Package 'bgw'

April 6, 2023

Type Package

Title Bunch-Gay-Welsch Statistical Estimation

Version 0.1.1

Maintainer David S. Bunch <dsbunch@ucdavis.edu>

Description Performs statistical estimation and inference-related computations by accessing and executing modified versions of 'Fortran' subroutines originally published in the Association for Computing Machinery (ACM) journal Transactions on Mathematical Software (TOMS) by Bunch, Gay and Welsch (1993) <doi:10.1145/151271.151279>. The acronym 'BGW' (from the authors' last names) will be used when making reference to technical content (e.g., algorithm, methodology) that originally appeared in ACM TOMS. A key feature of BGW is that it exploits the special structure of statistical estimation problems within a trust-region-based optimization approach to produce an estimation algorithm that is much more effective than the usual practice of using optimization methods and codes originally developed for general optimization. The 'bgw' package bundles 'R' wrapper (and related) functions with modified 'Fortran' source code so that it can be compiled and linked in the 'R' environment for fast execution. This version implements a function ('bgw_mle.R') that performs maximum likelihood estimation (MLE) for a user-provided model object that computes probabilities (a.k.a. probability densities). The motivation for producing this initial version is to provide fast, efficient, and reliable MLE for discrete choice models that can be called from the 'Apollo' choice modelling 'R' package: see <http://www.apollochoicemodelling.com>. However, estimation can also be performed in a stand-alone fashion without using 'Apollo' (as shown in simple examples). After this initial version is available on CRAN, an updated version of 'Apollo' (0.2.9) will be made available that automatically loads 'bgw'. Additional development can then occur, including more detailed examples in 'bgw' that refer to 'Apollo.' Note also that BGW capabilities are not limited to MLE, and future extension to other estimators (e.g., nonlinear least squares, generalized method of moments, etc.) is possible. The 'Fortran' code included in 'bgw' was modified by one of the original BGW authors (Bunch) under his rights as confirmed by direct consultation with the ACM Intellectual Property and Rights Manager. See <https://authors.acm.org/author-resources/author-rights>. The main

requirement is clear citation of the original publication (see above).

License GPL-3

Encoding UTF-8

NeedsCompilation yes

Depends R (>= 4.0.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0),

VignetteBuilder knitr

RoxygenNote 7.2.3

Author David S. Bunch [aut, cre] (<https://orcid.org/0000-0001-8728-7072>),
David M. Gay [ctb],
Roy E. Welsch [ctb],
Stephane Hess [ctb],
David Palma [ctb]

Config/testthat/edition 3

Repository CRAN

Date/Publication 2023-04-06 17:50:02 UTC

R topics documented:

`bgw_checkSetting` *bgw_checkSetting*

Description

Checks to see if user-provided value for a bgw_setting is valid.

Usage

```
bgw_checkSetting(
  settingValue,
  bgw_setting_type,
  bgw_validDiscrete,
  bgw_contLB,
  bgw_contUB
)
```

Arguments

`settingValue`	Setting value (submitted by user).
`bgw_setting_type`	Type of setting being checked. Possible values are "discrete" and "continuous".
`bgw_validDiscrete`	List. Contains valid values for a discrete setting.
`bgw_contLB`	Numerical value. The lower bound for a valid continuous setting.
`bgw_contUB`	Numerical value. The upper bound for a valid continuous setting.

Value

Logical. Indicates if the setting is okay or not.

`bgw_drglg` *bgw_drglg*

Description

An r wrapper to call drglg_c, which in turn is a wrapper to call the Fortran subroutine drglg. drglg is the BGW iteration driver for performing statistical parameter estimation

Usage

```
bgw_drglg(
  d,
  dr,
  iv,
  liv,
  lv,
  n,
  nd,
  nn,
  p,
  ps,
  r,
  rd,
  v,
  x,
  rhoi,
  rhor,
  i_itsum
)
```

Arguments

d	Scaling vector
dr	Derivative of the choice probability model wrt x
iv	BGW internal vector of integer values
liv	Length of iv.
lv	Length of v.
n	Dimension of vector (r) of generalized residuals for the model
nd	Leading dimension dr. Must be at least ps.
nn	Leading dimension of r, rd
p	Dimension of x (as well as d, g) = number of parameters being estimated
ps	Number of non-nuisance parameters (= p in this implementation)
r	Vector of generalized residuals for the model
rd	Vector of storage space for regression diagonostics (not currently used)
v	BGW internal vector of numeric values
x	Parameter vector for which the objective function is being minimized
rhoi	Vector of integers for use by user (not currently used)
rhor	Vector of numeric values for use by user (not currently used)
i_itsum	Variable for passing itsum instruction back to bgw_mle

Value

out List of return values.

`bgw_itsum` *bgw_itsum*

Description

Prints iteration summary, info on initial and final x.

Usage

```
bgw_itsum(d, g, iv, v, x, p, betaIsNamed, betaNames = NULL, i_itsum)
```

Arguments

d	Scaling vector
g	Gradient of objective function (negative-log-likelihood) wrt x
iv	BGW internal vector of integer values
v	BGW internal vector of numeric values
x	Parameter vector for which the objective function is being minimized
p	Dimension of x
betaIsNamed	Logical.
betaNames	Character vector. If available, has beta parameter names.
i_itsum	Code from caller to select specific options

Value

iv There are iv values changed inside bgw_itsum. The iv vector is the integer workspace for Fortran BGW.

---

bgw_mle *bgw_mle*

Description

Performs maximum likelihood estimation (MLE) for the user-provided model defined in `bgw_calcR`.

Usage

```
bgw_mle(calcR, betaStart, calcJ = NULL, bgw_settings = NULL)
```

Arguments

calcR	Function that computes an n-vector (R) of model residuals for a p-vector of (numeric) parameters beta (the first argument). In this case the residuals are likelihoods (probabilities). (The beta vector can be named or unnamed.)
betaStart	Vector of initial starting values for beta. Can be either a named or unnamed vector.
calcJ	Function that computes the matrix of partial derivatives of R wrt beta (a.k.a. the Jacobian). If NULL, finite-difference derivatives are used. In matrix form, dim=c(p,n). However, it could be stored as a vector in column-major order.
bgw_settings	List. Contains control parameters for BGW estimation code. All parameters have default values, so user input is entirely optional.

- `printLevel`: Integer (0-3). Controls the level of detail for iteration information written to the console during estimation. 0 = silent, 1 = print starting values and final solution only, 2 = short line summary, 3 = long line summary (see documentation). Default = 3.

- `silent`: Logical. Suppresses all output. Default = FALSE. (Currently redundant with printLevel = 0.)
- `printNonDefaultSettings`: Logical. Echos any user-provided bgw_settings. Default = TRUE.
- `printStartingValues`: Logical. Default = TRUE. Print starting values for beta.
- `printFinalResults`: Logical. Default = TRUE. Print final status of estimation (convergence/error message), summary statistics (negative log-likelihood, iterations, function evals, etc.), beta, gradient, and (if available) estimated standard errors and t-ratios. (See vcHessianMethod for variance-covariance specification.)
- `maxIterations`: Numeric. Maximum number of iterations for the estimation. BGW default is 150.
- `maxFunctionEvals`: Numeric. Maximum number of objective function evaluations. BGW default is 200.
- `modelName`: Character. Used to create names for requested output files. Default is "bgw_mle_model."
- `outputDirectory`: Character. Name of sub-directory (~/outputDirectory) to write output files. Default is NULL (current directory).
- `writeItSummary`: Logical. If TRUE, iteration information is echoed in the output file "modelName_itSummary.csv". Default is FALSE. [Not currently implemented.]
- `writeIter`: Logical. If TRUE, parameters and log-likelihood for each iteration are written to "modelName_iterations.csv". Default is FALSE.
- `vcHessianMethod`: Character. Method for computing the Hessian approximation used for the variance-covariance matrix (VC = H^(-1)). Options are: "none","bhhh","finiteDifferences", or "fdFunction" ("finiteDifferences" automatically uses gradient differences if a gradient is available, otherwise it uses objective function differences. "fdFunction" allows the user to use objective function differences even if a gradient is available). Default is "bhhh."
- `scalingMethod`: Character. Method used to compute a scaling vector (scaleVec_i, i=1,..,p). Define a matrix D = diag(scaleVec_1,..., scaleVec_p). Values in scaleVec are chosen so that the elements of D*beta are roughly comparable in size. The re-scaled beta is used when computing trial steps using trust regions, and when computing stopping criteria. Options are: "adaptive" and "none." The "adaptive" method is described in Bunch, Gay, and Welsch (1993), where scaleVec is updated at each iteration. For "none," scaleVec is set to a p-vector of ones for the entire search. Default is "adaptive."

Details

This function has been written to provide an R-based interface to Fortran estimation software published in Bunch, Gay and Welsch (1993), "Algorithm 717-Subroutines for Maximum Likelihood and Quasi-Likelihood Estimation of Parameters in Nonlinear Regression Models," ACM Transactions on Mathematical Software, 19 (1), March 1993, 109-130. The letters BGW will be used in various ways to denote the source of the estimation functionality.

A primary motivation was to develop a more efficient maximum likelihood estimation function for use in the Apollo choice modelling package: see http://www.apollochoicemodelling.com/. However, we have adopted a design whereby the BGW package is wholly independent of Apollo, and can be used in a stand-alone fashion. Note also that the BGW Fortran subroutines are written to support general statistical estimation for an arbitrary objective/criterion function. So, although this version of the package is specifically written for MLE, the package may see future updates that expand the number of estimation options (for, e.g., nonlinear least squares, generalized method of moments, etc.).

Remark: Following the convention in the numerical optimization literature, BGW minimizes the objective function. That is, bgw_mle minimizes the negative-log-likelihood for the model calcR.

Value

model object of class 'bgw_mle'. Output of a bgw maximum likelihood estimation procedure. A list with the following attributes:

- `betaStart`: Vector of initial starting values.
- `bgw_settings`: List. The same as the input argument.
- `hasAnalyticGrad`: Logical. Indicates in an analytical gradient calculation was used. If the user has not provided a calcJ function (see input parameter), it is set to FALSE.
- `numParams`: Numeric. Number of model parameters used in calcR.
- `numResids`: Numeric. Number of independent observations (model residuals) in data set = dimension of calcR output.
- `code`: Numeric. Numeric return code from BGW.
- `message`: Character. Message statement characterizing termination of MLE search.
- `betaStop`: Vector. Value of parameter vector at conclusion of MLE search. See message to determine if beta is a valid estimate.
- `finalLL`: Numeric. Value of log-likelihood at betaStop.
- `iterations`: Numeric. Number of iterations used in MLE search. In BGW, this is the same as the number of gradient evaluations.
- `functionEvals`: Numeric. Number of function evaluations used in MLE search. (This excludes function evaluations used by any finite-difference calculations for the gradient and/or the vcHessian.
- `gradient`: Vector. The gradient evaluated at betaStop.
- `scaleVec`: Vector. The scaling vector at the conclusion of the MLE search. Note: In the current version, this will be a p-vector of 1's (used throughout the search). In future versions, additional scaling options may be implemented.
- `estimate`: Vector. MLE parameter vector obtained by BGW. The same as betaStop if a valid convergence condition is achieved. Null otherwise.
- `maximum`: Numeric. Final log-likelihood value for a (successful) MLE search.
- `hessianMethodAttempted`: String. Requested method for computing vcHessian (from bgw_settings).
- `hessianMethodUsed`: String. Method actually used for computing vcHessian (if vcHessian was requested, and if the computation was successful. If not, a message indicating 'no request' or 'singular vcHessian' is provided.)

- vcHessianConditionNumber: Numeric. Estimated upper bound on reciprocal of Euclidean condition number of vcHessian (if available). Set to -1 if unavailable.
- varcovBGW: Matrix. p-by-p matrix containing estimate of the variance-covariance matrix (if requested and available).
- vcVec: Vector. Lower triangle of variance-covariance matrix stored in vector form (row-major order, if requested and available).
- seBGW: Vector. Estimated standard errors for parameter estimates (if requested and available).
- tstatBGW: Vector. Estimated t-statistics (versus 0, if requested and available).

---

bgw_mle_setup *bgw_mle_setup*

Description

Sets up R-level storage for bgw_mle.

This function replaces multiple Fortran subroutines from the BGW Fortran code due at least in part to the prohibition against using Fortran write statements in R packages. The current design produces two vectors (iv_r and v_r) that mirror the main vectors required by the Fortran code. For the moment, the idea is to create named vectors to facilitate coding in bgw_mle. It may be that these could be deleted (or overwritten by an as.numeric() conversion) prior to the main Fortran calls.

Usage

```
bgw_mle_setup(p, n, hasAnalyticModelDeriv, control = NULL)
```

Arguments

p	Number of parameters (components of x) being estimated. (Determined in bgw_mle from size of 'start' vector.
n	Number of model residuals (in vector r). (Determined in bgw_mle by the size of the output vector from CalcR.)
hasAnalyticModelDeriv	Logical. TRUE if CalcRJ has been provided. FALSE means that bgw_mle must employ finite-difference gradients (which has implications for storage allocation).
control	List of bgw_mle control parameters (optional). If not provided, BGW default parameters will be used. If provided, default parameters will be overwritten by those corresponding parameters provided by the caller (but these must also be checked).

Value

iv and v vectors used by BGW Fortran.

bgw_writeIterations *Writes the vector [beta,ll] to a file called* modelname_iterations.csv *# Was created using apollo_writeTheta as a starting point... Because this is an internal function, the inputs will be assumed to be clean.*

Description

Writes the vector [beta,ll] to a file called modelname_iterations.csv # Was created using apollo_writeTheta as a starting point... Because this is an internal function, the inputs will be assumed to be clean.

Usage

```
bgw_writeIterations(beta, ll, outputFile)
```

Arguments

beta	vector of parameters to be written (for now, no fixed betas).
ll	scalar representing the log-likelihood of the whole model.
outputFile	Character. Name of the output file.

Value

Nothing.

Index