

Package ‘RFlocalfdr’

July 21, 2025

Title Significance Level for Random Forest Impurity Importance Scores

Version 0.9

Description Sets a significance level for Random Forest MDI (Mean Decrease in Impurity, Gini or sum of squares) variable importance scores, using an empirical Bayes approach.
See Dunne et al. (2022) <[doi:10.1101/2022.04.06.487300](https://doi.org/10.1101/2022.04.06.487300)>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

LazyDataCompression xz

Imports minpack.lm, sn, fitdistrplus, grDevices, graphics, stats,
ranger, randomForest

Depends R (>= 3.5.0)

Suggests rmarkdown, knitr, testthat (>= 3.0.0), RFlocalfdr.data, vita

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Robert Dunne [aut, cre] (ORCID:
<<https://orcid.org/0000-0003-1946-7279>>)

Maintainer Robert Dunne <rob.dunne@csiro.au>

Repository CRAN

Date/Publication 2025-01-30 22:10:10 UTC

Contents

count_variables	2
determine.C	3
determine_cutoff	4
f.fit	6
fit.to.data.set	7

fit.to.data.set.wrapper	8
imp20000	10
local.fdr	12
my.dsn	14
my.test1fun	16
my_PIMP	17
my_ranger_PIMP	18
plotQ	19
propTrueNullByLocalFDR	22
run.it.importances	22
significant.genes	24
Index	26

count_variables	<i>count the number of times each variable is used in a ranger random forest</i>
-----------------	--

Description

- count the number of times each variable is used in a ranger random forest.
- help(treeInfo) warns "splitvarID – ID of the splitting variable, 0-indexed. Caution, the variable order changes if the formula interface is used" However this should be investigated

Usage

```
count_variables(object)
```

Arguments

object a ranger forest object

Value

a table (0-indexed) giving the number of times each variable was used in the random forest

Examples

```
library(ranger)
rf1 <- ranger(Species ~ ., data = iris,importance="impurity", seed=123)
count_variables(rf1)
rf2 <- ranger(dependent.variable.name = "Species", data = iris,seed=123)
count_variables(rf2)
rf3<- ranger(y = iris[, 5], x = iris[, -5],seed=123)
count_variables(rf3)
```

determine.C

*determine.C***Description**

by assumption, there is a point q such that to the left of q , $f_B \sim f_0(z)$. That is, there is a q such that there are only null values to the left of q . We determine q using a change point method related to penalized model selection. See Gauran, Iris Ivy M. and Park, Junyong and Lim, Johan and Park, DoHwan and Zylstra, John and Peterson, Thomas and Kann, Maricel and Spouge, John L. "Empirical null estimation using zero-inflated discrete mixture distributions and its application to protein domain data" Biometrics, 2018 74:2

Usage

```
determine.C(f_fit, df, t1, trace.plot = FALSE, start_at = 30, debug.flag = 0)
```

Arguments

<code>f_fit</code>	object returned by <code>f.fit</code>
<code>df</code>	data frame containing <code>x</code> and <code>y</code>
<code>t1</code>	initial estimates of <code>xi</code> , <code>omega</code> , and <code>lambda</code> . Generally returned by <code>fit.to.data.set.wrapper</code>
<code>trace.plot</code>	– produce a plot of each fit with a 1 second sleep. Can be watched as a movie.
<code>start_at</code>	– <code>x <- f_fit\$midpoints</code> is of length 119 (quite arbitrary). We use the first <code>start_at</code> values of <code>x</code> to fit the skew-normal distribution.
<code>debug.flag</code>	– debugging level. If <code>debug.flag > 0</code> then some output is printed to the screen.

Value

– a vector of numbers of length equal to the rows in `df` (119 in this case). Say that this is `qq`. We determine the minimum value of `qq`. This is the value "C" such that – to the right of C, our data is generated from the NULL distribution – to the left of C, we have a mixture of the NULL and non-NULL distribution

Examples

```
data(imp20000)
imp<-log(imp20000$importances)
t2<-imp20000$counts
temp<-imp[t2 > 1] #see
temp<-temp[temp != -Inf]
temp <- temp - min(temp) + .Machine$double.eps
f_fit <- f.fit(temp)
y <- f_fit$zh$density
x <- f_fit$midpoints
df <- data.frame(x, y)
initial.estimates <- fit.to.data.set.wrapper(df, temp, try.counter = 3,return.all=FALSE)
initial.estimates<- initial.estimates$Estimate
```

```

qq<- determine.C(f_fit,df,initial.estimateds,start_at=40,trace.plot = FALSE)
cc<-x[which.min(qq)]
plot(x,qq,main="determine cc")
abline(v=cc)
# unfortunately the minima does not appear reasonable. In this case it is advisable to use the
# 95th quantile

#needs the chromosome 22 data in RFlocalfdr.data. Also has a long runtime.
library(RFlocalfdr.data)
data(ch22)
?ch22
t2 <-ch22$C
imp<-log(ch22$imp)
#Determine a cutoff to get a unimodal density.
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(25,30,35,40),plot=c(25,30,35,40),Q=0.75)
plot(c(25,30,35,40),res.temp[,3])
imp<-imp[t2 > 30]
debug.flag <- 0
f_fit<- f_fit(imp,debug.flag=debug.flag,temp.dir=temp.dir)
#makes the plot histogram_of_variable_importances.png
y<-f_fit$zh$density
x<-f_fit$midpoints
plot(density(imp),main="histogram and fitted spline")
lines(x,y,col="red")
df<-data.frame(x,y)
initial.estimateds <- fit.to.data.set.wrapper(df,imp,debug.flag=debug.flag,plot.string="initial",
                                             temp.dir=temp.dir,try.counter=3)

initial.estimateds <- data.frame(summary(initial.estimateds)$parameters)$Estimate
# 1.102303 1.246756 1.799169
qq<- determine.C(f_fit,df,initial.estimateds,start_at=37,trace.plot = TRUE)
cc<-x[which.min(qq)]
plot(x,qq,main="determine cc")
abline(v=cc)

```

determine_cutoff	<i>evaluate a measure that can be used to determining a significance level for the Mean Decrease in Impurity measure returned by a Random Forest model</i>
------------------	--

Description

evaluate a measure that can be used to determining a significance level for the Mean Decrease in Impurity measure returned by a Random Forest model

Usage

```
determine_cutoff(
```

```

    imp,
    t2,
    cutoff = c(0, 1, 4, 10, 15, 20),
    Q = 0.75,
    plot = NULL,
    verbose = 0,
    try.counter = 3
  )

```

Arguments

<code>imp</code>	vector of MDI variable importances
<code>t2</code>	number of times each variable is used in the a ranger forest. Returned by <code>count_variables</code> for a ranger RF
<code>cutoff</code>	values to evaluate
<code>Q</code>	– we examine the fit up to the quantile Q
<code>plot</code>	for 4 selected values of the cutoff. The plot contains • The data (black) and the fitted density (red) • The skew-normal fit (blue) • The quantile Q (vertical red line)
<code>verbose</code>	<code>verbose=0</code> , no output to screen <code>verbose=1</code> , track the cutoff value being used
<code>try.counter</code>	passed to <code>fit.to.data.set.wrapper</code>

Value

res a matrix if size `length(cutoff)` by 3. We model the histogram of `imp` with a kernel density estimate, `y`. Let `t1` be fitted values of the skew normal. Then `res` contains three columns

- `sum((y-t1)^2)`
- `sum(abs(y-t1))` and
- `max(abs(y-t1))`, evaluated up to the quantile Q

Examples

```

data(imp20000)
imp <- log(imp20000$importances)
t2<- imp20000$counts
length(imp)
hist(imp,col=6,lwd=2,breaks=100,main="histogram of importances")
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(0,1,2,3),plot=c(0,1,2,3),Q=0.75,try.counter=1)

plot(c(0,1,2,3),res.temp[,3])
# the minimum is at 1 so
imp<-imp[t2 > 1]

qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2, use_95_q=TRUE)

```

```

length(aa$probabilities) #11#
names(aa$probabilities)
#[1] "X101" "X102" "X103" "X104" "X105" "X2994" "X9365" "X10718"
# [9] "X13371" "X15517" "X16460"
# so the observed FDR is 0.54

library(ranger)
library(RFlocalfdr.data)
data(smoking)
y<-smoking$y
smoking_data<-smoking$rma
y.numeric <-ifelse((y=="never-smoked"),0,1)
rf1 <- ranger::ranger(y=y.numeric ,x=smoking_data,importance="impurity",seed=123, num.trees = 10000,
  classification=TRUE)
t2 <-count_variables(rf1)
imp<-log(rf1$variable.importance)
plot(density((imp)))
# Detemine a cutoff to get a unimodal density.
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(1,2,3,4),plot=c(1,2,3,4),Q=0.75)
plot(c(1,2,3,4),res.temp[,3])

```

f.fit

fit a spline to the histogram of imp

Description

fit a spline to the histogram of imp

Usage

```
f.fit(imp, df = 10, debug.flag = 0, temp.dir = NULL)
```

Arguments

imp	the variable importances
df	the degrees of freedom for the spline fit
debug.flag	either 0 (no debugging information), 1 or 2
temp.dir	if debug flag is >0 then information is written to temp.dir

Value

a list with the following components

- "x" – midpoints of the histogram
- "zh" – a histogram object as returned by "hist"
- "f.spline" – the spline fit. The fit is given by a glm mode `glm(zh$counts ~ splines::ns(x), poisson)`
- "counts" the counts from the histogram

Examples

```
data(imp200000)
imp <- log(imp200000$importances)
res <- f.fit(imp)
plot(res$zh, xlab="importances", main="histogram of importances")
points(res$midpoints,res$counts, col="grey90")
lines(res$zh$breaks[-1],res$f.spline,col="blue", lwd=3)
legend("topleft",c("spline fit"), col="blue", lwd=3)
```

fit.to.data.set

fit.to.data.set

Description

This function fit a skew normal to a set of data

Usage

```
fit.to.data.set(
  df,
  imp,
  debug.flag = 0,
  plot.string = "",
  temp.dir = NULL,
  try.counter = 3,
  return.all = FALSE
)
```

Arguments

df	contains x and y, midpoints and counts from a histogram of imp
imp	importances
debug.flag	debug flag
plot.string	file name for a debugging plot
temp.dir	directory for debugging output
try.counter	try.counter=1 my.dsn xi= 1 try.counter=2 xi= mean(x) try.counter=3 start xi, omega, lambda from the parameters retuned by fitdistrplus::fitdist
return.all	TRUE, return the full ouput of minpack.lm::nlsLM, FALSE , return summary of parameters

Value

If the skew-normal fitting routine is succesful, then the matrix of parmaters and standard errors is returned. – othwise a "try-error" message is returned

Examples

```

data(imp20000)
imp<-log(imp20000$importances)
t2<-imp20000$counts
temp<-imp[t2 > 1] #see
temp<-temp[temp != -Inf]
temp <- temp - min(temp) + .Machine$double.eps
f_fit <- f.fit(temp)
y <- f_fit$zh$density
x <- f_fit$midpoints
df <- data.frame(x, y)
fitted_parameters <- fit.to.data.set(df, temp, try.counter = 3)
fitted_parameters

hist(temp, breaks = 200, freq = FALSE)
lines(df$x, df$y, type = "l", col = "green", lwd = 2,
      xlim = c(0, max(df$x) + 0.5))
curve(sn::dsn(x, xi = fitted_parameters$Estimate[1], omega = fitted_parameters$Estimate[2],
             alpha = fitted_parameters$Estimate[3]), add = TRUE,
      col = "purple", lwd = 3, xlim = c(0, 16))
curve(my.dsn(x, xi = fitted_parameters$Estimate[1], omega = fitted_parameters$Estimate[2],
             lambda = fitted_parameters$Estimate[3]), add = TRUE,
      col = "orange", lwd = 3)

library(RFlocalfdr.data)
data(ch22)
imp<-log(ch22$imp)
t2<-ch22$C
temp<-imp[t2 > 30] #
temp<-temp[temp != -Inf]
temp <- temp - min(temp) + .Machine$double.eps
f_fit <- f.fit(temp)
y <- f_fit$zh$density
x <- f_fit$midpoints
df <- data.frame(x, y)
mm.df3 <- fit.to.data.set(df, temp)
mm.df3
##              Estimate Std..Error  t.value    Pr...t..
## xi.xi          1.102303 0.03669284 30.04136 1.485263e-56
## omega.omega    1.246756 0.04716184 26.43569 6.276349e-51
## lambda.alpha  1.799169 0.17343872 10.37351 3.103195e-18

```

fit.to.data.set.wrapper

fit.to.data.set.wrapper

Description

This function allows you to express your love of cats.

Usage

```
fit.to.data.set.wrapper(
  df,
  imp,
  debug.flag = 0,
  plot.string = "",
  temp.dir = temp.dir,
  return.all = TRUE,
  try.counter = 3
)
```

Arguments

df	contains x and y, midpoints and counts from a histogram of imp
imp	importances
debug.flag	debug flag
plot.string	file name for a debugging plot, passed on to fit.to.data.set
temp.dir	directory for debugging output, passed on to fit.to.data.set
return.all	passed to fit.to.data.set. If TRUE then the full output of minpack.lm::nlsLM is returned. Otherwise just the matrix of coefficients and t-values is returned.
try.counter	passed on to fit.to.data.set try.counter=1 my.dsn xi= 1 try.counter=2 xi= mean(x) try.counter=3 start xi, omega, lambda from the parameters retuned by fitdistrplus::fitdist

Value

If the skew-normal fitting routine is succesful, then the matrix of parmaters and standard errors is returned. – othwise a "try-error" message is returned

Examples

```
data(imp20000)
imp<-log(imp20000$importances)
t2<-imp20000$counts
temp<-imp[t2 > 1] #see
temp<-temp[temp != -Inf]
temp <- temp - min(temp) + .Machine$double.eps
f_fit <- f.fit(temp)
y <- f_fit$zh$density
x <- f_fit$midpoints
df <- data.frame(x, y)
fitted_parameters <- fit.to.data.set.wrapper(df, temp, try.counter = 3,return.all=FALSE)
fitted_parameters

hist(temp, breaks = 200, freq = FALSE)
lines(df$x, df$y, type = "l", col = "green", lwd = 2,
      xlim = c(0, max(df$x) + 0.5))
curve(sn::dsn(x, xi = fitted_parameters$Estimate[1], omega = fitted_parameters$Estimate[2],
```

```

        alpha = fitted_parameters$Estimate[3]), add = TRUE,
        col = "purple", lwd = 3, xlim = c(0, 16))
curve(my.dsn(x, xi = fitted_parameters$Estimate[1], omega = fitted_parameters$Estimate[2],
        lambda = fitted_parameters$Estimate[3]), add = TRUE,
        col = "orange", lwd = 3)

library(RFlocalfdr.data)
data(ch22)
t2 <-ch22$C
imp<-log(ch22$imp)
imp<-imp[t2 > 30]
imp <- imp[imp != -Inf]
imp <- imp - min(imp) + .Machine$double.eps
f_fit <- f.fit(imp )
y <- f_fit$zh$density
x <- f_fit$midpoints
C <- quantile(imp,probs=0.75)
df2 <- data.frame(x[x < C], y[x < C])
initial.estimates <- fit.to.data.set.wrapper(df2, imp)
#Nonlinear regression model
# model: y ~ my.dsn(x, xi = xi, omega = omega, lambda = lambda)
# data: df
#      xi.xi  omega.omega lambda.alpha
#      1.163    1.193    1.574
# residual sum-of-squares: 0.06269
#
#Number of iterations to convergence: 23
#Achieved convergence tolerance: 1.49e-08

```

imp20000

20000 importance values

Description

A dataset containing 20000 importance values

Usage

```
imp20000
```

Format

A vector variable importances with 20000 values

imp1 importances

Examples

```

require(ranger)
inv.logit <-function (x) {
  plogis(x)}

make_data <- function(nVars, nSamples) {
  as.matrix(sapply(1:nVars, function(t){sample(0:2, nSamples, replace=TRUE)}))
}

make_cont_response <- function(X, w) {
  (X-1) %*% w
}

make_response <- function(X, w) {
  as.factor(inv.logit((X-1) %*% w * 2 ) > runif(nrow(X)))
}

nVars <- 20000
nSamples <- 1000

set.seed(19)
X<- make_data(nVars,nSamples)

w <- rep(0, times = nVars)
w[101] <- 1
w[102] <- 1/sqrt(2)
w[103] <- 1/sqrt(4)
w[104] <- 1/sqrt(8)
w[105] <- 1/sqrt(16)
y <- make_response(X, w)

colnames(X) <- c(make.names(1:20000))
set.seed(19)
rf1<-ranger::ranger(y=y,x=X, num.trees = 2000,importance="impurity")
table(y,predict(rf1,data=X)$predictions)
#00B prediction error: 41.30 %
table(y,predict(rf1,data=X)$predictions)

t2 <-count_variables(rf1)
head(t2)
dim(t2)

imp<-rf1$variable.importance
imp<-log(imp)
plot(density((imp)))
hist(imp,col=6,lwd=2,breaks=100,main="histogram of importances")

res.temp <- determine_cutoff(imp, t2 ,cutoff=c(0,1,2,3),plot=c(0,1,2,3),Q=0.75,try.counter=1)
plot(c(0,1,2,3),res.temp[,3])
imp<-imp[t2 > 1]

```

```

qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2, use_95_q=TRUE)
length(aa$probabilities)
names(aa$probabilities)
#' #[1] "X101" "X102" "X103" "X104" "X105" "X2994" "X9365" "X10718"
# [9] "X13371" "X15517" "X16460"

counts<-t2
imp20000 <- list(imp,counts)
names(imp20000) <-c("importances","counts")

```

local.fdr

local.fdr

Description

calculate the local

Usage

```

local.fdr(
  f,
  x,
  FUN = my.dsn,
  p0 = 1,
  debug.flag = 0,
  plot.string = "",
  temp.dir = NULL,
  ...
)

```

Arguments

f	object returned by call to f.fit
x	f_fit\$midpoints
FUN	my.dsn
p0	estimated proportion of null importances
debug.flag	either 0 (no debugging information), 1 or 2
plot.string	file name for a debugging plot
temp.dir	directory for debugging output
...	arguments passed to FUN

Value

returns an estimate of the local false discovery rate.

Examples

```
data(imp20000)
imp<-log(imp20000$importances)
t2<-imp20000$counts
temp<-imp[t2 > 1] #see
temp<-temp[temp != -Inf]
temp <- temp - min(temp) + .Machine$double.eps
f_fit <- f.fit(temp)
y <- f_fit$zh$density
x <- f_fit$midpoints
df <- data.frame(x, y)
fitted_parameters <- fit.to.data.set(df, temp, try.counter = 3)
fitted_parameters

aa <- local.fdr(f_fit, df$x, FUN = my.dsn, xi = fitted_parameters$Estimate[1],
               omega = fitted_parameters$Estimate[2], lambda = fitted_parameters$Estimate[3],
               debug.flag = 0, plot.string = "initial")

plot(x,y,axes=FALSE,type="l",col="blue",main = "local fdr",
      xlab="importances",ylab="")
axis(2, pretty( c(0,max(y)+0.5*max(y)),10))

oldpar <- par(new = TRUE)
plot(x, aa, type="l",col="green",main = "",xlab="",ylab="",axes=FALSE)
abline(h = 0.2)
axis(4, pretty( aa,10))

axis(1,pretty(x,10))
box() #- to make it look "as usual
legend("topright",c("density importances","local fdr"),col=c("blue","green"),lty=1)
par(oldpar)

library(RFlocalfdr.data)
data(ch22)
imp<-log(ch22$imp)
t2<-ch22$C
imp<-imp[t2 > 30]
imp <- imp - min(imp) + .Machine$double.eps
debug.flag <- 0
f_fit <- f.fit(imp, debug.flag = debug.flag)
y <- f_fit$zh$density
x <- f_fit$midpoints
df <- data.frame(x, y)
initial.estimates <- fit.to.data.set.wrapper(df, imp, debug.flag = debug.flag,
return.all = FALSE)

aa <- local.fdr(f_fit, df$x, FUN = my.dsn, xi = initial.estimates$Estimate[1],
```

```

omega = initial.estimate$Estimate[2], lambda = initial.estimate$Estimate[3], debug.flag = 0,
      plot.string = "initial")

plot(x,y,axes=FALSE,type="l",col="blue",main = "local fdr",
      xlab="importances",ylab="")
axis(2, pretty( c(0,max(y)+0.5*max(y)),10))

oldpar <- par(new = TRUE)
plot(x, aa, type="l",col="green",main = "",xlab="",ylab="",axes=FALSE)
abline(h = 0.2)
axis(4, pretty( aa,10))

axis(1,pretty(x,10))
box() #- to make it look "as usual"
legend("topright",c("density importances","local fdr"),col=c("blue","green"),lty=1)
par(oldpar)

```

my.dsn

my.dsn

Description

density of skew-normal using the approximation of Ashour, Samir K. and Abdel-hameed, Mahmood A.

Usage

```

my.dsn(x, xi = 0, omega = 1, lambda = 1)

dsn(x, xi = 0, omega = 1, alpha = 0, tau = 0)

psn(q, xi = -Inf, omega = 1, alpha = 0, tau = 0, ...)

qsn(p, xi = Inf, omega = 1, alpha = 0, tau = 0, ...)

```

Arguments

x	vector of quantiles. Missing values ('NA's) and 'Inf's are allowed.
xi	vector of location parameters.
omega	vector of scale parameters; must be positive
lambda	param
alpha	vector of slant parameter(s)
tau	= 0
q	vector of quantiles
...	arguments passed to sn
p	vector of probabilities. Missing values ('NA's) are allowed

Details

See https://en.wikipedia.org/wiki/Skew_normal_distribution for discussion of the skew-normal. Using the approximation of Ashour, Samir K. and Abdel-hameed, Mahmood A. "Approximate Skew Normal Distribution", Journal of Advanced Research, 2010, 1:4. It accepts the parameters ξ , ω , λ (Ashour et. al. 2010). Other formulations may use different parameterizations. The sn (skew-normal) package include the extended skew-normal (ESN) distribution. For the SN the tau parameter is 0.

RFlocalfdr also uses wrappers around the functions dsn, qsn and psn from the package "sn" <https://cran.r-project.org/web/packages/sn/>. This is due to the fact that `fitdistrplus::fitdist(imp, "sn", start = list(xi = mean(imp))...` returns warnings such as The dsn function should return a zero-length vector when input has length zero and not raise an error The psn function should have its first argument named: q as in base R These wrappers ensure conformity with the expectations of `fitdistrplus::fitdist`

Value

fits the Density function for the skew-normal (SN) distribution.

Examples

```
library(sn)
curve(sn::dsn(x,xi=0, omega=1, alpha=1, tau=0),xlim=c(-10,10),col="blue")
curve(sn::dsn(x,xi=0, omega=1, alpha=0.1, tau=0),xlim=c(-10,10),col="blue",add=TRUE)
curve(sn::dsn(x,xi=1, omega=2, alpha=2, tau=0),xlim=c(-10,20),col="blue",add=TRUE)
curve(sn::dsn(x,xi=3, omega=4, alpha=4, tau=0),xlim=c(-10,20),col="blue",add=TRUE)

curve(my.dsn(x),xlim=c(-10,10),col="red",add=TRUE)
curve(my.dsn(x,lambda=0.1),xlim=c(-10,10),col="red",add=TRUE)
curve(my.dsn(x,xi=1, omega=2, lambda=2),xlim=c(-10,20),col="red",add=TRUE)
curve(my.dsn(x,xi=3, omega=4, lambda=4),xlim=c(-10,20),col="red",add=TRUE)

#dsn, qsn and psn are wrappers around the provided functions provided by sn. This is done to
# overcome some checking done by fitdistrplus

library(sn)
getAnywhere("dsn")
RFlocalfdr::my.test1fun("sn::dsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))
RFlocalfdr::my.test1fun("sn::psn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))
RFlocalfdr::my.test1fun("sn::qsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))
#all return FALSE

detach("package:sn", unload=TRUE)
getAnywhere("dsn")
RFlocalfdr::my.test1fun("dsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))#TRUE
RFlocalfdr::my.test1fun("psn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))#TRUE
RFlocalfdr::my.test1fun("qsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))#TRUE
```

my.test1fun

my.test1fun

Description

tests the compliance of skew-normal distribution functions with the expectations of the package `fitdistrplus`

Usage

```
my.test1fun(fn, start.arg, fix.arg, dpqr)
```

Arguments

fn	• name of the function to be tested "dsn", "psn" or "qsn"
start.arg	• the starting arguments for fitting the density
fix.arg	• fixed arguments
dpqr	– are we testing the "d", "p" or "q" function? not needed as it can be inferred from the argument "fn"

Details

RFlocalfdr also uses wrappers around the functions `dsn`, `qsn` and `psn` from the package "sn" <https://cran.r-project.org/web/packages/sn/>. This is due to the fact that `fitdistrplus::fitdist(imp, "sn", start = list(xi = mean(imp))...` returns warnings such as The `dsn` function should return a zero-length vector when input has length zero and not raise an error The `psn` function should have its first argument named: `q` as in base R These wrappers ensure conformity with the expectations of `fitdistrplus::fitdist`

Value

fits the Density function for the skew-normal (SN) distribution.

Examples

```
library(sn)
curve(sn::dsn(x,xi=0, omega=1, alpha=1, tau=0),xlim=c(-10,10),col="blue")
curve(sn::dsn(x,xi=0, omega=1, alpha=0.1, tau=0),xlim=c(-10,10),col="blue",add=TRUE)
curve(sn::dsn(x,xi=1, omega=2, alpha=2, tau=0),xlim=c(-10,20),col="blue",add=TRUE)
curve(sn::dsn(x,xi=3, omega=4, alpha=4, tau=0),xlim=c(-10,20),col="blue",add=TRUE)

curve(my.dsn(x),xlim=c(-10,10),col="red",add=TRUE)
curve(my.dsn(x,lambda=0.1),xlim=c(-10,10),col="red",add=TRUE)
curve(my.dsn(x,xi=1, omega=2, lambda=2),xlim=c(-10,20),col="red",add=TRUE)
curve(my.dsn(x,xi=3, omega=4, lambda=4),xlim=c(-10,20),col="red",add=TRUE)

#dsn, qsn and psn are wrappers around the provided functions provided by sn. This is done to
# overcome some checking done by fitdistrplus
```

```

library(sn)
getAnywhere("dsn")
RFlocalfdr::my.test1fun("sn::dsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))
RFlocalfdr::my.test1fun("sn::psn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))
RFlocalfdr::my.test1fun("sn::qsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))
#all return FALSE

detach("package:sn", unload=TRUE)
getAnywhere("dsn")
RFlocalfdr::my.test1fun("dsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))#TRUE
RFlocalfdr::my.test1fun("psn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))#TRUE
RFlocalfdr::my.test1fun("qsn", list(xi = -Inf, omega =1, alpha=0 ), fix.arg = list(tau = 0))#TRUE

```

my_PIMP	<i>my_PIMP based on the PIMP function from the vita package. 'PIMP' implements the test approach of Altmann et al. (2010) for the permutation variable importance measure 'VarImp' returned by the randomForest package (Liaw and Wiener (2002)) for classification and regression.</i>
---------	---

Description

my_PIMP applies the same method as PIMP but to the MDI (mean decrease in impurity) variable importance (mean decrease in Gini index for classification and mean decrease in MSE for regression).

Usage

```
my_PIMP(X, y, rForest, S = 100, parallel = FALSE, ncores = 0, seed = 123, ...)
```

Arguments

X	data matrix of size n by p
y	class labels for classification (factor) or real values for regression. Of length n
rForest	an object of class randomForest, importance must be set to "impurity".
S	The number of permutations for the response vector 'y'. Default is 'S=100'
parallel	Should the PIMP-algorithm run parallel? Default is parallel=FALSE and the number of cores is set to one. The parallelized version of the PIMP-algorithm are based on mclapply and so is not available on Windows
ncores	The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. If 'ncores=0', then the half of CPU cores on the current host are used.
seed	a single integer value to specify seeds. The "combined multiple-recursive generator" from L'Ecuyer (1999) is set as random number generator for the parallelized version of the PIMP-algorithm. Default is 'seed = 123'.
...	additional arguments passed to randomForest

Value

an object of class PIMP

Examples

```
library(RFlocalfdr.data)
library(ranger)
library(vita) #vita: Variable Importance Testing Approaches
data(smoking)
?smoking
y<-smoking$y
y<-factor(y)
smoking_data<-smoking$rma

cl.ranger <- ranger::ranger(y=y, x=smoking_data,mtry = 3,num.trees = 1000, importance = 'impurity')
system.time(pimp.varImp.cl<-my_ranger_PIMP(smoking_data,y,cl.ranger,S=10, parallel=TRUE, ncores=2))
#CRAN limits the number of cores available to packages to 2, for performance reasons.
pimp.t.cl <- vita::PimpTest(pimp.varImp.cl,para = FALSE)
aa <- summary(pimp.t.cl,pless = 0.05)
length(which(aa$cmat2[,"p-value"]< 0.05))
hist(aa$cmat2[,"p-value"],breaks=20)
```

my_ranger_PIMP

my_ranger_PIMP based on the PIMP function from the vita package. 'PIMP' implements the test approach of Altmann et al. (2010) for the permutation variable importance measure 'VarImp' returned by the randomForest package (Liaw and Wiener (2002)) for classification and regression.

Description

my_PIMP applies the same method as PIMP but to the MDI (mean decrease in impurity) variable importance (mean decrease in Gini index for classification and mean decrease in MSE for regression). my_ranger_PIMP applies the same method to the ranger RF package

Usage

```
my_ranger_PIMP(
  X,
  y,
  rForest,
  S = 100,
  parallel = FALSE,
  ncores = 0,
  seed = 123,
  ...
)
```

Arguments

<code>X</code>	data matrix of size <code>n</code> by <code>p</code>
<code>y</code>	class labels for classification (factor) or real values for regression. Of length <code>n</code>
<code>rForest</code>	an object of class <code>ranger</code> , importance must be set to "impurity".
<code>S</code>	The number of permutations for the response vector 'y'. Default is 'S=100'
<code>parallel</code>	Should the PIMP-algorithm run parallel? Default is 'parallel=FALSE' and the number of cores is set to one. The parallelized version of the PIMP-algorithm are based on <code>mclapply</code> and so is not available on Windows
<code>ncores</code>	The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. If 'ncores=0', then the half of CPU cores on the current host are used.
<code>seed</code>	a single integer value to specify seeds. The "combined multiple-recursive generator" from L'Ecuyer (1999) is set as random number generator for the parallelized version of the PIMP-algorithm. Default is 'seed = 123'.
<code>...</code>	additional arguments passed to <code>ranger</code>

Value

an object of class `PIMP`

Examples

```
library(RFlocalfdr.data)
library(ranger)
library(vita) #vita: Variable Importance Testing Approaches
data(smoking)
?smoking
y<-smoking$y
y<-factor(y)
smoking_data<-smoking$rma

cl.ranger <- ranger::ranger(y=y, x=smoking_data,mtry = 3,num.trees = 1000, importance = 'impurity')
system.time(pimp.varImp.cl<-my_ranger_PIMP(smoking_data,y,cl.ranger,S=10, parallel=TRUE, ncores=2))
#CRAN limits the number of cores available to packages to 2, for performance reasons.
pimp.t.cl <- vita::PimpTest(pimp.varImp.cl,para = FALSE)
aa <- summary(pimp.t.cl,pless = 0.05)
length(which(aa$cmat2[, "p-value"]< 0.05))
hist(aa$cmat2[, "p-value"],breaks=20)
```

Description

produces a plot showing the q values

- `q_95`, the 95th quantile of the data
- `q` using the penalized selection method of Gauran et.al 2018

Usage

```
plotQ(imp, debug.flag = 0, temp.dir = NULL, try.counter = 3, start_at = 37)
```

Arguments

<code>imp</code>	"reduction in impurity" importances from a random forest model
<code>debug.flag</code>	either 0 (no debugging information), 1 or 2
<code>temp.dir</code>	if debug flag is >0 then information is written to temp.dir
<code>try.counter</code>	where to explain this?
<code>start_at</code>	– <code>x <- f_fit\$midpoints</code> is of length 119 (quite arbitrary). We use the first <code>start_at</code> values of <code>x</code> to fit the skew-normal distribution.

Details

We estimate a value "q" such that: to the left of "q", the density is composed solely of NULL importance values to the right of "q" we have a density that is a mixture of null and non-null importance values. The method of Gauran et.al 2018 may not work in cases where the data distribution is not well modelled by a skew-normal. The `q_95` value can be used as a workaround in these cases. In many cases they will be very similar

Value

- `df`, contains `x` and `y`, midpoints and counts from a histogram of `imp`
- `final.estimates_C_0.95`, the output from the fitting routine `nlsLM` in `minpack.lm`, where `df` has been truncated at the value `C_0.95` (the 0.95 quantile of the skew-Normal distribution fitted to the `imp` histogram)
- `final.estimates_cc`, as for `final.estimates_C_0.95` but with `cc` determined by the procedure of Gauran et.al 2018
- `temp.dir`, the directory where debugging information may be written
- `C_0.95`, the 0.95 quantile of the skew-Normal distribution fitted to the `imp` histogram
- `cc`, determined by the procedure of Gauran et.al 2018
- `fileConn`, a file connection for writing debugging information
- `f_fit`, a spline fit to the histogram
- `ww` the minimum value of the local `fdr`

Examples

```

data(imp20000)
imp <- log(imp20000$importances)
t2 <- imp20000$counts
plot(density((imp)))
hist(imp,col=6,lwd=2,breaks=100,main="histogram of importances")
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(0,1,2,3),plot=c(0,1,2,3),Q=0.75,try.counter=1)
plot(c(0,1,2,3),res.temp[,3])
imp<-imp[t2 > 1]
qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2, use_95_q=TRUE)
length(aa$probabilities) #11#
names(aa$probabilities)

```

```

library(RFlocalfdr.data)
data(ch22)
?ch22
#document how the data set is created
plot(density(log(ch22$imp)))
t2 <-ch22$t2
imp<-log(ch22$imp)
#Determine a cutoff to get a unimodal density.
# This was calculated previously. See determine_cutoff
imp<-imp[t2 > 30]
qq <- plotQ(imp,debug.flag = 0)

```

```

data(smoking)
?smoking
y<-smoking$y
smoking_data<-smoking$rma
y.numeric <-ifelse((y=="never-smoked"),0,1)

```

```

library(ranger)
rf1 <-ranger(y=y.numeric,x=smoking_data,importance="impurity",seed=123, num.trees = 10000,
             classification=TRUE)
t2 <-count_variables(rf1)
imp<-log(rf1$variable.importance)
plot(density(imp),xlab="log importances",main="")
cutoffs <- c(2,3,4,5)
res.con<- determine_cutoff(imp,t2,cutoff=cutoffs,plot=c(2,3,4,5))

plot(cutoffs,res.con[,3],pch=15,col="red",cex=1.5,ylab="max(abs(y - t1))")
cutoffs[which.min(res.con[,3])]

```

```

temp<-imp[t2 > 3]
temp <- temp - min(temp) + .Machine$double.eps
qq <- plotQ(temp)
ppp<-run.it.importances(qq,temp,debug.flag = 0)
aa<-significant.genes(ppp,temp,cutoff=0.05,debug.flag=0,do.plot=TRUE,use_95_q=TRUE)
length(aa$probabilities) # 17

```

propTrueNullByLocalFDR

propTrueNullByLocalFDR

Description

Estimate proportion of NULL p-values. Based on .propTrueNullByLocalFDR in limma: Linear Models for Microarray Data written by Belinda Phipson and Gordon Smyth

Usage

```
propTrueNullByLocalFDR(p)
```

Arguments

p probabilities

Value

An estimate of the proportion of null p-values by the local fdr

run.it.importances

run.it.importances

Description

run.it.importances

Usage

```
run.it.importances(qq, imp, debug.flag = 0, temp.dir = NULL)
```

Arguments

qq object returned by plotQ
imp "reduction in impurity" importances from a random forest model
debug.flag either 0 (no debugging information), 1 or 2
temp.dir if debug flag is >0 then information is written to temp.dir

Value

return a list containing

- "C_0.95" estimate of the cutoff "C" such that there are only null values to the left of C. Based on the 95th quantile of the density
- "cc" estimate of the cutoff "C" based on the procedure of Gauran, et al., 2918, Biometrics,
- "estimates_C_0.95" estimate of the parameters of the SN using the data up to the C estimate
- "estimates_cc" estimate of the parameters of the SN using the data up to the C estimate
- "fdr_0.95" estimate of the fdr curve using the SN from "estimates_C_0.95"
- "fdr_cc" estimate of the fdr curve using the SN from "estimates_cc"
- "x" the x values from plotQ
- "temp.dir" the temp directory for debugging
- "p0" the estimate of the proportion of null values (can be 1)

Examples

```
data(imp20000)
imp <- log(imp20000$importances)
t2 <- imp20000$counts
plot(density((imp)))
hist(imp,col=6,lwd=2,breaks=100,main="histogram of importances")
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(0,1,2,3),plot=c(0,1,2,3),Q=0.75,try.counter=1)
plot(c(0,1,2,3),res.temp[,3])
imp<-imp[t2 > 1]
qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2, use_95_q=TRUE)
length(aa$probabilities) #11#
names(aa$probabilities)
```

```
library(RFlocalfdr.data)
data(ch22)
? ch22
#document how the data set is created
plot(density(log(ch22$imp)))
t2 <-ch22$C
imp<-log(ch22$imp)
#Determine a cutoff to get a unimodal density.
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(1,10,20,30),plot=c(1,10,20,30),Q=0.75)
plot(c(1,2,3,4),res.temp[,3])

res.temp <- determine_cutoff(imp, t2 ,cutoff=c(25,30,35,40),plot=c(25,30,35,40),Q=0.75)
plot(c(25,30,35,40),res.temp[,3])
imp<-imp[t2 > 30]
qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2)
length(aa$probabilities) # 6650
```

```
aa<-significant.genes(ppp,imp,cutoff=0.05,debug.flag=0,do.plot=2)
length(aa$probabilities) # 3653
```

significant.genes	<i>significant.genes</i>
-------------------	--------------------------

Description

This function selects the significant "genes" and makes some plots

Usage

```
significant.genes(
  object,
  imp,
  cutoff = 0.2,
  use_95_q = TRUE,
  do.plot = TRUE,
  debug.flag = 0
)
```

Arguments

object	object returned by run.it.importance
imp	importances
cutoff	cutoff
use_95_q	use the 0.95 q value
do.plot	do.plot either TRUE or FALSE (no plot)
debug.flag	debug.flag either 0 (no debugging information), 1 or 2

Value

A list containing

- probabilities (from the fitted SN distribution) and names of the significant variables
- the estimated FDR

Examples

```
data(imp20000)
imp <- log(imp20000$importances)
t2 <- imp20000$counts
plot(density((imp)))
hist(imp,col=6,lwd=2,breaks=100,main="histogram of importances")
res.temp <- determine_cutoff(imp, t2 ,cutoff=c(0,1,2,3),plot=c(0,1,2,3),Q=0.75,try.counter=1)
plot(c(0,1,2,3),res.temp[,3])
```

```
imp<-imp[t2 > 1]
qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2, use_95_q=TRUE)
length(aa$probabilities) #11#
names(aa$probabilities)

library(RFlocalfdr.data)
data(ch22)
? ch22
plot(density(log(ch22$imp)))
t2 <-ch22$C
imp<-log(ch22$imp)
# Detemine a cutoff to get a unimodal density.
# This may take several attempts. The default values of cutoff=c(0,1,4,10,15,20) will not find
# the minimum here.
#which occurs at 30
plot(c(25,30,35,40),res.temp[,3])
imp<-imp[t2 > 30]
qq <- plotQ(imp,debug.flag = 0)
ppp<-run.it.importances(qq,imp,debug=0)
aa<-significant.genes(ppp,imp,cutoff=0.2,debug.flag=0,do.plot=2)
length(aa$probabilities) # 6650
aa<-significant.genes(ppp,imp,cutoff=0.05,debug.flag=0,do.plot=2)
length(aa$probabilities) # 3653
```

Index

- * **cats**
 - propTrueNullByLocalFDR, [22](#)
- * **counts**
 - count_variables, [2](#)
- * **datasets**
 - imp20000, [10](#)
- * **genes**
 - significant.genes, [24](#)
- * **importance**
 - plotQ, [19](#)
 - run.it.importances, [22](#)
- * **normal**
 - my.dsn, [14](#)
 - my.test1fun, [16](#)
- * **significant**
 - significant.genes, [24](#)
- * **skew**
 - my.dsn, [14](#)
 - my.test1fun, [16](#)
- * **spline**
 - f.fit, [6](#)
- * **variable**
 - plotQ, [19](#)
 - run.it.importances, [22](#)

count_variables, [2](#)

determine.C, [3](#)

determine_cutoff, [4](#)

dsn(my.dsn), [14](#)

f.fit, [6](#)

fit.to.data.set, [7](#)

fit.to.data.set.wrapper, [8](#)

imp20000, [10](#)

local.fdr, [12](#)

my.dsn, [14](#)

my.test1fun, [16](#)

my_PIMP, [17](#)

my_ranger_PIMP, [18](#)

plotQ, [19](#)

propTrueNullByLocalFDR, [22](#)

psn(my.dsn), [14](#)

qsn(my.dsn), [14](#)

run.it.importances, [22](#)

significant.genes, [24](#)