Package 'EPLSIM'

November 4, 2026

Type Package

Title Partial Linear Single Index Models for Environmental Mixture Analysis

Version 1.1.0

Date 2026-11-04

Maintainer Yuyan Wang <yuyan.wang@nyumc.org>

Description Collection of ancillary functions and utilities for Partial Linear Single Index Models for Environmental mixture analyses, which currently provides functions for scalar, binary and count outcomes. The outputs of these functions include the single index function, single index coefficients, partial linear coefficients, mixture overall effect, exposure main and interaction effects, and differences of quartile effects. In the future, we will add functions for ordinal, survival, and longitudinal outcomes, as well as models for time-dependent exposures. See Wang et al (2020) <doi:10.1186/s12940-020-00644-4> for an overview.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 2.10)

Imports splines, ggplot2, MASS, ciTools, mgcv, stats, graphics, utils

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

URL https://github.com/YuyanWangSixTwo/EPLSIM

BugReports https://github.com/YuyanWangSixTwo/EPLSIM/issues

VignetteBuilder knitr

LazyData true

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation no

Author Yuyan Wang [aut, cre] (ORCID: <https://orcid.org/0000-0003-3747-0762>),
Mengling Liu [aut, ctb],
Myeonggyun Lee [ctb]

Repository CRAN

Date/Publication 2026-11-04 18:10:09 UTC

Contents

confounder.trans 2
e.interaction.plot 3
e.main.plot 5
interquartile.quartile.plot 6
mixture.overall.plot 7
nhanes 8
nhanes.new 9
plsi.log.auto 10
plsi.log.auto.constraint 13
plsi.logistic.auto 16
plsi.logistic.auto.constraint 18
plsi.lr.auto 21
plsi.lr.auto.constraint 22
plsi.lr.v1 24
plsi.lr.v2 26
si.coef.plot 27
si.fun.plot 28

Index **30**

`confounder.trans` *Transformation for confounders vector Z before modeling*

Description

Transformation for confounders vector Z before modeling

Usage

```
confounder.trans(Z_continuous, Z_discrete, data)
```

Arguments

`Z_continuous`	A character name vector for continuous confounders
`Z_discrete`	A character name vector for discrete confounders
`data`	Orginial data set

Value

Transformed confounder vector and data set ready for further analysis.

Author(s)

Yuyan Wang

Examples

```
# example to normalize the continuous confounders and
# make dummy variables for categorical confounders
dat.cov <- data.frame(
  age = c(1.5, 2.3, 3.1, 4.8, 5.2),
  sex = c(1, 2, 1, 2, 2),
  race = c(1, 2, 3, 4, 5)
)

# specify the confounder vector
Z.name <- c("age", "sex", "race")

# set levels and make the reference level first for categorical confounders
dat.cov$sex <- factor(dat.cov$sex, 1:2, c('Male', 'Female'))
dat.cov$race <- factor(dat.cov$race,1:5,c("NH-White", "NH-Black",
                                          "MexicanAmerican", "OtherRace", "Hispanic"))

# transform the confounder vector and check
cov_m <- confounder.trans(Z_continuous = c("age"), Z_discrete = c("sex", "race"), data = dat.cov)
Z.name <- cov_m$New.Name
dat.cov <- cov_m$Updated.data
print(Z.name)
```

---

e.interaction.plot *Plot interaction effect of two exposures from PLSIM*

Description

Plot interaction effect of two exposures from PLSIM

Usage

```
e.interaction.plot(
  fit,
  data,
  exp_1,
  exp_2,
  type = c("linear", "logistic", "log")
)
```

Arguments

fit	Fitted model from `plsi.lr.auto()`, `plsi.logistic.auto()`, or `plsi.log.auto()`
data	Original data set
exp_1	Exposure 1 name
exp_2	Exposure 2 name

type — Which outcome scale to plot. "linear" (default) plots the predicted (continuous) outcome, for a fit from plsi.lr.auto(). "logistic" plots the predicted probability, for a fit from plsi.logistic.auto(). "log" plots the predicted count, for a fit from plsi.log.auto().

Value

Plot of interaction effect of two exposures with others at average level. Invisibly returns a list with elements panel_1 and panel_2, the data frames underlying the left and right panels respectively (predicted values at each combination of exp_1/ exp_2 value and the other exposure's quantiles).

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to plot interaction effect of two exposures of continuous outcome
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

# plot two exposures' interaction effect of predicted (continuous) outcome
e.interaction.plot(model_lr_auto, data, "X4_a.tocopherol", "X3_g.tocopherol", type = "linear")
e.interaction.plot(model_lr_auto, data, "X4_a.tocopherol", "X10_2.3.4.6.7.8.hxcdf", type = "linear")

# exchange exposures' names
e.interaction.plot(model_lr_auto, data, "X8_3.3.4.4.5.pncb", "X6_PCB156", type = "linear")
e.interaction.plot(model_lr_auto, data, "X6_PCB156", "X8_3.3.4.4.5.pncb", type = "linear")

## End(Not run)
```

`e.main.plot` *Plot single exposure's main effect from PLSIM*

Description

Plot single exposure's main effect from PLSIM

Usage

```
e.main.plot(
  fit,
  data,
  exp_name,
  type = c("linear", "logistic", "log"),
  outlier.range = 10
)
```

Arguments

`fit`	Fitted model from `plsi.lr.auto()`, `plsi.logistic.auto()`, or `plsi.log.auto()`
`data`	Original data set
`exp_name`	Exposure name
`type`	Which outcome scale to plot. `"linear"` (default) plots the predicted (continuous) outcome, for a `fit` from `plsi.lr.auto()`. `"logistic"` plots the predicted probability, for a `fit` from `plsi.logistic.auto()`. `"log"` plots the predicted count, for a `fit` from `plsi.log.auto()`.
`outlier.range`	`range` argument passed to `boxplot()` to identify and drop extreme values of the raw exposure before plotting. Default 10 (a permissive threshold; increase to drop fewer points, or set to `Inf` to keep all points).

Value

Plot of exposure's main effect with other exposures at their confounder-adjusted reference level. Invisibly returns the data frame used to build the plot.

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to plot some exposure's main effect of continuous outcome
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
```

```
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

# plot some exposure's main effect of predicted (continuous) outcome
e.main.plot(model_lr_auto, data, exp_name = c("X4_a.tocopherol"), type = "linear")
e.main.plot(model_lr_auto, data, exp_name = c("X5_PCB99"), type = "linear")
e.main.plot(model_lr_auto, data, exp_name = c("X10_2.3.4.6.7.8.hxcdf"), type = "linear")

## End(Not run)
```

`interquartile.quartile.plot` *Plot interquartile effect of specific exposure based on quartile of other exposures*

Description

Plot interquartile effect of specific exposure based on quartile of other exposures

Usage

```
interquartile.quartile.plot(fit, data, type = c("linear", "logistic", "log"))
```

Arguments

`fit`	Fitted model from `plsi.lr.auto()`, `plsi.logistic.auto()`, or `plsi.log.auto()`
`data`	Original data set
`type`	Which outcome scale to plot. `"linear"` (default) plots the difference in predicted (continuous) outcome, for a `fit` from `plsi.lr.auto()`. `"logistic"` plots the difference in predicted probability, for a `fit` from `plsi.logistic.auto()`. `"log"` plots the difference in predicted count, for a `fit` from `plsi.log.auto()`.

Value

Plot of main interquartile effect of exposure based on quartile of other exposures

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to interquartile effect based on quartile of other exposures of continuous outcome
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

# plot interquartile quartile of difference in predicted (continuous) outcome
interquartile.quartile.plot(model_lr_auto, data, type = "linear")

## End(Not run)
```

`mixture.overall.plot` *Plot mixture's overall effect based on quantile of exposures from PLSIM*

Description

Plot mixture's overall effect based on quantile of exposures from PLSIM

Usage

```
mixture.overall.plot(fit, data, type = c("linear", "logistic", "log"))
```

Arguments

`fit`	Fitted model from `plsi.lr.auto()`, `plsi.logistic.auto()`, or `plsi.log.auto()`
`data`	Original data set
`type`	Which outcome scale to plot. `"linear"` (default) plots the predicted (continuous) outcome, for a `fit` from `plsi.lr.auto()`. `"logistic"` plots the predicted probability, for a `fit` from `plsi.logistic.auto()`. `"log"` plots the predicted count, for a `fit` from `plsi.log.auto()`.

Value

Plot of predicted outcomes based on quantile of exposures

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to plot mixture's overall effect of continuous outcome
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

# plot mixture overall effect of predicted (continuous) outcome
mixture.overall.plot(model_lr_auto, data, type = "linear")

## End(Not run)
```

---

nhanes *This is data from NHANES 2003–2004*

Description

A data set containing outcome triglyceride, ten exposures, and three confounders.

Usage

```
nhanes
```

Format

An object of class `data.frame` with 800 rows and 14 columns.

Details

triglyceride outcome triglyceride level, unite mg/dl

a1.trans.b.carotene exposure: trans-b-carotene (ug/dL)

a5.Retinol exposure: retinol (ug/dL)

a6.g.tocopherol exposure: g-tocopherol (ug/dL)

a7.a.Tocopherol exposure: a-tocopherol (ug/dL)

a10.PCB99 exposure: Polychlorinated Biphenyl (PCB) 99 Lipid Adj (ng/g)

a13.PCB156 exposure: Polychlorinated Biphenyl (PCB) 156 Lipid Adj (ng/g)

a19.PCB206 exposure: Polychlorinated Biphenyl (PCB) 206 Lipid Adj (ng/g)

a20.3.3.4.4.5.pncb exposure: 3,3,4,4,5-Pentachlorobiphenyl (pncb) Lipid Adj (pg/g)

a21.1.2.3.4.7.8.hxcdf exposure: 1,2,3,4,7,8-hxcdf Lipid Adj (pg/g)

a22.2.3.4.6.7.8.hxcdf exposure: 2,3,4,6,7,8-hxcdf Lipid Adj (pg/g)

age subject age at measurement

sex subject sex

race subject race

Author(s)

Yuyan Wang <yuyan.wang@nyumc.org>

`nhanes.new` *This is an updated data from original data based on NHANES 2003–2004 survey*

Description

A data set containing outcome triglyceride, re-named ten exposures, and transformed confounders.

Usage

```
nhanes.new
```

Format

An object of class `data.frame` with 789 rows and 17 columns.

Details

triglyceride outcome triglyceride level, unite mg/dl

X1_trans.b.carotene renamed exposure: trans-b-carotene (ug/dL)

X2_retinol renamed exposure: retinol (ug/dL)

X3_g.tocopherol renamed exposure: g-tocopherol (ug/dL)

X4_a.tocopherol renamed exposure: a-tocopherol (ug/dL)

X5_PCB99 renamed exposure: Polychlorinated Biphenyl (PCB) 99 Lipid Adj (ng/g)

X6_PCB156 renamed exposure: Polychlorinated Biphenyl (PCB) 156 Lipid Adj (ng/g)

X7_PCB206 renamed exposure: Polychlorinated Biphenyl (PCB) 206 Lipid Adj (ng/g)

X8_3.3.4.4.5.pncb renamed exposure: 3,3,4,4,5-Pentachlorobiphenyl (pncb) Lipid Adj (pg/g)

X9_1.2.3.4.7.8.hxcdf renamed exposure: 1,2,3,4,7,8-hxcdf Lipid Adj (pg/g)

X10_2.3.4.6.7.8.hxcdf renamed exposure: 2,3,4,6,7,8-hxcdf Lipid Adj (pg/g)

AGE.c rescaled continuous confounder: subject age at measurement

SEX.Female categorical confounder dummy variable: subject sex as Female

RACE.NH.Black categorical dummy variable: subject race as Non-Hispanic Black

RACE.MexicanAmerican categorical dummy variable: subject race as Mexican American

RACE.OtherRace categorical dummy variable: subject race as Other Races

RACE.Hispanic categorical dummy variable: subject race as Hispanic

Author(s)

Yuyan Wang <yuyan.wang@nyumc.org>

`plsi.log.auto` *Partial linear single index log-linear regression with automatic smoothness selection*

Description

Count-outcome analog of `plsi.lr.auto()` / `plsi.logistic.auto()`: same partial linear single index model, but for non-negative count data via a log link. The link function is estimated with a penalized regression spline (`mgcv::gam`, REML smoothing-parameter selection), so there is no `spline.num` to hand-tune, only an upper bound `k` on basis complexity, with REML shrinking unneeded wiggliness toward a straight line on the log scale.

Usage

```
plsi.log.auto(
  data,
  Y.name,
  X.name,
  Z.name,
  family = c("nb", "poisson"),
  k = 10,
  bs = "cr",
  initial.random.num = 5,
  seed = 2026
)
```

Arguments

`data`	A data set including all needed variables
`Y.name`	Variable name for the count outcome. Must be non-negative integers (0, 1, 2, ...).
`X.name`	Variable name vector for exposures
`Z.name`	Variable name vector for confounders
`family`	Count distribution to use. `"nb"` (default) fits a negative binomial GAM (`mgcv::nb()`), which estimates its own dispersion parameter and is the safer default for real count data, which is usually overdispersed (variance > mean) relative to Poisson. `"poisson"` fits a Poisson GAM, valid only if the mean-equals- variance assumption actually holds; if unsure, leave the default.
`k`	Upper bound on basis dimension for the smooth link function. Default 10.
`bs`	Smooth basis type passed to `mgcv::s()`; default `"cr"` (cubic regression spline).
`initial.random.num`	Number of random initials for the single-index direction search
`seed`	A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2026.

Details

As with `plsi.logistic.auto()`, the "confounder-free" single-index model cannot be built by subtracting confounder effects from `y` on the response scale (counts aren't additive that way either), and it cannot be built by residualizing the full model's own *fitted* linear predictor, that would leave a value that is already an exact smooth function of the single index, so refitting a smooth to it would just retrace the same curve with near-zero (and spuriously tiny) standard errors. Instead, the confounder contribution from the full model is fixed as a known `offset()` and a second, confounder-free count GAM (same `family` as the full model) is refit directly against the actual observed `y`, preserving real count sampling variability. Predictions from this model are made at `offset = 0`, consistent with how the offset was netted out of every training point.

Value

A list of model estimation and prediction results, structured analogously to `plsi.logistic.auto()`'s output:

- `si.coefficient`: Single-index direction estimates and inference.
- `confounder.coefficient`: Confounder log-rate coefficients, plus rate ratios and their 95% CIs.
- `si.fun`: The estimated single-index link function, on both the log scale (`fit`/`se`/`lwr`/`upr`) and the count scale (`count.fit`/`count.lwr`/`count.upr`), obtained by exponentiating the log-scale CI so it stays non-negative.
- `si.fun.model`: A confounder-free `gam` of the count outcome on the single index alone (via a fixed offset for confounder contribution), predict() from this needs only `single_index_estimated`, not the original confounders. See Details.
- `full.model`: The full fitted `gam` (smooth + confounders), kept for reference/diagnostics.

Author(s)

Yuyan Wang

Examples

```
## Not run:
data(nhanes.new)
data <- nhanes.new

X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

# demo count outcome (illustrative only as nhanes.new has no native count variable).
Simulated from a *true* single-index combination of the exposures, run through a
nonlinear log-rate link, plus a confounder effect, so the example actually has an
exposure-outcome relationship for plsi.log.auto() to recover, rather than
depending on the confounder alone.

set.seed(2026)
beta_true <- c(0.30, -0.20, 0.10, 0.40, -0.30, 0.20, -0.10, 0.25, -0.15, 0.35)
beta_true <- beta_true / sqrt(sum(beta_true^2)) # unit norm, matching the model's
# own identifiability constraint
x_std <- scale(data[, X.name]) # standardize so the demo index is on a sane, comparable scale
single_index_true <- as.vector(x_std %*% beta_true)
log_rate <- 0.3 + 0.4 * sin(single_index_true) + 0.05 * data$AGE.c
data$n.events <- stats::rpois(nrow(data), lambda = exp(log_rate))

Y.name <- "n.events"

k <- 10
bs <- "cr"
```

```
initial.random.num <- 1
seed <- 2026

model_log_auto <- plsi.log.auto(data = data, Y.name = Y.name, X.name = X.name,
                      Z.name = Z.name, family = "nb", k = k, bs = bs,
                      initial.random.num = initial.random.num, seed = seed)

## End(Not run)
```

`plsi.log.auto.constraint` *Partial linear single index log-linear (count) regression with automatic smoothness selection and built-in identifiability constraint*

Description

Count-outcome analog of `plsi.lr.auto.constraint()`: the same model as `plsi.log.auto()` (penalized regression spline link via `mgcv::gam` with a log link, negative binomial or Poisson family, REML smoothing-parameter selection), but with the single-index direction searched over a reparameterization that satisfies the identifiability constraint by construction instead of normalizing after the fact.

Usage

```
plsi.log.auto.constraint(
  data,
  Y.name,
  X.name,
  Z.name,
  family = c("nb", "poisson"),
  k = 10,
  bs = "cr",
  initial.random.num = 5,
  seed = 2026,
  maxit = 100
)
```

Arguments

`data`	A data set including all needed variables
`Y.name`	Variable name for the count outcome. Must be non-negative integers (0, 1, 2, ...).
`X.name`	Variable name vector for exposures (at least 2).
`Z.name`	Variable name vector for confounders
`family`	Count distribution to use. `"nb"` (default) fits a negative binomial GAM (`mgcv::nb()`), which estimates its own dispersion parameter and is the safer default for real, usually overdispersed, count data. `"poisson"` fits a Poisson GAM, valid only if the mean-equals-variance assumption holds.

`k` Upper bound on basis dimension for the smooth link function. Default 10.

`bs` Smooth basis type passed to `mgcv::s()`; default `"cr"` (cubic regression spline).

`initial.random.num`
Number of random initials for the single-index direction search

`seed` A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2026.

`maxit` Maximum number of Nelder-Mead iterations per starting value for the σ-space search. Default 100, matching `plsi.lr.auto.constraint()`. With many exposures this is often too few for the simplex to converge, which can leave the numerical information matrix indefinite and the single-index SEs inflated; check `optim.convergence` (and the warning issued when the selected start did not converge) and increase `maxit` if needed. Ignored when there are exactly two exposures (Brent search).

Details

The single-index coefficient β is constrained to $\|\beta\| = 1$ and $\beta_1 > 0$ for identifiability. The search is reparameterized in terms of $\sigma = (\sigma_1, \ldots, \sigma_{p-1})$ via $\beta(\sigma) = \{(1 - \|\sigma\|^2)^{1/2}, \sigma_1, \ldots, \sigma_{p-1}\}^T$. Both constraints hold automatically for any σ with $\|\sigma\| < 1$, so `optim()` runs unconstrained over σ. Standard errors for β are obtained from the Hessian in σ-space via the delta method.

The anchor exposure, the one assigned β_1, the coordinate constrained > 0, is chosen as the candidate exposure with the largest *absolute* correlation with the count outcome, so an exposure strongly but negatively associated with the outcome can still be selected as the anchor. Which exposure was selected is reported in `attr(si.coefficient, "note")` and in `anchor.note` of the returned list.

The confounder-free single-index model is built exactly as in `plsi.log.auto()`: the confounder contribution from the full model is fixed as a known `offset()` and a second count `gam` (same `family`) is refit directly against the observed `y`.

Standard errors are computed from a numerical Hessian in a local chart of the unit sphere anchored on the exposure with the largest $|\beta_j|$ (not necessarily the $\beta_1 > 0$ anchor), then mapped to β by the delta method. This keeps the finite-difference steps away from the $\|\sigma\| = 1$ boundary when the estimated β_1 is small; the point estimates are unaffected.

With exactly two exposures, σ is one-dimensional; the search then uses `optim(method = "Brent")` on $(-1, 1)$ instead of Nelder-Mead, which `optim()` flags as unreliable in one dimension.

The returned object works with all plotting functions using `type = "log"`.

Value

A list of model estimation and prediction results, structured analogously to `plsi.log.auto()`'s output, plus:

- `anchor.note`: Which exposure was used as the anchor ($\beta_1 > 0$) and why; also attached as `attr(si.coefficient, "note")`.
- `initial.sigma.table`: The starting values actually used by the σ-space search.

- optim.convergence: For every starting value, the optim() convergence code (0 = converged, 1 = maxit reached), the number of log-likelihood evaluations, and the final log-likelihood.

The remaining elements are as in plsi.log.auto():

- si.coefficient: Single-index direction estimates and inference.
- confounder.coefficient: Confounder log-rate coefficients, plus rate ratios and their 95% CIs.
- si.fun: The estimated single-index link function, on both the log scale (fit/se/lwr/upr) and the count scale (count.fit/count.lwr/count.upr).
- si.fun.model: A confounder-free count gam on the single index alone (confounder contribution held as a fixed offset); predict() from this needs only single_index_estimated (and confounder_offset = 0).
- full.model: The full fitted gam (smooth + confounders).

Author(s)

Yuyan Wang

Examples

```
## Not run:
data(nhanes.new)
data <- nhanes.new

X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

# demo count outcome (illustrative only as nhanes.new has no native count
# variable), simulated from a true single-index combination of the exposures
set.seed(2026)
beta_true <- c(0.30, -0.20, 0.10, 0.40, -0.30, 0.20, -0.10, 0.25, -0.15, 0.35)
beta_true <- beta_true / sqrt(sum(beta_true^2))
x_std <- scale(data[, X.name])
single_index_true <- as.vector(x_std %*% beta_true)
log_rate <- 0.3 + 0.4 * sin(single_index_true) + 0.05 * data$AGE.c
data$n.events <- stats::rpois(nrow(data), lambda = exp(log_rate))

model_log_con <- plsi.log.auto.constraint(data = data, Y.name = "n.events",
                   X.name = X.name, Z.name = Z.name, family = "nb",
                   k = 10, bs = "cr", initial.random.num = 1, seed = 2026)
model_log_con$si.coefficient
si.fun.plot(model_log_con$si.fun, type = "log")

## End(Not run)
```

`plsi.logistic.auto` *Partial linear single index logistic regression with automatic smoothness selection*

Description

Binary-outcome analog of `plsi.lr.auto()`: same partial linear single index model, but with a logistic link. The link function is estimated with a penalized regression spline (`mgcv::gam`, `family = binomial()`, REML smoothing-parameter selection), so there is no `spline.num` to hand-tune, only an upper bound `k` on basis complexity, with REML shrinking unneeded wiggliness toward a straight line on the logit scale.

Usage

```
plsi.logistic.auto(
  data,
  Y.name,
  X.name,
  Z.name,
  k = 10,
  bs = "cr",
  initial.random.num = 5,
  seed = 2026
)
```

Arguments

`data`	A data set including all needed variables
`Y.name`	Variable name for the binary outcome. Must be coded 0/1, or a 2-level factor (which will be coerced to 0/1 by level order, with a message reporting which level became 0 and which became 1).
`X.name`	Variable name vector for exposures
`Z.name`	Variable name vector for confounders
`k`	Upper bound on basis dimension for the smooth link function. Default 10.
`bs`	Smooth basis type passed to `mgcv::s()`; default `"cr"` (cubic regression spline).
`initial.random.num`	Number of random initials for the single-index direction search
`seed`	A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2026.

Details

Because the outcome is binary, the "confounder-free" single-index model cannot be built by subtracting confounder effects from `y` on the response scale the way `plsi.lr.auto()` does (probabilities aren't additive), and it cannot be built by residualizing the full model's own *fitted* linear predictor either, that would leave a value that is already an exact smooth function of the single index, so refitting a smooth to it would just retrace the same curve with near-zero (and spuriously tiny) standard errors. Instead, the confounder contribution from the full model is fixed as a known `offset()` and a second, confounder-free binomial `gam` is refit directly against the actual observed `y`, preserving real binomial sampling variability. Predictions from this model are made at `offset = 0`, consistent with how the offset was netted out of every training point.

Value

A list of model estimation and prediction results, structured analogously to `plsi.lr.auto()`'s output:

- `si.coefficient`: Single-index direction estimates and inference.
- `confounder.coefficient`: Confounder log-odds coefficients, plus odds ratios and their 95% CIs.
- `si.fun`: The estimated single-index link function, on both the logit scale (`fit`/`se`/`lwr`/`upr`) and the probability scale (`prob.fit`/`prob.lwr`/`prob.upr`), obtained by back-transforming the logit-scale CI so it stays between 0 and 1.
- `si.fun.model`: A confounder-free `gam` of the logit-scale residual on the single index alone, predict() from this needs only `single_index_estimated`, not the original confounders. See Details.
- `full.model`: The full fitted `gam` (smooth + confounders), kept for reference/diagnostics.

Author(s)

Yuyan Wang

Examples

```
## Not run:
data(nhanes.new)
data <- nhanes.new

X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

# demo binary outcome (illustrative only as nhanes.new has no native binary variable).
 Simulated from a *true* single-index combination of the exposures, run through a
 nonlinear logit link, plus a confounder effect, and drawn as genuine Bernoulli noise
 (not a hard threshold on an existing variable), a hard threshold produces near-perfect
 separation and destabilizes the fit, whereas real classification uncertainty is both
 more realistic and numerically well-behaved.
```

```
set.seed(2026)
beta_true <- c(0.30, -0.20, 0.10, 0.40, -0.30, 0.20, -0.10, 0.25, -0.15, 0.35)
beta_true <- beta_true / sqrt(sum(beta_true^2))
x_std <- scale(data[, X.name])
single_index_true <- as.vector(x_std %*% beta_true)
log_odds <- -0.2 + 0.5 * sin(single_index_true) + 0.05 * data$AGE.c
data$high.triglyceride <- stats::rbinom(nrow(data), size = 1, prob = stats::plogis(log_odds))

Y.name <- "high.triglyceride"

k <- 10
bs <- "cr"
initial.random.num <- 1
seed <- 2026

model_logistic_auto <- plsi.logistic.auto(data = data, Y.name = Y.name, X.name = X.name,
                      Z.name = Z.name, k = k, bs = bs,
                      initial.random.num = initial.random.num, seed = seed)

## End(Not run)
```

`plsi.logistic.auto.constraint` *Partial linear single index logistic regression with automatic smoothness selection and built-in identifiability constraint*

Description

Binary-outcome analog of `plsi.lr.auto.constraint()`: the same model as `plsi.logistic.auto()` (penalized regression spline link via `mgcv::gam`, `family = binomial()`, REML smoothing-parameter selection), but with the single-index direction searched over a reparameterization that satisfies the identifiability constraint by construction instead of normalizing after the fact.

Usage

```
plsi.logistic.auto.constraint(
  data,
  Y.name,
  X.name,
  Z.name,
  k = 10,
  bs = "cr",
  initial.random.num = 5,
  seed = 2026,
  maxit = 100
)
```

Arguments

data	A data set including all needed variables
Y.name	Variable name for the binary outcome. Must be coded 0/1, or a 2-level factor (which will be coerced to 0/1 by level order, with a message reporting which level became 0 and which became 1).
X.name	Variable name vector for exposures (at least 2).
Z.name	Variable name vector for confounders
k	Upper bound on basis dimension for the smooth link function. Default 10.
bs	Smooth basis type passed to mgcv::s(); default "cr" (cubic regression spline).
initial.random.num	Number of random initials for the single-index direction search
seed	A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2026.
maxit	Maximum number of Nelder-Mead iterations per starting value for the σ-space search. Default 100, matching plsi.lr.auto.constraint(). With many exposures this is often too few for the simplex to converge, which can leave the numerical information matrix indefinite and the single-index SEs inflated; check optim.convergence (and the warning issued when the selected start did not converge) and increase maxit if needed. Ignored when there are exactly two exposures (Brent search).

Details

The single-index coefficient β is constrained to $\|\beta\| = 1$ and $\beta_1 > 0$ for identifiability. The search is reparameterized in terms of $\sigma = (\sigma_1, \ldots, \sigma_{p-1})$ via $\beta(\sigma) = \{(1 - \|\sigma\|^2)^{1/2}, \sigma_1, \ldots, \sigma_{p-1}\}^T$. Both constraints hold automatically for any σ with $\|\sigma\| < 1$, so optim() runs unconstrained over σ. Standard errors for β are obtained from the Hessian in σ-space via the delta method.

The anchor exposure, the one assigned β_1, the coordinate constrained > 0, is chosen as the candidate exposure with the largest *absolute* correlation (point-biserial) with the 0/1 outcome, so an exposure strongly but negatively associated with the outcome can still be selected as the anchor. Which exposure was selected is reported in attr(si.coefficient, "note") and in anchor.note of the returned list.

The confounder-free single-index model is built exactly as in plsi.logistic.auto(): the confounder contribution from the full model is fixed as a known offset() and a second binomial gam is refit directly against the observed y. See ?plsi.logistic.auto for why the residualization approach used for continuous outcomes does not carry over to binary outcomes.

Standard errors are computed from a numerical Hessian in a local chart of the unit sphere anchored on the exposure with the largest $|\beta_j|$ (not necessarily the $\beta_1 > 0$ anchor), then mapped to β by the delta method. This keeps the finite-difference steps away from the $\|\sigma\| = 1$ boundary when the estimated β_1 is small; the point estimates are unaffected.

With exactly two exposures, σ is one-dimensional; the search then uses optim(method = "Brent") on $(-1, 1)$ instead of Nelder-Mead, which optim() flags as unreliable in one dimension.

The returned object works with all plotting functions using type = "logistic".

Value

A list of model estimation and prediction results, structured analogously to `plsi.logistic.auto()`'s output, plus:

- `anchor.note`: Which exposure was used as the anchor ($\beta_1 > 0$) and why; also attached as `attr(si.coefficient, "note")`.
- `initial.sigma.table`: The starting values actually used by the σ-space search.
- `optim.convergence`: For every starting value, the `optim()` convergence code (0 = converged, 1 = `maxit` reached), the number of log-likelihood evaluations, and the final log-likelihood.

The remaining elements are as in `plsi.logistic.auto()`:

- `si.coefficient`: Single-index direction estimates and inference.
- `confounder.coefficient`: Confounder log-odds coefficients, plus odds ratios and their 95% CIs.
- `si.fun`: The estimated single-index link function, on both the logit scale (`fit`/`se`/`lwr`/`upr`) and the probability scale (`prob.fit`/`prob.lwr`/`prob.upr`).
- `si.fun.model`: A confounder-free binomial `gam` on the single index alone (confounder contribution held as a fixed offset); predict() from this needs only `single_index_estimated` (and `confounder_offset = 0`).
- `full.model`: The full fitted `gam` (smooth + confounders).

Author(s)

Yuyan Wang

Examples

```
## Not run:
data(nhanes.new)
data <- nhanes.new

X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

# demo binary outcome (illustrative only as nhanes.new has no native binary
# variable), simulated from a true single-index combination of the exposures
set.seed(2026)
beta_true <- c(0.30, -0.20, 0.10, 0.40, -0.30, 0.20, -0.10, 0.25, -0.15, 0.35)
beta_true <- beta_true / sqrt(sum(beta_true^2))
x_std <- scale(data[, X.name])
single_index_true <- as.vector(x_std %*% beta_true)
log_odds <- -0.2 + 0.5 * sin(single_index_true) + 0.05 * data$AGE.c
data$high.triglyceride <- stats::rbinom(nrow(data), size = 1, prob = stats::plogis(log_odds))

model_logistic_con <- plsi.logistic.auto.constraint(data = data,
```

```
                        Y.name = "high.triglyceride", X.name = X.name,
                        Z.name = Z.name, k = 10, bs = "cr",
                        initial.random.num = 1, seed = 2026)
model_logistic_con$si.coefficient
si.fun.plot(model_logistic_con$si.fun, type = "logistic")

## End(Not run)
```

`plsi.lr.auto` *Partial linear single index regression with automatic smoothness selection*

Description

Same model as `plsi.lr.v2()`, but the link function is estimated with a penalized regression spline (`mgcv::gam`, REML smoothing-parameter selection) instead of a fixed-df natural spline. This removes the need to hand-tune `spline.num`: you supply an upper bound on basis complexity (`k`), and REML shrinks unneeded wiggliness toward a straight line automatically. Effective degrees of freedom actually used by the fitted link function are reported in `si.fun.edf`.

Usage

```
plsi.lr.auto(
  data,
  Y.name,
  X.name,
  Z.name,
  k = 10,
  bs = "cr",
  initial.random.num = 5,
  seed = 2026
)
```

Arguments

`data`	A data set including all needed variables
`Y.name`	Variable name for scalar outcome
`X.name`	Variable name vector for exposures
`Z.name`	Variable name vector for confounders
`k`	Upper bound on basis dimension for the smooth link function (analogous to `spline.num` in `plsi.lr.v2`, but not a fixed df, REML penalizes down from this ceiling). Default 10.
`bs`	Smooth basis type passed to `mgcv::s()`; default `"cr"` (cubic regression spline).
`initial.random.num`	Number of random initials for the single-index direction search

seed — A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2026.

Value

A list of model estimation and prediction results, structured analogously to `plsi.lr.v2()`'s output.

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to run plsi.lr.auto
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

## End(Not run)
```

---

plsi.lr.auto.constraint

Partial linear single index regression with automatic smoothness selection

---

Description

Same model as `plsi.lr.v2()`, but the link function is estimated with a penalized regression spline (`mgcv::gam`, REML smoothing-parameter selection) instead of a fixed-df natural spline. This removes the need to hand-tune `spline.num`: you supply an upper bound on basis complexity (`k`), and REML shrinks unneeded wiggliness toward a straight line automatically. Effective degrees of freedom actually used by the fitted link function are reported in `si.fun.edf`.

Usage

```
plsi.lr.auto.constraint(
  data,
  Y.name,
  X.name,
  Z.name,
  k = 10,
  bs = "cr",
  initial.random.num = 5,
  seed = 2026
)
```

Arguments

`data`	A data set including all needed variables
`Y.name`	Variable name for scalar outcome
`X.name`	Variable name vector for exposures
`Z.name`	Variable name vector for confounders
`k`	Upper bound on basis dimension for the smooth link function (analogous to `spline.num` in `plsi.lr.v2`, but not a fixed df, REML penalizes down from this ceiling). Default 10.
`bs`	Smooth basis type passed to `mgcv::s()`; default `"cr"` (cubic regression spline).
`initial.random.num`	Number of random initials for the single-index direction search
`seed`	A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2026.

Details

The single-index coefficient β is constrained to $\|\beta\| = 1$ and $\beta_1 > 0$ for identifiability. Rather than optimizing over β directly and projecting/normalizing after the fact, the search is reparameterized in terms of $\sigma = (\sigma_1, \ldots, \sigma_{p-1})$ via $\beta(\sigma) = \{(1 - \|\sigma\|^2)^{1/2}, \sigma_1, \ldots, \sigma_{p-1}\}^T$. Both constraints hold automatically for any σ with $\|\sigma\| < 1$, so `optim()` runs unconstrained over σ and $\beta_1 > 0$, $\|\beta\| = 1$ never need to be imposed after the fact. Standard errors for β are obtained from the Hessian in σ-space via the delta method.

The anchor exposure – the one assigned β_1, the coordinate constrained > 0 – is chosen as the candidate exposure with the largest *absolute* correlation with `Y.name` (not the largest signed/positive correlation), so an exposure strongly but negatively associated with the outcome can still be selected as the anchor. Which exposure was selected, and its correlation with `Y.name`, is reported in `attr(si.coefficient, "note")` and in `anchor.note` of the returned list.

Value

A list of model estimation and prediction results, structured analogously to `plsi.lr.v2()`'s output.

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to run plsi.lr.auto.constraint
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto.constraint(data = data,
                      Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

## End(Not run)
```

---

plsi.lr.v1 *Partial linear single index linear regression for scalar outcome, earliest version*

Description

Partial linear single index linear regression for scalar outcome, earliest version

Usage

```
plsi.lr.v1(
  data,
  Y.name,
  X.name,
  Z.name,
  spline.num,
  spline.degree,
  initial.random.num
)
```

Arguments

data	A data set including all needed variables
Y.name	Variable name for scalar outcome
X.name	Variable name vector for exposures
Z.name	Variable name vector for confounders
spline.num	A number representing the degree of freedom of B-spline basis for link function
spline.degree	A number representing the degree of the piece-wise polynomial of B-spline basis for link function
initial.random.num	A number representing the number of random initials used in the function

Value

A list of model estimation and prediction results

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to run the function
data(nhanes.new)
dat <- nhanes.new

# specify variable names
Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

# specify spline degree of freedom
spline.num = 5
# specify spline degree
spline.degree = 3
# specify number of random initials for estimation
initial.random.num = 1

# run the model
set.seed(2023)
model_1 <- plsi.lr.v1(data = dat, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                      spline.num, spline.degree, initial.random.num)

## End(Not run)
```

`plsi.lr.v2` *Partial linear single index linear regression for scalar outcome, second version*

Description

Partial linear single index linear regression for scalar outcome, second version

Usage

```
plsi.lr.v2(
  data,
  Y.name,
  X.name,
  Z.name,
  spline.num,
  spline.degree,
  initial.random.num,
  seed = 2026
)
```

Arguments

`data`	A data set including all needed variables
`Y.name`	Variable name for scalar outcome
`X.name`	Variable name vector for exposures
`Z.name`	Variable name vector for confounders
`spline.num`	A number representing the degree of freedom of B-spline basis for link function
`spline.degree`	A number representing the degree of the piece-wise polynomial of B-spline basis for link function
`initial.random.num`	A number representing the number of random initials used in the function
`seed`	A single integer used to seed the RNG so results (random initials and downstream optim() convergence) are reproducible. Set to NULL to skip seeding and use whatever RNG state is already active in the calling environment. Default is 2023.

Value

A list of model estimation and prediction results

Author(s)

Yuyan Wang

Examples

```
## Not run:
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

spline.num <- 5
spline.degree <- 3
initial.random.num <- 1
seed = 2026

model_lr_v2 <- plsi.lr.v2(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                      spline.num, spline.degree, initial.random.num, seed = seed)

## End(Not run)
```

`si.coef.plot` *Plot estimated single index coefficients frpm PLSIM*

Description

Plot estimated single index coefficients frpm PLSIM

Usage

```
si.coef.plot(si.coef.est)
```

Arguments

`si.coef.est`	A data set of estimated single index coefficients

Value

Single index coefficient plot

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to plot estimated single index function of continuous outcome
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)

# plot estimated single index coefficients
si.coef.plot(model_lr_auto$si.coefficient)

# check estimated single index coefficients
model_lr_auto$si.coefficient

## End(Not run)
```

`si.fun.plot` *Plot estimated single index function*

Description

Plot estimated single index function

Usage

```
si.fun.plot(
  si.ci,
  type = c("linear", "logistic", "log"),
  xlab = "Single index: u",
  ylab = NULL
)
```

Arguments

`si.ci`	A data set of estimated index and corresponding single index values, typically the `si.fun` element returned by `plsi.lr.auto()`, `plsi.logistic.auto()`, or `plsi.log.auto()` (or the analogous `plsi.lr.v2()` output for the linear case).

type	Which outcome scale to plot. "linear" (default) plots the predicted outcome g(u) using the fit/lwr/upr columns, as returned for a continuous outcome. "logistic" plots the predicted probability using the prob.fit/prob.lwr/prob.upr columns returned by plsi.logistic.auto(), with the y-axis bounded between 0 and 1. "log" plots the predicted count using the count.fit/count.lwr/count.upr columns returned by plsi.log.auto(), with the y-axis bounded below at 0.
xlab	X-axis label. Default "Single index: u".
ylab	Y-axis label. If NULL (default), a label appropriate to type is used automatically.

Value

Single index function plot

Author(s)

Yuyan Wang

Examples

```
## Not run:
# example to plot estimated single index function of continuous outcome
data(nhanes.new)
data <- nhanes.new

Y.name <- "log.triglyceride"
X.name <- c("X1_trans.b.carotene", "X2_retinol", "X3_g.tocopherol", "X4_a.tocopherol",
            "X5_PCB99", "X6_PCB156", "X7_PCB206",
            "X8_3.3.4.4.5.pncb", "X9_1.2.3.4.7.8.hxcdf", "X10_2.3.4.6.7.8.hxcdf")
Z.name <- c("AGE.c", "SEX.Female", "RACE.NH.Black",
           "RACE.MexicanAmerican", "RACE.OtherRace", "RACE.Hispanic" )

k <- 10
bs <- "cr"
initial.random.num <- 1
seed = 2026

model_lr_auto <- plsi.lr.auto(data = data, Y.name = Y.name, X.name = X.name, Z.name = Z.name,
                    k = k, bs = bs, initial.random.num = initial.random.num, seed = seed)
# plot single index function of predicted (continuous) outcome
si.fun.plot(model_lr_auto$si.fun, type = "linear")

## End(Not run)
```

Index

∗ **automatic**
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
∗ **coefficients**
si.coef.plot, 27
∗ **confounder**
confounder.trans, 2
∗ **count**
plsi.log.auto, 10
plsi.log.auto.constraint, 13
∗ **datasets**
nhanes, 8
nhanes.new, 9
∗ **effect**
e.interaction.plot, 3
e.main.plot, 5
interquartile.quartile.plot, 6
mixture.overall.plot, 7
∗ **exposures**
e.interaction.plot, 3
∗ **exposure**
e.main.plot, 5
∗ **function**
si.fun.plot, 28
∗ **index**
e.interaction.plot, 3
e.main.plot, 5
interquartile.quartile.plot, 6
mixture.overall.plot, 7
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
plsi.lr.v1, 24
plsi.lr.v2, 26
si.coef.plot, 27
si.fun.plot, 28
∗ **interaction**
e.interaction.plot, 3
interquartile.quartile.plot, 6
∗ **interquartile**
interquartile.quartile.plot, 6
∗ **linear**
e.interaction.plot, 3
e.main.plot, 5
interquartile.quartile.plot, 6
mixture.overall.plot, 7
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
plsi.lr.v1, 24
plsi.lr.v2, 26
si.coef.plot, 27
si.fun.plot, 28
∗ **log-linear**
plsi.log.auto, 10
plsi.log.auto.constraint, 13
∗ **logistic**
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
∗ **main**
e.main.plot, 5
∗ **mixture**
mixture.overall.plot, 7
∗ **models**
plsi.lr.v1, 24
plsi.lr.v2, 26
∗ **new**
plsi.lr.v2, 26

∗ **one**
e.main.plot, 5
∗ **outcome**
plsi.log.auto, 10
plsi.log.auto.constraint, 13
∗ **overall**
mixture.overall.plot, 7
∗ **partial**
e.interaction.plot, 3
e.main.plot, 5
interquartile.quartile.plot, 6
mixture.overall.plot, 7
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
plsi.lr.v1, 24
plsi.lr.v2, 26
si.coef.plot, 27
si.fun.plot, 28
∗ **quartile**
interquartile.quartile.plot, 6
∗ **regression**
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
plsi.lr.v1, 24
plsi.lr.v2, 26
∗ **single**
e.interaction.plot, 3
e.main.plot, 5
interquartile.quartile.plot, 6
mixture.overall.plot, 7
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
plsi.lr.v1, 24
plsi.lr.v2, 26
si.coef.plot, 27
si.fun.plot, 28
∗ **smoothness**
plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
∗ **two**
e.interaction.plot, 3
∗ **version**
plsi.lr.v2, 26

confounder.trans, 2

e.interaction.plot, 3
e.main.plot, 5

interquartile.quartile.plot, 6

mixture.overall.plot, 7

nhanes, 8
nhanes.new, 9

plsi.log.auto, 10
plsi.log.auto.constraint, 13
plsi.logistic.auto, 16
plsi.logistic.auto.constraint, 18
plsi.lr.auto, 21
plsi.lr.auto.constraint, 22
plsi.lr.v1, 24
plsi.lr.v2, 26

si.coef.plot, 27
si.fun.plot, 28