Package ‘evinf’

February 23, 2024

Type Package
Title Inference with Extreme Value Inflated Count Data
Version 0.8.8
Description Allows users to model and draw inferences from extreme value inflated count data, and to evaluate these models and compare to non extreme-value inflated counterparts. The package is built to be compatible with standard presentation tools such as ‘broom’, ‘tidy’, and ‘modelsummary’.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports dplyr, Rcpp, RcppArmadillo, foreach, doParallel, magrittr, doRNG, tibble, mistr, tidyr, purrr, MASS, pscl, MLmetrics, Rdpack, stringi, stringr, rlang, methods, stats, utils, parallel
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.2.3
RdMacros Rdpack
Depends generics, R (>= 2.10)
URL https://github.com/Doktorandahl/evinf
BugReports https://github.com/Doktorandahl/evinf/issues
NeedsCompilation yes
Author David Randahl [cre, aut], Johan Vegelius [aut]
Maintainer David Randahl <david.randahl@pcr.uu.se>
Repository CRAN
Date/Publication 2024-02-23 09:40:11 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>coefficient_extractor</td>
<td>2</td>
</tr>
<tr>
<td>coefficient_extractor.evinb</td>
<td>3</td>
</tr>
<tr>
<td>coefficient_extractor.evzinb</td>
<td>4</td>
</tr>
<tr>
<td>coefficient_extractor.nbboot</td>
<td>4</td>
</tr>
<tr>
<td>coefficient_extractor.zinbboot</td>
<td>5</td>
</tr>
<tr>
<td>compare_models</td>
<td>6</td>
</tr>
<tr>
<td>evinb</td>
<td>7</td>
</tr>
<tr>
<td>evzinb</td>
<td>9</td>
</tr>
<tr>
<td>genevzinb</td>
<td>12</td>
</tr>
<tr>
<td>genevzinb2</td>
<td>12</td>
</tr>
<tr>
<td>glance.evinb</td>
<td>13</td>
</tr>
<tr>
<td>glance.evzinb</td>
<td>13</td>
</tr>
<tr>
<td>glance.nbboot</td>
<td>14</td>
</tr>
<tr>
<td>glance.zinbboot</td>
<td>15</td>
</tr>
<tr>
<td>gm_evzinb</td>
<td>16</td>
</tr>
<tr>
<td>hks</td>
<td>16</td>
</tr>
<tr>
<td>lr_test</td>
<td>17</td>
</tr>
<tr>
<td>oob_evaluation</td>
<td>18</td>
</tr>
<tr>
<td>predict.evinb</td>
<td>19</td>
</tr>
<tr>
<td>predict.evzinb</td>
<td>20</td>
</tr>
<tr>
<td>predict.zinbboot</td>
<td>22</td>
</tr>
<tr>
<td>print.evinb</td>
<td>23</td>
</tr>
<tr>
<td>print.evzinb</td>
<td>23</td>
</tr>
<tr>
<td>print.evzinbcomp</td>
<td>24</td>
</tr>
<tr>
<td>revinb_fit</td>
<td>25</td>
</tr>
<tr>
<td>revzinb_fit</td>
<td>25</td>
</tr>
<tr>
<td>summary.evinb</td>
<td>26</td>
</tr>
<tr>
<td>summary.evzinb</td>
<td>27</td>
</tr>
<tr>
<td>tidy.evinb</td>
<td>28</td>
</tr>
<tr>
<td>tidy.evzinb</td>
<td>29</td>
</tr>
<tr>
<td>tidy.nbboot</td>
<td>31</td>
</tr>
<tr>
<td>tidy.zinbboot</td>
<td>32</td>
</tr>
</tbody>
</table>

Index 34

---

**coefficient_extractor**  
*Bootstrap coefficient extractor*

**Description**

Bootstrap coefficient extractor

**Usage**

```r
coefficient_extractor(object, ...)
```
coefficient_extractor.evinb

Arguments

  object  a fitted model with bootstraps of class evzinb, evinb, nbboot, or zinbboot
  ...    Component to be extracted (not for nbboot). Alternatives are 'nb','zi','evinf','pareto', and 'all'

Value

  A tibble with coefficient values, one row per bootstrap and component

Examples

  data(genevzinb2)
  model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
  coefficient_extractor(model, component = 'all')
coefficient_extractor.evzinb

Bootstrap coefficient extractor

Description
Bootstrap coefficient extractor

Usage
## S3 method for class 'evzinb'
coefficient_extractor(
  object,
  component = c("nb", "zi", "evinf", "pareto", "all"),
  ...
)

Arguments
  object A fitted evzinb model with bootstraps
  component Which component should be extracted
  ... Not in use

Value
A tibble with coefficient values, one row per bootstrap and component

Examples
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
coefficient_extractor(model, component = 'all')

coefficient_extractor.nbboot

Bootstrap coefficient extractor

Description
Bootstrap coefficient extractor

Usage
## S3 method for class 'nbboot'
coefficient_extractor(object, ...)


**coefficient_extractor.zinbboot**

**Arguments**

- **object**: A fitted nbboot model with bootstraps
  - ... Not in use

**Value**

A tibble with coefficient value, one row per bootstrap

**Examples**

```r
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
zinb_comp <- compare_models(model)
coefficient_extractor(zinb_comp$nb)
```

---

**coefficient_extractor.zinbboot**

*Bootstrap coefficient extractor*

**Description**

Bootstrap coefficient extractor

**Usage**

```r
## S3 method for class 'zinbboot'
coefficient_extractor(object, component = c("nb", "zi", "all"), ...)
```

**Arguments**

- **object**: A fitted evinb model with bootstraps
- **component**: Which component should be extracted
  - ... Not in use

**Value**

A tibble with coefficient values, one row per bootstrap and component

**Examples**

```r
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps=10)
zinb_comp <- compare_models(model)
coefficient_extractor(zinb_comp$zinb)
```
Function to compare evzinb or evinb models with zinb and nb models

Description
Function to compare evzinb or evinb models with zinb and nb models

Usage
```
compare_models(
  object,
  nb_comparison = TRUE,
  zinb_comparison = TRUE,
  winsorize = FALSE,
  razorize = FALSE,
  cutoff_value = 10,
  init_theta = NULL,
  multicore = FALSE,
  ncores = NULL
)
```

Arguments
- **object**: A fitted evzinb or evinb model object
- **nb_comparison**: Should comparison be made with a negative binomial model?
- **zinb_comparison**: Should comparions be made with the zinb model?
- **winsorize**: Should winsorizing be done in the comparisons?
- **razorize**: Should razorizing (trimming) be done in the comparisons?
- **cutoff_value**: Integer: Which observation should be used as a basis for winsorizing/razorising. E.g. 10 means that everything larger than the 10th observation will be winsorized/razorised
- **init_theta**: Optional initial value for theta in the NB specification
- **multicore**: Logical: should multiple cores be used
- **ncores**: Number of cores if multicore is used

Value
A list with the original model as the first object and compared models as the following objects

Examples
```
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
compare_models(model)
```
Running an extreme value inflated negative binomial model with bootstrapping

Description
Running an extreme value inflated negative binomial model with bootstrapping

Usage

```r
evinb(formula_nb, formula_evi = NULL, formula_pareto = NULL, data, bootstrap = TRUE, n_bootstraps = 100, multicore = FALSE, ncores = NULL, block = NULL, boot_seed = NULL, max.diff.par = 0.01, max.no.em.steps = 500, max.no.em.steps.warmup = 5, c.lim = c(50, 1000), max.upd.par.pl.multinomial = 0.5, max.upd.par.nb = 0.5, max.upd.par.pl = 0.5, no.m.bfgs.steps.multinomial = 3, no.m.bfgs.steps.nb = 3, no.m.bfgs.steps.pl = 3, pdf.pl.type = "approx", eta.int = c(-1, 1), init.Beta.multinom.PL = NULL, init.Beta.NB = NULL, init.Beta.PL = NULL, init.Alpha.NB = 0.01, init.C = 200, verbose = FALSE)
```

Arguments

- `formula_nb`: Formula for the negative binomial (count) component of the model
- `formula_evi`: Formula for the extreme-value inflation component of the model. If NULL taken as the same formula as `nb`
formula_pareto: Formula for the pareto (extreme value) component of the model. If NULL taken as the same formula as nb.

data: Data to run the model on.

bootstrap: Should bootstrapping be performed. Needed to obtain standard errors and p-values.

n_bootstraps: Number of bootstraps to run. For use of bootstrapped p-values, at least 1,000 bootstraps are recommended. For approximate p-values, a lower number can be sufficient.

multicore: Should multiple cores be used?

ncores: Number of cores if multicore is used. Default (NULL) is one less than the available number of cores.

block: Optional string indicating a case-identifier variable when using block bootstrapping.

boot_seed: Optional bootstrap seed to ensure reproducible results.

max.diff.par: Tolerance for EM algorithm. Will be considered to have converged if the maximum absolute difference in the parameter estimates are lower than this value.

max.no.em.steps: Maximum number of EM steps to run. Will be considered to not have converged if this number is reached and convergence is not reached.

max.no.em.steps.warmup: Number of EM steps in the warmup rounds.

c.lim: Integer range defining the possible values of C.

max.upd.par.pl.multinomial: Maximum parameter change step size in the extreme value inflation component.

max.upd.par.nb: Maximum parameter change step size in the count component.

max.upd.par.pl: Maximum parameter change step size in the pareto component.

no.m.bfgs.steps.multinomial: Number of BFGS steps for the multinomial model.

no.m.bfgs.steps.nb: Number of BFGS steps for the negative binomial model.

no.m.bfgs.steps.pl: Number of BFGS steps for the pareto model.

pdf.pl.type: Probability density function type for the pareto component. Either 'approx' or 'exact'. 'approx' is advised in most cases.

eta.int: Initial values for eta.

init.Beta.multinom.PL: Initial values for beta parameters in the extreme value inflation component. Vector of same length as number of parameters in the extreme value inflation component or NULL (which gives starting values of 0).

init.Beta.NB: Initial values for beta parameters in the count component. Vector of same length as number of parameters in the count component or NULL (which gives starting values of 0).
evzinb

init.Beta.PL  Initial values for beta parameters in the pareto component. Vector of same length as number of parameters in the pareto component or NULL (which gives starting values of 0)

init.Alpha.NB  Initial value of Alpha NB, integer or NULL (giving a starting value of 0)

init.C  Initial value of C. Integer which should be within the C_lim range.

verbose  Should progress be printed for the first run of evinb

Value
An object of class 'evinb'

Examples

data(genevzinb2)
model <- evinb(y~x1+x2+x3,data=genevzinb2, n_bootstraps = 10)

Description
Running an extreme value and zero inflated negative binomial model with bootstrapping

Usage

evinb(
  formula_nb,
  formula_zi = NULL,
  formula_evi = NULL,
  formula_pareto = NULL,
  data,
  bootstrap = TRUE,
  n_bootstraps = 100,
  multicore = FALSE,
  ncores = NULL,
  block = NULL,
  boot_seed = NULL,
  max.diff.par = 0.01,
  max.no.em.steps = 500,
  max.no.em.steps.warmup = 5,
  c.lim = c(50, 1000),
  max.upd.par.zc.multinomial = 0.5,
  max.upd.par.pl.multinomial = 0.5,
  max.upd.par.nb = 0.5,
  max.upd.par.pl = 0.5,
no.m.bfgs.steps.multinomial = 3,
no.m.bfgs.steps.nb = 3,
no.m.bfgs.steps.pl = 3,
pdf.pl.type = "approx",
eta.int = c(-1, 1),
init.Beta.multinom.ZC = NULL,
init.Beta.multinom.PL = NULL,
init.Beta.NB = NULL,
init.Beta.PL = NULL,
init.Alpha.NB = 0.01,
init.C = 200,
verbose = FALSE
)

Arguments

formula_nb
  Formula for the negative binomial (count) component of the model

formula_zi
  Formula for the zero-inflation component of the model. If NULL taken as the
  same formula as nb

formula_evi
  Formula for the extreme-value inflation component of the model. If NULL taken
  as the same formula as nb

formula_pareto
  Formula for the pareto (extreme value) component of the model. If NULL taken
  as the same formula as nb

data
  data to run the model on

bootstrap
  Should bootstrapping be performed. Needed to obtain standard errors and p-
  values

n_bootstraps
  Number of bootstraps to run. For use of bootstrapped p-values, at least 1,000
  bootstraps are recommended. For approximate p-values, a lower number can be
  sufficient

multicore
  Should multiple cores be used?

ncores
  Number of cores if multicore is used. Default (NULL) is one less than the
  available number of cores

block
  Optional string indicating a case-identifier variable when using block bootstrapping

boot_seed
  Optional bootstrap seed to ensure reproducible results.

max.diff.par
  Tolerance for EM algorithm. Will be considered to have converged if the maximum
  absolute difference in the parameter estimates are lower than this value

max.no.em.steps
  Maximum number of EM steps to run. Will be considered to not have converged
  if this number is reached and convergence is not reached

max.no.em.steps.warmup
  Number of EM steps in the warmup rounds

c.lim
  Integer range defining the possible values of C

max.upd.par.zc.multinomial
  Maximum parameter change step size in the zero inflation component
max.upd.par.pl.multinomial
   Maximum parameter change step size in the extreme value inflation component
max.upd.par.nb     Maximum parameter change step size in the count component
max.upd.par.pl     Maximum parameter change step size in the pareto component
no.m.bfgs.steps.multinomial
   Number of BFGS steps for the multinomial model
no.m.bfgs.steps.nb  Number of BFGS steps for the negative binomial model
no.m.bfgs.steps.pl  Number of BFGS steps for the pareto model
pdf.pl.type
   Probability density function type for the pareto component. Either 'approx' or 'exact'. 'approx' is advised in most cases
eta.int
   Initial values for eta
init.Beta.multinom.ZC
   Initial values for beta parameters in the zero value inflation component. Vector of same length as number of parameters in the zero value inflation component or NULL (which gives starting values of 0)
init.Beta.multinom.PL
   Initial values for beta parameters in the extreme value inflation component. Vector of same length as number of parameters in the extreme value inflation component or NULL (which gives starting values of 0)
init.Beta.NB        Initial values for beta parameters in the count component. Vector of same length as number of parameters in the count component or NULL (which gives starting values of 0)
init.Beta.PL        Initial values for beta parameters in the pareto component. Vector of same length as number of parameters in the pareto component or NULL (which gives starting values of 0)
init.Alpha.NB       Initial value of Alpha NB, integer or NULL (giving a starting value of 0)
init.C              Initial value of C. Integer which should be within the C_lim range.
verbose             Logical: should progress of the full run of the model be tracked?

Value
   An object of class 'evzinb'

Examples
   data(genevzinb2)
   model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
genevzinb
Simulated data from the EVZBINB distribution

Description
A simulated dataset of 1,000 observations with one dependent and three dependent variables generated using the EVZINB distribution

Usage
genevzinb

Format
```r
## 'genevzinb' A tibble with 1,000 rows and 4 columns:

y  Dependent variable following EVZINB distribution
x1, x2, x3  Continuous independent variables following the random normal distribution
```

genevzinb2
Simulated data from the EVZBINB distribution

Description
A simulated dataset of 100 observations with one dependent and three dependent variables generated using the EVZINB distribution

Usage
genevzinb2

Format
```r
## 'genevzinb2' A tibble with 100 rows and 4 columns:

y  Dependent variable following EVZINB distribution
x1, x2, x3  Continuous independent variables following the random normal distribution
```
Description

EVZINB and EVINB glance functions

Usage

## S3 method for class 'evinb'
_glance(x, ...)

Arguments

x An EVZINB or EVINB object
...
Further arguments to be passed to glance()

Value

An EVZINB glance function

See Also

   glance

Examples

data(genevzinb2)
model <- evinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
glance(model)

Description

EVZINB and EVINB glance functions

Usage

## S3 method for class 'evzinb'
_glance(x, ...)

Arguments

x An EVZINB or EVINB object
...
Further arguments to be passed to glance()
Arguments

x  
An EVZINB or EVINB object

...  
Further arguments to be passed to glance()

Value

An EVZINB glance function

See Also

glance

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
glance(model)
**Examples**

```r
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
zinb_comp <- compare_models(model)
glance(zinb_comp$nb)
```

---

**Description**

zinbboot and nboot glance functions

**Usage**

```r
## S3 method for class 'zinbboot'
glance(x, ...)
```

**Arguments**

- `x`  
  An nbboot or zinbboot object

- `...`  
  Further arguments to be passed to `glance()`

**Value**

An nbboot glance function

**See Also**

`glance`

**Examples**

```r
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
zinb_comp <- compare_models(model)
glance(zinb_comp$nb)
```
Description

A goodness-of-fit gof tibble for GOF metrics when using modelsummary. The GM tibble can be used to obtain correct table output when making regression tables with modelsummary.

Usage

gm_evzinb

Format

## 'gm_evzinb' A tibble with 7 rows and 3 columns:

<table>
<thead>
<tr>
<th>raw</th>
<th>The modelsummary/broom internal name for the statistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean</td>
<td>The table output for the statistic</td>
</tr>
<tr>
<td>fmt</td>
<td>The number of decimals reported for each statistic by default (can be adapted)</td>
</tr>
</tbody>
</table>

hks

Description

A reduced replication data set from Hultman et al. (2013) United Nations Peacekeeping and Civilian Protection in Civil War. Used to reproduce the the results from Randahl and Vegelius (2023). Note, to reproduce any other results from Hultman et al. (2013) please download the original replication dataset using the link under source.

Usage

hks

Format

A tibble with 3746 rows and 12 columns:

- **conflict_id** The Uppsala Conflict Data Programme conflict ID for the conflict
- **osvAll** The number of observed fatalities from one-sided violence against civilian in the specified conflict-month
- **troopLag** The number of UN military troops in thousands of troops (lagged)
- **policeLag** The number of UN police in thousands of troops (lagged)
militaryobserversLag  The number of UN military troops in thousands of troops (lagged)
epduration  The number of months the current conflict-episode has been ongoing
lntpop  The natural logarithm of the population of the country in which the conflict takes place
lnbrv_AllLag  The natural logarithm of the total number of battle related deaths in the conflict in the previous month
osvAllLagDum  A dummy variable taking the value 1 if any one-sided violence against civilians took place in the previous conflict month
incomp  A dummy variable taking the value 1 if the conflict is about government and 0 otherwise
lntroopLag  The log1p logarithm of troopLag
lnepdur  The log1p logarithm of the episode duration

Source
https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6EBCGA

References


---

**lr_test**

Likelihood ratio test for individual variables of evzinb

**Description**
Likelihood ratio test for individual variables of evzinb

**Usage**

```r
lr_test(
  object,
  vars,
  single = TRUE,
  bootstrap = FALSE,
  multicore = FALSE,
  ncores = NULL,
  verbose = FALSE
)
```
Arguments

- **object**: EVZINB or EVZINB object to perform likelihood ratio test on.
- **vars**: Either a list of character vectors with variable names which to be restricted in the LR test or a character vector of variable names. If a list, each character vector of the list will be run separately, allowing for multiple variables to be restricted as once. If a character vector, parameter 'single' can be used to determine whether all variables in the vector should be restricted at once (single = FALSE) or if the variables should be restricted one by one (single = TRUE).
- **single**: Logical. Determining whether variables in 'vars' should be restricted individually (single = TRUE) or all at once (single = FALSE).
- **bootstrap**: Should LR tests be conducted on each bootstrapped sample or only on the original sample.
- **multicore**: Logical. Should the function be run in parallel?
- **ncores**: Number of cores to use if multicore = TRUE.
- **verbose**: Logical. Should the function be verbose?

Value

A tibble with one row per performed LR test

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
lr_test(model, 'x1')

---

**oob_evaluation**

*Out of bag predictive performance of EVZINB and EVINB models*

Description

Out of bag predictive performance of EVZINB and EVINB models

Usage

```r
oob_evaluation(
  object, 
  predict_type = c("harmonic", "explog"),
  metric = c("rmsle", "rmse", "mse", "mae")
)
```
**Arguments**

- **object**: A fitted `evzinb` or `evinb` with bootstraps on which to conduct out-of-bag evaluation.
- **predict_type**: What type of prediction should be made? Harmonic mean, or `exp(log(prediction))`?
- **metric**: What metric should be used for the out of bag evaluation? Default options include `rmsle`, `rmse`, `mse`, and `mae`. Can also take a user supplied function of the form `function(y_pred,y_true)` which returns a single value.

**Value**

A vector of oob evaluation metrics of the length of the number of bootstraps in the `evzinb/evinb` object.

**Examples**

data(genevzinb2)
model <- evzinb(y~x1+x2+x3,data=genevzinb2, n_bootstraps = 10)
oob_evaluation(model)

---

**predict.evinb**  
*Predictions from evinb object*

**Description**

Predictions from evinb object

**Usage**

```r
# S3 method for class 'evinb'
predict(
  object,
  newdata = NULL,
  type = c("harmonic", "explog", "counts", "pareto_alpha", "evinf", "count_state",
           "states", "all", "quantile"),
  pred = c("original", "bootstrap_median", "bootstrap_mean"),
  quantile = NULL,
  confint = FALSE,
  conf_level = 0.9,
  multicore = FALSE,
  ncores = NULL,
  ...
)
```
Arguments

- **object**: An evinb object for which to produce predicted values.
- **newdata**: Optional new data (tibble) to produce predicted values from.
- **type**: Character string, 'harmonic' for the harmonic mean and 'explog' for exponentiated expected log, 'counts' for predicted count of the negative binomial component, 'pareto_alpha' for the predicted pareto alpha value, 'states' for the predicted component states (prior), 'count_state' for predicted probability of the count state, 'evinf' for predicted probability of the pareto state, 'all' for all predicted values, and 'quantile' for quantile prediction.
- **pred**: Type of prediction to be used, defaults to the original prediction from the fitted model, with alternatives being the bootstrapped median or mean. Note that bootstrap mean may yield infinite values, especially when doing quantile prediction.
- **quantile**: Quantile for which to produce quantile prediction.
- **confint**: Should confidence intervals be made for the predictions? Note: only available for vector type predictions and not 'states' and 'all'.
- **conf_level**: What confidence level should be used for confidence intervals.
- **multicore**: Should multicore be used when calculating quantile prediction? Often it is enough to run quantile prediction on a single core, but in cases of large data or very skewed distributions it may be useful to run multicore.
- **ncores**: Number of cores to be used for multicore.
- **...**: Other arguments passed to predict function.

Value

A vector of predicted values for type 'harmonic', 'explog', 'counts', 'pareto_alpha', 'evinf', 'count_state', and 'quantile' or a tibble of predicted values for type 'states' and 'all' or if confint=T.

Examples

```r
data(genevzinb2)
model <- evinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
predict(model)
predict(model, type='all') # Getting all of the available predicted values
```

**Description**

Predictions from evzinb object
Usage

```r
## S3 method for class 'evzinb'
predict(
  object,
  newdata = NULL,
  type = c("harmonic", "explog", "counts", "pareto_alpha", "zi", "evinf", "count_state", "states", "all", "quantile"),
  pred = c("original", "bootstrap_median", "bootstrap_mean"),
  quantile = NULL,
  confint = FALSE,
  conf_level = 0.9,
  multicore = FALSE,
  ncores = NULL,
  ...
)
```

Arguments

- **object**: An `evzinb` object for which to produce predicted values
- **newdata**: Optional new data (tibble) to produce predicted values from
- **type**: Character string, 'harmonic' for the harmonic mean and 'explog' for exponentiated expected log, 'counts' for predicted count of the negative binomial component, 'pareto_alpha' for the predicted pareto alpha value, 'states' for the predicted component states (prior), 'count_state' for predicted probability of the count state, 'evinf' for predicted probability of the pareto state, 'zi' for the predicted probability of the zero state, 'all' for all predicted values, and 'quantile' for quantile prediction.
- **pred**: Type of prediction to be used, defaults to the original prediction from the fitted model, with alternatives being the bootstrapped median or mean. Note that bootstrap mean may yield infinite values, especially when doing quantile prediction
- **quantile**: Quantile for which to produce quantile prediction
- **confint**: Should confidence intervals be made for the predictions? Note: only available for vector type predictions and not 'states' and 'all'.
- **conf_level**: What confidence level should be used for confidence intervals
- **multicore**: Should multicore be used when calculating quantile prediction? Often it is enough to run quantile prediction on a single core, but in cases of large data or very skewed distributions it may be useful to run multicore
- **ncores**: Number of cores to be used for multicore.
- **...**: Other arguments passed to predict function

Value

A vector of predicted values for type 'harmonic', 'explog', 'counts', 'pareto_alpha', 'zi', 'evinf', 'count_state', and 'quantile' or a tibble of predicted values for type 'states' and 'all' or if confint=T
Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
predict(model)
predict(model, type = 'all') # Getting all of the available predicted values

predict.zinbboot  Prediction for zinbboot

Description

Prediction for zinbboot

Usage

## S3 method for class 'zinbboot'
predict(
  object,
  newdata = NULL,
  type = c("predicted", "counts", "zi", "evinf", "count_state", "states", "all", "quantile"),
  pred = c("original", "bootstrap_median", "bootstrap_mean"),
  quantile = NULL,
  confint = FALSE,
  conf_level = 0.9,
  ...)

Arguments

object  a fitted zinbboot object
newdata  Data to make predictions on
"predicted", "counts", "zi", "evinf", "count_state", "states", "all", "quantile"
"initial", "bootstrap_median", "bootstrap_mean"
Quantile for quantile prediction
Should confidence intervals be created?
Confidence level when predicting with CIs
...  Not used

Value

Predictions from zinbboot
print.evinb  

**EVINB print function**

**Description**

EVINB print function

**Usage**

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

**Arguments**

- x: A fitted evinb model
- ...: Not used

**Value**

An evinb print function

**Examples**

```r
data(genevzinb2)
model <- evinb(y~x1+x2+x3,data=genevzinb2, n_bootstraps = 10)
print(model)
```

---

print.evzinb  

**EVZINB print function**

**Description**

EVZINB print function

**Usage**

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

**Arguments**

- x: A fitted evzinb model
- ...: Not used
Value

An evzinb print function

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
print(model)

print.evzinbcomp

Value

An evzinb print function

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
print(model)

print.evzinbcomp

Compare_models print function

Description

Compare_models print function

Usage

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

Arguments

x A fitted evinb model
...
Not used

Value

An evinb print function

Examples

data(genevzinb2)
model <- evinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10)
print(model)
revinb_fit

Random draws from a fitted evinb model

Description
Random draws from a fitted evinb model

Usage
revinb_fit(object, newdata = NULL, n_draws = 1)

Arguments
- object: A fitted EVINB object
- newdata: Optional newdata
- n_draws: Number of random draws to make

Value
A vector of randomly drawn values from the fitted evinb if n_draws == 1, or a list of length n_draws with random drawn values if n_draws > 1

Examples
data(genevzinb2)
model <- evinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
revinb_fit(model)

revzinb_fit

Random draws from a fitted evzinb model

Description
Random draws from a fitted evzinb model

Usage
revzinb_fit(object, newdata = NULL, n_draws = 1)

Arguments
- object: A fitted EVZINB object
- newdata: Optional newdata
- n_draws: Number of random draws to make
**Value**

A vector of randomly drawn values from the fitted evzinb if `n_draws == 1`, or a list of length `n_draws` with random drawn values if `n_draws > 1`.

**Examples**

```r
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
revzinb_fit(model)
```

---

**summary.evzinb**

**EVINB summary function**

**Description**

EVINB summary function

**Usage**

```r
## S3 method for class 'evzinb'
summary(
  object,
  coef = c("original", "bootstrapped_mean", "bootstrapped_median"),
  standard_error = TRUE,
  p_value = c("bootstrapped", "approx", "both", "none"),
  bootstrapped_props = c("none", "mean", "median"),
  approx_t_value = TRUE,
  symmetric_bootstrap_p = TRUE,
  ...
)
```

**Arguments**

- `object`:
an EVINB object with bootstraps
- `coef`:
  Type of coefficients. Original are the coefficient estimates from the non-bootstrapped version of the model. 'bootstrapped_mean' are the mean coefficients across bootstraps, and 'bootstrapped_median' are the median coefficients across bootstraps
- `standard_error`:
  Should standard errors be computed?
- `p_value`:
  What type of p_values should be computed? 'bootstrapped' are bootstrapped p_values through confidence interval inversion. 'approx' are p-values based on the t-value produced by dividing the coefficient with the standard error.
- `bootstrapped_props`:
  Type of bootstrapped proportions of component proportions to be returned
- `approx_t_value`:
  Should approximate t-values be returned
Should bootstrap p-values be computed as symmetric (leaving alpha/2 percent in each tail)? FALSE gives non-symmetric, but narrower, intervals. TRUE corresponds most closely to conventional p-values.

Additional arguments passed to the summary function

Value
An EVINB summary object

Examples

data(genevzinb2)
model <- evinb(y~x1+x2+x3,data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
summary(model)
What type of p_values should be computed? 'bootstrapped' are bootstrapped p_values through confidence interval inversion. 'approx' are p-values based on the t-value produced by dividing the coefficient with the standard error.

Type of bootstrapped proportions of component proportions to be returned

Should approximate t-values be returned

Should bootstrap p-values be computed as symmetric (leaving alpha/2 percent in each tail)? FALSE gives non-symmetric, but narrower, intervals. TRUE corresponds most closely to conventional p-values.

Additional arguments passed to the summary function

Value

An EVZINB summary object

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
summary(model)

### S3 method for class 'evinb'
tidy(
  x,
  component = c("evi", "count", "pareto", "all"),
  coef_type = c("original", "bootstrap_mean", "bootstrap_median"),
  standard_error = TRUE,
  p_value = c("bootstrapped", "approx", "none"),
  confint = c("none", "bootstrapped", "approx"),
  conf_level = 0.95,
  approx_t_value = TRUE,
  symmetric_bootstrap_p = TRUE,
  ...
)
Arguments

- `x`: An evinb object
- `component`: Which component should be shown?
- `coef_type`: Type of coefficients. Original are the coefficient estimates from the non-bootstrapped version of the model. 'bootstrapped_mean' are the mean coefficients across bootstraps, and 'bootstrapped_median' are the median coefficients across bootstraps
- `standard_error`: Should standard errors be computed?
- `p_value`: What type of p-values should be computed? 'bootstrapped' are bootstrapped p-values through confidence interval inversion. 'approx' are p-values based on the t-value produced by dividing the coefficient with the standard error.
- `confint`: What type of confidence should be computed. Same options as p_value
- `conf_level`: What confidence level should be used for the confidence interval
- `approx_t_value`: Should approximate t-values be returned
- `symmetric_bootstrap_p`: Should bootstrap p-values be computed as symmetric (leaving alpha/2 percent in each tail)? FALSE gives non-symmetric, but narrower, intervals. TRUE corresponds most closely to conventional p-values.

Other arguments passed to tidy function

Value

An EVINB tidy function

Examples

```r
data(genevzinb2)
model <- evinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
tidy(model)
```

Description

EVZINB tidy function
Usage

```r
## S3 method for class 'evzinb'
tidy(
x,
component = c("zi", "evi", "count", "pareto", "all"),
coef_type = c("original", "bootstrap_mean", "bootstrap_median"),
standard_error = TRUE,
p_value = c("bootstrapped", "approx", "none"),
confint = c("none", "bootstrapped", "approx"),
conf_level = 0.95,
approx_t_value = TRUE,
symmetric_bootstrap_p = TRUE,
...
)
```

Arguments

- `x` An evzinb object
- `component` Which component should be shown?
- `coef_type` Type of coefficients. Original are the coefficient estimates from the non-bootstrapped version of the model. 'bootstrap_mean' are the mean coefficients across bootstraps, and 'bootstrap_median' are the median coefficients across bootstraps
- `standard_error` Should standard errors be computed?
- `p_value` What type of p_values should be computed? 'bootstrapped' are bootstrapped p_values through confidence interval inversion. 'approx' are p-values based on the t-value produced by dividing the coefficient with the standard error.
- `confint` What type of confidence should be computed. Same options as p_value
- `conf_level` What confidence level should be used for the confidence interval
- `approx_t_value` Should approximate t-values be returned
- `symmetric_bootstrap_p` Should bootstrap p-values be computed as symmetric (leaving alpha/2 percent in each tail)? FALSE gives non-symmetric, but narrower, intervals. TRUE corresponds most closely to conventional p-values.

Value

An EVZINB tidy function

Examples

```r
data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
tidy(model)
```
tidy.nbboot

Tidy function for nbboot

Description

Tidy function for nbboot

Usage

## S3 method for class 'nbboot'
tidy(
x, 
coef_type = c("original", "bootstrap_mean", "bootstrap_median"),
standard_error = TRUE,
p_value = c("bootstrapped", "approx", "none"),
confint = c("none", "bootstrapped", "approx"),
conf_level = 0.95,
approx_t_value = TRUE,
symmetric_bootstrap_p = TRUE,
include_ylev = FALSE,
...
)

Arguments

x A fitted bootstrapped zero-inflated model
c_coef_type What type of coefficient should be reported, original, bootstrapped mean, or bootstrapped median
standard_error Should bootstrapped standard errors be reported?
p_value What type of p-value should be reported? Bootstrapped p_values, approximate p_values, or none?
confint What type of confidence intervals should be reported? Bootstrapped p_values, approximate p_values, or none?
conf_level Confidence level for confidence intervals
approx_t_value Should approximate t_values be reported
_symmetric_bootstrap_p Should bootstrap p-values be computed as symmetric (leaving alpha/2 percent in each tail)? FALSE gives non-symmetric, but narrower, intervals. TRUE corresponds most closely to conventional p-values.
include_ylev Logical. Should y.lev be included in the tidy output? Makes for nicer tables when using modelsummary
... Other arguments to be passed to tidy
tidy.zinbboot

Value

A tidy function for a bootstrapped nb model

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3,data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
zinb_comp <- compare_models(model)
tidy(zinb_comp$nb)

tidy.zinbboot Tidy function for zinbboot

Description

Tidy function for zinbboot

Usage

## S3 method for class 'zinbboot'
tidy(
  x,
  component = c("zi", "count", "all"),
  coef_type = c("original", "bootstrap_mean", "bootstrap_median"),
  standard_error = TRUE,
  p_value = c("bootstrapped", "approx", "none"),
  confint = c("none", "bootstrapped", "approx"),
  conf_level = 0.95,
  approx_t_value = TRUE,
  symmetric_bootstrap_p = TRUE,
  ...
)

Arguments

x A fitted bootstrapped zero-inflated model
component Which component should be shown?
coef_type What type of coefficient should be reported, original, bootstrapped mean, or bootstrapped median
standard_error Should bootstrapped standard errors be reported?
p_value What type of p-value should be reported? Bootstrapped p_values, approximate p-values, or none?
confint What type of confidence intervals should be reported? Bootstrapped p_values, approximate p-values, or none?
tidy.zinbboot

| conf_level | Confidence level for confidence intervals |
| approx_t_value | Should approximate t_values be reported |
| symmetric_bootstrap_p | Should bootstrap p-values be computed as symmetric (leaving alpha/2 percent in each tail)? FALSE gives non-symmetric, but narrower, intervals. TRUE corresponds most closely to conventional p-values. |
| ... | Other arguments to be passed to tidy |

Value

A tidy function for a bootstrapped zinb model

Examples

data(genevzinb2)
model <- evzinb(y~x1+x2+x3, data=genevzinb2, n_bootstraps = 10, multicore = TRUE, ncores = 2)
zinb_comp <- compare_models(model)
tidy(zinb_comp$zinb)
Index

* datasets
  - genevnzb, 12
  - genevnzb2, 12
  - gm_evznb, 16
  - hks, 16

  coefficient_extractor, 2
  coefficient_extractor.evinb, 3
  coefficient_extractor.evznb, 4
  coefficient_extractor.nbboot, 4
  coefficient_extractor.zinbboot, 5
  compare_models, 6

  evinz, 7
  evznb, 9

  genevnzb, 12
  genevnzb2, 12
  glance, 13–15
  glance.evinb, 13
  glance.evznb, 13
  glance.nbboot, 14
  glance.zinbboot, 15
  gm_evznb, 16

  hks, 16
  lr_test, 17
  oob_evaluation, 18

  predict.evinb, 19
  predict.evznb, 20
  predict.zinbboot, 22
  print.evinb, 23
  print.evznb, 23
  print.evznbcomp, 24

  revinz_fit, 25
  revznb_fit, 25

  summary.evinb, 26
  summary.evznb, 27
  tidy.evinb, 28
  tidy.evznb, 29
  tidy.nbboot, 31
  tidy.zinbboot, 32