Package 'StratifiedMedicine'

April 22, 2020

Type Package

Title Stratified Medicine

Version 0.2.3

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) <arXiv:1912.03337>. 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 (>= 3.4),

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

RoxygenNote 6.1.1

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

Suggests knitr, rmarkdown, MASS, BART, randomForestSRC, grf, survRM2, TH.data, coin, rpart, testthat (>= 2.1.0)

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-22 12:02:09 UTC

R topics documented:

`filter_glmnet` *Filter: Elastic Net (glmnet)*

Description

Filter variables through elastic net (Zou and Hastie 2005). Default is to regress Y~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~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)

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) Regularization Paths for Generalized Linear Models via Coordinate Descent, https://web.stanford.edu/~hastie/Papers/glmnet.pdf Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010 Vol. 33(1), 1-22 Feb 2010.

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 (T=VI_j/SE(VE_j) for feature j through the delete-d jackknife), as described in Ishwaran and Lu 2017. Default is to remove variables if one-sided (VI>0) 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)

References

Wright, M. N. & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. J Stat Softw 77:1-17. https://doi.org/10.18637/jss.v077.i01.

Examples

```
library(StratifiedMedicine)
library(ranger)

## 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
```

`filter_train` *Filter: Train Filter Model*

Description

Wrapper function to train a filter model. Options include elastic net (glmnet) and random forest based variable importance (ranger). Used directly in PRISM.

Usage

```
filter_train(Y, A, X, family = "gaussian", filter, 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.
`family`	Outcome type ("gaussian", "binomial", "survival"). Default is "gaussian".
`filter`	Filter function. Potentially reduces covariate space, (Y, A, X) ==> (Y, A, Xstar).
`hyper`	Hyper-parameters for the filter model (must be list). Default is NULL.
`...`	Any additional parameters, not currently passed through.

Value

Trained filter model and vector of variable names that pass the filter.

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

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 = filter_train(Y, A, X, filter="filter_glmnet")
mod1$filter.vars

mod2 = filter_train(Y, A, X, filter="filter_ranger")
mod2$filter.vars
```

---

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, null = FALSE,
  ...)
```

Arguments

n	sample size (default=800)
seed	seed number (default=513413)
family	Outcome type ("gaussian", "binomial", "survival")
null	Simulate null hypothesis of no treatment effect and no subgruops. Default is FALSE.
...	Any additional parameters, not currently passed through.

Value

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

`param_combine` *Overall Population Estimate: Aggregating Subgroup-Specific Parameter Estimates*

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` *Parameter Estimation: Cox Regression*

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_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 (log 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=logHR, SE=SE(logHR), LCL/UCL = lower/upper confidence limit on logHR scale, pval = p-value).

References

Andersen, P. and Gill, R. (1982). Cox's regression model for counting processes, a large sample study. Annals of Statistics 10, 1100-1120.

See Also

param_combine

Examples

```
library(StratifiedMedicine)
library(survival)
require(TH.data)
require(coin)

surv.dat = GBSG2
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")

## Parameter-Estimation ##
```

```
params = param_cox(Y, A, X, Subgrps = res_weibull$Subgrps.train, alpha_ovrl=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). For continuous and binary outcomes, this outputs estimates for E(Y|A=1), E(Y|A=0), and E(Y|A=1)-E(Y|A=0).

Usage

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

Arguments

Y	The outcome variable. Must be numeric (binary, continuous)
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
...	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).

References

Funk et al. Doubly Robust Estimation of Causal Effects. Am J Epidemiol 2011. 173(7): 761-767.

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_ovrl=0.05,
                  alpha_s=0.05)
params
```

---

param_ple *Parameter Estimation: Patient-Level Estimates*

Description

For each identified subgroup and in the overall population, average the patient-level estimates of E(Y|A=1), E(Y|A=0), and E(Y|A=1)-E(Y|A=0). Pseudo-outcomes are used for variance estimates (Jemielita and Mehrotra 2019).

Usage

```
param_ple(Y, A, X, mu_hat, Subgrps, alpha_ovrl, 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_ovrl	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).

References

Jemielita T, Mehrotra D. PRISM: Patient Response Identifiers for Stratified Medicine. https://arxiv.org/abs/1912.03337

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_ovrl=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_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 (RMST) and corresponding variability metrics, for overall and subgroups.

- param.dat - Parameter estimates and variability metrics

References

Uno et al. Moving beyond the hazard ratio in quantifying the between-group difference in survival analysis. Journal of clinical Oncology 2014, 32, 2380-2385.

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")

# 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", sparse = FALSE, K = 10,
  ...)
```

Arguments

`Y`	The outcome variable. Must be numeric.
`A`	Treatment variable. (a=1,...A)
`X`	Covariate space.
`Xtest`	Test set
`family`	Outcome type ("gaussian", "binomial"), default is "gaussian" which uses wbart (BART for continuous outcomes). Probit-based BART ("pbart") is used for family="binomial"

sparse	Whether to perform variable selection based on sparse Dirichlet prior instead of uniform. See "gbart" in the BART R package for more details as well as Linero 2016. Default is FALSE.
K	For survival, coarse times per the quantiles 1/K,..,K/K. For AFT (abart), K=10. For more accurate survival predictions, set K=100 (default in abart).
...	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.

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

References

Chipman, H., George, E., and McCulloch R. (2010) Bayesian Additive Regression Trees. The Annals of Applied Statistics, 4,1, 266-298 <doi: 10.1214/09-AOAS285>.

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)
```

---

ple_causal_forest *Patient-level Estimates: Causal Forest*

Description

Uses the causal forest algorithm (grf R package) to obtain patient-level estimates, E(Y|A=1), E(Y|A=0), and E(Y|A=1)-E(Y|A=0). Usable 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.

- mod - trained model(s)
- pred.fun - Prediction function for trained model(s)

References

Athey S, Tibshirani J, Wagner S. Generalized Random Forests. https://arxiv.org/abs/1610.01271

---

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).

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

References

Friedman, J., Hastie, T. and Tibshirani, R. (2008) Regularization Paths for Generalized Linear Models via Coordinate Descent, https://web.stanford.edu/~hastie/Papers/glmnet.pdf Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010 Vol. 33(1), 1-22 Feb 2010.

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")
```

---

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, family = "gaussian", byTrt = ifelse(family
  == "survival", FALSE, TRUE), min.node.pct = 0.1, ...)
```

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"
byTrt	If TRUE, fit treatment-specific ranger models. If FALSE, fit a single ranger model with covariate space (X, A, X*A). For "gaussian" or "binomial", default is TRUE. For "survival", default is FALSE.
min.node.pct	Minimum sample size in forest nodes (n*min.node.pct)
...	Any additional parameters, not currently passed through.

Value

Trained random forest (ranger) model(s).

- mod - trained model(s)
- pred.fun - Prediction function for trained model(s)

References

Wright, M. N. & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. J Stat Softw 77:1-17. https://doi.org/10.18637/jss.v077.i01.

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)
```

ple_rfsrc *Patient-level Estimates: randomForestSRC*

Description

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

Usage

```
ple_rfsrc(Y, A, X, Xtest, ntree = 1000, byTrt = TRUE, upweight = 100,
  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
ntree	Number of trees (default=1000)
byTrt	If TRUE, fit treatment-specific rfsrc models. If FALSE, fit a single rfsrc model with covariate space (X, A, X*A).
upweight	Whether to upweight the probability that the treatment variable is included as a splitting variable (through rfsrc's xvar.wt argument). Default=100 (other variables receive weight of 1). Only applicable for single rfsrc model (byTrt=FALSE).
min.node.pct	Minimum sample size in forest nodes (n*min.node.pct)
family	Outcome type ("gaussian", "binomial", "survival"), default is "gaussian".
...	Any additional parameters, not currently passed through.

Value

Trained random forest (rfsrc) model(s).

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

References

- Breiman L. (2001). Random forests, Machine Learning, 45:5-32.
- Ishwaran H., Kogalur U.B., Blackstone E.H. and Lauer M.S. (2008). Random survival forests, Ann. App. Statist., 2:841-860.

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_rfsrc(Y, A, X, Xtest=X)
```

`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)) or (RMST(X,A=1), RMST(X,A=0), RMST(X, A=1)-RMST(X, A=0)).

- mod - 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. Options include "tree", "forest", "resample", and "PLE:waterfall".

Usage

```
## S3 method for class 'PRISM'
plot(x, type = "tree", estimand = NULL,
  grid.data = NULL, grid.thres = ">0", tree.thres = NULL,
  est.resamp = TRUE, tree.plots = "outcome", nudge_out = 0.1,
  width_out = 0.5, nudge_dens = ifelse(tree.plots == "both", 0.3, 0.1),
  width_dens = 0.5, ...)
```

Arguments

x	PRISM object
type	Type of plot (default="tree", tree plot + parameter estimates/outcome and or probability density plots). Other options include "forest" (forest plot for overall and subgroups),"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). For "tree" and "forest", CIs are based on the observed data unless resampling is used. For bootstrap resampling, if calibrate=TRUE, then calibrated CIs along are shown, otherse CIs based on the percentile method are shown.
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: "<=").
tree.thres	Probability threshold, ex: P(Mean(A=1 vs A=0)>c. Default=NULL, which defaults to using ">0", unless param="param_cox", which "P(HR(A=1 vs A=0))<1". If a density plot is included, setting tree.thres=">c" will use green colors for values above c, and red colors for values below c. If tree.thres="<c", the reverse color scheme is used.
est.resamp	Should plot present resampling based estimates? Default=TRUE if bootstrap or CV based resampling is used. Only applicable for type="submod". If bootstrap calibration is used, calibrated CIs are presented. If no calibration, then percentile Cis are presented with the smoothed bootstrap point-estimates.
tree.plots	Type of plots to include in the "tree" plot. Default="outcome" (boxplots of treatment-specific outcomes, or counterfactual estimates if PLE!=NULL). For "density", the estimated probability density of the treatment effects is shown (normal approximation, unless resampling is used). "both" combines both plots.
nudge_out	Nudge tree outcome plot (see ggparty for details)
width_out	Width of tree outcome plot (see ggparty for details)
nudge_dens	Nudge tree density plot
width_dens	Width of density tree outcome plot
...	Additional arguments (currently ignored).

Value

Plot (ggplot2) object

See Also

PRISM

plot_dependence *Partial dependence plots: Single Variable (marginal effect) or heat map (2 to 3 variables).*

Description

Partial dependence plots: Single Variable (marginal effect) or heat map (2 to 3 variables).

Usage

```
plot_dependence(object, vars, grid.data = NULL, grid.thres = ">0",
  estimand = NULL, ...)
```

Arguments

object	Fitted PRISM object
vars	Variables to visualize (ex: c("var1", "var2", "var3")). If no grid.data provided, defaults to using seq(min(var), max(var)) for each continuous variables. For categorical, uses all categories.
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: "<=").
estimand	Estimand for which to generate dependendence or heat map plots.
...	Additional arguments (currently ignored).

Value

Plot (ggplot2) object

References

- Friedman, J. Greedy function approximation: A gradient boosting machine. Annals of statistics (2001): 1189-1232
- Zhao, Qingyuan, and Trevor Hastie. Causal interpretations of black-box models. Journal of Business & Economic Statistics, to appear. (2017).

plot_importance *Importance Plot: Visualize relative importance of variables*

Description

Importance is currently based on the PRISM filter model. For elastic net (filter_glmnet). variables with non-zero coefficients are shown. For random forest variable importance (filter_ranger), variables are sorted by their p-values, and "top_n" will show only the "top_n" most importance variables (based on p-values).

Usage

```
plot_importance(object, top_n = NULL, ...)
```

Arguments

object	PRISM object
top_n	Show top_n variables only, default=NULL (show all)
...	Additional arguments (currently ignored).

Value

Plot (ggplot2) object

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=X)
summary(res2)
```

`predict.PRISM` *PRISM: Patient Response Identifier for Stratified Medicine (Predictions)*

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, X) ) # all #
summary( predict(res0, X, type="ple") )
summary( predict(res0, X, type="submod") )
```

---

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)
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)
```

PRISM *PRISM: Patient Response Identifier for Stratified Medicine*

Description

PRISM algorithm. Given a data-set of (Y, A, X) (Outcome, treatment, covariates), the PRISM identifies potential subgroups along with point-estimate and variability metrics; with and without resampling (bootstrap or cross-validation based). 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 = NULL, calibrate = FALSE,
  alpha.mat = NULL, 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. (Defaults support binary treatment, either numeric or factor). If A=NULL, 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 "gaussion" (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 is "ple_ranger". For continuous/binomial outcome data, this fits treatment specific random forest models. For survival outcome data, this fits a single forest, with expanded covariate space (A, X, X*A). (treatment-specific random forest models). "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.
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 "Bootstrap", "Permutation", and "CV" (cross-validation). Default=NULL (No resampling).
stratify	Stratified resampling (Default=TRUE)

R	Number of resamples (default=NULL; R=100 for Permutation/Bootstrap and R=5 for CV)
calibrate	Bootstrap calibration for nominal alpha (Loh et al 2016). Default=FALSE. For TRUE, outputs the calibrated alpha level and calibrated CIs for the overall population and subgroups. Not applicable for permutation/CV resampling.
alpha.mat	Grid of alpha values for calibration. Default=NULL, which uses seq(alpha/1000,alpha,by=0.005) for alpha_ovrl/alpha_s.
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)

Details

PRISM is a general framework with five key steps:

0. Estimand: Determine the question of interest (ex: mean treatment difference)

1. Filter: Reduce covariate space by removing noise covariates. Options include elastic net (filter_glmnet) and random forest variable importance (filter_ranger).

2. Patient-Level Estimates (ple): Estimate counterfactual patient-level quantities, for example, the individual treatment effect, E(Y|A=1)-E(Y|A=0). Options include: treatment-specific or virtual twins (Y~A+X+A*X) through random forest (ple_ranger, ple_rfsrc), elastic net (ple_glmnet), BART (ple_bart) and causal forest (ple_causal_forest).

3. Subgroup Model (submod): Partition the data into subsets or subgroups of patients. Options include: conditional inference trees (observed outcome or individual treatment effect/PLE; submod_ctree), MOB GLM (submod_glmtree), MOB OLS (submod_lmtree), optimal treatment regimes (submod_otr), rpart (submod_rpart), and MOB Weibull (submod_weibull).

4. Parameter Estimation (param): For the overall population and the discovered subgroups (if any), obtain point-estimates and variability metrics. Options include: cox regression (param_cox), double robust estimator (param_dr), linear regression (param_lm), average of patient-level estimates (param_ple), and restricted mean survival time (param_rmst).

Steps 1-4 also support user-specific models. If treatment is provided (A!=NULL), the default settings are as follows:

Y is continuous (family="gaussian"): Elastic Net Filter ==> Treatment-Specific random forest models ==> MOB (OLS) ==> Average of patient-level estimates (param_ple)

Y is binary (family="binomial"): Elastic Net Filter ==> Treatment-Specific random forest models ==> MOB (GLM) ==> Average of patient-level estimates (param_ple)

Y is right-censored (family="survival"): Elastic Net Filter ==> Virtual twin survival random forest models ==> MOB (Weibull) ==> Cox regression (param_cox)

If treatment is not provided (A=NULL), the default settings are as follows:

Y is continuous (family="gaussian"): Elastic Net Filter ==> Random Forest ==> ctree ==> linear regression

Y is binary (family="binomial"): Elastic Net Filter ==> Random Forest ==> ctree ==> linear regression

Y is right-censored (family="survival"): Elastic Net Filter ==> Survival Random Forest ==> ctree ==> RMST

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 variablity 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

Friedman, J., Hastie, T. and Tibshirani, R. (2008) Regularization Paths for Generalized Linear Models via Coordinate Descent, https://web.stanford.edu/~hastie/Papers/glmnet.pdf Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010 Vol. 33(1), 1-22 Feb 2010.

Jemielita T, Mehrotra D. PRISM: Patient Response Identifiers for Stratified Medicine. https://arxiv.org/abs/1912.03337

Hothorn T, Hornik K, Zeileis A (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15(3), 651–674.

Wright, M. N. & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. J Stat Softw 77:1-17. https://doi.org/10.18637/jss.v077.i01.

Zeileis A, Hothorn T, Hornik K (2008). Model-Based Recursive Partitioning. Journal of Computational and Graphical Statistics, 17(2), 492–514.

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)

# Without filtering #

res1 = PRISM(Y=Y, A=A, X=X, filter="None" )
summary(res1)
plot(res1)


# Search for Prognostic Only (omit A from function) #

res3 = PRISM(Y=Y, X=X)
summary(res3)
plot(res3)


## With bootstrap (No filtering) ##

library(ggplot2)
  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")


## Examples: Binary Outcome ##

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

# Run Default: filter_glmnet, ple_ranger, submod_glmtree, param_ple #
res0 = PRISM(Y=Y, A=A, X=X)

plot(res0)


# Survival Data ##

  library(survival)
  library(ggplot2)
  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  )

# PRISM: glmnet ==> Random Forest to estimate Treatment-Specific RMST
# ==> MOB (Weibull) ==> Cox for HRs#
res_weib = PRISM(Y=Y, A=A, X=X)
plot(res_weib, type="PLE:waterfall")
plot(res_weib)

# PRISM: glmnet ==> Random Forest to estimate Treatment-Specific RMST
# ==> OTR (CTREE, uses RMST estimates as input) ==> Cox for HRs #
res_otr = PRISM(Y=Y, A=A, X=X)
plot(res_otr)

# PRISM: ENET ==> CTREE ==> Cox; with bootstrap #
res_ctree1 = PRISM(Y=Y, A=A, X=X, ple="None", submod = "submod_ctree",
                   resample="Bootstrap", R=50, verbose.resamp = TRUE)
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")
```

`submod_ctree` *Subgroup Identification: Conditional Inference Trees (ctree)*

Description

Uses the ctree (conditional inference trees through partykit R package) algorithm to identify subgroups (Hothorn, Hornik, Zeileis 2006). Usable for continuous, binary, or survival outcomes. Option to use the observed outcome or PLEs (i.e. individual treatment effect) for subgroup identification.

Usage

```
submod_ctree(Y, A, X, Xtest, mu_train, alpha = 0.05,
  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)
alpha	Significance level for variable selection (default=0.05)
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

References

Hothorn T, Hornik K, Zeileis A (2006). Unbiased Recursive Partitioning: A Conditional Inference Framework. Journal of Computational and Graphical Statistics, 15(3), 651–674.

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_glmtree *Subgroup Identification: Model-based partitioning (glmtree)*

Description

Uses the glmtree (model-based partitioning, glm; through partykit R package) algorithm to identify subgroups (Zeileis, Hothorn, Hornik 2008). Usable for continuous and binary outcomes.

Usage

```
submod_glmtree(Y, A, X, Xtest, mu_train, glm.fam = binomial,
  link = "identity", alpha = 0.05, minsize = floor(dim(X)[1] * 0.1),
  maxdepth = 4, parm = NULL, ...)
```

Arguments

Y	The outcome variable. Must be numeric
A	Treatment variable. (a=1,...A)
X	Covariate space.
Xtest	Test set
mu_train	Patient-level estimates (See PLE_models)
glm.fam	Family for GLM; default=binomial
link	Link function for GLM; default="identity"
alpha	Significance level for variable selection (default=0.05)
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)
parm	Model parameters included in parameter instability tests (default=NULL, all parameters)
...	Any additional parameters, not currently passed through.

Value

Trained lmtree model.

- mod - lmtree model object

References

Zeileis A, Hothorn T, Hornik K (2008). Model-Based Recursive Partitioning. Journal of Computational and Graphical Statistics, 17(2), 492–514.

Examples

```
library(StratifiedMedicine)

## Binomial ##
dat_bin = generate_subgrp_data(family="binomial")
Y = dat_bin$Y
X = dat_bin$X
A = dat_bin$A


res_glmtree1 = submod_glmtree(Y, A, X, Xtest=X)
```

```
res_glmtree2 = submod_glmtree(Y, A, X, Xtest=X, link="logit")
plot(res_glmtree1$mod)
plot(res_glmtree2$mod)
```

---

submod_lmtree *Subgroup Identification: Model-based partitioning (lmtree)*

Description

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

Usage

```
submod_lmtree(Y, A, X, Xtest, mu_train, alpha = 0.05,
  minsize = floor(dim(X)[1] * 0.1), maxdepth = 4, parm = 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)
alpha	Significance level for variable selection (default=0.05)
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)
parm	Model parameters included in parameter instability tests (default=NULL, all parameters)
...	Any additional parameters, not currently passed through.

Value

Trained lmtree model.

- mod - lmtree model object

References

Zeileis A, Hothorn T, Hornik K (2008). Model-Based Recursive Partitioning. Journal of Computational and Graphical Statistics, 17(2), 492–514.

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(PLE>thres)~X with weights=abs(PLE) in ctree. For example, PLE could refer to individual treatment effect, E(Y|A=1,X)-E(Y|A=0, X)

Usage

```
submod_otr(Y, A, X, Xtest, mu_train, alpha = 0.05,
  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)
alpha	Significance level for variable selection (default=0.05)
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

References

Zhao et al. (2012) Estimated individualized treatment rules using outcome weighted learning. Journal of the American Statistical Association, 107(409): 1106-1118.

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$mu_train)
plot(res_otr$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

References

Breiman L, Friedman JH, Olshen RA, and Stone CJ. (1984) Classification and Regression Trees. Wadsworth

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.

- mod - 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, alpha = 0.05,
  minsize = floor(dim(X)[1] * 0.1), maxdepth = 4, parm = 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)
`alpha`	Significance level for variable selection (default=0.05)
`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)
`parm`	Model parameters included in parameter instability tests (default=NULL, all parameters)
`...`	Any additional parameters, not currently passed through.

Value

Trained MOB (Weibull) model.

- mod - MOB (Weibull) model object

References

- Zeileis A, Hothorn T, Hornik K (2008). Model-Based Recursive Partitioning. Journal of Computational and Graphical Statistics, 17(2), 492–514.
- Seibold H, Zeileis A, Hothorn T. Model-based recursive partitioning for subgroup analyses. Int J Biostat, 12 (2016), pp. 45-63

Examples

```
library(StratifiedMedicine)
library(survival)

## 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 ple 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