

# Package ‘fuzzyforest’

May 8, 2026

**Title** Fuzzy Forests

**Version** 1.0.8

**Description** Fuzzy forests, a new algorithm based on random forests, is designed to reduce the bias seen in random forest feature selection caused by the presence of correlated features. Fuzzy forests uses recursive feature elimination random forests to select features from separate blocks of correlated features where the correlation within each block of features is high and the correlation between blocks of features is low. One final random forest is fit using the surviving features. This package fits random forests using the 'randomForest' package and allows for easy use of 'WGCNA' to split features into distinct blocks. See D. Conn, Ngun, T., C. Ramirez, and G. Li (2019) <[doi:10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)> for further details.

**Depends** R (>= 3.2.1)

**License** GPL-3

**LazyData** true

**Imports** randomForest, foreach, doParallel, parallel, ggplot2, mvtnorm

**Suggests** WGCNA, testthat

**RoxygenNote** 6.1.1

**NeedsCompilation** no

**Author** Daniel Conn [aut, cre],  
Tuck Ngun [aut],  
Christina M. Ramirez [aut]

**Maintainer** Daniel Conn <[djconn17@gmail.com](mailto:djconn17@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-03-25 16:40:16 UTC

## Contents

ctg . . . . . 2

example_ff . . . . .	2
ff . . . . .	3
ff.formula . . . . .	5
fuzzyforest . . . . .	8
fuzzy_forest . . . . .	8
iterative_RF . . . . .	9
Liver_Expr . . . . .	10
modplot . . . . .	10
multi_class_lr . . . . .	11
predict.fuzzy_forest . . . . .	12
print.fuzzy_forest . . . . .	13
screen_control . . . . .	14
select_control . . . . .	15
select_RF . . . . .	16
wff . . . . .	17
wff.formula . . . . .	19
WGCNA_control . . . . .	21

## Index 22

---

ctg	<i>Cardiotocography Data Set</i>
-----	----------------------------------

---

### Description

A data set containing measurements of fetal heart rate and uterine contraction from cardiotocograms. This data set was obtained from the [UCI machine learning repository](<https://archive.ics.uci.edu/ml/index.html>) For our examples we extract a random sub sample of 100 observations.

### Usage

```
data(ctg)
```

### Format

A data frame with 100 rows and 21.

---

example_ff	<i>Fuzzy Forest Example</i>
------------	-----------------------------

---

### Description

An example of a fuzzy\_forest object derived from fitting fuzzy forests on the ctg data set. The source code used to produce example\_ff can be seen in the vignette "fuzzyforest\_introduction".

### Format

.RData

**Description**

Fits the fuzzy forests algorithm. Note that a formula interface for fuzzy forests also exists: [ff.formula](#).

**Usage**

```
## Default S3 method:
ff(X, y, Z = NULL, module_membership,
   screen_params = screen_control(min_ntree = 500),
   select_params = select_control(min_ntree = 500), final_ntree = 5000,
   num_processors = 1, nodesize, test_features = NULL, test_y = NULL,
   ...)

ff(X, ...)
```

**Arguments**

X	A data.frame. Each column corresponds to a feature vectors.
y	Response vector. For classification, y should be a factor. For regression, y should be numeric.
Z	A data.frame. Additional features that are not to be screened out at the screening step.
module_membership	A character vector giving the module membership of each feature.
screen_params	Parameters for screening step of fuzzy forests. See <a href="#">screen_control</a> for details. screen_params is an object of type screen_control.
select_params	Parameters for selection step of fuzzy forests. See <a href="#">select_control</a> for details. select_params is an object of type select_control.
final_ntree	Number of trees grown in the final random forest. This random forest contains all selected features.
num_processors	Number of processors used to fit random forests.
nodesize	Minimum terminal nodesize. 1 if classification. 5 if regression. If the sample size is very large, the trees will be grown extremely deep. This may lead to issues with memory usage and may lead to significant increases in the time it takes the algorithm to run. In this case, it may be useful to increase nodesize.
test_features	A data.frame containing features from a test set. The data.frame should contain the features in both X and Z.
test_y	The responses for the test set.
...	Additional arguments currently not used.

**Value**

An object of type `fuzzy_forest`. This object is a list containing useful output of fuzzy forests. In particular it contains a data.frame with a list of selected the features. It also includes a random forest fit using the selected features.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

**References**

- Conn, D., Ngun, T., Ramirez C.M., Li, G. (2019). "Fuzzy Forests: Extending Random Forest Feature Selection for Correlated, High-Dimensional Data." *Journal of Statistical Software*, **91**(9). doi: [10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)
- Breiman, L. (2001). "Random Forests." *Machine Learning*, **45**(1), 5-32. doi: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)
- Zhang, B. and Horvath, S. (2005). "A General Framework for Weighted Gene Co-Expression Network Analysis." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). doi: [10.2202/15446115.1128](https://doi.org/10.2202/15446115.1128)

**See Also**

`ff.formula`, `print.fuzzy_forest`, `predict.fuzzy_forest`, `modplot`

**Examples**

```
#ff requires that the partition of the covariates be previously determined.
#ff is also handy if the user wants to test out multiple settings of WGCNA
#prior to running fuzzy forests.
```

```
library(mvtnorm)
gen_mod <- function(n, p, corr) {
  sigma <- matrix(corr, nrow=p, ncol=p)
  diag(sigma) <- 1
  X <- rmvnorm(n, sigma=sigma)
  return(X)
}

gen_X <- function(n, mod_sizes, corr){
  m <- length(mod_sizes)
  X_list <- vector("list", length = m)
  for(i in 1:m){
    X_list[[i]] <- gen_mod(n, mod_sizes[i], corr[i])
  }
  X <- do.call("cbind", X_list)
  return(X)
}

err_sd <- .5
n <- 500
mod_sizes <- rep(25, 4)
```

```
corr <- rep(.8, 4)
X <- gen_X(n, mod_sizes, corr)
beta <- rep(0, 100)
beta[c(1:4, 76:79)] <- 5
y <- X%%beta + rnorm(n, sd=err_sd)
X <- as.data.frame(X)

Xtest <- gen_X(n, mod_sizes, corr)
ytest <- Xtest%%beta + rnorm(n, sd=err_sd)
Xtest <- as.data.frame(Xtest)

cdist <- as.dist(1 - cor(X))
hclust_fit <- hclust(cdist, method="ward.D")
groups <- cutree(hclust_fit, k=4)

screen_c <- screen_control(keep_fraction = .25,
                           ntree_factor = 1,
                           min_ntree = 250)
select_c <- select_control(number_selected = 10,
                           ntree_factor = 1,
                           min_ntree = 250)

ff_fit <- ff(X, y, module_membership = groups,
            screen_params = screen_c,
            select_params = select_c,
            final_ntree = 250)
#extract variable importance rankings
vims <- ff_fit$feature_list

#plot results
modplot(ff_fit)

#obtain predicted values for a new test set
preds <- predict(ff_fit, new_data=Xtest)

#estimate test set error
test_err <- sqrt(sum((ytest - preds)^2)/n)
```

---

ff.formula

*Fuzzy forests algorithm*

---

## Description

Implements formula interface for `ff`.

## Usage

```
## S3 method for class 'formula'
ff(formula, data = NULL, module_membership, ...)
```

**Arguments**

formula	Formula object.
data	data used in the analysis.
module_membership	A character vector giving the module membership of each feature.
...	Additional arguments

**Value**

An object of type `fuzzy_forest`. This object is a list containing useful output of fuzzy forests. In particular it contains a `data.frame` with list of selected features. It also includes the random forest fit using the selected features.

**Note**

See `ff` for additional arguments. Note that the matrix, `Z`, of features that do not go through the screening step must be specified separately from the formula. `test_features` and `test_y` are not supported in formula interface. As in the `randomForest` package, for large data sets the formula interface may be substantially slower.

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

**References**

- Conn, D., Ngun, T., Ramirez C.M., Li, G. (2019). "Fuzzy Forests: Extending Random Forest Feature Selection for Correlated, High-Dimensional Data." *Journal of Statistical Software*, **91**(9). doi: [10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)
- Breiman, L. (2001). "Random Forests." *Machine Learning*, **45**(1), 5-32. doi: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)
- Zhang, B. and Horvath, S. (2005). "A General Framework for Weighted Gene Co-Expression Network Analysis." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). doi: [10.2202/15446115.1128](https://doi.org/10.2202/15446115.1128)

**See Also**

`ff`, `print.fuzzy_forest`, `predict.fuzzy_forest`, `modplot`

**Examples**

```
#ff requires that the partition of the covariates be previously determined.
#ff is also handy if the user wants to test out multiple settings of WGCNA
#prior to running fuzzy forests.
library(mvtnorm)
gen_mod <- function(n, p, corr) {
  sigma <- matrix(corr, nrow=p, ncol=p)
  diag(sigma) <- 1
  X <- rmvnorm(n, sigma=sigma)
  return(X)
}
```

```

gen_X <- function(n, mod_sizes, corr){
  m <- length(mod_sizes)
  X_list <- vector("list", length = m)
  for(i in 1:m){
    X_list[[i]] <- gen_mod(n, mod_sizes[i], corr[i])
  }
  X <- do.call("cbind", X_list)
  return(X)
}

err_sd <- .5
n <- 500
mod_sizes <- rep(25, 4)
corr <- rep(.8, 4)
X <- gen_X(n, mod_sizes, corr)
beta <- rep(0, 100)
beta[c(1:4, 76:79)] <- 5
y <- X%%beta + rnorm(n, sd=err_sd)
X <- as.data.frame(X)
dat <- as.data.frame(cbind(y, X))

Xtest <- gen_X(n, mod_sizes, corr)
ytest <- Xtest%%beta + rnorm(n, sd=err_sd)
Xtest <- as.data.frame(Xtest)

cdist <- as.dist(1 - cor(X))
hclust_fit <- hclust(cdist, method="ward.D")
groups <- cutree(hclust_fit, k=4)

screen_c <- screen_control(keep_fraction = .25,
                          ntree_factor = 1,
                          min_ntree = 250)
select_c <- select_control(number_selected = 10,
                          ntree_factor = 1,
                          min_ntree = 250)

ff_fit <- ff(y ~ ., data=dat,
            module_membership = groups,
            screen_params = screen_c,
            select_params = select_c,
            final_ntree = 250)
#extract variable importance rankings
vims <- ff_fit$feature_list

#plot results
modplot(ff_fit)

#obtain predicted values for a new test set
preds <- predict(ff_fit, new_data=Xtest)

#estimate test set error
test_err <- sqrt(sum((ytest - preds)^2)/n)

```

---

 fuzzyforest

*fuzzyforest: an implementation of the fuzzy forest algorithm in R.*


---

### Description

This package implements fuzzy forests and integrates the fuzzy forests algorithm with the package, **WGCNA**.

### Note

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

---

 fuzzy\_forest

*Fuzzy Forest Object*


---

### Description

Fuzzy forests returns an object of type fuzzyforest.

### Usage

```
fuzzy_forest(feature_list, final_rf, module_membership,
             WGCNA_object = NULL, survivor_list, selection_list)
```

### Arguments

`feature_list` List of selected features along with variable importance measures.

`final_rf` A final random forest fit using the features selected by fuzzy forests.

`module_membership` Module membership of each feature.

`WGCNA_object` If applicable, output of WGCNA analysis.

`survivor_list` List of features that have survived screening step.

`selection_list` List of features retained at each iteration of selection step.

### Value

An object of type `fuzzy_forest`.

### Note

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

---

`iterative_RF`*Fits iterative random forest algorithm.*

---

**Description**

Fits iterative random forest algorithm. Returns data.frame with variable importances and top rated features. For now this is an internal function that I've used to explore how recursive feature elimination works in simulations. It may be exported at a later time.

**Usage**

```
iterative_RF(X, y, drop_fraction, keep_fraction, mtry_factor,  
            ntree_factor = 10, min_ntree = 5000, num_processors = 1, nodesize)
```

**Arguments**

<code>X</code>	A data.frame. Each column corresponds to a feature vectors.
<code>y</code>	Response vector.
<code>drop_fraction</code>	A number between 0 and 1. Percentage of features dropped at each iteration.
<code>keep_fraction</code>	A number between 0 and 1. Proportion features from each module to retain at screening step.
<code>mtry_factor</code>	A positive number. Mtry for each random forest is set to $\text{ceiling}(\sqrt{p}\text{mtry\_factor})$ where p is the current number of features.
<code>ntree_factor</code>	A number greater than 1. ntree for each random is ntree_factor times the number of features. For each random forest, ntree is set to $\max(\text{min\_ntree}, \text{ntree\_factor} * p)$ .
<code>min_ntree</code>	Minimum number of trees grown in each random forest.
<code>num_processors</code>	Number of processors used to fit random forests.
<code>nodesize</code>	Minimum nodesize.

**Value**

A data.frame with the top ranked features.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

---

 Liver\_Expr

*Liver Expression Data from Female Mice*


---

### Description

A data set containing gene expression levels in liver tissue from female mice. This data set is a subset of the liver expression data set from the WGCNA tutorial <https://horvath.genetics.ucla.edu/html/CoexpressionNetwork/Rpackages/WGCNA/Tutorials/>. The tutorial contains further information about the data set as well as extensive examples of WGCNA.

### Usage

```
data(Liver_Expr)
```

### Format

A data frame with 66 rows and 3601

### Details

- The first column contains weight (g) for the 66 mice.
- The other 3600 columns contain the liver expression levels.

---

 modplot

*Plots relative importance of modules.*


---

### Description

The plot is designed to depict the size of each module and what percentage of selected features fall into each module. In particular, it is easy to determine which module is over-represented in the group of selected features.

### Usage

```
modplot(object, main = NULL, xlab = NULL, ylab = NULL,
        module_labels = NULL)
```

### Arguments

object	A fuzzy_forest object.
main	Title of plot.
xlab	Title for the x axis.
ylab	Title for the y axis.
module_labels	Labels for the modules. A data.frame or character matrix with first column giving the current name of module and second column giving the assigned name of each module.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

**See Also**

[ff](#), [wff](#), [ff.formula](#), [wff.formula](#)

---

multi_class_lr	<i>Multinomial Logistic Regression</i>
----------------	--

---

**Description**

Function to generate multi-class data from a multinomial logistic regression. Assumes there are 5 classes. Only supports two modules for now. Currently this function is used for testing.

**Usage**

```
multi_class_lr(n, mod1_size = 10, mod2_size = 10, rho = 0.8,  
              beta = NULL)
```

**Arguments**

n	Sample size.
mod1_size	Size of first module.
mod2_size	Size of second module.
rho	Correlation of covariates.
beta	A matrix of parameters.

**Value**

list with design matrix X, outcome y, and beta.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

---

predict.fuzzy\_forest *Predict method for fuzzy\_forest object. Obtains predictions from fuzzy forest algorithm.*

---

### Description

Predict method for fuzzy\_forest object. Obtains predictions from fuzzy forest algorithm.

### Usage

```
## S3 method for class 'fuzzy_forest'  
predict(object, new_data, ...)
```

### Arguments

object	A fuzzy_forest object.
new_data	A matrix or data.frame containing new_data. Pay close attention to ensure feature names match between training set and test set data.frame.
...	Additional arguments not in use.

### Value

A vector of predictions

### Note

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

### See Also

[ff](#), [wff](#), [ff.formula](#), [wff.formula](#)

### Examples

```
library(mvtnorm)  
gen_mod <- function(n, p, corr) {  
  sigma <- matrix(corr, nrow=p, ncol=p)  
  diag(sigma) <- 1  
  X <- rmvnorm(n, sigma=sigma)  
  return(X)  
}  
  
gen_X <- function(n, mod_sizes, corr){  
  m <- length(mod_sizes)  
  X_list <- vector("list", length = m)  
  for(i in 1:m){  
    X_list[[i]] <- gen_mod(n, mod_sizes[i], corr[i])  
  }  
}
```

```

    X <- do.call("cbind", X_list)
    return(X)
  }

  err_sd <- .5
  n <- 500
  mod_sizes <- rep(25, 4)
  corr <- rep(.8, 4)
  X <- gen_X(n, mod_sizes, corr)
  beta <- rep(0, 100)
  beta[c(1:4, 76:79)] <- 5
  y <- X%%beta + rnorm(n, sd=err_sd)
  X <- as.data.frame(X)

  Xtest <- gen_X(n, mod_sizes, corr)
  ytest <- Xtest%%beta + rnorm(n, sd=err_sd)
  Xtest <- as.data.frame(Xtest)

  cdist <- as.dist(1 - cor(X))
  hclust_fit <- hclust(cdist, method="ward.D")
  groups <- cutree(hclust_fit, k=4)

  screen_c <- screen_control(keep_fraction = .25,
                            ntree_factor = 1,
                            min_ntree = 250)
  select_c <- select_control(number_selected = 10,
                             ntree_factor = 1,
                             min_ntree = 250)

  ff_fit <- ff(X, y, module_membership = groups,
              screen_params = screen_c,
              select_params = select_c,
              final_ntree = 250)
  #extract variable importance rankings
  vims <- ff_fit$feature_list

  #plot results
  modplot(ff_fit)

  #obtain predicted values for a new test set
  preds <- predict(ff_fit, new_data=Xtest)

  #estimate test set error
  test_err <- sqrt(sum((ytest - preds)^2)/n)

```

---

print.fuzzy\_forest      *Print fuzzy\_forest object. Prints output from fuzzy forests algorithm.*

---

### Description

Print fuzzy\_forest object. Prints output from fuzzy forests algorithm.

**Usage**

```
## S3 method for class 'fuzzy_forest'
print(x, ...)
```

**Arguments**

x                    A fuzzy\_forest object.  
...                   Additional arguments not in use.

**Value**

data.frame with list of selected features and variable importance measures.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

---

screen\_control                    *Set Parameters for Screening Step of Fuzzy Forests*

---

**Description**

Creates screen\_control object for controlling how feature selection will be carried out on each module.

**Usage**

```
screen_control(drop_fraction = 0.25, keep_fraction = 0.05,
               mtry_factor = 1, min_ntree = 500, ntree_factor = 1)
```

**Arguments**

drop\_fraction    A number between 0 and 1. Percentage of features dropped at each iteration.  
keep\_fraction    A number between 0 and 1. Proportion of features from each module that are retained from screening step.  
mtry\_factor       In the case of regression, mtry is set to  $\text{ceiling}(\sqrt{p} * \text{mtry\_factor})$ . In the case of classification, mtry is set to  $\text{ceiling}((p/3) * \text{mtry\_factor})$ . If either of these numbers is greater than p, mtry is set to p.  
min\_ntree        Minimum number of trees grown in each random forest.  
ntree\_factor     A number greater than 1. ntree for each random forest is ntree\_factor times the number of features. For each random forest, ntree is set to  $\max(\text{min\_ntree}, \text{ntree\_factor} * p)$ .

**Value**

An object of type screen\_control.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

**References**

Conn, D., Ngun, T., Ramirez C.M., Li, G. (2019). "Fuzzy Forests: Extending Random Forest Feature Selection for Correlated, High-Dimensional Data." *Journal of Statistical Software*, **91**(9). doi: [10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)

**Examples**

```
drop_fraction <- .25
keep_fraction <- .1
mtry_factor <- 1
min_ntree <- 5000
ntree_factor <- 5
screen_params <- screen_control(drop_fraction=drop_fraction,
                                keep_fraction=keep_fraction,
                                mtry_factor=mtry_factor,
                                min_ntree=min_ntree,
                                ntree_factor=ntree_factor)
```

---

 select\_control

 Set Parameters for Selection Step of Fuzzy Forests
 

---

**Description**

Creates selection\_control object for controlling how feature selection will be carried out after features from different modules have been combined.

**Usage**

```
select_control(drop_fraction = 0.25, number_selected = 5,
              mtry_factor = 1, min_ntree = 500, ntree_factor = 1)
```

**Arguments**

drop_fraction	A number between 0 and 1. Percentage of features dropped at each iteration.
number_selected	A positive number. Number of features that will be selected by fuzzyforests.
mtry_factor	In the case of regression, mtry is set to $\text{ceiling}(\sqrt{p}) * \text{mtry\_factor}$ . In the case of classification, mtry is set to $\text{ceiling}((p/3) * \text{mtry\_factor})$ . If either of these numbers is greater than p, mtry is set to p.
min_ntree	Minimum number of trees grown in each random forest.
ntree_factor	A number greater than 1. ntree for each random forest is ntree_factor times the number of features. For each random forest, ntree is set to $\max(\text{min\_ntree}, \text{ntree\_factor} * p)$ .

**Value**

An object of type selection\_control.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

**References**

Conn, D., Ngun, T., Ramirez C.M., Li, G. (2019). "Fuzzy Forests: Extending Random Forest Feature Selection for Correlated, High-Dimensional Data." *Journal of Statistical Software*, **91**(9). doi: [10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)

**Examples**

```
drop_fraction <- .25
number_selected <- 10
mtry_factor <- 1
min_ntree <- 5000
ntree_factor <- 5
select_params <- select_control(drop_fraction=drop_fraction,
                               number_selected=number_selected,
                               mtry_factor=mtry_factor,
                               min_ntree=min_ntree,
                               ntree_factor=ntree_factor)
```

---

select\_RF

*Carries out the selection step of fuzzyforest algorithm.*

---

**Description**

Carries out the selection step of fuzzyforest algorithm. Returns data.frame with variable importances and top rated features.

**Usage**

```
select_RF(X, y, drop_fraction, number_selected, mtry_factor, ntree_factor,
          min_ntree, num_processors, nodesize)
```

**Arguments**

X	A data.frame. Each column corresponds to a feature vectors. Could include additional covariates not a part of the original modules.
y	Response vector.
drop_fraction	A number between 0 and 1. Percentage of features dropped at each iteration.
number_selected	Number of features selected by fuzzyforest.

mtry_factor	In the case of regression, mtry is set to $\text{ceiling}(\sqrt{p} * \text{mtry\_factor})$ . In the case of classification, mtry is set to $\text{ceiling}((p/3) * \text{mtry\_factor})$ . If either of these numbers is greater than p, mtry is set to p.
ntree_factor	A number greater than 1. ntree for each random is ntree_factor times the number of features. For each random forest, ntree is set to $\max(\text{min\_ntree}, \text{ntree\_factor} * p)$ .
min_ntree	Minimum number of trees grown in each random forest.
num_processors	Number of processors used to fit random forests.
nodesize	Minimum nodesize

**Value**

A data.frame with the top ranked features.

**Note**

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

---

wff

*WGCNA based fuzzy forest algorithm*


---

**Description**

Fits fuzzy forests using WGCNA to cluster features into distinct modules. Requires installation of WGCNA package. Note that a formula interface for WGCNA based fuzzy forests also exists: [wff.formula](#).

**Usage**

```
## Default S3 method:
wff(X, y, Z = NULL,
    WGCNA_params = WGCNA_control(power = 6),
    screen_params = screen_control(min_ntree = 500),
    select_params = select_control(min_ntree = 500), final_ntree = 5000,
    num_processors = 1, nodesize, test_features = NULL, test_y = NULL,
    ...)

wff(X, ...)
```

**Arguments**

X	A data.frame. Each column corresponds to a feature vector. WGCNA will be used to cluster the features in X. As a result, the features should be all be numeric. Non-numeric features may be input via Z.
y	Response vector. For classification, y should be a factor. For regression, y should be numeric.

Z	Additional features that are not to be screened out at the screening step. WGCNA is not carried out on features in Z.
WGCNA_params	Parameters for WGCNA. See <code>blockwiseModules</code> function from WGCNA and <code>WGCNA_control</code> for details. WGCNA_params is an object of type <code>WGCNA_control</code> .
screen_params	Parameters for screening step of fuzzy forests. See <code>screen_control</code> for details. screen_params is an object of type <code>screen_control</code> .
select_params	Parameters for selection step of fuzzy forests. See <code>select_control</code> for details. select_params is an object of type <code>select_control</code> .
final_ntree	Number of trees grown in the final random forest. This random forest contains all selected features.
num_processors	Number of processors used to fit random forests.
nodesize	Minimum terminal nodesize. 1 if classification. 5 if regression. If the sample size is very large, the trees will be grown extremely deep. This may lead to issues with memory usage and may lead to significant increases in the time it takes the algorithm to run. In this case, it may be useful to increase nodesize.
test_features	A <code>data.frame</code> containing features from a test set. The <code>data.frame</code> should contain the features in both X and Z.
test_y	The responses for the test set.
...	Additional arguments currently not used.

### Value

An object of type `fuzzy_forest`. This object is a list containing useful output of fuzzy forests. In particular it contains a `data.frame` with list of selected features. It also includes the random forest fit using the selected features.

### Note

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

### References

Conn, D., Ngun, T., Ramirez C.M., Li, G. (2019). "Fuzzy Forests: Extending Random Forest Feature Selection for Correlated, High-Dimensional Data." *Journal of Statistical Software*, **91**(9). doi: [10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)

Breiman, L. (2001). "Random Forests." *Machine Learning*, **45**(1), 5-32. doi: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324)

Zhang, B. and Horvath, S. (2005). "A General Framework for Weighted Gene Co-Expression Network Analysis." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). doi: [10.2202/15446115.1128](https://doi.org/10.2202/15446115.1128)

### See Also

`wff.formula`, `print.fuzzy_forest`, `predict.fuzzy_forest`, `modplot`

**Examples**

```

data(ctg)
y <- ctg$NSP
X <- ctg[, 2:22]
WGCNA_params <- WGCNA_control(p = 6, minModuleSize = 1, nThreads = 1)
mtry_factor <- 1; min_ntree <- 500; drop_fraction <- .5; ntree_factor <- 1
screen_params <- screen_control(drop_fraction = drop_fraction,
                               keep_fraction = .25, min_ntree = min_ntree,
                               ntree_factor = ntree_factor,
                               mtry_factor = mtry_factor)
select_params <- select_control(drop_fraction = drop_fraction,
                               number_selected = 5,
                               min_ntree = min_ntree,
                               ntree_factor = ntree_factor,
                               mtry_factor = mtry_factor)

library(WGCNA)
wff_fit <- wff(X, y, WGCNA_params = WGCNA_params,
              screen_params = screen_params,
              select_params = select_params,
              final_ntree = 500)

#extract variable importance rankings
vims <- wff_fit$feature_list

#plot results
modplot(wff_fit)

```

---

wff.formula

*WGCNA based fuzzy forest algorithm*


---

**Description**

Implements formula interface for [wff](#).

**Usage**

```

## S3 method for class 'formula'
wff(formula, data = NULL, ...)

```

**Arguments**

formula	Formula object.
data	data used in the analysis.
...	Additional arguments

**Value**

An object of type `fuzzy_forest`. This object is a list containing useful output of fuzzy forests. In particular it contains a `data.frame` with list of selected features. It also includes the random forest fit using the selected features.

**Note**

See `ff` for additional arguments. Note that the matrix, `Z`, of features that do not go through the screening step must be specified separately from the formula. `test_features` and `test_y` are not supported in formula interface. As in the `randomForest` package, for large data sets the formula interface may be substantially slower.

This work was partially funded by NSF IIS 1251151 and AMFAR 8721SC.

**See Also**

`wff`, `print.fuzzy_forest`, `predict.fuzzy_forest`, `modplot`

**Examples**

```
data(ctg)
y <- ctg$NSP
X <- ctg[, 2:22]
dat <- as.data.frame(cbind(y, X))
WGCNA_params <- WGCNA_control(p = 6, minModuleSize = 1, nThreads = 1)
mtry_factor <- 1; min_ntree <- 500; drop_fraction <- .5; ntree_factor <- 1
screen_params <- screen_control(drop_fraction = drop_fraction,
                                keep_fraction = .25, min_ntree = min_ntree,
                                ntree_factor = ntree_factor,
                                mtry_factor = mtry_factor)
select_params <- select_control(drop_fraction = drop_fraction,
                                number_selected = 5,
                                min_ntree = min_ntree,
                                ntree_factor = ntree_factor,
                                mtry_factor = mtry_factor)

library(WGCNA)
wff_fit <- wff(y ~ ., data=dat,
              WGCNA_params = WGCNA_params,
              screen_params = screen_params,
              select_params = select_params,
              final_ntree = 500)

#extract variable importance rankings
vims <- wff_fit$feature_list

#plot results
modplot(wff_fit)
```

---

`WGCNA_control`*Set Parameters for WGCNA Step of Fuzzy Forests*

---

**Description**

Creates `WGCNA_control` object for controlling WGCNA will be carried out.

**Usage**

```
WGCNA_control(power = 6, ...)
```

**Arguments**

<code>power</code>	Power of adjacency function.
<code>...</code>	Additional arguments. See <code>blockwiseModules</code> from the <code>WGCNA</code> package for details.

**Value**

An object of type `WGCNA_control`.

**Note**

This work was partially funded by NSF IIS 1251151.

**References**

Conn, D., Ngun, T., Ramirez C.M., Li, G. (2019). "Fuzzy Forests: Extending Random Forest Feature Selection for Correlated, High-Dimensional Data." *Journal of Statistical Software*, **91**(9). doi: [10.18637/jss.v091.i09](https://doi.org/10.18637/jss.v091.i09)

Zhang, B. and Horvath, S. (2005). "A General Framework for Weighted Gene Co-Expression Network Analysis." *Statistical Applications in Genetics and Molecular Biology*, **4**(1). doi: [10.2202/15446115.1128](https://doi.org/10.2202/15446115.1128)

**Examples**

```
WGCNA_params <- WGCNA_control(p=7, minModuleSize=30, TOMType = "unsigned",  
                               reassignThreshold = 0, mergeCutHeight = 0.25,  
                               numericLabels = TRUE, pamRespectsDendro = FALSE)
```

# Index

- \* **R**
  - example\_ff, 2
- \* **datasets**
  - ctg, 2
  - Liver\_Expr, 10
- \* **object**
  - example\_ff, 2

ctg, 2

example\_ff, 2

ff, 3, 5, 6, 11, 12, 20

ff.formula, 3, 4, 5, 11, 12

fuzzy\_forest, 4, 6, 8, 18, 20

fuzzyforest, 8

fuzzyforest-package (fuzzyforest), 8

iterative\_RF, 9

Liver\_Expr, 10

modplot, 4, 6, 10, 18, 20

multi\_class\_lr, 11

predict.fuzzy\_forest, 4, 6, 12, 18, 20

print.fuzzy\_forest, 4, 6, 13, 18, 20

screen\_control, 3, 14, 18

select\_control, 3, 15, 18

select\_RF, 16

wff, 11, 12, 17, 19, 20

wff.formula, 11, 12, 17, 18, 19

WGCNA\_control, 18, 21