

Package ‘StratifiedMedicine’

September 4, 2019

Type Package

Title Stratified Medicine

Version 0.1.3

Author Thomas Jemielita [aut, cre]

Maintainer Thomas Jemielita <thomasjemielita@gmail.com>

Description A toolkit for stratified medicine, subgroup identification, and precision medicine. Current tools include (1) filtering models (reduce covariate space), (2) patient-level estimate models (counterfactual patient-level quantities, for example the individual treatment effect), (3) subgroup identification models (find subsets of patients with similar treatment effects), and (4) parameter estimation and inference (for the overall population and discovered subgroups). These tools can directly feed into stratified medicine algorithms including PRISM (patient response identifiers for stratified medicine; Jemielita and Mehrotra 2019 (in progress)). PRISM is a flexible and general framework which accepts user-created models/functions. This package is in beta and will be continually updated.

License GPL-3

Encoding UTF-8

LazyData true4

Depends R (>= 3.1),

Imports dplyr, partykit, ranger, survival, glmnet, ggplot2, mvtnorm

RoxygenNote 6.1.1

URL <https://github.com/thomasjemielita/StratifiedMedicine>

Suggests knitr, rmarkdown, MASS, zoo, BART, grf, survRM2, TH.data, coin, rpart

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-09-04 14:20:05 UTC

R topics documented:

filter_glmnet	3
filter_ranger	4
generate_subgrp_data	5
param_combine	6
param_cox	6
param_dr	8
param_lm	9
param_ple	10
param_rmst	11
ple_bart	12
ple_causal_forest	14
ple_glmnet	15
ple_ranger	16
ple_train	17
plot.PRISM	18
predict.ple_bart	19
predict.ple_causal_forest	20
predict.ple_glmnet	21
predict.ple_ranger	22
predict.ple_train	23
predict.PRISM	24
predict.submod_ctree	25
predict.submod_lmtree	26
predict.submod_otr	27
predict.submod_rpart	28
predict.submod_train	29
predict.submod_weibull	30
PRISM	31
PRISM_resamp	34
PRISM_train	36
submod_ctree	37
submod_lmtree	38
submod_otr	39
submod_rpart	40
submod_train	42
submod_weibull	43
summary.PRISM	44
Index	45

filter_glmnet	<i>Filter: Elastic Net (glmnet)</i>
---------------	-------------------------------------

Description

Filter variables through elastic net (Zou and Hastie 2005). Default is to regress $Y \sim X$ (search for prognostic variables). Variables with estimated coefficients of zero (depends on lambda choice; default is lambda.min) are filtered. Usable for continuous, binary, and survival outcomes.

Usage

```
filter_glmnet(Y, A, X, lambda = "lambda.min", family = "gaussian",
             interaction = FALSE, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
lambda	Lambda for elastic net model (default="lambda.min"). Other options include "lambda.1se" and fixed values
family	Outcome type ("gaussian", "binomial", "survival"), default is "gaussian"
interaction	Regress $Y \sim X + A + A * X$ (interaction between covariates and treatment)? Default is FALSE. If TRUE, variables with zero coefficients (both X and X*A terms) are filtered.
...	Any additional parameters, not currently passed through.

Value

Filter model and variables that remain after filtering.

- mod - Filtering model
- filter.vars - Variables that remain after filtering (could be all)

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Default: Regress Y~X (search for prognostic factors) #
```

```

mod1 = filter_glmnet(Y, A, X)
mod2 = filter_glmnet(Y, A, X, lambda = "lambda.min") # same as default
mod3 = filter_glmnet(Y, A, X, lambda = "lambda.1se")
mod1$filter.vars
mod2$filter.vars
mod3$filter.vars

# Interaction=TRUE; Regress Y~X+A+X*A (search for prognostic and/or predictive) #
mod4 = filter_glmnet(Y, A, X, interaction=TRUE)
mod4$filter.vars

```

filter_ranger

Filter: Random Forest (ranger) Variable Importance

Description

Filtering through Random Forest Variable Importance with p-values. P-values are obtained through subsampling based T-statistics, as described in Ishwaran and Lu 2017. Default is to remove variables with p-values ≥ 0.10 . Used for continuous, binary, or survival outcomes.

Usage

```

filter_ranger(Y, A, X, b = 0.66, K = 200, DF2 = FALSE, FDR = FALSE,
  pval.thres = 0.1, family = "gaussian", ...)

```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
b	Subsample size (n^b)
K	Number of samples (default=200)
DF2	2-DF test statistic (default=FALSE)
FDR	FDR correction for p-values (default=FALSE)
pval.thres	p-value threshold for filtering (default=0.10)
family	Outcome type ("gaussian", "binomial", "survival"), default is "gaussian"
...	Any additional parameters, not currently passed through.

Value

Filter model and variables that remain after filtering.

- mod - Filtering model
- filter.vars - Variables that remain after filtering (could be all)

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

mod1 = filter_ranger(Y, A, X, K=200) # Same as default #
mod1$filter.vars
mod1$mod # summary of variable importance outputs
```

generate_subgrp_data *Generate Subgroup Data-sets*

Description

Simulation/real data-sets; useful for testing new models and PRISM configurations.

Usage

```
generate_subgrp_data(n = 800, seed = 513413, family, ...)
```

Arguments

n	sample size (default=800)
seed	seed number (default=513413)
family	Outcome type ("gaussian", "binomial", "survival")
...	Any additional parameters, not currently passed through.

Value

Simulation data set (Y=outcome, A=treatment, X=covariates)

param_combine	<i>Overall Population Estimate: Aggregating Subgroup-Specific Parameter Estimates</i>
---------------	---

Description

Function that combines subgroup-specific estimates to obtain an overall population estimate. Options including sample size weighting and adaptive weighting (default; as described in Marceau-West and Mehrotra 2019 in progress).

Usage

```
param_combine(param.dat, combine = "adaptive", alpha_ovrl = 0.05, ...)
```

Arguments

param.dat	Parameter data-set with subgroup-specific point estimates, SEs, and sample sizes.
combine	Method to combine subgroup-specific estimates. Default is "adaptive". combine="SS" uses sample size weighting.
alpha_ovrl	Two-sided alpha level for overall population. Default=0.05
...	Any additional parameters, not currently passed through.

Value

Data-frame with overall population point estimate, SE, and CI

See Also

[param_cox](#), [param_lm](#), [param_rmst](#)

param_cox	<i>Parameter Estimation: Cox Regression</i>
-----------	---

Description

For each identified subgroup, fit separate cox regression models. Point-estimates and variability metrics in the overall population are obtained by aggregating subgroup specific results (adaptive weighting or sample size weighting).

Usage

```
param_cox(Y, A, X, mu_hat, Subgrps, alpha_ovrl, alpha_s,
  combine = "adaptive", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
mu_hat	Patient-level estimates (See PLE_models)
Subgrps	Identified subgroups (can be the overall population)
alpha_ovr1	Two-sided alpha level for overall population
alpha_s	Two-sided alpha level at subgroup
combine	For overall population, method of combining subgroup-specific results. Default is "adaptive", "SS" corresponds to sample size weighting.
...	Any additional parameters, not currently passed through.

Value

Data-set with parameter estimates (hazard ratio) and corresponding variability metrics, for overall and subgroups. Subgrps=0 corresponds to the overall population by default.

- param.dat - Parameter estimates and variability metrics (est=HR, SE=SE(logHR), LCL/UCL = lower/upper confidence limit on HR scale, pval = p-value).

See Also

[param_combine](#)

Examples

```
library(StratifiedMedicine)
# Survival Data #
require(TH.data); require(coin)

# MOB-Weibull Subgroup Model ##
res_weibull = submod_train(Y, A, X, Xtest=X, family="survival",
                          submod="submod_weibull")
plot(res_weibull$mod)

## Parameter-Estimation ##
params = param_cox(Y, A, X, Subgrps = res_weibull$Subgrps.train, alpha_ovr1=0.05,
                  alpha_s=0.05)
params
```

param_dr *Parameter Estimation: Double-robust estimator*

Description

For each identified subgroup and in the overall population, use the double robust estimator (Funk et al 2011). Usable for continuous and binary outcomes, specifically for the estimand $E(Y|X,A=1)-E(Y|X,A=0)$.

Usage

```
param_dr(Y, A, X, mu_hat, Subgrps, alpha_ovr1, alpha_s, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
mu_hat	Patient-level estimates (See PLE_models)
Subgrps	Identified subgroups (can be the overall population)
alpha_ovr1	Two-sided alpha level for overall population
alpha_s	Two-sided alpha level at subgroup
...	Any additional parameters, not currently passed through.

Value

Data-set with parameter estimates (average treatment effect) and corresponding variability metrics, for overall and subgroups. Subgrps=0 corresponds to the overall population by default.

- param.dat - Parameter estimates and variability metrics (est, SE, LCL/UCL = lower/upper confidence limits, pval = p-value).

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

## Estimate PLEs (ranger) ##
res_ranger = ple_train(Y, A, X, Xtest=X, ple="ple_ranger")

## Identify Subgroups: MOB (lmtree) ##
res_lmtree = submod_train(Y, A, X, Xtest=X, submod="submod_lmtree")
```



```
## Parameter-estimation ##
params = param_dr(Y, A, X, mu_hat = res_ranger$mu_train,
                 Subgrps = res_lmtree$Subgrps.train, alpha_ovrl=0.05,
                 alpha_s=0.05)

params
```

 param_lm

Parameter Estimation: Linear Regression

Description

For each identified subgroup, fit separate linear regression models. Point-estimates and variability metrics in the overall population are obtained by aggregating subgroup specific results (adaptive weighting or sample size weighting).

Usage

```
param_lm(Y, A, X, mu_hat, Subgrps, alpha_ovrl, alpha_s,
        combine = "adaptive", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
mu_hat	Patient-level estimates (See PLE_models)
Subgrps	Identified subgroups (can be the overall population)
alpha_ovrl	Two-sided alpha level for overall population
alpha_s	Two-sided alpha level at subgroup
combine	For overall population, method of combining subgroup-specific results. Default is "adaptive", "SS" corresponds to sample size weighting.
...	Any additional parameters, not currently passed through.

Value

Data-set with parameter estimates (average treatment effect) and corresponding variability metrics, for overall and subgroups. Subgrps=0 corresponds to the overall population by default.

- param.dat - Parameter estimates and variability metrics (est, SE, LCL/UCL = lower/upper confidence limits, pval = p-value).

See Also

[param_combine](#)

Examples

```

library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

## Identify Subgroups: MOB (lmtree) ##
res_lmtree = submod_train(Y, A, X, Xtest=X, submod="submod_lmtree")

## Parameter-estimation ##
params = param_lm(Y, A, X, Subgrps = res_lmtree$Subgrps.train, alpha_ovr1=0.05,
                  alpha_s=0.05)

params

```

param_ple

Parameter Estimation: Patient-Level Estimates

Description

Parameter estimation and inference through patient-level estimates. Usable for continuous and binary outcomes (possibly survival, needs further evaluation).

Usage

```
param_ple(Y, A, X, mu_hat, Subgrps, alpha_ovr1, alpha_s, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
mu_hat	Patient-level estimates (See PLE_models)
Subgrps	Identified subgroups (can be the overall population)
alpha_ovr1	Two-sided alpha level for overall population
alpha_s	Two-sided alpha level at subgroup
...	Any additional parameters, not currently passed through.

Value

Data-set with parameter estimates and corresponding variability metrics, for overall and subgroups. Subgrps=0 corresponds to the overall population by default.

- param.dat - Parameter estimates and variability metrics (est, SE, LCL/UCL = lower/upper confidence limits, pval = p-value).

Examples

```

library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A
train = data.frame(Y, A, X)

## Estimate PLEs (ranger) ##
res_ranger = ple_train(Y, A, X, Xtest=X, ple = "ple_ranger")

## Identify Subgroups: MOB (lmtree) ##
res_lmtree = submod_train(Y, A, X, Xtest=X, submod="submod_lmtree")

## Parameter-estimation ##
params = param_ple(Y, A, X, mu_hat = res_ranger$mu_train,
                  Subgrps = res_lmtree$Subgrps.train, alpha_ovr1=0.05,
                  alpha_s=0.05)
params

```

param_rmst

Parameter Estimation: Restricted Mean Survival Time (RMST)

Description

For each identified subgroup, estimate the restricted mean survival time (RMST), based on the method described in the R package "survRM2". Point-estimates and variability metrics in the overall population are obtained by aggregating subgroup specific results (adaptive weighting or sample size weighting).

Usage

```

param_rmst(Y, A, X, mu_hat, Subgrps, alpha_ovr1, alpha_s,
           combine = "adaptive", ...)

```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
mu_hat	Patient-level estimates (See PLE_models)
Subgrps	Identified subgroups (can be the overall population)
alpha_ovr1	Two-sided alpha level for overall population

alpha_s	Two-sided alpha level at subgroup
combine	For overall population, method of combining subgroup-specific results. Default is "adaptive", "SS" corresponds to sample size weighting.
...	Any additional parameters, not currently passed through.

Value

Data-set with parameter estimates (RMST) and corresponding variability metrics, for overall and subgroups.

- param.dat - Parameter estimates and variability metrics

See Also

[param_combine](#)

Examples

```
library(StratifiedMedicine)
# Survival Data #
require(TH.data); require(coin)
data("GBSG2", package = "TH.data")
surv.dat = GBSG2
# Design Matrices ###
Y = with(surv.dat, Surv(time, cens))
X = surv.dat[,!(colnames(surv.dat) %in% c("time", "cens")) ]
A = rbinom( n = dim(X)[1], size=1, prob=0.5 ) ## simulate null treatment

# MOB-Weibull Subgroup Model ##
res_weibull = submod_train(Y, A, X, Xtest=X, family="survival",
                          submod = "submod_weibull")
plot(res_weibull$mod)

# Parameter-Estimation ##
require(survRM2)
params = param_rmst(Y, A, X, Subgrps = res_weibull$Subgrps.train, alpha_ovrl=0.05,
                   alpha_s=0.05)
params
```

ple_bart

Patient-level Estimates: BART

Description

Uses the BART algorithm (Chipman et al 2010; BART R package) to obtain patient-level estimates. Used for continuous or binary outcomes. Covariate by treatment interactions are automatically included in BART model (as in Hahn et al 2017).

Usage

```
ple_bart(Y, A, X, Xtest, family = "gaussian", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
family	Outcome type ("gaussian", "binomial"), default is "gaussian"
...	Any additional parameters, not currently passed through.

Value

Trained BART model(s) and patient-level estimates ($E(Y|X,1)$, $E(Y|X,0)$, $E(Y|X,1)-E(Y|X,0)$) for train/test sets.

- mods - trained model(s)
- mu_train - Patient-level estimates (training set)
- mu_test - Patient-level estimates (test set)

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A
train = data.frame(Y, A, X)

# BART #

require(BART)
mod1 = ple_bart(Y, A, X, Xtest=X)
summary(mod1$mu_train)
summary(predict(mod1, newdata=X))
```

ple_causal_forest *Patient-level Estimates: Causal Forest*

Description

Uses the causal forest algorithm (Athey, Tibshirani, and Wager 2019; grf R package) to obtain patient-level estimates. Used for continuous or binary outcomes.

Usage

```
ple_causal_forest(Y, A, X, Xtest, tune = FALSE, num.trees = 500,
  family = "gaussian", mod.A = "mean", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
tune	If TRUE, use grf automatic hyper-parameter tuning. If FALSE (default), no tuning.
num.trees	Number of trees (default=500)
family	Outcome type ("gaussian", "binomial"), default is "gaussian"
mod.A	Model for estimating P(A X). Default is "mean" calculates the sample mean. If mod.A="RF", estimate P(A X) using regression_forest (applicable for non-RCTs).
...	Any additional parameters, not currently passed through.

Value

Trained causal_forest and regression_forest models.

- mods - trained model(s)

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

require(grf)
```

```
mod1 = ple_causal_forest(Y, A, X, Xtest=X)
summary(mod1$mu_train)
```

ple_glmnet

Patient-level Estimates: Elastic Net (glmnet)

Description

Uses the elastic net (glmnet R package) to obtain patient-level estimates. Usable for continuous, binary, or survival outcomes.

Usage

```
ple_glmnet(Y, A, X, Xtest, lambda = "lambda.min", family, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
lambda	Lambda for elastic-net (default = "lambda.min"). Other options include "lambda.1se" or fixed values
family	Outcome type ("gaussian", "binomial", "survival"), default is "gaussian"
...	Any additional parameters, not currently passed through.

Value

Trained glmnet model(s).

- mods - trained model(s)
- lambda - Lambda used for elastic-net (passes to prediction function)
- X - Covariate Space (in model matrix form)

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
```

```
A = dat_ctns$A

mod1 = ple_glmnet(Y, A, X, Xtest=X, family="gaussian")
summary(mod1$mu_train$PLE)
```

ple_ranger

Patient-level Estimates: Ranger

Description

Uses treatment-specific (or with explicit X*A interactions) random forest models (ranger) to obtain patient-level estimates. Used for continuous, binary, or survival outcomes.

Usage

```
ple_ranger(Y, A, X, Xtest, byTrt = TRUE, min.node.pct = 0.1,
  family = "gaussian", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
byTrt	If TRUE, fit treatment-specific ranger models. If FALSE, fit a single ranger model with covariate space (X, A, X*A).
min.node.pct	Minimum sample size in forest nodes (n*min.node.pct)
family	Outcome type ("gaussian", "binomial"), default is "gaussian"
...	Any additional parameters, not currently passed through.

Value

Trained random forest (ranger) model(s).

- mods - trained model(s)
- A - treatment variable (training set)
- X - covariate space (training set)

See Also

[PRISM](#), [ranger](#)

Examples

```

library(StratifiedMedicine)
## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Counter-factual Random Forest (treatment-specific ranger models) #
mod1 = ple_ranger(Y, A, X, Xtest=X)
summary( predict(mod1, newdata=data.frame(A,X) ) ) # oob predictions for training
summary( predict(mod1, newdata=data.frame(X) ) ) # new-predictions, no oob here

```

ple_train

*Patient-level Estimates: Train Model***Description**

Wrapper function to train a patient-level estimate (ple) model. Used directly in PRISM and can be used to directly fit a ple model by name.

Usage

```
ple_train(Y, A, X, Xtest, family = "gaussian", ple, hyper = NULL, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
family	Outcome type ("gaussian", "binomial", "survival"). Default is "gaussian".
ple	PLE (Patient-Level Estimate) function. Maps the observed data to PLEs. (Y,A,X) ==> PLE(X).
hyper	Hyper-parameters for the ple model (must be list). Default is NULL.
...	Any additional parameters, not currently passed through.

Value

Trained ple models and patient-level estimates for train/test sets. For family="gaussian" or "binomial", output estimates of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$. For survival, output estimates of $(HR(X,A=1), HR(X,A=0), HR(X, A=1)-HR(X, A=0))$.

- mods - trained model(s)
- mu_train - Patient-level estimates (training set)
- mu_test - Patient-level estimates (test set)

See Also

[PRISM](#)

Examples

```
library(StratifiedMedicine)
## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Fit ple_ranger directly (treatment-specific ranger models) #
mod1 = ple_ranger(Y, A, X, Xtest=X)
summary(mod1$mu_train)

# Fit through ple_train wrapper #
mod2 = ple_train(Y=Y, A=A, X=X, Xtest=X, ple="ple_ranger" )
summary(mod2$mu_train)
```

plot.PRISM

plot.PRISM

Description

Plots PRISM results, either forest plot (estimate with CIs) or resampling distribution.

Usage

```
## S3 method for class 'PRISM'
plot(x, type = "forest", estimand = NULL,
     grid.data = NULL, grid.thres = ">0", ...)
```

Arguments

x	PRISM object
type	Type of plot (default="forest", forest plot for overall and subgroups). Other options include "PLE:waterfall" (waterfall plot of PLEs), "PLE:density" (density plot of PLEs), "resample" (resampling distribution of parameter estimates for overall and subgroups), and "heatmap" (heatmap of ple estimates/probabilities).
estimand	For "resample" plot only, must be specify which estimand to visualize. Default=NULL.
grid.data	Input grid of values for 2-3 covariates (if 3, last variable cannot be continuous). This is required for type="heatmap". Default=NULL.
grid.thres	Threshold for PLE, ex: I(PLE>thres). Used to estimate P(PLE>thres) for type="heatmap". Default is ">0". Direction can be reversed and can include equality sign (ex: "<=").
...	Additional arguments (currently ignored).

Value

Plot (ggplot2) object

See Also

[PRISM](#)

predict.ple_bart

Predict Patient-level Estimates: BART

Description

Get estimates of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$ using trained BART model(s).

Usage

```
## S3 method for class 'ple_bart'
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained BART model(s).
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Data-frame with predictions of $(E(Y|X,1), E(Y|X,0), E(Y|X,1)-E(Y|X,0))$

Examples

```

library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

mod1 = ple_bart(Y, A, X, Xtest=X)
summary(mod1$mu_train)
summary(predict(mod1)) # Training set predictions #
summary(predict(mod1, newdata=X)) # Test set, MCMC needs to re-run #

```

predict.ple_causal_forest

Predict Patient-level Estimates: Causal Forest

Description

Get estimates of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$ using trained regression_forest and causal_forest model(s).

Usage

```

## S3 method for class 'ple_causal_forest'
predict(object, newdata = NULL, ...)

```

Arguments

object	Trained random forest (ranger) model(s).
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Data-frame with predictions of $(E(Y|X,1), E(Y|X,0), E(Y|X,1)-E(Y|X,0))$

Examples

```

library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X

```

```

A = dat_ctns$A

mod1 = ple_causal_forest(Y, A, X, Xtest=X)
summary(mod1$mu_train)
summary(predict(mod1)) # Training set predictions (oob) #
summary(predict(mod1, newdata=X)) # Test data, no oob #

```

predict.ple_glmnet *Predict Patient-level Estimates: glmnet*

Description

For continuous/binary (family="gaussian" or "binomial"), output estimates of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$. For survival, output estimates of $(HR(X,A=1), HR(X,A=0), HR(X,A=1)-HR(X,A=0))$.

Usage

```

## S3 method for class 'ple_glmnet'
predict(object, newdata = NULL, ...)

```

Arguments

object	Trained glmnet model(s).
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Data-frame with predictions of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$ for continuous/binary outcomes. For survival, returns $(HR(X,A=1), HR(X,A=0), HR(X,A=1)-HR(X,A=0))$.

Examples

```

library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

mod1 = ple_glmnet(Y, A, X, Xtest=X, family="gaussian")
summary(mod1$mu_train)
summary(predict(mod1)) # Training set predictions, newdata=NULL

```

```
summary(predict(mod1, newdata=X)) # Test set predictions
```

```
predict.ple_ranger      Predict Patient-level Estimates: Ranger
```

Description

Get estimates of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$ using trained random forest (ranger) model(s).

Usage

```
## S3 method for class 'ple_ranger'
predict(object, newdata = NULL, oob = FALSE, ...)
```

Arguments

object	Trained random forest (ranger) model(s).
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
oob	Use out-of-bag predictions (default=TRUE). Only applicable for training data (newdata=NULL).
...	Any additional parameters, not currently passed through.

Value

Data-frame with predictions of $(E(Y|X,A=1), E(Y|X,A=0), E(Y|X,A=1)-E(Y|X,A=0))$ or survival probabilities and difference in restricted mean survival time (RMST), $(S(T|X,A=1), S(T|X,A=0), RMST(A=1,X)-RMST(A=0,X))$

Examples

```
library(StratifiedMedicine)
## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Default (treatment-specific ranger models) #
mod1 = ple_ranger(Y, A, X, Xtest=X)
summary( predict(mod1 ) ) # oob predictions for training
summary( predict(mod1, newdata=X ) ) # new-predictions, no oob here
```

predict.ple_train *Patient-level Estimates Model: Prediction*

Description

Prediction function for the trained patient-level estimate (ple) model.

Usage

```
## S3 method for class 'ple_train'  
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained ple model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Data-frame with predictions (depends on trained ple model).

See Also

[PRISM](#)

Examples

```
library(StratifiedMedicine)  
## Continuous ##  
dat_ctns = generate_subgrp_data(family="gaussian")  
Y = dat_ctns$Y  
X = dat_ctns$X  
A = dat_ctns$A  
  
# Fit through ple_train wrapper #  
mod2 = ple_train(Y=Y, A=A, X=X, Xtest=X, ple="ple_ranger" )  
summary(mod2$mu_train)  
  
res2 = predict(mod2) # newdata=NULL, training data #  
res3 = predict(mod2, newdata=X) # test data #  
summary(res2)  
summary(res3)
```

predict.PRISM	<i>PRISM: Patient Response Identifier for Stratified Medicine (Predictions)</i>
---------------	---

Description

Predictions for PRISM algorithm. Given the training set (Y,A,X) or new test set (Xtest), output ple predictions and identified subgroups with correspond parameter estimates.

Usage

```
## S3 method for class 'PRISM'
predict(object, newdata = NULL, type = "all", ...)
```

Arguments

object	Trained PRISM model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
type	Type of prediction. Default is "all" (ple, submod, and param predictions). Other options include "ple" (ple predictions), "submod" (submod predictions with associated parameter estimates).
...	Any additional parameters, not currently passed through.

Value

Data-frame with predictions (ple, submod, or both).

Examples

```
## Load library ##
library(StratifiedMedicine)

##### Examples: Continuous Outcome #####

dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Run Default: filter_glmnet, ple_ranger, submod_lmtree, param_ple #
res0 = PRISM(Y=Y, A=A, X=X)
summary( predict(res0) ) # all #
summary( predict(res0, type="ple") )
summary( predict(res0, type="submod") )
```

predict.submod_ctree *Predict submod: CTREE*

Description

Predict subgroups and obtain subgroup-specific estimates, $E(Y|X)$ or $PLE(X)$, for a trained ctree model (depends on if outcome_PLE argument)

Usage

```
## S3 method for class 'submod_ctree'  
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained ctree model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Identified subgroups with subgroup-specific predictions of $E(Y|X)$ or $PLE(X)$.

- Subgrps - Identified subgroups
- pred - Predictions, $E(Y|X)$ or $PLE(X)$ by subgroup.

Examples

```
library(StratifiedMedicine)  
  
## Continuous ##  
dat_ctns = generate_subgrp_data(family="gaussian")  
Y = dat_ctns$Y  
X = dat_ctns$X  
A = dat_ctns$A  
  
res_ctree1 = submod_ctree(Y, A, X, Xtest=X)  
# Predict subgroups / estimates #  
out = predict(res_ctree1, newdata=X)
```

predict.submod_lmtree *Predict submod: lmtree*

Description

Predict subgroups and obtain subgroup-specific estimates of $E(Y|A=1)-E(Y|A=0)$.

Usage

```
## S3 method for class 'submod_lmtree'  
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained lmtree model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Identified subgroups with subgroup-specific predictions of $E(Y|A=1)-E(Y|A=0)$.

- Subgrps - Identified subgroups
- pred - Predictions, $E(Y|A=1)-E(Y|A=0)$ by subgroup.

Examples

```
library(StratifiedMedicine)  
  
## Continuous ##  
dat_ctns = generate_subgrp_data(family="gaussian")  
Y = dat_ctns$Y  
X = dat_ctns$X  
A = dat_ctns$A  
  
res_lmtree1 = submod_lmtree(Y, A, X, Xtest=X)  
# Predict subgroups / estimates #  
out = predict(res_lmtree1)
```

predict.submod_otr *Predict submod: OTR CTREE*

Description

Predict subgroups and obtain subgroup-specific estimates, $P(\text{PLE} > \text{thres})$, for a trained ctree OTR model.

Usage

```
## S3 method for class 'submod_otr'  
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained ctree model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Identified subgroups with subgroup-specific predictions of $P(\text{PLE} > \text{thres})$.

- Subgrps - Identified subgroups
- pred - Predictions, $P(\text{PLE} > \text{thres})$ by identified subgroup.

Examples

```
library(StratifiedMedicine)  
  
## Continuous ##  
dat_ctns = generate_subgrp_data(family="gaussian")  
Y = dat_ctns$Y  
X = dat_ctns$X  
A = dat_ctns$A  
  
## Estimate PLEs (through Ranger) ##  
res.ple = ple_train(Y, A, X, Xtest=X, family="gaussian", ple="ple_ranger")  
  
## Fit OTR Subgroup Model ##  
res_otr = submod_otr(Y, A, X, Xtest=X, mu_train = res.ple$mu_train)  
out = predict(res_otr, newdata=X)  
plot(res_otr$mod)
```

predict.submod_rpart *Predict submod: rpart*

Description

Predict subgroups and obtain subgroup-specific estimates, $E(Y|X)$ or $PLE(X)$, for a trained ctree model (depends on if outcome_PLE argument)

Usage

```
## S3 method for class 'submod_rpart'  
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained ctree model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Identified subgroups with subgroup-specific predictions of $E(Y|X)$ or $PLE(X)$.

- Subgrps - Identified subgroups
- pred - Predictions, $E(Y|X)$ or $PLE(X)$ by subgroup.

Examples

```
library(StratifiedMedicine)  
  
## Continuous ##  
dat_ctns = generate_subgrp_data(family="gaussian")  
Y = dat_ctns$Y  
X = dat_ctns$X  
A = dat_ctns$A  
  
res_rpart = submod_rpart(Y, A, X, Xtest=X)  
# Predict subgroups / estimates #  
out = predict(res_rpart, newdata=X)
```

predict.submod_train *Subgroup Identification: Train Model (Predictions)*

Description

Prediction function for the trained subgroup identification model (submod).

Usage

```
## S3 method for class 'submod_train'
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained submod model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Identified subgroups with subgroup-specific predictions (depends on subgroup model)

- Subgrps - Identified subgroups
- pred - Predictions, depends on subgroup model

Examples

```
library(StratifiedMedicine)
## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Fit submod_lmtree directly #
mod1 = submod_lmtree(Y, A, X, Xtest=X)
out1 = predict(mod1)
plot(mod1$mod)

# Fit through submod_train wrapper #
mod2 = submod_train(Y=Y, A=A, X=X, Xtest=X, submod="submod_lmtree")
out2 = predict(mod2)
plot(mod2$fit$mod)
```

```
predict.submod_weibull
```

Predict submod: Model-based partitioning (Weibull)

Description

Predict subgroups and obtain subgroup-specific point-estimates (in pprogress).

Usage

```
## S3 method for class 'submod_weibull'
predict(object, newdata = NULL, ...)
```

Arguments

object	Trained MOB (Weibull) model.
newdata	Data-set to make predictions at (Default=NULL, predictions correspond to training data).
...	Any additional parameters, not currently passed through.

Value

Identified subgroups with subgroup-specific predictions.

- Subgrps - Identified subgroups
- pred - Predictions, based on weibull regression fit, estimate hazard ratio by subgroup.

Examples

```
library(StratifiedMedicine)
# Survival Data #
require(TH.data); require(coin)
data("GBSG2", package = "TH.data")
surv.dat = GBSG2
# Design Matrices #
Y = with(surv.dat, Surv(time, cens))
X = surv.dat[,! (colnames(surv.dat) %in% c("time", "cens")) ]
A = rbinom( n = dim(X)[1], size=1, prob=0.5 ) ## simulate null treatment

res_weibull = submod_weibull(Y, A, X, Xtest=X, family="survival")
out = predict(res_weibull)
plot(res_weibull$mod)
```

Description

PRISM algorithm. Given a data-set of (Y, A, X) (Outcome, treatment, covariates), the PRISM identifies potential subgroup along with point and variability metrics. This four step procedure (filter, ple, submod, param) is flexible and accepts user-inputs at each step.

Usage

```
PRISM(Y, A = NULL, X, Xtest = NULL, family = "gaussian",
      filter = "filter_glmnet", ple = NULL, submod = NULL,
      param = NULL, alpha_ovrl = 0.05, alpha_s = 0.05,
      filter.hyper = NULL, ple.hyper = NULL, submod.hyper = NULL,
      param.hyper = NULL, bayes = NULL, prefilter_resamp = FALSE,
      resample = NULL, stratify = TRUE, R = 100, filter.resamp = NULL,
      ple.resamp = NULL, submod.resamp = NULL, verbose = TRUE,
      verbose.resamp = FALSE, seed = 777)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (ex: a=1,...,A, should be numeric). Default is NULL, which searches for prognostic variables (Y~X).
X	Covariate space. Variables types (ex: numeric, factor, ordinal) should be set to align with subgroup model (submod argument). For example, for lmtree, binary variables coded as numeric (ex: 0, 1) are treated differently than the corresponding factor version (ex: "A", "B"). Filter and PLE models provided in the StratifiedMedicine package can accomodate all variable types.
Xtest	Test set. Default is NULL which uses X (training set). Variable types should match X.
family	Outcome type. Options include "gaussian" (default), "binomial", and "survival".
filter	Maps (Y,A,X) => (Y,A,X.star) where X.star has potentially less covariates than X. Default is "filter_glmnet", "None" uses no filter.
ple	PLE (Patient-Level Estimate) function. Maps the observed data to PLEs. (Y,A,X) ==> PLE(X). Default for "gaussian"/"binomial" is "ple_ranger" (treatment-specific random forest models). The default for "survival" is "ple_glmnet" (elastic net (glmnet) cox regression). "None" uses no ple.
submod	Subgroup identification model function. Maps the observed data and/or PLEs to subgroups. Default of "gaussian"/"binomial" is "submod_lmtree" (MOB with OLS loss). Default for "survival" is "submod_weibull" (MOB with weibull loss). "None" uses no submod.

param	Parameter estimation and inference function. Based on the discovered subgroups, perform inference through the input function (by name). Default for "gaussian"/"binomial" is "param_PLE", default for "survival" is "param_cox".
alpha_ovr1	Two-sided alpha level for overall population. Default=0.05
alpha_s	Two-sided alpha level at subgroup level. Default=0.05
filter.hyper	Hyper-parameters for the Filter function (must be list). Default is NULL.
ple.hyper	Hyper-parameters for the PLE function (must be list). Default is NULL.
submod.hyper	Hyper-parameters for the SubMod function (must be list). Default is NULL.
param.hyper	Hyper-parameters for the Param function (must be list). Default is NULL.
bayes	Based on input point estimates/SEs, this uses a bayesian based approach to obtain ests, SEs, CIs, and posterior probabilities. Currently includes "norm_norm" (normal prior at overall estimate with large uninformative variance; normal posterior). Default=NULL.
prefilter_resamp	Option to filter the covariate space (based on filter model) prior to resampling. Default=FALSE.
resample	Resampling method for resample-based estimates and variability metrics. Options include "Boostrap", "Permutation", and "CV". Default=NULL (No resampling).
stratify	Stratified resampling (Default=TRUE)
R	Number of resamples (default=100)
filter.resamp	Filter function during resampling, default=NULL (use filter)
ple.resamp	PLE function during resampling, default=NULL (use ple)
submod.resamp	submod function for resampling, default=NULL (use submod)
verbose	Detail progress of PRISM? Default=TRUE
verbose.resamp	Output iterations during resampling? Default=FALSE
seed	Seed for PRISM run (Default=777)

Value

Trained PRISM object. Includes filter, ple, submod, and param outputs.

- filter.mod - Filter model
- filter.vars - Variables remaining after filtering
- ple.fit - Fitted ple model (model fit, other fit outputs)
- mu_train - Patient-level estimates (train)
- mu_test - Patient-level estimates (test)
- submod.fit - Fitted submod model (model fit, other fit outputs)
- out.train - Training data-set with identified subgroups
- out.test - Test data-set with identified subgroups
- Rules - Subgroup rules / definitions
- param.dat - Parameter estimates and variability metrics (depends on param)
- resamp.dist - Resampling distributions (NULL if no resampling is done)
- bayes.fun - Function to simulate posterior distribution (NULL if no bayes)

References

Jemielita and Mehrotra (2019 in progress)

Examples

```
## Load library ##
library(StratifiedMedicine)

##### Examples: Continuous Outcome #####

dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Run Default: filter_glmnet, ple_ranger, submod_lmtree, param_ple #
res0 = PRISM(Y=Y, A=A, X=X)
summary(res0)
plot(res0, type="PLE:density") # distribution of PLEs
plot(res0, type="PLE:waterfall") # PLE waterfall plot
plot(res0$submod.fit$mod) # Plot of subgroup model
res0$param.dat # overall/subgroup specific parameter estimates/inference
plot(res0) # Forest plot: overall/subgroup specific parameter estimates (CIs)

# Without filtering #
res1 = PRISM(Y=Y, A=A, X=X, filter="None" )
summary(res1)
plot(res1$submod.fit$mod)
plot(res1)

## With bootstrap (No filtering) ##

res_boot = PRISM(Y=Y, A=A, X=X, resample = "Bootstrap", R=50, verbose.resamp = TRUE)
# Plot of distributions and P(est>0) #
plot(res_boot, type="resample", estimand = "E(Y|A=1)-E(Y|A=0)"+geom_vline(xintercept = 0)
aggregate(I(est>0)~Subgrps, data=res_boot$resamp.dist, FUN="mean")

# Survival Data ##

library(survival)
require(TH.data); require(coin)
data("GBSG2", package = "TH.data")
surv.dat = GBSG2
# Design Matrices ###
Y = with(surv.dat, Surv(time, cens))
X = surv.dat[,!colnames(surv.dat) %in% c("time", "cens")] ]
set.seed(513)
A = rbinom( n = dim(X)[1], size=1, prob=0.5 )

# Default: PRISM: glmnet ==> MOB (Weibull) ==> Cox; bootstrapping posterior prob/inference #
```

```

res_weibull1 = PRISM(Y=Y, A=A, X=X, ple=NULL, resample="Bootstrap", R=100,
                    verbose.resamp = TRUE)
plot(res_weibull1$submod.fit$mod)
plot(res_weibull1)
plot(res_weibull1, type="resample", estimand = "HR(A=1 vs A=0)") + geom_vline(xintercept = 1)
aggregate(I(est<1)~Subgrps, data=res_weibull1$resamp.dist, FUN="mean")

# PRISM: ENET ==> CTREE ==> Cox; bootstrapping for posterior prob/inference #
res_ctree1 = PRISM(Y=Y, A=A, X=X, ple=NULL, submod = "submod_ctree",
                  resample="Bootstrap", R=100, verbose.resamp = TRUE)
plot(res_ctree1$submod.fit$submod.fit$mod)
plot(res_ctree1)
plot(res_ctree1, type="resample", estimand="HR(A=1 vs A=0)") + geom_vline(xintercept = 1)
aggregate(I(est<1)~Subgrps, data=res_ctree1$resamp.dist, FUN="mean")

```

PRISM_resamp	<i>PRISM (Resample): Patient Response Identifier for Stratified Medicine</i>
--------------	--

Description

Based on initial PRISM fit (PRISM_train), run resampling (Bootstrap, Permutation, or cross-validation). Used directly in PRISM.

Usage

```

PRISM_resamp(PRISM.fit, Y, A, X, Xtest = NULL, family = "gaussian",
             filter = "filter_glmnet", ple = NULL, submod = NULL,
             param = NULL, alpha_ovr1 = 0.05, alpha_s = 0.05,
             filter.hyper = NULL, ple.hyper = NULL, submod.hyper = NULL,
             param.hyper = NULL, verbose = TRUE, prefilter_resamp = FALSE,
             resample = NULL, R = 100, stratify = TRUE)

```

Arguments

PRISM.fit	Fitted PRISM model
Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (ex: a=1,...,A, should be numeric). Default is NULL, which searches for prognostic variables (Y~X).
X	Covariate space. Variables types (ex: numeric, factor, ordinal) should be set to align with subgroup model (submod argument). For example, for lmtree, binary variables coded as numeric (ex: 0, 1) are treated differently than the corresponding factor version (ex: "A", "B"). Filter and PLE models provided in the StratifiedMedicine package can accommodate all variable types.
Xtest	Test set. Default is NULL which uses X (training set). Variable types should match X.

family	Outcome type. Options include "gaussian" (default), "binomial", and "survival".
filter	Maps (Y,A,X) => (Y,A,X.star) where X.star has potentially less covariates than X. Default is "filter_glmnet", "None" uses no filter.
ple	PLE (Patient-Level Estimate) function. Maps the observed data to PLEs. (Y,A,X) ==> PLE(X). Default for "gaussian"/"binomial" is "ple_ranger" (treatment-specific random forest models). The default for "survival" is "ple_glmnet" (elastic net (glmnet) cox regression). "None" uses no ple.
submod	Subgroup identification model function. Maps the observed data and/or PLEs to subgroups. Default of "gaussian"/"binomial" is "submod_lmtree" (MOB with OLS loss). Default for "survival" is "submod_weibull" (MOB with weibull loss). "None" uses no submod.
param	Parameter estimation and inference function. Based on the discovered subgroups, perform inference through the input function (by name). Default for "gaussian"/"binomial" is "param_PLE", default for "survival" is "param_cox".
alpha_ovrl	Two-sided alpha level for overall population. Default=0.05
alpha_s	Two-sided alpha level at subgroup level. Default=0.05
filter.hyper	Hyper-parameters for the Filter function (must be list). Default is NULL.
ple.hyper	Hyper-parameters for the PLE function (must be list). Default is NULL.
submod.hyper	Hyper-parameters for the SubMod function (must be list). Default is NULL.
param.hyper	Hyper-parameters for the Param function (must be list). Default is NULL.
verbose	Detail progress of PRISM? Default=TRUE
prefilter_resamp	Option to filter the covariate space (based on filter model) prior to resampling. Default=FALSE.
resample	Resampling method for resample-based estimates and variability metrics. Options include "Bootstrap", "Permutation", and "CV". Default=NULL (No resampling).
R	Number of resamples (default=100)
stratify	Stratified resampling (Default=TRUE)

Value

Trained PRISM object. Includes filter, ple, submod, and param outputs.

- param.dat - Parameter estimates and variability metrics (depends on param)
- resamp.dist - - Resampling distributions

PRISM_train

*PRISM (Train): Patient Response Identifier for Stratified Medicine***Description**

Train the PRISM algorithm given a training set (Y, A, X) and test set (Xtest). Used directly in PRISM and PRISM_resamp.

Usage

```
PRISM_train(Y, A, X, Xtest = NULL, family = "gaussian",
  filter = "filter_glmnet", ple = NULL, submod = NULL,
  param = NULL, alpha_ovr1 = 0.05, alpha_s = 0.05,
  filter.hyper = NULL, ple.hyper = NULL, submod.hyper = NULL,
  param.hyper = NULL, verbose = TRUE)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (ex: a=1,...,A, should be numeric). Default is NULL, which searches for prognostic variables (Y~X).
X	Covariate space. Variables types (ex: numeric, factor, ordinal) should be set to align with subgroup model (submod argument). For example, for lmtree, binary variables coded as numeric (ex: 0, 1) are treated differently than the corresponding factor version (ex: "A", "B"). Filter and PLE models provided in the StratifiedMedicine package can accomodate all variable types.
Xtest	Test set. Default is NULL which uses X (training set). Variable types should match X.
family	Outcome type. Options include "gaussian" (default), "binomial", and "survival".
filter	Maps (Y,A,X) => (Y,A,X.star) where X.star has potentially less covariates than X. Default is "filter_glmnet", "None" uses no filter.
ple	PLE (Patient-Level Estimate) function. Maps the observed data to PLEs. (Y,A,X) ==> PLE(X). Default for "gaussian"/"binomial" is "ple_ranger" (treatment-specific random forest models). The default for "survival" is "ple_glmnet" (elastic net (glmnet) cox regression). "None" uses no ple.
submod	Subgroup identification model function. Maps the observed data and/or PLEs to subgroups. Default of "gaussian"/"binomial" is "submod_lmtree" (MOB with OLS loss). Default for "survival" is "submod_weibull" (MOB with weibull loss). "None" uses no submod.
param	Parameter estimation and inference function. Based on the discovered subgroups, perform inference through the input function (by name). Default for "gaussian"/"binomial" is "param_PLE", default for "survival" is "param_cox".
alpha_ovr1	Two-sided alpha level for overall population. Default=0.05
alpha_s	Two-sided alpha level at subgroup level. Default=0.05

filter.hyper	Hyper-parameters for the Filter function (must be list). Default is NULL.
ple.hyper	Hyper-parameters for the PLE function (must be list). Default is NULL.
submod.hyper	Hyper-parameters for the SubMod function (must be list). Default is NULL.
param.hyper	Hyper-parameters for the Param function (must be list). Default is NULL.
verbose	Detail progress of PRISM? Default=TRUE

Value

Trained PRISM object. Includes filter, ple, submod, and param outputs.

- filter.mod - Filter model
- filter.vars - Variables remaining after filtering
- ple.fit - Fitted ple model (model fit, other fit outputs)
- mu_train - Patient-level estimates (train)
- mu_test - Patient-level estimates (test)
- submod.fit - Fitted submod model (model fit, other fit outputs)
- Subgrps.train - Training data-set with identified subgroups
- Subgrps.test - Test data-set with identified subgroups
- Rules - Subgroup rules / definitions
- param.dat - Parameter estimates and variability metrics (depends on param)

submod_ctree	<i>Subgroup Identification: Conditional Inference Trees (ctree)</i>
--------------	---

Description

Uses the ctree (conditional inference trees) algorithm to identify subgroups (Hothorn, Hornik, Zeileis 2006). Usable for continuous, binary, or survival outcomes. Option to use the observed outcome or PLEs for subgroup identification.

Usage

```
submod_ctree(Y, A, X, Xtest, mu_train, minbucket = floor(dim(X)[1] *
  0.1), maxdepth = 4, outcome_PLE = FALSE, family = "gaussian", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models)

minbucket	Minimum number of observations in a tree node. Default = floor(dim(train)[1]*0.05)
maxdepth	Maximum depth of any node in the tree (default=4)
outcome_PLE	If TRUE, use PLE as outcome (mu_train must contain PLEs).
family	Outcome type ("gaussian", "binomial", "survival), default is "gaussian"
...	Any additional parameters, not currently passed through.

Value

Trained ctree model.

- mod - ctree model object

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

res_ctree1 = submod_ctree(Y, A, X, Xtest=X, family="gaussian")
res_ctree2 = submod_ctree(Y, A, X, Xtest=X, family="gaussian", maxdepth=2, minsize=100)
plot(res_ctree1$mod)
plot(res_ctree2$mod)
```

submod_lmtree

Subgroup Identification: Model-based partitioning (lmtree)

Description

Uses the lmtree (model-based partitioning, OLS) algorithm to identify subgroups (Zeileis, Hothorn, Hornik 2008). Usable for continuous and binary outcomes.

Usage

```
submod_lmtree(Y, A, X, Xtest, mu_train, minsize = floor(dim(X)[1] * 0.1),
  maxdepth = 4, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models)
minsize	Minimum number of observations in a tree node. Default = floor(dim(train)[1]*0.05)
maxdepth	Maximum depth of any node in the tree (default=4)
...	Any additional parameters, not currently passed through.

Value

Trained lmtree model.

- mod - lmtree model object

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A
train = data.frame(Y, A, X)
# Outcome/treatment must be labeled as Y/A #

res_lmtree1 = submod_lmtree(Y, A, X, Xtest=X)
res_lmtree2 = submod_lmtree(Y, A, X, Xtest=X, maxdepth=2, minsize=100)
plot(res_lmtree1$mod)
plot(res_lmtree2$mod)
```

submod_otr

Subgroup Identification: Optimal Treatment Regime (through ctree)

Description

For continuous, binary, or survival outcomes, regress $I(\text{PLE} > \text{thres}) \sim X$ with $\text{weights} = \text{abs}(\text{PLE})$ in ctree.

Usage

```
submod_otr(Y, A, X, Xtest, mu_train, minbucket = floor(dim(X)[1] * 0.1),
  maxdepth = 4, thres = ">0", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models)
minbucket	Minimum number of observations in a tree node. Default = floor(dim(train)[1]*0.05)
maxdepth	Maximum depth of any node in the tree (default=4)
thres	Threshold for PLE, ex: I(PLE>thres). Default is ">0". Direction can be reversed and can include equality sign (ex: "<=")
...	Any additional parameters, not currently passed through.

Value

Trained ctree (optimal treatment regime) model.

- mod - tree (OTR) model object

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

## Estimate PLEs (through Ranger) ##
res.ple = ple_model(Y, A, X, Xtest=X, family="gaussian", ple="ple_ranger")

## Fit OTR Subgroup Model ##
res_otr = submod_otr(Y, A, X, Xtest=X, mu_train = res.ple$mu_train)
plot(res_otr$mod)
```

submod_rpart

Subgroup Identification: CART (rpart)

Description

Uses the CART algorithm (rpart) to identify subgroups. Usable for continuous and binary outcomes. Option to use the observed outcome or PLEs for subgroup identification.

Usage

```
submod_rpart(Y, A, X, Xtest, mu_train, minbucket = floor(dim(X)[1] *
  0.1), maxdepth = 4, outcome_PLE = FALSE, family = "gaussian", ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models)
minbucket	Minimum number of observations in a tree node. Default = floor(dim(train)[1]*0.05)
maxdepth	Maximum depth of any node in the tree (default=4)
outcome_PLE	If TRUE, use PLE as outcome (mu_train must contain PLEs). Else use observed outcome Y
family	Outcome type ("gaussian", "binomial"), default is "gaussian"
...	Any additional parameters, not currently passed through.

Value

Trained rpart (CART).

- mod - rpart model as partykit object

Examples

```
library(StratifiedMedicine)

## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

require(rpart)
res_rpart1 = submod_rpart(Y, A, X, Xtest=X)
res_rpart2 = submod_rpart(Y, A, X, Xtest=X, maxdepth=2, minbucket=100)
plot(res_rpart1$mod)
plot(res_rpart2$mod)
```

submod_train

Subgroup Identification: Train Model

Description

Wrapper function to train a subgroup model (submod). Used directly in PRISM and can be used to directly fit a submod model by name.

Usage

```
submod_train(Y, A, X, Xtest, mu_train = NULL, family = "gaussian",
            submod, hyper = NULL, ...)
```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models). Default=NULL
family	Outcome type ("gaussian", "binomial", "survival"). Default="gaussian".
submod	Subgroup identification (submod) function. Maps the observed data and/or PLEs to subgroups.
hyper	Hyper-parameters for submod (must be list). Default is NULL.
...	Any additional parameters, not currently passed through.

Value

Trained subgroup model and subgroup predictions/estimates for train/test sets.

- fit - trained subgroup model
- Subgrps.train - Identified subgroups (training set)
- Subgrps.test - Identified subgroups (test set)
- pred.train - Predictions (training set)
- pred.test - Predictions (test set)
- Rules - Definitions for subgroups, if provided in fitted submod output.

See Also

[PRISM](#)

Examples

```

library(StratifiedMedicine)
## Continuous ##
dat_ctns = generate_subgrp_data(family="gaussian")
Y = dat_ctns$Y
X = dat_ctns$X
A = dat_ctns$A

# Fit submod_lmtree directly #
mod1 = submod_lmtree(Y, A, X, Xtest=X)
plot(mod1$mod)

# Fit through submod_train wrapper #
mod2 = submod_train(Y=Y, A=A, X=X, Xtest=X, submod="submod_lmtree")
plot(mod2$fit$mod)

```

submod_weibull *Subgroup Identification: Model-based partitioning (Weibull)*

Description

Uses the MOB (with weibull loss function) algorithm to identify subgroups (Zeileis, Hothorn, Hornik 2008; Seibold, Zeileis, Hothorn 2016). Usable for survival outcomes.

Usage

```

submod_weibull(Y, A, X, Xtest, mu_train, minsize = floor(dim(X)[1] *
  0.1), maxdepth = 4, ...)

```

Arguments

Y	The outcome variable. Must be numeric or survival (ex; Surv(time,cens))
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models)
minsize	Minimum number of observations in a tree node. Default = floor(dim(train)[1]*0.05)
maxdepth	Maximum depth of any node in the tree (default=4)
...	Any additional parameters, not currently passed through.

Value

Trained MOB (Weibull) model.

- mod - MOB (Weibull) model object

Examples

```

library(StratifiedMedicine)

## Load TH.data (no treatment; generate treatment randomly to simulate null effect) ##
data("GBSG2", package = "TH.data", envir = e <- new.env() )
surv.dat = e$GBSG2
## Design Matrices ###
Y = with(surv.dat, Surv(time, cens))
X = surv.dat[,!(colnames(surv.dat) %in% c("time", "cens")) ]
A = rbinom( n = dim(X)[1], size=1, prob=0.5 )
res_weibull = submod_weibull(Y, A, X, Xtest=X, family="survival")
plot(res_weibull$mod)

```

summary.PRISM

PRISM: Patient Response Identifier for Stratified Medicine (Summary)

Description

Predictions for PRISM algorithm. Given the training set (Y,A,X) or new test set (Xtest), output predictions and identified subgroups with correspond parameter estimates.

Usage

```

## S3 method for class 'PRISM'
summary(object, ...)

```

Arguments

object	Trained PRISM model.
...	Any additional parameters, not currently passed through.

Value

List of key PRISM outputs: (1) Configuration, (2) Variables that pass filter (if filter is used), (3) Number of Identified Subgroups, and (4) Parameter Estimates, SEs, and CIs for each subgroup/estimand

Index

[filter_glmnet](#), [3](#)
[filter_ranger](#), [4](#)

[generate_subgrp_data](#), [5](#)

[param_combine](#), [6](#), [7](#), [9](#), [12](#)
[param_cox](#), [6](#), [6](#)
[param_dr](#), [8](#)
[param_lm](#), [6](#), [9](#)
[param_ple](#), [10](#)
[param_rmst](#), [6](#), [11](#)
[ple_bart](#), [12](#)
[ple_causal_forest](#), [14](#)
[ple_glmnet](#), [15](#)
[ple_ranger](#), [16](#)
[ple_train](#), [17](#)
[plot.PRISM](#), [18](#)
[predict.ple_bart](#), [19](#)
[predict.ple_causal_forest](#), [20](#)
[predict.ple_glmnet](#), [21](#)
[predict.ple_ranger](#), [22](#)
[predict.ple_train](#), [23](#)
[predict.PRISM](#), [24](#)
[predict.submod_ctree](#), [25](#)
[predict.submod_lmtree](#), [26](#)
[predict.submod_otr](#), [27](#)
[predict.submod_rpart](#), [28](#)
[predict.submod_train](#), [29](#)
[predict.submod_weibull](#), [30](#)
[PRISM](#), [16](#), [18](#), [19](#), [23](#), [31](#), [42](#)
[PRISM_resamp](#), [34](#)
[PRISM_train](#), [36](#)

[ranger](#), [16](#)

[submod_ctree](#), [37](#)
[submod_lmtree](#), [38](#)
[submod_otr](#), [39](#)
[submod_rpart](#), [40](#)
[submod_train](#), [42](#)

[submod_weibull](#), [43](#)
[summary.PRISM](#), [44](#)