Package 'CanonicalFamilyExtra'

May 6, 2026

Type Package

Title Extra Canonical Link Family Objects for Generalized Linear Models

Version 1.0.0

Description Extra family objects in ``weird'' scenarios, particularly logistic or log-linear model with unbounded or non-binary/non-integer outcomes. Provides binomial_extra() and poisson_extra() as generalizations of binomial() and poisson(). The use of canonical link with the corresponding working likelihood in glm() ensures convexity, making model fitting reliable and independent of starting value. Robert WM Wedderburn (1974) <doi:10.1093/biomet/61.3.439> and Peter McCullagh (1983) <doi:10.1214/aos/1176346056> justified this method to fit generalized linear (mean) models with quasi-/working likelihood.

License GPL (>= 2)

Imports stats

Suggests gee, geepack, SuperLearner, ipred, glmnet

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Hongxiang Qiu [aut, cre]

Maintainer Hongxiang Qiu <david940408@gmail.com>

Repository CRAN

Date/Publication 2026-05-06 20:30:17 UTC

Contents

`binomial_extra` *Family object for fitting binomial (e.g., logistic, probit) regression model with potentially unbounded continuous outcome*

Description

A family object for fitting binomial models (i.e., generalized linear models with range contained in the open unit interval $(0, 1)$) with continuous outcomes that may fall outside the unit interval $[0, 1]$. Also works with `glmnet::glmnet()` as well as `SL.glmnet.extra()` and `screen.glmnet.extra()`. Generally, the object aims to fit a model ranged in $[0, 1]$.

Usage

```
binomial_extra(
  link = "logit",
  variance = "mu(1-mu)",
  family = c("gaussian", "binomial")
)
```

Arguments

`link`	see `stats::family()`. Default to `"logit"`. When `variance="mu(1-mu)"`, use with care the links incompatible with the range $[0, 1]$, namely `"log"`, `"identity"`, `"sqrt"`, `"inverse"`, and `"1/mu^2"`.
`variance`	see `stats::family()`. Default to `"mu(1-mu)"`, same as logistic regression. `"constant"` (Gaussian working mean-variance relationship) also works. Other `variance` might lead to unexpected errors.
`family`	The family of the returned family object. Either `"gaussian"` or `"binomial"`. Default to `"gaussian"`. Does not to matter for `stats::glm()`. See details below when running GEE.

Details

This family is useful, for example, when the estimand is a conditional probability function while the outcome is a transformed pseudo-outcome so that the estimator is multiply robust, or estimating a regression function with known bounds while the outcome might not respect the known bounds. Naive approaches such as `glm(family=binomial())`, `glm(family=quasibinomial())`, `glm(family=gaussian(link="logit"))`, `glm(family=quasi(link="logit",variance="constant"))` etc. might not work appropriately in such cases.

Particularly for logistic model, because of using the binomial working likelihood and its canonical link, the model fitting is a convex problem and does not depend on starting value.

The output has `family="gaussian"` by default to be compatible with other learners in `SuperLearner::SuperLearner()`, because when the outcome is continuous, other learners might not perform correctly with `family="binomial"`.

When running `geepack::geeglm()` or `gee::gee()` with `binomial_extra`, the working mean-variance relationship is determined by the `family` of `binomial_extra`. Need to specify `family="binomial"` to use binomial working mean-variance relationship. The default `"gaussian"` will lead to Gaussian working mean-variance relationship.

Value

a family object

Examples

```
set.seed(123)
expit <- binomial()$linkinv

# glm
x <- rnorm(100)
y <- expit(1 + x) + rnorm(100)
glm(y~x, family = binomial_extra()) # or family=binomial_extra, or family="binomial_extra"

# Counterexamples: These naive approaches yield errors or are not reliable
try(glm(y~x, family = binomial()))
try(glm(y~x, family = quasibinomial()))
try(glm(y~x, family = gaussian(link = "logit")))
# setting starting value might fix this approach,
# but the problem is non-convex and requires valid starting value
try(glm(y~x, family = gaussian(link = "logit"), start = c(-1,0)))
try(glm(y~x, family = quasi(link = "logit", variance = "constant")))

#glmnet
X <- matrix(rnorm(100 * 5), nrow = 100)
y <- expit(1 + X[,1]) + rnorm(100)
require(glmnet)
glmnet(X, y, family = binomial_extra())
# or family=binomial_extra; cannot use family="binomial_extra"

# Counterexamples: These naive approaches yield errors
try(glmnet(X, y, family = binomial()))
try(glmnet(X, y, family = gaussian(link = "logit")))
try(glmnet(X, y, family = quasi(link = "logit", variance = "constant")))

# other links/variance for glm
x <- rnorm(100)
y <- expit(1 + x) + rnorm(100)
glm(y~x, family = binomial_extra("probit")) # probit regression
glm(y~x, family = binomial_extra(variance = "constant")) # least squares

# within SuperLearner
X <- matrix(rnorm(100 * 3), nrow = 100)
y <- expit(1 + X[,1]) + rnorm(10)
require(SuperLearner)
SuperLearner(y, data.frame(X), family=binomial_extra(),
             SL.library = c("SL.glm", "SL.ipredbagg"), cvControl = list(V = 2))

#GEE
x <- rnorm(100)
y <- expit(1 + x) + rnorm(100)
id <- rep(1:20, each=5)
geepack::geeglm(y ~ x, family = binomial_extra(family = "binomial"), id = id)
```

```
gee::gee(y ~ x, family = binomial_extra(family = "binomial"), id = id)
```

`poisson_extra` *Family object for fitting log-linear model with potentially unbounded continuous outcome, particularly with Poisson working likelihood*

Description

A family object for fitting Poisson models (i.e., generalized linear models with range contained in the open unit interval $(0, \infty)$) with continuous outcomes that may not be non-negative integers. Also works with `glmnet::glmnet()` as well as `SL.glmnet.extra()` and `screen.glmnet.extra()`. Generally, the object aims to fit a model ranged in $(0, \infty)$.

Usage

```
poisson_extra(link = "log", variance = "mu", family = c("gaussian", "poisson"))
```

Arguments

`link`	see `stats::family()`. Default to `"log"`. When `variance="mu"`, use with care the links incompatible with the range $(0, \infty)$, namely `"identity"`, `"sqrt"`, `"inverse"`, and `"1/mu^2"`.
`variance`	see `stats::family()`. Default to `"mu"`, same as Poisson regression. `"constant"` (Gaussian working mean-variance relationship) also works. Other `variance` might lead to unexpected errors.
`family`	The family of the returned family object. Either `"gaussian"` or `"poisson"`. Default to `"gaussian"`. Does not to matter for `stats::glm()`. See details below when running GEE.

Details

This family is useful, for example, when the estimand is a conditional probability function while the outcome is a transformed pseudo-outcome so that the estimator is multiply robust, or estimating a positive regression function while the outcome might be negative or non-integers. Naive approaches such as `glm(family=poisson())`, `glm(family=quasipoisson())`, `glm(family=gaussian(link="log"))`, `glm(family=quasi(link="log",variance="constant"))` etc. might not work appropriately or reliably in such cases.

Particularly for log-linear model, because of using the Poisson working likelihood and its canonical link, the model fitting is a convex problem and does not depend on starting value.

The output has `family="gaussian"` by default to be compatible with other learners in `SuperLearner::SuperLearner()`, because when the outcome is continuous, other learners might not perform correctly with `family="poisson"`.

When running `geepack::geeglm()` or `gee::gee()` with `poisson_extra`, the working mean-variance relationship is determined by the `family` of `poisson_extra`. Need to specify `family="poisson"` to use Poisson working mean-variance relationship. The default `"gaussian"` will lead to Gaussian working mean-variance relationship.

Value

a family object

Examples

```
set.seed(123)

# glm
x <- rnorm(100)
y <- exp(-1 + x) + rnorm(100)
glm(y~x, family = poisson_extra()) # or family=poisson_extra, or family="poisson_extra"

# Counterexamples: These naive approaches yield errors or are not reliable
try(glm(y~x, family = poisson()))
try(glm(y~x, family = quasipoisson()))
try(glm(y~x, family = gaussian(link = "log")))
# setting starting value might fix this approach,
# but the problem is non-convex and requires valid starting value
try(glm(y~x, family = gaussian(link = "log"), start = c(-1,0)))
try(glm(y~x, family = quasi(link = "log", variance = "constant")))
# setting starting value might fix this approach,
# but the problem is non-convex and requires valid starting value
try(glm(y~x, family = quasi(link = "log", variance = "constant"), start = c(-1,0)))

#glmnet
X <- matrix(rnorm(100 * 5), nrow = 100)
y <- exp(1 + X[,1]) + rnorm(100)
require(glmnet)
glmnet(X, y, family = poisson_extra())
# or family=poisson_extra; cannot use family="poisson_extra"

# Counterexamples: These naive approaches yield errors or are not reliable
try(glmnet(X, y, family = poisson()))
try(glmnet(X, y, family = gaussian(link = "log")))
try(glmnet(X, y, family = quasi(link = "log", variance = "constant")))

# within SuperLearner
X <- matrix(rnorm(100 * 3), nrow = 100)
y <- exp(1 + X[,1]) + rnorm(10)
require(SuperLearner)
SuperLearner(y, data.frame(X), family=poisson_extra(),
             SL.library = c("SL.glm", "SL.ipredbagg"), cvControl = list(V = 2))

#GEE
x <- rnorm(100)
y <- exp(-1 + x) + rnorm(100)
id <- rep(1:20, each=5)
geepack::geeglm(y ~ x, family = poisson_extra(family = "poisson"), id = id)
gee::gee(y ~ x, family = poisson_extra(family = "poisson"), id = id)
```

screen.glmnet.extra *SuperLearner screener using cv.glmnet that works with the extra families*

Description

A wrapper of `glmnet::cv.glmnet()` similar to `SuperLearner::screen.glmnet()`, except that `family` can be `binomial_extra`. `SuperLearner::screen.glmnet()` only passes the name of `family` (e.g., "gaussian", "binomial") and thus cannot pass the full customized families like `binomial_extra()`.

Usage

```
screen.glmnet.extra(
  Y,
  X,
  family,
  alpha = 1,
  minscreen = 2,
  nfolds = 10,
  nlambda = 100,
  ...
)
```

Arguments

`Y`	see `SuperLearner::screen.glmnet()`
`X`	see `SuperLearner::screen.glmnet()`
`family`	similar to `SuperLearner::screen.glmnet()`. Unlike `SuperLearner::screen.glmnet()`, the entire family object rather than the name of `family` (e.g., "gaussian", "binomial") will be passed to `glmnet::cv.glmnet()`
`alpha`	see `SuperLearner::screen.glmnet()`
`minscreen`	see `SuperLearner::screen.glmnet()`
`nfolds`	see `SuperLearner::screen.glmnet()`
`nlambda`	see `SuperLearner::screen.glmnet()`
`...`	see `SuperLearner::screen.glmnet()`

Value

A logical vector with the length equal to the number of columns in X. TRUE indicates the variable (column of X) should be included. Similar to the value of `SuperLearner::screen.glmnet()`.

Examples

```
set.seed(321)
expit <- binomial()$linkinv
X <- matrix(rnorm(100 * 5), nrow = 100)
y <- expit(1 + X[,1] + X[,2]) + rnorm(100)
require(SuperLearner)
SL.library<-list(c("SL.glmnet.extra","All"),
                 c("SL.glmnet.extra","screen.glmnet.extra"))
SuperLearner(y, data.frame(X), family=binomial_extra(),
             SL.library = SL.library, cvControl = list(V = 2))
```

`SL.glmnet.extra` *SuperLearner wrapper for cv.glmnet that works with the extra families*

Description

A wrapper of `glmnet::cv.glmnet()` similar to `SuperLearner::SL.glmnet()`, except that `family` can be `binomial_extra`. `SuperLearner::SL.glmnet()` only passes the name of `family` and thus cannot pass the full customized families like `binomial_extra()`.

Usage

```
SL.glmnet.extra(
  Y,
  X,
  newX,
  family,
  obsWeights,
  id,
  alpha = 1,
  nfolds = 10,
  nlambda = 100,
  useMin = TRUE,
  loss = "deviance",
  ...
)
```

Arguments

`Y`	see `SuperLearner::SL.glmnet()`
`X`	see `SuperLearner::SL.glmnet()`
`newX`	see `SuperLearner::SL.glmnet()`
`family`	similar to `SuperLearner::SL.glmnet()`. Unlike `SuperLearner::SL.glmnet()`, the entire family object rather than the name of `family` (e.g., "gaussian", "binomial") will be passed to `glmnet::cv.glmnet()`

obsWeights	see SuperLearner::SL.glmnet()
id	see SuperLearner::SL.glmnet()
alpha	see SuperLearner::SL.glmnet()
nfolds	see SuperLearner::SL.glmnet()
nlambda	see SuperLearner::SL.glmnet()
useMin	see SuperLearner::SL.glmnet()
loss	see SuperLearner::SL.glmnet()
...	see SuperLearner::SL.glmnet()

Value

A list with two elements:

pred The predicted values for the rows in newX.

fit A list. Contains all objects necessary to get predictions for new observations from specific algorithm.

Similar to the value of SuperLearner::SL.glmnet().

Examples

```
set.seed(321)
expit <- binomial()$linkinv
X <- matrix(rnorm(100 * 5), nrow = 100)
y <- expit(1 + X[,1] + X[,2]) + rnorm(100)
require(SuperLearner)
SL.library<-list(c("SL.glmnet.extra","All"),
                 c("SL.glmnet.extra","screen.glmnet.extra"))
SuperLearner(y, data.frame(X), family=binomial_extra(),
             SL.library = SL.library, cvControl = list(V = 2))
```

Index