Package ‘WALS’

November 3, 2023

Version 0.2.4

Date 2023-11-02

Title Weighted-Average Least Squares Model Averaging

Depends R (>= 4.0.0)

Imports Formula (>= 1.2-3), MASS (>= 7.3-51.6), methods, Rdpack(>= 2.1.3), stats

Suggests AER, BayesVarSel, BMS, testthat (>= 3.1.10)

Description Implements Weighted-Average Least Squares model averaging for negative binomial regression models of Huynh (2023) (mimeo), generalized linear models of De Luca, Magnus, Peracchi (2018) <doi:10.1016/j.jeconom.2017.12.007> and linear regression models of Magnus, Powell, Pruefer (2010) <doi:10.1016/j.jeconom.2009.07.004>, see also Magnus, De Luca (2016) <doi:10.1111/joes.12094>. Weighted-Average Least Squares for the linear regression model is based on the original 'MATLAB' code by Magnus and De Luca <https://www.janmagnus.nl/items/WALS.pdf>, see also Kumar, Magnus (2013) <doi:10.1007/s13571-013-0060-9> and De Luca, Magnus (2011) <doi:10.1177/1536867X1201100402>.

License GPL-2 | GPL-3

URL https://github.com/kevhuy/WALS

BugReports https://github.com/kevhuy/WALS/issues

LazyData true

RdMacros Rdpack

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Kevin Huynh [aut, cre] (<https://orcid.org/0000-0002-4621-2274>)

Maintainer Kevin Huynh <Kevin.Huynh@unibas.ch>

Repository CRAN

Date/Publication 2023-11-03 07:20:06 UTC

R topics documented:

`checkSingularitySVD` *Internal function: Check singularity of SVDed matrix*

Description

Checks whether matrix is singular based on singular values of SVD.

Usage

```
checkSingularitySVD(singularValues, tol, rtol, digits = 5)
```

Arguments

`singularValues`	Vector of singular values.
`tol`	Absolute tolerance, singular if `min(singularValues)` < `tol`
`rtol`	Relative tolerance, singular if `min(singularValues)` / `max(singularValues)` < `rtol`
`digits`	The number significant digits to show in case a warning is triggered by singularity.

`computeGamma1` *Internal function: Compute model-averaged estimator of focus regressors in walsNB*

Description

Exploits the SVD of the design matrix of the focus regressors $\bar{Z}_1$, the model-averaged estimator for the auxiliary regressors $\hat{\gamma}_2$ and the Sherman-Morrison-Woodbury formula for computing the model-averaged estimator of the focus regressors in walsNB.

Usage

```
computeGamma1(
  gamma2,
  Z2start,
  Z2,
  U,
  V,
  singularVals,
  ellStart,
  gStart,
  epsilonStart,
  qStart,
  y0Start,
  tStart,
  psiStart
)
```

Arguments

`gamma2`	Model-averaged estimate for auxiliary regressors from `computePosterior`.
`Z2start`	Transformed design matrix of auxiliary regressors $\bar{Z}_2$. See details.
`Z2`	Another transformed design matrix of auxiliary regressors Z_2. See details.
`U`	Left singular vectors of $\bar{Z}_1$ from `svd`.
`V`	Right singular vectors of $\bar{Z}_1$ from `svd`.
`singularVals`	Singular values of $\bar{Z}_1$ from `svd`.

ellStart	Vector $\bar{\ell}$ see details.
gStart	Derivative of dispersion parameter ρ of NB2 with respect to $\alpha = \log(\rho)$ evaluated at starting values of one-step ML. gStart is a scalar. See section "ML estimation" of Huynh (2023a).
epsilonStart	Scalar $\bar{\epsilon}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
qStart	Vector $\bar{q}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
y0Start	Vector $\bar{y}_0$, see section "One-step ML estimator" of Huynh (2023a) for definition.
tStart	Scalar $\bar{t}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
psiStart	Diagonal matrix $\bar{\Psi}$, see section "One-step ML estimator" of Huynh (2023a) for definition.

Details

See section "Simplification for computing $\hat{\gamma}_1$" in the appendix of Huynh (2023b) for details of the implementation and for the definitions of argument `ellStart`.

All parameters that contain "start" feature the starting values for the one-step ML estimation of submodels. See section "One-step ML estimator" of Huynh (2023a) for details.

The argument `Z2start` is defined as (Huynh 2023a)

$$\bar{Z}_2 := \bar{X}_2 \bar{\Delta}_2 \bar{\Xi}^{-1/2},$$

and `Z2` is defined as

$$Z_2 := X_2 \bar{\Delta}_2 \bar{\Xi}^{-1/2}.$$

Uses `svdLSplus` under-the-hood.

References

Huynh K (2023a). “Weighted-Average Least Squares for Negative Binomial Regression.” University of Basel. Mimeo.

Huynh K (2023b). “WALS: Weighted-Average Least Squares Model Averaging in R.” University of Basel. Mimeo.

`computeGamma1r` *Internal function: Computes fully restricted one-step ML estimator for transformed regressors in walsNB*

Description

Computes one-step ML estimator of fully restricted model (coefs of transformed regressors of $\bar{Z}_1$) in walsNB by using SVD on transformed design matrix of the focus regressors $\bar{Z}_1$. The matrix $\bar{Z}_1$ should have full column rank.

Usage

```
computeGamma1r(
  U,
  V,
  singularVals,
  ellStart,
  gStart,
  epsilonStart,
  qStart,
  y0Start,
  tStart,
  psiStart
)
```

Arguments

`U`	Left singular vectors of $\bar{Z}_1$ from `svd`.
`V`	Right singular vectors of $\bar{Z}_1$ from `svd`.
`singularVals`	Singular values of $\bar{Z}_1$ from `svd`.
`ellStart`	Vector $\bar{\ell}$ see details.
`gStart`	Derivative of dispersion parameter ρ of NB2 with respect to $\alpha = \log(\rho)$ evaluated at starting values of one-step ML. `gStart` is a scalar. See section "ML estimation" of Huynh (2023a).
`epsilonStart`	Scalar $\bar{\epsilon}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`qStart`	Vector $\bar{q}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`y0Start`	Vector $\bar{y}_0$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`tStart`	Scalar $\bar{t}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`psiStart`	Diagonal matrix $\bar{\Psi}$, see section "One-step ML estimator" of Huynh (2023a) for definition.

Details

See section "Simplification for computing $\tilde{\gamma}_{1r}$" in the appendix of Huynh (2023b) for details of the implementation and for the definitions of argument `ellStart`.

All parameters that contain "start" feature the starting values for the one-step ML estimation of submodels. See section "One-step ML estimator" of Huynh (2023a) for details.

Uses `svdLSplus` under-the-hood.

References

Huynh K (2023a). “Weighted-Average Least Squares for Negative Binomial Regression.” University of Basel. Mimeo.

Huynh K (2023b). “WALS: Weighted-Average Least Squares Model Averaging in R.” University of Basel. Mimeo.

`computeGammaUnSVD` *Internal function: Computes unrestricted one-step ML estimator for transformed regressors in walsNB*

Description

Computes one-step ML estimator for the unrestricted model in walsNB (coefs of transformed regressors $\bar{Z}$) by using SVD on entire transformed design matrix $\bar{Z}$. The matrix $\bar{Z}$ should have full column rank.

Usage

```
computeGammaUnSVD(
  U,
  V,
  singularVals,
  ellStart,
  gStart,
  epsilonStart,
  qStart,
  y0Start,
  tStart,
  psiStart
)
```

Arguments

`U`	Left singular vectors of $\bar{Z}$ or $\bar{Z}_1$ from `svd`.
`V`	Right singular vectors of $\bar{Z}$ or $\bar{Z}_1$ from `svd`.
`singularVals`	Singular values of $\bar{Z}$ or $\bar{Z}_1$ from `svd`.
`ellStart`	Vector $\bar{\ell}$ see details.
`gStart`	Derivative of dispersion parameter ρ of NB2 with respect to $\alpha = \log(\rho)$ evaluated at starting values of one-step ML. `gStart` is a scalar. See section "ML estimation" of Huynh (2023a).
`epsilonStart`	Scalar $\bar{\epsilon}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`qStart`	Vector $\bar{q}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`y0Start`	Vector $\bar{y}_0$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`tStart`	Scalar $\bar{t}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
`psiStart`	Diagonal matrix $\bar{\Psi}$, see section "One-step ML estimator" of Huynh (2023a) for definition.

Details

See section "Simplification for computing $\tilde{\gamma}_u$" in the appendix of Huynh (2023b) for details of the implementation and for the definitions of argument `ellStart`.

All parameters that contain "start" feature the starting values for the one-step ML estimation of submodels. See section "One-step ML estimator" of Huynh (2023a) for details.

Uses `svdLSplus` under-the-hood.

References

Huynh K (2023a). "Weighted-Average Least Squares for Negative Binomial Regression." University of Basel. Mimeo.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

`computePosterior` *Internal function: Compute posterior mean and variance of normal location problem*

Description

Computes the posterior mean and variance of the normal location problem with fixed variance to 1, i.e. $x|\gamma \sim N(\gamma, 1)$. The priors for γ are either `weibull`, `subbotin` or `laplace`. Their properties are briefly discussed in Magnus and De Luca (2016). Default method of computePosterior uses numerical integration. This is used for the `weibull` and `subbotin` priors. For the `laplace` prior closed form expressions exist for the integrals. In the original MATLAB code, the Gauss-Kronrod quadrature was used for numerical integration. Here we use the default `integrate` which combines Gauss-Kronrod with Wynn's Epsilon algorithm for extrapolation.

Usage

```
computePosterior(object, ...)

## S3 method for class 'familyPrior'
computePosterior(object, x, ...)

## S3 method for class 'familyPrior_laplace'
computePosterior(object, x, ...)
```

Arguments

`object`	Object of class "`familyPrior`", e.g. from `weibull`, should contain all necessary parameters needed for the posterior.
`...`	Further arguments passed to methods.
`x`	vector. Observed values, i.e. in WALS these are the regression coefficients of the transformed regressor Z2 standardized by the standard deviation: γ_{2u}/s.

Details

See section "Numerical integration in Bayesian estimation step" in the appendix of Huynh (2023b) for details.

`computePosterior.familyPrior_laplace()` is the specialized method for the S3 class "`familyPrior_laplace`" and computes the posterior first and second moments of the normal location problem with a Laplace prior using the analytical formula (without numerical integration). For more details, see De Luca et al. (2020) and the original code of Magnus and De Luca.

References

De Luca G, Magnus JR, Peracchi F (2020). "Posterior moments and quantiles for the normal location model with Laplace prior." *Communications in Statistics - Theory and Methods*, **0**(0), 1-11. doi:10.1080/03610926.2019.1710756.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

Original MATLAB code on Jan Magnus' website. `https://www.janmagnus.nl/items/WALS.pdf`

`computeX2M1X2` *Internal function: Computes X2M1X2 for walsNB when SVD is applied to Z1*

Description

Exploits the SVD of $\bar{Z}_1$ to compute $\bar{X}_2^\top \bar{M}_1 \bar{X}_2$ to avoid directly inverting $\bar{Z}_1^\top \bar{Z}_1$.

Usage

```
computeX2M1X2(
  X2,
  X2start,
  qStart,
  U,
  UellStart,
  ellStart,
  psiStart,
  gStart,
  epsilonStart,
  geB
)
```

Arguments

X2	Design matrix for auxiliary regressors
X2start	Transformed design matrix for auxiliary regressors. Refers to $\bar{X}_2 = \bar{\Psi}^{1/2} X_2$.
qStart	Vector $\bar{q}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
U	U of SVD of Z_1. See details.
UellStart	Vector $U\bar{\ell}$, see details.
ellStart	Vector $\bar{\ell}$ see details.
psiStart	Diagonal matrix $\bar{\Psi}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
gStart	Derivative of dispersion parameter ρ of NB2 with respect to $\alpha = \log(\rho)$ evaluated at starting values of one-step ML. `gStart` is a scalar. See section "ML estimation" of Huynh (2023a).
epsilonStart	Scalar $\bar{\epsilon}$, see section "One-step ML estimator" of Huynh (2023a) for definition.
geB	$\bar{g}\bar{\epsilon}/(1 + B)$. In code `gStart*epsilonStart / (1+B)`. See details for definition of B. `gStart` is $\bar{g}$ and `epsilonStart` is $\bar{\epsilon}$.

Details

See section "Simplification for computing $\bar{X}_2^{\top} \bar{M}_1 \bar{X}_2$" in the appendix of Huynh (2023b) for details of the implementation and for the definitions of arguments `Uellstart`, `ellStart`, and `geB`.

All parameters that contain "start" feature the starting values for the one-step ML estimation of submodels. See section "One-step ML estimator" of Huynh (2023a) for details.

References

Huynh K (2023a). “Weighted-Average Least Squares for Negative Binomial Regression.” University of Basel. Mimeo.

Huynh K (2023b). “WALS: Weighted-Average Least Squares Model Averaging in R.” University of Basel. Mimeo.

`controlGLM` *Control function for initial GLM fit*

Description

Defines controllable parameters of initial GLM fit in `walsGLM`.

Usage

```
controlGLM(restricted = FALSE, controlGLMfit = list())
```

Arguments

`restricted` If TRUE, then initial fit in `glm.fit` only considers the focus regressors. By default FALSE, then the unrestricted model is estimated in `glm.fit` (i.e. all regressors).

`controlGLMfit` List. Arguments to be passed to `control` argument of `glm.fit`. See also `glm.control`.

Value

Returns a list containing the parameters specified in the arguments to be used in `walsGLM` (and `walsGLMfitIterate`).

See Also

walsGLM, walsGLMfitIterate, glm.fit, glm.control.

Examples

```
data("HMDA", package = "AER")
fitBinomial <- walsGLM(deny ~ pirat + hirat + lvrat + chist + mhist + phist |
                        selfemp + afam, data = HMDA,
                        family = binomialWALS(),
                        prior = weibull(),
                        controlInitGLM = controlGLM(restricted = TRUE,
                                                    controlGLMfit = list(trace = TRUE)))
```

`controlNB` *Control function for initial NB fit*

Description

Defines controllable parameters of initial NB fit in `walsNB`.

Usage

```
controlNB(
  start = list(mu = NULL, logTheta = NULL),
  method = "BFGS",
  controlOptim = list(maxit = 100),
  initThetaMASS = FALSE,
  initMASS = TRUE,
  restricted = FALSE,
  eps = .Machine$double.eps^0.25,
  epsilonMASS = 1e-08
)
```

Arguments

start	Optional starting values for fitNB2. Only used if initMASS = FALSE.
method	Optimization method used in optim. Only used if initMASS = FALSE.
controlOptim	List with parameters controlling optimization process of optim. Only used if initMASS = FALSE.
initThetaMASS	If TRUE, then initial $\log\theta$ of fitNB2 is estimated using theta.ml (ML-estimation over 1 variable) based on regression coefficients from Poisson regression. If FALSE, then initial $\log\theta = 0$ is used.
initMASS	If TRUE (default), then initial fit in fitNB2 is estimated via glm.nb and initThetaMASS is ignored. If FALSE, then the initial fit is estimated by minimizing the log-likelihood using optim.
restricted	If TRUE, then initial fit in fitNB2 only considers the focus regressors. By default FALSE, then the unrestricted model is estimated in fitNB2 (i.e. all regressors).
eps	Controls argument eps in fitNB2 for generating starting value for logTheta ($\log\theta$) via theta.ml.
epsilonMASS	Sets epsilon in control argument of glm.nb.

Value

Returns a list containing the parameters specified in the arguments to be used in walsNB (and walsNBfitIterate).

See Also

walsNB, walsNBfitIterate.

Examples

```
data("NMES1988", package = "AER")
walsNB(visits ~ health + chronic + age + gender | I((age^2)/10) +
       married + region, data = NMES1988, prior = weibull(),
       controlInitNB = controlNB(initMASS = FALSE, restricted = TRUE))
```

---

ddweibull *Internal function: double (reflected) Weibull density*

Description

Wrapper around dweibull to use the parametrization on pp. 131 of Magnus and De Luca (2016).

Usage

```
ddweibull(x, q, b, log = FALSE)
```

Arguments

x	vector of quantiles.
q	q in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
b	c in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
log	logical; if TRUE, probabilities p are given as log(p).

Details

The density function is
$$\pi(x) = \frac{qc}{2}|x|^{q-1}\exp(-c|x|^q).$$

Value

Gives the (log-)density.

References

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

See Also

weibull, dweibull.

---

dlaplace *Internal function: Laplace density*

Description

Wrapper around `dsubbotin` with fixed `q = 1`. Uses the parametrization on pp. 131 of Magnus and De Luca (2016).

Usage

```
dlaplace(x, b, log = FALSE)
```

Arguments

x	vector of quantiles.
b	c in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
log	logical; if TRUE, probabilities p are given as log(p).

Details

The density function is

$$\pi(x) = \frac{c}{2} \exp(-c|x|).$$

Value

Gives the (log-)density.

References

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

See Also

laplace, dsubbotin.

`dsubbotin` *Internal function: Subbotin density*

Description

Subbotin density, uses the parametrization on pp. 131 of Magnus and De Luca (2016). Also called generalized normal distribution.

Usage

```
dsubbotin(x, q, b, log = FALSE)
```

Arguments

`x`	vector of quantiles.
`q`	q in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
`b`	c in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
`log`	logical; if TRUE, probabilities p are given as log(p).

Details

The density function is

$$\pi(x) = \frac{qc^{1/q}}{2\Gamma(1/q)} \exp(-c|x|^q).$$

Value

Gives the (log-)density.

References

Magnus JR, De Luca G (2016). “Weighted-average least squares (WALS): A survey.” *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

See Also

subbotin.

`familyPrior` *Family Objects for Prior Distributions in WALS*

Description

`"familyPrior"` objects provide a convenient way to specify the prior distribution used for the Bayesian posterior mean estimation of the WALS estimators in `wals`, `walsGLM` and `walsNB`

Usage

```
familyPrior(object, ...)

weibull(q = 0.887630085544086, b = log(2))

subbotin(q = 0.799512530172489, b = 0.937673273794677)

laplace(b = log(2))

## S3 method for class 'familyPrior'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'wals'
familyPrior(object, ...)
```

Arguments

`object, x`	Object of of class `"familyPrior"` or `"wals"`. The function `familyPrior()` accesses the `"familyPrior"` objects that are stored in objects of class `"wals"`.
`...`	Further arguments passed to methods.
`q`	q in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
`b`	c in Magnus and De Luca (2016). Parameter of reflected generalized gamma distribution. See below for details.
`digits`	The number of significant digits to display.

Details

`familyPrior()` is a generic function that extracts the family used in "`wals`" objects.

The density function of the reflected generalized gamma distribution is

$$\pi(x) = \frac{qc^{(1-\alpha)/q}}{2\Gamma((1-\alpha)/q)} |x|^{-\alpha} \exp(-c|x|^q).$$

The double (reflected) Weibull, Subbotin and Laplace distributions are all special cases of the reflected generalized gamma distribution. The Laplace distribution is also a special case of the double Weibull and of the Subbotin distribution.

The double (reflected) Weibull density sets $q = 1 - \alpha$, the Subbotin density sets $\alpha = 0$ and the Laplace density sets $\alpha = 0$ and $q = 1$.

The default values for the parameters `q` and `b` are minimax regret solutions for the corresponding priors. The double (reflected) Weibull and Subbotin prior are both neutral and robust. In contrast, the Laplace prior is only neutral but not robust. See section 9 "Enter Bayes: Neutrality and Robustness" of Magnus and De Luca (2016) for details and Table 1 for the optimal parameter values.

Value

An object of class "`familyPrior`" to be used in `wals`, `walsGLM` and `walsNB`. This is a list with the elements

`q`	Parameter q.
`alpha`	Parameter α (of the reflected generalized gamma distribution).
`b`	Parameter c.
`delta`	Parameter $\delta = (1-\alpha)/q$.
`printPars`	vector. Contains the parameters that are shown in printing functions, e.g. `print.familyPrior()`.
`prior`	String with the name of the prior distribution.

`laplace()` returns an object of the specialized class "`familyPrior_laplace`" that inherits from "`familyPrior`". This allows separate processing of the Laplace prior in the estimation functions as closed-form formulas exists for its posterior mean and variance. The list elements are the same as for objects of class "`familyPrior`".

References

Magnus JR, De Luca G (2016). “Weighted-average least squares (WALS): A survey.” *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

See Also

wals, walsGLM, walsNB, computePosterior, ddweibull, dsubbotin, dlaplace.

Examples

```
## Use in wals():
fit <- wals(gdpgrowth ~ lgdp60 + equipinv + school60 + life60 + popgrowth |
              law + tropics + avelf + confucian, data = GrowthMPP,
              prior = weibull(q = 0.8, b = log(1.8)))
summary(fit)
```

`familyWALS` *Extended Family Objects for Models*

Description

Objects of class "familyWALS" inherit from "family" and extend those with (transformation) functions required for walsGLM and walsNB.

Usage

```
familyWALS(object, ...)

poissonWALS(link = "log")

binomialWALS(link = "logit")

negbinFixedWALS(scale, link)

negbinWALS(scale, link)

## S3 method for class 'walsGLM'
familyWALS(object, ...)
```

Arguments

object	The function familyWALS() extracts the family objects stored in "walsGLM" objects.
...	Further arguments passed to methods. The negbinWALS() family currently only accepts "log", while negbinFixedWALS() supports both "log" and "canonical".
link	Specifies the model link function. Typically a character string or an object of class "link-glm" generated by make.link. See family for more details. Currently, only a limited number of links are supported. See below for more details.
scale	dispersion parameter of NB2 to be used, always larger than 0.

Details

familyWALS() is a generic function that extracts the family used in "walsGLM" objects.

negbinFixedWALS() creates the "familyWALS" object for negative binomial distribution type 2 (NB2) with fixed dispersion parameter. It extends negativeBinomial.

negbinWALS() creates objects of the specialized class "familyNBWALS" which inherits from "familyWALS" and "family". It constructs the "familyNBWALS" object for the negative binomial distribution type 2 (NB2) with variable dispersion parameter by extending negativeBinomial and negbinFixedWALS with functions required in walsNB. negbinWALS **should only be used in** walsNBfit **and not in** walsGLM **because the NB2 with variable dispersion parameter is not a GLM!**

Supported links:

Currently, binomialWALS() and poissonWALS() only support their canonical links, i.e. "logit" and "log", respectively. negbinFixedWALS() supports both, the "canonical" link and the "log" link, however, the first is not recommended due to numerical difficulties in the fitting process. In contrast, negbinWALS() only supports the "log" link.

Value

An object of class "familyWALS" to be used in walsGLM that inherits from "family". This is a list that contains elements returned from the corresponding family function that it extends. Additionally, the following elements are available:

theta.eta	function. Derivative of the canonical parameter θ with respect to the linear link η, i.e. $d\theta/d\eta$.
psi	function. ψ defined on p. 3 of (De Luca et al. 2018).
initializeY	function. Preprocesses the response, e.g. in binomialWALS() it transforms factors to numeric 0s and 1s.
transformY	function. Transforms the response to $\bar{y}$. See eq. (5) in (De Luca et al. 2018) for GLMs and (Huynh 2023a) for negbinWALS() used in walsNB.
transformX	function. Transforms the regressors to $\bar{X}_1$ and $\bar{X}_2$, respectively. See eq. (5) in (De Luca et al. 2018) for GLMs and (Huynh 2023a) for negbinWALS() used in walsNB.
density	function. The probability density/mass function of the family.

poissonWALS() and negbinFixedWALS() return objects of class "familyWALScount" that inherit from "familyWALS" and "family". These are lists that contain the same elements as "familyWALS" objects described above.

negbinWALS() creates an object of class "familyNBWALS" (only used internally in walsNB) that inherits from "familyWALScount", "familyWALS" and "family". This is a list that contains all elements returned from negbinFixed and the elements described above for objects of class "familyWALS". Additionally contains the following elements with functions required in walsNB that are described in (Huynh 2023a):

q	function. Computes $\bar{q}$.
g	function. Computes $\bar{g}$.
transformY0	function. Computes $\bar{y}_0$.

t	function. Computes $\bar{t}$.
epsilon	function. Computes $\bar{\epsilon}$.
epsiloninv	function. Computes $\bar{\epsilon}^{-1}$.
kappaSum	function. Computes $\bar{\kappa}^\top \mathbf{1}$.
computeAlpha	function. Computes the log-dispersion parameter $\log(\rho)$ given (model-averaged) estimates of the regression coefficients of the transformed regressors γ_1 and γ_2.

References

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Huynh K (2023a). "Weighted-Average Least Squares for Negative Binomial Regression." University of Basel. Mimeo.

See Also

family, walsGLM.

Examples

```
## Use in walsGLM():
data("NMES1988", package = "AER")
NMES1988 <- na.omit(NMES1988)
fitPoisson <- walsGLM(emergency ~ health + chronic + age + gender |
                       I((age^2)/10) + married + region, family = poissonWALS(),
                      data = NMES1988, prior = laplace())
summary(fitPoisson)

## Plot derivatives of binomialWALS() with default 'logit' link:
bi <- binomialWALS()
plot(bi$mu.eta, from = -10, to = 10)
plot(bi$theta.eta, from = -10, to = 10) # constant. logit is canonical link
```

---

fitNB2 — *Internal function: Fits a NB2 regression via maximum likelihood with log-link for mean and dispersion parameter.*

Description

Internal fitting function for NB2 regression models. Used for fitting the starting values of the one-step ML estimators in `walsNB`. Only works with log-link so far, no other links tested.

Usage

```
fitNB2(X, Y, family, control = controlNB())
```

Arguments

- `X` Design matrix.
- `Y` Count response vector.
- `family` Object of class "`familyNBWALS`" generated by `negbinWALS`.
- `control` List of parameters for controlling the optimization process. Use `controlNB` to generate the list.

Details

The available parameters for controlling the optimization are documented in `controlNB`.

Value

A list with the following elements

- `coefficients` fitted coefficients from NB2 regression
- `theta` fitted dispersion parameter from NB2 regression
- `convergence` 0 indicates successful completion. All error codes except for `99` are generated by `optim`. Possible error codes are
  - `1` indicates that the iteration limit `maxit` had been reached.
  - `10` degeneracy of the Nelder-Mead simplex.
  - `51` warning from "L-BFGS-B" method; see component `message` for further details.
  - `52` error from "L-BFGS-B" method; see component `message` for further details.
  - `99` (only possible if `controlNB(initMASS = TRUE)`) indicates convergence issues in `glm.nb`.
- `ll` log-likelihood of fitted NB2 regression model
- `message` If `controlNB(initMASS = FALSE)`, character string giving any additional information returned by the optimizer, else `NULL`.
- `start` If `controlNB(initMASS = FALSE)`, contains a vector with the starting values used for `optim`.

See Also

controlNB, negbinWALS, glm.nb, optim.

gammaToBeta *Internal function: Transform gammas back to betas*

Description

Transforms posterior means $\hat{\gamma}_2$ and variances corresponding to transformed auxiliary regressors Z_2 back to regression coefficients $\hat{\beta}$ of original regressors X_1 and X_2.

Usage

```
gammaToBeta(
  posterior,
  y,
  Z1,
  Z2,
  Delta1,
  D2,
  sigma,
  Z1inv,
  method = "original",
  svdZ1
)
```

Arguments

posterior	Object returned from `computePosterior`.
y	Response y.
Z1	Transformed focus regressors Z_1.
Z2	Transformed auxiliary regressors Z_1.
Delta1	Δ_1 or $\bar{\Delta}_1$.
D2	From `semiorthogonalize`, if `postmult` = `FALSE` was used, then D2 = $\Delta_2 T\Lambda^{-1/2}$, where T are the eigenvectors of Ξ and Λ the diagonal matrix containing the corresponding eigenvalues. If `postmult` = `TRUE` was used, then D2 = $\Delta_2 T\Lambda^{-1/2}T^\top = \Delta_2\Xi^{-1/2}$.
sigma	Prespecified or estimated standard deviation of the error term.
Z1inv	$(Z_1^\top Z_1)^{-1}$.
method	Character. $\hat{\gamma}_1$ is obtained from a linear regression of Z_1 on pseudo-responses $y - Z_2\hat{\gamma}_2$. If `method` = `original`, then we use `lm.fit` to perform the linear regression, if `method` = `"svd"`, then reuse the SVD of Z_1 in `svdZ1` to perform the regression.
svdZ1	Optional, only needed if `method` = `"svd"`. SVD of Z_1 computed using `svd`.

Details

The same transformations also work for GLMs, where we replace X_1, X_2, Z_1 and Z_2 with $\bar{X}_1$, $\bar{X}_2$, $\bar{Z}_1$ and $\bar{Z}_2$, respectively. Generally, we need to replace all variables with their corresponding "bar" version. Further, for GLMs `sigma` is always 1.

See Magnus and De Luca (2016), De Luca et al. (2018) and Huynh (2023b) for the definitions of the variables.

References

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

`GrowthMP` *Determinants of Economic Growth*

Description

Growth regression data used in Masanjala and Papageorgiou (2008).

Usage

```
GrowthMP
```

Format

A data frame with 37 observations on 25 variables:

gdpgrowth Average growth rate of GDP per capita from 1960 - 1992 at purchasing power parity.

lgdp60 Logarithm of GDP per capita in 1960.

yrsopen Fraction of years economy open from 1960 - 1990.

mining Fraction of GDP in mining.

primexp70 Share of exports of primary products in GDP in 1970.

invest Ratio of real domestic investment (public and private) to real GDP.

rerd Real exchange rate distortion.

school60 Average years of primary schooling for population over 25 years of age in 1960.

life60 Life expectancy at age 0 in 1960.

popgrowth Average growth rate of population from 1960 - 1990.

war factor. `"yes"` if country participates in at least one external war from 1960 to 1985. `"no"` else.

revcoup Average number of revolutions and coups per year from 1960 - 1990.

rights Index of political rights ranging from 1 (most restrictive) to 7 (most freedom)

civil Index of civil liberties ranging from 1 (most restrictive) to 7 (most freedom)

out Index of outward orientation.

capitalism Degree of capitalism.

colony factor. Shows if the country used to be `"british"` or `"french"` colony. If neither of them applies, then `"none"`.

english Fraction of English speakers.

foreign Fraction speaking foreign language.

frac Probability that two random people are from different ethnolinguistic groups.

protestant Fraction of population Protestant.

catholic Fraction of population Catholic.

muslim Fraction of population Muslim.

area Size of country in millions of square kilometers.

abslat Distance from the equator.

Details

The dataset of Masanjala and Papageorgiou (2008) is a subset of sub-Sahara African countries from the data used in Sala-I-Martin (1997). See Table A2. in Masanjala and Papageorgiou (2008) for the original sources of the variables. This dataset is also used for replication purposes in Amini and Parmeter (2012).

To replicate the WALS estimates in Amini and Parmeter (2012), use all variables except for a constant as auxiliary regressors and divide all regressors by their in-sample maximum before running `wals(..., prescale = FALSE)` (**NOTE: It is not recommended to use** `prescale = FALSE` **as this runs an old version of the WALS estimator,** `prescale = FALSE` **should only be used for replication purposes**). The resulting coefficients and standard errors have to be divided by the maximum of the regressors again to get the values presented in Table I of the paper.

Source

Journal of Applied Econometrics Data Archive. The data was taken from the archive entry of Amini and Parmeter (2012) for replication purposes but they can also be found in the archive entry of Masanjala and Papageorgiou (2008).

https://journaldata.zbw.eu/dataset/comparison-of-model-averaging-techniques-assessing-growth-deter

References

Amini SM, Parmeter CF (2012). "Comparison of model averaging techniques: assessing growth determinants." *Journal of Applied Econometrics*, **27**(5), 870-876. doi:10.1002/jae.2288.

Masanjala WH, Papageorgiou C (2008). "Rough and lonely road to prosperity: a reexamination of the sources of growth in Africa using Bayesian model averaging." *Journal of Applied Econometrics*, **23**(5), 671-682. doi:10.1002/jae.1020.

Sala-I-Martin X (1997). "I Just Ran Two Million Regressions." *The American Economic Review*, **87**(2), 178–183.

Examples

```
## Replicate second panel of Table I in Amini & Parmeter (2012)
## NOTE: Authors manually scale data, then rescale the resulting coefs and se.
X <- model.matrix(gdpgrowth ~ ., data = GrowthMP)
scaleVector <- apply(X, MARGIN = 2, max)
Xscaled <- apply(X, MARGIN = 2, function(x) x/max(x))
Xscaled <- Xscaled[,-1]
datscaled <- as.data.frame(cbind(gdpgrowth = GrowthMP$gdpgrowth, Xscaled))

fitMP <- wals(gdpgrowth ~ 1 | ., data = datscaled, prescale = FALSE,
              prior = laplace(), eigenSVD = FALSE)
tableMP <- cbind("coef" = coef(fitMP)/scaleVector,
                 "se" = sqrt(diag(vcov(fitMP)))/scaleVector)
printVars <- c("(Intercept)", "lgdp60", "yrsopen", "mining", "primexp70",
               "invest")
print(round(tableMP[printVars,], 4))
```

GrowthMPP *Determinants of Economic Growth*

Description

Growth regression data used in Magnus et al. (2010).

Usage

```
GrowthMPP
```

Format

A data frame with 72 observations on 11 variables:

country factor. Name of the country.

gdpgrowth Average growth rate of GDP per capita from 1960 - 1996 at purchasing power parity.

lgdp60 Logarithm of GDP per capita in 1960.

equipinv Average real equipment investment share of GDP from 1960 - 1985 comprising investments in electrical and nonelectrical machinery (in relative prices constant across countries).

school60 Enrollment rate for primary education in 1960.

life60 Life expectancy at age 0 in 1960.

popgrowth Average growth rate of population from 1960 - 1996.

law Index for the overall maintenance of the rule of law ('law and order tradition').

tropics Proportion of country's land area within geographical tropics.

avelf Average of five different indices of ethnolinguistic fragmentation which is measured as the probability of two random people in a country not sharing the same language.

confucian Fraction of Confucian population in 1970 and 1980.

Details

The dataset is used in Magnus et al. (2010) to illustrate the WALS model averaging approach and combines the data used in Sala-I-Martin et al. (2004) and Sala-I-Martin (1997). See the references for more detailed descriptions and original sources of the variables.

Source

WALS package for MATLAB (and Stata) provided on Jan Magnus' personal website. https://www.janmagnus.nl/items/WALS.pdf.

References

Magnus JR, Powell O, Prüfer P (2010). "A comparison of two model averaging techniques with an application to growth empirics." *Journal of Econometrics*, **154**(2), 139-153. doi:10.1016/j.jeconom.2009.07.004.

Sala-I-Martin X (1997). "I Just Ran Two Million Regressions." *The American Economic Review*, **87**(2), 178–183.

Sala-I-Martin X, Doppelhofer G, Miller RI (2004). "Determinants of Long-Term Growth: A Bayesian Averaging of Classical Estimates (BACE) Approach." *American Economic Review*, **94**(4), 813-835. doi:10.1257/0002828042002570.

Examples

```
## Replicate Table 2 in Magnus et al. (2010)
# NOTE: prescale = FALSE, still used old version of WALS in Magnus et al. (2010).
# Not recommended anymore!
fitMPP <- wals(gdpgrowth ~ lgdp60 + equipinv + school60 + life60 + popgrowth |
                 law + tropics + avelf + confucian, data = GrowthMPP,
               prior = laplace(), prescale = FALSE)
tableMPP <- cbind("coef" = coef(fitMPP), "se" = sqrt(diag(vcov(fitMPP))))
print(round(tableMPP, 4))
```

---

negativeBinomial *Negative binomial family*

Description

Reconstruct family object for negative binomial type 2 (NB2) with fixed scale parameter theta. Analogous to `negative.binomial` in `MASS` (Venables and Ripley 2002) but `MASS` uses non-canonical link.

Usage

```
negativeBinomial(theta, link = "log")
```

Arguments

`theta`	dispersion parameter of NB2, always larger than 0.
`link`	specifies link function, currently only "log" and "canonical" are supported.

References

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*, Statistics and Computing, 4th edition. Springer-Verlag, New York. doi:10.1007/9780387217062, `https://www.stats.ox.ac.uk/pub/MASS4/`.

See Also

family, familyWALS, negbinWALS, negbinFixedWALS.

`predict.wals` *Methods for wals and walsMatrix Objects*

Description

Methods for extracting information from fitted model-averaging objects of classes `"wals"` and `"walsMatrix"`. `"walsMatrix"` objects inherit from `"wals"`, so the methods for `"wals"` also work for objects of class `"walsMatrix"`.

Usage

```
## S3 method for class 'wals'
predict(object, newdata, na.action = na.pass, ...)

## S3 method for class 'walsMatrix'
predict(object, newX1, newX2, ...)

## S3 method for class 'wals'
fitted(object, ...)

## S3 method for class 'wals'
residuals(object, ...)

## S3 method for class 'wals'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'wals'
summary(object, ...)

## S3 method for class 'summary.wals'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

```
## S3 method for class 'wals'
coef(object, type = c("all", "focus", "aux"), transformed = FALSE, ...)

## S3 method for class 'wals'
vcov(object, type = c("all", "focus", "aux"), transformed = FALSE, ...)

## S3 method for class 'wals'
nobs(object, ...)

## S3 method for class 'wals'
terms(x, type = c("focus", "aux"), ...)

## S3 method for class 'wals'
model.matrix(object, type = c("focus", "aux"), ...)
```

Arguments

`object, x`	An object of class `"wals"`, `"walsMatrix"` or `"summary.wals"`.
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
`na.action`	Function determining what should be done with missing values in `newdata`. The default is to predict `NA`.
`...`	Further arguments passed to methods.
`newX1`	Focus regressors matrix to be used for the prediction.
`newX2`	Auxiliary regressors matrix to be used for the prediction.
`digits`	The number of significant digits to display.
`type`	Character specifying the part of the model that should be returned. For details see below.
`transformed`	Logical specifying whether the coefficients/covariance matrix of original regressors (`FALSE`, default) or the transformed regressors (`TRUE`) should be returned.

Details

A set of standard extractor functions for fitted model objects is available for objects of class `"wals"` and `"walsMatrix"`, including methods to the generic functions `print` and `summary` which print the model-averaged estimation of the coefficients along with some further information. As usual, the `summary` method returns an object of class `"summary.wals"` containing the relevant summary statistics which can then be printed using the associated `print` method. Inspired by De Luca and Magnus (2011), the summary statistics also show `Kappa` which is an indicator for the numerical stability of the method, i.e. it shows the square root of the condition number of the matrix $\Xi = \Delta_2 X_2^\top M_1 X_2 \Delta_2$. The summary further provides information on the prior used along with its parameters. The `summary()`, `print.summary()`, `print()` and `logLik()` methods are also inspired by the corresponding methods for objects of class `"lm"` in `stats` version 4.3.1 (2023-06-16) (R Core Team 2023), see e.g. `print.summary.lm`.

The `residuals` method computes raw residuals (observed - fitted).

For `coef` and `vcov`, the `type` argument, either `"all"`, `"focus"` or `"aux"`, specifies which part of the coefficient vector/covariance matrix of the estimates should be returned. Additionally, the `transformed` argument specifies whether to return the estimated coefficients/covariance matrix for the original regressors X or of the transformed regressors Z.

The extractors `terms` and `model.matrix` behave similarly to `coef`, but they only allow `type = "focus"` and `type = "aux"`. They extract the corresponding component of the model. This is similar to the implementation of these extractors in `countreg` version 0.2-1 (2023-06-13) (Zeileis and Kleiber 2023; Zeileis et al. 2008), see e.g. `terms.hurdle()`.

Value

`predict.wals()` and `predict.walsMatrix()` return a vector containing the predicted means.

`fitted.wals()` returns a vector containing the fitted means for the data used in fitting.

`residuals.wals()` returns the raw residuals of the fitted model, i.e. response - fitted mean.

`print.wals()` invisibly returns its input argument `x`, i.e. an object of object of class `"wals"`.

`summary.wals` returns an object of class `"summary.wals"` which contains the necessary fields for printing the summary in `print.summary.wals()`.

`print.summary.wals()` invisibly returns its input argument `x`, i.e. an object of object of class `"summary.wals"`.

`coef.wals()` returns a vector containing the fitted coefficients. If `type = "focus"`, only the coefficients of the focus regressors are returned and if `type = "aux"`, only the coefficients of auxiliary regressors are returned. Else if `type = "all"`, the coefficients of both focus and auxiliary regressors are returned. Additionally if `transformed = FALSE`, `coef.wals()` returns the estimated coefficients for the original regressors X (β coefficients) and else if `transformed = TRUE` the coefficients of the transformed regressors Z (γ coefficients).

`vcov.wals()` returns a matrix containing the estimated (co-)variances of the fitted regression coefficients. If `type = "focus"`, only the submatrix belonging to the focus regressors is returned and if `type = "aux"`, only the submatrix corresponding to the auxiliary regressors is returned. Else if `type = "all"`, the complete covariance matrix is returned. Additionally if `transformed = FALSE`, `vcov.wals()` returns the estimated covariance matrix for the original regressors X (β coefficients) and else if `transformed = TRUE` the covariance matrix of the transformed regressors Z (γ coefficients).

`nobs.wals()` returns the number of observations used for fitting the model.

`terms.wals()` returns the *terms* representation of the fitted model. It is of class `c("terms", "formula")`, see `terms` and `terms.object` for more details. If `type = "focus"`, then returns the terms for the focus regressors, else if `type = "aux"` returns the terms for the auxiliary regressors.

`model.matrix.wals()` either returns the design matrix of the focus regressors (`type = "focus"`) or of the auxiliary regressors (`type = "aux"`). See `model.matrix` for more details.

References

De Luca G, Magnus JR (2011). “Bayesian model averaging and weighted-average least squares: Equivariance, stability, and numerical issues.” *The Stata Journal*, **11**(4), 518–544. doi:10.1177/1536867X1201100402.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Zeileis A, Kleiber C (2023). *countreg: Count Data Regression*. R package version 0.2-1, https://r-forge.r-project.org/projects/countreg/.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of Statistical Software*, **27**(8), 1–25. doi:10.18637/jss.v027.i08.

See Also

wals

Examples

```
## Example for wals objects
fitGrowth <- wals(gdpgrowth ~ lgdp60 + equipinv + school60 + life60 + popgrowth |
                  law + tropics + avelf + confucian, data = GrowthMPP,
                  prior = laplace())
summary(fitGrowth)
fitted(fitGrowth)
vcov(fitGrowth, type = "aux")
familyPrior(fitGrowth)
nobs(fitGrowth)

## Example for walsMatrix objects
X1 <- model.matrix(fitGrowth, type = "focus")
X2 <- model.matrix(fitGrowth, type = "aux")
y <- GrowthMPP$gdpgrowth
fitGrowthMatrix <- wals(X1, X2, y, prior = laplace())
coef(fitGrowthMatrix)
```

---

predict.walsGLM — *Methods for walsGLM, walsGLMmatrix, walsNB and walsNBmatrix Objects*

Description

Methods for extracting information from fitted model-averaging objects of classes "walsGLM", "walsGLMmatrix", "walsNB" and "walsNBmatrix".

Usage

```
## S3 method for class 'walsGLM'
predict(
  object,
  newdata,
  type = c("response", "link", "variance", "prob", "density", "logDens"),
```

```
  at = NULL,
  na.action = na.pass,
  log = FALSE,
  ...
)

## S3 method for class 'walsGLMmatrix'
predict(
  object,
  newX1,
  newX2,
  newY = NULL,
  type = c("response", "link", "variance", "prob", "density", "logDens"),
  at = NULL,
  log = FALSE,
  ...
)

## S3 method for class 'walsGLM'
residuals(object, type = c("deviance", "pearson", "response"), ...)

## S3 method for class 'walsGLM'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'walsGLM'
summary(object, ...)

## S3 method for class 'summary.walsGLM'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'walsGLM'
logLik(object, ...)

## S3 method for class 'walsNB'
summary(object, ...)

## S3 method for class 'summary.walsNB'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`object, x`	An object of class "`walsGLM`", "`walsGLMmatrix`", "`walsNB`", "`walsNBmatrix`", "`summary.walsGLM`" or "`summary.walsNB`".
`newdata`	Optionally, a data frame in which to look for variables with which to predict. If omitted, the original observations are used.
`type`	Character specifying the type of prediction, residual or model part to be returned. For details see below.
`at`	Optional. Only available if a family of class "`familyWALScount`" was used for

	fitting. If `type = "prob"`, a numeric vector at which the probabilities are evaluated. By default `0:max(y)` is used where `y` is the original observed response.
`na.action`	Function determining what should be done with missing values in `newdata`. The default is to predict `NA`.
`log`	Logical. If `TRUE`, then returns the log-density. If `FALSE` (default), then returns density. Only relevant if `type = "density"`.
`...`	Further arguments passed to methods.
`newX1`	Focus regressors matrix to be used for the prediction.
`newX2`	Auxiliary regressors matrix to be used for the prediction.
`newY`	Response vector to be used in predictions. Only relevant when `type = "prob"`, `type = "density"` or `type = "logDens"`.
`digits`	The number of significant digits to display.

Details

As the `"-matrix"` classes `"walsGLMmatrix"` and `"walsNBmatrix"` inherit from the "non-matrix" classes, i.e. `"walsGLM"` and `"walsNB"`, respectively, the following text will treat them as equivalent because they inherit all methods but `predict` from their "non-matrix" versions. Thus, when `"walsGLM"` or `"walsNB"` are mentioned, we also refer to their `"-matrix"` versions, except when explicitly stated. Moreover, note that `"walsNB"` and `"walsNBmatrix"` inherit most methods from `"walsGLM"` and `"walsGLMmatrix"`.

A set of standard extractor functions for fitted model objects is available for objects of class `"walsGLM"` and `"walsNB"`, including methods to the generic functions `print` and `summary` which print the model-averaged estimation of the coefficients along with some further information.

The `summary` methods returns an object of class `"summary.walsGLM"` for objects of class `"walsGLM"` and an object of class `"summary.walsNB"` for objects of class `"walsNB"`. They contain the relevant summary statistics which can then be printed using the associated `print()` methods. Inspired by De Luca and Magnus (2011), the summary statistics also show `Kappa` which is an indicator for the numerical stability of the method, i.e. it shows the square root of the condition number of the matrix $\bar{\Xi} = \bar{\Delta}_2 \bar{X}_2^{\top} \bar{M}_1 \bar{X}_2 \bar{\Delta}_2$. The summary further shows the deviance and provides information on the prior and family used.

A `logLik` method is provided that returns the log-likelihood given the family used and the model-averaged estimates of the coefficients.

`"walsGLM"` inherits from `"wals"`, while `"walsNB"` inherits from both, `"walsGLM"` and `"wals"`. Thus, see `predict.wals` for more methods.

The `predict` and `residuals` methods, especially the different types of predictions/residuals controlled by `type`, are inspired by the corresponding methods in `countreg` version 0.2-1 (2023-06-13) (Zeileis and Kleiber 2023; Zeileis et al. 2008), see e.g. `predict.hurdle()` from `countreg`, and `stats` version 4.3.1 (2023-06-16) (R Core Team 2023), see e.g. `residuals.glm`. The `summary()`, `print.summary()`, `print()` and `logLik()` methods are also inspired by the corresponding methods for objects of class `"glm"` in `stats`, see e.g. `print.summary.glm`.

`coef` and `vcov` are inherited from `"wals"` (see `predict.wals` for more), except for objects of class `"walsNB"` (see `vcov.walsNB`). The `type` argument specifies which part of the coefficient vector/covariance matrix of the estimates should be returned. For `type = "all"`, they return the complete vector/matrix. For `type = "focus"` and `type = "aux"` they return only the part corresponding to the focus and auxiliary regressors, respectively. Additionally, the user can choose whether to return the estimated coefficients/covariance matrix for the original regressors X (β coefficients) or of the transformed regressors Z (γ coefficients).

The extractors `terms` and `model.matrix` are also inherited from `"wals"`. They only allow `type = "focus"` and `type = "aux"` and extract the corresponding component of the model.

Value

`predict.walsGLM()` and `predict.walsGLMmatrix()` return different types of predictions depending on the argument `type`:

- `type = "response"`: vector. Predicted mean
- `type = "link"`: vector. Predicted linear link
- `type = "variance"`: vector. Predicted variance
- `type = "prob"`: matrix. Only available if a family of class `"familyWALScount"` was used for fitting or for objects of class `"walsNB"` or `"walsNBmatrix"`. Returns the probability at counts specified by `at`.
- `type = "density"`: vector. Predicted density
- `type = "logDens"`: vector. For convenience, returns predicted log-density. Equivalent to setting `type = "density"` and `log = TRUE`.

If `type = "prob"`, `type = "density"` or `type = "logDens"`, then `newdata` must contain the response or `newY` must be specified depending on the class of the object.

`residuals.walsGLM()` returns different types of residuals depending on the argument `type`:

- `type = "deviance"`: deviance residuals
- `type = "pearson"`: Pearson residuals (raw residuals scaled by square root of variance function)
- `type = "response"`: raw residuals (observed - fitted)

`print.walsGLM()` invisibly returns its input argument `x`, i.e. an object of object of class `"walsGLM"`.

`summary.walsGLM()` returns an object of class `"summary.walsGLM"` which contains the necessary fields for printing the summary in `print.summary.walsGLM()`.

`print.summary.walsGLM()` invisibly returns its input argument `x`, i.e. an object of object of class `"summary.walsGLM"`.

`logLik.walsGLM()` returns the log-likelihood of the fitted model.

`summary.walsNB()` returns an object of class `"summary.walsNB"` which contains the necessary fields for printing the summary in `print.summary.walsNB()`.

`print.summary.walsNB()` invisibly returns its input argument `x`, i.e. an object of object of class `"summary.walsNB"`.

References

De Luca G, Magnus JR (2011). "Bayesian model averaging and weighted-average least squares: Equivariance, stability, and numerical issues." *The Stata Journal*, **11**(4), 518–544. doi:10.1177/1536867X1201100402.

R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Zeileis A, Kleiber C (2023). *countreg: Count Data Regression*. R package version 0.2-1, https://r-forge.r-project.org/projects/countreg/.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of Statistical Software*, **27**(8), 1–25. doi:10.18637/jss.v027.i08.

See Also

walsGLM, walsNB, predict.wals.

Examples

```
## Example for walsGLM objects
data("HMDA", package = "AER")
fitBinomial <- walsGLM(deny ~ pirat + hirat + lvrat + chist + mhist + phist |
                        selfemp + afam, family = binomialWALS(), data = HMDA,
                       prior = weibull())
summary(fitBinomial)
vcov(fitBinomial, type = "focus")
logLik(fitBinomial)
predict(fitBinomial, newdata = HMDA[1:10,], type = "response")
familyWALS(fitBinomial)

## Example for walsNB objects
data("NMES1988", package = "AER")

fWals <- (visits ~ chronic + age + I((age^2)/10) + insurance + medicaid |
           adl + gender + married + income + school + afam + employed)
fitNB <- walsNB(fWals, data = NMES1988, link = "log", prior = weibull(),
                method = "fullSVD")
summary(fitNB)
coef(fitNB, type = "aux")
residuals(fitNB, type = "pearson")
predict(fitNB, newdata = NMES1988[1:10,], type = "prob")
terms(fitNB, type = "aux")
```

`predictCounts` *Internal methods: Predict probability for counts*

Description

Predicts the probability of counts given a family object of class "familyWALScount". Only works for count data models.

Usage

```
predictCounts(x, ...)

## S3 method for class 'familyWALScount'
predictCounts(x, yUnique, rowNames, eta, ...)
```

Arguments

`x`	object of class "familyWALScount".
`...`	Further parameters passed to `density()` function in `family`.
`yUnique`	vector. The counts (larger or equal to zero) which to predict probabilities for.
`rowNames`	vector. The names of the observations.
`eta`	vector. The fitted linear link $\hat{\eta}$ of the model.

Details

"familyWALScount" objects are used in the fitting methods `walsNB`, `walsNBmatrix`, `walsGLM` or `walsGLMmatrix`. For the latter two, only the family `poissonWALS` is currently supported.

`predictCounts()` is not available for objects of any class except for "familyWALScount".

The `predictCounts.familyWALScount()` method is a modified version of the `predict.hurdle()` method from the `countreg` package version 0.2-1 (2023-06-13) (Zeileis and Kleiber 2023; Zeileis et al. 2008) using the argument `type = "prob"`.

Value

Returns a matrix of dimension `length(eta)` times `length{yUnique}` with the predicted probabilities of the counts given in `yUnique` for every observation in `eta`.

References

Zeileis A, Kleiber C (2023). *countreg: Count Data Regression*. R package version 0.2-1, https://r-forge.r-project.org/projects/countreg/.

Zeileis A, Kleiber C, Jackman S (2008). "Regression Models for Count Data in R." *Journal of Statistical Software*, **27**(8), 1–25. doi:10.18637/jss.v027.i08.

`semiorthogonalize` *Internal function: Semiorthogonal-type transformation of X2 to Z2*

Description

Uses the matrix Z2s (called $\bar{\Xi}$ in eq. (9) of De Luca et al. (2018)) to transform $\bar{X}_2$ to $\bar{Z}_2$, i.e. to perform $\bar{Z}_2 = \bar{X}_2 \bar{\Delta}_2 \bar{\Xi}^{-1/2}$. For WALS in the linear regression model, the variables do not have a "bar".

Usage

```
semiorthogonalize(Z2s, X2, Delta2, SVD = TRUE, postmult = FALSE)
```

Arguments

- `Z2s` Matrix for which we take negative square root in $X2 * Delta2 * Z2s^{1/2}$.
- `X2` Design matrix of auxiliary regressors to be transformed to Z2
- `Delta2` Scaling matrix such that diagonal of $\bar{\Delta}_2 \bar{X}_2^\top \bar{M}_1 \bar{X}_2 \Delta_2$ is one (ignored scaling by n because not needed in code). See De Luca et al. (2018)
- `SVD` If `TRUE`, uses `svd` to compute eigendecomposition of `Z2s`, otherwise uses `eigen`.
- `postmult` If `TRUE`, then it uses $Z2s^{-1/2} = T\Lambda^{-1/2}T^\top$, where T contains the eigenvectors of $Z2s$ in its columns and Λ the corresponding eigenvalues. If `FALSE` it uses $Z2s^{-1/2} = T\Lambda^{-1/2}$.

On the "semiorthogonal-type" transformation

For WALS GLM (and WALS in the linear regression model), the transformation is semiorthogonal (ignored scaling by n for clarity and because it is not needed in the code) in the sense that $\bar{M}_1 \bar{Z}_2$ is semiorthogonal since

$$\bar{Z}_2^\top \bar{M}_1 \bar{Z}_2 = (\bar{Z}_2^\top \bar{M}_1)(\bar{M}_1 \bar{Z}_2) = I_{k_2},$$

where $\bar{M}_1$ is an idempotent matrix.

For WALS in the NB2 regression model, $\bar{M}_1 \bar{Z}_2$ is not semiorthogonal anymore due to the rank-1 perturbation in $\bar{M}_1$ which causes $\bar{M}_1$ to not be idempotent anymore, see the section "Transformed model" in Huynh (2023a).

On the use of `postmult = TRUE`

The transformation of the auxiliary regressors Z_2 for linear WALS in eq. (12) of Magnus and De Luca (2016) differs from the transformation for WALS GLM (and WALS NB) in eq. (9) of De Luca et al. (2018):

In Magnus and De Luca (2016) the transformed auxiliary regressors are

$$Z_2 = X_2 \Delta_2 T \Lambda^{-1/2},$$

where T contains the eigenvectors of $\Xi = \Delta_2 X_2^\top M_1 X_2 \Delta_2$ in the columns and Λ the respective eigenvalues. This definition is used when `postmult = FALSE`.

In contrast, De Luca et al. (2018) defines

$$Z_2 = X_2 \Delta_2 T \Lambda^{-1/2} T^\top,$$

where we ignored scaling by n and the notation with "bar" for easier comparison. This definition is used when `postmult = TRUE` and is strongly preferred for `walsGLM` and `walsNB`.

See Huynh (2023b) for more details.

References

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Huynh K (2023a). "Weighted-Average Least Squares for Negative Binomial Regression." University of Basel. Mimeo.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

`snbinom` *Internal function: first derivatives of NB2 PMF*

Description

First derivatives of NB2 PMF used in `fitNB2`. Code is taken from the function `snbinom()` in the `countreg` package version 0.2-1 (2023-06-13) (Zeileis and Kleiber 2023).

Usage

```
snbinom(x, mu, size, parameter = c("mu", "size"), drop = TRUE)
```

Arguments

`x`	Vector of quantiles.
`mu`	Vector of means.
`size`	Vector of dispersion parameter. If a scalar is given, the value is recycled.
`parameter`	Specifies which parameter the derivative is taken for. `parameter = c("mu", "size")` returns a matrix with derivatives for both parameters.
`drop`	If `TRUE`, drops empty dimensions of return using `drop`. If `FALSE` does not apply `drop`.

Value

A vector or matrix containing the first derivatives.

References

Zeileis A, Kleiber C (2023). *countreg: Count Data Regression*. R package version 0.2-1, https://r-forge.r-project.org/projects/countreg/.

`svdLSplus` *Internal function: Uses SVD components to compute final estimate via Sherman-Morrison-Woodbury formula.*

Description

Solves the equation system in walsNB via Sherman-Morrison-Woodbury formula for the unrestricted estimator $\tilde{\gamma}_u$.

Usage

```
svdLSplus(U, V, singularVals, y, ell, geB)
```

Arguments

`U`	Left singular vectors of $\bar{Z}$ or $\bar{Z}_1$ from `svd`.
`V`	Right singular vectors of $\bar{Z}$ or $\bar{Z}_1$ from `svd`.
`singularVals`	Singular values of $\bar{Z}$ or $\bar{Z}_1$ from `svd`.
`y`	"Pseudo"-response, see details.
`ell`	Vector $\bar{\ell}$ from section "Simplification for computing $\tilde{\gamma}_u$" Huynh (2023b)
`geB`	Scalar $\bar{g}\bar{\epsilon}/(1+B)$. See section "Simplification for computing $\tilde{\gamma}_u$" Huynh (2023b) for definition of $\bar{g}$, $\bar{\epsilon}$ and B.

Details

The function can be reused for the computation of the fully restricted estimator $\tilde{\gamma}_{1r}$ and the model averaged estimator $\hat{\gamma}_1$.

For $\tilde{\gamma}_{1r}$ and $\hat{\gamma}_1$ use `U`, `V` and `singularVals` from SVD of $\bar{Z}_1$.

For $\hat{\gamma}_u$ and $\tilde{\gamma}_{1r}$ use same pseudo-response $\bar{y_0} - \bar{t}\bar{\epsilon}\bar{\Psi}^{-1/2}\bar{q}$ in argument `y`.

For $\hat{\gamma}_1$ use pseudo-response $\bar{y_0} - \bar{t}\bar{\epsilon}\bar{\Psi}^{-1/2}\bar{q} - (\bar{Z}_2 + \bar{g}\bar{\epsilon}\bar{\Psi}^{-1/2}\bar{q}\bar{q}^{\top} Z_2)\hat{\gamma}_2$.

See section "Note on function svdLSplus from WALS" in Huynh (2023b).

References

Huynh K (2023b). “WALS: Weighted-Average Least Squares Model Averaging in R.” University of Basel. Mimeo.

vcov.walsNB *Calculate Variance-Covariance Matrix for a "walsNB" object*

Description

This method always raises an error because the covariance matrix of the walsNB estimator has not been derived yet.

Usage

```
## S3 method for class 'walsNB'
vcov(object, ...)
```

Arguments

object	An object of class "walsNB".
...	For expansion in the future.

Value

No return value, only raises error because no covariance matrix estimator exists yet.

wals *Weighted-Average Least Squares for linear regression models*

Description

Performs model averaging for linear regression models using the Weighted-Average Least Squares method by Magnus et al. (2010). See also De Luca and Magnus (2011), Kumar and Magnus (2013) and Magnus and De Luca (2016).

Usage

```
wals(x, ...)

## S3 method for class 'formula'
wals(
  formula,
  data,
  subset = NULL,
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  prior = weibull(),
  model = TRUE,
```

```
  keepY = TRUE,
  keepX = FALSE,
  sigma = NULL,
  ...
)

## S3 method for class 'matrix'
wals(
  x,
  x2,
  y,
  subset = NULL,
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  prior = weibull(),
  keepY = TRUE,
  keepX = FALSE,
  sigma = NULL,
  ...
)

## Default S3 method:
wals(x, ...)
```

Arguments

`x`	Design matrix of focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.
`...`	Arguments for workhorse `walsFit`.
`formula`	an object of class `"Formula"` (or one that can be coerced to that class, e.g. `"formula"`): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment which the function is called from.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	**not implemented yet.**
`weights`	**not implemented yet.**
`offset`	**not implemented yet.**
`prior`	Object of class `"familyPrior"`. For example `weibull` or `laplace`.
`model`	if `TRUE` (default), then the model.frame is stored in the return.
`keepY`	if `TRUE` (default), then the response is stored in the return.
`keepX`	if `TRUE`, then the model matrices are stored in the return. the return.

sigma	if NULL (default), then the variance of the error term is estimated. See walsFit for more details.
x2	Design matrix of auxiliary regressors. Usually does not include a constant column and can also be generated using model.matrix.
y	Response as vector.

Details

R port of MATLAB code wals.m (version 2.0, revision 18 December 2013) by J.R. Magnus and G. De Luca, available from https://www.janmagnus.nl/items/WALS.pdf. Calculates WALS estimates when focus regressors (X1) are present in all submodels and model averaging takes place over the auxiliary regressors (X2).

Formulas should always contain two parts, i.e. they should be of the form "y ~ X11 + X12 | X21 + X22", where the variables before "|" are the focus regressors (includes a constant by default) and the ones after "|" are the auxiliary regressors.

WARNING: Interactions in formula do not work properly yet. It is recommended to manually create the interactions beforehand and then to insert them as 'linear terms' in the formula.

`wals.default()` raises an error if `x` is not an object of class `"matrix"` or a class that extends `"matrix"`. Otherwise it calls `wals.matrix()`. It is a modified version of `glmboost.default` from the `mboost` package version 2.9-8 (2023-09-06) (Hofner et al. 2014).

Value

`wals.formula()` returns an object of class `"wals"`. This is a list that contains all elements returned from walsFit and additionally

y	If keepY = TRUE, contains the response vector.
x	list. If keepX = TRUE, then it is a list with elements x1 and x2 containing the design matrices of the focus and auxiliary regressors, respectively.
weights	returns the argument weights.
offset	returns the argument offset.
cl	Call of the function.
formula	formula used.
terms	List containing the model terms of the focus and auxiliary regressors separately, as well as for the full model.
levels	List containing the levels of the focus and auxiliary regressors separately, as well as for the full model.
contrasts	List containing the contrasts of the design matrices of focus and auxiliary regressors.
model	If model = TRUE, contains the model frame.

See returns of walsFit for more details.

`wals.matrix()` returns an object of class `"walsMatrix"`, which inherits from `"wals"`. This is a list that contains all elements returned from walsFit and additionally the response `y`, the list `x` with model matrices `x1` and `x2`, the call `cl`, `offset` and `weights`.

`wals.default()` raises an error if `x` is not an object of class `"matrix"` or a class that extends `"matrix"`. Otherwise returns an object of class `"walsMatrix"`. See above for more details.

References

De Luca G, Magnus JR (2011). "Bayesian model averaging and weighted-average least squares: Equivariance, stability, and numerical issues." *The Stata Journal*, **11**(4), 518–544. doi:10.1177/1536867X1201100402.

Hofner B, Mayr A, Robinzonov N, Schmid M (2014). "Model-based Boosting in R: A Hands-on Tutorial Using the R Package mboost." *Computational Statistics*, **29**, 3–35.

Kumar K, Magnus JR (2013). "A characterization of Bayesian robustness for a normal location parameter." *Sankhya B*, **75**(2), 216–237. doi:10.1007/s13571013006009.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

Magnus JR, Powell O, Prüfer P (2010). "A comparison of two model averaging techniques with an application to growth empirics." *Journal of Econometrics*, **154**(2), 139-153. doi:10.1016/j.jeconom.2009.07.004.

Examples

```
## Replicate table on p. 534 of De Luca & Magnus (2011)
fitDM <- wals(gdpgrowth ~ lgdp60 + equipinv + school60 + life60 + popgrowth |
                  law + tropics + avelf + confucian, data = GrowthMPP,
                prior = laplace())
tableDM <- cbind("coef" = coef(fitDM), "se" = sqrt(diag(vcov(fitDM))))
print(round(tableDM, 7))


## Replicate first panel of Table I in Amini & Parmeter (2012)
data("datafls", package = "BMS")

# NOTE: Authors manually scale data, then rescale the resulting coefs and se.
X <- model.matrix(y ~ ., data = datafls)
Xscaled <- apply(X, MARGIN = 2, function(x) x/max(x))
Xscaled <- Xscaled[,-1]
scaleVector <- apply(X, MARGIN = 2, function(x) max(x))
flsScaled <- as.data.frame(cbind(y = datafls$y, Xscaled))

# NOTE: prescale = FALSE, still used old version of WALS in Magnus et al. (2010).
# Not recommended anymore!
fitFLS <- wals(y ~ 1 | ., data = flsScaled, prescale = FALSE, eigenSVD = FALSE,
               prior = laplace())
tableFLS <- cbind('coef' = coef(fitFLS)/scaleVector,
                  'se' = sqrt(diag(vcov(fitFLS)))/scaleVector)
printVars <- c("(Intercept)", "GDP60", "Confucian", "LifeExp", "EquipInv",
               "SubSahara", "Muslim", "RuleofLaw")
print(round(tableFLS[printVars,], 4))


## Replicate third panel of Table I in Amini & Parmeter (2012)
data("SDM", package = "BayesVarSel")
```

```
# rescale response
SDM$y <- SDM$y / 100

# NOTE: Authors manually scale data, then rescale the resulting coefs and se.
X <- model.matrix(y ~ ., data = SDM)
Xscaled <- apply(X, MARGIN = 2, function(x) x/max(x))
Xscaled <- Xscaled[,-1]
scaleVector <- apply(X, MARGIN = 2, function(x) max(x))
SDMscaled <- as.data.frame(cbind(y = SDM$y, Xscaled))

# NOTE: prescale = FALSE, still used old version of WALS in Magnus et al. (2010).
# Not recommended anymore!
fitDW <- wals(y ~ 1 | ., data = SDMscaled, prescale = FALSE, eigenSVD = FALSE,
              prior = laplace())
tableDW <- cbind(coef(fitDW)/scaleVector, sqrt(diag(vcov(fitDW)))/scaleVector)
printVars <- c("(Intercept)", "EAST", "P60", "IPRICE1", "GDPCH60L", "TROPICAR")
print(round(tableDW[printVars,], 5))

## Example for wals.matrix()
X <- model.matrix(mpg ~ disp + hp + wt + vs + am + carb, data = mtcars)
X1 <- X[,c("(Intercept)", "disp", "hp", "wt")] # focus
X2 <- X[,c("vs", "am", "carb")] # auxiliary
y <- mtcars$mpg

wals(X1, X2, y, prior = weibull())
```

walsFit — *Fitter function for Weighted Average Least Squares estimation*

Description

Workhorse function behind `wals` and `walsGLM`.

Usage

```
walsFit(
  X1,
  X2,
  y,
  sigma = NULL,
  prior = weibull(),
  method = "original",
  svdTol = .Machine$double.eps,
  svdRtol = 1e-06,
  keepUn = FALSE,
  eigenSVD = TRUE,
  prescale = TRUE,
```

```
  postmult = FALSE,
  ...
)
```

Arguments

X1
: Design matrix for focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.

X2
: Design matrix for auxiliary regressors. Usually does not include a constant column and can also be generated using `model.matrix`.

y
: Response as vector.

sigma
: if NULL (default), then the variance of the error term is estimated, see p.136 of Magnus and De Luca (2016). If sigma is specified, then the unrestricted estimator is divided by sigma before performing the Bayesian posterior mean estimation.

prior
: Object of class `"familyPrior"`. For example `weibull` or `laplace`.

method
: Specifies method used. Available methods are `"original"` (default) or `"svd"`.

svdTol
: Tolerance for rank of matrix $\bar{Z}_1$ Only used if `method = "svd"`. Checks if smallest eigenvalue in SVD of $\bar{Z}_1$ and $\bar{Z}$ is larger than `svdTol`, otherwise reports a rank deficiency.

svdRtol
: Relative tolerance for rank of matrix $\bar{Z}_1$. Only used if `method = "svd"`. Checks if ratio of largest to smallest eigenvalue in SVD of $\bar{Z}_1$ is larger than `svdRtol`, otherwise reports a rank deficiency.

keepUn
: If `TRUE`, keeps the estimators of the unrestricted model, i.e. $\tilde{\gamma}_u$.

eigenSVD
: If `TRUE`, then `semiorthogonalize` uses `svd` to compute the eigendecomposition of $\bar{\Xi}$ instead of `eigen`. In this case, the tolerances of `svdTol` and `svdRtol` are used to determine whether $\bar{\Xi}$ is of full rank (need it for $\bar{\Xi}^{-1/2}$).

prescale
: If `TRUE` (default), prescales the regressors X1 and X2 with Δ_1 and Δ_2, respectively, to improve numerical stability and make the coefficients of the auxiliary regressors scale equivariant. See De Luca and Magnus (2011) for more details. **WARNING: It is not recommended to set `prescale = FALSE`.** The option `prescale = FALSE` only exists for historical reasons.

postmult
: If `TRUE`, then it computes

$$Z_2 = X_2 \Delta_2 T \Lambda^{-1/2} T^{\top},$$

where T contains the eigenvectors and Λ the eigenvalues from the eigenvalue decomposition

$$\Xi = \Delta_2 X_2^{\top} M_1 X_2 \Delta_2 = T \Lambda T^{\top},$$

instead of

$$Z_2 = X_2 \Delta_2 T \Lambda^{-1/2}.$$

See Huynh (2023b) for more details. The latter is used in the original MATLAB code for WALS in the linear regression model (Magnus et al. 2010; De Luca and Magnus 2011; Kumar and Magnus 2013; Magnus and De Luca 2016), see eq. (12) of Magnus and De Luca (2016). The first form is required in eq. (9) of De Luca et al. (2018). It is not recommended to set `postmult = FALSE` when using `walsGLM` and `walsNB`.

...	Arguments for internal function computePosterior.

Value

A list containing

coef	Model averaged estimates of all coefficients.
beta1	Model averaged estimates of the coefficients of the focus regressors.
beta2	Model averaged estimates of the coefficients of the auxiliary regressors.
gamma1	Model averaged estimates of the coefficients of the transformed focus regressors.
gamma2	Model averaged estimates of the coefficients of the transformed auxiliary regressors.
vcovBeta	Estimated covariance matrix of the regression coefficients.
vcovGamma	Estimated covariance matrix of the coefficients of the transformed regressors.
sigma	Estimated or prespecified standard deviation of the error term.
prior	familyPrior. The prior specified in the arguments.
method	Stores method used from the arguments.
betaUn1	If keepUn = TRUE, contains the unrestricted estimators of the coefficients of the focus regressors.
betaUn2	If keepUn = TRUE, contains the unrestricted estimators of the coefficients of the auxiliary regressors.
gammaUn1	If keepUn = TRUE, contains the unrestricted estimators of the coefficients of the transformed focus regressors.
gammaUn2	If keepUn = TRUE, contains the unrestricted estimators of the coefficients of the transformed auxiliary regressors.
fitted.values	Estimated conditional means of the data.
residuals	Residuals, i.e. response - fitted mean.
X1names	Names of the focus regressors.
X2names	Names of the auxiliary regressors.
k1	Number of focus regressors.
k2	Number of auxiliary regressors.
n	Number of observations.
condition	Condition number of the matrix $\Xi = \Delta_2 X_2^\top M_1 X_2 \Delta_2$.

References

De Luca G, Magnus JR (2011). "Bayesian model averaging and weighted-average least squares: Equivariance, stability, and numerical issues." *The Stata Journal*, **11**(4), 518–544. doi:10.1177/1536867X1201100402.

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

Kumar K, Magnus JR (2013). "A characterization of Bayesian robustness for a normal location parameter." *Sankhya B*, **75**(2), 216–237. doi:10.1007/s1357101300609.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

Magnus JR, Powell O, Prüfer P (2010). "A comparison of two model averaging techniques with an application to growth empirics." *Journal of Econometrics*, **154**(2), 139-153. doi:10.1016/j.jeconom.2009.07.004.

See Also

wals, walsGLM.

Examples

```
X <- model.matrix(gdpgrowth ~ lgdp60 + equipinv + school60 + life60 + popgrowth
                  + law + tropics + avelf + confucian, data = GrowthMPP)
X1 <- X[, c("(Intercept)", "lgdp60", "equipinv", "school60", "life60", "popgrowth")]
X2 <- X[, c("law", "tropics", "avelf", "confucian")]
y <- GrowthMPP$gdpgrowth

walsFit(X1, X2, y, prior = weibull(), method = "svd")
```

---

walsGLM — *Weighted Average Least Squares for Generalized Linear Models*

Description

Performs model averaging of generalized linear models (GLMs) using the Weighted-Average Least Squares method described in De Luca et al. (2018).

Usage

```
walsGLM(x, ...)

## S3 method for class 'formula'
walsGLM(
  formula,
  family,
  data,
  subset = NULL,
  na.action = NULL,
  weights = NULL,
```

```
  offset = NULL,
  prior = weibull(),
  controlInitGLM = controlGLM(),
  model = TRUE,
  keepY = TRUE,
  keepX = FALSE,
  iterate = FALSE,
  tol = 1e-06,
  maxIt = 50,
  nIt = NULL,
  verbose = FALSE,
  ...
)

## S3 method for class 'matrix'
walsGLM(
  x,
  x2,
  y,
  family,
  subset = NULL,
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  prior = weibull(),
  controlInitGLM = controlGLM(),
  keepY = TRUE,
  keepX = FALSE,
  iterate = FALSE,
  tol = 1e-06,
  maxIt = 50,
  nIt = NULL,
  verbose = FALSE,
  ...
)

## Default S3 method:
walsGLM(x, ...)
```

Arguments

x	Design matrix of focus regressors. Usually includes a constant (column full of 1s) and can be generated using model.matrix.
...	Arguments for workhorse walsGLMfit.
formula	an object of class "Formula" (or one that can be coerced to that class, e.g. "formula"): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	Object of class "familyWALS".

data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment which the function is called from.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
na.action	**not implemented yet.**
weights	**not implemented yet.**
offset	**not implemented yet.**
prior	Object of class "familyPrior". For example weibull or laplace.
controlInitGLM	Controls estimation of starting values for one-step ML, see controlGLM.
model	if TRUE (default), then the model.frame is stored in the return.
keepY	if TRUE (default), then the response is stored in the return.
keepX	if TRUE, then the model matrices are stored in the return. the return.
iterate	if TRUE then the WALS algorithm is iterated using the previous estimates as starting values.
tol	Only used if iterate = TRUE and nIt = NULL. If the Euclidean distance between the previous and current coefficient vector divided by the square root of the length of the vector falls below tol, then the algorithm stops. See walsGLMfitIterate for more details.
maxIt	Only used if iterate = TRUE and nIt = NULL. Aborts iterative fitting when number of iterations exceed maxIt.
nIt	Only used if iterate = TRUE. If this is specified, then tol is ignored and the algorithm iterates nIt times. This option should not be used unless the user has a specific reason to run the algorithm nIt times, e.g. for replication purposes.
verbose	If verbose = TRUE, then it prints the iteration process (only relevant if iterate = TRUE).
x2	Design matrix of auxiliary regressors. Usually does not include a constant column and can also be generated using model.matrix.
y	Response as vector.

Details

Computes WALS estimates when focus regressors (X1) are present in all submodels and model averaging takes place over the auxiliary regressors (X2).

Formulas should always contain two parts, i.e. they should be of the form "y ~ X11 + X12 | X21 + X22", where the variables before "|" are the focus regressors (includes a constant by default) and the ones after "|" are the auxiliary regressors.

WARNING: Interactions in formula do work work properly yet. It is recommended to manually create the interactions beforehand and then to insert them as 'linear terms' in the formula.

walsGLM.default() raises an error if x is not an object of class "matrix" or a class that extends "matrix". Otherwise it calls walsGLM.matrix(). It is a modified version of glmboost.default from the mboost package version 2.9-8 (2023-09-06) (Hofner et al. 2014).

Value

walsGLM.formula() returns an object of class "walsGLM" which inherits from "wals". This is a list that contains all elements returned from walsGLMfitIterate and additionally

cl	Call of the function.
formula	formula used.
terms	List containing the model terms of the focus and auxiliary regressors separately, as well as for the full model.
levels	List containing the levels of the focus and auxiliary regressors separately, as well as for the full model.
contrasts	List containing the contrasts of the design matrices of focus and auxiliary regressors.
model	If model = TRUE, contains the model frame.

See returns of walsGLMfit and walsGLMfitIterate for more details.

walsGLM.matrix() returns an object of class "walsGLMmatrix", which inherits from "walsGLM", "walsMatrix" and "wals". This is a list that contains all elements returned from walsGLMfitIterate and additionally the call in cl.

walsGLM.default() raises an error if x is not an object of class "matrix" or a class that extends "matrix". Otherwise returns an object of class "walsGLMmatrix". See above for more details.

References

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Hofner B, Mayr A, Robinzonov N, Schmid M (2014). "Model-based Boosting in R: A Hands-on Tutorial Using the R Package mboost." *Computational Statistics*, **29**, 3–35.

Examples

```
data("HMDA", package = "AER")
fitBinomial <- walsGLM(deny ~ pirat + hirat + lvrat + chist + mhist + phist |
                         selfemp + afam, data = HMDA, family = binomialWALS(),
                         prior = weibull())
summary(fitBinomial)

data("NMES1988", package = "AER")
fitPoisson <- walsGLM(emergency ~ health + chronic + age + gender |
                        I((age^2)/10) + married + region, data = NMES1988,
                        family = poissonWALS(), prior = laplace())
summary(fitPoisson)

## Example for walsGLM.matrix()
data("HMDA", package = "AER")
X <- model.matrix(deny ~ pirat + hirat + lvrat + chist + mhist + phist + selfemp + afam,
                  data = HMDA)
X1 <- X[,c("(Intercept)", "pirat", "hirat", "lvrat", "chist2", "chist3",
```

```
        "chist4", "chist5", "chist6", "mhist2", "mhist3", "mhist4", "phistyes")]
X2 <- X[,c("selfempyes", "afamyes")]
y <- HMDA$deny
fit <- walsGLM(X1, X2, y, family = binomialWALS(), prior = weibull())
summary(fit)
```

---

walsGLMfit   *Fitter function for Weighted Average Least Squares estimation of GLMs*

Description

Workhorse function behind `walsGLM` and used internally in `walsGLMfitIterate`.

Usage

```
walsGLMfit(
  X1,
  X2,
  y,
  betaStart1,
  betaStart2,
  family,
  prior = weibull(),
  postmult = TRUE,
  ...
)
```

Arguments

`X1`	Design matrix for focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.
`X2`	Design matrix for auxiliary regressors. Usually does not include a constant column and can also be generated using `model.matrix`.
`y`	Response as vector.
`betaStart1`	Starting values for coefficients of focus regressors X1.
`betaStart2`	Starting values for coefficients of auxiliary regressors X2.
`family`	Object of class "`familyWALS`".
`prior`	Object of class "`familyPrior`". For example `weibull` or `laplace`.
`postmult`	If `TRUE` (default), then it computes $$\bar{Z}_2 = \bar{X}_2 \bar{\Delta}_2 \bar{T} \bar{\Lambda}^{-1/2} \bar{T}^\top,$$ where $\bar{T}$ contains the eigenvectors and $\bar{\Lambda}$ the eigenvalues from the eigenvalue decomposition $$\bar{\Xi} = \bar{T} \bar{\Lambda} \bar{T}^\top,$$

instead of

$$\bar{Z}_2 = \bar{X}_2 \bar{\Delta}_2 \bar{T} \bar{\Lambda}^{-1/2}.$$

See Huynh (2023b) for more details. The latter is used in the original MATLAB code for WALS in the linear regression model, see eq. (12) of Magnus and De Luca (2016). The first form is required in eq. (9) of De Luca et al. (2018). **Thus, it is not recommended to set** `postmult = FALSE`**.**

`...` Further arguments passed to `walsFit`.

Details

Uses `walsFit` under the hood after transforming the regressors `X1` and `X2` and the response `y`. For more details, see (Huynh 2023b) and De Luca et al. (2018).

Value

A list containing all elements returned by `walsFit`, except for `residuals`, and additionally (some fields are replaced)

`condition`	Condition number of the matrix $\bar{\Xi} = \bar{\Delta}_2 \bar{X}_2^\top \bar{M}_1 \bar{X}_2 \bar{\Delta}_2$.
`family`	Object of class `"familyWALS"`. The family used.
`betaStart`	Starting values of the regression coefficients for the one-step ML estimators.
`fitted.link`	Linear link fitted to the data.
`fitted.values`	Estimated conditional mean for the data. Lives on the scale of the response.

References

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

See Also

walsGLM, walsGLMfitIterate, walsFit.

Examples

```
data("HMDA", package = "AER")
X <- model.matrix(deny ~ pirat + hirat + lvrat + chist + mhist + phist + selfemp + afam,
                  data = HMDA)
X1 <- X[,c("(Intercept)", "pirat", "hirat", "lvrat", "chist2", "chist3",
        "chist4", "chist5", "chist6", "mhist2", "mhist3", "mhist4", "phistyes")]
X2 <- X[,c("selfempyes", "afamyes")]
y <- HMDA$deny
```

```
# starting values from glm.fit()
betaStart <- glm.fit(X, y, family = binomialWALS())$coefficients
k1 <- ncol(X1)
k2 <- ncol(X2)

str(walsGLMfit(X1, X2, y,
               betaStart1 = betaStart[1:k1],
               betaStart2 = betaStart[(k1 + 1):(k1 + k2)],
               family = binomialWALS(), prior = weibull()))
```

---

walsGLMfitIterate *Iteratively fitting walsGLM, internal function for walsGLM.formula and walsGLM.matrix.*

Description

Wrapper around `walsGLMfit` that allows iteratively (re-)fitting `walsGLM` models.

Usage

```
walsGLMfitIterate(
  y,
  X1,
  X2,
  family,
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  prior = weibull(),
  controlInitGLM = controlGLM(),
  keepY = TRUE,
  keepX = FALSE,
  iterate = FALSE,
  tol = 1e-06,
  maxIt = 50,
  nIt = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`y`	Response as vector.
`X1`	Design matrix for focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.

`X2`	Design matrix for auxiliary regressors. Usually does not include a constant column and can also be generated using `model.matrix`.
`family`	Object of class "`familyWALS`".
`na.action`	Not implemented yet.
`weights`	Not implemented yet.
`offset`	Not implemented yet.
`prior`	Object of class "`familyPrior`". For example `weibull` or `laplace`.
`controlInitGLM`	Controls estimation of starting values for one-step ML, see `controlGLM`.
`keepY`	If `TRUE`, then output keeps response.
`keepX`	If `TRUE`, then output keeps the design matrices.
`iterate`	if `TRUE` then the WALS algorithm is iterated using the previous estimates as starting values.
`tol`	Only used if `iterate = TRUE` and `nIt = NULL`. If the Euclidean distance between the previous and current coefficient vector divided by the square root of the length of the vector falls below `tol`, then the algorithm stops. See below for more details.
`maxIt`	Only used if `iterate = TRUE` and `nIt = NULL`. Aborts iterative fitting when number of iterations exceed `maxIt`.
`nIt`	Only used if `iterate = TRUE`. If this is specified, then `tol` is ignored and the algorithm iterates `nIt` times.
`verbose`	If `verbose = TRUE`, then it prints the iteration process (only relevant if `iterate = TRUE`).
`...`	Arguments to be passed to the workhorse function `walsGLMfit`.

Details

The parameter `tol` is used to control the convergence of the iterative fitting algorithm. Let i be the current iteration step for the coefficient vector $\beta_i = (\beta_{i,1}, \ldots, \beta_{i,k})', k > 0$. If

$$\frac{||\beta_i - \beta_{i-1}||_2}{\sqrt{k}} = \sqrt{\frac{\sum_{j=1}^{k} (\beta_{i,j} - \beta_{i-1,j})^2}{k}} < \texttt{tol},$$

then the fitting process is assumed to have converged and stops.

Value

A list containing all elements returned from `walsGLMfit` and additionally the following elements:

`y`	If `keepY = TRUE`, contains the response vector.
`x`	list. If `keepX = TRUE`, then it is a list with elements `x1` and `x2` containing the design matrices of the focus and auxiliary regressors, respectively.
`initialFit`	List containing information (e.g. convergence) on the estimation of the starting values for `walsGLMfit`. See `glm.fit` for more information.
`weights`	returns the argument `weights`.

offset	returns the argument offset.
converged	Logical. Only relevant if iterate = TRUE. Equals TRUE if iterative fitting converged, else FALSE. Is NULL if iterate = FALSE.
it	Number of iterations run in the iterative fitting algorithm. NULL if iterate = FALSE.
deviance	Deviance of the fitted regression model.
residuals	Raw residuals, i.e. response - fitted mean.

See Also

walsGLM, walsGLMfit.

Examples

```
data("HMDA", package = "AER")
X <- model.matrix(deny ~ pirat + hirat + lvrat + chist + mhist + phist + selfemp + afam,
                  data = HMDA)
X1 <- X[,c("(Intercept)", "pirat", "hirat", "lvrat", "chist2", "chist3",
        "chist4", "chist5", "chist6", "mhist2", "mhist3", "mhist4", "phistyes")]
X2 <- X[,c("selfempyes", "afamyes")]
y <- HMDA$deny

str(walsGLMfitIterate(y, X1, X2, family = binomialWALS(), prior = weibull(),
                      iterate = TRUE))
```

---

walsNB *Weighted-Average Least Squares for Negative Binomial Regression*

Description

Performs model averaging for NB2 regression models using the Weighted-Average Least Squares method of Huynh (2023a).

Usage

```
walsNB(x, ...)

## S3 method for class 'formula'
walsNB(
  formula,
  data,
  subset = NULL,
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  link = "log",
```

```
  prior = weibull(),
  controlInitNB = controlNB(),
  model = TRUE,
  keepY = TRUE,
  keepX = FALSE,
  iterate = FALSE,
  tol = 1e-06,
  maxIt = 50,
  nIt = NULL,
  verbose = FALSE,
  ...
)

## S3 method for class 'matrix'
walsNB(
  x,
  x2,
  y,
  link = "log",
  subset = NULL,
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  prior = weibull(),
  controlInitNB = controlNB(),
  model = TRUE,
  keepY = TRUE,
  keepX = FALSE,
  iterate = FALSE,
  tol = 1e-06,
  maxIt = 50,
  nIt = NULL,
  verbose = FALSE,
  ...
)

## Default S3 method:
walsNB(x, ...)
```

Arguments

`x`	Design matrix of focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.
`...`	Arguments for workhorse `walsNBfit`.
`formula`	an object of class "`Formula`" (or one that can be coerced to that class, e.g. "`formula`"): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame`

	to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment which the function is called from.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	**not implemented yet.**
`weights`	**not implemented yet.**
`offset`	**not implemented yet.**
`link`	specifies the link function, currently only `"log"` is supported.
`prior`	Object of class `"familyPrior"`. For example `weibull` or `laplace`.
`controlInitNB`	Controls estimation of starting values for one-step ML, see `controlNB`.
`model`	if `TRUE` (default), then the model.frame is stored in the return.
`keepY`	if `TRUE` (default), then the response is stored in the return.
`keepX`	if `TRUE`, then the model matrices are stored in the return. the return.
`iterate`	if `TRUE` then the WALS algorithm is iterated using the previous estimates as starting values.
`tol`	Only used if `iterate = TRUE` and `nIt = NULL`. If the Euclidean distance between the previous and current coefficient vector divided by the square root of the length of the vector falls below `tol` and the absolute difference between the previous and current dispersion parameter falls below `tol`, then the algorithm stops. See `walsNBfitIterate` for more details.
`maxIt`	Only used if `iterate = TRUE` and `nIt = NULL`. Aborts iterative fitting when number of iterations exceed `maxIt`.
`nIt`	Only used if `iterate = TRUE`. If this is specified, then `tol` is ignored and the algorithm iterates `nIt` times. This option should not be used unless the user has a specific reason to run the algorithm `nIt` times, e.g. for replication purposes.
`verbose`	If `verbose = TRUE`, then it prints the iteration process of internal function `walsNBfitIterate` (only relevant if `iterate = TRUE`).
`x2`	Design matrix of auxiliary regressors. Usually does not include a constant column and can also be generated using `model.matrix`.
`y`	Count response as vector.

Details

Computes WALS estimates when focus regressors (X1) are present in all submodels and model averaging takes place over the auxiliary regressors (X2).

Formulas should always contain two parts, i.e. they should be of the form "`y ~ X11 + X12 | X21 + X22`", where the variables before "`|`" are the focus regressors (includes a constant by default) and the ones after "`|`" are the auxiliary regressors.

WARNING: Interactions in formula do not work properly yet. It is recommended to manually create the interactions beforehand and then to insert them as 'linear terms' in the formula.

See `predict.walsGLM` and `predict.wals` for some class methods that the fitted objects inherit from "`walsGLM`" and "`wals`", respectively.

walsNB.default() raises an error if x is not an object of class "matrix" or a class that extends "matrix". Otherwise it calls walsNB.matrix(). It is a modified version of glmboost.default from the mboost package version 2.9-8 (2023-09-06) (Hofner et al. 2014).

Value

walsNB.formula() returns an object of class "walsNB" which inherits from "walsGLM" and "wals". This is a list that contains all elements returned from walsNBfitIterate and additionally

cl	Call of the function.
formula	formula used.
terms	List containing the model terms of the focus and auxiliary regressors separately, as well as for the full model.
levels	List containing the levels of the focus and auxiliary regressors separately, as well as for the full model.
contrasts	List containing the contrasts of the design matrices of focus and auxiliary regressors.
model	If model = TRUE, contains the model frame.

See returns of walsNBfit and walsNBfitIterate for more details.

walsNB.matrix() returns an object of class "walsNBmatrix", which inherits from "walsNB", "walsGLMmatrix", "walsGLM" and "wals". This is a list that contains all elements returned from walsNBfitIterate and additionally the call in cl.

walsNB.default() raises an error if x is not an object of class "matrix" or a class that extends "matrix". Otherwise returns an object of class "walsNBmatrix". See above for more details.

References

Hofner B, Mayr A, Robinzonov N, Schmid M (2014). "Model-based Boosting in R: A Hands-on Tutorial Using the R Package mboost." *Computational Statistics*, **29**, 3–35.

Huynh K (2023a). "Weighted-Average Least Squares for Negative Binomial Regression." University of Basel. Mimeo.

Examples

```
## Example for walsNB.formula()
data("NMES1988", package = "AER")

fitWeibull <- walsNB(visits ~ health + chronic + age + gender | I((age^2)/10) +
                     married + region, data = NMES1988, prior = weibull())
summary(fitWeibull)

fitLaplace <- walsNB(visits ~ health + chronic + age + gender | I((age^2)/10) +
                     married + region, data = NMES1988, prior = laplace())
summary(fitLaplace)

## Example for walsNB.matrix()
data("NMES1988", package = "AER")
```

```
X <- model.matrix(visits ~ health + chronic + age + gender + married + region,
                  data = NMES1988)
X1 <- X[, c("(Intercept)", "healthpoor", "healthexcellent", "chronic",
        "age", "gendermale")]
X2 <- X[, c("marriedyes", "regionnortheast", "regionmidwest", "regionwest")]
y <- NMES1988$visits
fit <- walsNB(X1, X2, y, prior = weibull())
summary(fit)
```

---

walsNBfit — *Fitter function for Weighted Average Least Squares estimation of NB2 regression model*

Description

Workhorse function behind `walsNB` and used internally in `walsNBfitIterate`.

Usage

```
walsNBfit(
  X1,
  X2,
  y,
  betaStart1,
  betaStart2,
  rhoStart,
  family,
  prior,
  method = c("fullSVD", "original"),
  svdTol = .Machine$double.eps,
  svdRtol = 1e-06,
  keepUn = FALSE,
  keepR = FALSE,
  eigenSVD = TRUE,
  postmult = TRUE,
  ...
)
```

Arguments

`X1`	Design matrix for focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.
`X2`	Design matrix for auxiliary regressors. Usually does not include a constant column and can also be generated using `model.matrix`.
`y`	Count response as vector.
`betaStart1`	Starting values for coefficients of focus regressors X1.

`betaStart2` Starting values for coefficients of auxiliary regressors X2.

`rhoStart` Starting value for log-dispersion parameter of NB2

`family` Object of class `"familyNBWALS"`. Currently only supports `negbinWALS`.

`prior` Object of class `"familyPrior"`. For example `weibull` or `laplace`.

`method` Specifies method used. Available methods are `"fullSVD"` (default) or `"original"`. See details.

`svdTol` Tolerance for rank of matrix $\bar{Z}_1$ and $\bar{Z}$. Only used if `method = "fullSVD"`. Checks if smallest eigenvalue in SVD of $\bar{Z}_1$ and $\bar{Z}$ is larger than `svdTol`, otherwise reports a rank deficiency.

`svdRtol` Relative tolerance for rank of matrix $\bar{Z}_1$ and $\bar{Z}$. Only used if `method = "fullSVD"`. Checks if ratio of largest to smallest eigenvalue in SVD of $\bar{Z}_1$ and $\bar{Z}$ is larger than `svdRtol`, otherwise reports a rank deficiency.

`keepUn` If `TRUE`, keeps the one-step ML estimators of the unrestricted model, i.e. $\tilde{\gamma}_u$ and $\tilde{\beta}_u$.

`keepR` If `TRUE`, keeps the one-step ML estimators of the fully restricted model, i.e. $\tilde{\gamma}_r$ and $\tilde{\beta}_r$.

`eigenSVD` If `TRUE`, then `semiorthogonalize()` uses `svd()` to compute the eigendecomposition of $\bar{\Xi}$ instead of `eigen()`. In this case, the tolerances of `svdTol` and `svdRtol` are used to determine whether $\bar{\Xi}$ is of full rank (need it for $\bar{\Xi}^{-1/2}$).

`postmult` If `TRUE` (default), then it computes

$$\bar{Z}_2 = \bar{X}_2\bar{\Delta}_2\bar{T}\bar{\Lambda}^{-1/2}\bar{T}^{\top},$$

where $\bar{T}$ contains the eigenvectors and $\bar{\Lambda}$ the eigenvalues from the eigenvalue decomposition

$$\bar{\Xi} = \bar{T}\bar{\Lambda}\bar{T}^{\top},$$

instead of

$$\bar{Z}_2 = \bar{X}_2\bar{\Delta}_2\bar{T}\bar{\Lambda}^{-1/2}.$$

See Huynh (2023b) for more details. The latter is used in the original MATLAB code for WALS in the linear regression model, see eq. (12) of Magnus and De Luca (2016). The first form is required in eq. (9) of De Luca et al. (2018). **Thus, it is not recommended to set** `postmult = FALSE`**.**

`...` Arguments for internal function `computePosterior`.

Details

The method to be specified in `method` mainly differ in the way they compute the fully restricted and unrestricted estimators for the transformed regressors Z, i.e. $\tilde{\gamma}_{1r}$, and $\tilde{\gamma}_u$.

"fullSVD" Recommended approach. First applies an SVD to $\bar{Z}_1$ to compute $\bar{X}_2^{\top}\bar{M}_1\bar{X}_2$: It is used for computing the inverse of

$$\bar{X}_1^{\top}\bar{X}_1 + \bar{g}\bar{\epsilon}X_1^{\top}\bar{q}\bar{q}^{\top}X_1,$$

when using the Sherman-Morrison-Woodbury formula. We further leverage the SVD of $\bar{Z}_1$ and additionally $\bar{Z}$ to compute the unrestricted estimator $\tilde{\gamma}_u$ and the fully restricted estimator

$\tilde{\gamma}_r$. For $\tilde{\gamma}_u$, we simply use the SVD of $\bar{Z}$ to solve the full equation system derived from the one-step ML problem for more details. The SVD of $\bar{Z}_1$ is further used in computing the model averaged estimator for the focus regressors $\hat{\gamma}_1$.

Described in more detail in the appendix of Huynh (2023b).

"original" Computes all inverses directly using `solve` and does not make use of the Sherman-Morrison-Woodbury formula for certain inverses. Specifically, it directly inverts the matrix $\bar{Z}_1^{\top}\bar{Z}_1$ using `solve` in order to compute $\bar{M}_1$. Moreover, it computes the fully unrestricted estimators of the focus regressors $\tilde{\gamma}_{1u}$ and of the auxiliary regressors $\tilde{\gamma}_{2u}$ and the fully restricted estimator $\tilde{\gamma}_{1r}$ by directly implementing the formulas derived in Huynh (2023a). This method should only be used as reference and for easier debugging.

All variables in the code that contain "start" in their name are computed using the starting values of the one-step ML estimators. See section "One-step ML estimator" of (Huynh 2023a) for details.

Value

A list containing

`coef`	Model averaged estimates of all coefficients.
`beta1`	Model averaged estimates of the coefficients of the focus regressors.
`beta2`	Model averaged estimates of the coefficients of the auxiliary regressors.
`rho`	Model averaged estimate of the log-dispersion parameter of the NB2 distribution.
`gamma1`	Model averaged estimates of the coefficients of the transformed focus regressors.
`gamma2`	Model averaged estimates of the coefficients of the transformed auxiliary regressors.
`condition`	Condition number of the matrix $\bar{\Xi} = \bar{\Delta}_2\bar{X}_2^{\top}\bar{M}_1\bar{X}_2\bar{\Delta}_2$.
`vcovBeta`	`NULL`, not implemented yet, placeholder for estimated covariance matrix of the regression coefficients.
`vcovGamma`	`NULL`, not implemented yet, placeholder for estimated covariance matrix of the coefficients of the transformed regressors.
`betaStart`	Starting values of the regression coefficients for the one-step ML estimators.
`rhoStart`	Starting values of the dispersion parameter for the one-step ML estimators.
`method`	Stores `method` used from the arguments.
`prior`	`familyPrior`. The `prior` specified in the arguments.
`betaUn1`	If `keepUn = TRUE`, contains the unrestricted one-step ML estimators of the coefficients of the focus regressors. Else `NULL`.
`betaUn2`	If `keepUn = TRUE`, contains the unrestricted one-step ML estimators of the coefficients of the auxiliary regressors. Else `NULL`.
`gammaUn1`	If `keepUn = TRUE`, contains the unrestricted one-step ML estimators of the coefficients of the transformed focus regressors. Else `NULL`.
`gammaUn2`	If `keepUn = TRUE`, contains the unrestricted one-step ML estimators of the coefficients of the transformed auxiliary regressors. Else `NULL`.

gamma1r	If keepR = TRUE, contains the fully restricted one-step ML estimator for the transformed regressors (only focus regressors). Else NULL.
k1	Number of focus regressors.
k2	Number of auxiliary regressors.
n	Number of observations.
X1names	Names of the focus regressors.
X2names	Names of the auxiliary regressors.
familyStart	The family object of class "familyNBWALS" used for the estimation of the starting values.
family	The family object of class "familyNBWALS" used later for predictions.
fitted.link	Linear link fitted to the data.
fitted.values	Estimated conditional mean for the data. Lives on the scale of the response.

References

De Luca G, Magnus JR, Peracchi F (2018). "Weighted-average least squares estimation of generalized linear models." *Journal of Econometrics*, **204**(1), 1–17. doi:10.1016/j.jeconom.2017.12.007.

Huynh K (2023a). "Weighted-Average Least Squares for Negative Binomial Regression." University of Basel. Mimeo.

Huynh K (2023b). "WALS: Weighted-Average Least Squares Model Averaging in R." University of Basel. Mimeo.

Magnus JR, De Luca G (2016). "Weighted-average least squares (WALS): A survey." *Journal of Economic Surveys*, **30**(1), 117-148. doi:10.1111/joes.12094.

See Also

walsNB, walsNBfitIterate.

Examples

```
data("NMES1988", package = "AER")
NMES1988 <- na.omit(NMES1988)
form <- (visits ~ health + chronic + age + insurance + adl + region + gender
         + married + income + school + employed)
X <- model.matrix(form, data = NMES1988)
focus <- c("(Intercept)", "healthpoor", "healthexcellent", "chronic", "age",
        "insuranceyes")
aux <- c("adllimited", "regionnortheast", "regionmidwest", "regionwest",
         "gendermale", "marriedyes", "income", "school", "employedyes")
X1 <- X[, focus]
X2 <- X[, aux]
y <- NMES1988$visits

# starting values from glm.nb() from MASS
startFit <- MASS::glm.nb(y ~ X[,-1])
```

```
betaStart <- coef(startFit)
rhoStart <- startFit$theta
k1 <- ncol(X1)
k2 <- ncol(X2)

str(walsNBfit(X1, X2, y, rhoStart, family = negbinWALS(scale = rhoStart, link = "log"),
              betaStart1 = betaStart[1:k1],
              betaStart2 = betaStart[(k1 + 1):(k1 + k2)],
              prior = weibull(), method = "fullSVD"))
```

`walsNBfitIterate` *Iteratively fitting walsNB, internal function for walsNB.formula and walsNB.matrix.*

Description

Wrapper around `walsNBfit` that allows iteratively (re-)fitting `walsNB` models.

Usage

```
walsNBfitIterate(
  y,
  X1,
  X2,
  link = "log",
  na.action = NULL,
  weights = NULL,
  offset = NULL,
  prior = weibull(),
  controlInitNB = controlNB(),
  keepY = TRUE,
  keepX = FALSE,
  iterate = FALSE,
  tol = 1e-06,
  maxIt = 50,
  nIt = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`y`	Count response as vector.
`X1`	Design matrix for focus regressors. Usually includes a constant (column full of 1s) and can be generated using `model.matrix`.
`X2`	Design matrix for auxiliary regressors. Usually does not include a constant column and can also be generated using `model.matrix`.

link	specifies the link function, currently only "log" is supported.
na.action	Not implemented yet.
weights	Not implemented yet.
offset	Not implemented yet.
prior	Object of class "familyPrior". For example weibull or laplace.
controlInitNB	Controls estimation of starting values for one-step ML, see controlNB.
keepY	If TRUE, then output keeps response.
keepX	If TRUE, then output keeps the design matrices.
iterate	if TRUE then the WALS algorithm is iterated using the previous estimates as starting values.
tol	Only used if iterate = TRUE and nIt = NULL. If the Euclidean distance between the previous and current coefficient vector divided by the square root of the length of the vector falls below tol and the absolute difference between the previous and current dispersion parameter falls below tol, then the algorithm stops. See below for more details.
maxIt	Only used if iterate = TRUE and nIt = NULL. Aborts iterative fitting when number of iterations exceed maxIt.
nIt	Only used if iterate = TRUE. If this is specified, then tol is ignored and the algorithm iterates nIt times.
verbose	If verbose = TRUE, then it prints the iteration process (only relevant if iterate = TRUE).
...	Arguments to be passed to the workhorse function walsNBfit.

Details

The parameter tol is used to control the convergence of the iterative fitting algorithm. Let i be the current iteration step for the coefficient vector $\beta_i = (\beta_{i,1}, \ldots, \beta_{i,k})'$, $k > 0$, and dispersion parameter ρ_i. If

$$\frac{||\beta_i - \beta_{i-1}||_2}{\sqrt{k}} = \sqrt{\frac{\sum_{j=1}^{k} (\beta_{i,j} - \beta_{i-1,j})^2}{k}} < \texttt{tol},$$

and

$$|\rho_i - \rho_{i-1}| < \texttt{tol},$$

then the fitting process is assumed to have converged and stops.

Value

A list containing all elements returned from walsNBfit and additionally the following elements:

y	If keepY = TRUE, contains the response vector.
x	list. If keepX = TRUE, then it is a list with elements x1 and x2 containing the design matrices of the focus and auxiliary regressors, respectively.
initialFit	List containing information (e.g. convergence) on the estimation of the starting values for walsNBfit. See return of fitNB2 for more information.

weights	returns the argument weights.
offset	returns the argument offset.
converged	Logical. Only relevant if iterate = TRUE. Equals TRUE if iterative fitting converged, else FALSE. Is NULL if iterate = FALSE.
it	Number of iterations run in the iterative fitting algorithm. NULL if iterate = FALSE.
deviance	Deviance of the fitted (conditional) NB2 regression model.
residuals	Raw residuals, i.e. response - fitted mean.

See Also

walsNB, walsNBfit.

Examples

```
data("NMES1988", package = "AER")
NMES1988 <- na.omit(NMES1988)
form <- (visits ~ health + chronic + age + insurance + adl + region + gender
         + married + income + school + employed)
X <- model.matrix(form, data = NMES1988)
focus <- c("(Intercept)", "healthpoor", "healthexcellent", "chronic", "age",
        "insuranceyes")
aux <- c("adllimited", "regionnortheast", "regionmidwest", "regionwest",
         "gendermale", "marriedyes", "income", "school", "employedyes")
X1 <- X[, focus]
X2 <- X[, aux]
y <- NMES1988$visits

str(walsNBfitIterate(y, X1, X2, prior = weibull(), link = "log",
                     method = "fullSVD", iterate = TRUE))
```

Index