Package ‘modeLLtest’

June 11, 2020

Type Package
Title Compare Models with Cross-Validated Log-Likelihood
Version 1.0.2
Date 2020-06-10
Author Shana Scogin <shanarscogin@gmail.com>, Sarah Petersen <sarahllpetersen@gmail.com>, Jeff Harden <jeff.harden@nd.edu>, Bruce A. Desmarais <bdesmarais@psu.edu>
Maintainer Shana Scogin <shanarscogin@gmail.com>

URL https://github.com/ShanaScogin/modeLLtest
License GPL-3
NeedsCompilation yes
BugReports https://github.com/ShanaScogin/modeLLtest/issues
Imports stats, quantreg, survival, coxrobust, MASS, Rcpp
Encoding UTF-8
LazyData TRUE
LazyLoad TRUE
Suggests knitr, rmarkdown, testthat
VignetteBuilder knitr
SystemRequirements GNU make
RoxygenNote 6.1.1
LinkingTo Rcpp, RcppArmadillo
Description

Applies cross-validated log-likelihood difference in means test to compare two methods of estimating a formula. The output identifies the more appropriate model.

In choosing between OLS and MR, please cite:


For other applications of the CVDM test, please cite:


Usage

cvdm(formula, data, method1 = c("OLS", "MR", "RLM", "RLM-MM"), method2 = c("OLS", "MR", "RLM", "RLM-MM"), subset, na.action, ...)
Arguments

- **formula**: A formula object, with the dependent variable on the left of a ~ operator, and the independent variables on the right.

- **data**: A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.

- **method1**: A method to estimate the model. Currently takes Ordinary Least Squares ("OLS"), Median Regression ("MR"), Robust Linear Regression ("RLM") using M-estimation, and Robust Linear Regression using MM-estimation ("RLM-MM"). The algorithm method used to compute the fit for the median regression is the modified version of the Barrodale and Roberts algorithm for l1-regression, which is the `rq` default by R package quantreg. See quantreg `rq` function documentation for more details. Fitting for the robust regressions is done by iterated re-weighted least squares (IWLS) and is taken from the MASS package `rlm` function. The MM-estimation is the M-estimation with Tukey’s biweight initialized by a specific S-estimate. The M-estimation, which can be achieved in this package with the option "RLM", is the default for the MASS `rlm` function. See MASS package `rlm` documentation for details.

- **method2**: A method to estimate the model. Options are same as for method1.

- **subset**: Expression indicating which subset of the rows of data should be used in the fit. All observations are included by default.

- **na.action**: A missing-data filter function, applied to the model.frame, after any subset argument has been used.

- **...**: Optional arguments, currently unsupported.

Details

This function implements the cross-validated difference in means (CVDM) test between two methods of estimating a formula. The function takes a formula and two methods and computes a vector of cross-validated log-likelihoods (CVLLs) for each method using the leave-one-out method. These output test score is the cross-validated Johnson’s t-test. A positive test statistic supports the first method and a negative test statistic supports the second. Singular matrices during the leave-one-out cross-validation process are skipped.

Value

An object of class `cvdm` computed by the cross-validated log likelihood difference in means test (CVDM). The object is the Cross-Validated Johnson’s t-test. A positive test statistic supports the first method and a negative test statistic supports the second. See `cvdm_object` for more details.

References


Examples

```r
set.seed(123456)
b0 <- .2 # True value for the intercept
b1 <- .5 # True value for the slope
n <- 500 # Sample size
X <- runif(n, -1, 1)
Y <- b0 + b1 * X + rnorm(n, 0, 1) # N(0, 1 error)

obj_cvdm <- cvdm(Y ~ X, data.frame(cbind(Y, X)), method1 = "OLS", method2 = "MR")
```

---

**cvdm_object**  
Cross-Validated Difference in Means (CVDM) Object

---

**Description**

This class of objects is returned by the `cvdm` function to compare two methods of estimating a formula.

**Value**

The following components must be included in a legitimate `cvdm` object.

- **best**: name of the estimation method favored by the `cvdm` test.
- **test_stat**: object returned by the bias-corrected Johnson's t-test. A positive test statistic supports method 1 and a negative test statistic supports method 2.
- **p_value**: p-value for the test statistic.
- **n**: number of observations.
- **df**: degrees of freedom.

The object also contain the following: `call`, `x`, and `y`. See `lm` documentation for more.

**See Also**

- `cvdm`
cvll

Cross-Validated Log Likelihood (CVLL)

Description

Extracts the leave-one-out cross-validated log-likelihoods from a method of estimating a formula.

Usage

```r
cvll(formula, data, method = c("OLS", "MR", "RLM", "RLM-MM"), subset, 
    na.action, ...)
```

Arguments

- `formula`: A formula object, with the dependent variable on the left of a ~ operator, and the independent variables on the right.
- `data`: A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model.
- `method`: A method to estimate the model. Currently takes Ordinary Least Squares ("OLS"), Median Regression ("MR"), Robust Linear Regression ("RLM") using M-estimation, and Robust Linear Regression using MM-estimation ("RLM-MM"). The algorithm method used to compute the fit for the median regression is the modified version of the Barrodale and Roberts algorithm for l1-regression, which is the `rq` default by R package quantreg. See quantreg `rq` function documentation for more details. Fitting for the robust regressions is done by iterated re-weighted least squares (IWLS) and is taken from the MASS package `rlm` function. The MM-estimation is the M-estimation with Tukey’s biweight initialized by a specific S-estimate. The M-estimation, which can be achieved in this package with the option "RLM", is the default for the MASS `rlm` function. See MASS package `rlm` documentation for details.
- `subset`: Expression indicating which subset of the rows of data should be used in the fit. All observations are included by default.
- `na.action`: A missing-data filter function, applied to the model.frame, after any subset argument has been used.
- `...`: Optional arguments, currently unsupported.

Details

This function extracts a vector of leave-one-out cross-validated log likelihoods (CVLLs) from a method of estimating a formula. Singular matrices during the leave-one-out cross-validation process are skipped.

Value

An object of class cvll computed by the cross-validated log likelihood (CVLL). See `cvdm_object` for more details.
References


Examples

```r
set.seed(123456)
b0 <- .2  # True value for the intercept
b1 <- .5  # True value for the slope
n <- 500  # Sample size
X <- runif(n, -1, 1)

Y <- b0 + b1 * X + rnorm(n, 0, 1)  # N(0, 1 error)

obj_cvll <- cvll(Y ~ X, data.frame(cbind(Y, X)), method = "OLS")
```

---

**cvlldiff**

*Cross-Validated Difference in Means (CVDM) Test with Vector Imputs*

**Description**

Applies cross-validated log-likelihood to test between two methods of estimating a formula. The output identifies the vector from the more appropriate model.

Please cite:


**Usage**

`cvlldiff(vector1, vector2, df)`

**Arguments**

- `vector1`: A numeric vector of cross-validated log-likelihoods.
- `vector2`: A numeric vector of cross-validated log-likelihoods.
- `df`: A value of the degrees of freedom in the models.
Details

This function implements the cross-validated difference in means (CVDM) test between two vectors of cross-validated log-likelihoods. A positive test statistic supports the method that produced the first vector and a negative test statistic supports the second.

Value

An object of class cvlldiff computed by the cross-validated log likelihood difference in means test (CVDM). The test statistic object is the Cross-Validated Johnson’s t-test. A positive test statistic supports the first method and a negative test statistic supports the second. See cvdm_object for more details.

References


Examples

```r
set.seed(123456)
b0 <- .2 # True value for the intercept
b1 <- .5 # True value for the slope
n <- 500 # Sample size
X <- runif(n, -1, 1)
Y <- b0 + b1 * X + rnorm(n, 0, 1) # N(0, 1 error)
cvll_ols <- cvll(Y ~ X, data.frame(cbind(Y, X)), method = "OLS")
cvll_mr <- cvll(Y ~ X, data.frame(cbind(Y, X)), method = "MR")
obj_compare <- cvlldiff(cvll_ols$cvll, cvll_mr$cvll, cvll_ols$df)
```

---

**cvlldiff_object**  
**Cross-Validated Difference in Means (CVDM) Object from General cvlldiff Function**

Description

This class of objects is returned by the cvlldiff function to compare vectors of cross-validated log-likelihood values.

Value

The following components must be included in a legitimate cvlldiff object.

- **best**: name of the estimation method favored by the cvdm test.
test_stat  object returned by the bias-corrected Johnson’s t-test. A positive test statistic supports the method that generated the first vector of cross-validated log-likelihood values and a negative test statistic supports the method that generated the second vector.

p_value  p-value for the test statistic.

See Also

cvlldiff

cvll_object  \textit{Cross-Validated Log-Likelihood (CVLL) Object}

\subsection*{Description}

This class of objects is returned by the \texttt{cvll} function.

\subsection*{Value}

The following components must be included in a legitimate cvll object.

\begin{itemize}
  \item \texttt{cvll}  vector of cross-validated log-likelihood values using the leave-one-out method.
  \item \texttt{n}  number of observations.
  \item \texttt{df}  degrees of freedom.
  \item \texttt{method}  method of estimation.
\end{itemize}

The object also contain the following: \texttt{call}, \texttt{x}, and \texttt{y}. See \texttt{lm} documentation for more.

See Also

cvll

cvmf  \textit{Cross-Validated Median Fit (CVMF) Test}

\subsection*{Description}

Usage

cvmf(formula, data, method = c("exact", "approximate", "efron", "breslow"),
      trunc = 0.95, subset, na.action, f.weight = c("linear", "quadratic", "exponential"), weights, singular.ok = TRUE)

Arguments

formula A formula object, with the response on the left of a ~ operator, and the terms on the right. The response must be a survival object as returned by the Surv function from the survival package.
data A data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model or in the subset and the weights argument.
method A character string specifying the method for tie handling in coxph(). If there are no tied death times all the methods are equivalent. Following the coxph function in the survival package, the Efron approximation is used as the default. The survival package justifies this due to the Efron method being more accurate when dealing with tied death times, and is as efficient computationally than the common Breslow method. The "exact partial likelihood" is equivalent to a 'conditional logistic model, and is appropriate when the times are a small set of discrete values. This argument does not exist in the coxr function in the coxrobust package. For coxr, method is based on a smooth modification of the partial likelihood. See documentation from survival package for more on coxph method and coxrobust package for coxr method.
trunc A value that determines the trimming level for the robust estimator. The default is 0.95. Roughtly, quantile of the sample \( T_i \exp(\beta'Z_i) \). It is an argument in the coxr function in the coxrobust package.
subset Expression indicating which subset of the rows of data should be used in the fit. All observations are included by default.
na.action A missing-data filter function, applied to the model.frame, after any subset argument has been used.
f.weight A type of weighting function for coxr in the coxrobust package. The default is quadratic. See coxr documentation for more.
weights A vector of case weights for coxph in the survival package. See coxph documentation for more.
singular.ok Logical value indicating how to handle collinearity in the model matrix. If TRUE, the program will automatically skip over columns of the X matrix that are linear combinations of earlier columns. In this case the coefficients for such columns will be NA, and the variance matrix will contain zeros. For ancillary calculations, such as the linear predictor, the missing coefficients are treated as zeros.

Details

This function implements the cross-validated median fit (CVMF) test. The function cvmf() tests between the partial likelihood maximization (PLM) and the iteratively reweighted robust (IRR)
method of estimation for a given application of the Cox model. The Cox model is a partial parametric model that does not make assumptions about the baseline hazard. It can be estimated via PLM, the standard estimator, or IRR, a robust estimator that identifies and downweights outliers. The choice between the two methods involves a trade-off between bias and efficiency. PLM is more efficient, but biased under specification problems. IRR reduces bias, but results in high variance due to the loss of efficiency. The cvmf() function returns an object to identify the preferred estimation method.

See also `coxph`, `coxr`, `Surv`
**Value**

The following components must be included in a legitimate `cvmf` object.

- **best**: name of the model of estimation favored by the `cvmf` test.
- **p**: p-value of the binomial test used to test between estimation models.
- **cvmf**: full output of the binomial test used to test between estimation methods. See documentation for `binom.test` for more information.
- **coef_names**: names of the coefficients.
- **irr**: full output for the iteratively reweighted robust (IRR) method of estimating the Cox model. See documentation for `coxr` in the package coxrobust for more information.
- **plm**: full output for the partial likelihood maximization (PLM) method of estimating the Cox model. See documentation for `coxph` in the package survival for more information.
- **irr_coefs**: estimates obtained from IRR method of estimating the Cox model. See documentation for `coxr` in the package coxrobust for more information.
- **plm_coefs**: estimates obtained from PLM method of estimating the Cox model. See documentation for `coxph` in the package survival for more information.
- **cvpl_irr**: observation-wise contributions to the log-partial likelihood for IRR method of estimating the Cox model. See Desmarais and Hardin (Political Analysis 20:113-135, 2012) for more about the test and Verweij and Houwelingen (Statistics in Medicine 12(24): 2305–14, 1993) for more about the measure.
- **cvpl_plm**: observation-wise contributions to the log-partial likelihood for PLM method of estimating the Cox model. See Desmarais and Hardin (Political Analysis 20:113-135, 2012) for more about the test and Verweij and Houwelingen (Statistics in Medicine 12(24): 2305–14, 1993) for more about the measure.

The object also contain the following: `call`, `x`, and `y`.

**See Also**

- `cvmf`

---

**govtform**  
*Data from Golder (2010) on government formation in Western Europe*

**Description**

Data from a study on Western European government formation duration. Data is at the country-level (N = 409). Variable names are taken directly from original dataset. The data is publicly available and has been included here with the endorsement of the author. Please see the original codebook for a more detailed description of the variables.
Usage

data(govtform)

Format

A data frame with 410 rows and 18 variables. The following are taken from the codebook at Dr. Sona N. Golder’s Harvard Dataverse Page.

countriname  names of countries used in analysis
country  unique number identifying each country
cabinet  unique number identifying each country. Begins with country code, followed by cabinets 1 - n
bargainingdays  the number of days between either an election or the resignation of the previous government and the day on which the new government is officially inaugurated
datein  date on which a government took office. Format is YYMMDD
dateout  date on which a government left office. Format is YYMMDD
postelection  dichotomous variable that equals 1 if a government is the first to form after an election (more uncertainty) and 0 if it forms in an interelection period (less uncertainty)
nonpartisan  dichotomous variable that equals 1 if the government is nonpartisan and 0 otherwise
legislative_parties  a fraction representing the number of parties that have won legislative seats. See codebook for more detail
inconclusive  the number of inconclusive bargaining rounds prior to a new government successfully forming
cabinetname  cabinet name identified by surname of prime minister (followed by a number if the PM presided over more than one cabinet)
singleparty_majority  dichotomous variable that equals 1 if a single party controls a majority of the legislative seats, 0 otherwise
polarization  measures the level of ideological polarization in the party system. See codebook for more detail
continuation  dichotomous variable that equals 1 if the outgoing government or formateur gets the first opportunity to form a new government, 0 otherwise. See codebook for more detail
positive_parl  dichotomous variable that equals 1 if a new government requires the explicit support of a legislative majority in order to take office, 0 otherwise. See codebook for more detail
post_legislative_parties  interaction term made by multiplying the postelection variable with the legislative_parties variable
post_polariz  interaction term made by multiplying the postelection variable with the polarization variable
post_positive  interaction term made by multiplying the postelection variable with the positive_parl variable

Source

Dr. Sona N. Golder’s Harvard Dataverse Page
References


Examples

library(survival)
library(coxrobust)
library(modeLLtest)

# Survival models with data from Golder (2010)
data(govtform)
golder_surv <- Surv(govtform$bargainingdays)
golder_x <- cbind(govtform$postelection, govtform$legislative_parties,
govtform$polarization, govtform$positive_parl, govtform$post_legislative_parties,
govtform$post_polariz, govtform$post_positive, govtform$continuation,
govtform$singleparty_majority)
colnames(golder_x) <- c("govtform$postelection", "govtform$legislative_parties",
"govtform$polarization", "govtform$positive_parl", "govtform$post_legislative_parties",
"govtform$post_polariz", "govtform$post_positive", "govtform$continuation",
"govtform$singleparty_majority")
golder_cox <- coxph(golder_surv ~ golder_x, method = "efron",
data = govtform)
golder_robust <- coxr(golder_surv ~ golder_x, data = govtform)

# Comparing PLM to IRR methods of estimating the survival model
obj_cvmf_golder <- cvmf(golder_surv ~ golder_x, method = "efron",
data = govtform)
obj_cvmf_golder

Description

modeLLtest Cross Validated Log Likelihood test functions

To use this package, decide which specification(s) of a model and distributions you wish compare. The functions in this package compare the fits of one model specification between a median regression and ordinary least squares (cvdm()), between the fits of one model specification between two estimations of a Cox model (cvmf()), and between two model specification and one distribution (cvll()).
Description

Data from a study on the relationship between land tenure and voter turnout in the three rounds of parliamentary elections in Nepal from the restoration of democracy in 1990 to 1999. Data is at the district-level (N = 75). Variable names are taken directly from original dataset. The data is publicly available and has been included here with the endorsement of the authors.

Usage

data(nepaldem)

Format

A data frame with 76 rows and 73 variables:

- sn: a column of identifiers. This column is not a variable
- district: names of the district in Nepal used in analysis
- householdsize: average size of household in district
- total_holding: total land holding
- noown_single_tenure: number of households that own and cultivate land under single tenure
- norent_single_ten: number of households that rent for service and cultivate land under single tenure
- noother_single_ten: number of households that cultivate under single tenure and have another set up other than those above
- nomore1_ten_hold: number of households with more than one tenure
- noholding_below1_pa: number of households that hold less than 1.0 hectares of land
- noholding_2to3_pa: number of households that hold 2 to 3 hectares of land
- noholding_4to5_pa: number of households that hold 4 to 5 hectares of land
- noholding_6to9_pa: number of households that hold 6 to 9 hectares of land
- noholding_10_pa: number of households with more than 10 parcels of land
- total_ha: total hectares of land
- total.Parcel: total parcels of land
- no_hold_fixmoney2: subsection of number of households with fixed cash rent
- no_hold_fixproduct2: subsection of households with fixed product rent
- no_hold_share2: subsection of households participating in sharecropping
- no_hold_services2: subsection of households participating in sharecropping
- no_hold_mortgage2: subsection of households with a mortgage
- no_hold_fixmoney1: subsection of households with fixed cash rent
no_hold_fixproduct1 subsection of households with fixed product rent
no_hold_share1 subsection of households participating in sharecropping
no_hold_services1 subsection of households with rent for service
no_hold_mortgage1 subsection of households with a mortgage

totalhouseholds total number of households
landless number of landless households

totalvoters1991 total number of voters in 1991
totalcastedvote1991 total number of votes cast in 1991
totalvalidvote1991 total number of valid votes in 1991
constituency1991 constituency in 1991
totalcontestants1991 total number of candidates contesting elections in 1991
totalvoters1994 total number of voters in 1994
totalcastedvote1994 total number of votes cast in 1994
totalvalidvote1994 total number of valid votes in 1994
constituency1994 constituency in 1994
totalcontestants1994 total number of candidates contesting elections in 1994
totalvoters1999 total number of voters in 1999
totalcastedvote1999 total number of votes cast in 1999
totalvalidvote1999 total number of valid votes in 1999
constituency1999 constituency in 1999
totalcontestants1999 total number of candidates contesting elections in 1999

pop_2001 population in 2001
hdi_1996 HDI 1996 (index 0 to 1)

percent_regvote1991 election turnout for 1991 as measured by the percentage of registered voters who voted in the national parliamentary election
percent_regvote1994 election turnout for 1994 as measured by the percentage of registered voters who voted in the national parliamentary election
percent_regvote1999 election turnout for 1999 as measured by the percentage of registered voters who voted in the national parliamentary election
per_total_hold_sharecrop  percent of sharecropping households
per_total_hold_fixmoney  percent of households that have a fixed cash rent
per_total_hold_fixproduct  percent of households that have a fixed product rent
per_total_hold_service  percent of households that have rent for service
per_total_hold_mortgage  percent of households with a mortgage
per_noholding_below1_pa
landless_1000  landless households (in 1,000s)
totalkilled_1000  total number of people killed (in 1,000s). This serves as a measure of political violence during the insurgency
cast_eth_fract  caste and ethnic fractionalization
linguistic_fract  linguistic fractionalization
landless_gap  landless households (in 1,000s) gap
below1pa_gap  percent smallholder households gap
sharecrop_gap  percent sharecropping households gap
service_gap  percent rent for service households gap
fixmoney_gap  percent fixed cash rent households gap
fixprod_gap  percent fixed product rent households gap
hdi_gap  HDI 1996 (index 0 to 1) gap
ln_pop2001  population in 2001 (logged)
hdi_gap1  HDI 1996 (index 0 to 1) gap (positive values)

Source

Journal of Peace Research Replication Datasets

References


Examples

library(MASS)
library(modelLLtest)

# Models from Joshi and Mason (2008)
data(nepaldem)
model_1991 <- rlm(percent_regvote1991 ~ landless_gap +
  below1pa_gap + sharecrop_gap + service_gap + fixmoney_gap +
  fixprod_gap + per_without_instcredit + hdi_gap1 + ln_pop2001 +
  totalcontestants1991 + cast_eth_fract, data = nepaldem)

model_1994 <- rlm(percent_regvote1994 ~ landless_gap +
below1pa_gap + sharecrop_gap + service_gap + fixmoney_gap +
fixprod_gap + per_without_instcredit + hdi_gap1 + ln_pop2001 +
totalcontestants1994 + cast_eth_fract, data = nepaldem)

model_1999a <- rlm(percent_regvote1999 ~ landless_gap +
below1pa_gap + sharecrop_gap + service_gap + fixmoney_gap +
fixprod_gap + per_without_instcredit + hdi_gap1 + ln_pop2001 +
totalcontestants1999 + cast_eth_fract, data = nepaldem)

model_1999b <- rlm(percent_regvote1999 ~ landless_gap +
below1pa_gap + sharecrop_gap + service_gap + fixmoney_gap +
fixprod_gap + per_without_instcredit + totoalkilled_1000 +
hdi_gap1 + ln_pop2001 + totalcontestants1999 + cast_eth_fract,
data = nepaldem)

# Comparing OLS to RR fit for model_1999b
obj_cvdm_jm <- cvdm(percent_regvote1999 ~ landless_gap +
below1pa_gap + sharecrop_gap + service_gap + fixmoney_gap +
fixprod_gap + per_without_instcredit + totoalkilled_1000 +
hdi_gap1 + ln_pop2001 + totalcontestants1999 + cast_eth_fract,
data = nepaldem, method1 = "OLS", method2 = "RLM-MM")

obj_cvdm_jm
Index

*Topic **cox**
  cvmf_object, 10

*Topic **cvmf**
  cvmf_object, 10
  cvlldiff_object, 7

*Topic **cvmf_object**
  cvmf, 8, 10, 11
  cvmf_object, 10, 10

*Topic **datasets**
  govtform, 11
  nepaldem, 14

binom.test, 11

coxph, 9–11

coxr, 9–11

cvdm, 2, 4

cvdm_object, 3, 4, 5, 7

cvll, 5, 8

cvll_object, 8

cvlldiff, 6, 7, 8

cvlldiff_object, 7

cvmf, 8, 10, 11

cvmf_object, 10, 10

govtform, 11

lm, 4, 8

modeLLtest, 13
modeLLtest-package (modeLLtest), 13

nepaldem, 14

rlm, 3, 5

rq, 3, 5

Surv, 9, 10