R包randomForest的随机森林分类模型以及对重要变量的选择

随机森林（random forest）是一种组成式的有监督学习方法，是[决策树](https://mp.weixin.qq.com/s?__biz=MzIxNzc1Mzk3NQ==&mid=2247484634&idx=1&sn=cc4ffdb0ec695c9c950d8682e86d5560&chksm=97f5b4c2a0823dd49e08f37abac74cba51bd6149b0c526adb0b33ffd1d28cdb631379570d6a6&token=1478814531&lang=zh_CN#rd)的扩展。

随机森林通过对对象和变量进行抽样构建预测模型，即生成多个决策树，并依次对对象进行分类。最后将各决策树的分类结果汇总，所有预测类别中的众数类别即为随机森林所预测的该对象的类别，分类准确率提升。

随机森林工作过程可概括如下：

（1）假设训练集中共有N个对象、M个变量，从训练集中随机有放回地抽取N个对象构建决策树；

（2）在每一个节点随机抽取m<M个变量，将其作为分割该节点的候选变量，每一个节点处的变量数应一致；

（3）完整生成所有决策树，无需剪枝（最小节点为1）；

（4）重复（1）-（3）过程，获得大量决策树；终端节点的所属类别由节点对应的众数类别决定；

（5）对于新的观测点，用所有的树对其进行分类，其类别由多数决定原则生成。

相较于其它分类方法，随机森林通常具有如下优势：

分类准确率通常更高；

能够有效处理具有高维特征（多元）的数据集，而且不需要降维；

在处理大数据集时也具有优势；

可应用于具有大量缺失值的数据中；

能够在分类的同时度量变量对分类的相对重要性。

本篇使用微生物群落研究中的16S扩增子测序数据，展示R包randomForest中的随机森林方法。

注：randomForest包根据[经典决策树](https://mp.weixin.qq.com/s?__biz=MzIxNzc1Mzk3NQ==&mid=2247484634&idx=1&sn=cc4ffdb0ec695c9c950d8682e86d5560&chksm=97f5b4c2a0823dd49e08f37abac74cba51bd6149b0c526adb0b33ffd1d28cdb631379570d6a6&token=1478814531&lang=zh_CN#rd)生成随机森林；如果期望根据[条件推断树](https://mp.weixin.qq.com/s?__biz=MzIxNzc1Mzk3NQ==&mid=2247484634&idx=1&sn=cc4ffdb0ec695c9c950d8682e86d5560&chksm=97f5b4c2a0823dd49e08f37abac74cba51bd6149b0c526adb0b33ffd1d28cdb631379570d6a6&token=1478814531&lang=zh_CN#rd)生成随机森林，可使用party包。当预测变量间高度相关时，基于条件推断树的随机森林可能效果更好。

示例数据，R代码的百度盘链接：

<https://pan.baidu.com/s/10MWBfjBnYIzf6Cx2Zd9CjA>

## 数据集

示例文件“otu\_table.txt”为来自16S测序所获得的细菌OTUs丰度表格，共计120个样本，其中60个来自环境c（c组），60个来自环境h（h组）。

接下来使用该数据：

（1）任一OTUs的丰度都很难作为判别两种不同环境的标准，因此接下来综合考虑所有OTUs的丰度并进行建模，目的是找到能够稳定区分两种环境的代表性OTUs组合（作为生物标志物）；

（2）通过代表性OTUs的丰度构建预测模型，即仅通过这些OTUs的丰度就能够判断样本分类。

首先读入数据预处理。

#读取 OTUs 丰度表

otu <- read.table('otu\_table.txt', sep = '\t', row.names = 1, header = TRUE, fill = TRUE)

#过滤低丰度 OTUs 类群，它们对分类贡献度低，且影响计算效率

#120 个样本，就按 OTUs 丰度的行和不小于 120 为准吧

otu <- otu[which(rowSums(otu) >= 120), ]

#合并分组，得到能够被 randomForest 识别计算的格式

group <- read.table('group.txt', sep = '\t', row.names = 1, header = TRUE, fill = TRUE)

otu <- data.frame(t(otu))

otu\_group <- cbind(otu, group)

#将总数据集分为训练集（占 70%）和测试集（占 30%）

set.seed(123)

select\_train <- sample(120, 120\*0.7)

otu\_train <- otu\_group[select\_train, ]

otu\_test <- otu\_group[-select\_train, ]

## 随机森林构建

**模型拟合**

randomForest包方法的细节介绍可参考：

<https://www.stat.berkeley.edu/~breiman/RandomForests/>

#randomForest 包的随机森林

library(randomForest)

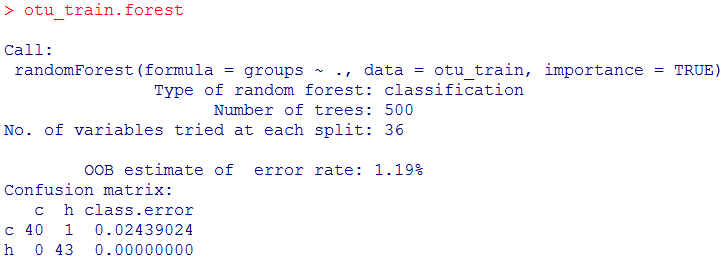
#随机森林计算（默认生成 500 棵决策树），详情 ?randomForest

set.seed(123)

otu\_train.forest <- randomForest(groups ~ ., data = otu\_train, importance = TRUE)

otu\_train.forest

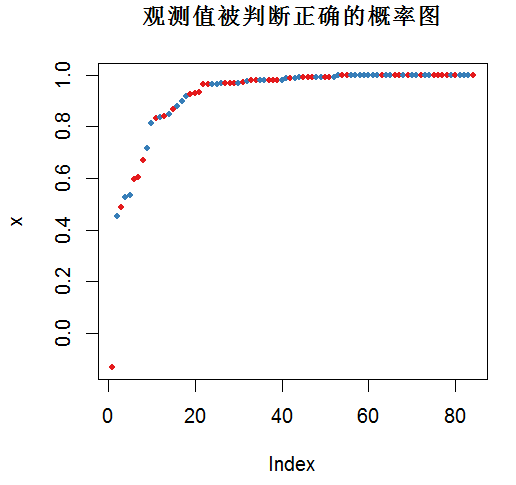
plot(margin(otu\_train.forest, otu\_train$groups), main = '观测值被判断正确的概率图')



randomForest()函数从训练集中有放回地随机抽取84个观测点，在每棵树的每个节点随机抽取36个变量，从而生成了500棵经典决策树。

生成树时没有用到的样本点所对应的类别可由生成的树估计，与其真实类别比较即可得到袋外预测（out-of-bag，OOB）误差，即OOB estimate of error rate，可用于反映分类器的错误率。此处为为1.19%，显示分类器模型的精准度是很高的，可以有效识别两类分组。

Confusion matrix比较了预测分类与真实分类的情况，class.error代表了错误分类的样本比例，这里是很低的：c 组的41个样本中40个正确分类，h组的43个样本全部正确分类。



概率图显示绝大部分样本的分类具有非常高的正确率。

若识别模糊，则出现偏离。

**分类器性能测试**

不妨使用构建好的分类器分类训练集样本，查看判别的样本分类情况。

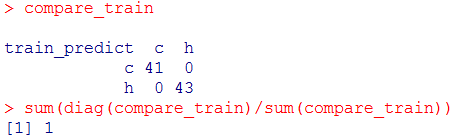
#训练集自身测试

train\_predict <- predict(otu\_train.forest, otu\_train)

compare\_train <- table(train\_predict, otu\_train$groups)

compare\_train

sum(diag(compare\_train)/sum(compare\_train))



拟合的分类模型返回来重新识别训练集数据时，甚至纠正了在拟合时的错误划分。

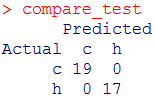
接下来使用测试集数据，进一步评估分类器性能。

#使用测试集评估

test\_predict <- predict(otu\_train.forest, otu\_test)

compare\_test <- table(otu\_test$groups, test\_predict, dnn = c('Actual', 'Predicted'))

compare\_test



显示初步构建的随机森林分类器能够准确分类训练集外的样本。

## 寻找代表性的OTUs组合

**变量重要性**

随机森林除了分类器外的另一常用功能是识别重要的变量，即计算变量的相对重要程度。

在这里，就是期望寻找能够稳定区分两种环境的代表性OTUs组合（作为生物标志物）。

##关键 OTUs 识别

#查看表示每个变量（OTUs）重要性的得分

#summary(otu\_train.forest)

importance\_otu <- otu\_train.forest$importance

head(importance\_otu)

#或者使用函数 importance()

importance\_otu <- data.frame(importance(otu\_train.forest))

head(importance\_otu)

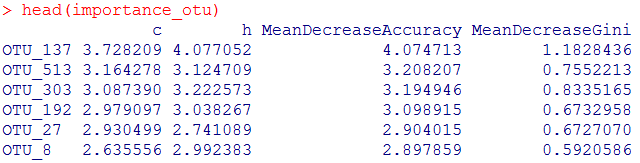
#可以根据某种重要性的高低排个序，例如根据“Mean Decrease Accuracy”指标

importance\_otu <- importance\_otu[order(importance\_otu$MeanDecreaseAccuracy, decreasing = TRUE), ]

head(importance\_otu)

#输出表格

#write.table(importance\_otu, 'importance\_otu.txt', sep = '\t', col.names = NA, quote = FALSE)



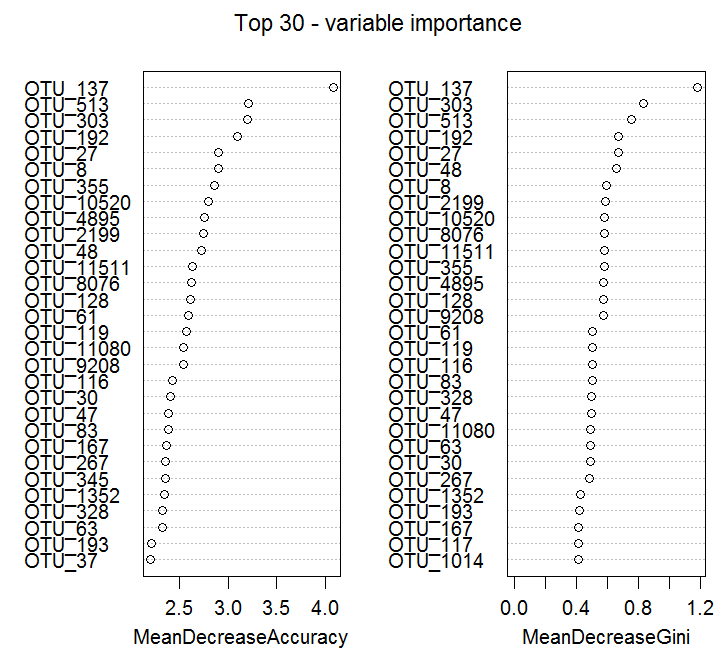
此处“Mean Decrease Accuracy”和“Mean Decrease Gini”为随机森林模型中的两个重要指标。其中，“mean decrease accuracy”表示随机森林预测准确性的降低程度，该值越大表示该变量的重要性越大；“mean decrease gini”计算每个变量对分类树每个节点上观测值的异质性的影响，从而比较变量的重要性。该值越大表示该变量的重要性越大。

到这一步，可从中筛选一些关键OTUs作为代表物种，作为有效区分两种环境的生物标志物。

该图展示了其中top30关键的OTUs，将它们划分为“关键OTUs”的依据为模型中的两个重要指标（两个指标下各自包含30个OTUs，默认由高往低排）。

#作图展示 top30 重要的 OTUs

varImpPlot(otu\_train.forest, n.var = min(30, nrow(otu\_train.forest$importance)), main = 'Top 30 - variable importance')



**交叉验证**

那么，选择多少重要的变量（本示例为OTUs）是更合适的呢？

可根据计算得到的各OUTs重要性的值（如“Mean Decrease Accuracy”），将OTUs由高往低排序后，通过执行重复5次的十折交叉验证，根据交叉验证曲线对OTU进行取舍。交叉验证法的作用就是尝试利用不同的训练集/验证集划分来对模型做多组不同的训练/验证，来应对单独测试结果过于片面以及训练数据不足的问题。此处使用训练集本身进行交叉验证。

##交叉验证帮助选择特定数量的 OTUs

#5 次重复十折交叉验证

set.seed(123)

otu\_train.cv <- replicate(5, rfcv(otu\_train[-ncol(otu\_train)], otu\_train$group, cv.fold = 10,step = 1.5), simplify = FALSE)

otu\_train.cv

#提取验证结果绘图

otu\_train.cv <- data.frame(sapply(otu\_train.cv, '[[', 'error.cv'))

otu\_train.cv$otus <- rownames(otu\_train.cv)

otu\_train.cv <- reshape2::melt(otu\_train.cv, id = 'otus')

otu\_train.cv$otus <- as.numeric(as.character(otu\_train.cv$otus))

#拟合线图

library(ggplot2)

library(splines) #用于在 geom\_smooth() 中添加拟合线，或者使用 geom\_line() 替代 geom\_smooth() 绘制普通折线

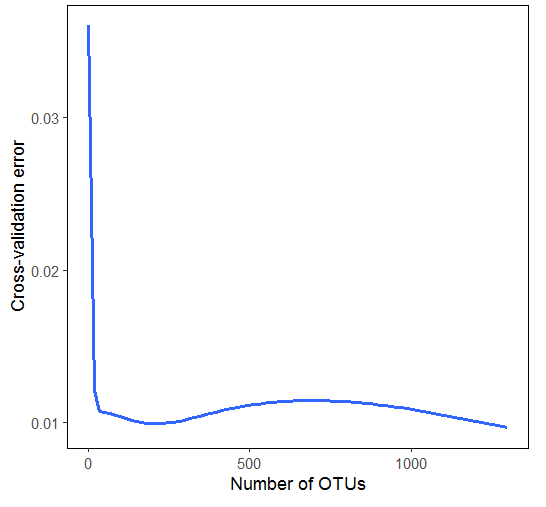
p <- ggplot(otu\_train.cv, aes(otus, value)) +

geom\_smooth(se = FALSE, method = 'glm', formula = y~ns(x, 6)) +

theme(panel.grid = element\_blank(), panel.background = element\_rect(color = 'black', fill = 'transparent')) +

labs(title = '',x = 'Number of OTUs', y = 'Cross-validation error')

p



交叉验证曲线展示了模型误差与用于拟合的OTUs数量之间的关系。误差首先会随OTUs数量的增加而减少，开始时下降非常明显，但到了特定范围处，下降幅度将不再有显著变化，甚至有所增加。再根据简约性原则，大致选择重要性排名前30的OTUs就可以了。

#大约提取前 30 个重要的 OTUs

p + geom\_vline(xintercept = 30)

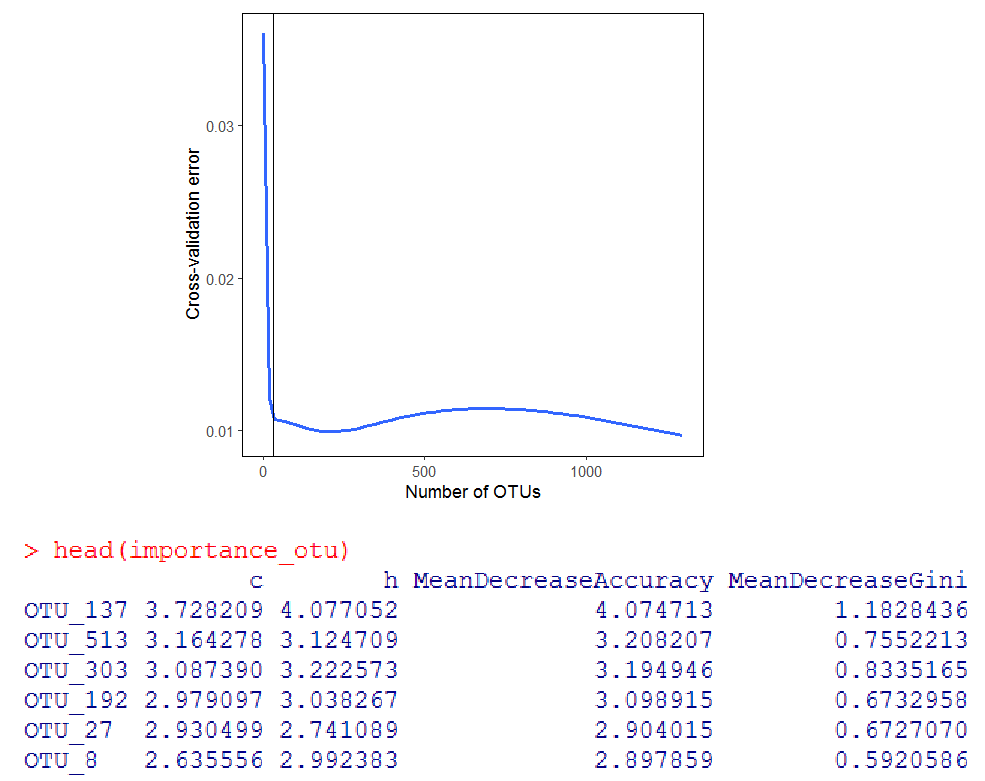
#根据 OTUs 重要性排序后选择，例如根据“Mean Decrease Accuracy”指标

importance\_otu <- importance\_otu[order(importance\_otu$MeanDecreaseAccuracy, decreasing = TRUE), ]

head(importance\_otu)

#输出表格

#write.table(importance\_otu[1:30, ], 'importance\_otu\_top30.txt', sep = '\t', col.names = NA, quote = FALSE)



## 简约分类器

如上的交叉验证曲线可反映出，并非所有变量都有助于预测模型，重要性排名靠后的变量的效应不明显甚至存在着负效应。就本文的示例而言，有些OTUs对于分类的贡献度并不高，有些可能在组间区别不大甚至会增加错误率。

因此，对于一开始构建的随机森林分类器，很多变量其实是可以剔除的。不妨就以上述选择的前30个最重要的OTUs代替原数据集中所有的OTUs进行建模，一方面助于简化分类器模型，另一方面还可提升分类精度。

##简约分类器

#选择 top30 重要的 OTUs，例如上述已经根据“Mean Decrease Accuracy”排名获得

otu\_select <- rownames(importance\_otu)[1:30]

#数据子集的训练集和测试集

otu\_train\_top30 <- otu\_train[ ,c(otu\_select, 'groups')]

otu\_test\_top30 <- otu\_test[ ,c(otu\_select, 'groups')]

#随机森林计算（默认生成 500 棵决策树），详情 ?randomForest

set.seed(123)

otu\_train.forest\_30 <- randomForest(groups ~ ., data = otu\_train\_top30, importance = TRUE)

otu\_train.forest\_30

plot(margin(otu\_train.forest\_30, otu\_test\_top30$groups), main = '观测值被判断正确的概率图')

#训练集自身测试

train\_predict <- predict(otu\_train.forest\_30, otu\_train\_top30)

compare\_train <- table(train\_predict, otu\_train\_top30$groups)

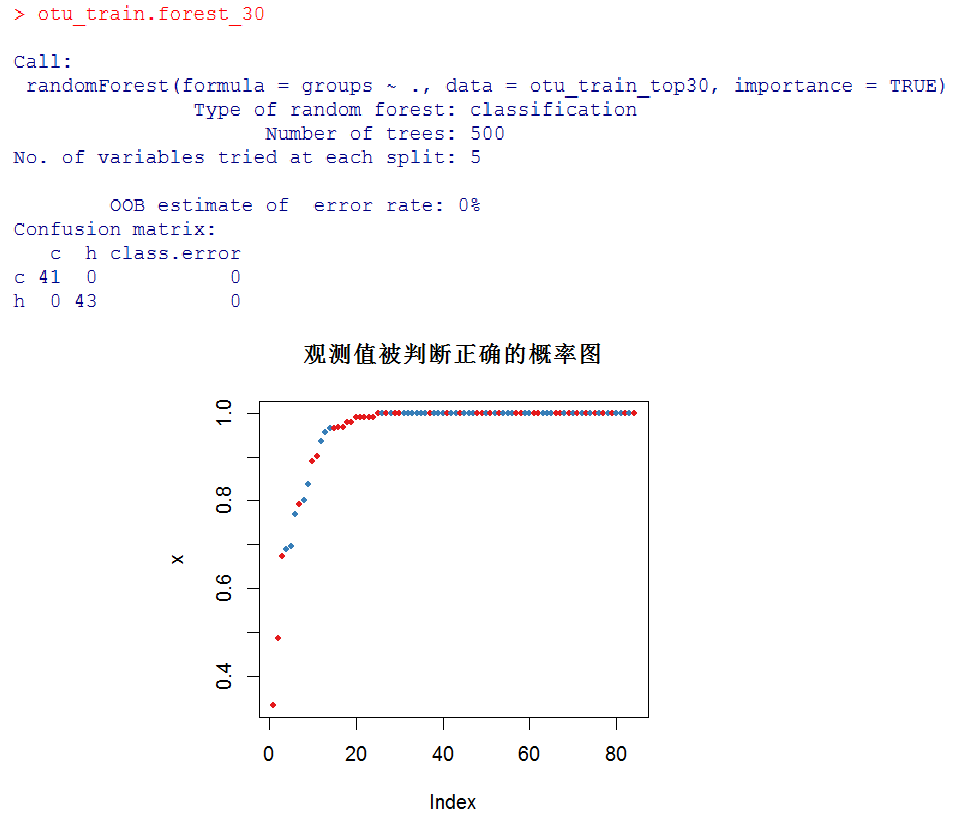
compare\_train

#使用测试集评估

test\_predict <- predict(otu\_train.forest\_30, otu\_test\_top30)

compare\_test <- table(otu\_test\_top30$groups, test\_predict, dnn = c('Actual', 'Predicted'))

compare\_test



与上文使用所有OTUs构建的分类器相比，OOB estimate of error rate降低，且Confusion matrix中也无错误分类（先前是有一个错误的），表现为精度提高。

再使用训练集和测试集评估分类器性能。

#训练集自身测试

train\_predict <- predict(otu\_train.forest\_30, otu\_train\_top30)

compare\_train <- table(train\_predict, otu\_train\_top30$groups)

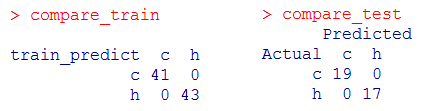
compare\_train

#使用测试集评估

test\_predict <- predict(otu\_train.forest\_30, otu\_test\_top30)

compare\_test <- table(otu\_test\_top30$groups, test\_predict, dnn = c('Actual', 'Predicted'))

compare\_test



即便只根据30个（重要的）OTUs的丰度判断样本分类，也是能够准确划分的。

将由分类器预测得到的[样本分类绘制在排序图](https://mp.weixin.qq.com/s?__biz=MzIxNzc1Mzk3NQ==&mid=2247484622&idx=2&sn=5085e09806661318c1ee25789356779d&chksm=97f5b4d6a0823dc05b49f6cfb63eead0850b4e2ab8ab053cace7b7059a769ec945540d22637e&token=1478814531&lang=zh_CN#rd)中。

##NMDS 排序图中展示分类

#NMDS 降维

nmds <- vegan::metaMDS(otu, distance = 'bray')

result <- nmds$points

result <- as.data.frame(cbind(result, rownames(result)))

#获得上述的分类预测结果

predict\_group <- c(train\_predict, test\_predict)

predict\_group <- as.character(predict\_group[rownames(result)])

#作图

colnames(result)[1:3] <- c('NMDS1', 'NMDS2', 'samples')

result$NMDS1 <- as.numeric(as.character(result$NMDS1))

result$NMDS2 <- as.numeric(as.character(result$NMDS2))

result$samples <- as.character(result$samples)

result <- cbind(result, predict\_group)

head(result)

ggplot(result, aes(NMDS1, NMDS2, color = predict\_group)) +

geom\_polygon(data = plyr::ddply(result, 'predict\_group', function(df) df[chull(df[[1]], df[[2]]), ]), fill = NA) +

geom\_point()

