

Package ‘ebrahim.gof’

October 1, 2026

Type Package

Title Goodness-of-Fit and Calibration Tests for Logistic Regression

Version 2.9.0

Date 2026-10-01

Maintainer Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

Description Provides a unified battery of goodness-of-fit and calibration tests for binary logistic regression, runnable in a single call via 'run.all.gof()'. Around twenty-five tests spanning five decades of literature are aggregated and grouped by the departure each is built to detect: global and standardized statistics, partition tests such as Hosmer-Lemeshow, directed and covariate-space tests, smoothing and resampling tests, and calibration tests. Each is obtained from its own package where installed and attributed to its authors. The package also implements the author's own procedures for sparse data, where the Hosmer-Lemeshow test loses power: the omnibus Ebrahim-Farrington test 'ef.gof()', the directed 'edge.gof()' and its covariate-space variant 'cdef.gof()', the Cauchy-combination ensemble 'edges.gof()', 'DeepGOF-1' (a pretrained convolutional statistic whose level comes from the analyst's own parametric bootstrap rather than from the network), and 'legoft()' (a frozen-weight combination whose weights are fixed offline and ship frozen, so two analysts obtain the same p-value). For penalized (ridge) logistic regression, where shrinkage biases the fitted probabilities and invalidates the usual chi-squared references, the corrected statistics are referred either to a pre pivoting bootstrap by 'shrink.gof()' or to a closed-form reference by 'calm.gof()', which needs a single fit and is validated for designs in which the number of predictors is a sizeable fraction of the sample size. For more details see Hosmer (1980) <doi:10.1080/03610928008827941> and Farrington (1996) <doi:10.1111/j.2517-6161.1996.tb02086.x>.

License GPL-3

URL <https://ebrahimkhaled.github.io/ebrahim.gof/>,
<https://github.com/ebrahimkhaled/ebrahim.gof>

BugReports <https://github.com/ebrahimkhaled/ebrahim.gof/issues>

Depends R (>= 3.5.0)

Imports CompQuadForm, parallel, stats

Suggests testthat (>= 3.0.0), knitr, rmarkdown, ResourceSelection,
ggplot2, statmod, mgcv, BAGofT, randomForest, dcov, givitiR,
callr, TH.data

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Ebrahim Khaled Ebrahim [aut, cre] (ORCID:

<<https://orcid.org/0009-0006-7839-8778>>),

Jiawei Zhang [ctb, cph] (author of the BAGofT package code adapted in
R/bagoft_fast.R),

Jie Ding [ctb, cph] (author of the BAGofT package code adapted in
R/bagoft_fast.R),

Yuhong Yang [ctb, cph] (author of the BAGofT package code adapted in
R/bagoft_fast.R)

Repository CRAN

Date/Publication 2026-10-01 15:40:45 UTC

Contents

ebrahim.gof-package	3
bagoft.fast	5
calm.gof	7
cdef.gof	9
deepgof1	11
deepgof1.external	13
def.ensemble.gof	15
def.gof	17
deploy.gof	20
edge.gof	21
edge.stream	23
edges.gof	25
ef.gof	26
gof.features	29
gof_demo	29
gof_demo_grouped	30
gof_install_suggests	32
legoft	33
legoft.localize	34
plot.gof_battery	35
print.gof_battery	35
projection.gof	36

run.all.external	38
run.all.gof	40
shrink.gof	49

Index	51
--------------	-----------

ebrahim.gof-package	<i>ebrahim.gof: Goodness-of-Fit and Calibration Tests for Logistic Regression</i>
---------------------	---

Description

A unified toolbox of goodness-of-fit and calibration tests for binary logistic regression, callable in a single line via [run.all.gof](#). The package is aimed particularly at *sparse* data, where the classical Hosmer–Lemeshow test loses power, and at *penalized* fits, where it is not merely weak but invalid.

Choosing a test

The battery is the place to start; the table below is for when you already know something about what you are looking for.

No prior idea of what is wrong [run.all.gof](#) — runs the whole battery and groups the results by the departure each test detects.

Sparse data, no direction in mind [ef.gof](#) — the omnibus Ebrahim–Farrington test, which groups automatically and needs no model object.

Misfit expected in the shape of the calibration curve [edge.gof](#) — spends its few degrees of freedom on the smooth directions where structured misfit concentrates.

Misfit expected in the covariates themselves [cdef.gof](#) — the covariate-space directed test.

Unwilling to choose one direction [edges.gof](#) — a Cauchy combination over the directed bases, which pays little for the directions that turn out to be empty.

Wanting one number, reproducible between analysts [legoft](#) — weights fixed offline and shipped frozen, so nothing is retrained when you call it; [legoft.localize](#) then says which domain of evidence carries the misfit, with familywise error control.

Willing to spend a bootstrap for more power [deepgof1](#) — a pretrained convolutional statistic whose level comes from your own parametric bootstrap rather than from the network; its map shows where the misfit lies.

A penalized (ridge) fit [calm.gof](#) for a closed-form reference from a single fit, or [shrink.gof](#) for the same correction referred to a prepivoting bootstrap. Under a penalty the usual chi-squared references are wrong, not just conservative.

A scorer of your own [gof.features](#) turns a fit into a feature vector and [deploy.gof](#) applies a scorer to it.

Aggregated tests (for comparison)

`run.all.gof` also runs, in one call, a wide range of classical and modern tests — Hosmer–Lemeshow, McCullagh, Osius–Rojek, le Cessie–van Houwelingen, Stute–Zhu, the binary-adaptive **BAGofT** test, and the **givitiR** calibration test. Each aggregated test is obtained from its own package (where installed) and is attributed to its authors; these are provided for head-to-head comparison, not claimed as original to this package. `gof_install_suggests` installs the optional packages they need.

Data

`gof_demo` is a bundled example dataset with a documented, reproducible misfit for illustrating the battery.

Citing the methods

Each of the author’s tests has a paper behind it. Run `citation("ebrahim.gof")` for the current references; they are kept there rather than duplicated here, because several are moving from preprint to journal.

Author(s)

Maintainer: Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg> ([ORCID](#))

Other contributors:

- Jiawei Zhang (author of the BAGofT package code adapted in R/bagoft_fast.R) [contributor, copyright holder]
- Jie Ding (author of the BAGofT package code adapted in R/bagoft_fast.R) [contributor, copyright holder]
- Yuhong Yang (author of the BAGofT package code adapted in R/bagoft_fast.R) [contributor, copyright holder]

See Also

The vignette `vignette("ebrahim-gof-toolbox", package = "ebrahim.gof")`.

Examples

```
set.seed(1)
n <- 200
x <- rnorm(n)
y <- rbinom(n, 1, plogis(0.3 + 0.9 * x))
fit <- glm(y ~ x, family = binomial())

# the omnibus test on the fitted probabilities
ef.gof(y, fitted(fit))

# the directed test, when misfit is expected in the calibration shape
edge.gof(fit)
```

bagoft.fast

*Fast BAGofT: the Binary Adaptive Goodness-of-Fit Test***Description**

`bagoft.fast()` computes the BAGofT test of Zhang, Ding and Yang (2023) for a fitted binary `glm`, and returns the same p-values as the **BAGofT** package (version 1.0.0) for the same random seed – identical, not approximately equal. It is the call

```
BAGofT::BAGofT(testGlmBi(formula, link), parRF(), data,
  nsplits = 100, ne = floor(5 * sqrt(n)), nsim = 100)
```

with the package’s own defaults, computed without its per-split overhead (formula parsing, model frames, xtabs, cut labels and the predict wrappers, repeated $(nsim + 1) \times nsplits$ times).

Usage

```
bagoft.fast(
  object,
  data = NULL,
  nsplits = 100,
  nsim = 100,
  ne = NULL,
  ntree = 60,
  Kmax = NULL,
  nmin = NULL,
  mtry = NULL,
  maxnodes = NULL
)
```

Arguments

<code>object</code>	A fitted binary <code>glm</code> (family = binomial, any link) with unit prior weights and no offset.
<code>data</code>	Optional data frame to run the test on, as in <code>BAGofT(data = ...)</code> . It must contain the response (numeric 0/1) and the model’s variables; every other column is offered to the forest. Default: the model frame of <code>object</code> .
<code>nsplits</code>	Number of random splits (default 100, the package default).
<code>nsim</code>	Number of simulated responses used to calibrate the p-value (default 100, the package default). <code>nsim = 0</code> returns the statistics only.
<code>ne</code>	Size of the held-out part of each split. Default <code>floor(5 * sqrt(n))</code> , the package default.
<code>ntree, Kmax, nmin, mtry, maxnodes</code>	The random-forest partitioner’s settings, as in <code>BAGofT::parRF()</code> : number of trees (default 60), maximum number of groups (default <code>floor(ne / nmin)</code>), minimum group size (default <code>ceiling(sqrt(ne))</code>), variables tried at each node (default 1, the value the package ends up with), and maximum number of terminal nodes (default <code>min(n - ne, 5 * ncol)</code> , <code>ncol</code> the number of covariate columns).

Details

The test. Each split fits the model on $n - n_e$ observations, grows a random forest of the Pearson residuals on the covariates, uses the forest to choose an adaptive partition of the covariate space, and computes a Hosmer-Lemeshow-type chi-squared statistic on the n_e held-out observations. The split p-values are averaged over `nsplits` splits, and the average is calibrated against `nsim` responses simulated from the full-data fit. The calibrated p-value is `p.value`; `p.value2` and `p.value3` calibrate the median and the minimum instead.

Read `p.value`, **not** `pmean` **or** `pmin`. `pmean`, `pmedian` and `pmin` are the observed *statistics* (summaries of the split p-values), not p-values, and are not uniform under the null.

What the forest sees. As in the package's default `parRF(parVar = ".")`, the forest partitions on every column of data except the response – not only the model's terms. By default data is the model frame, so the forest sees the model's variables; pass a wider data to let it look for structure in covariates the model leaves out, exactly as the package does. With more than five such columns the package's pre-selection runs first in each split (`dcPre()`): the five columns with the largest distance correlation with the residuals, computed by `dcov`, and so it does here.

Differences from the package. None in the result where the package runs. Where it does not, this function still does: with a single covariate BAGofT 1.0.0 stops (its `parRF()` drops the data frame to a vector), while here the forest simply uses the one column; and a formula with transformed terms (e.g. `log(x)`) is evaluated through the fitted model's own design matrix. Offsets and non-unit prior weights are not supported, as in the package.

Speed. The work that remains is the forests themselves: $(nsim + 1) \times nsplits = 10,100$ forests with the defaults. At $n = 200$ with two covariates the default test took 99 s against 239 s for the package, and 188 s at $n = 500$ (package about 360 s); the forests themselves are about half of it. Reduce `nsim` or `nsplits` for a quicker, noisier answer.

Value

An object of class "bagoft_fast": a list with `p.value`, `p.value2`, `p.value3` (the calibrated p-values for the mean, median and minimum split p-value), `pmean`, `pmedian`, `pmin` (the observed statistics), `simRes` (the simulated statistics), and `settings`. The elements shared with `BAGofT: :BAGofT()` have the same names and values. `singleSplit.results` is not returned.

Author(s)

The procedure and the code it is adapted from are by Jiawei Zhang, Jie Ding and Yuhong Yang (package **BAGofT**, GPL-3). The fast re-expression is by Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>.

References

Zhang J, Ding J, Yang Y (2023). "Is a classification procedure good enough? A goodness-of-fit assessment tool for classification learning." *Journal of the American Statistical Association*, 118(542), 1115–1125. doi:10.1080/01621459.2021.1979010

Zhang J, Ding J, Yang Y (2021). BAGofT: A Binary Regression Adaptive Goodness-of-Fit Test. R package version 1.0.0. <https://CRAN.R-project.org/package=BAGofT>

See Also

`run.all.gof`, where this is the engine of the row "BAGofT".

Examples

```

if (requireNamespace("randomForest", quietly = TRUE)) {
  set.seed(1)
  n <- 100
  x1 <- rnorm(n); x2 <- rnorm(n)
  y <- rbinom(n, 1, plogis(0.5 * x1 + 0.5 * x2))
  fit <- glm(y ~ x1 + x2, family = binomial())
  set.seed(2)
  r <- bagoft.fast(fit, nsplits = 20, nsim = 20) # reduced for speed
  r
  ## the same numbers from the BAGofT package, same seed:
  ## set.seed(2)
  ## BAGofT::BAGofT(BAGofT::testGlmBi(y ~ x1 + x2, link = "logit"),
  ##                 BAGofT::parRF(), data = data.frame(y, x1, x2),
  ##                 nsplits = 20, nsim = 20)$p.value
}

```

calm.gof

Closed-form goodness-of-fit test for penalized logistic regression (CALM)

Description

Refers the shrinkage-corrected grouped goodness-of-fit statistics to an analytic reference distribution, so that no bootstrap is needed. It is the companion of [shrink.gof](#), which refers the same statistics to a pre pivoting bootstrap: the statistics are identical, only the reference differs. CALM stands for Calibration Assessment under Lambda-shrunk Models.

Usage

```

calm.gof(
  X,
  y,
  lambda,
  G = 10,
  basis = c("decile", "adaptive", "edge"),
  lambda_scale = c("theory", "glmnet"),
  tau = 0.2,
  inflate = c("kappa", "none")
)

```

Arguments

X	numeric matrix or data frame of predictors, without an intercept column. Standardize the columns as you would before any ridge fit.
y	numeric or integer vector of 0/1 responses.

lambda	the ridge penalty. On the theory scale by default, that is on the scale of the log-likelihood; pass <code>lambda_scale = "glmnet"</code> to give <code>glmnet</code> 's value instead, which is λ / n .
G	number of equal-frequency groups. Default 10.
basis	which statistics to compute: any of "decile", "adaptive" and "edge". Default: all three.
lambda_scale	"theory" (default) or "glmnet".
tau	threshold of the degree rule, in (0, 1). Default 0.2. Values below 0.2 keep the cubic direction too often in the proportional regime.
inflate	whether to inflate the reference for the selection effect, "kappa" (default) or "none". On the adaptive basis the inflation changes the size by at most 0.008 in the designs examined.

Details

Under a ridge penalty the grouped standardized residuals are displaced by shrinkage. Subtracting an estimate of that displacement restores the maximum likelihood covariance to first order, but the resulting reference is conservative, because it standardizes by the Bernoulli variance at the *fitted* probabilities, which shrinkage inflates towards one quarter. CALM replaces it by a de-noised estimate of the null Bernoulli variance, obtained from the fit alone through the observable adjustments of Bellec (2025), and inflates the result for the effect of grouping on an index that depends on the response.

Two bases are returned. The EDGE basis projects the corrected residual onto orthogonal polynomials in the group-mean fitted probability, and `SC.EDGE.adaptive` chooses the degree from the data: it keeps the cubic direction when the aspect ratio is small, or when the observable index correlation $\hat{\rho}^6 \geq \tau$, and uses degree two otherwise. That basis is the one to prefer. The decile basis `SC.HL` is the shrinkage-corrected Hosmer-Lemeshow statistic and is reported for continuity with that tradition; because it spends every group direction it cannot avoid the direction the selection effect occupies, and its reference relies on a constant calibrated by simulation, which does not transfer to every design. See the reference for the designs in which it fails.

Value

An object of class "calm.gof": a list with components `SC.HL`, `SC.EDGE` and `SC.EDGE.adaptive` (each a list with `statistic` and `p.value`, and for the adaptive basis also the chosen degree and the observable `rho_hat`), together with the penalty on both scales and the observables used to build the reference.

Scope

The reference is validated for aspect ratios p/n up to 0.4 and for penalties that shrink towards zero; the lasso is not covered, because the displacement requires a differentiable penalty. The test asks whether the logistic form is correct along the fitted index. It does not ask whether the shrunk probabilities are calibrated, which under a penalty they are not, by an amount the analyst chose when selecting `lambda`.

The reference is validated for outcomes that are not strongly unbalanced. Once p/n is an appreciable fraction the level is lost as the prevalence falls: at $p/n = 0.25$ the smooth basis rejects 0.140,

0.574 and 0.884 of correctly specified models at prevalences 0.30, 0.15 and 0.08, and the decile basis 0.060, 0.204 and 0.492. At fixed dimension the decile basis is unaffected and the smooth one degrades far more slowly, to 0.130 at prevalence 0.08. What fails there is the response-dependent grouping inherited from the Hosmer-Lemeshow construction rather than the reference itself. With few events and p/n an appreciable fraction, neither statistic is validated and [shrink.gof](#) is the less badly behaved of the two.

References

- Bellec, P. C. (2025). Observable adjustments in single-index models for regularized M-estimators with bounded p/n . *The Annals of Statistics*, **53**(2), 531–560.
- Davies, R. B. (1980). Algorithm AS 155: the distribution of a linear combination of chi-squared random variables. *Journal of the Royal Statistical Society, Series C*, **29**(3), 323–333.
- Ebrahim, E. K. (2026). A closed-form reference distribution for goodness-of-fit testing under penalized logistic regression in the proportional regime.

See Also

[shrink.gof](#) for the bootstrap reference for the same statistics, and [edge.gof](#) for the EDGE test on an unpenalized fit.

Examples

```
set.seed(1)
n <- 300; p <- 20
X <- matrix(rnorm(n * p), n)
y <- rbinom(n, 1, 1 / (1 + exp(-(X[, 1] - 0.5 * X[, 2]))))
calm.gof(X, y, lambda = 100)
```

cdef.gof

Covariate-Space Directed Ebrahim-Farrington (CDEF) Goodness-of-Fit Test

Description

A directed goodness-of-fit test for binary logistic regression whose direction lives in *covariate space* (functions of the predictors) rather than in fitted-probability space like [def.gof](#). It projects the standardized residuals onto a covariate-space basis (polynomials and pairwise products, natural splines, or a combination that also includes fitted-probability bends) and calibrates the quadratic form with the Farrington estimation-adjusted projection, exactly as in [def.gof](#). This makes it sensitive to omitted interactions and to local / oscillatory departures that fitted-probability grouping can miss.

Usage

```
cdef.gof(
  object,
  predicted_probs = NULL,
  X = NULL,
  basis = c("poly", "spline", "combined"),
  method = c("satterthwaite", "imhof")
)
```

Arguments

object	A fitted binary logistic glm , or a binary (0/1) response vector y (then supply predicted_probs and X).
predicted_probs	Numeric predicted probabilities; required when object is a y vector.
X	Design/covariate matrix (with or without an intercept column); required when object is a y vector. Ignored when object is a glm.
basis	One of "poly" (squares, cubes, pairwise products), "spline" (natural cubic splines per covariate plus a pairwise term; needs splines), or "combined" (covariate polynomials plus fitted-probability bends).
method	One of "satterthwaite" (default) or "imhof".

Details

Let $\tilde{r}_i = (y_i - \hat{p}_i) / \sqrt{\hat{p}_i(1 - \hat{p}_i)}$ be the standardized residuals and Z a covariate-space basis matrix. The statistic is $S = (Z'\tilde{r})'(Z'Z)^{-1}(Z'\tilde{r})$, whose null distribution is a weighted sum of χ_1^2 variables with weights the eigenvalues of $(Z'Z)^{-1}Z'\Omega Z$, where $\Omega = I - V^{1/2}X(X'VX)^{-1}X'V^{1/2}$ adjusts for estimating $\hat{\beta}$. The p-value uses a Satterthwaite scaled- χ^2 approximation (default) or Imhof's method (**CompQuadForm**). Rank-deficient bases are reduced automatically.

Value

A one-row data.frame with Test, Basis, Test_Statistic, df, Method, and p_value.

References

Farrington, C. P. (1996). On Assessing Goodness of Fit of Generalized Linear Models to Sparse Data. *JRSS-B* 58(2), 349-360.

See Also

[def.gof](#), [ef.gof](#), [run.all.gof](#).

Examples

```
set.seed(1)
n <- 600; x1 <- runif(n, -3, 3); x2 <- rnorm(n)
# truth has an omitted interaction; fit the additive model
```

```

y <- rbinom(n, 1, plogis(0.3 + 0.8 * x1 - 0.5 * x2 + 0.4 * x1 * x2))
fit <- glm(y ~ x1 + x2, family = binomial())
cdef.gof(fit) # covariate-space directed test (poly basis)
cdef.gof(fit, basis = "spline") # for local / oscillatory misfit

```

deepgof1

DeepGOF-1: a pretrained goodness-of-fit test for logistic regression

Description

Tests whether a fitted binomial glm is correctly specified, using a convolutional network that was trained once, offline, on simulated departures and is shipped frozen with this package. The analyst never trains anything: the network reads the model's residual map and the p-value is the rank of the observed score within the analyst's own parametric bootstrap, so the level does not depend on what the network learned.

Usage

```

deepgof1(
  fit,
  B = 199L,
  K = 6L,
  reading = c("axes", "allpairs", "combined", "columns"),
  covariates = NULL
)

```

Arguments

fit	a fitted glm with family = binomial(), a 0/1 outcome and at least one covariate.
B	number of parametric-bootstrap replicates. The p-value lies on a grid of $1/(B+1)$, so B = 199 makes the nominal .05 attainable exactly.
K	grid resolution. Leave at 6: the shipped weights were trained at K = 6 and are not valid at any other resolution.
reading	"axes" (the default), "allpairs", "combined" or "columns" (the rule of versions 2.7.0 and 2.8.0). See Details for when to use which.
covariates	optional character vector naming the covariates to form the pairs from, for the all-pairs and combined readings. By default every covariate of the model that is not constant is used.

Details

A residual map is a K by K grid over the empirical ranks of two covariates; each cell holds a standardized residual sum, approximately standard normal under a correct model. Misfit therefore has a location on the map – an omitted quadratic paints a stripe, an omitted interaction a saddle – which is what the convolutional statistic reads.

Four readings of the map are offered. The default, `reading = "axes"`, draws one map, over the two covariates whose terms contribute most to the linear predictor (the standard deviation of each covariate's total contribution; for a covariate that enters as one untransformed column this is $|\hat{\beta}_j| \hat{\sigma}_j$), and chooses them again in every bootstrap replicate. A covariate that enters as `ns(x, 3)`, `poly(x, 2)`, `I(x^2)` or a factor is one covariate, and the map is drawn over the ranks of `x` itself. When the misfit lies on the covariates with the strongest effects, among others that carry little signal, this rule finds them almost every time and has the most power. It reads a covariate by its effect in the fitted model, so it can pass over a covariate whose effect is a pure U-shape with no linear slope.

`reading = "allpairs"` scores the map of every pair of covariates and takes the largest score as the statistic. The same maximum is taken in every bootstrap replicate, so the p-value needs no correction for the choice of pair. It does not depend on the fitted effects, so it finds a U-shaped covariate, and with few covariates (three or so) it has more power than the default; with many covariates the maximum over $p(p-1)/2$ pairs costs power when one pair carries the misfit. The pair that reaches the maximum is returned as `axes`, and its map as `map`, so the result also says where the misfit lies; `covariates` restricts the pairs to a chosen set.

`reading = "combined"` runs both from the same bootstrap refits and reports the smaller of their two p-values, calibrated exactly: the observed data and the `B` replicates are exchangeable under the null, so the observed minimum is ranked among the `B + 1` minima. It costs no more than the all-pairs reading. `reading = "columns"` is the rule of versions 2.7.0 and 2.8.0, over two model-matrix columns, kept to reproduce earlier results; for models whose covariates all enter as one untransformed column it gives the same p-value as the default.

Transformed covariates are read from the data the model was fitted to; factors enter by their level codes.

With one covariate there is no pair to choose, and every reading is the same map of 36 quantile cells along its ranks. The shipped network was trained on two-covariate maps only: on such models the bootstrap still gives it its level, but it has less power than a network trained on this map would.

Ties among covariate values, as with binary, categorical or rounded covariates, are broken at random. The random order is drawn once per call and used for the observed map and for every bootstrap map, so the p-value does not depend on the order of the rows in the data. With heavily tied axes the p-value can vary noticeably from one seed to the next; report the seed.

The test is a small-sample instrument. Against the classical partition tests it gains most at n of 50 to 200 and the gain decays as n grows; because each map uses two covariates at a time, misfit that depends on three or more covariates jointly is harder for it to see than for covariate-space or smoothing tests. It is not one of the tests `run.all.gof` selects: call it directly on the same fitted model and read its p-value beside the panel. See `run.all.gof` for the classical battery.

Value

An object of class `"deepgof1"`: a list with `statistic` (the observed score), `p.value`, `B`, `K`, `reading`, `axes` (the covariates of the map that gave the statistic), `map` (that map, a `K` by `K` matrix of standardized residual sums whose rows follow the first axis), `pairs` (for the all-pairs and combined readings, the observed score of every pair), `components` (for `"combined"`, the p-values of the axis rule and the all-pairs reading), `boot` (the `B` bootstrap statistics of the reported statistic) and `method`.

Reproducibility

A bootstrap refit that fails to converge is scored +Inf, so it counts against rejection – the conservative direction. Set a seed before calling for a reproducible p-value. When some covariate column has tied values, the random tie-breaking uses the same seed, so from version 2.8.0 such calls give a different p-value for a given seed than earlier versions did; calls without ties give the same p-value as before. Version 2.9.0 refits each bootstrap sample on the fitted model's design matrix. Earlier versions refitted the formula on the model frame, which fails for every term that transforms a covariate ($\log(x)$, $\text{ns}(x, 3)$, $\text{poly}(x, 2)$): each replicate was then scored +Inf and the p-value was 1. For models without such terms the default reading gives the same p-value as in 2.8.0.

References

Ebrahim EK (2026). "DeepGOF-1: A Pretrained Convolutional Goodness-of-Fit Test for Logistic Regression with a Computable Consistency Certificate." Manuscript under review. Reproduction materials and frozen weights: [doi:10.5281/zenodo.22113220](https://doi.org/10.5281/zenodo.22113220)

Besag, J. and Clifford, P. (1989). Generalized Monte Carlo significance tests. *Biometrika* **76**, 633–642. [doi:10.1093/biomet/76.4.633](https://doi.org/10.1093/biomet/76.4.633)

See Also

[run.all.gof](#), [ef.gof](#), [legoft](#)

Examples

```
set.seed(1)
n <- 150
x1 <- runif(n, -3, 3); x2 <- rnorm(n)
# a model with an omitted quadratic term
y <- rbinom(n, 1, plogis(0.3 + 0.8 * x1 - 0.5 * x2 + 0.9 * (x1^2 - mean(x1^2))))
fit <- glm(y ~ x1 + x2, family = binomial())
deepgof1(fit, B = 49) # B = 49 to keep the example fast; use the default in practice
# every pair of covariates, with the pair where the misfit is largest
deepgof1(fit, B = 49, reading = "allpairs")$axes
```

deepgof1.external

DeepGOF-1 for frozen predictions: external validation of a risk model

Description

Tests whether given probabilities are calibrated for given 0/1 outcomes within subgroups of the covariates, for a model that was fitted elsewhere and is not refitted: a published risk score checked on new patients, or the predictions of any model, logistic or not, on a validation set. The null hypothesis is that each y_i is Bernoulli(p_i) with p_i as given.

Usage

```

deepgof1.external(
  y,
  p,
  X,
  B = 199L,
  K = 6L,
  reading = c("combined", "axes", "allpairs"),
  axes = NULL
)

```

Arguments

y	0/1 outcomes.
p	the predicted probabilities to be checked, strictly between 0 and 1, one per outcome.
X	a numeric matrix or data frame of covariates, one row per outcome, with column names. At least one column.
B	number of Monte Carlo draws; the p-value lies on a grid of $1 / (B + 1)$.
K	grid resolution; leave at 6, the resolution the shipped weights were trained at.
reading	"combined" (the default), "axes" or "allpairs", as in deepgof1 . The combined reading is the exact minimum of the other two, calibrated over the same draws.
axes	optional names of two columns of X for the axis-rule map, fixing it in advance.

Details

The residual map of [deepgof1](#) is drawn over the ranks of the covariates and standardized by the given probabilities, and the shipped network scores it. The reference distribution is built by drawing $y^* \sim \text{Bernoulli}(p)$ and scoring again; with nothing estimated, the law of the score is the same under the whole null, so the rank p-value is exactly valid at every sample size (Besag and Clifford 1989).

Because the map lays the residuals out over the covariates, the test checks calibration within patient subgroups (strong calibration in the hierarchy of Van Calster et al. 2016). A wrong overall rate or a wrong calibration slope is the same for every patient and is found better by tests along the predicted risk, such as the calibration belt ([run.all.gof](#) with GiViTI); a miscalibration that differs between patients with the same predicted risk, a missed U-shape, threshold or interaction, is found better by this test, which also shows where it lies.

The axis rule needs a ranking of the covariates by their effect. Without coefficients, it regresses the logit of p on the covariates by least squares and scores each covariate by $|b_j|\hat{\sigma}_j$; for a published logistic model in these covariates this recovers its own coefficients exactly. Name the axes with axes to fix them in advance.

Value

An object of class "deepgof1", as returned by [deepgof1](#).

References

Besag, J. and Clifford, P. (1989). Generalized Monte Carlo significance tests. *Biometrika*, 76(4), 633–642.

Van Calster, B., Nieboer, D., Vergouwe, Y., De Cock, B., Pencina, M. J. and Steyerberg, E. W. (2016). A calibration hierarchy for risk models was defined: from utopia to empirical data. *Journal of Clinical Epidemiology*, 74, 167–176.

See Also

[deepgof1](#)

Examples

```
set.seed(1)
n <- 400
X <- data.frame(x1 = rnorm(n), x2 = rnorm(n), x3 = rnorm(n))
p <- plogis(-0.5 + 0.8 * X$x1 + 0.6 * X$x2) # the published model
y <- rbinom(n, 1, plogis(qlogis(p) + 0.8 * X$x1 * X$x2)) # the new patients
deepgof1.external(y, p, X, B = 99)
```

def.ensemble.gof	<i>Combine Directed GOF Tests into One Decision (Ensemble)</i>
------------------	--

Description

Combines the three Directed Ebrahim-Farrington (DEF) basis tests ("poly2", "poly3", "stukel") into a single goodness-of-fit decision, so the user does not have to choose a basis. By default the p-values are combined with the Cauchy Combination Test (CCT), which controls the error rate under the strong dependence between tests computed on the same fitted model. The omnibus EF test can optionally be added to the vote.

Usage

```
def.ensemble.gof(
  object,
  predicted_probs = NULL,
  X = NULL,
  components = c("poly2", "poly3", "stukel"),
  add_ef = FALSE,
  combine = c("cct", "minp", "fisher"),
  G = 10,
  extra_pvalues = NULL,
  weights = c("unit", "score")
)
```

Arguments

object	A fitted binary logistic glm , or a binary (0/1) vector y (then supply predicted_probs).
predicted_probs	Numeric predicted probabilities; required when object is a y vector.
X	Optional design matrix, threaded to def.gof for the exact calibration (only used with the y/predicted_probs form).
components	Character vector, a subset of c("poly2", "poly3", "stukel", "sym"). Default is "poly2", "poly3" and "stukel", the three bases EDGES is defined on.
add_ef	Logical; if TRUE, the omnibus EF p-value (ef.gof) is appended to the components. Default FALSE.
combine	One of "cct" (default), "minp", "fisher".
G	Integer number of groups passed to def.gof/ef.gof (default 10), or "auto" for <code>max(10, ceiling(n / 25))</code> as in def.gof .
extra_pvalues	Optional named numeric vector of additional p-values to include (e.g. a Tsatis test computed elsewhere). Default NULL.
weights	"unit" (default) or "score", passed to def.gof for every component.

Details

Because the component tests are computed on the same fit, their p-values are strongly dependent. The CCT (combine = "cct") has an asymptotic standard-Cauchy null whose tail is robust to this dependence, so it needs no calibration. The "minp" (Sidak) and "fisher" rules assume independence and are offered for comparison only; under positive dependence "minp" is conservative and "fisher" is anti-conservative, so they should be calibrated by simulation before use (not done here).

With no event, or no non-event, the model has no maximum-likelihood fit: the p-value is NA, with one warning of class `def_degenerate`.

Value

A one-row `data.frame` with columns Test, Combiner, Components, k, and p_value.

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

Liu, Y. and Xie, J. (2020). Cauchy combination test. *JASA*, 115(529), 393-402.

See Also

[def.gof](#), [ef.gof](#).

Examples

```
data("gof_demo", package = "ebrahim.gof")
wrong <- glm(outcome ~ age + bmi + sex + treatment,
             data = gof_demo, family = binomial())
def.ensemble.gof(wrong)           # CCT of the three DEF bases
def.ensemble.gof(wrong, add_ef = TRUE) # add the omnibus EF

## the corrected model, for contrast
right <- glm(outcome ~ poly(age, 2) + bmi + sex + treatment,
             data = gof_demo, family = binomial())
def.ensemble.gof(right)
```

def.gof

Directed Ebrahim-Farrington (DEF) Goodness-of-Fit Test

Description

Performs the Directed Ebrahim-Farrington (DEF) goodness-of-fit test for a fitted binary logistic regression model. DEF concentrates its power on a small set of calibration-curve "shape" directions by projecting the grouped standardized residuals onto a low-dimensional basis and testing the squared length of that projection.

Naming note: this test is published under the name **EDGE** (Efficient Directed Grouped Examination), and [edge.gof](#) is the primary interface going forward. `def.gof()` is retained, unchanged, as a fully supported legacy name.

Usage

```
def.gof(
  object,
  predicted_probs = NULL,
  X = NULL,
  G = 10,
  basis = c("poly3", "poly2", "stukel", "sym", "ensemble"),
  method = c("satterthwaite", "imhof"),
  weights = c("unit", "score"),
  external = FALSE
)
```

Arguments

<code>object</code>	A fitted binary logistic glm , or a binary (0/1) response vector <code>y</code> (then supply <code>predicted_probs</code>).
<code>predicted_probs</code>	Numeric predicted probabilities; required when <code>object</code> is a <code>y</code> vector, ignored when it is a <code>glm</code> .

X	Optional design matrix, used only with the y/predicted_probs form: it enables the exact estimation-adjusted (Ω) calibration (logit working weights assumed). Without it the conservative χ_k^2 reference is used and a warning is issued. Ignored when object is a glm, and ignored (with a warning) when external = TRUE.
G	Integer number of equal-frequency groups (default 10; must be ≥ 3), or "auto" for $\max(10, \text{ceiling}(n / 25))$, the partition rule of the EDGE paper.
basis	One of "poly3" (default), "poly2", "stukel", "sym", or "ensemble". "sym" is one column, $\eta \eta $ at the logit η of each group's mean fitted risk: Stukel's (1988) symmetric direction, aimed at tails that are too heavy or too light on both sides, for example a probit or cauchit truth fitted by a logit.
method	One of "satterthwaite" (default) or "imhof". Ignored when weights = "score".
weights	"unit" (default) is the statistic as published, $S = r' P_Z r$ referred to a weighted chi-squared law. "score" multiplies each column by the square root of its group's variance, which for a logit fit makes the statistic the score test for adding the grouped shape to the model (a score-type test for other links). It is referred to chi-squared on the rank of its information matrix, which is the number of columns unless one is redundant (see Details).
external	Logical, default FALSE. TRUE treats the predicted probabilities as frozen (external validation of a fixed model): $\Omega = I$, a constant column joins the basis, and the statistic is referred to χ^2 on the number of basis columns (see Details of def.gof). Supply y and predicted_probs, or a glm whose fitted probabilities are then taken as frozen. Requires weights = "unit" and a basis other than "ensemble"; method is not used.

Details

The observations are sorted by predicted probability and split into G equal-frequency groups; the standardized grouped residual vector r is projected onto a basis matrix Z of smooth shapes, giving $S = (Z'r)'(Z'Z)^{-1}(Z'r)$. Its null distribution is a weighted sum of χ_1^2 variables with weights equal to the eigenvalues of $(Z'Z)^{-1}Z'\Omega Z$, where $\Omega = I - U(X'WX)^{-1}U'$ is the estimation-adjusted covariance of the grouped residuals. The p-value uses a Satterthwaite scaled- χ^2 approximation (default) or Imhof's method (if the **CompQuadForm** package is installed). Bases: "poly2", "poly3" (default), "stukel", "sym"; "ensemble" runs "poly2", "poly3" and "stukel" and combines them via [def.ensemble.gof](#).

Equal-frequency groups split tied fitted risks by row order. With many ties, as with grouped data or a model on discrete covariates, the result can therefore depend on the order of the rows, and randomising the row order is advised.

With weights = "score" each column of Z is multiplied by $\sqrt{V_g}$, the square root of its group's variance, so that $Z'r$ becomes $\sum_g z_g(O_g - E_g)$: for a logit fit, the score for adding the grouped shape to the model as a step covariate. Its information after adjusting for the fitted coefficients is $Z'\Omega Z$, and the statistic $u'I^{-1}u$ is referred to a χ^2 law on the rank of that information (the number of columns unless one is redundant), read from it after scaling to a correlation matrix. For a logit fit this is the Rao score test for adding the grouped columns, and it agrees with `anova(..., test = "Rao")` up to glm's convergence tolerance; for other links it is a score-type test. A column whose information after the fit is below 10^{-10} times its information before the fit ($Z'_s Z_s$, with Z_s the weighted columns) is one the model already spans, as when the fitted logit is constant. It is left out, and when no column is left the p-value is NA, with a warning of class `def_no_information`. The

unit form is the statistic as published; the score form keeps a shape on the logit scale from losing its signal when the group variances differ strongly, as they do at high discrimination.

External mode. With `external = TRUE` the predicted probabilities are taken as frozen, as when a published model is checked on new data with its coefficients fixed. Nothing is estimated from these data, so $\Omega = I$ exactly, and no score equation absorbs the overall level, so a column of ones is added to the basis before redundant columns are dropped. The statistic $S = r'P_Zr$ is then referred to χ^2 on the number of columns kept: $d + 1$ for a d -column basis (4 for "poly3" and "stukel", 3 for "poly2", 2 for "sym"; one fewer for "stukel" when every group lies on one side of 0.5). The groups are the same equal-frequency groups as in the default mode. No "conservative" warning is given, since $\Omega = I$ is exact here, and Method is "external". Only `weights = "unit"` is available: the score form is a different projection, and it has not been validated for frozen predictions. When object is a glm, its response and fitted probabilities are used as the frozen predictions; the fit is not otherwise used, so this is a test of those predictions and not the estimation-adjusted test of the model.

With fewer events (or fewer non-events) than groups, the grouped reference distribution is unreliable. The p-value is still returned, with a warning; a smaller G avoids it. With no event, or no non-event, the model has no maximum-likelihood fit, and the p-value is NA, with a warning of class `def_degenerate`.

Value

A one-row data.frame with columns `Test`, `Basis`, `Test_Statistic` (the statistic S), `df`, `Method`, and `p_value`. For `weights = "score"`, Method is "score" and `df` is the integer rank the statistic is referred to. For `external = TRUE`, Method is "external" and `df` is the integer number of basis columns, constant included. When `basis = "ensemble"`, the return is that of `def.ensemble.gof`.

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

- Ebrahim EK, El-Kotory A (2026). "A Directional Hosmer-Lemeshow Goodness-of-Fit Test for Sparse Logistic Regression." arXiv:2607.15454 [stat.ME]. doi:10.48550/arXiv.2607.15454
- Ebrahim EK, El-Kotory A (2026). "EDGE: A Closed-Form Directed Goodness-of-Fit Test for Sparse Logistic Regression." arXiv:2608.20511 [stat.ME]. doi:10.48550/arXiv.2608.20511

See Also

`ef.gof`, `def.ensemble.gof`.

Examples

```
## gof_demo carries a documented smooth calibration misfit: the risk bends in age,
## and a model linear in age misses it. The point of a directed test is to see that.
data("gof_demo", package = "ebrahim.gof")
wrong <- glm(outcome ~ age + bmi + sex + treatment,
             data = gof_demo, family = binomial())
def.gof(wrong) # default poly3 basis
```

```

def.gof(wrong, basis = "stukel")    # tail-shape basis
def.gof(wrong, basis = "sym")      # symmetric tail direction, one column
def.gof(wrong, weights = "score")  # score form of the poly3 basis
def.gof(wrong, basis = "ensemble") # combine poly2, poly3 and stukel (CCT)

## give the model the term it was missing, and the same test stands down
right <- glm(outcome ~ poly(age, 2) + bmi + sex + treatment,
             data = gof_demo, family = binomial())
def.gof(right)

## external validation: freeze the model fitted on one half, test it on the other
dev <- gof_demo[1:250, ]; val <- gof_demo[-(1:250), ]
frozen <- glm(outcome ~ age + bmi + sex + treatment, data = dev, family = binomial())
p_val <- predict(frozen, newdata = val, type = "response")
def.gof(val$outcome, predicted_probs = p_val, external = TRUE)

```

deploy.gof

Deployable learned-ensemble GOF test via parametric bootstrap

Description

Turns a pre-trained ensemble meta into a deployable goodness-of-fit test for *any* fitted model: it scores the model, then calibrates the p-value by a per-dataset parametric bootstrap from the fitted model (so no knowledge of the truth or the data-generating design is required). Validity comes from the bootstrap, independent of how meta was trained.

Usage

```
deploy.gof(object, meta, B = 99, feature_fn = gof.features)
```

Arguments

object	A fitted binary logistic glm .
meta	A pre-trained scorer: either a function <code>f(features)</code> returning a scalar misfit score, or an object with a <code>predict</code> method consuming a one-row feature matrix.
B	Number of parametric-bootstrap resamples (default 99).
feature_fn	Function mapping a fitted glm to its feature vector (default gof.features).

Value

A one-row data.frame with the score, B, and the bootstrap p_value.

See Also

[gof.features](#), [cdef.gof](#).

edge.gof

*EDGE: Directed Goodness-of-Fit Test for Binary Logistic Regression***Description**

`edge.gof()` is the primary interface to the EDGE test (Efficient Directed Grouped Examination): a grouped, *directed* goodness-of-fit test for binary logistic regression under sparse data. EDGE projects the grouped standardized residuals onto a small pre-specified basis of calibration shapes (cubic "poly3" by default) and refers the resulting quadratic form to its closed-form weighted chi-squared null distribution – no refit, no resampling, no tuning.

`edge.gof()` computes exactly the same statistic as the legacy name `def.gof` (retained for backward compatibility); the returned Test label is "EDGE".

Usage

```
edge.gof(
  object,
  predicted_probs = NULL,
  X = NULL,
  G = c("auto", 10),
  basis = "poly3",
  method = "satterthwaite",
  weights = "unit",
  external = FALSE,
  y = NULL
)
```

Arguments

object	A fitted binary logistic glm , or a binary (0/1) response vector y (then supply predicted_probs).
predicted_probs	Numeric predicted probabilities; required when object is a y vector, ignored when it is a glm.
X	Optional design matrix, used only with the y/predicted_probs form: it enables the exact estimation-adjusted (Ω) calibration (logit working weights assumed). Without it the conservative χ_k^2 reference is used and a warning is issued. Ignored when object is a glm, and ignored (with a warning) when external = TRUE.
G	Number of groups: "auto", a number, or a vector of them; the default <code>c("auto", 10)</code> reports both partitions.
basis	One of "poly3" (default), "poly2", "stukel", "sym", or "ensemble". "sym" is one column, $\eta/ \eta $ at the logit η of each group's mean fitted risk: Stukel's (1988) symmetric direction, aimed at tails that are too heavy or too light on both sides, for example a probit or cauchit truth fitted by a logit.
method	One of "satterthwaite" (default) or "imhof". Ignored when weights = "score".

weights	"unit" (default) is the statistic as published, $S = r'P_Z r$ referred to a weighted chi-squared law. "score" multiplies each column by the square root of its group's variance, which for a logit fit makes the statistic the score test for adding the grouped shape to the model (a score-type test for other links). It is referred to chi-squared on the rank of its information matrix, which is the number of columns unless one is redundant (see Details).
external	Logical, default FALSE. TRUE treats the predicted probabilities as frozen (external validation of a fixed model): $\Omega = I$, a constant column joins the basis, and the statistic is referred to χ^2 on the number of basis columns (see Details of def.gof). Supply y and predicted_probs, or a glm whose fitted probabilities are then taken as frozen. Requires weights = "unit" and a basis other than "ensemble"; method is not used.
y	Optional alias for object when frozen predictions are tested: <code>edge.gof(y = y, predicted_probs = p, external = TRUE)</code> .

Details

Two partitions, side by side. By default the test is reported at two partitions, as the EDGE paper recommends. The *default partition*, $G = \text{"auto"}$ ($\max(10, \lceil n/25 \rceil)$ groups), refines with the sample and has more power, above all against misfit at the extremes of risk. *Ten groups*, $G = 10$, keep many records in the extreme groups and so tolerate more corrupted records, at some cost in power. Choose which one decides in the analysis plan, from what is known about how the data were collected, not after seeing the results. The first row is marked `Role = "verdict"` and the second `Role = "check"`: with the default G the default partition decides and ten groups are the robustness check; to let ten groups decide, give $G = c(10, \text{"auto"})$. If only the default partition rejects, the signal sits in the extreme groups: check those records. Give a single G to get one row.

Value

A data.frame with one row per partition and columns `Test ("EDGE")`, `Partition`, `Role ("verdict" for the first row, "check" for the second)`, `Basis`, `Test_Statistic`, `df`, `Method` and `p_value`, as documented in [def.gof](#).

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

- Ebrahim EK, El-Kotory A (2026). "EDGE: A Closed-Form Directed Goodness-of-Fit Test for Sparse Logistic Regression." arXiv:2608.20511 [stat.ME]. doi:10.48550/arXiv.2608.20511
- Ebrahim EK, El-Kotory A (2026). "A Directional Hosmer-Lemeshow Goodness-of-Fit Test for Sparse Logistic Regression." arXiv:2607.15454 [stat.ME]. doi:10.48550/arXiv.2607.15454

See Also

[def.gof](#) (legacy name), [ef.gof](#), [def.ensemble.gof](#), [run.all.gof](#).

Examples

```
set.seed(1)
x <- runif(500, -3, 3)
y <- rbinom(500, 1, plogis(0.6 * x))
fit <- glm(y ~ x, family = binomial())
edge.gof(fit) # cubic basis, at the default partition and at ten groups
edge.gof(fit, G = 10) # one partition only
edge.gof(fit, basis = "stukel") # Stukel-shape basis
edge.gof(fit, basis = "sym") # Stukel's symmetric direction, one column
edge.gof(fit, basis = "sym", weights = "score") # its score form
edge.gof(fit, G = "auto") # the default partition alone: max(10, ceiling(n / 25)) = 20 here
```

edge.stream	<i>EDGE for Streaming Data: Calibration Monitoring Without Recomputation</i>
-------------	--

Description

Keeps the directed test of a frozen model up to date as new patients arrive, without revisiting the records already seen. The external-mode statistic of [def.gof](#) depends on the data only through four sums per risk group – observed events, expected events, the binomial variance and the count – so each new record updates one group in constant time, and the test is recomputed from the G group summaries alone.

Usage

```
edge.stream(p_ref = NULL, breaks = NULL, G = 10, basis = "poly3")

## S3 method for class 'edge_stream'
update(object, y, p, ...)

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

## S3 method for class 'edge_stream'
print(x, ...)
```

Arguments

p_ref	Optional numeric vector of reference predicted probabilities; the cut points are its G -quantiles.
breaks	Optional increasing numeric vector of interior cut points in $(0, 1)$ ($G - 1$ of them); used instead of p_ref.
G	Number of risk groups (default 10).
basis	Calibration basis: "poly3" (default), "poly2", "stukel" or "sym", as in def.gof .

object	An edge_stream object.
y	Binary (0/1) outcomes of the new records.
p	Predicted probabilities of the new records, made without their outcomes.
...	Unused.
x	An edge_stream object.

Details

The partition. The groups are fixed in advance by cut points on the predicted risk: either given as breaks, or the G -quantiles of a reference set of predictions p_{ref} (for example the development data, or the first batch). Because the cut points depend on predictions only and never on outcomes, the statistic keeps its χ^2_{d+1} reference under a calibrated model however the risk distribution of later patients drifts: the groups need not stay of equal size. Small groups weaken the protection against corrupted records that equal-frequency groups give, so `summary()` reports the smallest group.

Exactness. After any sequence of updates the statistic equals the one-shot external statistic computed on all records seen, with the same partition: the sums are additive, so the order and the batching of the updates do not matter.

Repeated looks. One test at any time is valid. Testing after every batch and acting on the first rejection is a sequential procedure, and the plain chi-squared reference does not control its overall false-alarm rate; spend the level across looks (for example, Bonferroni over a planned number of looks) or test at fixed times.

Value

An object of class `edge_stream`. Add data with `update(object, y, p)`; read the test with `summary(object)`, a one-row data.frame in the format of `def.gof(..., external = TRUE)` with the number of records and the smallest group added.

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

See Also

[def.gof](#), [run.all.external](#).

Examples

```
set.seed(1)
p_dev <- plogis(rnorm(5000, -1.5, 1))           # predictions on the development data
s <- edge.stream(p_ref = p_dev, G = 10)
for (day in 1:30) {                             # a deployed model, 100 patients a day
  p <- plogis(rnorm(100, -1.5, 1))
  y <- rbinom(100, 1, p)
  s <- update(s, y, p)
}
summary(s)
```

edges.gof	<i>EDGES: Cauchy-Combination Ensemble of Directed GOF Tests (Alias)</i>
-----------	---

Description

Alias for `def.ensemble.gof()`; see the EDGES paper. `edges.gof()` is the brand name (EDGES = the Cauchy-combination ensemble of the EDGE directed bases) used in the manuscript. It takes exactly the same arguments as `def.ensemble.gof` and returns exactly the same value; the legacy name `def.ensemble.gof()` is retained unchanged for back-compatibility.

Usage

```
edges.gof(...)
```

Arguments

... Arguments passed on to `def.ensemble.gof` (e.g. `object`, `predicted_probs`, `X`, `components`, `add_ef`, `combine`, `G`, `extra_pvalues`, `weights`).

Value

A one-row `data.frame` with columns `Test`, `Combiner`, `Components`, `k`, and `p_value`.

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

Liu, Y. and Xie, J. (2020). Cauchy combination test. *JASA*, 115(529), 393-402.

See Also

`def.ensemble.gof` (legacy name), `edge.gof`, `def.gof`.

Examples

```
set.seed(1)
x <- runif(500, -3, 3)
y <- rbinom(500, 1, plogis(0.6 * x))
fit <- glm(y ~ x, family = binomial())
edges.gof(fit)           # identical to def.ensemble.gof(fit)
```

ef.gof

Ebrahim-Farrington Goodness-of-Fit Test for Logistic Regression

Description

Performs the Ebrahim-Farrington goodness-of-fit test for logistic regression models. This test is particularly effective for binary data and sparse datasets, providing an improved alternative to the traditional Hosmer-Lemeshow test.

Usage

```
ef.gof(
  y,
  predicted_probs = NULL,
  model = NULL,
  m = NULL,
  G = 10,
  method = c("chisq", "normal")
)
```

Arguments

y	A fitted binary logistic glm (then predicted_probs is taken from it automatically), or a numeric vector of binary responses (0/1) for binary data / counts of successes for grouped data.
predicted_probs	Numeric vector of predicted probabilities from the logistic regression model. Must be same length as y.
model	Optional glm object. Required only for the original Farrington test with grouped data (when m is provided and G is NULL).
m	Optional numeric vector of trial counts for each observation (for grouped data). If NULL, data is assumed to be binary.
G	Optional integer specifying the number of groups for binary data grouping. Default is 10. If NULL, no grouping is performed and m must be provided.
method	Reference distribution for the grouped EF statistic: "chisq" (default) refers T_{EF} to a χ^2_{G-2} distribution; "normal" uses the standardized Z_{EF} (the behaviour of package versions $\leq 1.0.0$).

Details

The Ebrahim-Farrington test is based on Farrington's (1996) theoretical framework but simplified for practical implementation with binary data. The test uses a modified Pearson chi-square statistic with data-dependent grouping, where observations are grouped by their predicted probabilities.

For binary data (when G is specified), the test automatically groups observations into G groups based on predicted probabilities and applies the simplified Ebrahim-Farrington statistic:

$$Z_{EF} = \frac{T_{EF} - (G - 2)}{\sqrt{2(G - 2)}}$$

where T_{EF} is the modified Pearson chi-square statistic, and G is the number of groups.

For grouped data (when m is provided), the test applies the original Farrington test with full variance calculations.

Value

A data frame with the following columns:

Test	Character string identifying the test performed
Test_Statistic	Numeric value of the standardized test statistic
p_value	Numeric p-value for the test

Note

- For binary data with automatic grouping (G specified): Use the Ebrahim-Farrington test which is computationally efficient and doesn't require the model specification.
- For grouped data (m provided *and* G set to NULL): Use the original Farrington test, which requires the fitted model object. Leaving G at its default keeps the automatic grouping, and m and `model` are then ignored.
- The test statistic follows a standard normal distribution under the null hypothesis of adequate model fit.
- For binary data with $m=1$ for all observations and no grouping, the test is not applicable and will return a p-value of 1.

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

- Farrington CP (1996). "On Assessing Goodness of Fit of Generalized Linear Models to Sparse Data." *Journal of the Royal Statistical Society, Series B*, **58**(2), 349-360. doi:10.1111/j.2517-6161.1996.tb02086.x
- Hosmer DW, Lemeshow S (1980). "A goodness-of-fit test for the multiple logistic regression model." *Communications in Statistics - Theory and Methods*, **9**(10), 1043-1069. doi:10.1080/03610928008827941
- Ebrahim EK, El-Kotory A (2026). "A Directional Hosmer-Lemeshow Goodness-of-Fit Test for Sparse Logistic Regression." arXiv:2607.15454 [stat.ME]. doi:10.48550/arXiv.2607.15454
- Ebrahim EK (2026). "Goodness-of-Fit Tests and Calibration Machine-Learning Algorithms for Logistic Regression with Sparse Data." M.Sc. thesis, Alexandria University. arXiv:2608.11140 [stat.ME]. doi:10.48550/arXiv.2608.11140

gof.features

*Goodness-of-fit evidence features for a fitted model***Description**

Builds the evidence vector used by the learned-ensemble goodness-of-fit test: one-sided z-scores $\Phi^{-1}(1 - p)$ from a panel of GOF tests plus the covariate-space directed tests. Larger values mean stronger evidence of misfit.

Usage

```
gof.features(
  object,
  tests = c("HL", "HL-equalwidth", "Pigeon-Heyse", "Tsiatis", "Xie", "EF", "DEF.poly2",
            "DEF.poly3", "DEF.stukel")
)
```

Arguments

object	A fitted binary logistic glm .
tests	Character vector of run.all.gof test names to use as panel features (default: a fast partition + DEF-family panel).

Value

A named numeric vector of evidence features.

See Also

[deploy.gof](#), [cdef.gof](#), [run.all.gof](#).

gof_demo

*Synthetic binary outcome data with a smooth calibration misfit***Description**

A small, fully synthetic dataset for demonstrating the goodness-of-fit and calibration battery. It was generated reproducibly (see `data-raw/make_gof_demo.R`) from a logistic data-generating process whose true linear predictor includes a *quadratic* term in (standardized) age. A model that regresses outcome on age linearly (together with bmi, sex and treatment) is therefore mildly misspecified, through a smooth, low-dimensional calibration distortion. This is the regime in which the directed Ebrahim–Farrington / EDGE test ([edge.gof](#), [def.gof](#)) is designed to have more power than classical omnibus tests such as Hosmer–Lemeshow.

Usage

```
gof_demo
```

Format

A data frame with 800 rows and 5 variables:

outcome binary response, 0/1 (event rate about 0.27).

age continuous covariate, years (range about 20–70). The true model depends on age quadratically.

bmi continuous covariate, body mass index in kg/m².

sex binary covariate, 0 = female, 1 = male.

treatment binary covariate, 0 = control, 1 = treated.

Details

The true data-generating linear predictor is

$$\eta = -0.6 + 0.8z_a - 0.7z_a^2 + 0.5z_b + 0.4\text{sex} - 0.3\text{treatment},$$

where $z_a = (\text{age} - 45)/14$ and $z_b = (\text{bmi} - 27)/4$, and $\Pr(\text{outcome} = 1) = \text{plogis}(\eta)$.

Source

Simulated; see `data-raw/make_gof_demo.R` in the package sources.

Examples

```
data("gof_demo", package = "ebrahim.gof")
fit <- glm(outcome ~ age + bmi + sex + treatment,
           data = gof_demo, family = binomial)
edge.gof(fit)
```

gof_demo_grouped	<i>Grouped-covariate companion to gof_demo (replicated covariate patterns)</i>
------------------	--

Description

A companion dataset to [gof_demo](#) built from the *same* data-generating process and seed discipline (see `data-raw/make_gof_demo.R`), except that the covariates are coarsened *before* the linear predictor is computed: age is rounded to 10-year bins (20, 30, ..., 70) and bmi to whole integers. The recorded covariates are therefore exactly the covariates the outcome was generated from, and many observations share a covariate pattern (328 distinct patterns among 800 observations, versus one pattern per observation in `gof_demo`).

Usage

gof_demo_grouped

Format

A data frame with 800 rows and 5 variables:

outcome binary response, 0/1 (event rate about 0.27).

age age in years, rounded to 10-year bins (20, 30, ..., 70). The true model depends on (binned) age quadratically.

bmi body mass index in kg/m², rounded to whole integers.

sex binary covariate, 0 = female, 1 = male.

treatment binary covariate, 0 = control, 1 = treated.

Details

Its purpose is to demonstrate the sparse-versus-grouped distinction that [run.all.gof](#) surfaces: the battery reports the per-observation ("sparse") and per-covariate-pattern ("grouped") forms of the pattern-sensitive tests (Pearson, deviance, McCullagh) side by side, and on replicated-pattern data such as this the two forms can disagree on the same fitted model. On the all-continuous `gof_demo` (every observation its own pattern) the two forms coincide – the degenerate case.

The true data-generating linear predictor has the same form as for [gof_demo](#),

$$\eta = -0.6 + 0.8z_a - 0.7z_a^2 + 0.5z_b + 0.4\text{sex} - 0.3\text{treatment},$$

with $z_a = (\text{age} - 45)/14$ and $z_b = (\text{bmi} - 27)/4$ computed from the *binned* age and bmi, and $\Pr(\text{outcome} = 1) = \text{plogis}(\eta)$.

Source

Simulated; see `data-raw/make_gof_demo.R` in the package sources.

See Also

[gof_demo](#), [run.all.gof](#)

Examples

```
data("gof_demo_grouped", package = "ebrahim.gof")
fit <- glm(outcome ~ age + bmi + sex + treatment,
           data = gof_demo_grouped, family = binomial)
# sparse and grouped forms reported side by side:
run.all.gof(fit, include_slow = FALSE, install = "no")
```

`gof_install_suggests` *Install the optional packages used by `run.all.gof()`*

Description

The slow tests in `run.all.gof` rely on optional packages that live in `Suggests` (**givitiR** and **callr** for the GiViTI calibration test, **mgcv** for the GAM tests, **randomForest** and **dcov** for the adaptive BAGofT test (**BAGofT** itself is used only on request), and **ResourceSelection** for the Lai-Liu test). Per CRAN policy the package never installs them on its own; this helper installs the missing ones for you, asking first.

Usage

```
gof_install_suggests(pkgs = NULL, ask = interactive(), update = FALSE)
```

Arguments

<code>pkgs</code>	Optional character vector of package names to consider. Defaults to the full optional set used by the battery.
<code>ask</code>	Logical; when TRUE (the default in interactive sessions) you are shown the list of packages to be installed/updated and asked to confirm first. Set <code>ask = FALSE</code> to proceed without a prompt (e.g. in a setup script you control).
<code>update</code>	Logical; when FALSE (the default) only the <i>missing</i> packages are installed and anything already present is left untouched. When TRUE the function also checks (via old.packages) which of the present packages are out of date and offers to update those too. The update check contacts your CRAN mirror, so it is a little slower.

Value

Invisibly, the character vector of packages that were installed or updated (empty if nothing was needed).

See Also

[run.all.gof](#)

Examples

```
## Not run:
# install whatever optional packages are missing, after confirming:
gof_install_suggests()

# also update any that are out of date:
gof_install_suggests(update = TRUE)

# just the GiViTI dependencies, no prompt:
gof_install_suggests(c("givitiR", "callr"), ask = FALSE)
```

```
## End(Not run)
```

legoft	<i>LEGofT: frozen-weight combination goodness-of-fit test for binary logistic regression</i>
--------	--

Description

Combines eleven classical and directed goodness-of-fit statistics with weights that were fixed offline and ship frozen, and calibrates the combination by a parametric bootstrap at the fitted parameters. Nothing is retrained when you call it.

Usage

```
legoft(object, B = 199, seed = NULL, weights = NULL)
```

Arguments

object	a fitted binomial glm.
B	number of parametric-bootstrap replicates for the reference distribution. 199 gives a smallest attainable p-value of 0.005; raise it for smaller p-values.
seed	optional integer for reproducibility.
weights	optional named vector of member weights; defaults to the frozen rule. Supplying your own makes the result no longer the shipped procedure – say so if you report it.

Details

The p-value is exact in finite samples when the null parameters are known. With the parameters estimated – the case here – the calibration is asymptotic; simulation at $n = 500$ put the empirical size at 0.037 against a nominal 0.05.

Value

an object of class "legoft": the statistic, its bootstrap p-value, the member p-values, and B_used.

See Also

[legoft.localize](#) for which domain of evidence carries the misfit.

Examples

```
set.seed(1)
x1 <- runif(300, -3, 3); x2 <- rnorm(300)
y <- rbinom(300, 1, plogis(0.3 + 0.8 * x1 - 0.5 * x2 + 0.25 * x1^2))
fit <- glm(y ~ x1 + x2, family = binomial())
legoft(fit, B = 99, seed = 1)
```

legoft.localize	<i>Localize misspecification with familywise error control</i>
-----------------	--

Description

Reports which of two domains of evidence carries the misfit, using closed testing over the domains. A domain is implicated only when every intersection containing it rejects, so the probability of implicating any collection of domains whose pooled evidence is jointly exchangeable with the reference draws' is at most α .

Usage

```
legoft.localize(object, B = 199, seed = NULL, alpha = 0.05)
```

Arguments

object	a fitted binomial glm.
B, seed	as in legoft .
alpha	familywise level.

Details

The two domains are defined by what the members can see, not by taxonomy. **INDEX** holds the seven statistics that read the linear index – the grouping tests, which stratify on fitted risk, and the directed tests, which examine bends in that same index. **COV** holds the four that read directions orthogonal to it. The calibration and link readings are pooled deliberately: they are not separately identifiable, because fitted risk is a monotone transform of the index.

A verdict says where the *evidence* lies. It does not say which part of the model to repair: a departure of one kind can move members assigned to the other domain, and an unimplicated domain is not thereby certified correct.

Value

an object of class "legoft_localize": the verdict, the intersection p-values, and the member p-values.

See Also

[legoft](#)

Examples

```
set.seed(1)
x1 <- runif(400, -3, 3); x2 <- rnorm(400)
y <- rbinom(400, 1, plogis(0.3 + 0.8 * x1 - 0.5 * x2 + 0.4 * x1 * x2))
fit <- glm(y ~ x1 + x2, family = binomial())
legoft.localize(fit, B = 99, seed = 1)
```

plot.gof_battery	<i>Plot the GiViTI calibration belt from a goodness-of-fit battery</i>
------------------	--

Description

Draws the GiViTI calibration belt stored on a [run.all.gof](#) result that was produced with `calibration_plot = TRUE`. The belt shows the fitted calibration curve with a confidence region against the 45-degree line.

Usage

```
## S3 method for class 'gof_battery'
plot(x, ...)
```

Arguments

x	A gof_battery object from run.all.gof .
...	Passed to the givitiR plot method.

Value

x, invisibly.

print.gof_battery	<i>Print a goodness-of-fit battery</i>
-------------------	--

Description

Formats the [run.all.gof](#) result as a compact, readable table: rows grouped by test family, p-values shown to four decimals (or scientific for very small values, "-" when not available), and a significance flag. The object is still a plain `data.frame` underneath, so all the raw columns remain available for programmatic use.

Usage

```
## S3 method for class 'gof_battery'
print(x, ...)
```

Arguments

x	A gof_battery object returned by run.all.gof .
...	Ignored.

Value

x, invisibly.

projection.gof

*Projection Goodness-of-Fit Test for Binary Regression***Description**

projection.gof() computes the projection test of Escanciano (2006) as defined for logistic regression by Liu et al. (2024, Sec. 2.1), and refers it to their model-based bootstrap. The residual-marked empirical process is taken along every direction of the covariate space and its Cramer-von Mises norm is integrated over the unit sphere, so the test is consistent against any departure of the mean function, including departures that are invisible along the fitted linear predictor (where the Stute-Zhu test looks).

Usage

```
projection.gof(object, B = 1000, scale = FALSE, tol = 1e-10)
```

Arguments

object	A fitted binary <code>glm</code> (family = binomial, any link) with unit prior weights and at least one covariate.
B	Number of bootstrap replicates (default 1000, as in Liu et al.'s examples).
scale	Logical; standardize each covariate column before computing the angles. Default FALSE, the statistic of Liu et al.
tol	A difference vector shorter than tol times the largest covariate norm (at least 1) is treated as the zero vector.

Details

With $e_i = y_i - \hat{p}_i$ and X_i the covariate vector of observation i (the model matrix without its intercept), the statistic is

$$T = n^{-2} \sum_i \sum_j \sum_l e_i e_j A_{ijl},$$

$$A_{ijl} = \int_{S^p} I(X_i^T w \leq X_l^T w) I(X_j^T w \leq X_l^T w) dw,$$

with dw the uniform probability measure on the unit sphere. For $u = X_i - X_l$ and $v = X_j - X_l$ both nonzero, $A_{ijl} = \{\pi - \angle(u, v)\}/(2\pi)$; it is 1/2 when exactly one of them is zero and 1 when both are (Escanciano 2006, Appendix). The constant does not affect the bootstrap p-value. Liu et al. print the weight as a multiple of $\arccos(\cdot)$; the integral fixes the orientation as $\pi - \arccos(\cdot)$, and a Monte Carlo over random directions agrees with the form used here (see the package tests).

The weight uses the covariates on their own scale, as Liu et al. do. The angle is not invariant to rescaling one column, so `scale = TRUE`, which standardizes each column first, gives a different test; it is offered for covariates in unrelated units.

Reference distribution. The p-value comes from the model-based bootstrap of Liu et al. (2024, Sec. 2.1; Dikta et al. 2006): B responses are drawn from the fitted probabilities, the model is refitted to each with the same design, link and offset, and $p = (1 + \#\{T^* \geq T\})/(B + 1)$. A refit that fails

scores $T^* = -\infty$, which can only make the test conservative; the count is returned. The default $B = 1000$ is the number of bootstrap samples Liu et al. use in both of their data examples.

Cost. The weight matrix is computed once, in $O(n^3p)$ time and $O(n^2)$ memory, and each bootstrap replicate then costs one refit and one quadratic form. In pure R the weight takes about 0.2 s at $n = 200$ and 3 to 5 s at $n = 500$, and the whole test with $B = 1000$ about 1 s and 6 s; at $n = 2000$ it takes minutes and several n-by-n matrices of memory. With a single covariate the weight has an exact rank form and is much faster.

Value

An object of class "htest" with elements statistic (T), parameter (B), p.value, method, data.name, and additionally boot (the B bootstrap statistics), n_failed (refits that failed) and n_nonconverged (refits that did not converge; their fitted values are still used).

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

- Escanciano JC (2006). "A consistent diagnostic test for regression models using projections." *Econometric Theory*, 22(6), 1030–1051. doi:[10.1017/S0266466606060506](https://doi.org/10.1017/S0266466606060506)
- Liu H, Li X, Chen F, Haerdle W, Liang H (2024). "A comprehensive comparison of goodness-of-fit tests for logistic regression models." *Statistics and Computing*, 34, 175. doi:[10.1007/s11222024-104875](https://doi.org/10.1007/s11222024-104875)
- Dikta G, Kvesic M, Schmidt C (2006). "Bootstrap approximations in model checks for binary data." *Journal of the American Statistical Association*, 101(474), 521–530. doi:[10.1198/016214505000001032](https://doi.org/10.1198/016214505000001032)

See Also

[run.all.gof](#), where the test is the row "Projection".

Examples

```
set.seed(1)
n <- 150
x1 <- rnorm(n); x2 <- rnorm(n)
y <- rbinom(n, 1, plogis(0.5 * x1 - 0.5 * x2))
fit <- glm(y ~ x1 + x2, family = binomial())
projection.gof(fit, B = 99)

## an omitted interaction: invisible to a test that looks only along the
## fitted linear predictor, visible along other directions
y2 <- rbinom(n, 1, plogis(0.5 * x1 - 0.5 * x2 + 1.5 * x1 * x2))
bad <- glm(y2 ~ x1 + x2, family = binomial())
projection.gof(bad)          # B = 1000
```

run.all.external

Run the External-Validation Tests at Once

Description

Checks the calibration of predictions that were made *without* the data at hand – a published model, or any model, applied to new patients – and returns one tidy data.frame, one row per test, in the format of `run.all.gof`. Only the outcomes `y` and the predicted probabilities `p` are needed, so the predictions may come from a logistic regression, a random forest, a neural network or a clinical score: nothing is refitted.

Usage

```
run.all.external(y, p, G = 10, X = NULL, include_slow = FALSE)
```

Arguments

<code>y</code>	Binary (0/1) outcomes of the validation sample.
<code>p</code>	Predicted probabilities for the same records, produced without them.
<code>G</code>	Number of risk groups for the directed and Hosmer–Lemeshow tests (default 10).
<code>X</code>	Optional covariate matrix or data frame, for le Cessie’s test only.
<code>include_slow</code>	Logical; run le Cessie’s test when <code>X</code> is given (default FALSE).

Details

Why a separate battery. `run.all.gof(y, predicted_probs = p)` treats the predictions as fitted to `y`, and each reference distribution then allows for the parameters the fit spent. When the predictions are frozen nothing was spent: the Hosmer–Lemeshow statistic is referred to χ^2_G , not χ^2_{G-2} ; Stukel’s terms are added to the frozen linear predictor as an offset; and the directed test takes $\Omega = I$ with a constant column, because no score equation absorbs the overall level. Using the internal references on frozen predictions makes every one of these tests conservative.

The tests (Family in brackets).

- EDGE (`G=...`) [Directed] – the directed grouped test in external mode, `def.gof(y, predicted_probs = p, external = TRUE)`: the cubic basis plus a constant, four degrees of freedom. Run at `G` (ten by default, the setting that keeps its level when a few records carry corrupted predictors) and at `G = "auto"`, `max(10, ⌈n/25⌉)`, which has more power on clean data.
- Cox recalibration [Calibration] – the likelihood-ratio test that the calibration intercept is 0 and the slope 1 in `glm(y ~ logit(p))` (Cox 1958; Miller et al. 1991). The Note gives both estimates.
- Calibration-in-the-large [Calibration] – the score test that the intercept is 0 with the slope held at 1: $(O - E) / \sqrt{\sum p(1 - p)}$.
- Spiegelhalter-z [Calibration] – Spiegelhalter’s (1986) z test, two-sided.

- GiViTI [Calibration] – the GiViTI calibration test in its external mode (Nattino et al. 2014), run in an isolated **callr** process; needs **givitIR** and **callr**. When it selects a polynomial of degree one it returns the same p -value as the Cox test.
- Hosmer-Lemeshow (external) [Partition] – G equal-frequency risk groups, referred to χ_G^2 .
- Stukel (offset) [Directed] – Stukel’s two terms added to the frozen linear predictor, likelihood-ratio χ_2^2 .
- le Cessie (external) [Smoothing] – le Cessie and van Houwelingen’s kernel statistic over covariate space with $\Omega = I$; only when X is given and `include_slow = TRUE`, because it builds an $n \times n$ kernel.

Three descriptive rows carry no p -value: the ratio of observed to expected events, the calibration slope, and the c-statistic (area under the ROC curve).

Reading the panel. In a simulation of external validation (EDGE paper, Supporting Information) the directed test at ten groups had the power of the Cox test and the GiViTI belt on average, led them on curved departures and trailed them on a shift, and kept its level with ten reversed predictions in 1000 records, where the Cox test and the belt did not. The Cox test says whether the predictions need recalibrating; the directed test says whether a recalibration would be enough.

Value

A data.frame of class `gof_battery` with columns `Test`, `Family`, `Statistic`, `df`, `p_value` and `Note`, printed by the same method as [run.all.gof](#).

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

See Also

[def.gof](#) (its external argument), [run.all.gof](#).

Examples

```
set.seed(1)
n <- 1000
x <- rnorm(n)
p <- plogis(-1 + 0.8 * x)           # a published model, frozen
y <- rbinom(n, 1, plogis(-1 + 0.6 * x)) # new patients: the model is overfitted
run.all.external(y, p)
```

run.all.gof

*Run a Battery of Goodness-of-Fit Tests at Once***Description**

Runs several goodness-of-fit tests for a binary logistic regression in one call and returns one tidy data.frame, one row per test. Pass a fitted glm to run the whole battery; pass (y, predicted_probs) to run the tests that need only predictions. Each test is wrapped so that a failure of one test never aborts the whole run.

Usage

```
run.all.gof(
  object,
  predicted_probs = NULL,
  X = NULL,
  tests = "all",
  G = 10,
  include_slow = TRUE,
  parallel = FALSE,
  ncores = NULL,
  calibration_plot = FALSE,
  install = c("ask", "no", "yes"),
  control = list()
)
```

Arguments

object	A fitted binary logistic glm , or a binary (0/1) response vector y (then supply predicted_probs).
predicted_probs	Numeric predicted probabilities; required when object is a y vector.
X	Optional design matrix; lets the directed (DEF) tests run from the (y, predicted_probs) form.
tests	Either "all" (default) or a character vector of test names to run (e.g. c("EF", "DEF.poly3", "HL")).
G	Integer number of groups passed to the grouping tests (default 10), or "auto" for max(10, ceiling(n / 25)) (see def.gof). "auto" is resolved once, so every row, the ensemble rows included, uses the same number of groups; the directed rows record it in Note.
include_slow	Logical; when TRUE (the default) the full battery runs, including the slow tests: le Cessie-van Houwelingen smoothing ($O(n^2)$ - $O(n^3)$), the GAM tests, Stute-Zhu, eHL, BAGofT, Projection, and GiViTL. Set FALSE for a quick run with the fast tests only. A one-time message notes this whenever slow tests are included.
parallel	Logical; when TRUE, the resampling loops of the slow bootstrap tests (Stute-Zhu and Lai-Liu-HL) are run on a local PSOCK cluster via parLapply (works on

	all platforms, including Windows). All other tests are unaffected. The default FALSE keeps every loop sequential, exactly as in previous versions.
ncores	Integer; the number of worker processes used when parallel = TRUE. The default NULL uses <code>max(1, parallel::detectCores() - 1)</code> . Values below 2 fall back to the sequential path.
calibration_plot	Logical; when TRUE and GiViTI is among the tests, also compute and draw the GiViTI calibration belt and store it on the result (retrievable with <code>plot()</code>). Default FALSE.
install	One of "ask" (default), "no", or "yes", controlling what happens when a test in the run needs an optional package that is not installed. In an <i>interactive</i> session, "ask" lists the missing packages and asks before installing, and "yes" installs them without asking; "no" never installs (the test is just skipped with a note). In a non-interactive session (scripts, R CMD check) nothing is ever installed, regardless of this setting. See gof_install_suggests .
control	Optional named list of per-test options. Recognized entries: "Stute-Zhu" = <code>list(B = ...)</code> (bootstrap replicates); GiViTI = <code>list(devel = "internal"/"external")</code> ; "Lai-Liu-HL" = <code>list(n0 = ..., k = ..., alpha = ...)</code> ; Stukel = <code>list(form = "joint"/"lr"/"marginal")</code> (the joint score test by default, the likelihood-ratio refit, or the pre-2.8.0 marginal sum); <code>DEF.poly2</code> , <code>DEF.poly3</code> , <code>DEF.stukel</code> and <code>DEF.sym</code> = <code>list(weights = "unit"/"score", G = ...)</code> , where G may be "auto" (see def.gof); and BAGofT = <code>list(...)</code> which forwards to the binary adaptive test – <code>nsim</code> (resampling iterations; default 100), <code>nsplits</code> , <code>ne</code> (the estimation-split size), and the random-forest partitioner's tuning <code>Kmax</code> (maximum number of adaptive partition cells), <code>ntree</code> , <code>nmin</code> , <code>mtry</code> , <code>maxnodes</code> , and <code>engine</code> ("auto", "fast" or "package"; see bagoft.fast). Example: <code>list(BAGofT = list(nsim = 200, Kmax = 8, ntree = 500))</code> ; and <code>Projection</code> = <code>list(B = ..., scale = FALSE, max_n = 3000)</code> (bootstrap refits, standardized covariates, and the sample size above which the row is skipped; see projection.gof).

Details

How to read the battery. Every test here answers the same question – does the fitted model describe the data it was fitted to – but they differ in the departure each is built to notice. That is why the panel is more informative than any single p-value: agreement across families is evidence of fit, and disagreement tells you *what kind* of misfit is present. A test that rejects points at the departure its own construction is sensitive to.

The one thing the panel cannot do is rescue an invalid reference distribution. Under sparse data – almost every covariate pattern unique – the classical chi-square references fail, which is the situation this package was written for.

Every row the battery returns carries a Family label, and the sections below are those labels: find the label in the output, then the section of the same name here.

Global and Standardized statistics (Family "Global", "Standardized"). These compare observed and fitted responses over the whole sample without grouping.

- Pearson – the sum of squared Pearson residuals. Its chi-square reference assumes many observations per covariate pattern. On sparse data that assumption fails and the test is unreliable in both directions; it is reported for completeness and comparison, not for use.

- Deviance – the likelihood-ratio statistic against the saturated model. On ungrouped binary data its expectation depends on the fitted risks alone and not on whether they agree with the responses, so it sits above $n - p$ when the fitted risks are near one half and far below it when they are extreme; the direction of the error is set by the risk profile rather than by the fit. On a correctly specified model with mid-range risks it therefore rejects almost always. It is reported for completeness and comparison, not for use.
- Osius-Rojek – rescues the Pearson statistic by standardizing it with its asymptotic mean and variance computed under increasing sample size rather than increasing cell counts, giving a normal reference that stays valid when patterns are unique. Fast, parameter-free, and a sensible default. It can be anti-conservative at small samples and conservative under a strongly skewed covariate.
- McCullagh – standardizes the Pearson statistic by its *exact* conditional moments rather than asymptotic ones (computed by the Kuss 2002 algorithm). Typically the most powerful of the global statistics, at a cost that grows steeply with sample size.
- Copas-RSS – the unweighted sum of squares of the raw residuals, referred to its own moments. Weighting each residual equally rather than by its variance makes it comparatively sensitive to misfit in the middle of the risk range.
- Information-Matrix – the White/Orme test. It compares two estimators of the information matrix that agree only if the model is correctly specified, so it is an omnibus check on the whole specification rather than on calibration alone.

Partition tests (Family "Partition"). These sort observations by fitted risk, group them, and compare observed with expected counts group by group.

- HL – the Hosmer-Lemeshow test, the field's default: G groups cut at percentiles of fitted risk (the "deciles of risk" when $G = 10$), referred to chi-square on $G - 2$ degrees of freedom. Note that this reference was established by simulation, not derivation. Familiar and cheap, but modest in power, and its result depends on the grouping.
- HL-equalwidth – the same statistic with groups cut at fixed probability intervals instead of percentiles. When fitted risks are concentrated in a narrow range, intervals can come out empty and the test may not be computable at all, which is why percentile grouping is usually preferred.
- Pigeon-Heyse – a variance-corrected Hosmer-Lemeshow statistic that accounts for the variability of fitted probabilities within each group. The correction is conservative in sparse designs, so a non-rejection carries less weight than the nominal level suggests.
- F-test – deviance residuals compared across the risk groups by a one-way analysis of variance. It is markedly liberal under sparsity and grows more so with sample size; it is included for comparison.
- Lai-Liu-HL – Lai and Liu's procedure for using the Hosmer-Lemeshow test in large samples, where any test eventually rejects. It standardizes power to a reference sample size n_0 and so returns no p-value: the statistic is the standardized power and the accept/reject decision appears in the Note column. Tune with `control = list("Lai-Liu-HL" = list(n0 = ..., k = ...))`.
- HL-largeN – Nattino, Pennell and Lemeshow's (2020) large-sample Hosmer-Lemeshow test. Instead of perfect fit it tests whether the misfit, measured by $\epsilon = \sqrt{\lambda/n}$, exceeds a tolerance ϵ_0 : the ordinary statistic is referred to a noncentral χ^2_{G-2} with noncentrality $\epsilon_0^2 n$. By their

convention ϵ_0 is the misfit that would be just significant at $n_0 = 1e6$; change it with `control = list("HL-largeN" = list(n0 = ...))`. The Note gives ϵ_0 and the estimate $\hat{\epsilon}$. Meant for samples in the tens of thousands and above; in small samples it is conservative.

Directed tests (Family "Directed"). Rather than asking whether anything is wrong, these ask whether a *particular* shape of departure is present, which buys power when the guess is right.

- EF, EF-normal – the omnibus Ebrahim-Farrington test, with the chi-square and normal references respectively. Built for sparse data, where the classical grouped statistics lose their reference.
- DEF.poly2, DEF.poly3, DEF.stukel, DEF.sym – the directed forms, each aiming the test at a smooth departure in the shape of the calibration curve: a quadratic or cubic drift in the linear predictor, Stukel's asymmetry-and-tail family, or Stukel's symmetric direction, tails too heavy or too light on both sides. Powerful when the misfit resembles the chosen basis, weaker when it does not. Each row takes weights and G through control, for example `control = list(DEF.sym = list(weights = "score", G = "auto"))`; see [def.gof](#). On a sample with no event, or no non-event, there is no fitted model; these rows and the Stukel row are then NA, and Note says why.
- Stukel – a score test against Stukel's generalized logistic link, which nests the logit and lets the two tails bend independently. It is aimed squarely at link misspecification. The two tail directions are tested jointly on 2 degrees of freedom (1 when every fitted risk lies on one side of one half, which Note then says). Testing them jointly gives up a little power against one-sided (cloglog-type) departures. Up to 2.7.0 this row summed two marginal statistics and was liberal; see NEWS. `control = list(Stukel = list(form = "lr"))` gives the likelihood-ratio test for the same two directions, and `form = "marginal"` the old sum, for reproducing earlier results only. The likelihood-ratio refit can fail to converge under separation; the row is then NA, with a note. The joint form leaves out a direction whose information after the fit is below 10^{-10} times its information before the fit, as when the fitted logit is constant; when no direction is left the row is NA, with a note.

Covariate-space tests (Family "Covariate-space"). These partition the covariates themselves rather than the fitted risk, so they can see structure that risk-ordering averages away – an omitted interaction, for instance, need not disturb the marginal calibration at all.

- Tsiatis – a score test that adds indicator variables for regions of the covariate space and asks whether they improve the fit.
- Xie – clusters the covariate space and compares observed with expected counts per cluster, using the corrected degrees of freedom $G - k/2 - 1$ with k the number of predictors.
- Pulkstenis-Robinson – crosses a categorical covariate with risk groups, so it needs at least one categorical predictor. The package auto-detects one (any factor, character or logical, or a numeric with few distinct values, controlled by `getOption("ebrahim.gof.pr.maxlev", 6)`) and returns NA with a note when there is none.

Smoothing and GAM tests (Family "Smoothing", "GAM"). These replace grouping with a smoother, so nothing is lost to an arbitrary choice of bin edges.

- le-Cessie – the le Cessie-van Houwelingen score test, which smooths the residuals over the covariate space with a kernel and asks whether the smoothed surface is further from zero than chance allows. Sensitive to local structure that omnibus statistics average away, and one of the most expensive tests here, with cost growing sharply in the sample size.

- HL-GAM, PR-GAM, Xie-GAM – variants that fit a deliberately overfitted generalized additive model and use it to define the grouping, letting the data rather than the analyst choose where the boundaries fall. These need **mgcv**.

Resampling tests (Family "Bootstrap"). When a statistic has no usable closed-form reference, these build one by simulation.

- Stute-Zhu – a cumulative-residual test: residuals are accumulated along the fitted linear predictor and the largest excursion of that path is compared with a parametric bootstrap. It needs no binning and no bandwidth, and in practice it is the best-behaved test in the battery on size. Set the number of resamples with `control = list("Stute-Zhu" = list(B = ...))`.
- BAGofT – the binary adaptive test, which splits the data, uses one part to learn a partition that separates fitted from observed, and tests on the other. Its behaviour depends strongly on how many splits and resamples it is given; set them with `control = list(BAGofT = list(nsim = ...))` and be aware that the published default is far more expensive than a single split. By default the row is computed by `bagoft.fast`, which returns the same p-value as the **BAGofT** package for the same seed and needs only **randomForest** (and **dcov** above five covariates); `control = list(BAGofT = list(engine = "package"))` calls **BAGofT** itself.
- Projection – the projection test of Escanciano (2006) as defined for logistic regression by Liu et al. (2024): the cumulative residual process is taken along every direction of the covariate space, not only along the fitted linear predictor as in Stute-Zhu, so it also sees departures such as an omitted interaction. Model-based bootstrap with $B = 1000$ refits by default; see `projection.gof`. Its weight matrix costs $O(n^3)$ time and $O(n^2)$ memory, so the row is skipped above $n = 3000$. Set `control = list(Projection = list(B = ..., max_n = ...))`.

Calibration tests (Family "Calibration"). These come from clinical prediction, and ask directly whether predicted risks match observed frequencies.

- GiViTI, GiViTI-external – the GiViTI polynomial calibration test, which fits a polynomial of the fitted risk and tests whether it departs from the identity, under the internal and external development assumptions respectively. The two are *not* interchangeable: on data used to fit the model the internal form is the appropriate one. It also produces the calibration belt, which shows *where* on the risk scale a model drifts; see `calibration_plot`. Wraps **givitiR** in an isolated **callr** subprocess, so a failure inside its compiled dependencies returns NA instead of ending your session. Select with `control = list(GiViTI = list(devel = "internal"))`.
- eHL – an e-value form of the Hosmer-Lemeshow test, reported as $p = \min(1, 1/e)$. E-values are safe under optional stopping, which conventional p-values are not, but the conversion shown here is conservative.

Combinations (Family "Ensemble"). Rather than choosing one test, these pool several.

- `Ensemble.Vote(3DEF)` and `Ensemble.Univ(3DEF+EF)` – Cauchy combinations of the directed tests, and of those plus the omnibus EF. The Cauchy combination is valid without knowing how the members correlate, which is what makes pooling dependent tests possible at all. The point is to avoid having to guess the departure in advance, at the cost of being slightly less powerful than the single best member would have been. Both rows always combine the unit form of `DEF.poly2`, `DEF.poly3` and `DEF.stukel` at the battery's G. When `control` gives those rows other weights or another G, the ensemble rows do not follow, and their Note says "unit form".

- See also [legoft](#), a pretrained combination whose weights are fixed offline and ship frozen, so two analysts running it on the same data obtain the same p-value.

Implementation notes. Tsiatis and Xie cluster the covariate space with k-means using a fixed internal seed, so results are reproducible and your own random stream is left untouched. The equal-frequency groups of HL, F-test, EF and the DEF rows split tied fitted risks by row order, so with many ties (grouped data, or a model on discrete covariates) their results can depend on the order of the rows; randomising the row order is advised. Every bundled test reproduces the implementation used in the original simulation study: Osius-Rojek follows **LogisticDx**'s `gof.glm`, Copas-RSS follows the **rms** `gof` residual, and HL follows `ResourceSelection::hoslem.test`. The exception is `Stukel`: its default joint score statistic agrees with `anova(..., test = "Rao")` for the augmented model, up to `glm`'s convergence tolerance, and only `form = "marginal"` reproduces **LogisticDx** (through `statmod::glm.scoretest` when **statmod** is installed).

Procedures the battery does not select. Some of the package's own methods are not part of the panel and are called directly on the fitted model, their p-values read beside it: [deepgof1](#) and [legoft](#) are of that kind. Naming them in tests does not reach them.

For goodness of fit after *penalized* fitting, where none of the above references are valid because the coefficients are shrunk, use [calm.gof](#) or [shrink.gof](#). These take the design, the response and the penalty rather than a fitted `glm`, so they are called separately from the battery.

Value

A data.frame (of class `gof_battery`) with columns `Test`, `Family`, `Statistic`, `df`, `p_value`, and `Note`, one row per test. A dedicated `print` method shows the rows grouped by family with formatted p-values and significance flags; the underlying columns remain available for programmatic use.

Note

Grouped vs sparse forms. Pearson, Deviance and McCullagh are reported in two forms: the default (sparse / one-trial) form and a "(grouped)" form computed on the distinct covariate patterns (each a $\text{Binomial}(m_g, P_g)$). The two are identical when every covariate pattern is unique (fully sparse data, as in the simulation) and differ only when patterns repeat ($m_g > 1$). To avoid clutter, the "(grouped)" row is shown *only* when it actually differs from the sparse form (i.e., when some pattern repeats); on fully sparse data it is a duplicate and is omitted. Osius-Rojek is always computed on covariate patterns, matching its classical (**LogisticDx**) definition.

Farrington vs EF. The *original* Farrington (1996) test is a grouped (covariate-pattern) test. The Ebrahim-Farrington (EF) test is its *sparse-data* counterpart: it does not group by covariate pattern but forms G data-dependent bins of the predicted risk, so it applies directly to fully sparse data. Use EF for sparse binary data; the grouped Farrington form is appropriate only when covariate patterns repeat.

Reproducibility of the parallel path. With `parallel = TRUE` the cluster's random-number streams are initialized with [clusterSetRNGStream](#), seeded deterministically from the session's current RNG state. Two runs from the same [set.seed](#) state (and the same `ncores`) therefore give identical bootstrap p-values. Note that the parallel L'Ecuyer-CMRG streams necessarily differ from the serial RNG stream, so `parallel = TRUE` results differ (within Monte-Carlo error) from `parallel = FALSE` results at the same seed; this is standard and both are valid. Results also depend on `ncores`, because the replicates are split across workers.

Author(s)

Ebrahim Khaled Ebrahim <ebrahimkhaled@alexu.edu.eg>

References

The aggregated tests are due to their original authors; they are provided here for comparison and credited as follows.

Farrington CP (1996). "On Assessing Goodness of Fit of Generalized Linear Models to Sparse Data." *Journal of the Royal Statistical Society B*, **58**(2), 349–360. doi:10.1111/j.25176161.1996.tb02086.x

Hosmer DW, Lemeshow S (1980). "Goodness of Fit Tests for the Multiple Logistic Regression Model." *Communications in Statistics – Theory and Methods*, **9**(10), 1043–1069. doi:10.1080/03610928008827941

McCullagh P (1985). "On the Asymptotic Distribution of Pearson's Statistic in Linear Exponential Family Models." *International Statistical Review*, **53**(1), 61–67. doi:10.2307/1402880

Osius G, Rojek D (1992). "Normal Goodness-of-Fit Tests for Multinomial Models with Large Degrees of Freedom." *Journal of the American Statistical Association*, **87**(420), 1145–1152. doi:10.1080/01621459.1992.10476271

le Cessie S, van Houwelingen JC (1991). "A Goodness-of-Fit Test for Binary Regression Models, Based on Smoothing Methods." *Biometrics*, **47**(4), 1267–1282. doi:10.2307/2532385

Stukel TA (1988). "Generalized Logistic Models." *Journal of the American Statistical Association*, **83**(402), 426–431. doi:10.1080/01621459.1988.10478613

Stute W, Zhu LX (2002). "Model Checks for Generalized Linear Models." *Scandinavian Journal of Statistics*, **29**(3), 535–545. doi:10.1111/14679469.00304

Escanciano JC (2006). "A Consistent Diagnostic Test for Regression Models Using Projections." *Econometric Theory*, **22**(6), 1030–1051. doi:10.1017/S0266466606060506

Liu H, Li X, Chen F, Haerdle W, Liang H (2024). "A Comprehensive Comparison of Goodness-of-Fit Tests for Logistic Regression Models." *Statistics and Computing*, **34**, 175. doi:10.1007/s11222024104875

Tsiatis AA (1980). "A Note on a Goodness-of-Fit Test for the Logistic Regression Model." *Biometrika*, **67**(1), 250–251. doi:10.1093/biomet/67.1.250

Xie XJ, Pendergast J, Clarke W (2008). "Increasing the Power: A Practical Approach to Goodness-of-Fit Test for Logistic Regression Models with Continuous Predictors." *Computational Statistics & Data Analysis*, **52**(5), 2703–2713. doi:10.1016/j.csda.2007.09.027

Pulkstenis E, Robinson TJ (2002). "Two Goodness-of-Fit Tests for Logistic Regression Models with Continuous Covariates." *Statistics in Medicine*, **21**(1), 79–93. doi:10.1002/sim.943

Nattino G, Finazzi S, Bertolini G (2014). "A New Calibration Test and a Reappraisal of the Calibration Belt for the Assessment of Prediction Models Based on Dichotomous Outcomes." *Statistics in Medicine*, **33**(14), 2390–2407. doi:10.1002/sim.6100

Pigeon JG, Heyse JF (1999). "An Improved Goodness of Fit Statistic for Probability Prediction Models." *Biometrical Journal*, **41**(1), 71–82. doi:10.1002/(SICI)15214036(199903)41:1<71::AID-BIMJ71>3.0.CO;2O

Copas JB (1989). "Unweighted Sum of Squares Test for Proportions." *Journal of the Royal Statistical Society C*, **38**(1), 71–80. doi:10.2307/2347682

- White H (1982). "Maximum Likelihood Estimation of Misspecified Models." *Econometrica*, **50**(1), 1–25. doi:[10.2307/1912526](https://doi.org/10.2307/1912526)
- Orme C (1988). "The Calculation of the Information Matrix Test for Binary Data Models." *The Manchester School*, **56**(4), 370–376. doi:[10.1111/j.14679957.1988.tb01339.x](https://doi.org/10.1111/j.14679957.1988.tb01339.x)
- Kuss O (2002). "Global Goodness-of-Fit Tests in Logistic Regression with Sparse Data." *Statistics in Medicine*, **21**(24), 3789–3801. doi:[10.1002/sim.1421](https://doi.org/10.1002/sim.1421)
- Lai X, Liu L (2018). "A Simple Test Procedure in Standardizing the Power of Hosmer-Lemeshow Test in Large Data Sets." *Journal of Statistical Computation and Simulation*, **88**(13), 2463–2472. doi:[10.1080/00949655.2018.1467912](https://doi.org/10.1080/00949655.2018.1467912)
- Nattino G, Finazzi S, Bertolini G (2014). "A New Calibration Test and a Reappraisal of the Calibration Belt for the Assessment of Prediction Models Based on Dichotomous Outcomes." *Statistics in Medicine*, **33**(14), 2390–2407. doi:[10.1002/sim.6100](https://doi.org/10.1002/sim.6100)
- Zhang J, Ding J, Yang Y (2023). "Is a Classification Procedure Good Enough? A Goodness-of-Fit Assessment Tool for Classification Learning." *Journal of the American Statistical Association*, **118**(542), 1115–1125. doi:[10.1080/01621459.2021.1979010](https://doi.org/10.1080/01621459.2021.1979010)
- Liu Y, Xie J (2020). "Cauchy Combination Test: A Powerful Test with Analytic p-Value Calculation under Arbitrary Dependency Structures." *Journal of the American Statistical Association*, **115**(529), 393–402. doi:[10.1080/01621459.2018.1554485](https://doi.org/10.1080/01621459.2018.1554485)
- Hosmer DW, Hosmer T, le Cessie S, Lemeshow S (1997). "A Comparison of Goodness-of-Fit Tests for the Logistic Regression Model." *Statistics in Medicine*, **16**(9), 965–980. doi:[10.1002/\(sici\)1097-0258\(19970515\)16:9<965::aidsim509>3.0.co;2o](https://doi.org/10.1002/(sici)1097-0258(19970515)16:9<965::aidsim509>3.0.co;2o)
- The methods introduced by this package, and the studies that evaluate them, are reported in the following. Reproduction materials for each are archived and citable.
- Ebrahim EK, El-Kotory A (2026). "A Directional Hosmer-Lemeshow Goodness-of-Fit Test for Sparse Logistic Regression." arXiv:2607.15454 [stat.ME]. doi:[10.48550/arXiv.2607.15454](https://doi.org/10.48550/arXiv.2607.15454)
- Ebrahim EK, El-Kotory A (2026). "Benchmarking Goodness-of-Fit and Calibration Algorithms for Logistic Regression Classifiers: A Large-Scale Simulation Study under Sparse Data." *Journal of Intelligent Computing and Data Science*, in press. arXiv:2607.16344 [stat.ME]. doi:[10.48550/arXiv.2607.16344](https://doi.org/10.48550/arXiv.2607.16344) Reproduction materials: doi:[10.5281/zenodo.21286171](https://doi.org/10.5281/zenodo.21286171)
- Ebrahim EK (2026). "Goodness-of-Fit Tests and Calibration Machine-Learning Algorithms for Logistic Regression with Sparse Data." M.Sc. thesis, Alexandria University. arXiv:2608.11140 [stat.ME]. doi:[10.48550/arXiv.2608.11140](https://doi.org/10.48550/arXiv.2608.11140)
- Ebrahim EK (2026). "EDGE: a directed goodness-of-fit test for sparse logistic regression." Reproduction materials. doi:[10.5281/zenodo.21247541](https://doi.org/10.5281/zenodo.21247541)
- Ebrahim EK (2026). "EDGES: A Selection-Free Ensemble Goodness-of-Fit Test." Reproduction materials. doi:[10.5281/zenodo.21320865](https://doi.org/10.5281/zenodo.21320865)
- Ebrahim EK (2026). "Detection Subspaces: A Theory of Goodness-of-Fit Tests." Reproduction materials. doi:[10.5281/zenodo.21687498](https://doi.org/10.5281/zenodo.21687498)
- Ebrahim EK (2026). "Shrinkage Invalidates the Hosmer-Lemeshow Test." Reproduction materials for [shrink.gof](https://github.com/ekbrahim/shrink.gof). doi:[10.5281/zenodo.21900114](https://doi.org/10.5281/zenodo.21900114)

See Also

[ef.gof](#), [def.gof](#), [def.ensemble.gof](#), [gof_install_suggests](#).

Examples

```

set.seed(1)
n <- 500
x <- runif(n, -3, 3)
y <- rbinom(n, 1, 1 / (1 + exp(-(0.6 * x))))
fit <- glm(y ~ x, family = binomial())

## The fast tests. Every covariate pattern here is unique, so this is the
## sparse case the package is written for.
res <- run.all.gof(fit, include_slow = FALSE)
res

## The return value is a plain data.frame, so the panel can be read
## programmatically as well as printed.
res[res$p_value < 0.05, c("Test", "Family", "p_value")] # what rejected
table(res$Family)                                     # coverage by family

## A correctly specified model: the panel should mostly agree, and any
## isolated rejection is the false positive you expect at the 5 percent level.
mean(res$p_value < 0.05, na.rm = TRUE)

## Now a model that is genuinely wrong -- the quadratic term is omitted.
y2 <- rbinom(n, 1, 1 / (1 + exp(-(0.6 * x + 0.5 * x^2))))
bad <- glm(y2 ~ x, family = binomial())
run.all.gof(bad, include_slow = FALSE)

## Pick specific tests, for instance one per family, which is the pairing the
## package recommends over relying on any single statistic.
run.all.gof(fit, tests = c("McCullagh", "HL", "Stukel", "Tsiatis"))

## Grouping tests take the number of groups; the choice is a convention
## rather than a derived optimum, so it is worth varying.
for (g in c(5, 10, 20))
  print(run.all.gof(fit, tests = "HL", G = g))

## The full battery (include_slow = TRUE by default). The slow tests need the
## suggested packages mgcv, randomForest, givitiR and callr; in an interactive
## session run.all.gof() offers to install any that are missing
## (install = "ask"). See also gof_install_suggests().
## The control= list forwards options to the individual tests; the reductions
## here keep the example quick without changing what it demonstrates.
run.all.gof(fit, install = "no",
            control = list("Stute-Zhu" = list(B = 50),
                          BAGofT = list(nsim = 20),
                          Projection = list(B = 99)))

## The GiViTI calibration belt shows WHERE on the risk scale a model drifts,
## which a single p-value cannot.
res2 <- run.all.gof(fit, tests = c("McCullagh", "GiViTI"),
                   calibration_plot = TRUE)
plot(res2) # redraw the stored belt

```

```
## Run the bootstrap loops on a PSOCK cluster. Seeds are handled internally,
## so a parallel run reproduces a serial one.
set.seed(1)
run.all.gof(fit, tests = "Stute-Zhu", parallel = TRUE, ncores = 2,
            control = list("Stute-Zhu" = list(B = 50)))
```

shrink.gof	<i>Shrinkage-corrected goodness-of-fit test for penalized (ridge) logistic regression</i>
------------	---

Description

The Hosmer–Lemeshow test is not valid when the coefficients are shrunk: penalization biases the fitted probabilities, the grouped residuals acquire a non-centrality, and the usual chi-squared reference is wrong. `shrink.gof()` removes that non-centrality and refers the corrected statistic to a bootstrap built from the debiased generator. For the same correction referred to a closed-form reference, needing one fit rather than B of them, see [calm.gof](#).

Usage

```
shrink.gof(
  X,
  y,
  lambda,
  G = 10,
  basis = c("edge", "decile"),
  B = 999,
  seed = NULL,
  penalize = NULL,
  uncorrected = FALSE
)
```

Arguments

X	numeric design matrix, without an intercept column.
y	binary response (0/1) of length <code>nrow(X)</code> .
lambda	ridge penalty on the theory scale, <code>lambda = n * lambda_glmnet</code> . Passing a glm-net lambda directly is the commonest error; multiply by n.
G	number of groups.
basis	"edge", "decile", or both.
B	bootstrap replicates.
seed	optional integer for reproducibility.

penalize	logical vector marking which columns of X are penalized; the intercept is never penalized.
uncorrected	if TRUE, also return the uncorrected statistic, which is what a naive application of Hosmer–Lemeshow to a penalized fit computes.

Details

The correction subtracts the estimated shrinkage non-centrality and prepivots against $\pi(\tilde{\beta})$ rather than $\pi(\hat{\beta})$ (Beran prepivoting), so the reference world matches the null being tested.

Reference implementation for: Ebrahim, E. K. (2026), "Shrinkage invalidates the Hosmer–Lemeshow test: goodness of fit for penalized logistic regression, with an application to glaucoma diagnosis."

Depends only on base R and **stats**, so results do not move with package versions.

Value

An object of class "shrink.gof": a list carrying the settings the test ran under (λ on the theory scale, G , B , n , p) and, for each requested basis, a component named SC.HL or SC.EDGE holding its statistic and bootstrap p.value, plus p.uncorrected when uncorrected = TRUE. Printed by print.shrink.gof.

Choosing between this and calm.gof

The statistics are the same; only the reference differs. [calm.gof](#) reads the exact tail of a weighted chi-squared law from a single fit, so it returns in a fraction of a second and its p-value carries no Monte Carlo error – which matters when the p-value is read as a magnitude rather than compared with a threshold. `shrink.gof()` needs B penalized refits and its p-value is granular to $1/(B + 1)$.

Prefer `calm.gof()` unless one of these applies: the columns to shrink are chosen through `penalize`, which `calm.gof()` does not accept; or $p \geq n$, which it refuses. Note also that `calm.gof()` takes a `lambda_scale` argument and `shrink.gof()` does not, so the same number passed to both is a `glmnet` penalty in one and a theory-scale penalty in the other – a factor of n apart. This function expects the theory scale.

See Also

[calm.gof](#), which refers the same corrected statistics to a closed-form reference and needs a single fit, and the section above for when to prefer it; [run.all.gof](#) for the unpenalized battery.

Examples

```
set.seed(1)
X <- matrix(rnorm(400 * 5), 400, 5)
y <- rbinom(400, 1, plogis(0.3 + X %*% c(0.8, -0.5, 0.3, 0, 0)))
shrink.gof(X, y, lambda = 40, basis = "decile", B = 99, seed = 1)
```

Index

- * **CALM**
 - calm.gof, 7
- * **DEF**
 - def.ensemble.gof, 15
 - def.gof, 17
 - edges.gof, 25
- * **DeepGOF-1**
 - deepgof1, 11
- * **EDGE**
 - edge.gof, 21
- * **Ebrahim-Farrington**
 - ef.gof, 26
- * **Hosmer-Lemeshow**
 - calm.gof, 7
 - ebrahim.gof-package, 3
 - ef.gof, 26
 - run.all.gof, 40
 - shrink.gof, 49
- * **LEGofT**
 - legoft, 33
 - legoft.localize, 34
- * **binary classification**
 - ebrahim.gof-package, 3
 - run.all.gof, 40
- * **bootstrap**
 - projection.gof, 36
- * **calibration**
 - calm.gof, 7
 - cdef.gof, 9
 - deepgof1, 11
 - deepgof1.external, 13
 - def.ensemble.gof, 15
 - def.gof, 17
 - deploy.gof, 20
 - ebrahim.gof-package, 3
 - edge.gof, 21
 - edges.gof, 25
 - ef.gof, 26
 - gof.features, 29
 - legoft, 33
 - legoft.localize, 34
 - run.all.gof, 40
 - shrink.gof, 49
- * **convolutional neural network**
 - deepgof1, 11
- * **covariate space**
 - cdef.gof, 9
 - deploy.gof, 20
 - gof.features, 29
- * **datasets**
 - gof_demo, 29
 - gof_demo_grouped, 30
- * **directed test**
 - cdef.gof, 9
 - def.gof, 17
 - deploy.gof, 20
 - edge.gof, 21
 - gof.features, 29
- * **ensemble**
 - def.ensemble.gof, 15
 - edges.gof, 25
- * **external validation**
 - deepgof1.external, 13
- * **familywise error**
 - legoft, 33
 - legoft.localize, 34
- * **frozen weights**
 - legoft, 33
 - legoft.localize, 34
- * **goodness-of-fit**
 - bagoft.fast, 5
 - calm.gof, 7
 - cdef.gof, 9
 - deepgof1, 11
 - def.ensemble.gof, 15
 - def.gof, 17
 - deploy.gof, 20
 - ebrahim.gof-package, 3

- edge.gof, 21
- edges.gof, 25
- ef.gof, 26
- gof.features, 29
- legoft, 33
- legoft.localize, 34
- projection.gof, 36
- run.all.gof, 40
- shrink.gof, 49
- * **high-dimensional**
 - calm.gof, 7
- * **logistic regression**
 - bagoft.fast, 5
 - calm.gof, 7
 - cdef.gof, 9
 - deepgof1, 11
 - def.ensemble.gof, 15
 - def.gof, 17
 - deploy.gof, 20
 - ebrahim.gof-package, 3
 - edge.gof, 21
 - edges.gof, 25
 - ef.gof, 26
 - gof.features, 29
 - legoft, 33
 - legoft.localize, 34
 - projection.gof, 36
 - run.all.gof, 40
 - shrink.gof, 49
- * **machine learning features**
 - cdef.gof, 9
 - deploy.gof, 20
 - gof.features, 29
- * **model diagnostics**
 - calm.gof, 7
 - cdef.gof, 9
 - deepgof1, 11
 - def.ensemble.gof, 15
 - def.gof, 17
 - deploy.gof, 20
 - ebrahim.gof-package, 3
 - edge.gof, 21
 - edges.gof, 25
 - ef.gof, 26
 - gof.features, 29
 - legoft, 33
 - legoft.localize, 34
 - run.all.gof, 40
- * **penalized regression**
 - calm.gof, 7
 - ebrahim.gof-package, 3
 - shrink.gof, 49
- * **pretrained**
 - deepgof1, 11
- * **random forest**
 - bagoft.fast, 5
- * **ridge regression**
 - calm.gof, 7
 - shrink.gof, 49
- * **sparse data**
 - ebrahim.gof-package, 3
 - edge.gof, 21
 - ef.gof, 26
- * **test battery**
 - ebrahim.gof-package, 3
 - run.all.gof, 40
- bagoft.fast, 5, 41, 44
- calm.gof, 3, 7, 45, 49, 50
- cdef.gof, 3, 9, 20, 29
- clusterSetRNGStream, 45
- deepgof1, 3, 11, 14, 15, 45
- deepgof1.external, 13
- def.ensemble.gof, 15, 18, 19, 22, 25, 47
- def.gof, 9, 10, 16, 17, 18, 21–25, 29, 39–41, 43, 47
- deploy.gof, 3, 20, 29
- ebrahim.gof (ebrahim.gof-package), 3
- ebrahim.gof-package, 3
- edge.gof, 3, 9, 17, 21, 25, 29
- edge.stream, 23
- edges.gof, 3, 25
- ef.gof, 3, 10, 13, 16, 19, 22, 26, 47
- glm, 5, 10, 16, 17, 20, 21, 29, 36, 40
- gof.features, 3, 20, 29
- gof_demo, 4, 29, 30, 31
- gof_demo_grouped, 30
- gof_install_suggests, 4, 32, 41, 47
- hoslem.test, 28
- legoft, 3, 13, 33, 34, 45
- legoft.localize, 3, 33, 34

old.packages, [32](#)

parLapply, [40](#)

plot.gof_battery, [35](#)

print.edge_stream(edge.stream), [23](#)

print.gof_battery, [35](#)

projection.gof, [36](#), [41](#), [44](#)

run.all.external, [24](#), [38](#)

run.all.gof, [3](#), [4](#), [6](#), [10](#), [12–14](#), [22](#), [29](#), [31](#),
[32](#), [35](#), [37–39](#), [40](#), [50](#)

set.seed, [45](#)

shrink.gof, [3](#), [7](#), [9](#), [45](#), [47](#), [49](#)

summary.edge_stream(edge.stream), [23](#)

update.edge_stream(edge.stream), [23](#)