Package 'mpersonalized'

March 29, 2018

Type Package

Title An R Package for Personalized Medicine in Meta-analysis and Multiple Outcomes

Version 0.1.0

Author Chensheng Kuang

Maintainer Chensheng Kuang <ckuang@wisc.edu>

Description Implements a general framework to solve the problem of personalized medicine in meta-analysis and multiple outcomes. This framework allows either separate rules for each study/outcome or a single rule for all the studies/outcomes, depending on the requirement of user. A flexible choice of penalty functions to increase estimation efficiency is also provided.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports glmnet,
SGL,
caret,
ggplot2,
gridExtra,
genlasso,
Matrix

Suggests knitr,
rmarkdown

VignetteBuilder knitr

LinkingTo Rcpp

R topics documented:

`coef.mp` *Coefficients of a Fitted "mp" Object*

Description

This function provides `coef` method for "mp" class objects.

Usage

```
## S3 method for class 'mp'
coef(mp)
```

Arguments

`mp`	A fitted "mp" object returned by "mpersonalized".

Value

A list object. Each element in the list is the fitted coefficients corresponding to one penalty parameter value in `mp$penalty_parameter_sequence`.

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with SGL penalty for this meta-analysis problem
mp_mod_diff = mpersonalized(problem = "meta-analysis",
                            Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                            penalty = "SGL", single_rule = FALSE)

mp_coef = coef(mp_mod_dfff)
set.seed(NULL)
```

`coef.mp_cv` *Coefficients of a Fitted "mp_cv" Object*

Description

This function provides `coef` method for "mp_cv" class objects.

Usage

```
## S3 method for class 'mp_cv'
coef(mp_cv)
```

Arguments

`mp_cv`	A fitted "mp" object returned by "mpersonalized".

Value

The fitted coefficients corresponding to the optimal penalty parameter.

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with lasso penalty for this meta-analysis problem
mp_cvmod_diff = mpersonalized_cv(problem = "meta-analysis",
                                 Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                                 penalty = "lasso", single_rule = FALSE)

mp_coef = coef(mp_cvmod_diff)
set.seed(NULL)
```

---

mpersonalized *A General Framework to Solve Personalized Medicine in the Settings of Meta-analysis/Multiple Outcomes*

Description

This function solves the personalized medicine problem by extending the contrast classificaiton framework (Zhang, 2012). By adding proper penalty to the original classification loss, variable selection could be implemented and estimation efficiency could be improved. Computation algorithm differs based on the penalty function employed, but mainly through ADMM algorithm, glmnet package and SGL package. This function is also flexible enough to let user choose whether different classification rules or a single rule should be estimated for multiple studies/outcomes.

Usage

```
mpersonalized(problem = c("meta-analysis", "multiple outcomes"), X, Trt,
  P = NULL, Xlist, Ylist, Trtlist, Plist = replicate(length(Xlist), NULL,
  simplify = FALSE), typelist = replicate(length(Xlist), "continuous",
  simplify = FALSE), penalty = c("none", "lasso", "GL", "SGL", "fused",
  "lasso+fused", "GL+fused", "SGL+fused"), lambda1 = NULL, lambda2 = NULL,
  single_rule_lambda = NULL, num_lambda1 = ifelse(!is.null(lambda1),
  length(lambda1), 10), num_lambda2 = ifelse(!is.null(lambda2),
  length(lambda2), 10),
  num_single_rule_lambda = ifelse(!is.null(single_rule_lambda),
  length(single_rule_lambda), 50), alpha = NULL, single_rule = FALSE,
  admm_control = NULL, contrast_builder_control = NULL)
```

Arguments

problem	A character string specifiy whether user want to solve "meta-analysis" or "multiple outcomes" problem. For `problem = "meta-analysis"`, user should also supply `Xlist`, `Ylist`, `Trtlist`. For `problem = "multiple outcomes"`, user should supply `X`, `Ylist`, `Trt`.
X	Covariate matrix that should be supplied when `problem = "multiple outcomes"` with rows indicating subjects and columns indicating covariates.

`Trt` Treatment vector that should be supplied when `problem = "multiple outcomes"`, which should be coded as 0 or 1.

`P` Propensity score vector when `problem = "multiple outcomes"`. If not supplied, then study is treated as randomzied trial and the propensity score is estimated as the proportion of 1's in `Trt` for every subject.

`Xlist` A list object that should be supplied when `problem = "meta-analysis"`, with kth element denoting the covariate matrix of study k.

`Ylist` When `problem = "meta-analysis"`, `Ylist` should be a list object with kth element denoting the response vector of study k. When `problem = "multiple outcomes"`, `Ylist` should be a list object with kth element denoting the kth outcome.

`Trtlist` A list object that should be supplied when `problem = "meta-analysis"`, with kth element denoting the treatment vector of study k (coded as 0 or 1).

`Plist` A list object that should be supplied when `problem = "meta-analysis"`, with $k$the element denoting the propensity score vector of study k. If not supplied, then each study is treated as randomized trial and the corresponding propensity score is estimated as the proportion of 1's in the kth element of `Trtlist` for all subjects.

`typelist` A list object with kth element denoting the type of outcome corresponding to the kth element in `Ylist`. Each element could be "continuous" or "binary".

`penalty` For different rules, the penalty could be "none", "lasso", "GL", "SGL", "fused", "lasso+fused", "GL+fused", "SGL+fused". For single rule, the penalty could be "none" or "lasso". User should always input `penalty` and then supply corresponding penalty parameters sequence if needed. Default option is "none".

`lambda1` λ_1 in the framework of different rules. If not supplied, a default sequence will be computed.

`lambda2` λ_2 in the framework of different rules. If not supplied, a default sequence will be computed.

`single_rule_lambda` λ_{single} in the framework of single rule.

`num_lambda1` If `lambda1` is not specified by user, user could still specify the length of the `lambda1` sequence. The default length is 10.

`num_lambda2` If `lambda2` is not specified by user, user could still specify the length of the `lambda2` sequence. The default length is 10.

`num_single_rule_lambda` If `single_rule_lambda` is not specified, user could still specify the length of the `single_rule_lambda` sequence. The default length is 50.

`alpha` α in the framework of different rules. If not supplied, a default value will be used depending on `penalty`.

`single_rule` A logical value, whether the single rule framework is used. Deafult is `FALSE`.

`admm_control` A list of parameters which user can specify to control the admm algorithm. In `admm_control`, the following parameters can be supplied: `abs.tol`, absolute tolerance; `rel.tol`, relative tolerance; `maxit`, maximum number of iterations; `rho`, Lagrangian parameter.

`contrast_builder_control` A list of parameters which user can specify to control estimation of contrast function. In `contrast_builder_control`, the following parameters could be supplied: `eff_aug`, a logical value whether efficiency augmentation should be implemented; `response_model`, a character string specify what outcome model to use if `eff_aug = TRUE`, `response_model` could be "lasso" or "linear"; `contrast_builder_folds`, the number of folds used in cross validation when `response_model = "lasso"`.

Details

Assume the total number of studies/outcomes is K and we denote the contrast estimator for the kth study/outcome as $\hat{C}_k$ and the corresponding recommendation rule as g_k.

If we want different rules for each study/outcome, this function solves meta-analysis/multiple outcomes problems for personalized medicine based on the framework

$$\min_{g_1,\ldots,g_K} \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n_k} \frac{|\hat{C}_k(X_i)|}{\sum_{i=1}^{n_k}|\hat{C}_k(X_i)|}\left[1\{\hat{C}_k(X_i)>0\} - g_k(X_i)\right]^2 + h(g_1,\ldots,g_K)$$

Here the regularization function h is of the form of a sum of sparse group lasso and fused lasso penalty

$$h = (1-\alpha)\lambda_1\sqrt{q}\sum_{j=1}^{p}\|\boldsymbol{\beta_j}\|_2 + \alpha\lambda_1\sum_{j=1}^{p}\|\boldsymbol{\beta_j}\|_1 + \lambda_2\sum_{j=1}^{p}\sum_{1\le a<b\le K}|\beta_{ja}-\beta_{jb}|$$

where $\boldsymbol{\beta_j} = (\beta_{j1},\ldots,\beta_{jK})$

By setting $\lambda_1, \lambda_2, \alpha$ differently, different penalties can be obtained.

- If $\lambda_1, \lambda_2 \neq 0$ and $\alpha \neq 0$ or 1, the penalty is "SGL+fused".
- If $\lambda_1, \lambda_2 \neq 0$ and $\alpha = 0$, the penalty is "GL+fused".
- If $\lambda_1, \lambda_2 \neq 0$ and $\alpha = 1$, the penalty is "lasso+fused".
- If $\lambda_1 = 0, \lambda_2 \neq 0$, the penalty is "fused".
- If $\lambda_1 \neq 0, \lambda_2 = 0$ and $\alpha \neq 0$ or 1, the penalty is "SGL".
- If $\lambda_1 \neq 0, \lambda_2 = 0$ and $\alpha = 0$, the penalty is "GL".
- If $\lambda_1 \neq 0, \lambda_2 = 0$ and $\alpha = 1$, the penalty is "lasso".
- If $\lambda_1, \lambda_2 = 0$, there is no penalty.

On the other hand, if we would like to fit a single rule for all studies/outcomes, we let $g_1 = \ldots = g_K$ and solve the following problem instead

$$\min_{g} \frac{1}{2}\sum_{k=1}^{K}\sum_{i=1}^{n_k} \frac{|\hat{C}_k(X_i)|}{\sum_{i=1}^{n_k}|\hat{C}_k(X_i)|}\left[1\{\hat{C}_k(X_i)>0\} - g(X_i)\right]^2 + h(g_1,\ldots,g_K) + \lambda_{single}\|\beta\|_1$$

Depending on the value of λ_{single}

- If $\lambda_{single} \neq 0$, the penalty is "lasso".
- If $\lambda_{single} = 0$, there is no penalty.

Value

An S3 object of class "mp", which contains the information of the fitted model. It could be supplied to some other functions in mperosnalized package for further analysis or prediction.

`penalty_parameter_sequence`
: A matrix object with each row denoting a configuration of the penalty parameters.

`interceptlist`
: A list object with each element denoting a vector of intercepts. The kth element corresponds to the kth row in `penalty_parameter_sequence`.

`betalist`
: A list object with each element denoting a coefficient matrix. The kth element corresponds to the kth row in `penalty_parameter_sequence`.

number_covariates
: Number of candidate covariates considered.

number_studies_or_outcomes
: Number of studies if problem = "meta-analysis" or number of outcomes if problem = "multiple outcomes".

References

Zhang, B. and Tsiatis, A. A. and Davidian, M. and Zhang, M. and Laber, E.(2012) *Estimating optimal treatment regimes from a classification perspective, Stat, 1(1):103-114.*

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with SGL penalty for this meta-analysis problem
mp_mod_diff = mpersonalized(problem = "meta-analysis",
                            Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                            penalty = "SGL", single_rule = FALSE)

# fir a single rule with lasso penalty
mp_mod_single = mpersonalized(problem = "meta-analysis",
                              Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                              penalty = "lasso", single_rule = TRUE)
set.seed(NULL)
```

---

mpersonalized_cv *Cross Validation for* mpersonalized

Description

This function implments mpersonalized and use cross validatation to tune penalty parameter. The optimal penalty parameter is selected by minimizing

$$\sum_{i=1}^{n_k} \frac{|\hat{C}_k(X_i)|}{\sum_{i=1}^{n_k}|\hat{C}_k(X_i)|}\left[1\{\hat{C}_k(X_i)>0\}-g_k(X_i)\right]^2$$

in the leave-out fold, where $\hat{C}_k(X_i)$ in the leave-out fold is independently estimated from the training set.

Usage

```
mpersonalized_cv(problem = c("meta-analysis", "multiple outcomes"), X, Trt,
  P = NULL, Xlist, Ylist, Trtlist, Plist = replicate(length(Xlist), NULL,
  simplify = FALSE), typelist = replicate(length(Xlist), "continuous",
  simplify = FALSE), penalty = c("lasso", "GL", "SGL", "fused", "lasso+fused",
  "GL+fused", "SGL+fused"), lambda1 = NULL, lambda2 = NULL,
  single_rule_lambda = NULL, num_lambda1 = ifelse(!is.null(lambda1),
  length(lambda1), 10), num_lambda2 = ifelse(!is.null(lambda2),
  length(lambda2), 10),
```

```
num_single_rule_lambda = ifelse(!is.null(single_rule_lambda),
length(single_rule_lambda), 50), alpha = NULL, single_rule = FALSE,
cv_folds = 5, admm_control = NULL, contrast_builder_control = NULL)
```

Arguments

problem	A character string specifiy whether user want to solve "meta-analysis" or "multiple outcomes" problem. For `problem = "meta-analysis"`, user should also supply `Xlist`, `Ylist`, `Trtlist`. For `problem = "multiple outcomes"`, user should supply `X`, `Ylist`, `Trt`.
X	Covariate matrix that should be supplied when `problem = "multiple outcomes"` with rows indicating subjects and columns indicating covariates.
Trt	Treatment vector that should be supplied when `problem = "multiple outcomes"`, which should be coded as 0 or 1.
P	Propensity score vector when `problem = "multiple outcomes"`. If not supplied, then study is treated as randomzied trial and the propensity score is estimated as the proportion of 1's in `Trt` for every subject.
Xlist	A list object that should be supplied when `problem = "meta-analysis"`, with kth element denoting the covariate matrix of study k.
Ylist	When `problem = "meta-analysis"`, `Ylist` should be a list object with kth element denoting the response vector of study k. When `problem = "multiple outcomes"`, `Ylist` should be a list object with kth element denoting the kth outcome.
Trtlist	A list object that should be supplied when `problem = "meta-analysis"`, with kth element denoting the treatment vector of study k (coded as 0 or 1).
Plist	A list object that should be supplied when `problem = "meta-analysis"`, with $k$the element denoting the propensity score vector of study k. If not supplied, then each study is treated as randomized trial and the corresponding propensity score is estimated as the proportion of 1's in the kth element of `Trtlist` for all subjects.
typelist	A list object with kth element denoting the type of outcome corresponding to the kth element in `Ylist`. Each element could be "continuous" or "binary".
penalty	For different rules, the penalty could be "lasso", "GL", "SGL", "fused", "lasso+fused", "GL+fused", "SGL+fused". For single rule, the penalty could only be "lasso". For `penalty = "none"`, use function `mpersonalized` instead. User should always input `penalty` and then supply correponding penalty parameters sequence if needed.
lambda1	λ_1 in the framework of different rules. If not supplied, a default sequence will be computed.
lambda2	λ_2 in the framework of different rules. If not supplied, a default sequence will be computed.
single_rule_lambda	λ_{single} in the framework of single rule.
num_lambda1	If `lambda1` is not specified by user, user could still specify the length of the `lambda1` sequence. The default length is 10.
num_lambda2	If `lambda2` is not specified by user, user could still specify the length of the `lambda2` sequence. The default length is 10.
num_single_rule_lambda	If `single_rule_lambda` is not specified, user could still specify the length of the `single_rule_lambda` sequence. The default length is 50.

`alpha` α in the framework of different rules. If not supplied, a default value will be used depending on `penalty`.

`single_rule` A logical value, whether the single treatment framework is used. Deafult is `FALSE`.

`cv_folds` Number of folds needed for cross-validation. Default is 5

`admm_control` A list of parameters which user can specify to control the admm algorithm. In `admm_control`, the following parameters can be supplied: `abs.tol`, absolute tolerance; `rel.tol`, relative tolerance; `maxit`, maximum number of iterations; `rho`, Lagrangian parameter.

`contrast_builder_control`
A list of parameters which user can specify to control estimation of contrast function. In `contrast_builder_control`, the following parameters could be supplied: `eff_aug`, a logical value whether efficiency augmentation should be implemented; `response_model`, a character string specify what outcome model to use if `eff_aug = TRUE`, `response_model` could be "lasso" or "linear"; `contrast_builder_folds`, the number of folds used in cross validation when `response_model = "lasso"`.

Value

An S3 object of class "mp_cv", which contains the information of the model with the optimal lambda. It can be supplied to some other functions in mperosnalized package for further analysis or prediction.

`penalty_parameter_sequence`
A matrix object with each row denoting a configuration of the penalty parameters.

`opt_penalty_parameter`
Optimal penalty parameter chosen by minimizing the cross validation error.

`interccept` The vector of intercepts corresponding to the optimal penalty parameter.

`beta` The coefficient matrix corresponding to the optimal penalty parameter.

`number_covariates`
Number of candidate covariates considered.

`number_studies_or_outcomes`
Number of studies if `problem = "meta-analysis"` or number of outcomes if `problem = "multiple outcomes"`.

Examples

```
set.seed(123)
sim_dat = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with group lasso penalty
mp_cvmod_diff = mpersonalized_cv(problem = "meta-analysis",
                                 Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                                 penalty = "GL", single_rule = FALSE)

mp_cvmod_diff$intercept
mp_cvmod_diff$beta

# fit a single rule with lasso penalty
mp_cvmod_single = mpersonalized_cv(problem = "meta-analysis",
```

```
                                  Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                                  penalty = "lasso", single_rule = TRUE)

mp_cvmod_single$intercept
mp_cvmod_single$beta
set.seed(NULL)
```

---

plot.mp *Interaction Plot for an "mp" Class Object.*

Description

This function plots interaction between received treatment and recommended treatment, which provides an estimate of treatment effect of the identified subgroup.

Usage

```
## S3 method for class 'mp'
plot(mp, penalty_index)
```

Arguments

mp	A fitted "mp" class object returned by mpersonalzied function
penalty_index	The index of penalty parameter configuration in mp$penalty_parameter_sequence. When mp$penalty = "none", penalty_index is automatically set to be 1.

Details

In the interaction plot, each point is the group mean given a received treatment and a recommended treatment. Although usually overestimating treatment effect in training set, interaction plots provides a sanity check for treatment recommendation rules. Given a specific index of penalty parameter, the function plots corresponding interaction plots.

Value

A list object with each element as the interaction plots for a penalty parameter configuration.

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with SGL penalty for this meta-analysis problem
mp_mod_diff = mpersonalized(problem = "meta-analysis",
                            Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                            penalty = "lasso", single_rule = FALSE)

# interaction plot of the 5th penalty parameter
plots = plot(mp = mp_mod_diff, penalty_index = 5)
set.seed(NULL)
```

plot.mp_cv *Interaction Plot for an "mp_cv" Class Object.*

Description

This function plots interaction between received treatment and recommended treatment, given the optimal penalty parameter.

Usage

```
## S3 method for class 'mp_cv'
plot(mp_cv)
```

Arguments

mp_cv	A fitted 'mp_cv' class object returned by mpersonalzied_cv function

Value

A list object representing the interaction plots for the optimal penalty parameter configuration. Specifically, kth element is the interaction plot for the kth study/outcome.

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with lasso penalty for this meta-analysis problem
mp_cvmod_diff = mpersonalized_cv(problem = "meta-analysis",
                                 Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                                 penalty = "lasso", single_rule = FALSE)

plots = plot(mp_cv = mp_cvmod_diff)
set.seed(NULL)
```

predict.mp *Prediction for a Fitted "mp" Object*

Description

This function predicts optimal treatment of new subjects for a mpersonalized model. If different rules are used in the fitting procedure, an overall treatment recommendation based on all stuides/outcomes could be provided together with optimal treatments for each study/outcome.

Usage

```
## S3 method for class 'mp'
predict(mp, newx = NULL, weight = NULL, overall_rec = TRUE)
```

Arguments

mp	A fitted "mp" object returned by "mpersonalized"
newx	Covariate matrix of new patients. If not supplied, by default the prediction is for the original dataset in the "mp" object. Notice: when problem = "meta-analysis" and the prediction is for the original dataset, subjects in each study are only predicted using the treatment recommendation rule of the study they belong to.
weight	A weight vector for the overall recommendation, only needed when overall_rec = TRUE. By default, equal weights are assigned to each study/outcome.
overall_rec	A logical value. If overall_rec = TRUE, an overall recommendation will be provided as an weighted average of the optimal treatment from each individual study/outcome. Only useful when newx is provided.

Details

This function predicts for each penalty parameter in the penalty_parameter_sequence of the "mp" object. The overall recommended treatment is given as an weighted average of the recommended treatments from each study/outcome, and the weight can be specified by user.

Value

A list object of two elements. .

opt_treatment	A list object with each element denoting the prediction based on a penalty parameter configuration in mp$penalty_parameter_sequence. Specifically, if newx is provided, each element is a recommendation matrix with each row denoting a subject and each column denoting a study/outcome; otherwise, each element is a list of vectors with each vector representing the optimal treatment for each study/outcome. If overall_rec = TRUE, the weighted overall recommended treatment will be further provided as well. If the overall recommened treatment is equal to 0.5, it means the weighted sum is equal for 0 and 1.
benefit_score	A list object of benefit scores computed from $g_1, \ldots, g_K$. Similar structure as opt_treatment.

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with SGL penalty for this meta-analysis problem
mp_mod_diff = mpersonalized(problem = "meta-analysis",
                            Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                            penalty = "SGL", single_rule = FALSE)

newx = matrix(rnorm(100 * mp_mod_diff$number_covariates), nrow = 100)

# predict on newx
pred_new = predict(mp = mp_mod_diff, newx = newx, overall_rec = TRUE)

# predict on old dataset
pred_old = predict(mp = mp_mod_diff)
set.seed(NULL)
```

predict.mp_cv *Prediction for a Fitted "mp_cv" Object*

Description

This function predicts optimal treatment of new subjects for a cross-validated mpersonalized model.

Usage

```
## S3 method for class 'mp_cv'
predict(mp_cv, newx = NULL, weight = NULL,
  overall_rec = TRUE)
```

Arguments

mp_cv	A fitted "mp_cv" object returned by "mpersonalized_cv" function
newx	Covariate matrix of new patients. If not supplied, by default the prediction is for the original dataset in the "mp_cv" object. Prediction results will differ based on whether newx is provided or not. Similar to predict.mp.
weight	A weight vector for the overall recommendation, only needed when overall_rec = TRUE. By default, equal weights are assigned to each study/outcome.
overall_rec	A logical value. If overall_rec = TRUE, an overall recommendation will be provided as an weighted average of the optimal treatment from each individual study/outcome. Only useful when newx is provided.

Value

A list object with two elements. Similar to the returned value of predict.mp, but now it only predicts for the optimal parameter penalty.

opt_treatment	If newx is provided, a recommendation matrix with each row denoting a subject and each column denoting a study/outcome; otherwise, each element is a list of vectors with each vector representing the optimal treatment for each study/outcome. If overall_rec = TRUE, the weighted overall recommended treatment will be further provided as well. If the overall recommened treatment is equal to 0.5, it means the weighted sum is equal for 0 and 1.
benefit_score	Benefit scores computed from $g_1, \ldots, g_K$. Similar to structure of opt_treatment.

Examples

```
set.seed(123)
sim_dat = simulated_dataset(n = 200, problem = "meta-analysis")
Xlist = sim_dat$Xlist; Ylist = sim_dat$Ylist; Trtlist = sim_dat$Trtlist

# fit different rules with group lasso penalty
mp_cvmod_diff = mpersonalized_cv(problem = "meta-analysis",
                                 Xlist = Xlist, Ylist = Ylist, Trtlist = Trtlist,
                                 penalty = "GL", single_rule = FALSE)

newx = matrix(rnorm(100 * mp_cvmod_diff$number_covariates), nrow = 100)
```

```
# predict on newx
pred_new = predict(mp_cv = mp_cvmod_diff, newx = newx, overall_rec = TRUE)

# predict on old dataset
pred_old = predict(mp_cv = mp_cvmod_diff)
set.seed(NULL)
```

`simulated_dataset` *Simulated Dataset Generator*

Description

Generate a simulated dataset, which could be used to demonstrate the features of the mpersonalized package.

Usage

```
simulated_dataset(n, problem = c("meta-analysis", "multiple outcomes"))
```

Arguments

`n`	Sample size for each study/outcome.
`problem`	A character string specified what problem the simulated dataset is generated for. `problem` can be set to "meta-analysis" or "multiple outcomes".

Details

In the simulated dataset, outcomes are generated from the model

$$Y = \delta_0 + \boldsymbol{X}\boldsymbol{\delta} + A(\theta_0 + \boldsymbol{X}\boldsymbol{\theta}) + \epsilon,$$

where $\boldsymbol{X}$ is the baseline covariates and A is the treatment indicator coded as 0,1. For different outcomes or studies, values of δ_0, $\boldsymbol{\delta}$, θ_0 and $\boldsymbol{\theta}$ are also different so as to represent the heterogeneity in real problems.

The number of different studies/outcomes is set to be 6 and total number of candidate covariates is 50. Treatment indicator A is generated with equal probability of 0 or 1.

This function randomly generates the coefficients for each study/outcome and then generates the baseline covariates and error term for each subject. Depending on the value of `problem`, generation of baseline covariates are slightly different. For `problem = "meta-analysis"`, baseline covariates are generated independently for each study; for `problem = "multiple outcomes"`, baseline covariates are the same across different outcomes.

Value

A list object of the ingredients from the simulated dataset. The elements of this list depends on value of `problem`.

For `problem = "meta-analysis"`,

`Xlist`	a list object with kth element denoting the baseline covariate matrix of kth study
`Ylist`	a list object with kth element denoting the response vector of kth study

`Trtlist`	a list object with kth element denoting the treatment vector of kth study and coded as 0 or 1
`B`	the coefficient matrix containing δ_0, $\boldsymbol{\delta}$, θ_0 and $\boldsymbol{\theta}$

For `problem = "multiple outcomes"`,

`X`	a matrix object denoting the baseline covariate matrix
`Ylist`	a list object with kth element denoting the response vector of kth outcome
`Trt`	a vector denoting the treatment and coded as 0 or 1)
`B`	the coefficient matrix containing δ_0, $\boldsymbol{\delta}$, θ_0 and $\boldsymbol{\theta}$

Examples

```
set.seed(123)
sim_dat  = simulated_dataset(n = 200, problem = "meta-analysis")
str(sim_dat$Xlist)
str(sim_dat$Ylist)
str(sim_dat$Trtlist)
set.seed(NULL)
```

Index