Package: parglm (via r-universe)

October 28, 2026

Title Parallel GLM

Version 0.2.0

Description Provides a parallel estimation method for generalized linear models without compiling with a multithreaded LAPACK or BLAS.

License GPL-2

URL https://github.com/remlapmot/parglm, https://remlapmot.github.io/parglm/

BugReports https://github.com/remlapmot/parglm/issues

Imports Matrix, parallelly, Rcpp

Suggests biglm, broom, broom.helpers, fastglm, glm2, gtsummary, knitr, lmtest, microbenchmark, rmarkdown, sandwich, speedglm, SuppDists, testthat

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Encoding UTF-8

NeedsCompilation yes

Author Benjamin Christoffersen [aut] (ORCID: <https://orcid.org/0000-0002-7182-1346>), Anthony Williams [cph], Boost developers [cph], Tom Palmer [aut, cre] (ORCID: <https://orcid.org/0000-0003-4655-4511>)

Maintainer Tom Palmer <remlapmot@hotmail.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-07-06 14:40:44 UTC

RemoteUrl https://github.com/cran/parglm

RemoteRef HEAD

RemoteSha 8175ffe6f42c8b9b35dfa607df7131ad2dc74980

Contents

parglm 2
parglm.control 4
tidy_parglm_robust 5

Index **7**

---

parglm *Fitting Generalized Linear Models in Parallel*

Description

A function like `glm` which can make the computation in parallel. The function supports most families listed in `family`. See "`vignette("parglm", "parglm")`" for run time examples.

Usage

```
parglm(
  formula,
  family = gaussian,
  data,
  weights,
  subset,
  na.action,
  start = NULL,
  offset,
  control = list(...),
  contrasts = NULL,
  model = TRUE,
  x = FALSE,
  y = TRUE,
  ...
)

parglm.fit(
  x,
  y,
  weights = rep(1, NROW(x)),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep(0, NROW(x)),
  family = gaussian(),
  control = list(),
  intercept = TRUE,
  ...
)
```

Arguments

formula	an object of class formula.
family	a family object.
data	an optional data frame, list or environment containing the variables in the model.
weights	an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	a function which indicates what should happen when the data contain NAs.
start	starting values for the parameters in the linear predictor.
offset	this can be used to specify an a priori known component to be included in the linear predictor during fitting.
control	a list of parameters for controlling the fitting process. For parglm.fit this is passed to parglm.control.
contrasts	an optional list. See the contrasts.arg of model.matrix.default.
model	a logical value indicating whether model frame should be included as a component of the returned value.
x, y	For parglm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value. For parglm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.
...	For parglm: arguments to be used to form the default control argument if it is not supplied directly. For parglm.fit: unused.
etastart	starting values for the linear predictor. Not supported.
mustart	starting values for the vector of means. Not supported.
intercept	logical. Should an intercept be included in the null model?

Details

The current implementation uses min(as.integer(n / p), nthreads) threads where n is the number of observations, p is the number of covariates, and nthreads is the nthreads element of the list returned by parglm.control. Thus, there is likely little (if any) reduction in computation time if p is almost equal to n. The current implementation cannot handle p > n.

Since parglm returns a standard glm object, it is compatible with the **sandwich** package for heteroskedasticity-consistent (HC) and cluster-robust standard errors via vcovHC and vcovCL. This requires model = TRUE (the default). See vignette("sandwich", "parglm") for examples.

Value

glm object as returned by glm but differs mainly by the qr element. The qr element in the object returned by parglm(.fit) only has the R matrix from the QR decomposition.

Examples

```
# mtcars has 32 rows, sufficient for 2 threads (>= 16 rows per thread)
f1 <- glm   (mpg ~ wt + hp, data = mtcars, family = Gamma(link = "log"))
f2 <- parglm(mpg ~ wt + hp, data = mtcars, family = Gamma(link = "log"),
             control = parglm.control(nthreads = 2L))
all.equal(coef(f1), coef(f2))
```

`parglm.control` *Auxiliary for Controlling GLM Fitting in Parallel*

Description

Auxiliary function for `parglm` fitting.

Usage

```
parglm.control(
  epsilon = 1e-08,
  maxit = 25,
  trace = FALSE,
  nthreads = parallelly::availableCores(omit = 1L),
  block_size = NULL,
  method = "LINPACK",
  nthreads_auto = missing(nthreads)
)
```

Arguments

`epsilon`	positive convergence tolerance.
`maxit`	integer giving the maximal number of IWLS iterations.
`trace`	logical indicating if output should be produced during estimation.
`nthreads`	number of cores to use. Defaults to `parallelly::availableCores(omit = 1L)`, which leaves one core free. You may get the best performance by using all available physical cores if your data set is sufficiently large.
`block_size`	number of observations to include in each parallel block.
`method`	string specifying which method to use. Either `"LINPACK"`, `"LAPACK"`, or `"FAST"`.
`nthreads_auto`	logical; for internal use only. Records whether `nthreads` was auto-detected (suppresses the thread-reduction warning when the dataset is small). Do not set this argument directly.

Details

The LINPACK method uses the same QR method as glm.fit for the final QR decomposition. This is the dqrdc2 method described in qr. All other QR decompositions except the last are made with DGEQP3 from LAPACK. See Wood, Goude, and Shaw (2015) for details on the QR method.

The FAST method computes the Fisher information and then solves the normal equation. This is faster but less numerically stable.

Value

A list with components named as the arguments.

References

Wood, S.N., Goude, Y. & Shaw, S. (2015) Generalized additive models for large datasets. Journal of the Royal Statistical Society, Series C 64(1): 139-155.

Examples

```
# use one core
f1 <- parglm(mpg ~ wt + hp, data = mtcars, family = Gamma(link = "log"),
             control = parglm.control(nthreads = 1L))

# use two cores (mtcars has 32 rows, sufficient for 2 threads)
f2 <- parglm(mpg ~ wt + hp, data = mtcars, family = Gamma(link = "log"),
             control = parglm.control(nthreads = 2L))
all.equal(coef(f1), coef(f2))
```

---

tidy_parglm_robust *Tidy a parglm model with robust standard errors*

Description

A drop-in tidy_fun for tbl_regression that computes heteroskedasticity-consistent (HC) or cluster-robust confidence intervals via **sandwich** and **lmtest**.

Usage

```
tidy_parglm_robust(
  x,
  vcov. = "HC3",
  conf.int = TRUE,
  conf.level = 0.95,
  exponentiate = FALSE,
  ...
)
```

Arguments

x	a parglm (or glm) model object.
vcov.	the robust variance-covariance estimator. A string is passed as the type argument to vcovHC (e.g. "HC3"). A function is called as vcov.(x) and should return a covariance matrix (use this for cluster-robust SEs via vcovCL). A matrix is used directly. Defaults to "HC3".
conf.int	logical; whether to include confidence intervals.
conf.level	confidence level for the intervals.
exponentiate	logical; whether to exponentiate the estimate and confidence interval limits.
...	unused; present for compatibility with the tidy_fun interface of tbl_regression.

Details

Pass this function as tidy_fun to tbl_regression:

```
# HC3 (default)
tbl_regression(fit, tidy_fun = tidy_parglm_robust)

# HC1
tbl_regression(fit, tidy_fun = \(x, ...) tidy_parglm_robust(x, vcov. = "HC1", ...))

# Cluster-robust
tbl_regression(fit, tidy_fun = \(x, ...) tidy_parglm_robust(
  x, vcov. = \(m) sandwich::vcovCL(m, cluster = ~ cluster_var), ...))
```

Value

a data.frame with columns term, estimate, std.error, statistic, p.value, and (when conf.int = TRUE) conf.low and conf.high.

Examples

```
fp <- parglm(mpg ~ wt + hp, data = mtcars,
             control = parglm.control(nthreads = 1L))
if (requireNamespace("sandwich", quietly = TRUE) &&
    requireNamespace("lmtest",   quietly = TRUE)) {
  tidy_parglm_robust(fp)
}
```

Index

family, *2, 3*
formula, *3*

glm, *2, 3*
glm.fit, *5*

model.matrix.default, *3*

parglm, 2, *4*
parglm.control, *3*, 4

qr, *5*

tbl_regression, *5, 6*
tidy_parglm_robust, 5

vcovCL, *3*, *6*
vcovHC, *3*, *6*