Package 'metaforest'

May 31, 2018

Type Package

Title Exploring Heterogeneity in Meta-Analysis using Random Forests

Version 0.1.2

Author Caspar J. van Lissa

Maintainer Caspar J. van Lissa <c.j.vanlissa@gmail.com>

Description Conduct random forests-based meta-analysis, obtain partial dependence plots for metaforest and classic meta-analyses, and cross-validate and tune metaforest- and classic meta-analyses in conjunction with the caret package. A requirement of classic meta-analysis is that the studies being aggregated are conceptually similar, and ideally, close replications. However, in many fields, there is substantial heterogeneity between studies on the same topic. Classic meta-analysis lacks the power to assess more than a handful of univariate moderators. MetaForest, by contrast, has substantial power to explore heterogeneity in meta-analysis. It can identify important moderators from a larger set of potential candidates, even with as little as 20 studies (Van Lissa, in preparation). This is an appealing quality, because many meta-analyses have small sample sizes. Moreover, MetaForest yields a measure of variable importance which can be used to identify important moderators, and offers partial prediction plots to explore the shape of the marginal relationship between moderators and effect size.

Depends R (>= 3.4.0), ggplot2, metafor, ranger, mmpf

Imports gtable, grid

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-31 06:38:14 UTC

R topics documented:

`ClusterMF` *Clustered MetaForest analysis for dependent data.*

Description

This function conducts a clustered MetaForest analysis for dependent data. Using clustered random sampling, the dataset is split into two cross-validation samples by study. All dependent effect sizes from each study are thus included in the same cross-validation sample. Then, two random forests are grown on these cross-validation samples, and for each random forest, the other sample is used to calculate prediction error and variable importance (see Janitza, Celik, & Boulesteix, 2016). The `predict.MetaForest` method uses all trees from both forests.

Usage

```
ClusterMF(formula, data, vi = "vi", study = NULL, whichweights = "random",
  num.trees = 500, mtry = NULL, method = "REML", tau2 = NULL, ...)
```

Arguments

`formula`	Formula. Specify a formula for the MetaForest model, for example, `yi ~ .` to predict the outcome `yi` from all moderators in the data.
`data`	Data.frame. Provide a data.frame containing the effect size, moderators, and the variance of the effect size. Defaults to 100.
`vi`	Character. Specify the name of the column in the `data` that contains the variances of the effect sizes. This column will be removed from the data prior to analysis. Defaults to `"vi"`.
`study`	Character. Specify the name of the column in the `data` that contains the study id. This column can be a vector of integers, or a factor. This column will be removed from the data prior to analysis.

whichweights	Character. Indicate what time of weights are required. A random-effects MetaForest is grown by specifying whichweights = "random". A fixed-effects MetaForest is grown by specifying whichweights = "fixed". An unweighted MetaForest is grown by specifying whichweights = "unif". Defaults to "random".
num.trees	Atomic integer. Specify the number of trees in the forest. Defaults to 500.
mtry	Atomic integer. Number of candidate moderators available for each split. Defaults to the square root of the number moderators (rounded down).
method	Character. Specify the method by which to estimate the residual variance. Can be set to one of the following: "DL", "HE", "SJ", "ML", "REML", "EB", "HS", or "GENQ". Default is "REML". See the metafor package for more information.
tau2	Numeric. Specify a predetermined value for the residual heterogeneity. Entering a value here supersedes the estimated tau2 value. Defaults to NULL.
...	Additional arguments are passed directly to ranger. It is recommended not to use additional arguments.

Value

List of length 3. The "forest" element of this list is an object of class "ranger", containing the results of the random forests analysis. The "rma_before" element is an object of class "rma.uni", containing the results of a random-effects meta-analysis on the raw data, without moderators. The "rma_after" element is an object of class "rma.uni", containing the results of a random-effects meta-analysis on the residual heterogeneity, or the difference between the effect sizes predicted by MetaForest and the observed effect sizes.

Examples

```
#Load and clean data from metafor
data <- get(data(dat.bourassa1996))
data <- escalc(measure = "OR", ai = lh.le, bi = lh.re, ci = rh.le, di= rh.re,
               data = data, add = 1/2, to = "all")
data$mage[is.na(data$mage)] <- median(data$mage, na.rm = TRUE)
data[c(5:8)] <- lapply(data[c(5:8)], factor)
data$yi <- as.numeric(data$yi)
mf.cluster.b1996 <- ClusterMF(formula = yi~ selection + investigator +
                              hand_assess + eye_assess + mage +sex,
                              data, study = "sample",
                              whichweights = "unif", num.trees = 300)
#Print MetaForest object
mf.cluster.b1996
#Check convergence plot
plot(mf.cluster.b1996)
#Check summary
summary(mf.cluster.b1996, digits = 4)
#Check variable importance plot
VarImpPlot(mf.cluster.b1996)
#Univariate partial dependence plot
PartialDependence(mf.cluster.b1996, vars = "eye_assess")
#Interpolated partial dependence plot for a bivariate interaction
```

```
PartialDependence(mf.cluster.b1996, vars = c("mage", "eye_assess"), interaction = TRUE)
```

`extract_proximity` *Extract proximity matrix for a MetaForest object.*

Description

Extract proximity matrix for a MetaForest object.

Usage

```
extract_proximity(fit, newdata)
```

Arguments

`fit`	object of class \'MetaForest\'.
`newdata`	new data with the same columns as the data used for `fit`

Value

an n x n matrix where position i, j gives the proportion of times observation i and j are in the same terminal node across all trees.

Examples

`MetaForest` *Conduct a MetaForest analysis to explore heterogeneity in meta-analytic data.*

Description

MetaForest uses a weighted random forest to explore heterogeneity in meta-analytic data. MetaForest is a wrapper for ranger (Wright & Ziegler, 2015). As input, MetaForest takes the study effect sizes and their variances (these can be computed, for example, using the metafor package), as well as the moderators that are to be included in the model. By default, MetaForest uses random-effects weights, and estimates the between-studies variance using a restricted maximum-likelihood estimator. However, it may be beneficial to first conduct an unweighted MetaForest, and then use the estimated residual heterogeneity from this model as the estimate of `tau2` for a random-effects weighted MetaForest.

Usage

```
MetaForest(formula, data, vi = "vi", whichweights = "random",
  num.trees = 500, mtry = NULL, method = "REML", tau2 = NULL, ...)
```

Arguments

formula	Formula. Specify a formula for the MetaForest model, for example, `yi ~ .` to predict the outcome `yi` from all moderators in the data.
data	Data.frame. Provide a data.frame containing the effect size, moderators, and the variance of the effect size. Defaults to 100.
vi	Character. Specify the name of the column in the `data` that contains the variances of the effect sizes. This column will be removed from the data prior to analysis. Defaults to `"vi"`.
whichweights	Character. Indicate what time of weights are required. A random-effects MetaForest is grown by specifying `whichweights = "random"`. A fixed-effects MetaForest is grown by specifying `whichweights = "fixed"`. An unweighted MetaForest is grown by specifying `whichweights = "unif"`. Defaults to `"random"`.
num.trees	Atomic integer. Specify the number of trees in the forest. Defaults to 500.
mtry	Atomic integer. Number of candidate moderators available for each split. Defaults to the square root of the number moderators (rounded down).
method	Character. Specify the method by which to estimate the residual variance. Can be set to one of the following: "DL", "HE", "SJ", "ML", "REML", "EB", "HS", or "GENQ". Default is "REML". See the metafor package for more information about these estimators.
tau2	Numeric. Specify a predetermined value for the residual heterogeneity. Entering a value here supersedes the estimated tau2 value. Defaults to NULL.
...	Additional arguments are passed directly to ranger. It is recommended not to use additional arguments.

Value

List of length 3. The "forest" element of this list is an object of class "ranger", containing the results of the random forests analysis. The "rma_before" element is an object of class "rma.uni", containing the results of a random-effects meta-analysis on the raw data, without moderators. The "rma_after" element is an object of class "rma.uni", containing the results of a random-effects meta-analysis on the residual heterogeneity, or the difference between the effect sizes predicted by MetaForest and the observed effect sizes.

Examples

```
#Example 1:
#Simulate data with a univariate linear model
set.seed(42)
data <- SimulateSMD()
#Conduct unweighted MetaForest analysis
mf.unif <- MetaForest(formula = yi ~ ., data = data$training,
                      whichweights = "unif", method = "DL")
#Print model
mf.unif
#Conduct random-effects weighted MetaForest analysis
mf.random <- MetaForest(formula = yi ~ ., data = data$training,
                        whichweights = "random", method = "DL",
```

```
                        tau2 = 0.0116)
#Print summary
summary(mf.random)

#Example 2: Real data from metafor
#Load and clean data
data <- dat.bangertdrowns2004
data[, c(4:12)] <- apply(data[ , c(4:12)], 2, function(x){
  x[is.na(x)] <- median(x, na.rm = TRUE)
  x})
data$subject <- factor(data$subject)
data$yi <- as.numeric(data$yi)
#Conduct MetaForest analysis
mf.bd2004 <- MetaForest(formula = yi~ grade + length + minutes + wic+
                                meta, data, whichweights = "unif")
#Print MetaForest object
mf.bd2004
#Check convergence plot
plot(mf.bd2004)
#Check summary
summary(mf.bd2004, digits = 4)
#Examine variable importance plot
VarImpPlot(mf.bd2004)
```

`ModelInfo_mf` *Returns a MetaForest ModelInfo list for use with caret*

Description

This function allows users to rely on the powerful `caret` package for cross-validating and tuning a MetaForest analysis. Methods for MetaForest are not included in the caret package, because the interface of caret is not entirely compatible with MetaForest's model call. Specifically, MetaForest is not compatible with the `train` methods for classes 'formula' or 'recipe', because the variance of the effect size must be a column of the training data x. The name of this column is specified using the argument 'vi'.

Usage

```
ModelInfo_mf()
```

Details

To train a clustered MetaForest (`clusterMF`), simply provide the optional argument 'study' to the train function, to specify the study ID. This should again refer to a column of x.

When training a clustered MetaForest, make sure to use 'index = groupKFold(your_study_id_variable, k = 10))' in traincontrol, to sample by study ID when creating cross-validation partitions; otherwise the testing error will be positively biased.

Value

ModelInfo list of length 17.

Examples

```
## Not run:
# Prepare data
data <- dat.bangertdrowns2004
data[, c(4:12)] <- apply(data[ , c(4:12)], 2, function(x){
  x[is.na(x)] <- median(x, na.rm = TRUE)
  x})
data$subject <- factor(data$subject)
data$yi <- as.numeric(data$yi)
# Load caret
library(caret)
set.seed(999)
# Specify the resampling method as 10-fold CV
fit_control <- trainControl(method = "cv", number = 10)
cv_mf_fit <- train(y = data$yi, x = data[,c(3:13, 16)],
                   method = ModelInfo_mf(), trControl = fit_control)


# Cross-validated clustered MetaForest
data <- get(data(dat.bourassa1996))
data <- escalc(measure = "OR", ai = lh.le, bi = lh.re, ci = rh.le, di= rh.re,
               data = data, add = 1/2, to = "all")
data$mage[is.na(data$mage)] <- median(data$mage, na.rm = TRUE)
data[c(5:8)] <- lapply(data[c(5:8)], factor)
data$yi <- as.numeric(data$yi)
# Set up 10-fold grouped CV
fit_control <- trainControl(method = "cv", index = groupKFold(data$sample,
                            k = 10))
# Set up a custom tuning grid for the three tuning parameters of MetaForest
rf_grid <- expand.grid(whichweights = c("random", "fixed", "unif"),
                       mtry = c(2, 4, 6),
                       min.node.size = c(2, 4, 6))
# Train the model
cv.mf.cluster <- train(y = data$yi, x = data[, c("selection", "investigator",
                                                 "hand_assess", "eye_assess",
                                                 "mage", "sex", "vi",
                                                 "sample")],
                       study = "sample", method = ModelInfo_mf(),
                       trControl = fit_control,
                       tuneGrid = rf_grid)

## End(Not run)
```

`ModelInfo_rma` *Returns an rma ModelInfo list for use with caret*

Description

This function allows users to rely on the powerful `caret` package for cross-validating and tuning a rma analysis. Methods for rma are not included in the caret package, because the interface of caret is not entirely compatible with rma's model call. Specifically, rma is not compatible with the `train` methods for classes 'formula' or 'recipe'. The variance of the effect sizes can be passed to the 'weights' parameter of `train`.

Usage

```
ModelInfo_rma()
```

Details

When using clustered data (effect sizes within studies), make sure to use 'index = groupKFold(your_study_id_variable, k = 10))' in traincontrol, to sample by study ID when creating cross-validation partitions; otherwise the testing error will be positively biased.

Value

ModelInfo list of length 13.

Examples

```
## Not run:
# Prepare data
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat$yi <- as.numeric(dat$yi)
dat$alloc <- factor(dat$alloc)
# Run rma
rma.model <- rma(y = dat$yi, mods = dat[, c("ablat", "year")], vi = dat$vi)
# R^2 is estimated to be .64
rma.model$R2
# Now, use cross-validation to see how well this model generalizes
# Leave-one-out cross-validation is more appropriate than 10-fold cv because
# the sample size is very small
fit_control <- trainControl(method = "LOOCV")
# Train the model without tuning, because rma has no tuning parameters
cv.mf.cluster <- train(y = dat$yi, x = dat[, c("ablat", "year")],
                       weights = dat$vi,
                       method = ModelInfo_rma(),
                       trControl = fit_control)
# Cross-validated R^2 is .08, suggesting substantial overfitting of the
# original rma model
cv.mf.cluster$results$Rsquared

## End(Not run)
```

PartialDependence *PartialDependence: Partial dependence plots*

Description

Partial dependence plots

Usage

```
PartialDependence(x, vars = NULL, interaction = FALSE, pi = NULL,
  rawdata = FALSE, resolution = NULL, ...)
```

Arguments

x	Model object.
vars	Character vector containing the moderator names for which to plot partial dependence plots. If empty, all moderators are plotted.
interaction	Logical, indicating whether a bivariate interaction should be plotted, using a heatmap. Only valid when the number of vars is 2.
pi	Numeric (0-1). What percentile interval should be plotted for the partial dependence predictions? Defaults to NULL. To obtain a 95% interval, set to .95.
rawdata	Logical, indicating whether to plot weighted raw data. Defaults to FALSE. Uses the same weights as the model object passed to the x argument.
resolution	Integer vector of length two, giving the resolution of the partial predictions. The first element indicates the resolution of the partial predictions; for Monte-Carlo integration, the second element gives the number of rows of the data to be sampled without replacement when averaging over values of the other predictors.
...	Additional arguments to be passed to marginalPrediction.

Details

Plots partial dependence plots (predicted effect size as a function of the value of each predictor variable) for a MetaForest- or rma model object. For rma models, it is advisable to mean-center numeric predictors, and to not include interaction effects, except when the rma model is bivariate, and the interaction argument is set to TRUE.

Value

A ggplot object.

Examples

```
# Partial dependence plot for MetaForest() model:
set.seed(42)
data <- SimulateSMD(k_train = 100, model = es * x[, 1] + es * x[, 2] + es *
                                          x[, 1] * x[, 2])$training
data$X2 <- cut(data$X2, breaks = 2, labels = c("Low", "High"))
mf.random <- MetaForest(formula = yi ~ ., data = data,
                        whichweights = "random", method = "DL",
                        tau2 = 0.2450)
# Examine univariate partial dependence plot for all variables in the model:
PartialDependence(mf.random)
## Not run:
# Examine bivariate partial dependence plot the interaction between X1 and X2:
PartialDependence(mf.random, vars = c("X1", "X2"), interaction = TRUE)

# Partial dependence plot for metafor rma() model:
dat <- escalc(measure="RR", ai=tpos, bi=tneg, ci=cpos, di=cneg, data=dat.bcg)
dat$yi <- as.numeric(dat$yi)
dat$alloc <- factor(dat$alloc)
dat$ablat_d <- cut(dat$ablat, breaks = 2, labels = c("low", "high"))
# Demonstrate partial dependence plot for a bivariate interaction
rma.model.int <- rma(yi, vi, mods=cbind(ablat, tpos),
                     data=dat, method="REML")
PartialDependence(rma.model.int, rawdata = TRUE, pi = .95,
                  interaction = TRUE)
# Compare partial dependence for metaforest and rma
dat2 <- dat
dat2[3:7] <- lapply(dat2[3:7],
                    function(x){as.numeric(scale(x, scale = FALSE))})
mf.model.all <- MetaForest(yi ~ ., dat2[, c(3:11)])
rma.model.all <- rma(dat$yi, dat2$vi,
                  mods = model.matrix(yi~., dat2[, c(3:10)])[, -1],
                  method="REML")
PartialDependence(mf.model.all, rawdata = TRUE, pi = .95)
PartialDependence(rma.model.all, rawdata = TRUE, pi = .95)

## End(Not run)
```

`plot.MetaForest` *Plots cumulative MSE for a MetaForest object.*

Description

Plots cumulative MSE for a MetaForest object.

Usage

```
## S3 method for class 'MetaForest'
plot(x, y, ...)
```

Arguments

x	MetaForest object.
y	not used for plot.MetaForest
...	Arguments to be passed to methods, not used for plot.MetaForest

Value

A ggplot object, visualizing the number of trees on the x-axis, and the cumulative mean of the MSE of that number of trees on the y-axis. As a visual aid to assess convergence, a dashed gray line is plotted at the median cumulative MSE value.

Examples

---

predict.MetaForest *MetaForest prediction*

Description

MetaForest prediction

Usage

```
## S3 method for class 'MetaForest'
predict(object, data = NULL, type = "response", ...)
```

Arguments

object	MetaForest object.
data	New test data of class data.frame.
type	Type of prediction. One of 'response', 'se', 'terminalNodes' with default 'response'. See below for details.
...	further arguments passed to or from other methods.

Value

Object of class MetaForest.prediction with elements

predictions	Predicted classes/values (only for classification and regression)
num.trees	Number of trees.
num.independent.variables	Number of independent variables.
treetype	Type of forest/tree. Classification, regression or survival.
num.samples	Number of samples.

See Also

ranger

Examples

```
set.seed(56)
data <- SimulateSMD(k_train = 100, model = es * x[,1] * x[,2])
#Conduct fixed-effects MetaForest analysis
mf.fixed <- MetaForest(formula = yi ~ ., data = data$training,
                       whichweights = "fixed", method = "DL")
predicted <- predict(mf.fixed, data = data$testing)$predictions
r2_cv <- sum((predicted - mean(data$training$yi)) ^ 2)/
         sum((data$testing$yi - mean(data$training$yi)) ^ 2)
```

---

print.MetaForest *Prints MetaForest object.*

Description

Prints MetaForest object.

Usage

```
## S3 method for class 'MetaForest'
print(x, digits = NULL, ...)
```

Arguments

x	an object used to select a method.
digits	minimal number of significant digits, see print.default.
...	further arguments passed to or from other methods.

Examples

print.summary.MetaForest *Prints summary.MetaForest object.*

Description

Prints summary.MetaForest object.

Usage

```
## S3 method for class 'summary.MetaForest'
print(x, digits, ...)
```

Arguments

x	an object used to select a method.
digits	minimal number of significant digits, see print.default.
...	further arguments passed to or from other methods.

Examples

SimulateSMD *Simulates a meta-analytic dataset*

Description

This function simulates a meta-analytic dataset based on the random-effects model. The simulated effect size is Hedges' G, an estimator of the Standardized Mean Difference (Hedges, 1981; Li, Dusseldorp, & Meulman, 2017). The functional form of the model can be specified, and moderators can be either normally distributed or Bernoulli-distributed. See Van Lissa, in preparation, for a detailed explanation of the simulation procedure.

Usage

```
SimulateSMD(k_train = 20, k_test = 100, mean_n = 40, es = 0.5,
  tau2 = 0.04, moderators = 5, distribution = "normal", model = es * x[,
  1])
```

Arguments

k_train	Atomic integer. The number of studies in the training dataset. Defaults to 20.
k_test	Atomic integer. The number of studies in the testing dataset. Defaults to 100.
mean_n	Atomic integer. The mean sample size of each simulated study in the meta-analytic dataset. Defaults to 40. For each simulated study, the sample size n is randomly drawn from a normal distribution with mean mean_n, and sd mean_n/3.
es	Atomic numeric vector. The effect size, also known as beta, used in the model statement. Defaults to .5.
tau2	Atomic numeric vector. The residual heterogeneity. For a range of realistic values encountered in psychological research, see Van Erp, Verhagen, Grasman, & Wagenmakers, 2017. Defaults to 0.04.
moderators	Atomic integer. The number of moderators to simulate for each study. Make sure that the number of moderators to be simulated is at least as large as the number of moderators referred to in the model parameter. Internally, the matrix of moderators is referred to as "x". Defaults to 5.
distribution	Atomic character. The distribution of the moderators. Can be set to either "normal" or "bernoulli". Defaults to "normal".
model	Expression. An expression to specify the model from which to simulate the mean true effect size, mu. This formula may use the terms "es" (referring to the es parameter of the call to SimulateSMD), and "x[,]" (referring to the matrix of moderators, x). Thus, to specify that the mean effect size, mu, is a function of the effect size and the first moderator, one would pass the value `model = es * x[ , 1]`. Defaults to es * x[, 1].

Value

List of length 4. The "training" element of this list is a data.frame with k_train rows. The columns are the variance of the effect size, vi; the effect size, yi, and the moderators, X. The "testing" element of this list is a data.frame with k_test rows. The columns are the effect size, yi, and the moderators, X. The "housekeeping" element of this list is a data.frame with k_train + k_test rows. The columns are n, the sample size n for each simulated study; mu_i, the mean true effect size for each simulated study; and theta_i, the true effect size for each simulated study.

Examples

```
set.seed(8)
SimulateSMD()
SimulateSMD(k_train = 50, distribution = "bernoulli")
SimulateSMD(distribution = "bernoulli", model = es * x[ ,1] * x[ ,2])
```

summary.MetaForest *Summarize MetaForest object.*

Description

Summarize MetaForest object.

Usage

```
## S3 method for class 'MetaForest'
summary(object, ..., digits = 2)
```

Arguments

object	an object for which a summary is desired.
...	additional arguments affecting the summary produced.
digits	the number of digits desired, defaults to 2.

Examples

VarImpPlot *Plots variable importance for a MetaForest object.*

Description

Plots variable importance for a MetaForest object.

Usage

```
VarImpPlot(mf, n.var = 30, sort = TRUE)
```

Arguments

mf	MetaForest object.
n.var	Number of moderators to plot.
sort	Should the moderators be sorted from most to least important?

Value

A ggplot object.

Examples

```
set.seed(42)
data <- SimulateSMD()
mf.random <- MetaForest(formula = yi ~ ., data = data$training,
                        whichweights = "random", method = "DL",
                        tau2 = 0.0116)
VarImpPlot(mf.random)
VarImpPlot(mf.random, n.var = 2)
VarImpPlot(mf.random, sort = FALSE)
```

`WeightedScatter` *Plots weighted scatterplots for meta-analytic data. Can plot effect size as a function of either continuous (numeric, integer) or categorical (factor, character) predictors.*

Description

Plots weighted scatterplots for meta-analytic data. Can plot effect size as a function of either continuous (numeric, integer) or categorical (factor, character) predictors.

Usage

```
WeightedScatter(data, yi = "yi", vi = "vi", vars = NULL, tau2 = NULL,
  summarize = TRUE)
```

Arguments

`data`	A data.frame.
`yi`	Character. The name of the column in `data` that contains the meta-analysis effect sizes. Defaults to `"yi"`.
`vi`	Character. The name of the column in the `data` that contains the variances of the effect sizes. Defaults to `"vi"`. By default, `vi` is used to calculate fixed-effects weights, because fixed effects weights summarize the data set at hand, rather than generalizing to the population.
`vars`	Character vector containing the names of specific moderator variables to plot. When set to `NULL`, the default, all moderators are plotted.
`tau2`	Numeric. Provide an optional value for tau2. If this value is provided, random-effects weights will be used instead of fixed-effects weights.
`summarize`	Logical. Should summary stats be displayed? Defaults to FALSE. If TRUE, a smooth trend line is displayed for continuous variables, using [stats::loess()] for less than 1000 observations, and [mgcv::gam()] for larger datasets. For categorical variables, box-and-whiskers plots are displayed. Outliers are omitted, because the raw data fulfill this function.

Value

A ggplot object.

Examples

```
set.seed(42)
data <- SimulateSMD(k_train = 100, model = es * x[, 1] + es * x[, 2] + es *
                      x[, 1] * x[, 2])$training
data$X2 <- cut(data$X2, breaks = 2, labels = c("Low", "High"))
data$X3 <- cut(data$X3, breaks = 2, labels = c("Small", "Big"))
WeightedScatter(data, summarize = FALSE)
WeightedScatter(data, vars = c("X3"))
WeightedScatter(data, vars = c("X1", "X3"))
WeightedScatter(data, tau2 = .04)
```

Index