

# Package ‘parglms’

April 12, 2018

**Title** support for parallelized estimation of GLMs/GEEs

**Version** 1.10.0

**Author** VJ Carey <stvjc@channing.harvard.edu>

**Description** support for parallelized estimation of GLMs/GEEs,  
catering for dispersed data

**Suggests** RUnit, sandwich, MASS

**Depends** methods

**Imports** BiocGenerics, BatchJobs, foreach, doParallel

**Maintainer** VJ Carey <stvjc@channing.harvard.edu>

**License** Artistic-2.0

**LazyLoad** yes

**BiocViews** statistics, genetics

**ByteCompile** TRUE

**NeedsCompilation** no

## R topics documented:

parglms-package	1
parGLM-methods	3

<b>Index</b>	<b>5</b>
--------------	----------

---

parglms-package	<i>support for parallelized estimation of GLMs/GEEs</i>
-----------------	---

---

## Description

support for parallelized estimation of GLMs/GEEs, catering for dispersed data

**Details**

The DESCRIPTION file:

```

Package:      parglms
Title:       support for parallelized estimation of GLMs/GEEs
Version:     1.10.0
Author:      VJ Carey <stvjc@channing.harvard.edu>
Description: support for parallelized estimation of GLMs/GEEs, catering for dispersed data
Suggests:   RUnit, sandwich, MASS
Depends:    methods
Imports:    BiocGenerics, BatchJobs, foreach, doParallel
Maintainer:  VJ Carey <stvjc@channing.harvard.edu>
License:    Artistic-2.0
LazyLoad:   yes
BiocViews:  statistics, genetics
ByteCompile: TRUE

```

Index of help topics:

```

parGLM-methods      fit GLM-like models with parallelized
                    contributions to sufficient statistics
parglms-package     support for parallelized estimation of
                    GLMs/GEEs

```

In version 0.0.0 we established an approach to fitting GLM from data that have been persistently dispersed and managed by a [Registry](#).

**Author(s)**

VJ Carey <stvjc@channing.harvard.edu>

Maintainer: VJ Carey <stvjc@channing.harvard.edu>

**References**

This package shares an objective with the [bigglm](#) methods of `biglm`. In `bigglm`, a small-RAM-footprint algorithm is employed, with sequential chunking to update statistics in each iteration. In `parGLM` the footprint is likewise controllable, but statistics in each iteration are evaluated in parallel over chunks.

**Examples**

```
showMethods("parGLM")
```

---

parGLM-methods	<i>fit GLM-like models with parallelized contributions to sufficient statistics</i>
----------------	---

---

## Description

This package addresses the problem of fitting GLM-like models in a scalable way, recognizing that data may be dispersed, with chunks processed in parallel, to create low-dimensional summaries from which model fits may be constructed.

## Methods

`signature(formula = "formula", store = "Registry")` The model data are assumed to lie in the `file.dir/jobs/*` folders, with `file.dir` defined in the store, which is an instance of [Registry](#).

Additional arguments must be supplied:

**family** a function that serves as a family for `stats::glm`

**binit** a vector of initial values for regression parameter estimation, must conform to expectations of `formula`

**maxit** an integer giving the maximum number of iterations allowed

**tol** a numeric giving the tolerance criterion

Failure to specify these triggers a fatal error.

The `Registry` instance can be modified to include a list element `'extractor'`. This must be a function with arguments `store`, and `codei`. The standard extraction function is

```
function(store, i) loadResult(store, i)
```

It must return a data frame, conformant with the expectations of `formula`. Limited checking is performed.

The `predict` method computes the linear predictor on data identified by `jobid` in a `BatchJobs` registry. Results are returned as output of `foreach` over the `jobids` specified in the `predict` call.

Note that setting option `parGLM.showiter` to `TRUE` will provide a message tracing progress of the optimization.

## Examples

```
if (require(MASS) & require(BatchJobs)) {
  # here is the 'sharding' of a small dataset
  data(anorexia) # N = 72
  # in .BatchJobs.R:
  # best setting for sharding a small dataset on a small machine:
  # cluster.functions = BatchJobs::makeClusterFunctionsInteractive()
  myr = makeRegistry("abc", file.dir=tempfile())
  chs = chunk(1:nrow(anorexia), n.chunks=18) # 4 recs/chunk
  f = function(x) {library(MASS); data(anorexia); anorexia[x,]}
  batchMap(myr, f, chs)
  submitJobs(myr) # now getResult(myr,1) gives back a data.frame
  waitForJobs(myr) # simple dispersal
  # now myr is populated
  oldopt = options()$parGLM.showiter
  options(parGLM.showiter=TRUE)
  pp = parGLM( Postwt ~ Treat + Prewt, myr,
```

```
family=gaussian, binit = c(0,0,0,0), maxit=10, tol=.001 )
print(summary(theLM <- lm(Postwt~Treat+Prewt, data=anorexia)))
print(pp$coefficients - coef(theLM))
if (require(sandwich)) {
  hc0 <- vcovHC(theLM, type="HC0")
  print(pp$robust.variance - hc0)
}
}
predict(pp, store=myr, jobids=2:3)
options(parGLM.showiter=oldopt)
```

# Index

\*Topic **methods**

parGLM-methods, 3

\*Topic **modeling**

parGLM-methods, 3

\*Topic **package**

parGLMs-package, 1

bigglm, 2

parGLM (parGLM-methods), 3

parGLM, formula, Registry-method  
(parGLM-methods), 3

parGLM-methods, 3

parGLMs (parGLMs-package), 1

parGLMs-package, 1

predict (parGLM-methods), 3

print (parGLM-methods), 3

Registry, 2, 3

summary (parGLM-methods), 3