

Package ‘RegParallel’

April 11, 2019

Type Package

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

Version 1.0.0

Author Kevin Blighe

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, Bayesian logistic regression, and negative binomial regression.

License GPL-3

Depends R (>= 3.5.0), doParallel, foreach, parallel, iterators, data.table, stringr, survival, arm, MASS, stats, utils, methods

Imports

Suggests RUnit, BiocGenerics, knitr, DESeq2, airway, magrittr, Biobase, GEOquery, biomaRt, survminer

URL <https://github.com/kevinblighe/RegParallel>

biocViews DiseaseModel

VignetteBuilder knitr

git_url <https://git.bioconductor.org/packages/RegParallel>

git_branch RELEASE_3_8

git_last_commit 5178d6f

git_last_commit_date 2018-10-30

Date/Publication 2019-04-11

R topics documented:

RegParallel-package	2
bayesglmParallel	2

clogitParallel	4
coxphParallel	6
glm.nbParallel	9
glmParallel	11
lmParallel	13
RegParallel	15

Index	22
--------------	-----------

RegParallel-package	<i>RegParallel: Standard regression functions in R enabled for parallel processing over large data-frames.</i>
---------------------	--

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, Bayesian logistic regression, and negative binomial regression.

bayesglmParallel	<i>Standard regression functions in R enabled for parallel processing over large data-frames - Bayesian logistic regression</i>
------------------	---

Description

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

Usage

```
bayesglmParallel(
  data,
  formula.list,
  FUN,
  variables,
  terms,
  startIndex,
  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.
formula.list	A list containing formulae that can be coerced to formula class via <code>as.formula()</code> . REQUIRED.
FUN	Regression function. Must be of form, for example: <code>function(formula, data) glm(formula = formula, family = binomial, data = data)</code> . REQUIRED.
variables	Vector of variable names in data to be tested independently. Each variable will have its own formula in <code>formula.list</code> . REQUIRED.
terms	Vector of terms used in the formulae in <code>formula.list</code> , 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: <code>'mclapply(...)'</code> , <code>system=linux/mac</code> and <code>nestedParallel=TRUE</code> ; <code>'parLapply(cl, ...)'</code> , <code>system=windows</code> and <code>nestedParallel=TRUE</code> ; <code>'lapply(...)'</code> , <code>nestedParallel=FALSE</code> . REQUIRED.
confllevel	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[,1:5000]
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,
  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,
confllevel,
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 <code>as.formula()</code> . REQUIRED.
FUN	Regression function. Must be of form, for example: <code>function(formula, data) glm(formula = formula, family = binomial, data = data)</code> . REQUIRED.
variables	Vector of variable names in data to be tested independently. Each variable will have its own formula in <code>formula.list</code> . REQUIRED.
terms	Vector of terms used in the formulae in <code>formula.list</code> , 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: <code>'mclapply(...)'</code> , <code>system=linux/mac</code> and <code>nestedParallel=TRUE</code> ; <code>'parLapply(cl, ...)'</code> , <code>system=windows</code> and <code>nestedParallel=TRUE</code> ; <code>'lapply(...)'</code> , <code>nestedParallel=FALSE</code> . REQUIRED.
confllevel	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))

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,
  conflevel = 50,
  excludeTerms = 'non-existent term',
  excludeIntercept = FALSE
)

# spot checks
m <- clogit(formula = as.integer(group) ~ gene145 * strata(cell) + dosage, data = data, ties = 'breslow', singular.ok = TRUE)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.5)))
res5[which(res5$Variable == 'gene145'),]

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

```

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

```
coxphParallel(
```

```

data,
formula.list,
FUN,
variables,
terms,
startIndex,
blocksize,
blocks,
APPLYFUN,
confllevel,
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 <code>as.formula()</code> . REQUIRED.
FUN	Regression function. Must be of form, for example: <code>function(formula, data) glm(formula = formula, family = binomial, data = data)</code> . REQUIRED.
variables	Vector of variable names in data to be tested independently. Each variable will have its own formula in <code>formula.list</code> . REQUIRED.
terms	Vector of terms used in the formulae in <code>formula.list</code> , 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: <code>'mclapply(...)</code> , <code>system=linux/mac</code> and <code>nestedParallel=TRUE</code> ; <code>'parLapply(cl, ...)</code> , <code>system=windows</code> and <code>nestedParallel=TRUE</code> ; <code>'lapply(...)</code> , <code>nestedParallel=FALSE</code> . REQUIRED.
confllevel	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(0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,1,1)
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,
  conflevel = 97.5,
  excludeTerms = c('group', 'cell'),
  excludeIntercept = FALSE
)

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

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

```

glm.nbParallel	<i>Standard regression functions in R enabled for parallel processing over large data-frames - negative binomial generalised linear model.</i>
----------------	--

Description

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

Usage

```
glm.nbParallel(
  data,
  formula.list,
  FUN,
  variables,
  terms,
  startIndex,
  blocksize,
  blocks,
  APPLYFUN,
  confllevel,
  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.
confllevel	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[,1:5000]
data[,4:ncol(data)] <- asinh(scale(data[,4:ncol(data)]))
variables <- colnames(data)[4:ncol(data)]
res7 <- RegParallel(
  data = data,
  formula = 'as.integer(factor(cell)) ~ [*] + group * dosage',
  FUN = function(formula, data)
    glm.nb(formula = formula,
           data = data),
  FUNtype = 'glm.nb',
  variables = variables,
  blocksize = 500,
  cores = 2,
  nestedParallel = FALSE,
  conflevel = 95,
  excludeTerms = NULL,
  excludeIntercept = FALSE
)

# spot checks
m <- glm.nb(formula = as.integer(factor(cell)) ~ gene99 + group * dosage, data = data)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.95)))
res7[which(res7$Variable == 'gene99'),]
```

```
m <- glm.nb(formula = as.integer(factor(cell)) ~ gene2000 + group * dosage, data = data)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.95)))
res7[which(res7$Variable == 'gene2000'),]
```

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,
  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.
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[,1:2000]
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,
  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,
  cores = 2,
  nestedParallel = TRUE,
  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

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

```
lmParallel(
```

```

data,
formula.list,
FUN,
variables,
terms,
startIndex,
blocksize,
blocks,
APPLYFUN,
confllevel,
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 <code>as.formula()</code> . REQUIRED.
FUN	Regression function. Must be of form, for example: <code>function(formula, data) glm(formula = formula, family=binomial, data = data)</code> . REQUIRED.
variables	Vector of variable names in data to be tested independently. Each variable will have its own formula in <code>formula.list</code> . REQUIRED.
terms	Vector of terms used in the formulae in <code>formula.list</code> , 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: <code>'mclapply(...)'</code> , <code>system=linux/mac</code> and <code>nestedParallel=TRUE</code> ; <code>'parLapply(cl, ...)'</code> , <code>system=windows</code> and <code>nestedParallel=TRUE</code> ; <code>'lapply(...)'</code> , <code>nestedParallel=FALSE</code> . REQUIRED.
confllevel	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([*]) ~ dosage ^ 3',
  FUN = function(formula, data)
    lm(formula = formula,
      data = data),
  FUNtype = 'lm',
  variables = variables,
  blocksize = 200,
  cores = 2,
  nestedParallel = FALSE,
  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'),]

```

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, Bayesian logistic regression, and negative binomial regression.

Usage

```
RegParallel(
  data,
  formula,
  FUN,
  FUNtype,
  variables,
  blocksize = 500,
  cores = 3,
  nestedParallel = FALSE,
  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.
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, ie., 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.
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, Bayesian logistic regression, and negative binomial 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,
  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,
  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,
  cores = 2,
  nestedParallel = TRUE,
  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,
    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(0,0,0,0,0,0,0,0,1,1,1,0,0,1,1,1,1,1,1)
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,
  conflevel = 97.5,
  excludeTerms = c('group', 'cell'),
  excludeIntercept = FALSE
)

```

```

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

```

```

m <- coxph(formula = Surv(time, as.integer(alive)) ~ group * gene267 + cell, data = data, ties = 'breslow', s
summary(m)
exp(cbind("Odds 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,
  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')
summary(m)
exp(cbind("Odds 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')
summary(m)
exp(cbind("Odds 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,
  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'),]

###

data <- modelling[,1:5000]
data[,4:ncol(data)] <- asinh(scale(data[,4:ncol(data)]))
variables <- colnames(data)[4:ncol(data)]
res7 <- RegParallel(
  data = data,
  formula = 'as.numeric(factor(cell)) ~ [*] + group * dosage',
  FUN = function(formula, data)
    glm.nb(formula = formula,
            data = data),
  FUNtype = 'glm.nb',
  variables = variables,
  blocksize = 500,
  cores = 2,
  nestedParallel = FALSE,
  conflevel = 95,
  excludeTerms = NULL,
  excludeIntercept = FALSE
)

# spot checks
m <- glm.nb(formula = as.numeric(factor(cell)) ~ gene99 + group * dosage, data = data)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.95)))
res7[which(res7$Variable == 'gene99'),]

m <- glm.nb(formula = as.numeric(factor(cell)) ~ gene2000 + group * dosage, data = data)
summary(m)
exp(cbind("Odds ratio" = coef(m), confint.default(m, level = 0.95)))
res7[which(res7$Variable == 'gene2000'),]

```

Index

`bayesglmParallel`, [2](#)

`clogitParallel`, [4](#)

`coxphParallel`, [6](#)

`data.table`, [3](#), [5](#), [7](#), [10](#), [12](#), [14](#), [17](#)

`glm.nbParallel`, [9](#)

`glmParallel`, [11](#)

`lmParallel`, [13](#)

`RegParallel`, [15](#)

`RegParallel-package`, [2](#)