Package 'nof1'

August 13, 2018

Type Package

Title Single Subject (N-Of-1) Designs to Answer Patient-Identified Research Questions

Version 0.5.0

Depends R (>= 2.10)

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

Description A package for running 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 6.0.1

Author Michael Seo [aut, cre], Christopher Schmid [aut]

Maintainer Michael Seo <michael_seo@brown.edu>

RemoteType github

RemoteHost https://api.github.com

RemoteRepo nof1

RemoteUsername MikeJSeo

RemoteRef master

RemoteSha e5681cddb9a5e30f4ac300784fc0e6d813abf57b

GithubRepo nof1

GithubUsername MikeJSeo

GithubRef master

GithubSHA1 e5681cddb9a5e30f4ac300784fc0e6d813abf57b

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

Description

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

Usage

```
nof1.data(Y, Treat, baseline = "baseline", ncat = NULL, response = NULL,
  Time = NULL, knots = NULL, alpha.prior = NULL, beta.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.
baseline	baseline Treatment name. This serves as a baseline/placebo when comparing different treatments.
ncat	Number of categories. Used in ordinal models.
response	Type of outcome. Can be normal, binomial, poisson or ordinal.
Time	parameter used for modelling splines. Still under development.
knots	parameter used for modelling splines. Still under development.
alpha.prior	Prior for the intercept of the model.
beta.prior	Prior for the treatment coefficient.
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.
gamma.prior	Prior 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
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, baseline = "Usual Routine", response = "ordinal")
str(nof1)
cat(nof1$code)
```

---

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 network.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 <- network.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 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 *Summarizes the result from the model into json format*

Description

Summarizes the result from the model into json format

Usage

```
summarize_nof1(nof1, result)
```

time_series_plot *time series plot across different interventions*

Description

time series plot across different interventions

Usage

```
time_series_plot(nof1, time = NULL, timestamp = NULL,
  timestamp.format = "%m/%d/%Y %H:%M", Outcome.name = "")
```

Arguments

nof1	nof1 object created using nof1.data
timestamp	time of the nof1 event occurring
timestamp.format	format of the timestamp
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(data$Y))
time_series_plot(nof1, timestamp = timestamp, timestamp.format = "%m-%d-%Y", Outcome.name = "Stress")
```

Index