Package 'nof1ins'

July 21, 2021

Type Package

Title N-of-1 study general analysis tool

Version 0.5.0

Depends R (>= 2.10)

Imports rjags (>= 4-6), splines, combinat, MASS, jsonlite, ggplot2, scales, coda (>= 0.13), RColorBrewer,dplyr, reshape2, tidyr

Author Jiabei Yang, Michael Seo, Christopher Schmid

Maintainer Jiabei Yang <jiabei_yang@brown.edu>

Description A package for running analysis for N of 1 studies. 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 7.1.1

R topics documented:

nof1-package . . . 2
frequency_plot . . . 2
kernel_plot . . . 3
nof1.data . . . 3
nof1.nma.data . . . 5
nof1.nma.run . . . 7
nof1.normal.simulation . . . 8
nof1.run . . . 9
probability_barplot . . . 10
raw_table . . . 10
stacked_percent_barplot . . . 11
summarize_nof1 . . . 12
time_series_plot . . . 12
trt_eff_plot . . . 13
wrap . . . 14
wrap2 . . . 14

Index **15**

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, ...)
```

Arguments

nof1	nof1 object created using nof1.data
...	parameters to pass to geom_bar or geom_histogram.

Examples

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

kernel_plot *Kernel density estimation plot*

Description

Creates a kernel density estimation plot for a specific outcome

Usage

```
kernel_plot(result, comp = T, ...)
```

Arguments

result	Modeling result of class nof1.result produced by nof1.run.
...	parameters to pass to geom_histogram.

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

Description

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

Usage

```
nof1.data(
  Y,
  Treat,
  ord.baseline.Treat = NULL,
  ord.model = NULL,
  response = NULL,
  ncat = NULL,
  bs.trend = F,
  y.time = NULL,
  knots.bt.block = NULL,
  block.no = NULL,
  bs.df = NULL,
  corr.y = F,
  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 NA's included in time order.
Treat	Treatment indicator vector with same length as the outcome. Can be character or numeric.
ord.baseline.Treat	Used for ordinal outcome to define a reference treatment level.
ord.model	Used for ordinal outcome to pick the model. Can be "cumulative" for proportional odds model or "acat" for restricted adjacent category model.
response	Type of outcome. Can be "normal" for continuous outcome, "binomial" for binary outcome, "poisson" for count outcome, or "ordinal" for ordinal or nominal outcome.
ncat	Number of categories. Used in ordinal models.
bs.trend	Indicator for whether the model should adjust for trend using splines. The default is F.
y.time	Parameter used for modeling splines. Time when the outcome is measured.
knots.bt.block	parameter used for modeling splines. Indicator for whether or not knots should be set at the end of each block except for the last block. If TRUE, user should then specify block.no; if FALSE, user should then specify bs.df.
block.no	Block number used for modeling splines for the setting where the knots are set at the end of each block. Block number with the same length as the outcome.
bs.df	Degrees of freedom for modeling splines when knots are not set at the end of each block.
corr.y	Indicator for whether the correlation among measurements shoule be modeled. The default is F.
alpha.prior	Prior for the intercept of the model. Not needed now since we are using treatment-specific intercept model.
beta.prior	Prior for the treatment-specific intercept.
eta.prior	Prior for modelling spline terms.
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.
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.
...	Arguments to be passed to bs().

Value

An object of class "nof1.data" that is used to run the model using nof1.run is a list containing

Y	Outcome
Treat	Treatment

ncat	Number of categories for ordinal response
nobs	Total number of observations in a study
Treat.name	Treatment name
response	Type of outcome
Treat_Treat.name	Vector in the model matrix for *Treat.name*
bs.trend	Indicator for whether the model should adjust for trend using splines
corr.y	Indicator for whether the correlation among measurements shoule be modeled
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.nma.data` *Make an N of 1 object containing data, priors, and a jags model file for (network) meta-analyses*

Description

Make an N of 1 object containing data, priors, and a jags model file for (network) meta-analyses

Usage

```
nof1.nma.data(
  Y,
  Treat,
  baseline.Treat,
  ID,
  response,
  model.linkfunc = NULL,
  model.intcpt = "fixed",
  model.slp = "random",
  ord.ncat = NULL,
  ord.model,
  ord.parallel = NULL,
  lvl2.cov = NULL,
  spline.trend = F,
  trend.type,
  y.time = NULL,
```

```
  knots = NULL,
  trend.df = NULL,
  step.trend = F,
  y.step = NULL,
  corr.y = F,
  alpha.prior = NULL,
  beta.prior = NULL,
  eta.prior = NULL,
  dc.prior = NULL,
  c1.prior = NULL,
  rho.prior = NULL,
  hy.prior = NULL,
  na.rm = T,
  ...
)
```

Arguments

Y	Outcome of the study. This should be a vector with NA's included in time order.
Treat	Treatment indicator vector with the same length as the outcome. Can be character or numeric.
baseline.Treat	Name of the reference treatment.
ID	Participant ID vector with the same length as the outcome.
response	Type of outcome. Can be "normal" for continuous outcome, "binomial" for binary outcome, "poisson" for count outcome, or "ordinal" for ordinal or nominal outcome.
model.linkfunc	Link function in the model.
model.intcpt	Form of intercept.
model.slp	Form of slope. For link function and the forms of intercept and slopes, currently implemented for 1) normal response: "identity"-"fixed/random"-"random", 2) poisson response: "log"-"fixed"-"random", 3) binomial response: "log/logit"-"fixed"-"random".
ord.ncat	Number of categories in ordinal response. The parameters for ordinal outcomes need to be tested.
ord.model	Used for ordinal outcome to pick the model. Can be "cumulative" for cumulative probability model or "acat" for adjacent category model.
ord.parallel	Whether or not the comparisons between categories or cumulative probabilities are parallel.
lvl2.cov	Participant level covariates for heterogeneous treatment effects. Should be a data frame with the columns being the covariates and the number of rows should be equal to the length of the outcome. For fixed-intercept model, participant level covariates will have interaction with treatment (slope) because there are fixed-intercepts adjusting for each participant in the model; for random-intercept model, participant level covariates will have interaction with all treatment-specific intercepts.
spline.trend	Indicator for whether the model should adjust for trend using splines. The default is F.
trend.type	"bs" for basis splines or "ns" for natural splines.

y.time	Time when the outcome is measured. Required when adjusting for trend or correlation (spline.trend/step.trend/corr.y is TRUE). Should be a vector of the same length as the outcome.
knots	Knots in y.time when adjusting for trend using splines. For trend.type = "bs", knots should be set at the end of each block except for the last block; for or trend.type = "ns", knots should be set at the end of each period except for the last period.
trend.df	Degrees of freedom for modeling splines when knots are not set.
step.trend	Indicator for whether to adjust for trend using step functions for each period.
y.step	Period number corresponding to the outcome. Should be a vector of the same length of the outcome.
corr.y	Indicator for whether the correlation among measurements shoule be modeled. The default is F.
alpha.prior	Prior for the fixed-intercepts.
beta.prior	Prior for random intercepts or slopes.
eta.prior	Prior for modelling spline terms or heterogeneous treatment effects.
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 correlation between random effects or the correlation among repeated measurements on each participant.
hy.prior	Prior for the variance of the residual errors for normal response, the standard deviation of random slopes for binary outcome, the variance of random effects for other types of outcomes.

---

nof1.nma.run *Run the model using the nof1 object for (network) meta-analyses*

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.nma.data.

Usage

```
nof1.nma.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.nma.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.

---

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 for individual analysis*

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)
```

---

probability_barplot *Posterior probability barplot*

Description

Creates a posterior probability barplot for treatments

Usage

```
probability_barplot(result.list)
```

Arguments

result.list	A list of one or more modeling results from nof1.run.

---

raw_table *Summary data table for nof1*

Description

Provides a summary data table for the particular outcome in a particular dataset.

Usage

```
raw_table(nof1)
```

Arguments

nof1	nof1 object created using nof1.data

Value

Gives a comprhensive table with several statistical values. Each column indicates the value given. For a normal or poisson response type the following are given:

Treat	The treatment recieved
mean	The average value of the outcome
sd	The standard deviation for the outcome
2.5%	2.5% of the data are equal to or less than this value
50%	50% of the data are equal to or less than this value
97.5%	97.5% of the data are equal to or less than this value

For a binomial or ordinal response type, returns a table where first row is each treatment and the following rows are the the number of data points taken at each possible value.

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)
```

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")
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	Modeling result of class nof1.result produced by nof1.run
alpha	The alpha value for the confidence interval. The default is 0.05.

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 quantiles of one coefficient minus the other when comparing two treatments
p.comp.coef.greater.0	The posterior probability of one coefficient minus the other greater than 0

time_series_plot *time series plot across different interventions*

Description

time series plot across different interventions

Usage

```
time_series_plot(
  result,
  baseline.treat.name = NULL,
  plot.by.treat = T,
  overlay.with.model = F,
  predict.model = F,
  trial.start = NULL,
  trial.end = NULL,
  timestamp.format = NULL,
  ...
)
```

Arguments

result	Modeling result of class nof1.result produced by nof1.run.
baseline.treat.name	Name for reference treatment. Default is NULL.
plot.by.treat	Whether or not the measurements should be plotted in different panels by treatment. The default is T.
overlay.with.model	Whether or not the model prediction should be plotted. The default is F.
predict.model	Indicator for whether or not a prediction line should be plotted over the study period. Option when overlay.with.model = TRUE.
trial.start	Start time of the trial specified with timestamp.format.
trial.end	End time of the trial specified with timestamp.format.
timestamp.format	Format of the trial.start and trial.end.

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")
```

---

trt_eff_plot *Errorbars for the credible interval of the treatment effect*

Description

Creates errorbars for the credible interval of estimated treatment effect

Usage

```
trt_eff_plot(result.list, level = 0.95, ...)
```

Arguments

result.list	A list of one or more modeling results from nof1.run.
level	The level of the credible intervals. The default is 0.95.
...	parameters to pass to geom_errorbar.

---

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

frequency_plot, 2

kernel_plot, 3

nof1-package, 2
nof1.data, 3, *9*
nof1.nma.data, 5, *7, 8*
nof1.nma.run, 7
nof1.normal.simulation, 8
nof1.run, *4*, 9

probability_barplot, 10

raw_table, 10

stacked_percent_barplot, 11
summarize_nof1, 12

time_series_plot, 12
trt_eff_plot, 13

wrap, 14
wrap2, 14