Package: fasthurdle (via r-universe)

October 13, 2026

Type Package

Title Fast Implementation of Hurdle Models

Version 1.2.0

Date 2026-04-06

Description Provides fast hurdle regression models for count data using C++ implementations via 'Rcpp'. Supports the same functionality as the hurdle function in the 'pscl' package, with improved performance for large datasets.

License GPL-2

Encoding UTF-8

OS_type unix

Depends R (>= 4.4)

Imports fastglm (>= 0.0.4), Rcpp (>= 1.0.14)

LinkingTo Rcpp, RcppArmadillo, roptim

Suggests car, pscl, testthat (>= 3.0.0), microbenchmark

Config/testthat/edition 3

RoxygenNote 7.3.2

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

Date/Publication 2026-08-13 22:01:29 UTC

RemoteUrl https://github.com/mkanai/fasthurdle

RemoteRef HEAD

RemoteSha 05b3ded57246277404569e03cc9ce3cae55a3414

Contents

acat_stagewise 2
CCT 3
coef.fasthurdle 4
extractAIC.fasthurdle 4
fast_negbin_hurdle 5

fasthurdle 7
fit_null_count 10
fit_null_zero 11
fitted.fasthurdle 12
hurdle.control 12
hurdle_scan 13
hurdletest 14
jiang_doerge_fdr 15
joint_score_test 16
logLik.fasthurdle 17
model.matrix.fasthurdle 18
predict.fasthurdle 18
predprob.fasthurdle 19
prepare_score_cache_count 20
prepare_score_cache_zero 20
print.fasthurdle 21
print.summary.fasthurdle 22
residuals.fasthurdle 22
score_test_count 23
score_test_zero 24
summary.fasthurdle 25
terms.fasthurdle 25
vcov.fasthurdle 26

Index **27**

`acat_stagewise` *ACAT stage-wise testing for hurdle model FDR control*

Description

Combines zero-inflation and count model p-values using the Cauchy combination test (CCT/ACAT) for omnibus screening, then applies stage-wise confirmation via the Holm procedure to classify the regulatory mode. This provides overall FDR control at the screening level with family-wise error rate control within each selected pair.

Usage

```
acat_stagewise(p_zero, p_count, alpha = 0.05)
```

Arguments

`p_zero`	Numeric vector of zero-inflation model p-values.
`p_count`	Numeric vector of count model p-values.
`alpha`	Significance threshold for both screening and confirmation (default: 0.05).

Value

A data.frame with columns:

p_omnibus	CCT-combined omnibus p-value.
q_omnibus	Benjamini-Hochberg adjusted p-value for the omnibus test.
selected	Logical; TRUE if q_omnibus < alpha.
p_adj_zero	Holm-adjusted zero-model p-value (NA if not selected).
p_adj_count	Holm-adjusted count-model p-value (NA if not selected).
sig_zero	Logical; zero component significant after Holm correction.
sig_count	Logical; count component significant after Holm correction.
mode	Factor classifying the regulatory mode as "dual", "switch", "rheostat", "omnibus_only", or "not_significant".
sig	Logical; TRUE if mode is not "not_significant".

References

Van den Berge, K., et al. (2017). stageR: a general stage-wise method for controlling the gene-level false discovery rate in differential expression and differential transcript usage. *Genome Biology*, 18, 151.

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.

CCT *Cauchy Combination Test (CCT / ACAT)*

Description

Combines p-values using the Cauchy distribution. This is an analytical p-value combination method that is valid under arbitrary dependency structures between the input p-values.

Usage

```
CCT(pvals, weights = NULL)
```

Arguments

pvals	A numeric vector of p-values between 0 and 1.
weights	A numeric vector of non-negative weights. If NULL, equal weights are used.

Details

Based on the STAAR package implementation. When any p-value equals 1, the original STAAR implementation returns 1. We adopt the SAIGE-QTL modification that instead returns a Bonferroni-corrected p-value (`min(1, min(p)*n)`), which is more conservative and informative in this edge case.

Value

A single combined p-value.

References

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.

`coef.fasthurdle` *Extract Model Coefficients from a Hurdle Model*

Description

Extract the estimated coefficients from a fitted hurdle model.

Usage

```
## S3 method for class 'fasthurdle'
coef(object, model = c("full", "count", "zero"), ...)
```

Arguments

`object`	A fitted model object of class "fasthurdle".
`model`	Character string specifying which model coefficients to extract. "full" extracts all coefficients, "count" extracts count model coefficients, and "zero" extracts zero hurdle model coefficients.
`...`	Additional arguments (currently ignored).

Value

A named vector of coefficients.

`extractAIC.fasthurdle` *Extract AIC from a Hurdle Model*

Description

Extract AIC from a fitted hurdle model.

Usage

```
## S3 method for class 'fasthurdle'
extractAIC(fit, scale = NULL, k = 2, ...)
```

Arguments

fit	A fitted model object of class "fasthurdle".
scale	Scale parameter for the AIC. Not used.
k	Penalty per parameter to be used in AIC.
...	Additional arguments (currently ignored).

Value

A numeric vector of length 2, with the degrees of freedom and the AIC.

---

fast_negbin_hurdle *Fast Negative Binomial Hurdle Model with Binomial Zero Hurdle*

Description

A specialized version of fasthurdle that only handles negative binomial count model with binomial zero hurdle model and logit link. This function is optimized for speed by skipping all unnecessary parameter checks and validations.

Usage

```
fast_negbin_hurdle(
  X,
  y,
  Z = NULL,
  offsetx = NULL,
  offsetz = NULL,
  method = "BFGS",
  maxit = 10000,
  separate = TRUE,
  score_test = NULL,
  null_fit_count = NULL,
  null_fit_zero = NULL,
  spa_cutoff = NULL,
  compute_fitted = FALSE
)
```

Arguments

X	Model matrix for the count component (and zero component if Z is NULL)
y	Response vector of counts
Z	Optional model matrix for the zero component. Default is NULL (use X).
offsetx	Optional offset vector for the count model. Default is NULL (no offset).
offsetz	Optional offset vector for the zero model. Default is NULL (no offset).

method	Optimization method used by optim. Default is "BFGS".
maxit	Maximum number of iterations. Default is 10000.
separate	Logical. If TRUE, count and zero components are estimated separately. Default is TRUE.
score_test	Optional. Column name (character) of the variable to compute score tests for in both the count and zero components. When specified, neither the full count nor zero model is fitted — only the null models. See Details.
null_fit_count	Optional. A pre-fitted count null model from fit_null_count. When provided with score_test, the count null is not re-fitted.
null_fit_zero	Optional. A pre-fitted zero null model from fit_null_zero. When provided with score_test, the zero null is not re-fitted.
spa_cutoff	Numeric or NULL. When score_test is used, apply saddlepoint approximation (SPA) for p-values when |z| exceeds this cutoff. Default is NULL (disabled). Set to 2 to enable SPA, which improves tail accuracy for sparse genes at small sample sizes (n < 50K).
compute_fitted	Logical. If FALSE (default), skip computing fitted values and residuals for speed. Set to TRUE if you need fitted.values, residuals, y, or x in the returned object. Not available when score_test is used.

Details

Score test mode

When score_test is specified, the function operates differently:

- Neither the full count nor zero model is fitted — only the null models.
- The count component uses the observed information (negative Hessian) instead of the expected Fisher information, making it robust to model misspecification.
- The zero component uses the expected FIM with SPA (already well-calibrated).
- For significant tests (|z| > spa_cutoff, or |z| > 2 when SPA is disabled), beta is refined via a short BFGS optimization from the score estimate (within ~3% of the full MLE). For non-significant tests, beta uses the ratio estimator (approximate).
- Covariate coefficients are the null model MLEs (valid under H0).
- loglik is the null models' log-likelihood, not the full model's.
- vcov has NA for covariate SEs; score-based SE for the test variable.
- theta is estimated from the count null model.
- compute_fitted is not available (fitted values require the full model).

Value

An object of class "fasthurdle" representing the fitted model.

Examples

```
## Not run:
# Load example data
data(bioChemists, package = "pscl")

# Create model matrix and response vector
X <- model.matrix(~ fem + mar + kid5 + phd + ment, data = bioChemists)
y <- bioChemists$art

# Fit the model (Wald test)
m <- fast_negbin_hurdle(X, y)
summary(m)

# Fit with score test for 'ment'
m <- fast_negbin_hurdle(X, y, score_test = "ment")
summary(m)

# High-throughput: cache null model, test many variables
X_null <- model.matrix(~ fem + mar + kid5 + phd, data = bioChemists)
null_fit_count <- fit_null_count(X_null, y, dist = "negbin")
m <- fast_negbin_hurdle(X, y, score_test = "ment", null_fit_count = null_fit_count)
summary(m)

## End(Not run)
```

`fasthurdle` *Fit Hurdle Regression Models for Count Data*

Description

Fits hurdle regression models for count data. The hurdle model combines a count data model (such as Poisson, negative binomial, or geometric) for positive counts with a hurdle component that models the zero counts. This implementation uses C++ for the core computations, making it faster than the original pscl::hurdle implementation.

Usage

```
fasthurdle(
  formula,
  data,
  subset,
  na.action,
  weights,
  offset,
  dist = c("poisson", "negbin", "geometric"),
  zero.dist = c("binomial", "poisson", "negbin", "geometric"),
  link = c("logit", "probit", "cloglog", "cauchit", "log"),
```

```
  control = hurdle.control(...),
  score_test = NULL,
  null_fit_count = NULL,
  null_fit_zero = NULL,
  spa_cutoff = NULL,
  model = TRUE,
  y = TRUE,
  x = FALSE,
  ...
)
```

Arguments

formula	A formula expression of the form y ~ x | z where y is the response and x and z are regressor variables for the count and zero components, respectively. If the | operator is not specified, the same regressors are used for both components.
data	An optional data frame containing the variables in the model.
subset	An optional vector specifying a subset of observations to be used.
na.action	A function which indicates what should happen when the data contain NAs.
weights	An optional vector of weights to be used in the fitting process.
offset	An optional offset for the count model. Can also be specified via the offset argument in the formula.
dist	Character string specifying the count distribution. Currently, "poisson", "negbin" (negative binomial), and "geometric" are supported.
zero.dist	Character string specifying the zero hurdle distribution. Currently, "binomial", "poisson", "negbin" (negative binomial), and "geometric" are supported.
link	Character string specifying the link function for the binomial zero hurdle model. Currently, "logit", "probit", "cloglog", "cauchit", and "log" are supported.
control	A list of control parameters passed to the optimizer. See hurdle.control.
score_test	Optional. Column name of the variable to compute a score test for in both the count and zero components. The score test evaluates significance at the null MLE, giving better-calibrated p-values and faster computation. For significant tests, beta is refined via a short BFGS optimization from the score estimate. See Details.
null_fit_count	Optional. A pre-fitted count null model from fit_null_count. When provided with score_test, the count null is not re-fitted.
null_fit_zero	Optional. A pre-fitted zero null model from fit_null_zero. When provided with score_test, the zero null is not re-fitted. Only supported with zero.dist = "binomial" and link = "logit".
spa_cutoff	Numeric or NULL. When score_test is used, apply saddlepoint approximation (SPA) for p-values when |z| exceeds this cutoff. Default is NULL (disabled). Set to 2 to enable SPA for improved tail accuracy at small sample sizes (n < 50K).
model	Logical. If TRUE, the model frame is included in the returned object.
y	Logical. If TRUE, the response vector is included in the returned object.
x	Logical. If TRUE, the model matrices are included in the returned object.
...	Additional arguments passed to hurdle.control.

Details

The hurdle model combines two components: a truncated count component for positive counts and a zero hurdle component that models the zeros. The probability mass function is given by:

$$f(y) = \begin{cases} f_{zero}(0) & \text{if } y = 0 \\ (1 - f_{zero}(0)) \cdot \frac{f_{count}(y)}{1 - f_{count}(0)} & \text{if } y > 0 \end{cases}$$

where f_{zero} is the zero hurdle distribution and f_{count} is the count distribution.

Score test mode

When `score_test` is specified, the full count model is still fitted (unlike `fast_negbin_hurdle` which skips the full model for speed). The score test results are stored in the `$score_test_count` and `$score_test_zero` slots alongside the standard Wald results. This allows comparison between the two testing approaches.

The count score test uses the observed information (negative Hessian) instead of the expected Fisher information, making it robust to model misspecification. Available for all count distributions (negbin, poisson, geometric). The zero score test uses the expected FIM with SPA (binomial/logit only), which is already well-calibrated.

For significant tests (|z| > `spa_cutoff`, or |z| > 2 when SPA is disabled), beta is refined via 5-iteration BFGS from the score estimate (within ~3% of the full MLE). SE is back-computed from the p-value for consistency.

Value

An object of class "fasthurdle" representing the fitted model. When `score_test` is used, `$score_test_count` and `$score_test_zero` contain the score test results (beta, se, statistic, pvalue) for each component.

Examples

```
## Not run:
# Load example data
data(bioChemists, package = "pscl")

# Fit a hurdle model with Poisson count component and binomial zero component
m1 <- fasthurdle(art ~ fem + mar + kid5 + phd + ment | fem + mar + kid5 + phd + ment,
  data = bioChemists, dist = "poisson", zero.dist = "binomial"
)
summary(m1)

# Fit a hurdle model with negative binomial count component
m2 <- fasthurdle(art ~ fem + mar + kid5 + phd + ment | fem + mar + kid5 + phd + ment,
  data = bioChemists, dist = "negbin", zero.dist = "binomial"
)
summary(m2)

## End(Not run)
```

`fit_null_count` *Fit a Null Count Model for Score Testing*

Description

Fits the null (reduced) count model for use with `score_test_count`. The fitted null can be cached (e.g., via `saveRDS`) and reused across multiple test variables, avoiding redundant model fitting.

Usage

```
fit_null_count(
  X_null,
  y,
  offsetx = NULL,
  weights = NULL,
  dist = c("negbin", "poisson", "geometric"),
  method = "BFGS",
  maxit = 10000
)
```

Arguments

`X_null`	Model matrix for the null model (intercept + covariates, no test variable).
`y`	Response vector of counts.
`offsetx`	Optional offset vector. Default is NULL (no offset).
`weights`	Optional weight vector. Default is NULL (unit weights).
`dist`	Count distribution: "negbin", "poisson", or "geometric".
`method`	Optimization method. Default is "BFGS".
`maxit`	Maximum iterations. Default is 10000.

Value

An object of class "fasthurdle_null" containing the null MLE parameters, convergence status, and metadata needed for score testing. This object can be saved with `saveRDS` and reloaded for reuse.

Examples

```
## Not run:
# Fit null model once per gene (covariates only)
null_fit_count <- fit_null_count(X_null, y, offsetx = off, dist = "negbin")
saveRDS(null_fit_count, "null_fit_gene1.rds")

# Reuse for each peak
null_fit_count <- readRDS("null_fit_gene1.rds")
for (peak in peaks) {
  model <- fast_negbin_hurdle(X, y,
```

```
    offsetx = off, score_test = "peak_acc",
    null_fit_count = null_fit_count
  )
  results[[peak]] <- model$score_test_count$pvalue
}

## End(Not run)
```

`fit_null_zero` *Fit Null Zero Model for Caching*

Description

Fits the zero model (binomial/logit) without the test variable, for reuse across multiple score tests on the same gene. Only logit link is supported.

Usage

```
fit_null_zero(
  Z_null,
  y,
  offsetz = NULL,
  weights = NULL,
  method = "BFGS",
  maxit = 10000
)
```

Arguments

`Z_null`	Zero model design matrix without the test variable.
`y`	Response vector of counts.
`offsetz`	Optional offset vector for zero model. Default is NULL.
`weights`	Optional weight vector. Default is NULL (unit weights).
`method`	Optimization method. Default is "BFGS".
`maxit`	Maximum iterations. Default is 10000.

Value

An object of class "fasthurdle_null_zero" for use with `fast_negbin_hurdle(..., null_fit_zero = ...)`.

fitted.fasthurdle *Extract Fitted Values from a Hurdle Model*

Description

Extract fitted values from a fitted hurdle model.

Usage

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

Arguments

object	A fitted model object of class "fasthurdle".
...	Additional arguments (currently ignored).

Value

A vector of fitted values.

hurdle.control *Control Parameters for Hurdle Models*

Description

Set control parameters for fitting hurdle models.

Usage

```
hurdle.control(
  method = "BFGS",
  maxit = 10000,
  trace = FALSE,
  separate = TRUE,
  start = NULL,
  ...
)
```

Arguments

method	Optimization method used by optim. Default is "BFGS".
maxit	Maximum number of iterations. Default is 10000.
trace	Logical. If TRUE, information about the fitting process is printed. Default is FALSE.
separate	Logical. If TRUE, count and zero components are estimated separately. Default is TRUE.
start	Optional list with starting values for the parameters.
...	Additional control parameters passed to optim.

Value

A list of control parameters.

---

hurdle_scan *Batch Score Test Scan for Hurdle Models*

Description

Tests multiple peaks (test variables) against a hurdle model in a single call. This is the primary production entry point for high-throughput cis-QTL mapping.

P-values are returned as `-log10(p)` to prevent underflow at genome-wide significance levels.

Usage

```
hurdle_scan(
  X,
  y,
  peaks,
  Z = NULL,
  offsetx = NULL,
  offsetz = NULL,
  null_fit_count = NULL,
  null_fit_zero = NULL,
  spa_cutoff = NULL
)
```

Arguments

X	Model matrix (n x p) containing both null covariates and test variables.
y	Response vector of counts.
peaks	Character vector of column names in X to test (required).
Z	Optional model matrix for the zero component. Default is NULL (use X).
offsetx	Optional offset vector for the count model. Default is NULL.

offsetz	Optional offset vector for the zero model. Default is NULL.
null_fit_count	Optional pre-fitted count null model (from fit_null_count).
null_fit_zero	Optional pre-fitted zero null model (from fit_null_zero).
spa_cutoff	Numeric or NULL. Apply saddlepoint approximation when |z| exceeds this cutoff. Default is NULL (disabled).

Value

A data.frame with columns:

peak	Peak/test variable name.
nlog10p_count	-log10(p-value) for the count component.
beta_count	Effect size estimate for the count component.
se_count	Standard error for the count component.
stat_count	Test statistic for the count component.
nlog10p_zero	-log10(p-value) for the zero component (if test vars are found in Z).
beta_zero	Effect size estimate for the zero component.
se_zero	Standard error for the zero component.
stat_zero	Test statistic for the zero component.

Attributes null_fit_count and null_fit_zero are attached for reuse across multiple genes.

Examples

```
## Not run:
# Standard batch score test
result <- hurdle_scan(X, y, peaks = c("peak1", "peak2", "peak3"))

## End(Not run)
```

---

hurdletest *Test for Equality of Count and Zero Model Coefficients*

Description

Test for equality of count and zero model coefficients in a hurdle model.

Usage

```
hurdletest(object, ...)
```

Arguments

object	A fitted model object of class "fasthurdle".
...	Additional arguments passed to car::linearHypothesis.

Value

An object of class "anova" containing the results of the test.

`jiang_doerge_fdr` *Jiang-Doerge two-stage FDR procedure for hurdle models*

Description

Performs a two-stage FDR procedure where stage 1 screens on one set of p-values (e.g., zero-inflation) and stage 2 confirms using a second set (e.g., count model), with FDR adjustment accounting for the selection step.

Usage

```
jiang_doerge_fdr(p_stage1, p_stage2, alpha1 = 0.1, alpha2 = 0.05)
```

Arguments

`p_stage1`	Numeric vector of stage 1 p-values (screening).
`p_stage2`	Numeric vector of stage 2 p-values (confirmation).
`alpha1`	FDR threshold for stage 1 screening (default: 0.1).
`alpha2`	FDR threshold for stage 2 confirmation (default: 0.05).

Value

A list with components:

`selected`	Logical vector of pairs passing stage 1.
`significant`	Logical vector of pairs significant after both stages.
`n_selected`	Number of pairs selected in stage 1.
`n_significant`	Number of significant pairs.
`pi0_selected`	Estimated proportion of true nulls among selected.
`alpha2_adjusted`	Adjusted alpha for stage 2.
`q_stage1`	BH-adjusted q-values from stage 1.
`q_stage2`	BH-adjusted q-values from stage 2 (1 for unselected).

References

Jiang, H. & Doerge, R.W. (2006). A two-step multiple comparison procedure for a large number of tests and multiple treatments. *Statistical Applications in Genetics and Molecular Biology*, 5, Article28.

`joint_score_test` *Joint 2-df score test with stage-wise mode classification*

Description

Combines zero and count model score test statistics via a joint chi-squared(2) test for omnibus screening, then applies Holm step-down for mode classification. Under the factorized hurdle likelihood, the zero and count score statistics are independent, so the joint statistic is simply their sum: T_joint = T_zero + T_count.

Usage

```
joint_score_test(
  p_zero = NULL,
  p_count = NULL,
  chisq_zero = NULL,
  chisq_count = NULL,
  alpha = 0.05
)
```

Arguments

`p_zero`	Numeric vector of zero-model p-values (default positional input). Converted internally to chi-squared(1) statistics via `qchisq(p, df = 1, lower.tail = FALSE)`.
`p_count`	Numeric vector of count-model p-values.
`chisq_zero`	Numeric vector of zero-model chi-squared statistics (1 df). From `score_test_zero()$statistic` or `summary()$coefficients$zero[,"z value"]^2`. If provided, `p_zero`/`p_count` must be `NULL`.
`chisq_count`	Numeric vector of count-model chi-squared statistics (1 df). From `score_test_count()$statistic` or `summary()$coefficients$count[,"z value"]^2`.
`alpha`	Significance threshold for both screening and confirmation (default: 0.05).

Details

By default accepts p-values (positional arguments). Chi-squared statistics can be supplied instead via `chisq_zero`/`chisq_count`; this avoids numerical underflow for very small p-values.

Value

A data.frame with columns:

`chisq_joint`	Joint chi-squared(2) statistic (T_zero + T_count).
`p_joint`	Joint p-value from chi-squared(2) distribution.
`q_joint`	Benjamini-Hochberg adjusted p-value for the joint test.
`selected`	Logical; TRUE if q_joint < alpha.

p_adj_zero	Holm-adjusted zero-model p-value (NA if not selected).
p_adj_count	Holm-adjusted count-model p-value (NA if not selected).
sig_zero	Logical; zero component significant after Holm correction.
sig_count	Logical; count component significant after Holm correction.
mode	Factor classifying the regulatory mode as "dual", "switch", "rheostat", "omnibus_only", or "not_significant".
sig	Logical; TRUE if mode is not "not_significant".

References

Van den Berge, K., et al. (2017). stageR: a general stage-wise method for controlling the gene-level false discovery rate in differential expression and differential transcript usage. *Genome Biology*, 18, 151.

---

logLik.fasthurdle *Extract Log-Likelihood from a Hurdle Model*

Description

Extract the log-likelihood from a fitted hurdle model.

Usage

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

Arguments

object	A fitted model object of class "fasthurdle".
...	Additional arguments (currently ignored).

Value

An object of class "logLik".

model.matrix.fasthurdle *Extract Model Matrix from a Hurdle Model*

Description

Extract the model matrix from a fitted hurdle model.

Usage

```
## S3 method for class 'fasthurdle'
model.matrix(object, model = c("count", "zero"), ...)
```

Arguments

object	A fitted model object of class "fasthurdle".
model	Character string specifying which model matrix to extract. "count" extracts count model matrix, and "zero" extracts zero hurdle model matrix.
...	Additional arguments (currently ignored).

Value

A model matrix.

predict.fasthurdle *Predict Method for Hurdle Models*

Description

Predict values from a fitted hurdle model.

Usage

```
## S3 method for class 'fasthurdle'
predict(
  object,
  newdata,
  type = c("response", "prob", "count", "zero"),
  na.action = na.pass,
  at = NULL,
  ...
)
```

Arguments

object	A fitted model object of class "fasthurdle".
newdata	An optional data frame containing the variables needed for prediction.
type	Character string specifying the type of prediction. "response" returns the expected value, "prob" returns the probability mass function, "count" returns the expected value of the count component, and "zero" returns the expected value of the zero hurdle component.
na.action	Function determining what to do with missing values in newdata.
at	Optional vector of counts for which to compute probabilities.
...	Additional arguments (currently ignored).

Value

A vector or matrix of predictions.

---

predprob.fasthurdle *Predict Probabilities from a Hurdle Model*

Description

Predict probabilities from a fitted hurdle model.

Usage

```
predprob.fasthurdle(obj, ...)
```

Arguments

obj	A fitted model object of class "fasthurdle".
...	Additional arguments passed to predict.fasthurdle.

Value

A matrix of predicted probabilities.

prepare_score_cache_count

Prepare cached quantities for fast per-peak count score tests

Description

Pre-computes null-only Hessian weights, FIM inverse, and SPA intermediates. When stored in the null_fit_count object, subsequent score_test_count calls use the cached quantities to avoid redundant O(n_pos) computation per peak. Supports all count distributions (negbin, poisson, geometric) via ZTNB formulas with appropriate theta parameterization.

Usage

```
prepare_score_cache_count(
  null_fit_count,
  y,
  X_null,
  offsetx = NULL,
  weights = NULL
)
```

Arguments

null_fit_count	Fitted null model from fit_null_count.
y	Response vector.
X_null	Null model matrix.
offsetx	Offset vector.
weights	Weight vector.

Value

The null_fit_count object with an attached score_cache element.

prepare_score_cache_zero

Prepare cached quantities for fast per-peak zero score tests

Description

Pre-computes null FIM inverse, score weights, and SPA intermediates for the zero (binomial/logistic) component. Analogous to prepare_score_cache_count for the count component.

Usage

```
prepare_score_cache_zero(
  null_fit_zero,
  y,
  Z_null,
  offsetz = NULL,
  weights = NULL
)
```

Arguments

null_fit_zero	Fitted null model from fit_null_zero.
y	Response vector.
Z_null	Null zero model matrix.
offsetz	Offset vector.
weights	Weight vector.

Value

The null_fit_zero object with an attached score_cache element.

---

print.fasthurdle *Print Method for Hurdle Models*

Description

Print a summary of a fitted hurdle model.

Usage

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

Arguments

x	A fitted model object of class "fasthurdle".
digits	Number of significant digits to use for printing.
...	Additional arguments passed to print methods.

Value

The fitted model object (invisibly).

print.summary.fasthurdle *Print Method for Summary of Hurdle Models*

Description

Print a summary of a fitted hurdle model.

Usage

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

Arguments

x	An object of class "summary.fasthurdle".
digits	Number of significant digits to use for printing.
...	Additional arguments passed to print methods.

Value

The summary object (invisibly).

residuals.fasthurdle *Extract Residuals from a Hurdle Model*

Description

Extract residuals from a fitted hurdle model.

Usage

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

Arguments

object	A fitted model object of class "fasthurdle".
type	Character string specifying the type of residuals. "pearson" returns Pearson residuals, and "response" returns response residuals.
...	Additional arguments (currently ignored).

Value

A vector of residuals.

score_test_count *Score test for the count component of a hurdle model*

Description

Tests whether a single test variable has a significant effect on the count component, using a score test with observed information. Uses pre-cached null quantities for O(n_pos) per-peak computation. Supports all count distributions (negbin, poisson, geometric).

Usage

```
score_test_count(
  X_null,
  x_test,
  y,
  offsetx = NULL,
  weights = NULL,
  dist = c("negbin", "poisson", "geometric"),
  null_fit_count = NULL,
  spa_cutoff = NULL,
  method = "BFGS",
  maxit = 10000
)
```

Arguments

X_null	Null model matrix (n x kx_null).
x_test	Test variable vector (length n).
y	Response vector.
offsetx	Offset vector (default: zeros).
weights	Weight vector (default: ones).
dist	Count distribution: "negbin", "poisson", or "geometric".
null_fit_count	Fitted null model from fit_null_count. If NULL, fitted internally.
spa_cutoff	Saddlepoint approximation cutoff. NULL or Inf disables SPA.
method	Optimization method for null model fitting (default: "BFGS").
maxit	Maximum iterations for null model fitting.

Value

A list with components beta, se, statistic, pvalue, spa_applied, null_par, null_convergence.

score_test_zero *Score Test for Zero (Binomial/Logit) Component*

Description

Score Test for Zero (Binomial/Logit) Component

Usage

```
score_test_zero(
  Z_null,
  z_test,
  y,
  offsetz = NULL,
  weights = NULL,
  null_fit_zero = NULL,
  spa_cutoff = NULL,
  method = "BFGS",
  maxit = 10000
)
```

Arguments

Z_null	Zero model design matrix without the test variable.
z_test	Test variable vector or single-column matrix.
y	Response vector.
offsetz	Optional offset vector. Default is NULL.
weights	Optional weight vector. Default is NULL.
null_fit_zero	Optional cached null zero model from fit_null_zero.
spa_cutoff	SPA cutoff. Default is NULL (disabled). Set to 2 to enable SPA for improved tail accuracy at small sample sizes.
method	Optimization method for null model. Default is "BFGS".
maxit	Maximum iterations for null model. Default is 10000.

Value

A list with beta, se, statistic, pvalue, spa_applied.

summary.fasthurdle *Summary Method for Hurdle Models*

Description

Compute a summary of a fitted hurdle model.

Usage

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

Arguments

object	A fitted model object of class "fasthurdle".
...	Additional arguments (currently ignored).

Value

An object of class "summary.fasthurdle".

terms.fasthurdle *Extract Terms from a Hurdle Model*

Description

Extract the terms object from a fitted hurdle model.

Usage

```
## S3 method for class 'fasthurdle'
terms(x, model = c("count", "zero"), ...)
```

Arguments

x	A fitted model object of class "fasthurdle".
model	Character string specifying which model terms to extract. "count" extracts count model terms, and "zero" extracts zero hurdle model terms.
...	Additional arguments (currently ignored).

Value

A terms object.

`vcov.fasthurdle` *Extract Variance-Covariance Matrix from a Hurdle Model*

Description

Extract the variance-covariance matrix from a fitted hurdle model.

Usage

```
## S3 method for class 'fasthurdle'
vcov(object, model = c("full", "count", "zero"), ...)
```

Arguments

`object`	A fitted model object of class "fasthurdle".
`model`	Character string specifying which model coefficients to extract. "full" extracts the full variance-covariance matrix, "count" extracts the count model variance-covariance matrix, and "zero" extracts the zero hurdle model variance-covariance matrix.
`...`	Additional arguments (currently ignored).

Value

A variance-covariance matrix.

Index

acat_stagewise, 2

CCT, 3
coef.fasthurdle, 4

extractAIC.fasthurdle, 4

fast_negbin_hurdle, 5, *9*
fasthurdle, 7
fit_null_count, *6*, *8*, 10, *14*, *20*, *23*
fit_null_zero, *6*, *8*, 11, *14*, *21*
fitted.fasthurdle, 12

hurdle.control, *8*, 12
hurdle_scan, 13
hurdletest, 14

jiang_doerge_fdr, 15
joint_score_test, 16

logLik.fasthurdle, 17

model.matrix.fasthurdle, 18

predict.fasthurdle, 18
predprob.fasthurdle, 19
prepare_score_cache_count, 20, *20*
prepare_score_cache_zero, 20
print.fasthurdle, 21
print.summary.fasthurdle, 22

residuals.fasthurdle, 22

score_test_count, *10*, 23
score_test_zero, 24
summary.fasthurdle, 25

terms.fasthurdle, 25

vcov.fasthurdle, 26