Package ‘ordbetareg’

February 25, 2022

Type Package
Title Ordered Beta Regression Models with Brms
Version 0.2.1
Description Implements ordered beta regression models, which are for modeling continuous variables with upper and lower bounds, such as survey sliders, dose-response relationships and indexes. For more information, see Kubinec (2022) <doi:10.31235/osf.io/2sx6y>. The package is a front-end to the R package 'brms', which facilitates a range of regression specifications, including hierarchical, dynamic and multivariate modeling.

BugReports https://github.com/saudiwin/ordbetareg_pack/issues
License MIT + file LICENSE
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Depends R (>= 3.5), faux, dplyr, brms, tidyr, stats
Suggests rmarkdown, knitr, ggplot2, modelsummary, marginaleffects, haven, stringr, Hmisc
VignetteBuilder knitr
NeedsCompilation no
Author Robert Kubinec [aut, cre] (<https://orcid.org/0000-0001-6655-4119>)
Maintainer Robert Kubinec <rmk7@nyu.edu>
Repository CRAN
Date/Publication 2022-02-25 08:00:09 UTC

R topics documented:

  normalize ................................................................. 2
  ordbetareg ............................................................... 3
  ord_fit_mean .............................................................. 6
normalize

Normalize Outcome/Response to \([0,1]\) Interval

Description
This function takes a continuous (double) column of data and converts it to have 0 as the lower bound and 1 as the upper bound.

Usage
normalize(outcome)

Arguments
outcome Any non-character vector. Factors will be converted to numeric via coercion.

Details
Beta regression can only be done with a response that is continuous with a lower bound of 0 and an upper bound of 1. However, it is straightforward to transform any lower and upper-bounded continuous variable to the \([0,1]\) interval. This function does the transformation and saves the original bounds as attributes so that the bounds can be reverse-transformed.

Value
A numeric vector with an upper bound of 1 and a lower bound of 0. The original bounds are saved in the attributes "lower_bound" and "upper_bound".

Examples
# set up arbitrary upper and lower-bounded vector
outcome <- runif(1000, min=-33, max=445)

# normalize to \([0,1]\)
trans_outcome <- normalize(outcome=trans_outcome)
summary(trans_outcome)

# only works with numeric vectors and factors
## Not run:
normalize(outcome=c('a','b'))

## End(Not run)
Description

This function allows you to estimate an ordered beta regression model via a formula syntax.

The `ordbetareg` package is essentially a wrapper around `brms` that enables the ordered beta regression model to be fit. This model has advantages over other alternatives for continuous data with upper and lower bounds, such as survey sliders, indexes, dose-response relationships, and visual analog scales (among others). The package allows for all of the many `brms` regression modeling functions to be used with the ordered beta regression distribution.

Usage

```r
ordbetareg(
  formula = NULL,
  data = NULL,
  phi_reg = FALSE,
  use_brm_multiple = FALSE,
  coef_prior_mean = 0,
  coef_prior_sd = 5,
  phi_prior = 0.1,
  dirichlet_prior = c(1, 1, 1),
  phi_coef_prior_mean = 0,
  phi_coef_prior_sd = 5,
  extra_prior = NULL,
  inits = "0",
  ...
)
```

Arguments

- **formula**: Either an R formula in the form response/DV ~ var1 + var2 etc. or formula object as created/called by the `brms` `brms::bf` function. *Please avoid using 0 or Intercept in the formula definition.
- **data**: An R data frame or tibble containing the variables in the formula.
- **phi_reg**: T/F. Whether you are including a linear model predicting the dispersion parameter, phi. Defaults to false.
- **use_brm_multiple**: (T/F) Whether the model should use `brms::brm_multiple` for multiple imputation over multiple dataframes passed as a list to the `data` argument.
- **coef_prior_mean**: The mean of the Normal distribution prior on the regression coefficients (for predicting the mean of the response). Default is 0.
coef_prior_sd  The SD of the Normal distribution prior on the regression coefficients (for predicting the mean of the response). Default is 5, which makes the prior weakly informative on the logit scale.

phi_prior  The mean parameter of the exponential prior on phi, which determines the dispersion of the beta distribution. The default is .1, which equals a mean of 10 and is thus weakly informative. If the response has very low variance (i.e. tightly) clusters around a specific value, then increasing this prior may be helpful. Checking the value of phi in the output of the model command will reveal if 10 is too small.

dirichlet_prior  A vector of three integers corresponding to the prior parameters for the dirichlet distribution (alpha parameter) governing the location of the cutpoints between the components of the response (continuous vs. degenerate). The default is 1 which puts equal probability on degenerate versus continuous responses. Likely only needs to be changed in a repeated sampling situation to stabilize the cutpoint locations across samples.

phi_coef_prior_mean  The mean of the Normal distribution prior on the regression coefficients for predicting phi, the dispersion parameter. Only useful if a linear model is being fit to phi. Default is 0.

phi_coef_prior_sd  The SD of the Normal distribution prior on the regression coefficients for predicting phi, the dispersion parameter. Only useful if a linear model is being fit to phi. Default is 5, which makes the prior weakly informative on the logit scale.

extra_prior  An additional prior, such as a prior for a specific regression coefficient, added to the model by passing one of the brms functions brms::set_prior or brms::prior_string with appropriate values.

inits  This parameter is used to determine starting values for the Stan sampler to begin Markov Chain Monte Carlo sampling. It is set by default at 0 because the non-linear nature of beta regression means that it is possible to begin with extreme values depending on the scale of the covariates. Setting this to 0 helps the sampler find starting values. It does, on the other hand, limit the ability to detect convergence issues with Rhat statistics. If that is a concern, such as with an experimental feature of brms, set this to "random" to get more robust starting values (just be sure to scale the covariates so they are not too large in absolute size).

...  All other arguments passed on to the brm function

Details

This function is a wrapper around the brms::brm function, which is a powerful Bayesian regression modeling engine using Stan. To fully explore the options available, including dynamic and hierarchical modeling, please see the documentation for the brm function above. As the ordered beta regression model is currently not available in brms natively, this modeling function allows a brms model to be fit with the ordered beta regression distribution.

This function allows you to set priors on the dispersion parameter, the cutpoints, and the regression coefficients (see below for options). However, to add specific priors on individual covariates, you
would need to use the `brms::set_prior` function by specifying an individual covariate (see function documentation) and passing the result of the function call to the `extra_prior` argument.

This function will also automatically normalize the outcome so that it lies in the \([0,1]\) interval, as required by beta regression. For further information, see the documentation for the `normalize` function.

To learn more about how the package works, see the vignette by using the command `browseVignettes(package='ordbetareg')`.

For more info about the distribution, see this paper: https://osf.io/preprints/socarxiv/2sx6y/

To cite the package, please cite the following paper:


**Value**

A `brms` object fitted with the ordered beta regression distribution.

**Examples**

```r
# load survey data that comes with the package
data("pew")

# prepare data
model_data <- select(pew, therm,
education="F_EDUCAT2_FINAL",
 region="F_CREGION_FINAL",
 income="F_INCOME_FINAL")

# It takes a while to fit the models. Run the code
# below if you want to load a saved fitted model from the
# package, otherwise use the model-fitting code
data("ord_fit_mean")

# fit the actual model
ord_fit_mean <- ordbetareg(formula=therm ~ education + income +
 (1|region),
data=model_data,
 cores=2, chains=2)

# access values of the coefficients
summary(ord_fit_mean)
```
ord_fit_mean  

**Fitted Ordered Beta Regression Model**

**Description**

A fitted ordered beta regression model to the mean of the thermometer column from the pew data.

**Usage**

ord_fit_mean

**Format**

an ordbetareg object

ord_fit_phi  

**Fitted Ordered Beta Regression Model (Phi Regression)**

**Description**

A fitted ordered beta regression model to the dispersion parameter of the thermometer column from the pew data.

**Usage**

ord_fit_phi

**Format**

an ordbetareg object

pew  

**Pew American Trends Panel Wave 28**

**Description**

A dataset with the non-missing responses for the 28th wave of the Pew American Trends Panel survey.

**Usage**

pew
**sim_data**

**Format**
A data frame with 140 variables and 2,538 observations.

**Source**

---

**sim_data**

*Simulated Ordered Beta Regression Values*

**Description**
The simulated draws used in the vignette for calculating statistical power.

**Usage**
sim_data

**Format**
A dataframe

---

**sim_ordbeta**

*Power Calculation via Simulation of the Ordered Beta Regression Model*

**Description**
This function allows you to calculate power curves (or anything else) via simulating the ordered beta regression model.

**Usage**
sim_ordbeta(
  N = 1000,
  k = 5,
  iter = 1000,
  cores = 1,
  rho = 0.5,
  phi = 1,
  cutpoints = c(-1, 1),
  beta_coef = NULL,
  beta_type = "continuous",
  treat_assign = 0.5,
  return_data = FALSE,
  seed = as.numeric(Sys.time()),
  ...
)

Arguments

N  The sample size for the simulation. Include a vector of integers to examine power/results for multiple sample sizes.

k  The number of covariates/predictors.

iter  The number of simulations to run. For power calculation, should be at least 500 (yes, this will take some time).

cores  The number of cores to use to parallelize the simulation.

rho  The correlation (between -1 and 1) of the predictors k.

phi  Value of the dispersion parameter in the beta distribution.

cutpoints  Value of the two cutpoints for the ordered model. By default are the values -1 and +1 (these are interpreted in the logit scale and so should not be too large). The farther apart, the fewer degenerate (0 or 1) responses there will be in the distribution.

beta_coef  If not null, a vector of length k of the true predictor coefficients/treatment values to use for the simulation. Otherwise, coefficients are drawn from a random uniform distribution from -1 to 1 for each predictor.

beta_type  Can be either continuous or binary. Use the latter for conventional treatments with two values.

treat_assign  If beta_type is set to binary, you can use this parameter to set the proportion of N assigned to treatment. By default, the parameter is set to 0.5 for equal/balanced treatment control groups.

return_data  Whether to return the simulated data as a list in the data column of the returned data frame.

seed  The seed to use to make the results reproducible. Set automatically to a date-time stamp.

...  Any other arguments are passed on to the brms::brm function to control modeling options.

Details

This function implements the simulation found in Kubinec (2022). This simulation allows you to vary the sample size, number & type of predictors, values of the predictors (or treatment values), correlation between predictors, and the power to target. The function returns a data frame with one row per simulation draw and covariate k.

Value

a tibble data frame with columns of simulated and estimated values and rows for each simulation iteration X coefficient combination. I.e., if there are five predictors, and 1,000 iterations, the resulting data frame will have 1,000 rows. If there are multiple values for N, then each value of N will have its own set of iterations, making the final size of the data a multiple of the number of sample sizes to iterate over. The data frame will have the following columns: 1.
Examples

# This function takes a while to run as it has
# to fit an ordered beta regression to each
# draw. The package comes with a saved
# simulation dataset you can inspect to see what the
# result looks like

data("sim_data")

# will take a while to run this

    sim_data <- sim_ordbeta(N=c(250,750),
                      k=1,
                      beta_coef = .5,
                      iter=5,cores=2,
                      beta_type="binary",
                      treat_assign=0.3)

# to get the power values by N, simply summarize/group
# by N with functions from the R package dplyr

sim_data %>%
  group_by(N) %>%
  summarize(mean_power=mean(power))
Index

* datasets
  ord_fit_mean, 6
  ord_fit_phi, 6
  pew, 6
  sim_data, 7

brms::bf, 3
brms::brm, 4, 8
brms::brm_multiple, 3
brms::prior_string, 4
brms::set_prior, 4, 5

normalize, 2, 5

ord_fit_mean, 6
ord_fit_phi, 6
ordbetaareg, 3

pew, 6

sim_data, 7
sim_ordbeta, 7