(100,200,400,300,200,250,600,1000,886,450)*1.5)
data$alive <- c(0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,1,1,1,1)
glmParallel

Standard regression functions in R enabled for parallel processing over large data-frames - generalised linear model

Description

This is a non-user function that is managed by RegParallel, the primary function.

Usage

glmParallel(
  data,
  formula.list,
  FUN,
  variables,
  terms,
  startIndex,
  blocksize,
  blocks,
gilmParallel

APPLYFUN,
conlevel,
excludeTerms,
excludeIntercept)

Arguments

data A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.

formula.list A list containing formulae that can be coerced to formula class via as.formula(). REQUIRED.

FUN Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family = binomial, data = data). REQUIRED.

variables Vector of variable names in data to be tested independently. Each variable will have its own formula in formula.list. REQUIRED.

terms Vector of terms used in the formulae in formula.list, excluding the primary variable of interest. REQUIRED.

startIndex Starting column index in data object from which processing can commence. REQUIRED.

blocksize Number of variables to test in each foreach loop. REQUIRED.

blocks Total number of blocks required to complete analysis. REQUIRED.

APPLYFUN The apply function to be used within each block during processing. Will be one of: 'mclapply(...)', system=linux/mac and nestedParallel=TRUE; 'parLapply(cl, ...)', system=windows and nestedParallel=TRUE; 'lapply(...)', nestedParallel=FALSE. REQUIRED.

conlevel Confidence level for calculating odds or hazard ratios. REQUIRED.

excludeTerms Remove these terms from the final output. These will simply be grepped out. REQUIRED.

excludeIntercept Remove intercept terms from the final output. REQUIRED.

Details

This is a non-user function that is managed by RegParallel, the primary function.

Value

A data.table object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>
**Examples**

```r
options(scipen=10)
options(digits=6)

col <- 20000
row <- 20
mat <- matrix(
  rexp(col*row, rate = .1),
  ncol = col)
colnames(mat) <- paste0('gene', 1:ncol(mat))
rownames(mat) <- paste0('sample', 1:nrow(mat))

modelling <- data.frame(
  cell = rep(c('B', 'T'), nrow(mat) / 2),
  group = c(rep(c('treatment'), nrow(mat) / 2), rep(c('control'), nrow(mat) / 2)),
  dosage = t(data.frame(matrix(rexp(row, rate = 1), ncol = row))),
  mat,
  row.names = rownames(mat))
data <- modelling[,1:2000]
variables <- colnames(data)[4:ncol(data)]
res1 <- RegParallel(
  data = data,
  formula = 'factor(group) ~ [+] + (cell:dosage) ^ 2',
  FUN = function(formula, data)
    glm(formula = formula,
        data = data,
        family = binomial(link = 'logit'),
        method = 'glm.fit'),
  FUNtype = 'glm',
  variables = variables,
  blocksize = 700,
  cores = 2,
  nestedParallel = TRUE,
  p.adjust = "none",
  conflevel = 99,
  excludeTerms = NULL,
  excludeIntercept = TRUE)

# spot checks
m <- glm(factor(group) ~ gene265 + (cell:dosage) ^ 2, data=data, family=binomial)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res1[which(res1$Variable == 'gene265'),]

m <- glm(factor(group) ~ gene1688 + (cell:dosage) ^ 2, data=data, family=binomial)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res1[which(res1$Variable == 'gene1688'),]
```
```r
###
data <- modelling[,1:500]
variables <- colnames(data)[4:ncol(data)]
res2 <- RegParallel(
data = data,
formula = '[*] ~ cell:dosage',
FUN = function(formula, data)
  glm(formula = formula,
data = data,
  family = gaussian,
  method = 'glm.fit'),
FUNtype = 'glm',
variables = variables,
blocksize = 496,
cores = 2,
nestedParallel = TRUE,
p.adjust = "none",
conflevel = 90,
excludeTerms = NULL,
excludeIntercept = FALSE
)

# spot checks
m <- glm(gene29 ~ cell:dosage, data=data, family=gaussian)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.90)))
res2[which(res2$Variable == 'gene29'),]
```

### lmParallel

**lmParallel**

*Standard regression functions in R enabled for parallel processing over large data-frames - linear model.*

#### Description

This is a non-user function that is managed by RegParallel, the primary function.

#### Usage

```r
lmParallel(
data, formula.list, FUN, variables, terms, startIndex, blocksize, blocks,
```

Arguments

data A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.

formula.list A list containing formulae that can be coerced to formula class via as.formula(). REQUIRED.

FUN Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family=binomial, data = data). REQUIRED.

variables Vector of variable names in data to be tested independently. Each variable will have its own formula in formula.list. REQUIRED.

terms Vector of terms used in the formulae in formula.list, excluding the primary variable of interest. REQUIRED.

startIndex Starting column index in data object from which processing can commence. REQUIRED.

blocksize Number of variables to test in each foreach loop. REQUIRED.

blocks Total number of blocks required to complete analysis. REQUIRED.

APPLYFUN The apply function to be used within each block during processing. Will be one of: 'mclapply(...)’, system=linux/mac and nestedParallel=TRUE; 'parLapply(cl, ...)’, system=windows and nestedParallel=TRUE; 'lapply(...)’, nestedParallel=FALSE. REQUIRED.

conflevel Confidence level for calculating odds or hazard ratios. REQUIRED.

excludeTerms Remove these terms from the final output. These will simply be grepped out. REQUIRED.

excludeIntercept Remove intercept terms from the final output. REQUIRED.

Details

This is a non-user function that is managed by RegParallel, the primary function.

Value

A data.table object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>
Examples

options(scipen=10)
options(digits=6)

col <- 20000
row <- 20
mat <- matrix(
  rexp(col*row, rate = .1),
  ncol = col)
colnames(mat) <- paste0('gene', 1:ncol(mat))
rownames(mat) <- paste0('sample', 1:nrow(mat))

modelling <- data.frame(
  cell = rep(c('B', 'T'), nrow(mat) / 2),
group = c(rep(c('treatment'), nrow(mat) / 2), rep(c('control'), nrow(mat) / 2)),
dosage = t(data.frame(matrix(rexp(row, rate = 1), ncol = row))),
  mat,
  row.names = rownames(mat))
data <- modelling[,1:500]
variables <- colnames(data)[4:ncol(data)]
res3 <- RegParallel(
  data = data,
  formula = 'as.numeric([x]) ~ dosage ^ 3',
  FUN = function(formula, data)
    lm(formula = formula,
      data = data),
  FUNtype = 'lm',
  variables = variables,
  blocksize = 200,
  cores = 2,
  nestedParallel = FALSE,
  p.adjust = "none",
  conflevel = 99.999,
  excludeTerms = NULL,
  excludeIntercept = FALSE
)

# spot checks
m <- lm(as.numeric(gene454) ~ dosage ^ 3, data=data)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.999999)))
res3[which(res3$Variable == 'gene454'),]

RegParallel

Standard regression functions in R enabled for parallel processing over large data-frames
Description

In many analyses, a large amount of variables have to be tested independently against the trait/endpoint of interest, and also adjusted for covariates and confounding factors at the same time. The major bottleneck in these is the amount of time that it takes to complete these analyses. With RegParallel, a large number of tests can be performed simultaneously. On a 12-core system, 144 variables can be tested simultaneously, with 1000s of variables processed in a matter of seconds via 'nested' parallel processing. Works for logistic regression, linear regression, conditional logistic regression, Cox proportional hazards and survival models, and Bayesian logistic regression. Also caters for generalised linear models that utilise survey weights created by the 'survey' CRAN package and that utilise 'survey::svyglm'.

Usage

RegParallel(
  data,
  design = NULL,
  formula,
  FUN,
  FUNtype,
  variables,
  blocksize = 500,
  cores = 3,
  nestedParallel = FALSE,
  p.adjust = 'none',
  conflevel = 95,
  excludeTerms = NULL,
  excludeIntercept = TRUE)

Arguments

data A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.
design A survey design, created by survey::svydesign. DEFAULT = NULL. OPTIONAL.
formula A valid formula. Excluding the '^[*]' term, which is reserved for RegParallel and indicates the position in the formula for the variable of interest, must pass as.formula() check. REQUIRED.
FUN Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family=binomial, data = data). REQUIRED.
FUNtype Regression function type. Must be one of 'glm', 'lm', 'coxph', 'clogit', 'bayesglm', or 'glm.nb'. REQUIRED.
variables Vector of variable names in data to be tested independently. Each variable will take the place of '^[*]' in the supplied formula. REQUIRED.
blocksize Number of variables to test in each foreach loop. DEFAULT = 500. OPTIONAL.
cores CPU cores / threads. DEFAULT = 3. OPTIONAL.
nestedParallel  In RegParallel, parallelisation initially occurs at the block level, i.e., multiple blocks of models are processed in parallel. If nestedParallel is enabled, a second level of parallelisation occurs within each block in addition. Warning! - this doubles the usage of cores. DEFAULT = FALSE. OPTIONAL.

p.adjust  Method for adjusting p-values for false discovery rate. Must be one of 'holm', 'hochberg', 'hommel', 'bonferroni', 'BH', 'BY', 'fdr', 'none'. See ?p.adjust for further details. DEFAULT = 'none'. OPTIONAL.

conflevel  Confidence level for calculating odds or hazard ratios. DEFAULT = 95. OPTIONAL.

excludeTerms  Remove these terms from the final output. These will simply be grepped out. DEFAULT = NULL. OPTIONAL.

excludeIntercept  Remove intercept terms from the final output. DEFAULT = TRUE. OPTIONAL.

Details
In many analyses, a large amount of variables have to be tested independently against the trait/endpoint of interest, and also adjusted for covariates and confounding factors at the same time. The major bottleneck in these is the amount of time that it takes to complete these analyses. With RegParallel, a large number of tests can be performed simultaneously. On a 12-core system, 144 variables can be tested simultaneously, with 1000s of variables processed in a matter of seconds via ‘nested’ parallel processing. Works for logistic regression, linear regression, conditional logistic regression, Cox proportional hazards and survival models, and Bayesian logistic regression.

Value
A data.table object.

Author(s)
Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples
options(scipen=10)
options(digits=6)

col <- 20000
row <- 20
mat <- matrix(rexp(col*row, rate = .1),
ncol = col)
colnames(mat) <- paste0('gene', 1:ncol(mat))
rownames(mat) <- paste0('sample', 1:nrow(mat))

modelling <- data.frame(
cell = rep(c('B', 'T'), nrow(mat) / 2),
group = c(rep(c('treatment'), nrow(mat) / 2), rep(c('control'), nrow(mat) / 2)),
dosage = t(data.frame(matrix(rexp(row, rate = 1), ncol = row))),
mat,
data <- modelling[,1:2000]
variables <- colnames(data)[4:ncol(data)]
res1 <- RegParallel(
data = data,
formula = 'factor(group) ~ [*] + (cell:dosage) ^ 2',
FUN = function(formula, data)
  glm(formula = formula,
data = data,
family = binomial(link = 'logit'),
method = 'glm.fit'),
FUNtype = 'glm',
variables = variables,
blocksize = 700,
tags = 2,
nestedParallel = TRUE,
#p.adjust = "bonferroni",
conflevel = 99,
excludeTerms = NULL,
excludeIntercept = TRUE)

# spot checks
m <- glm(factor(group) ~ gene265 + (cell:dosage) ^ 2, data=data, family=binomial)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res1[which(res1$Variable == 'gene265'),]

m <- glm(factor(group) ~ gene1688 + (cell:dosage) ^ 2, data=data, family=binomial)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res1[which(res1$Variable == 'gene1688'),]

###
data <- modelling[,1:500]
variables <- colnames(data)[4:ncol(data)]
res2 <- RegParallel(
data = data,
formula = '[*] ~ cell:dosage',
FUN = function(formula, data)
  glm(formula = formula,
data = data,
family = gaussian,
method = 'glm.fit'),
FUNtype = 'glm',
variables = variables,
blocksize = 496,
tags = 2,
nestedParallel = TRUE,
p.adjust = "none",
conflevel = 90,
excludeTerms = NULL,
excludeIntercept = FALSE
)

# spot checks
m <- glm(gene29 ~ cell:dosage, data=data, family=gaussian)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.90)))
res2[which(res2$Variable == 'gene29'),]

###
data <- modelling[,1:500]
variables <- colnames(data)[4:ncol(data)]
res3 <- RegParallel(
data = data,
formula = 'as.numeric([*]) ~ dosage ^ 3',
FUN = function(formula, data)
  lm(formula = formula,
data = data),
FUNtype = 'lm',
variables = variables,
blocksize = 200,
cores = 2,
nestedParallel = FALSE,
p.adjust = "holm",
conflevel = 99.999,
excludeTerms = NULL,
excludeIntercept = FALSE
)

# spot checks
m <- lm(as.numeric(gene454) ~ dosage ^ 3, data=data)
summary(m)$coefficients
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99999)))
res3[which(res3$Variable == 'gene454'),]

###
require(survival)
data <- modelling[,1:800]
variables <- colnames(data)[4:ncol(data)]
data$time <- c(100,200,400,300,200,250,600,1000,886,450,
c(100,200,400,300,200,250,600,1000,886,450)*1.5)
data$alive <- c(0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,1,1,1,1)
res4 <- RegParallel(
data = data,
formula = 'Surv(time, as.integer(alive)) ~ group * [+] + cell', 
FUN = function(formula, data) 
coxph(formula = formula, 
data = data, 
ties = 'breslow', 
singular.ok = TRUE), 
FUNtype = 'coxph', 
variables = variables, 
blocksize = 399, 
cores = 2, 
nestedParallel = FALSE, 
p.adjust = "hommel", 
conflevel = 97.5, 
excludeTerms = c('group', 'cell'), 
excludeIntercept = FALSE 
) 

# spot checks 
m <- coxph(formula = Surv(time, as.integer(factor(alive))) ~ group * gene12 + cell, data = data, ties = 'breslow', 
summary(m) 
exp(cbind("Hazards ratio" = coef(m), confint.default(m, level = 0.975))) 
res4[which(res4$Variable == 'gene12'),] 

m <- coxph(formula = Surv(time, as.integer(factor(alive))) ~ group * gene267 + cell, data = data, ties = 'breslow', 
summary(m) 
exp(cbind("Hazards ratio" = coef(m), confint.default(m, level = 0.975))) 
res4[which(res4$Variable == 'gene267'),] 

### 

data <- modelling[,1:500] 
variables <- colnames(data)[4:ncol(data)] 
res5 <- RegParallel( 
data = data, 
formula = 'as.integer(factor(group)) ~ [+] * strata(cell) + dosage', 
FUN = function(formula, data) 
clogit(formula = formula, 
data = data, 
ties = 'breslow', 
singular.ok = TRUE), 
FUNtype = 'clogit', 
variables = variables, 
blocksize = 200, 
cores = 2, 
nestedParallel = FALSE, 
p.adjust = "fdr", 
conflevel = 50, 
excludeTerms = 'non-existent term', 
excludeIntercept = FALSE 
)
# spot checks
m <- clogit(formula = as.integer(factor(group)) ~ gene145 * strata(cell) + dosage, data = data, ties = "breslow", singular.ok = TRUE)
summary(m)
exp(cbind("Hazards ratio" = coef(m), confint.default(m, level = 0.5)))
res5[which(res5$Variable == 'gene145'),]

m <- clogit(formula = as.integer(factor(group)) ~ gene34 * strata(cell) + dosage, data = data, ties = "breslow", singular.ok = TRUE)
summary(m)
exp(cbind("Hazards ratio" = coef(m), confint.default(m, level = 0.5)))
res5[which(res5$Variable == 'gene34'),]

###
data <- modelling[,1:5000]
variables <- colnames(data)[4:ncol(data)]
res6 <- RegParallel(
data = data,
formula = 'as.numeric(factor(cell)) ~ [~]:dosage',
FUN = function(formula, data)
  bayesglm(formula = formula, data = data, prior.mean = 2),
FUNtype = 'bayesglm',
variables = variables,
blocksize = 500,
cores = 2,
nestedParallel = FALSE,
p.adjust = "fdr",
conflevel = 99,
excludeTerms = NULL,
excludeIntercept = FALSE)

# spot checks
m <- bayesglm(formula = as.numeric(factor(cell)) ~ gene1645:dosage, data = data, prior.mean = 2)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res6[which(res6$Variable == 'gene1645'),]

m <- bayesglm(formula = as.numeric(factor(cell)) ~ gene3664:dosage, data = data, prior.mean = 2)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.99)))
res6[which(res6$Variable == 'gene3664'),]
svyglmParallel

Description
This is a non-user function that is managed by RegParallel, the primary function.

Usage
svyglmParallel(
  data,  # A data-frame that contains all model terms to be tested. Variables that have all zeros will, automatically, be removed. REQUIRED.
  design,  # A survey design, created by survey::svydesign. REQUIRED.
  formula.list,  # A list containing formulae that can be coerced to formula class via as.formula(). REQUIRED.
  FUN,  # Regression function. Must be of form, for example: function(formula, data) glm(formula = formula, family = binomial, data = data). REQUIRED.
  variables,  # Vector of variable names in data to be tested independently. Each variable will have its own formula in formula.list. REQUIRED.
  terms,  # Vector of terms used in the formulae in formula.list, excluding the primary variable of interest. REQUIRED.
  startIndex,  # Starting column index in data object from which processing can commence. REQUIRED.
  blocksize,  # Number of variables to test in each foreach loop. REQUIRED.
  blocks,  # Total number of blocks required to complete analysis. REQUIRED.
  APPLYFUN,  # The apply function to be used within each block during processing. Will be one of: `mclapply(...)`, system=linux/mac and nestedParallel=TRUE; `parLapply(cl, ...)`, system=windows and nestedParallel=TRUE; `lapply(...)` nestedParallel=FALSE. REQUIRED.
  conflevel,  # Confidence level for calculating odds or hazard ratios. REQUIRED.
  excludeTerms,  # Remove these terms from the final output. These will simply be grepped out. REQUIRED.
  excludeIntercept  # Remove intercept terms from the final output. REQUIRED.
)
Details

This is a non-user function that is managed by RegParallel, the primary function.

Value

A data.table object.

Author(s)

Kevin Blighe <kevin@clinicalbioinformatics.co.uk>

Examples

```r
require(survey)
data(nhanes)
design <- svydesign(id = ~ SDMVPSU,
                     strata = ~ SDMVSTRA,
                     weights = ~ WTMEC2YR,
                     nest = TRUE,
                     data = nhanes)
```
Index

bayesglmParallel, 2

clogitParallel, 5
coxphParallel, 7

data.table, 3, 6, 8, 10, 13, 16, 22

glmParallel, 9

lmParallel, 12

RegParallel, 14
RegParallel-package, 2

svyglmParallel, 20