Package

May 27, 2019

Type Package

Title Analysis of N of 1 Trials

Version 0.5.0

Depends R (>= 2.10)

Imports rjags (>= 4-6), splines, combinat, MASS, jsonlite, ggplot2, scales, coda (>= 0.13)

Author Jiabei Yang, Michael Seo, Christopher Schmid

Maintainer Jiabei Yang <jiabei_yang@brown.edu>

Description
A package for running analysis for N of 1 study trials. Runs Bayesian linear regression, ordinal/logistic regression, and poisson regression. Includes different plots to visualize the results.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

`nof1-package` *mcnet: A package for N of 1 study analysis using Bayesian methods*

Description

A package for running N of 1 study trials

Details

An N of 1 trial is a clinical trial in which a single patient is the entire trial, a single case study. The main purpose of this package was to serve as an analysis tool for one of the PCORI grants we were working with. It is designed for N of 1 trials and can fit bayesian versions of linear regression, logistic/ordinal regression, and poisson regression. Package includes number of different plotting tools for visualization.

`frequency_plot` *Frequency plot for raw data*

Description

Frequency plot for raw data

Usage

```
frequency_plot(nof1, xlab = NULL, title = NULL)
```

Arguments

`nof1`	nof1 object created using nof1.data
`xlab`	x axis label
`title`	title name

Examples

```
Y <- laughter$Y
Treat <- laughter$Treat
nof1 <- nof1.data(Y, Treat, ncat = 11, baseline = "Usual Routine", response = "ordinal")
frequency_plot(nof1)
```

nof1.data — *Make an N of 1 object containing data, priors, and a jags model file*

Description

Make an N of 1 object containing data, priors, and a jags model file

Usage

```
nof1.data(Y, Treat, response = NULL, ncat = NULL, bs.trend = F,
  y.time = NULL, knots.bt.block = NULL, block.no = NULL,
  alpha.prior = NULL, beta.prior = NULL, eta.prior = NULL,
  dc.prior = NULL, c1.prior = NULL, rho.prior = NULL,
  hy.prior = NULL)
```

Arguments

`Y`	Outcome of the study. This should be a vector with length of total number of observations.
`Treat`	Treatment indicator vector with same length as the outcome.
`response`	Type of outcome. Can be normal, binomial, poisson or ordinal.
`ncat`	Number of categories. Used in ordinal models.
`y.time`	parameter used for modelling splines. Time on the original scale with same length as the outcome. Still under development.
`alpha.prior`	Prior for the intercept of the model.
`beta.prior`	Prior for the treatment coefficient.
`eta.prior`	Prior for modelling splines. Still under development.
`dc.prior`	Prior for the length between cutpoints. Used only for ordinal logistic models.
`c1.prior`	Prior for the first cutpoint. Used only for ordinal logistic models.
`rho.prior`	Prior for the correlated error model. Still under development.
`hy.prior`	Prior for the heterogeneity parameter. Supports uniform, gamma, and half normal for normal and binomial response and wishart for multinomial response. It should be a list of length 3, where first element should be the distribution (one of dunif, dgamma, dhnorm, dwish) and the next two are the parameters associated with the distribution. For example, list("dunif", 0, 5) give uniform prior with lower bound 0 and upper bound 5 for the heterogeneity parameter. For wishart distribution, the last two parameter would be the scale matrix and the degrees of freedom.
`knots`	parameter used for modelling splines. Still under development.

Value

Creates list of variables that are used to run the model using `nof1.run`

`Y`	Outcome
`Treat`	Treatment
`baseline`	Baseline variable

ncat	Number of categories for ordinal response
nobs	Total number of observations in a study
Treat.name	Treatment name besides baseline treatment
response	The type of response variable
priors	Priors that the code will be using. Default priors are used if prior was not specified
code	Rjags model file code that is generated using information provided by the user. To view model file inside R, use `cat(nof1$code)`.

Examples

```
###Blocker data example
laughter
Y <- laughter$Y
Treat <- laughter$Treat
nof1 <- nof1.data(Y, Treat, ncat = 11, response = "ordinal")
str(nof1)
cat(nof1$code)
```

`nof1.normal.simulation` *Normal data simulation*

Description

Simulating sample normal data

Usage

```
nof1.normal.simulation(Base.size = 2, Treat.size = 8, prec = 0.5,
  alpha = 50, beta_A = -3, beta_B = -1)
```

`nof1.run` *Run the model using the nof1 object*

Description

This is the core function that runs the model in our program. Before running this function, we need to specify data, prior, JAGS code, etc. using `nof1.data`.

Usage

```
nof1.run(nof1, inits = NULL, n.chains = 3, max.run = 1e+05,
  setsize = 10000, n.run = 50000, conv.limit = 1.05,
  extra.pars.save = NULL)
```

Arguments

`nof1`	nof1 object created from `nof1.data` function
`inits`	Initial values for the parameters being sampled. If left unspecified, program will generate reasonable initial values.
`n.chains`	Number of chains to run
`max.run`	Maximum number of iterations that user is willing to run. If the algorithm is not converging, it will run up to `max.run` iterations before printing a message that it did not converge
`setsize`	Number of iterations that are run between convergence checks. If the algorithm converges fast, user wouldn't need a big setsize. The number that is printed between each convergence checks is the gelman-rubin diagnostics and we would want that to be below the conv.limit the user specifies.
`n.run`	Final number of iterations that the user wants to store. If after the algorithm converges, user wants less number of iterations, we thin the sequence. If the user wants more iterations, we run extra iterations to reach the specified number of runs
`conv.limit`	Convergence limit for Gelman and Rubin's convergence diagnostic.
`extra.pars.save`	Parameters that user wants to save besides the default parameters saved. See code using `cat(nof1$code)` to see which parameters can be saved.

Value

`nof1`	nof1 object
`inits`	Initial values that are either specified by the user or generated as a default
`pars.save`	Parameters that are saved. Add more parameters in extra.pars.save if other variables are desired
`data_rjags`	Data that is put into rjags function `jags.model`
`burnin`	Half of the converged sequence is thrown out as a burnin
`n.thin`	If the number of iterations user wants (n.run) is less than the number of converged sequence after burnin, we thin the sequence and store the thinning interval
`samples`	MCMC samples stored using jags. The returned samples have the form of mcmc.list and can be directly applied to coda functions
`max.gelman`	Maximum Gelman and Rubin's convergence diagnostic calculated for the final sample

Examples

```
laughter
Y <- laughter$Y
Treat <- laughter$Treat
nof1 <- nof1.data(Y, Treat, ncat = 11, baseline = "Usual Routine", response = "ordinal")
result <- nof1.run(nof1)
summary(result$samples)
```

raw_table *Summary data table for nof1*

Description

Summary data table for nof1

Usage

```
raw_table(nof1)
```

Arguments

nof1	nof1 object created using nof1.data

Examples

```
Y <- laughter$Y
Treat <- laughter$Treat
nof1 <- nof1.data(Y, Treat, ncat = 11, baseline = "Usual Routine", response = "ordinal")
raw_table(nof1)
```

stacked_percent_barplot *Stacked_percent_barplot for raw data (for ordinal or binomial data)*

Description

Stacked_percent_barplot for raw data (for ordinal or binomial data)

Usage

```
stacked_percent_barplot(nof1, title = NULL)
```

Arguments

nof1	nof1 object created using nof1.data
title	title name

Examples

```
Y <- laughter$Y
Treat <- laughter$Treat
nof1 <- nof1.data(Y, Treat, ncat = 11, baseline = "Usual Routine", response = "ordinal")
stacked_percent_barplot(nof1)
```

`summarize_nof1` *Function to present a summary of our results*

Description

A neat function to summarize the results.

Usage

```
summarize_nof1(result, alpha = 0.05)
```

Arguments

`result`	A list which contains data file created by `nof1.data` and the result file created by `nof1.run`
`alpha`	The alpha value for the confidence interval. If no value is entered will give the 95% confidence interval.

Value

The function computes and returns a list of summary statistics of the raw data and the fitted model.

`raw.y.mean`	The raw mean of the outcome for each treatment
`raw.y.median`	The raw median of the outcome for each treatment
`post.coef.mean`	The posterior mean of the coefficient for each treatment
`post.coef.median`	The posterior median of the coefficient for each treatment
`post.y.mean`	The posterior mean of the outcome for each treatment
`post.y.median`	The posterior median of the outcome for each treatment
`post.coef.ci`	The credible interval of the coefficient for each treatment
`post.y.ci`	The credible interval of the outcome for each treatment
`comp.treat.post.coef`	The posterior probabilities of one coefficient is greater than the other when comparing two treatments
`comp.treat.post.y`	The posterior probabilities of outcome is greater than the other when comparing two treatments

time_series_plot — *time series plot across different interventions*

Description

time series plot across different interventions

Usage

```
time_series_plot(nof1, timestamp = NULL,
  timestamp.format = "%m/%d/%Y %H:%M", x.name = "", y.name = "",
  title = NULL, normal.response.range = NULL)
```

Arguments

`nof1`	nof1 object created using nof1.data
`timestamp`	time of the nof1 event occurring
`timestamp.format`	format of the timestamp
`normal.response.range`	the range of the outcome if continuous; a vector of minimum and maximum
`Outcomes.name`	used to label y-axis outcome variable

Examples

```
Y <- laughter$Y
Treat <- laughter$Treat
nof1 <- nof1.data(Y, Treat, ncat = 11, baseline = "Usual Routine", response = "ordinal")
timestamp <- seq(as.Date('2015-01-01'),as.Date('2016-01-31'), length.out = length(Y))
time_series_plot(nof1, timestamp = timestamp, timestamp.format = "%m-%d-%Y", Outcome.name = "Stress")
```

wrap — *For PCORI purposes*

Description

For PCORI purposes

Usage

```
wrap(data, metadata)
```

`wrap2` *For PCORI purposes*

Description

For PCORI purposes

Usage

```
wrap2(data, metadata)
```

Index