

Package ‘assessor’

August 22, 2026

Title Assessment Tools for Regression Models with Discrete and
Semicontinuous Outcomes

Version 1.3.2

Description Provides assessment tools for regression models with discrete and
semicontinuous outcomes. The implemented methods are described in
Yang (2021) <[doi:10.1080/10618600.2021.1910042](https://doi.org/10.1080/10618600.2021.1910042)>,
Yang (2024) <[doi:10.1080/10618600.2024.2303336](https://doi.org/10.1080/10618600.2024.2303336)>,
Yang (2024) <[doi:10.1093/biomtc/ujae007](https://doi.org/10.1093/biomtc/ujae007)>, and
Yang (2026) <[doi:10.1002/cjs.70046](https://doi.org/10.1002/cjs.70046)>. It calculates double probability integral trans-
form (DPIT) residuals and constructs QQ plots, ordered curves, quasi-empirical residual distribu-
tion functions, and formal goodness-of-fit tests.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://jhlee1408.github.io/assessor/>

BugReports <https://github.com/jhlee1408/assessor/issues>

Imports tweedie, MASS, VGAM, np, pscl

Suggests statmod, rmarkdown, knitr, AER, testthat (>= 3.0.0), mgcv

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

NeedsCompilation no

Author Lu Yang [aut],
Jeonghwan Lee [cre, aut]

Maintainer Jeonghwan Lee <lee03938@umn.edu>

Repository CRAN

Date/Publication 2026-08-22 04:00:02 UTC

Contents

bballHR	2
dpit	4
dpit-methods	8
dpit_2pm	9
dpit_bin	11
dpit_nb	12
dpit_ordi	13
dpit_pois	14
dpit_tobit	15
dpit_tweedie	17
dpit_znb	18
dpit_zpois	19
gof_disc	21
LGPIF	22
MEPS	24
ord_curve	25
quasi_plot	27
Index	30

bballHR	<i>MLB Players' Home Run and Batted Ball Statistics with Red Zone Metrics (2017-2019)</i>
---------	---

Description

This dataset provides annual statistics for Major League Baseball (MLB) players, including home run counts, at-bats, mean exit velocities, launch angles, quantile statistics of exit velocities and launch angles, and red zone metrics. It is intended for analyzing batted ball performance, with additional variables on the red zone, which are defined as balls in play with a launch angle between 20 and 35 degrees and an exit velocity of at least 95 mph.

Usage

bballHR

Format

- A data frame with the following columns:
- name** Player's full name (character).
 - playerID** Player's unique identifier in the Lahman database (character).
 - teamID** Team abbreviation (character).
 - year** Season year (numeric).
 - HR** Home runs hit during the season (integer).

AB At-bats during the season (integer).

mean_exit_velo Average exit velocity (mph) over the season (numeric).

mean_launch_angle Average launch angle (degrees) over the season (numeric).

launch_angle_75 Launch angle at the 75th percentile of the player's distribution (numeric).

launch_angle_70 Launch angle at the 70th percentile of the player's distribution (numeric).

launch_angle_65 Launch angle at the 65th percentile of the player's distribution (numeric).

exit_velo_75 Exit velocity at the 75th percentile of the player's distribution (numeric).

exit_velo_80 Exit velocity at the 80th percentile of the player's distribution (numeric).

exit_velo_85 Exit velocity at the 85th percentile of the player's distribution (numeric).

count_red_zone Seasonal count of batted balls in the red zone, defined as a launch angle between 20 and 35 degrees and an exit velocity greater than or equal to 95 mph (integer).

prop_red_zone Proportion of batted balls that fall into the red zone (numeric).

BPF Ballpark factor, indicating the effect of the player's home ballpark on offensive statistics (integer).

Details

Mean Metrics mean_exit_velo and mean_launch_angle represent the player's average exit velocities and launch angles, respectively, over the course of a season.

Quantile Metrics The launch_angle_x and exit_velo_x columns denote the upper x -percentiles (e.g., 75th percentile) of the player's launch angle and exit velocity distributions for that year.

Red Zone Metrics count_red_zone gives the number of balls in play that fall into the red zone, while prop_red_zone represents the proportion of balls in play in this category.

BPF The Ballpark Factor (BPF) quantifies the influence of the player's home ballpark on offensive performance, with values above 100 indicating a hitter-friendly environment.

Source

- Player statistics: [Lahman R Package](#)
- Batted ball data: [Baseball Savant](#)
- Additional analysis: [Patterns of Home Run Hitting in the Statcast Era](#) by Jim Albert

Examples

```
data(bballHR)
head(bballHR)
```

dpit	<i>DPIT residuals for regression models with various non-continuous outcomes</i>
------	--

Description

Calculates DPIT residuals for regression models with non-continuous outcomes. In particular, model assumptions for GLMs with discrete outcomes (e.g., binary, Poisson, and negative binomial), ordinal regression models, zero-inflated regression models, and semicontinuous outcome models can be assessed using `dpit()`.

Usage

```
dpit(model)
```

Arguments

`model` A model object.

Details

This function deploys the appropriate computation based on the class of `model`. The supported model objects and outcome types are listed below.

In addition to the class-based interface, the package also provides distribution-specific DPIT residual calculators. If a fitted model comes from a different class but has a supported outcome distribution, users can call the corresponding distribution-based function directly. For instance, for a regression model with Poisson outcomes, one can use `dpit` to calculate the residuals if the model is fit using `glm` function, or to use `dpit_pois` upon supplying fitted mean values.

- **Discrete outcomes**

- `glm` with `family = binomial()` (see [dpit_bin](#)).
- `glm` with `family = poisson()` (see [dpit_pois](#)).
- `glm.nb` for negative binomial regression (see [dpit_nb](#)).
- `polr` for ordinal outcomes (see [dpit_ordi](#)).

- **Zero-inflated discrete outcomes**

- `zeroinfl` with `dist = "poisson"` (see [dpit_zpois](#)).
- `zeroinfl` with `dist = "negbin"` (see [dpit_znb](#)).

- **Semicontinuous outcomes**

- Tobit regression via `tobit` from AER or `vglm` from VGAM (see [dpit_tobit](#)).
- Tweedie regression via `glm` with a Tweedie family (see [dpit_tweedie](#)).

Formulation for Discrete and Zero-Inflated Outcomes:

The DPIT residual for the i th observation is defined as follows:

$$\hat{r}(Y_i|X_i) = \hat{G}\left(\hat{F}_M(Y_i|\mathbf{X}_i)\right)$$

where

$$\hat{G}(s) = \frac{1}{n-1} \sum_{j=1, j \neq i}^n \hat{F}_M \left(\hat{F}_M^{(-1)}(\mathbf{X}_j) \middle| \mathbf{X}_j \right)$$

and $\hat{F}_M$ refers to the fitted cumulative distribution function. The scale argument is supplied to `residuals()`, `summary()`, or `plot()`, methods for further displaying basic object information, extracting residuals, summarizing residuals, and producing a QQ-plot, respectively. When `scale="uniform"`, DPIT residuals should closely follow a uniform distribution, otherwise it implies model deficiency. When `scale="normal"`, it applies the normal quantile transformation to the DPIT residuals

$$\Phi^{-1} [\hat{r}(Y_i | \mathbf{X}_i)], i = 1, \dots, n.$$

The null pattern is the standard normal distribution in this case.

Formulation for Semicontinuous Outcomes:

The DPIT residuals for regression models with semicontinuous outcomes are

$$\hat{r}_i = \frac{\hat{F}_M(Y_i | \mathbf{X}_i)}{n} \sum_{j=1}^n 1 \left(\hat{p}_0(\mathbf{X}_j) \leq \hat{F}_M(Y_i | \mathbf{X}_i) \right), i = 1, \dots, n,$$

where $\hat{p}_0(\mathbf{X}_i)$ is the fitted probability of zero, and $\hat{F}_M(\cdot | \mathbf{X}_i)$ is the fitted cumulative distribution function for the i th observation. Furthermore,

$$\hat{F}_M(y | \mathbf{x}) = \hat{p}_0(\mathbf{x}) + (1 - \hat{p}_0(\mathbf{x})) \hat{G}_M(y | \mathbf{x})$$

where $\hat{G}_M$ is the fitted cumulative distribution for the positive data.

Value

A `dpit` object containing DPIT residuals.

References

- Yang, L. (2024). "Double probability integral transform residuals for regression models with discrete outcomes." *Journal of Computational and Graphical Statistics*, 33(3), 787–803.
- Yang, L. (2024). "Diagnostics for regression models with semicontinuous outcomes." *Biometrics*, 80(1), ujae007.

Examples

```
library(MASS)
n <- 500
set.seed(1234)
## Negative Binomial example
# Covariates
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
### Parameters
beta0 <- -2
beta1 <- 2
```

```

beta2 <- 1
size1 <- 2
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
# generate outcomes
y <- rnbino(n, mu = lambda1, size = size1)

# True model
model1 <- glm.nb(y ~ x1 + x2)
dpit.nb1 <- dpit(model1)
dpit.nb1
resid.nb1 <- residuals(dpit.nb1, scale = "uniform")
summary(dpit.nb1, scale = "uniform")
plot(dpit.nb1, scale = "uniform")

# Overdispersion
model2 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
dpit.nb2 <- dpit(model2)
resid.nb2 <- residuals(dpit.nb2, scale = "normal")
plot(dpit.nb2, scale = "normal")

## Binary example
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n, 1, 1)
x2 <- rbinom(n, 1, 0.7)
# Coefficients
beta0 <- -5
beta1 <- 2
beta2 <- 1
beta3 <- 3
q1 <- 1 / (1 + exp(beta0 + beta1 * x1 + beta2 * x2 + beta3 * x1 * x2))
y1 <- rbinom(n, size = 1, prob = 1 - q1)

# True model
model01 <- glm(y1 ~ x1 * x2, family = binomial(link = "logit"))
dpit.bin1 <- dpit(model01)
resid.bin1 <- residuals(dpit.bin1)
plot(dpit.bin1)

# Missing covariates
model02 <- glm(y1 ~ x1, family = binomial(link = "logit"))
dpit.bin2 <- dpit(model02)
resid.bin2 <- residuals(dpit.bin2)
plot(dpit.bin2)

## Poisson example
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
# Coefficients

```

```

beta0 <- -2
beta1 <- 2
beta2 <- 1
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
y <- rpois(n, lambda1)

# True model
poismodel1 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
dpit.poi1 <- dpit(poismodel1)
resid.poi1 <- residuals(dpit.poi1)
plot(dpit.poi1)

# Enlarge three outcomes
y <- rpois(n, lambda1) + c(rep(0, (n - 3)), c(10, 15, 20))
poismodel2 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
dpit.poi2 <- dpit(poismodel2)
resid.poi2 <- residuals(dpit.poi2)
plot(dpit.poi2)

## Ordinal example
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n, mean = 2)
# Coefficient
beta1 <- 3

# True model
p0 <- plogis(1, location = beta1 * x1)
p1 <- plogis(4, location = beta1 * x1) - p0
p2 <- 1 - p0 - p1
genemult <- function(p) {
  rmultinom(1, size = 1, prob = c(p[1], p[2], p[3]))
}
test <- apply(cbind(p0, p1, p2), 1, genemult)
y1 <- rep(0, n)
y1[which(test[, 1] == 1)] <- 0
y1[which(test[, 2] == 1)] <- 1
y1[which(test[, 3] == 1)] <- 2
multimodel <- polr(as.factor(y1) ~ x1, method = "logistic")
dpit.ord1 <- dpit(multimodel)
resid.ord1 <- residuals(dpit.ord1)
plot(dpit.ord1)

## Non-Proportionality
n <- 500
set.seed(1234)
x1 <- rnorm(n, mean = 2)
beta1 <- 3
beta2 <- 1
p0 <- plogis(1, location = beta1 * x1)
p1 <- plogis(4, location = beta2 * x1) - p0
p2 <- 1 - p0 - p1

```

```

genemult <- function(p) {
  rmultinom(1, size = 1, prob = c(p[1], p[2], p[3]))
}
test <- apply(cbind(p0, p1, p2), 1, genemult)
y1 <- rep(0, n)
y1[which(test[1, ] == 1)] <- 0
y1[which(test[2, ] == 1)] <- 1
y1[which(test[3, ] == 1)] <- 2
multimodel <- polr(as.factor(y1) ~ x1, method = "logistic")
dpit.ord2 <- dpit(multimodel)
resid.ord2 <- residuals(dpit.ord2)
plot(dpit.ord2)

```

dpit-methods

Methods for DPIT residual objects

Description

Methods for printing, summarizing, plotting, and extracting values from objects returned by `dpit()`, `dpit_2pm()`, and the distribution-specific DPIT calculators. The print method displays the fitted model call or calls, when available, and the sample size. The summary method reports quantiles, mean, and standard deviation for the selected scale. The residuals method extracts the DPIT residuals, and the plot method constructs a QQ plot of the DPIT residuals against its reference distribution.

Usage

```

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

## S3 method for class 'dpit'
residuals(object, scale = c("normal", "uniform"), ...)

## S3 method for class 'dpit'
summary(object, scale = c("normal", "uniform"), ...)

## S3 method for class 'summary.dpit'
print(x, ...)

## S3 method for class 'dpit'
plot(x, scale = c("normal", "uniform"), line_args = list(), ...)

```

Arguments

<code>x</code>	A <code>dpit</code> object to be printed or plotted by <code>print.dpit()</code> or <code>plot.dpit()</code> , or a <code>summary.dpit</code> object to be printed by <code>print.summary.dpit()</code> .
<code>...</code>	Additional arguments passed to or from methods. For <code>plot.dpit()</code> , graphical arguments are passed to <code>stats::qqplot()</code> for customizing the QQ plot (e.g., <code>pch</code> , <code>col</code> , <code>cex</code> , <code>xlab</code> , <code>ylab</code>).

object	A dpit object whose residuals are extracted by <code>residuals.dpit()</code> or summarized by <code>summary.dpit()</code> .
scale	You can choose the scale of the residuals among normal and uniform. The sample quantiles of the residuals are plotted against the theoretical quantiles of a standard normal distribution under the normal scale, and against the theoretical quantiles of a uniform (0,1) distribution under the uniform scale. The default scale is normal.
line_args	A named list of graphical parameters passed to <code>graphics::abline()</code> to modify the reference (red) 45° line in the QQ plot. If left empty, a default red dashed line is drawn.

Value

`residuals.dpit()` returns a numeric vector of DPIT residuals. `summary.dpit()` returns an object of class `summary.dpit`. The print and plot methods return their input invisibly.

dpit_2pm

Residuals for regression models with two-part outcomes

Description

Calculates DPIT residuals with model for two-part models with semicontinuous outcomes. For each component, `dpit_2pm` accepts either a fitted model or supplied probability integral transforms: exactly one of `model0` and `part0`, and exactly one of `model1` and `part1`.

Usage

```
dpit_2pm(model0, model1, y, part0, part1)
```

Arguments

model0	Model object for 0 outcomes (e.g., logistic regression)
model1	Model object for the continuous part (gamma regression)
y	Semicontinuous outcomes.
part0	Alternative argument to <code>model0</code> . One can supply the sequence of probabilities $P(Y_i = 0)$, $i = 1, \dots, n$.
part1	Alternative argument to <code>model1</code> . One can fit a regression model on the positive data and supply their probability integral transforms. Note that the length of <code>part1</code> is the number of positive values in <code>y</code> and can be shorter than <code>part0</code> .

Details

For formulation details on semicontinuous outcomes, see [dpit](#).

In two-part models, the probability of zero can be modeled using a logistic regression, `model0`, while the positive observations can be modeled using a gamma regression, `model1`. Users can choose to use different models and supply the resulting probabilities of zero and probability integral transforms. `part0` should be the sequence of fitted probabilities of zeros $\hat{p}_0(\mathbf{X}_i)$, $i = 1, \dots, n$. `part1` should be the probability integral transform of the positive part $\hat{G}(Y_i|\mathbf{X}_i)$. Note that the length of `part1` is the number of positive values in `y` and can be shorter than `part0`. Exactly one of `model0` and `part0`, and exactly one of `model1` and `part1`, must be supplied. Model and probability inputs may be mixed. Use `residuals()`, `summary()`, and `plot()` on the returned object to select the residual scale, summarize the values, and draw the QQ plot.

Value

A `dpit` object containing DPIT residuals.

Examples

```
library(MASS)
n <- 500
beta10 <- 1
beta11 <- -2
beta12 <- -1
beta13 <- -1
beta14 <- -1
beta15 <- -2
x11 <- rnorm(n)
x12 <- rbinom(n, size = 1, prob = 0.4)

p1 <- 1 / (1 + exp(-(beta10 + x11 * beta11 + x12 * beta12)))
lambda1 <- exp(beta13 + beta14 * x11 + beta15 * x12)
y2 <- rgamma(n, scale = lambda1 / 2, shape = 2)
y <- rep(0, n)
u <- runif(n, 0, 1)
ind1 <- which(u >= p1)
y[ind1] <- y2[ind1]

# models as input
mgamma <- glm(y[ind1] ~ x11[ind1] + x12[ind1], family = Gamma(link = "log"))
m10 <- glm(y == 0 ~ x12 + x11, family = binomial(link = "logit"))
dpit.model <- dpit_2pm(model0 = m10, model1 = mgamma, y = y)
resid.model <- residuals(dpit.model, scale = "normal")
summary(dpit.model, scale = "normal")
plot(dpit.model, scale = "normal")

# PIT as input
cdfgamma <- pgamma(y[ind1],
  scale = mgamma$fitted.values * gamma.dispersion(mgamma),
  shape = 1 / gamma.dispersion(mgamma)
)
p1f <- m10$fitted.values
```

```
dpit.pit <- dpit_2pm(y = y, part0 = p1f, part1 = cdfgamma)
resid.pit <- residuals(dpit.pit, scale = "uniform")
summary(dpit.pit, scale = "uniform")
plot(dpit.pit, scale = "uniform")
```

dpit_bin

*Residuals for regression models with binary outcomes***Description**

Computes DPIT residuals for regression models with binary outcomes using the observed responses (y) and their fitted distributional parameters (prob).

Usage

```
dpit_bin(y, prob)
```

Arguments

y	An observed outcome vector.
prob	A vector of fitted probabilities of one.

Details

For formulation details on discrete outcomes, see [dpit_pois](#).

Value

A dpit object containing DPIT residuals.

Examples

```
## Binary example
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n, 1, 1)
x2 <- rbinom(n, 1, 0.7)
# Coefficients
beta0 <- -5
beta1 <- 2
beta2 <- 1
beta3 <- 3
q1 <- 1 / (1 + exp(beta0 + beta1 * x1 + beta2 * x2 + beta3 * x1 * x2))
y1 <- rbinom(n, size = 1, prob = 1 - q1)

# True model
model01 <- glm(y1 ~ x1 * x2, family = binomial(link = "logit"))
fitted1 <- fitted(model01)
```

```

y1 <- model01$y
dpit.bin1 <- dpit_bin(y=y1, prob=fitted1)
resid.bin1 <- residuals(dpit.bin1)
plot(dpit.bin1)

# Missing covariates
model02 <- glm(y1 ~ x1, family = binomial(link = "logit"))
y2 <- model02$y
fitted2 <- fitted(model02)
dpit.bin2 <- dpit_bin(y=y2, prob=fitted2)
resid.bin2 <- residuals(dpit.bin2)
plot(dpit.bin2)

```

dpit_nb

*Residuals for regression models with negative binomial outcomes***Description**

Computes DPIT residuals for regression models with negative binomial outcomes using the observed counts (y) and their fitted distributional parameters (mu, size).

Usage

```
dpit_nb(y, mu, size)
```

Arguments

y	An observed outcome vector.
mu	A vector of fitted mean values.
size	A dispersion parameter of the negative binomial distribution.

Details

For formulation details on discrete outcomes, see [dpit](#).

Value

A dpit object containing DPIT residuals.

Examples

```

## Negative Binomial example
library(MASS)
n <- 500
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
### Parameters
beta0 <- -2
beta1 <- 2

```

```

beta2 <- 1
size1 <- 2
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
# generate outcomes
y <- rnbino(n, mu = lambda1, size = size1)

# True model
model1 <- glm.nb(y ~ x1 + x2)
y1 <- model1$y
fitted1 <- fitted(model1)
size1 <- model1$theta
dpit.nb1 <- dpit_nb(y=y1, mu=fitted1, size=size1)
resid.nb1 <- residuals(dpit.nb1)
plot(dpit.nb1)

# Overdispersion
model2 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
y2 <- model2$y
fitted2 <- fitted(model2)
dpit.nb2 <- dpit_pois(y=y2, mu=fitted2)
resid.nb2 <- residuals(dpit.nb2)
plot(dpit.nb2)

```

dpit_ordi

*Residuals for regression models with ordinal outcomes***Description**

Computes DPIT residuals for regression models with ordinal outcomes using observed outcomes (*y*), ordinal outcome levels (*level*) and their fitted category probabilities (*fitprob*).

Usage

```
dpit_ordi(y, level, fitprob)
```

Arguments

<i>y</i>	An observed ordinal outcome vector.
<i>level</i>	The response levels in their ordinal order. For instance, <code>c(0, 1, 2)</code> or <code>c("low", "medium", "high")</code> .
<i>fitprob</i>	A matrix of fitted category probabilities. Each row corresponds to an observation, and the columns must follow the order in <i>level</i> . Each row must sum to one.

Details

For formulation details on discrete outcomes, see [dpit](#).

Value

A dpit object containing DPIT residuals.

Examples

```
## Ordinal example
library(MASS)
n <- 500
x1 <- rnorm(n, mean = 2)
beta1 <- 3
# True model
p0 <- plogis(1, location = beta1 * x1)
p1 <- plogis(4, location = beta1 * x1) - p0
p2 <- 1 - p0 - p1
genemult <- function(p) {
  rmultinom(1, size = 1, prob = c(p[1], p[2], p[3]))
}
test <- apply(cbind(p0, p1, p2), 1, genemult)
y1 <- rep(0, n)
y1[which(test[1, ] == 1)] <- 0
y1[which(test[2, ] == 1)] <- 1
y1[which(test[3, ] == 1)] <- 2
multimodel <- polr(as.factor(y1) ~ x1, method = "logistic")

y1 <- multimodel$model[,1]
lev1 <- multimodel$lev
fitprob1 <- fitted(multimodel)

dpit.ord <- dpit_ordi(y=y1, level=lev1, fitprob=fitprob1)
resid.ord <- residuals(dpit.ord)
plot(dpit.ord)
```

dpit_pois

Residuals for regression models with poisson outcomes

Description

Computes DPIT residuals for Poisson outcomes regression using the observed counts (y) and their corresponding fitted mean values (mu).

Usage

```
dpit_pois(y, mu)
```

Arguments

y	An observed outcome vector.
mu	A vector of fitted mean values.

Details

For formulation details on discrete outcomes, see [dpit](#).

Value

A dpit object containing DPIT residuals.

Examples

```
## Poisson example
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
# Coefficients
beta0 <- -2
beta1 <- 2
beta2 <- 1
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
y <- rpois(n, lambda1)

# True model
poismodel <- glm(y ~ x1 + x2, family = poisson(link = "log"))
y1 <- poismodel$y
p1f <- fitted(poismodel)
dpit.poi <- dpit_pois(y=y1, mu=p1f)
resid.poi <- residuals(dpit.poi)
plot(dpit.poi)
```

dpit_tobit

Residuals for a tobit model

Description

Computes DPIT residuals for tobit regression models using the observed responses (y) and their corresponding fitted distributional parameters (mu, sd).

Usage

```
dpit_tobit(y, mu, sd)
```

Arguments

y	An observed outcome vector.
mu	A vector of fitted mean values of latent variables.
sd	A standard deviation of latent variables.

Details

For formulation details on semicontinuous outcomes, see [dpit](#).

Value

The `dpit` object containing DPIT residuals.

Examples

```
## Tobit regression model
library(VGAM)
n <- 500
beta13 <- 1
beta14 <- -3
beta15 <- 3

set.seed(1234)
x11 <- runif(n)
x12 <- runif(n)
lambda1 <- beta13 + beta14 * x11 + beta15 * x12
sd0 <- 0.3
yun <- rnorm(n, mean = lambda1, sd = sd0)
y <- ifelse(yun >= 0, yun, 0)

# Using VGAM package
# True model
fit1 <- vglm(formula = y ~ x11 + x12,
             tobit(Upper = Inf, Lower = 0, lmu = "identitylink"))
# Missing covariate
fit1miss <- vglm(formula = y ~ x11,
                 tobit(Upper = Inf, Lower = 0, lmu = "identitylink"))

dpit.tobit1 <- dpit_tobit(y = y, mu = VGAM::fitted(fit1), sd = sd0)
resid.tobit1 <- residuals(dpit.tobit1)
plot(dpit.tobit1)
dpit.tobit2 <- dpit_tobit(y = y, mu = VGAM::fitted(fit1miss), sd = sd0)
resid.tobit2 <- residuals(dpit.tobit2)
plot(dpit.tobit2)

# Using AER package
library(AER)
# True model
fit2 <- tobit(y ~ x11 + x12, left = 0, right = Inf, dist = "gaussian")
# Missing covariate
fit2miss <- tobit(y ~ x11, left = 0, right = Inf, dist = "gaussian")

dpit.aer1 <- dpit_tobit(y = y, mu = fitted(fit2), sd = sd0)
resid.aer1 <- residuals(dpit.aer1)
plot(dpit.aer1)
dpit.aer2 <- dpit_tobit(y = y, mu = fitted(fit2miss), sd = sd0)
resid.aer2 <- residuals(dpit.aer2)
plot(dpit.aer2)
```

dpit_tweedie

*Residuals for regression models with tweedie outcomes***Description**

Computes DPIT residuals for Tweedie-distributed outcomes using the observed responses (y), their fitted mean values (μ), the variance power parameter (ξ), and the dispersion parameter (ϕ).

Usage

```
dpit_tweedie(y, mu, xi, phi)
```

Arguments

<code>y</code>	Observed outcome vector.
<code>mu</code>	Vector of fitted mean values of each outcomes.
<code>xi</code>	Value of ξ such that the variance is $Var[Y] = \phi\mu^\xi$
<code>phi</code>	Dispersion parameter ϕ .

Details

For formulation details on semicontinuous outcomes, see [dpit](#).

Value

A `dpit` object containing DPIT residuals.

Examples

```
## Tweedie model
library(tweedie)
library(statmod)
n <- 300
x11 <- rnorm(n)
x12 <- rnorm(n)
beta0 <- 5
beta1 <- 1
beta2 <- 1
lambda1 <- exp(beta0 + beta1 * x11 + beta2 * x12)
y1 <- rtweedie(n, mu = lambda1, xi = 1.6, phi = 10)
# Choose parameter p
# True model
model1 <-
  glm(y1 ~ x11 + x12,
      family = tweedie(var.power = 1.6, link.power = 0)
  )
y1 <- model1$y
p.max <- get("p", envir = environment(model1$family$variance))
```

```
lambda1f <- model1$fitted.values
phi1f <- summary(model1)$dis
dpit.tweedie <- dpit_tweedie(y= y1, mu=lambda1f, xi=p.max, phi=phi1f)
resid.tweedie <- residuals(dpit.tweedie)
plot(dpit.tweedie)
```

dpit_znb	<i>Residuals for regression models with zero-inflated negative binomial outcomes</i>
----------	--

Description

Computes DPIT residuals for regression models with zero-inflated negative binomial outcomes using the observed counts (y) and their fitted distributional parameters (mu, pzero, size).

Usage

```
dpit_znb(y, mu, pzero, size)
```

Arguments

y	An observed outcome vector.
mu	A vector of fitted mean values for the count (non-zero) component.
pzero	A vector of fitted probabilities for the zero-inflation component.
size	A dispersion parameter of the negative binomial distribution.

Details

For formulation details on discrete outcomes, see [dpit](#).

Value

A dpit object containing DPIT residuals.

Examples

```
## Zero-Inflated Negative Binomial
library(pscl)
n <- 500
set.seed(1234)

# Covariates
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)

# Coefficients
beta0 <- -2
beta1 <- 2
```

```

beta2 <- 1
beta00 <- -2
beta10 <- 2

# NB dispersion (size = theta; larger => closer to Poisson)
theta_true <- 1.2

# Mean of NB count part
mu_true <- exp(beta0 + beta1 * x1 + beta2 * x2)

# Excess zero probability (logit)
p0 <- 1 / (1 + exp(-(beta00 + beta10 * x1)))

## simulate outcomes
z <- rbinom(n, size = 1, prob = 1 - p0)          # 1 => from NB, 0 => structural zero
y1 <- rnbinom(n, size = theta_true, mu = mu_true) # NB count draw
y <- ifelse(z == 0, 0, y1)

## True model
modelzero1 <- zeroinfl(y ~ x1 + x2 | x1, dist = "negbin", link = "logit")
y1 <- modelzero1$y
mu1 <- stats::predict(modelzero1, type = "count")
pzero1 <- stats::predict(modelzero1, type = "zero")
theta1 <- modelzero1$theta
dpit.zero1 <- dpit_znb(y = y1, pzero = pzero1, mu = mu1, size = theta1)
resid.zero1 <- residuals(dpit.zero1)
plot(dpit.zero1)

## Ignoring zero-inflation: NB only
modelzero2 <- MASS::glm.nb(y ~ x1 + x2)
y2 <- modelzero2$y
mu2 <- fitted(modelzero2)
theta2 <- modelzero2$theta
dpit.zero2 <- dpit_nb(y = y2, mu = mu2, size = theta2)
resid.zero2 <- residuals(dpit.zero2)
plot(dpit.zero2)

```

dpit_zpois

Residuals for regression models with zero-inflated Poisson outcomes

Description

Computes DPIT residuals for regression models with zero-inflated Poisson outcomes using the observed counts (y) and their fitted distributional parameters (μ , $pzero$).

Usage

```
dpit_zpois(y, mu, pzero)
```

Arguments

<code>y</code>	An observed outcome vector.
<code>mu</code>	A vector of fitted mean values for the count (non-zero) component.
<code>pzero</code>	A vector of fitted probabilities for the zero-inflation component.

Details

For formulation details on discrete outcomes, see [dpit](#).

Value

A `dpit` object containing DPIT residuals.

Examples

```
## Zero-Inflated Poisson
library(pscl)
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
# Coefficients
beta0 <- -2
beta1 <- 2
beta2 <- 1
beta00 <- -2
beta10 <- 2

# Mean of Poisson part
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
# Excess zero probability
p0 <- 1 / (1 + exp(-(beta00 + beta10 * x1)))
## simulate outcomes
y0 <- rbinom(n, size = 1, prob = 1 - p0)
y1 <- rpois(n, lambda1)
y <- ifelse(y0 == 0, 0, y1)
## True model
modelzero1 <- zeroinfl(y ~ x1 + x2 | x1, dist = "poisson", link = "logit")
y1 <- modelzero1$y
mu1 <- stats::predict(modelzero1, type = "count")
pzero1 <- stats::predict(modelzero1, type = "zero")
dpit.zero1 <- dpit_zpois(y= y1, pzero=pzero1, mu=mu1)
resid.zero1 <- residuals(dpit.zero1)
plot(dpit.zero1)

## Zero inflation
modelzero2 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
y2 <- modelzero2$y
mu2 <- fitted(modelzero2)
dpit.zero2 <- dpit_pois(y= y2, mu=mu2)
```

```
resid.zero2 <- residuals(dpit.zero2)
plot(dpit.zero2)
```

gof_disc

Goodness-of-fit test for discrete outcome regression models

Description

Goodness-of-fit test for discrete-outcome regression models. Works with GLMs (Poisson, binomial, negative binomial), ordinal outcome regression (MASS: `polr`), and zero-inflated regressions (zero-inflated Poisson and negative binomial fit via `pscl::zeroinfl`).

Usage

```
gof_disc(model, B=1e2, seed=NULL)
```

Arguments

model	A fitted model object (e.g., from <code>glm()</code> , <code>polr()</code> , <code>glm.nb()</code> or <code>zeroinfl</code>).
B	A positive integer giving the number of bootstrap samples. Default is 1e2.
seed	Random seed for bootstrap.

Details

Let $(Y_i, \mathbf{X}_i)$, $i = 1, \dots, n$ denote independent observations, and let $\hat{F}_M(\cdot \mid \mathbf{X}_i)$ be the fitted model-based CDF. It was shown in *Yang et al. (2026)* that under the correctly specified model,

$$\hat{H}(u) = \frac{1}{n} \sum_{i=1}^n \hat{h}(u, Y_i, \mathbf{X}_i)$$

should be close to the identity function, where

$$\hat{h}(u, y, \mathbf{x}) = \frac{u - \hat{F}_M(y-1 \mid \mathbf{x})}{\hat{F}_M(y \mid \mathbf{x}) - \hat{F}_M(y-1 \mid \mathbf{x})} \mathbf{1}\{\hat{F}_M(y-1 \mid \mathbf{x}) < u < \hat{F}_M(y \mid \mathbf{x})\} + \mathbf{1}\{u \geq \hat{F}_M(y \mid \mathbf{x})\}.$$

The test statistic

$$S_n = \int_0^1 \{\hat{H}(u) - u\}^2 du$$

measures the deviation of $\hat{h}(u, y, \mathbf{x})$ from the identity function, with p-values obtained by bootstrap. This method complements residual-based diagnostics by providing a formal check of model adequacy.

Value

An object of class "htest" containing the test statistic (S), the number of bootstrap samples (B), the p-value, the method description, and the model call.

References

Yang, L., Genest, C., & Nešlehová, J. G. (2026). "A goodness-of-fit test for regression models with discrete outcomes." *Canadian Journal of Statistics*, 54(2), e70046.

Examples

```
library(MASS)
library(pscl)
n <- 100
beta1 <- 1; beta2 <- 1
beta0 <- -2; beta00 <- -2; beta10 <- 2
size1 <- 2
set.seed(1)
x1 <- rnorm(n)
x2 <- rbinom(n,1,0.7)
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
p0 <- 1 / (1 + exp(-(beta00 + beta10 * x1)))
y0 <- rbinom(n, size = 1, prob = 1 - p0)
y1 <- rnegbin(n, mu=lambda1, theta=size1)
y <- ifelse(y0 == 0, 0, y1)
model1 <- zeroinfl(y ~ x1 + x2 | x1, dist = "negbin", link = "logit")
gof_disc(model1, B=10, seed=1234)
```

LGPIF	<i>LGPIF Data</i>
-------	-------------------

Description

Data from the Wisconsin Local Government Property Insurance Fund (LGPIF). The LGPIF was established to provide property insurance for local government entities that include counties, cities, towns, villages, school districts, fire departments, and other miscellaneous entities, and is administered by the Wisconsin Office of the Insurance Commissioner. Properties covered under this fund include government buildings, vehicles, and equipment.

Usage

LGPIF

Format

A data frame with 5677 rows and 41 variables:

- PolicyNum** Policy number
- Year** Policy year
- ClaimBC** Total building and contents (BC) claims in the year
- ClaimIM** Total inland marine (IM) claims in the year (contractor’s equipment)
- ClaimPN** Total comprehensive claims from new motor vehicles in the year

ClaimPO Total comprehensive claims from old motor vehicles in the year
ClaimCN Total collision claims from new vehicles in the year
ClaimCO Total collision claims from old vehicles in the year
TypeCity Indicator for city entity
TypeCounty Indicator for county entity
TypeMisc Indicator for miscellaneous entity
TypeSchool Indicator for school entity
TypeTown Indicator for town entity
TypeVillage Indicator for village entity
IsRC Indicator for replacement cost (motor vehicles)
CoverageBC Log coverage amount for building and contents (in millions of dollars)
InDeductBC Log deductible amount for building and contents
NoClaimCreditBC Indicator for no BC claims in prior year
yAvgBC Average BC claim amount
FreqBC Frequency of BC claims
CoverageIM Log coverage amount for inland marine (in millions of dollars)
InDeductIM Log deductible amount for inland marine
NoClaimCreditIM Indicator for no IM claims in prior year
yAvgIM Average IM claim amount
FreqIM Frequency of IM claims
CoveragePN Log coverage amount for comprehensive new vehicles (in millions of dollars)
NoClaimCreditPN Indicator for no PN claims in prior year
yAvgPN Average PN claim amount
FreqPN Frequency of PN claims
CoveragePO Log coverage amount for comprehensive old vehicles (in millions of dollars)
NoClaimCreditPO Indicator for no PO claims in prior year
yAvgPO Average PO claim amount
FreqPO Frequency of PO claims
CoverageCN Log coverage amount for collision of new vehicles (in millions of dollars)
NoClaimCreditCN Indicator for no CN claims in prior year
yAvgCN Average CN claim amount
FreqCN Frequency of CN claims
CoverageCO Log coverage amount for collision of old vehicles (in millions of dollars)
NoClaimCreditCO Indicator for no CO claims in prior year
yAvgCO Average CO claim amount
FreqCO Frequency of CO claims

Source

<https://sites.google.com/a/wisc.edu/jed-frees/>

References

Frees, E. W., Lee, G., & Yang, L. (2016). "Multivariate frequency-severity regression models in insurance." *Risks*, 4(1), 4.

MEPS

Healthcare expenditure data

Description

Healthcare expenditure data set.

Usage

MEPS

Format

A data frame with 29784 rows and 29 variables:

EXP the aggregate annual office based expenditure per participants, semicontinuous outcomes

AGE Age

GENDER 1 if female

ASIAN 1 if Asian

BLACK 1 if Black

NORTHEAST 1 if Northeast

MIDWEST 1 if Midwest

SOUTH 1 if South

USC 1 if have usual source of care

COLLEGE 1 if college or higher degrees

HIGHSCH 1 if high school degree

MARRIED 1 if married

WIDIVSEP 1 if widowed or divorced or separated

FAMSIZE Family Size

HINCOME 1 if high income

MINCOME 1 if middle income

LINCOME 1 if low income

NPOOR 1 if near poor

POOR 1 if poor

FAIR 1 if fair
 GOOD 1 if good
 VGOOD 1 if very good
 MNHPOOR 1 if poor or fair mental health
 ANYLIMIT 1 if any functional or activity limitation
 unemployed 1 if unemployed at the beginning of 2006
 EDUCHEALTH 1 if education, health and social services
 PUBADMIN 1 if public administration
 insured 1 if is insured at the beginning of the year 2006
 MANAGEDCARE if enrolled in an HMO or a gatekeeper plan

Source

<http://www.meps.ahrq.gov/mepsweb/>

ord_curve	<i>Ordered curve for assessing mean structures</i>
-----------	--

Description

Creates a plot to assess the mean structure of regression models. The plot compares the cumulative sum of the response variable and its hypothesized value. Deviation from the diagonal suggests the possibility that the mean structure of the model is incorrect.

Usage

```
ord_curve(model, thr, line_args=list(), ...)
```

Arguments

model	Regression model object (e.g., <code>lm</code> , <code>glm</code> , <code>glm.nb</code> , <code>polr</code>)
thr	Threshold variable (e.g., predictor, fitted values, or variable to be included as a covariate)
line_args	A named list of graphical parameters passed to <code>graphics::abline()</code> to modify the reference 45° line in the ordered curve. If left empty, a default red dashed line is drawn.
...	Additional graphical arguments passed to <code>graphics::plot()</code> for customizing the ordered curve (e.g., <code>pch</code> , <code>col</code> , <code>cex</code> , <code>xlab</code> , <code>ylab</code>).

Details

The ordered curve plots

$$\hat{L}_1(t) = \frac{\sum_{i=1}^n [Y_i 1(Z_i \leq t)]}{\sum_{i=1}^n Y_i}$$

against

$$\hat{L}_2(t) = \frac{\sum_{i=1}^n [\hat{\lambda}_i 1(Z_i \leq t)]}{\sum_{i=1}^n \hat{\lambda}_i},$$

where $\hat{\lambda}_i$ is the fitted mean, and Z_i is the threshold variable.

If the mean structure is correctly specified in the model, $\hat{L}_1(t)$ and $\hat{L}_2(t)$ should be close to each other.

If the curve is distant from the diagonal, it suggests incorrectness in the mean structure. Moreover, if the curve is above the diagonal, the summation of the response is larger than the fitted mean, which implies that the mean is underestimated, and vice versa.

The role of thr (threshold variable Z) is to determine the rule for accumulating $\hat{\lambda}_i$ and Y_i , $i = 1, \dots, n$ for the ordered curve. The candidate for thr could be any function of predictors such as a single predictor (e.g., x_1), a linear combination of predictor (e.g., $x_1 + x_2$), or fitted values (e.g., $\text{fitted}(\text{model})$). It can also be a variable being considered to be included in the mean function. If a variable leads to a large discrepancy between the ordered curve and the diagonal, including this variable in the mean function should be considered.

For more details, see the reference paper.

Value

Invisibly returns NULL. The plotted axes are

- x-axis: $\hat{L}_2(t)$
- y-axis: $\hat{L}_1(t)$

which are defined in Details.

References

Yang, L. (2024). "Double probability integral transform residuals for regression models with discrete outcomes." *Journal of Computational and Graphical Statistics*, 33(3), 787–803.

Examples

```
## Binary example of ordered curve
n <- 500
set.seed(1234)
x1 <- rnorm(n, 1, 1)
x2 <- rbinom(n, 1, 0.7)
beta0 <- -5
beta1 <- 2
```

```

beta2 <- 1
beta3 <- 3
q1 <- 1 / (1 + exp(beta0 + beta1 * x1 + beta2 * x2 + beta3 * x1 * x2))
y1 <- rbinom(n, size = 1, prob = 1 - q1)

## True Model
model0 <- glm(y1 ~ x1 * x2, family = binomial(link = "logit"))
ord_curve(model0, thr = model0$fitted.values) # set the threshold as fitted values

## Missing a covariate
model1 <- glm(y1 ~ x1, family = binomial(link = "logit"))
ord_curve(model1, thr = x2) # set the threshold as a covariate

## Poisson example of ordered curve
n <- 500
set.seed(1234)
x1 <- rnorm(n)
x2 <- rnorm(n)
beta0 <- 0
beta1 <- 2
beta2 <- 1
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)

y <- rpois(n, lambda1)

## True Model
poismodel1 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
ord_curve(poismodel1, thr = poismodel1$fitted.values)

## Missing a covariate
poismodel2 <- glm(y ~ x1, family = poisson(link = "log"))
ord_curve(poismodel2, thr = poismodel2$fitted.values)
ord_curve(poismodel2, thr = x2)

```

quasi_plot

Quasi empirical residuals functions

Description

Draw the quasi-empirical residual distribution functions for regression models with discrete outcomes. Specifically, the model assumption of GLMs with binary, ordinal, Poisson, negative binomial, zero-inflated Poisson, and zero-inflated negative binomial outcomes can be assessed using `quasi_plot()`. A plot far apart from the diagonal indicates lack of fit.

Usage

```
quasi_plot(model, line_args=list(), ...)
```

Arguments

model	Model object (e.g., glm, glm.nb, polr, zeroinfl)
line_args	A named list of graphical parameters passed to <code>graphics::abline()</code> to modify the reference 45° line in the quasi-empirical plot. If left empty, a default red dashed line is drawn.
...	Additional graphical arguments passed to <code>graphics::plot()</code> for customizing the quasi-empirical plot (e.g., lty, col, lwd, xlab, ylab).

Details

The quasi-empirical residual distribution function is defined as follows:

$$\hat{U}(s; \beta) = \sum_{i=1}^n W_n(s; \mathbf{X}_i, \beta) 1[F(Y_i | X_i) < H(s; X_i)]$$

where

$$W_n(s; \mathbf{X}_i, \beta) = \frac{K[(H(s; \mathbf{X}_i) - s)/\epsilon_n]}{\sum_{j=1}^n K[(H(s; \mathbf{X}_j) - s)/\epsilon_n]},$$

ϵ_n is the bandwidth selected using cross validation; $H(s, X_i) = \operatorname{argmin}_{F(k|X_i)} |F(k|X_i) - s|$, F is the CDF and K is a bounded, symmetric, and Lipschitz continuous kernel.

Value

Invisibly returns NULL. The function is called for its side effect of plotting the quasi-empirical residual distribution function $\hat{U}(s; \beta)$ against s .

References

Yang, L. (2021). "Assessment of regression models with discrete outcomes using quasi-empirical residual distribution functions." *Journal of Computational and Graphical Statistics*, 30(4), 1019–1035.

Examples

```
## Negative Binomial example
library(MASS)
# Covariates
n <- 500
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
### Parameters
beta0 <- -2
beta1 <- 2
beta2 <- 1
size1 <- 2
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
# generate outcomes
y <- rnbinom(n, mu = lambda1, size = size1)
```

```
# True model
model1 <- glm.nb(y ~ x1 + x2)
resid.nb1 <- quasi_plot(model1)

# Overdispersion
model2 <- glm(y ~ x1 + x2, family = poisson(link = "log"))
resid.nb2 <- quasi_plot(model2)

## Zero inflated Poisson example
library(pscl)
n <- 500
set.seed(1234)
# Covariates
x1 <- rnorm(n)
x2 <- rbinom(n, 1, 0.7)
# Coefficients
beta0 <- -2
beta1 <- 2
beta2 <- 1
beta00 <- -2
beta10 <- 2

# Mean of Poisson part
lambda1 <- exp(beta0 + beta1 * x1 + beta2 * x2)
# Excess zero probability
p0 <- 1 / (1 + exp(-(beta00 + beta10 * x1)))
## simulate outcomes
y0 <- rbinom(n, size = 1, prob = 1 - p0)
y1 <- rpois(n, lambda1)
y <- ifelse(y0 == 0, 0, y1)
## True model
modelzero1 <- zeroinfl(y ~ x1 + x2 | x1, dist = "poisson", link = "logit")
resid.zero1 <- quasi_plot(modelzero1)
```

Index

* datasets

bballHR, [2](#)

LGPIF, [22](#)

MEPS, [24](#)

bballHR, [2](#)

dpit, [4](#), [10](#), [12](#), [13](#), [15–18](#), [20](#)

dpit-methods, [8](#)

dpit_2pm, [9](#)

dpit_bin, [4](#), [11](#)

dpit_nb, [4](#), [12](#)

dpit_ordi, [4](#), [13](#)

dpit_pois, [4](#), [11](#), [14](#)

dpit_tobit, [4](#), [15](#)

dpit_tweedie, [4](#), [17](#)

dpit_znb, [4](#), [18](#)

dpit_zpois, [4](#), [19](#)

glm, [4](#)

glm.nb, [4](#)

gof_disc, [21](#)

LGPIF, [22](#)

MEPS, [24](#)

ord_curve, [25](#)

plot.dpit (dpit-methods), [8](#)

polr, [4](#)

print.dpit (dpit-methods), [8](#)

print.summary.dpit (dpit-methods), [8](#)

quasi_plot, [27](#)

residuals.dpit (dpit-methods), [8](#)

summary.dpit (dpit-methods), [8](#)

tobit, [4](#)

vglm, [4](#)

zeroinfl, [4](#)