Package 'simml'

July 23, 2025

Type Package

Title Single-Index Models with Multiple-Links

Version 0.3.0

Author Hyung Park, Eva Petkova, Thaddeus Tarpey, R. Todd Ogden

Maintainer Hyung Park <parkh15@nyu.edu>

Description A major challenge in estimating treatment decision rules from a randomized clinical trial dataset with covariates measured at baseline lies in detecting relatively small treatment effect modification-related variability (i.e., the treatment-by-covariates interaction effects on treatment outcomes) against a relatively large non-treatment-related variability (i.e., the main effects of covariates on treatment outcomes). The class of Single-Index Models with Multiple-Links is a novel single-index model specifically designed to estimate a single-index (a linear combination) of the covariates associated with the treatment effect modification-related variability, while allowing a nonlinear association with the treatment outcomes via flexible link functions. The models provide a flexible regression approach to developing treatment decision rules based on patients' data measured at baseline. We refer to Park, Petkova, Tarpey, and Ogden (2020) <doi:10.1016/j.jspi.2019.05.008> and Park, Petkova, Tarpey, and Ogden (2020) <doi:10.1111/biom.13320> (that allows an unspecified X main effect) for detail of the method. The main function of this package is simml().

License GPL-3

Imports mgcv

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2021-05-25 06:50:02 UTC

Contents

`der.link` *A subfunction used in estimation*

Description

This function computes the 1st derivative of the treatment-specific link function with respect to the single index, using finite difference.

Usage

```
der.link(g.fit, eps = 10^(-6))
```

Arguments

`g.fit`	a `mgcv::gam` object
`eps`	a small finite difference used in numerical differentiation.

See Also

`fit.simml`, `simml`

`fit.simml` *Single-index models with multiple-links (workhorse function)*

Description

`fit.simml` is the workhorse function for Single-index models with multiple-links (SIMML). The function estimates a linear combination (a single-index) of covariates X, and models the treatment-specific outcome y, via treatment-specific nonparametrically-defined link functions.

Usage

```
fit.simml(y, A, X, Xm = NULL, aug = NULL, rho = 0,
  family = "gaussian", R = NULL, bs = "ps", k = 8, sp = NULL,
  linear.link = FALSE, method = "GCV.Cp", gamma = 1, max.iter = 20,
  eps.iter = 0.01, trace.iter = TRUE, ind.to.be.positive = NULL,
  scale.si.01 = FALSE, lambda = 0, pen.order = 0, scale.X = TRUE,
  center.X = TRUE, ortho.constr = TRUE, beta.ini = NULL,
  si.main.effect = FALSE, random.effect = FALSE, z = NULL,
  plots = FALSE)
```

Arguments

y	a n-by-1 vector of treatment outcomes; y is a member of the exponential family; any distribution supported by mgcv::gam; y can also be an ordinal categorial response with R categories taking a value from 1 to R.
A	a n-by-1 vector of treatment variable; each element is assumed to take a value on a continuum.
X	a n-by-p matrix of baseline covarates.
Xm	a n-by-q design matrix associated with an X main effect model; the defult is NULL and it is taken as a vector of zeros
aug	a n-by-1 additional augmentation vector associated with the X main effect; the default is NULL and it is taken as a vector of zeros
rho	a tuning parameter associated with the additional augmentation vector aug; the default is 0.
family	specifies the distribution of y; e.g., "gaussian", "binomial", "poisson"; can be any family supported by mgcv::gam; can also be "ordinal", for an ordinal categorical response y.
R	the number of response categories for the case of family = "ordinal".
bs	basis type for the treatment (A) and single-index domains, respectively; the defult is "ps" (p-splines); any basis supported by mgcv::gam can be used, e.g., "cr" (cubic regression splines); see mgcv::s for detail.
k	basis dimension for the treatment (A) and single-index domains, respectively.
sp	smoothing paramter for the treatment-specific link functions; if NULL, then estimated from the data.
linear.link	if TRUE, the link function is restricted to be linear.
method	the smoothing parameter estimation method; "GCV.Cp" to use GCV for unknown scale parameter and Mallows' Cp/UBRE/AIC for known scale; any method supported by mgcv::gam can be used.
gamma	increase this beyond 1 to produce smoother models. gamma multiplies the effective degrees of freedom in the GCV or UBRE/AIC (see mgcv::gam for detail); the default is 1.
max.iter	an integer specifying the maximum number of iterations for beta.coef update.
eps.iter	a value specifying the convergence criterion of algorithm.
trace.iter	if TRUE, trace the estimation process and print the differences in beta.coef.
ind.to.be.positive	for identifiability of the solution beta.coef, the user can restrict the jth (e.g., j=1) component of beta.coef to be positive; by default, we match the "overall" sign of beta.coef with that of the linear estimate (i.e., the initial estimate), by restricting the inner product between the two to be positive.
scale.si.01	if TRUE, re-scale the index coefficients to restrict the index to the interval [0,1]; in such a case, an intercept term is induced.
lambda	a regularization parameter associated with the penalized LS for beta.coef update.

pen.order	0 indicates the ridge penalty; 1 indicates the 1st difference penalty; 2 indicates the 2nd difference penalty, used in a penalized least squares (LS) estimation of beta.coef.
scale.X	if TRUE, scale X to have unit variance.
center.X	if TRUE, center X to have zero mean.
ortho.constr	separates the interaction effects from the main effect (without this, the interaction effect can be confounded by the main effect; the default is TRUE.
beta.ini	an initial value for beta.coef; a p-by-1 vector; the defult is NULL, in which case a linear model estimate is used.
si.main.effect	if TRUE, once the convergence in the estimates of beta.coef is reached, include the main effect associated with the fitted single-index (beta.coef'X) to the final fit; the default is FALSE.
random.effect	if TRUE, as part of the main effects, the user can incorporate z-specific random intercepts.
z	a factor that specifies the random intercepts when random.effect = TRUE.
plots	if TRUE, produce a plot for the estimated effect contrast (for binary treatment cases) (on a linear predictor scale).

Details

SIMML captures the effect of covariates via a single-index and their interaction with the treatment via nonparametric link functions. Interaction effects are determined by distinct shapes of the link functions. The estimated single-index is useful for comparing differential treatment efficacy. The resulting simml object can be used to estimate an optimal treatment decision rule for a new patient with pretreatment clinical information.

Value

a list of information of the fitted SIMML including

beta.coef	the estimated single-index coefficients.
g.fit	a mgcv:gam object containing information about the estimated treatment-specific link functions.
beta.ini	the initial value used in the estimation of beta.coef
beta.path	solution path of beta.coef over the iterations
d.beta	records the change in beta.coef over the solution path, beta.path
scale.X	sd of pretreatment covariates X
center.X	mean of pretreatment covariates X
L	number of different treatment options
p	number of pretreatment covariates X
n	number of subjects
boot.ci	(1-boot.alpha/2) percentile bootstrap CIs (LB, UB) associated with beta.coef

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

`pred.simml`, `simml`

`generate.data` *A data generation function*

Description

`generate.data` generates an example dataset from a mean model that has a "main" effect component and a treatment-by-covariates interaction effect component (and a random component for noise).

Usage

```
generate.data(n = 200, p = 10, family = "gaussian",
  correlationX = 0, sigmaX = 1, sigma = 0.4, s = 2, delta = 1,
  pi.1 = 0.5, true.beta = NULL, true.eta = NULL)
```

Arguments

`n`	sample size.
`p`	dimension of covariates.
`family`	specifies the distribution of the outcome y; "gaussian", "binomial", "poisson"; the defult is "gaussian"
`correlationX`	correlation among the covariates.
`sigmaX`	standard deviation of the covariates.
`sigma`	standard deviation of the random noise term (for gaussian response).
`s`	controls the nonliarity of the treatment-specific link functions that define the interaction effect component. `s=1` linear `s=2` nonlinear
`delta`	controls the intensity of the main effect; can take any intermediate value, e.g., `delta= 1.4`. `delta=1` moderate main effect `delta=2` big main effect
`pi.1`	probability of being assigned to the treatment 1
`true.beta`	a p-by-1 vector of the true single-index coefficients (associated with the interaction effect component); if `NULL`, `true.beta` is set to be `(1, 0.5, 0.25, 0.125, 0,...0)'` (only the first 4 elements are nonzero).
`true.eta`	a p-by-1 vector of the true main effect coefficients; if `NULL`, `true.eta` is set to be `(0,..., 0.125, 0.25, 0.25, 1)'` (only the last 4 elements are nonzero).

Value

y	a n-by-1 vector of treatment outcomes.
A	a n-by-1 vector of treatment indicators.
X	a n-by-p matrix of pretreatment covariates.
SNR	the "signal" (interaction effect) to "nuisance" (main effect) variance ratio (SNR) in the canonical parameter function.
true.beta	the true single-index coefficient vector.
true.eta	the true main effect coefficient vector.
optTr	a n-by-1 vector of treatments, indicating the optimal treatment selections.
value.opt	the "value" implied by the optimal treatment decision rule, optTr.

---

ordinal.data *A function for ordinal categorical response data generation.*

Description

ordinal.data generates ordered category response data (with p covariates and a treatment variable).

Usage

```
ordinal.data(n = 400, p = 10, R = 11, delta = 1, s = "nonlinear",
  sigma = 0)
```

Arguments

n	sample size.
p	dimension of covariates.
R	number of response levels in y
delta	magnitude of "main" effect (i.e., "nuisance" effect) of the covariates; a large delta means a larger "nuisance" variance.
s	type of the treatment-by-covariates interation effect ("linear" or "nonlinear")
sigma	noise sd in the latent variable representation

Value

y	a n-by-1 vector of treatment outcomes.
A	a n-by-1 vector of treatment indicators.
X	a n-by-p matrix of pretreatment covariates.
SNR	the "signal" (interaction effect) to "nuisance" (main effect) variance ratio (SNR) in the canonical parameter function.
true.beta	the true single-index coefficient vector.
delta	magnitude of "main" effect.
s	type of the treatment-by-covariates interation effect.

`pred.simml` *SIMML prediction function*

Description

This function makes predictions from an estimated SIMML, given a (new) set of pretreatment covariates. The function returns a set of predicted outcomes for each treatment condition and a set of recommended treatment assignments (assuming a larger value of the outcome is better).

Usage

```
pred.simml(simml.obj, newX = NULL, newA = NULL, newXm = NULL,
  single.index = NULL, type = "link", maximize = TRUE)
```

Arguments

`simml.obj`	a `simml` object
`newX`	a (n-by-p) matrix of new values for the covariates X at which predictions are to be made.
`newA`	a (n-by-L) matrix of new values for the treatment A at which predictions are to be made.
`newXm`	a (n-by-q) matrix of new values for the covariates associated with the fitted main effect Xm at which predictions are to be made.
`single.index`	a length n vector specifying new values for the single-index at which predictions are to be made; the default is `NULL`.
`type`	the type of prediction required; the default "response" is on the scale of the response variable; the alternative "link" is on the scale of the linear predictors.
`maximize`	the default is `TRUE`, assuming a larger value of the outcome is better; if `FALSE`, a smaller value is assumed to be prefered.

Value

`pred.new`	a (n-by-L) matrix of predicted values; each column represents a treatment option.
`trt.rule`	a (n-by-1) vector of suggested treatment assignments

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

`simml`,`fit.simml`

simml *Single-index models with multiple-links (main function)*

Description

simml is the wrapper function for Single-index models with multiple-links (SIMML). The function estimates a linear combination (a single-index) of covariates X, and models the treatment-specific outcome y, via treatment-specific nonparametrically-defined link functions.

Usage

```
simml(y, A, X, Xm = NULL, aug = NULL, family = "gaussian",
  R = NULL, bs = "cr", k = 8, sp = NULL, linear.link = FALSE,
  method = "GCV.Cp", gamma = 1, rho = 0, beta.ini = NULL,
  ind.to.be.positive = NULL, scale.si.01 = FALSE, max.iter = 20,
  eps.iter = 0.01, trace.iter = TRUE, lambda = 0, pen.order = 0,
  scale.X = TRUE, center.X = TRUE, ortho.constr = TRUE,
  si.main.effect = FALSE, random.effect = FALSE, z = NULL,
  plots = FALSE, bootstrap = FALSE, nboot = 200, boot.conf = 0.95,
  seed = 1357)
```

Arguments

y	a n-by-1 vector of treatment outcomes; y is a member of the exponential family; any distribution supported by mgcv::gam; y can also be an ordinal categorial response with R categories taking a value from 1 to R.
A	a n-by-1 vector of treatment variable; each element is assumed to take a value in a finite discrete space.
X	a n-by-p matrix of baseline covarates.
Xm	a n-by-q design matrix associated with an X main effect model; the defult is NULL and it is taken as a vector of zeros
aug	a n-by-1 additional augmentation vector associated with the X main effect; the default is NULL and it is taken as a vector of zeros
family	specifies the distribution of y; e.g., "gaussian", "binomial", "poisson"; can be any family supported by mgcv::gam; can also be "ordinal", for an ordinal categorical response y.
R	the number of response categories for the case of family = "ordinal".
bs	basis type for the treatment (A) and single-index joint effect; the defult is "ps" (p-splines); any basis supported by mgcv::gam can be used, e.g., "cr" (cubic regression splines); see mgcv::s for detail.
k	basis dimension for the spline-type-represented treatment-specific link functions.
sp	smoothing paramter for the treatment-specific link functions; if NULL, then estimated from the data.

linear.link	if TRUE, the link function is restricted to be linear.
method	the smoothing parameter estimation method; "GCV.Cp" to use GCV for unknown scale parameter and Mallows' Cp/UBRE/AIC for known scale; any method supported by mgcv::gam can be used.
gamma	increase this beyond 1 to produce smoother models. gamma multiplies the effective degrees of freedom in the GCV or UBRE/AIC (see mgcv::gam for detail); the default is 1.
rho	a tuning parameter associated with the additional augmentation vector aug; the default is 0.
beta.ini	an initial value for beta.coef; a p-by-1 vector; the defult is NULL, in which case a linear model estimate is used.
ind.to.be.positive	for identifiability of the solution beta.coef, the user can restrict the jth (e.g., j=1) component of beta.coef to be positive; by default, we match the "overall" sign of beta.coef with that of the linear estimate (i.e., the initial estimate), by restricting the inner product between the two to be positive.
scale.si.01	if TRUE, re-scale the index coefficients to restrict the index to the interval [0,1]; in such a case, an intercept term is induced.
max.iter	an integer specifying the maximum number of iterations for beta.coef update.
eps.iter	a value specifying the convergence criterion of algorithm.
trace.iter	if TRUE, trace the estimation process and print the differences in beta.coef.
lambda	a regularization parameter associated with the penalized LS for beta.coef update; the default is 0, and the index coefficients are not penalized.
pen.order	0 indicates the ridge penalty; 1 indicates the 1st difference penalty; 2 indicates the 2nd difference penalty, used in a penalized least squares (LS) estimation of beta.coef.
scale.X	if TRUE, scale X to have unit variance.
center.X	if TRUE, center X to have zero mean.
ortho.constr	separates the interaction effects from the main effect (without this, the interaction effect can be confounded by the main effect; the default is TRUE.
si.main.effect	if TRUE, once the convergence in the estimates of beta.coef is reached, include the main effect associated with the fitted single-index (beta.coef'X) to the final fit; the default is FALSE.
random.effect	if TRUE, as part of the main effects, the user can incorporate z-specific random intercepts.
z	a factor that specifies the random intercepts when random.effect = TRUE.
plots	if TRUE, produce a plot for the estimated effect contrast (for binary treatment cases) (on a linear predictor scale).
bootstrap	if TRUE, compute bootstrap confidence intervals for the single-index coefficients, beta.coef; the default is FALSE.
nboot	when bootstrap=TRUE, a value specifying the number of bootstrap replications.
boot.conf	a value specifying the confidence level of the bootstrap confidence intervals; the defult is boot.conf = 0.95.
seed	when bootstrap=TRUE, randomization seed used in bootstrap resampling.

Details

SIMML captures the effect of covariates via a single-index and their interaction with the treatment via nonparametric link functions. Interaction effects are determined by distinct shapes of the link functions. The estimated single-index is useful for comparing differential treatment efficacy. The resulting `simml` object can be used to estimate an optimal treatment decision rule for a new patient with pretreatment clinical information.

Value

a list of information of the fitted SIMML including

`beta.coef`	the estimated single-index coefficients.
`g.fit`	a `mgcv:gam` object containing information about the estimated treatment-specific link functions.
`beta.ini`	the initial value used in the estimation of `beta.coef`
`beta.path`	solution path of `beta.coef` over the iterations
`d.beta`	records the change in `beta.coef` over the solution path, `beta.path`
`scale.X`	sd of pretreatment covariates X
`center.X`	mean of pretreatment covariates X
`L`	number of different treatment options
`p`	number of pretreatment covariates X
`n`	number of subjects
`boot.ci`	(1-boot.alpha/2) percentile bootstrap CIs (LB, UB) associated with `beta.coef`

Author(s)

Park, Petkova, Tarpey, Ogden

See Also

`pred.simml`, `fit.simml`

Examples

```
family <- "gaussian"   #"poisson"
delta = 1              # moderate main effect
s=2                    # if s=2 (s=1), a nonlinear (linear) contrast function
n=500                  # number of subjects
p=10                   # number of pretreatment covariates

# generate training data
data <- generate.data(n= n, p=p, delta = delta, s= s, family = family)
data$SNR  # the ratio of interactions("signal") vs. main effects("noise")
A <- data$A
y <- data$y
X <- data$X
```

```
# generate testing data
data.test <- generate.data(n=10^5, p=p, delta = delta,  s= s, family = family)
A.test <- data.test$A
y.test <- data.test$y
X.test <- data.test$X
data.test$value.opt     # the optimal "value"


# fit SIMML
#1) SIMML without X main effect
simml.obj1 <- simml(y, A, X, family = family)

#2) SIMML with X main effect (estimation efficiency for the g term of SIMML can be improved)
simml.obj2 <- simml(y, A, X, Xm = X, family = family)


# apply the estimated SIMML to the testing set and obtain treatment assignment rules.
simml.trt.rule1 <- pred.simml(simml.obj1, newX= X.test)$trt.rule
# "value" estimation (estimated by IPWE)
simml.value1 <-  mean(y.test[simml.trt.rule1 == A.test])
simml.value1

simml.trt.rule2 <- pred.simml(simml.obj2, newX= X.test)$trt.rule
simml.value2 <-  mean(y.test[simml.trt.rule2 == A.test])
simml.value2

# compare these to the optimal "value"
data.test$value.opt



# fit MC (modified covariates) model of Tien et al 2014
n.A <- summary(as.factor(A)); pi.A <- n.A/sum(n.A)
mc  <- (as.numeric(A) + pi.A[1] -2) *cbind(1, X)  # 0.5*(-1)^as.numeric(A) *cbind(1, X)
mc.coef  <-  coef(glm(y ~ mc, family =  family))
mc.trt.rule <- (cbind(1, X.test) %*% mc.coef[-1] > 0) +1
# "value" estimation (estimated by IPWE)
mc.value  <-  mean(y.test[mc.trt.rule == A.test])
mc.value


# visualization of the estimated link functions of SIMML
simml.obj1$beta.coef        # estimated single-index coefficients
g.fit <- simml.obj1$g.fit  # estimated trt-specific link functions; "g.fit" is a mgcv::gam object.
#plot(g.fit)


# can improve visualization by using the package "mgcViz"
#install.packages("mgcViz")
# mgcViz depends on "rgl". "rgl" depends on XQuartz, which you can download from xquartz.org
#library(mgcViz)
# transform the "mgcv::gam" object to a "mgcViz" object (to improve visualization)
```

```
g.fit <- getViz(g.fit)

plot1  <- plot( sm(g.fit,1) )  # for treatment group 1
plot1 + l_fitLine(colour = "red") + l_rug(mapping = aes(x=x, y=y), alpha = 0.8) +
  l_ciLine(mul = 5, colour = "blue", linetype = 2) +
  l_points(shape = 19, size = 1, alpha = 0.1) +
  xlab(expression(paste("z = ", alpha*minute, "x")))  +  ylab("y") +
  ggtitle("Treatment group 1 (Trt =1)") +  theme_classic()

plot2 <- plot( sm(g.fit,2) )   # for treatment group 2
plot2 + l_fitLine(colour = "red") + l_rug(mapping = aes(x=x, y=y), alpha = 0.8) +
  l_ciLine(mul = 5, colour = "blue", linetype = 2) +
  l_points(shape = 19, size = 1, alpha = 0.1) +
  xlab(expression(paste("z = ", alpha*minute, "x"))) +ylab("y") +
  ggtitle("Treatment group 2 (Trt =2)") + theme_classic()


trans = function(x) x + g.fit$coefficients[2]
plotDiff(s1 = sm(g.fit, 2), s2 = sm(g.fit, 1), trans=trans) +  l_ciPoly() +
  l_fitLine() + geom_hline(yintercept = 0, linetype = 2) +
  xlab(expression(paste("z = ", alpha*minute, "x")) ) +
  ylab("(Treatment 2 effect) - (Treatment 1 effect)") +
  ggtitle("Contrast between two treatment effects") +
  theme_classic()


# yet another way of visualization, using ggplot2
#library(ggplot2)
dat  <- data.frame(y= simml.obj1$g.fit$model$y,
                   x= simml.obj1$g.fit$model$single.index,
                   Treatment= simml.obj1$g.fit$model$A)
g.plot<- ggplot(dat, aes(x=x,y=y,color=Treatment,shape=Treatment,linetype=Treatment))+
   geom_point(aes(color=Treatment, shape=Treatment), size=1, fill="white") +
   scale_colour_brewer(palette="Set1", direction=-1) +
   xlab(expression(paste(beta*minute,"x"))) + ylab("y")
g.plot + geom_smooth(method=gam, formula= y~ s(x, bs=simml.obj1$bs, k=simml.obj1$k),
                     se=TRUE, fullrange=TRUE, alpha = 0.35)



# can obtain bootstrap CIs for beta.coef.
simml.obj <- simml(y,A,X,Xm=X, family=family,bootstrap=TRUE,nboot=15)  #nboot=500.
simml.obj$beta.coef
round(simml.obj$boot.ci,3)

# compare the estimates to the true beta.coef.
data$true.beta



# an application to data with ordinal categorical response
dat <- ordinal.data(n=500, p=5, R = 11,  # 11 response levels
                    s = "nonlinear",     # nonlinear interactions
```

```
                    delta = 1)
dat$SNR
y <- dat$y  # ordinal response
X <- dat$X  # X matrix
A <- dat$A  # treatment
dat$true.beta  # the "true" single-index coefficient


# 1) fit a cumulative logit simml, with a flexible link function
res <-  simml(y,A,X, family="ordinal", R=11)
res$beta.coef  # single-index coefficients.
res$g.fit$family$getTheta(TRUE)  # the estimated R-1 threshold values.

# 2) fit a cumulative logit simml, with a linear link function
res2 <-  simml(y,A,X, family="ordinal", R=11, linear.link = TRUE)
res2$beta.coef  # single-index coefficients.


family = mgcv::ocat(R=11)  # ocat: ordered categorical response family, with R categories.
# the treatment A's effect.
tmp <- mgcv::gam(y ~ A, family =family)
exp(coef(tmp)[2])  #odds ratio (OR) comparing treatment A=2 vs. A=1.

ind2 <- pred.simml(res)$trt.rule ==2  # subgroup recommended with A=2 under SIMML ITR
tmp2 <- mgcv::gam(y[ind2] ~ A[ind2], family = family)
exp(coef(tmp2)[2]) #OR comparing treatment A=2 vs. A=1, for subgroup recommended with A=2

ind1 <- pred.simml(res)$trt.rule ==1  # subgroup recommended with A=1 under SIMML ITR
tmp1 <- mgcv::gam(y[ind1] ~ A[ind1], family = family)
exp(coef(tmp1)[2]) #OR comparing treatment A=2 vs. A=1, for subgroup recommended with A=2
```

Index