Lab12. Decision Trees and Forests. Variable importance

We will use the library `party`. However, there is a number of other packages for classification and regression tree-based approach (CART): `randomForest`, `rpart`, `crat`, `maptree`, `partykit` and other.

```
library(party)
```

1. Consonant drop in Russian

Our student Varvara Sveshnikova wrote her BA paper on two cases of the consonant drop: (a) when in the complex -stvov- (like in *beschinstVovat'* 'to riot') another labial consonant is pronounced after it, and (b) when no consonant follows (in two contexts: *beschinstVuju* 'I riot', beschinstVo_ 'roistering'). The dataset includes the following data: `v.elision` — elision of [v] / no elision;
`group` — a group of test words, first (beschinstvovat'), second (beschinstvuju), third (beschinstvo);
`word` — root under analysis;
`position` — phrase position: strong, under logical stress (_I am not *CRYING*, I resent), weak (_He ALWAYS likes to *cry*).

Fit a CART model, using ctree() function, predicting v.elision variable by all others.

1.1 Visualize a model using plot() function. What is the number of observation in node 6?

1.2 Visualize a model using print() function. Which split have a statistic 14.01? In the party package, print view shows a stop criterion (1 - p-value, not smaller than 0.95 by default), t-statistic and the number of observations in each terminal node (weights).

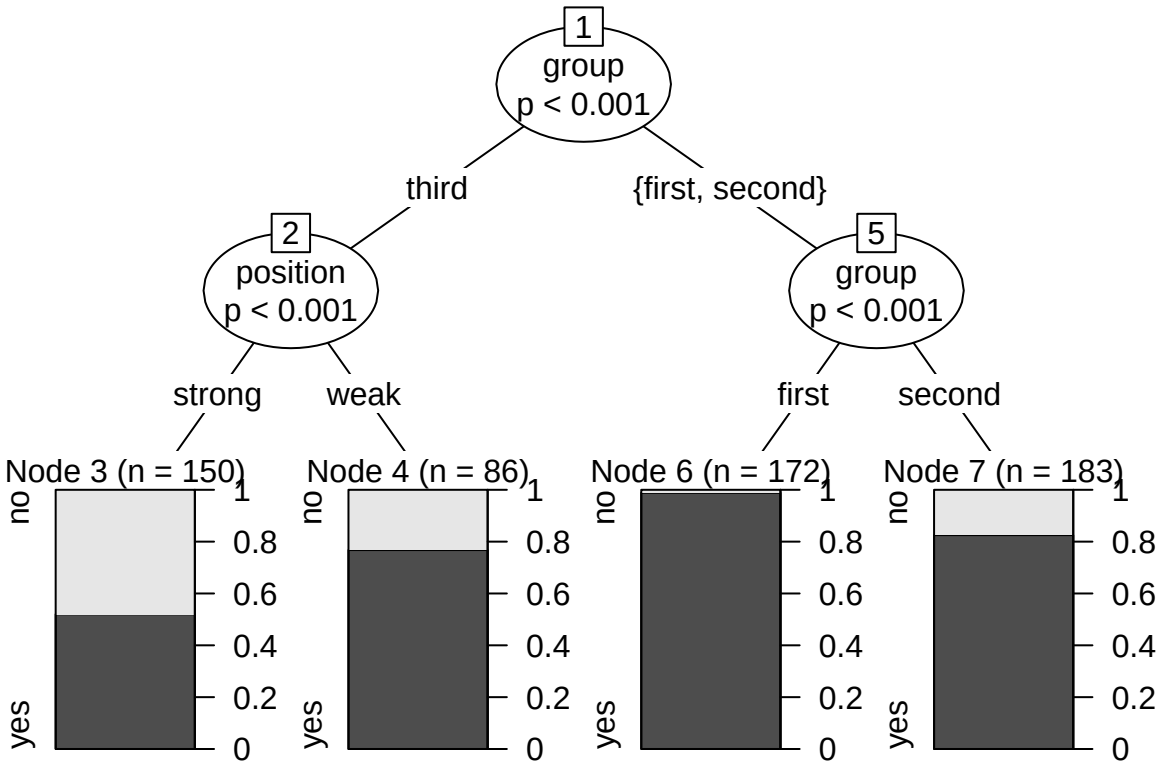
1.3 Predict a value of v.elision for word with a root "popech" in a third group, in a strong position. Fit a cforest model using additional argument controls=cforest_unbiased(ntree=1000, mtry=3).

1.4 Predict a value of v.elision for word with a root "popech" in a third group, in a strong position using `cforest` model. You need to add an argument OOB=TRUE. e. g. yes.

1.5 Calculate a variable importance for a group variable in the random forest model using varimp() function.

Code to use:

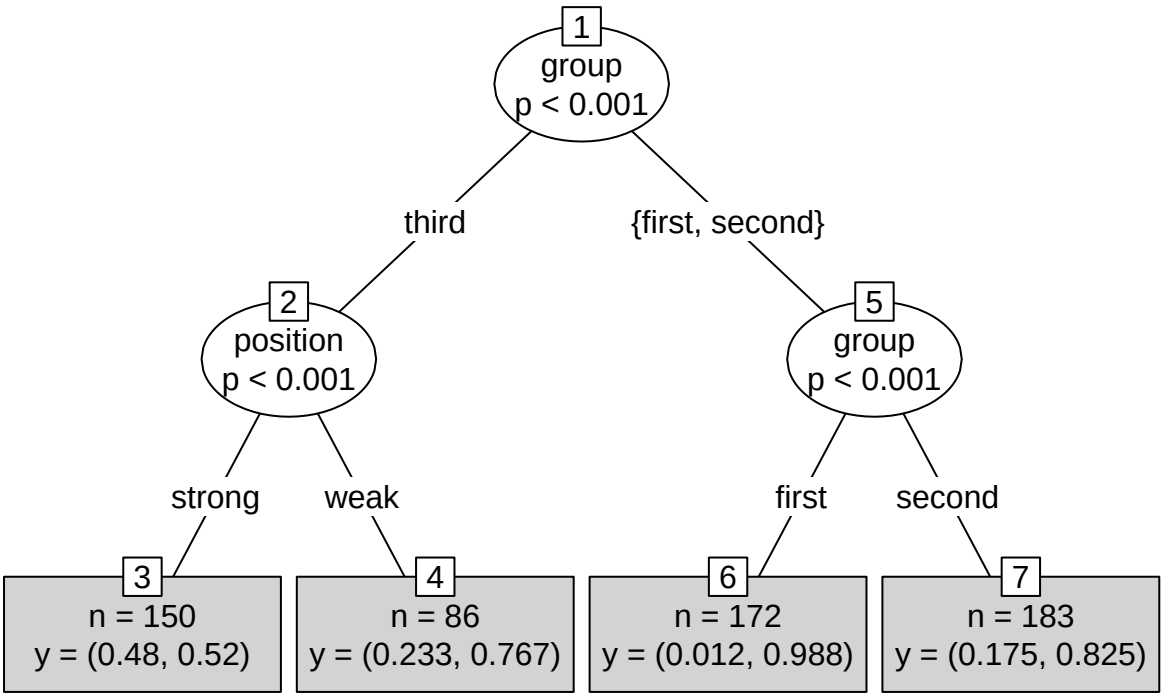
```
df <- read.csv("https://raw.githubusercontent.com/agricolamz/r_on_line_course_data/master/Sveshnikova.2(
fit <- party::ctree(v.elision~., data = df) # use the argument controls = ctree_control(...) to control
plot(fit)
```




```
print(fit)
```

```
##
##   Conditional inference tree with 4 terminal nodes
##
## Response:  v.elision
## Inputs:  group, word, position
## Number of observations:  591
##
## 1) group == {third}; criterion = 1, statistic = 87.011
##   2) position == {strong}; criterion = 0.999, statistic = 14.01
##     3)*  weights = 150
##   2) position == {weak}
##     4)*  weights = 86
## 1) group == {first, second}
##   5) group == {first}; criterion = 1, statistic = 27.204
##     6)*  weights = 172
##   5) group == {second}
##     7)*  weights = 183
```

```
plot(fit, type = "simple") # a simplified view
```




```
predict(fit, df[45,-1], response = TRUE)
```

```
## [1] yes
## Levels: no yes
```

```
fit2 <- cforest(v.elision~., data = df, control=cforest_unbiased(ntree=1000, mtry=2))
predict(fit2, df[45,-1],OOB=TRUE)
```

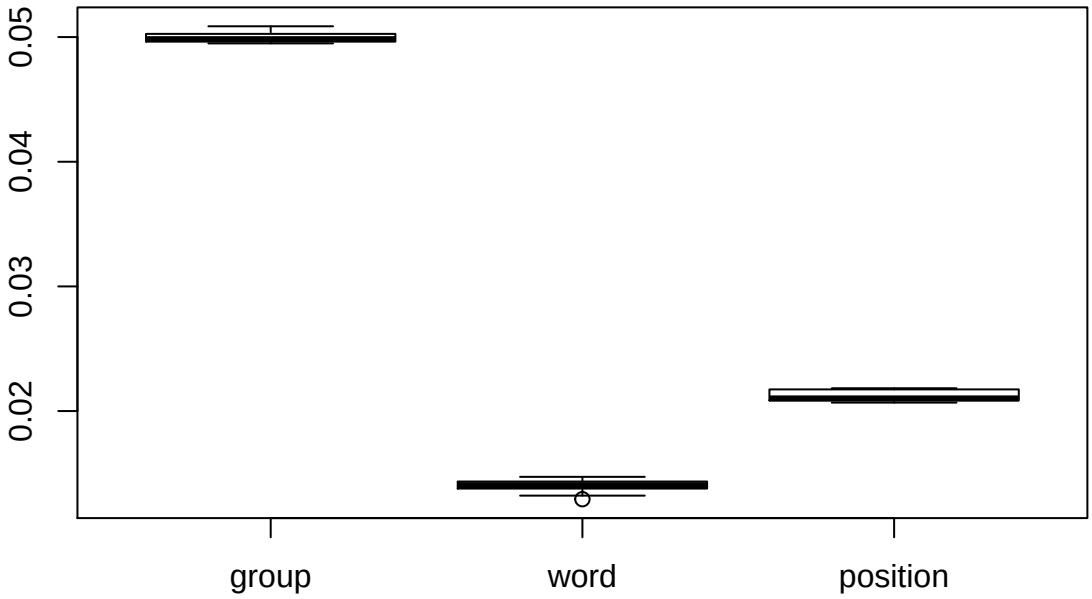
```
##   [1] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
##  [18] yes yes yes yes yes yes yes yes yes yes yes yes yes no  no  no  no  no
##  [35] no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no
##  [52] no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no
##  [69] no  no  no  no  no  no  no  no  no  yes yes yes yes yes yes yes yes yes
##  [86] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [103] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [120] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [137] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [154] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [171] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [188] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [205] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [222] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [239] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [256] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [273] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [290] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [307] yes yes yes yes yes yes yes yes yes yes yes no  no  no  no  no  no
## [324] no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no  no
## [341] no  no  no  no  no  no  no  no  no  no  no  yes yes yes yes yes yes
## [358] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [375] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [392] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [409] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
```

```
## [426] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [443] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [460] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [477] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [494] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [511] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [528] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [545] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [562] yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## [579] yes yes yes yes yes yes yes yes yes yes yes yes yes yes
## Levels: no yes
```

```
vi <- as.data.frame(sort(varimp(fit2), decreasing=TRUE))
vi
```

```
##          sort(varimp(fit2), decreasing = TRUE)
## group                               0.04994009
## position                            0.02164055
## word                                0.01412442
```

```
vi1 <- t(replicate(10, varimp(fit2)))
boxplot(vi1)
```



Model accuracy:

```
df.predicted <- predict(fit2, df[,-1], OOB=TRUE)
head(df.predicted)
```

```
## [1] yes yes yes yes yes yes
## Levels: no yes
```

```
table(df[,1], df.predicted)
```

```
##      df.predicted
##        no yes
##   no   48  78
##   yes  34 431
```

```
(sum(df[,1]==df.predicted)) / nrow(df) # accuracy
```

```
## [1] 0.8104907
```

2. /S/ deletion in Panamanian Spanish

Here's some data from Henrietta Cedergren's 1973 study of /s/-deletion in Panamanian Spanish (via Greg Guy and Scott Kiesling). Cedergren had noticed that speakers in Panama City, like in many dialects of Spanish, variably deleted the /s/ at the end of words. She undertook a study to find out if there was a change in progress: if final /s/ was systematically dropping out of Panamanian Spanish. The attached data are from interviews she performed across the city in four different social classes (1=highest, 2=second highest, 3=second lowest, 4= lowest), to see how the variation was structured in the community. She also investigated the linguistic constraints on deletion, so she coded for a phonetic constraint — whether the following segment was consonant, vowel, or pause — and the grammatical category of word that the /s/ is part of a: monomorpheme, where the s is part of the free morpheme (eg, menos) verb, where the s is the second singular inflection (eg, tu tienes, el tienes) determiner, where s is plural marked on a determiner (eg, los, las) adjective, where s is a nominal plural agreeing with the noun (eg, buenos) noun, where s marks a plural noun (eg, amigos)

Fit the CART model predicting s.deletion by phonetic environment and social class.

Data: https://raw.githubusercontent.com/LingData2019/LingData/master/data/cedergren73.csv

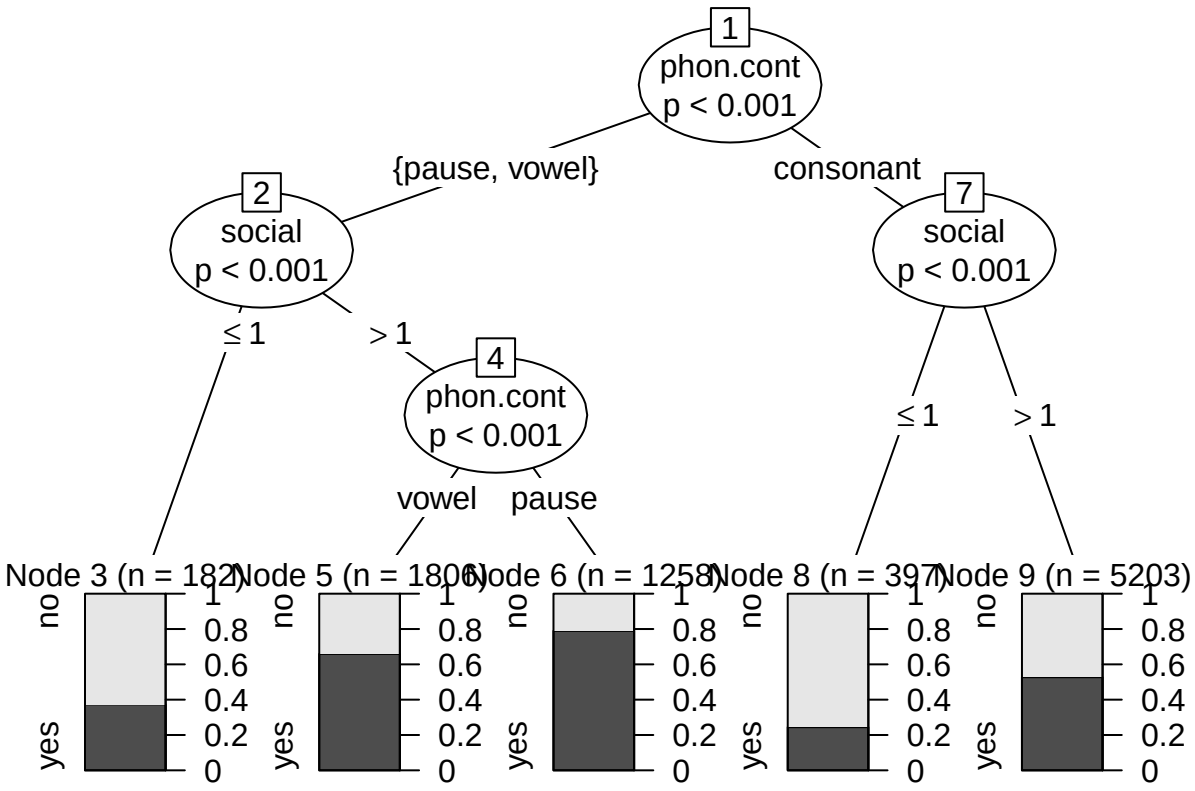
2.1 Visualize a model using plot() function. What is the number of observation in node 6?

2.2 Visualize a model using print() function. Which split have a statistic 61.559 (e. g. pause, vowel vs. consonant)?

2.3 Predict a value of s.delition for word said by person from 1 class, before consonant. Fit a `cforest` model using additional argument controls=cforest_unbiased(ntree=100, mtry=2).

2.4 Calculate a variable importance for the random forest model using varimp() function. Which of the variable is more important?

```
df <- read.csv("https://raw.githubusercontent.com/LingData2019/LingData/master/data/cedergren73.csv")
fit <- ctree(s.deletion~phon.cont+social, data = df)
plot(fit)
```




```
print(fit)
```

```
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  s.deletion
## Inputs:  phon.cont, social
## Number of observations:  8846
##
## 1) phon.cont == {pause, vowel}; criterion = 1, statistic = 344.156
##   2) social <= 1; criterion = 1, statistic = 92.762
##     3)*  weights = 182
##   2) social > 1
##     4) phon.cont == {vowel}; criterion = 1, statistic = 61.559
##       5)*  weights = 1806
##     4) phon.cont == {pause}
##       6)*  weights = 1258
## 1) phon.cont == {consonant}
##   7) social <= 1; criterion = 1, statistic = 118.054
##     8)*  weights = 397
##   7) social > 1
##     9)*  weights = 5203
```

```
predict(fit, df[1,-c(1:2)], response = TRUE)
```

```
## [1] no
## Levels: no yes
```

```
fit2 <- cforest(s.deletion~., data = df, controls=cforest_unbiased(ntree=100, mtry=2))
varimp(fit2)
```

```
##  gramm.cat  phon.cont     social
## 0.03529339 0.01363441 0.02060522
```

3. Vowel reduction in Russian

Pavel Duryagin ran an experiment on perception of vowel reduction in Russian language. The dataset shva includes the following variables: * `time1` - reaction time 1
* `duration` - duration of the vowel in the stimuly (in milliseconds, ms)
* `time2` - reaction time 2
* `f1`, `f2`, `f3` - the 1st, 2nd and 3rd formant of the vowel measured in Hz
* `vowel` - vowel classified according the 3-fold classification (A - a under stress, a - a/o as in the first syllable before the stressed one, y (stands for shva) - a/o as in the second etc. syllable before the stressed one or after the stressed syllable, cf. g[y]g[a]t[A]l[y] gogotala 'guffawed').

The dataset is available at .

Fit the CART model predicting vowel by f1 and f2.

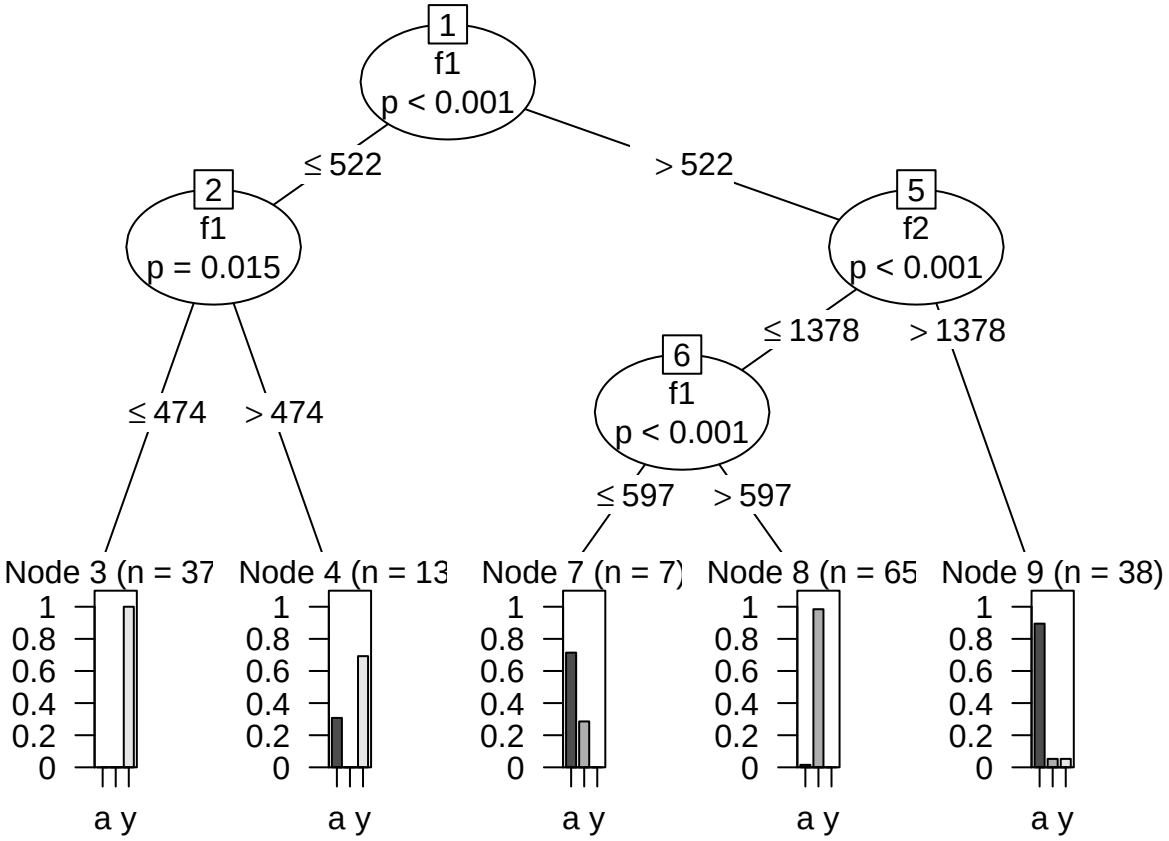
3.1 Visualize a model using plot() function. What is the number of observation in node 9?

3.2 Predict a value of vowel for sound with f1 = 600, f2 = 1300. Fit a cforest model using additional argument controls=cforest_unbiased(ntree=100, mtry=2).

3.3 Predict a value of vowel for sound with f1 = 600, f2 = 1300. You need to add an argument OOB=TRUE.

3.4 Calculate a variable importance for the random forest model using varimp() function. Which of the variable is more important?

```
shva <- read.csv("https://raw.githubusercontent.com/agricolamz/2018-MAG_R_course/master/data/duryagin_R
fit <- ctree(vowel~f1+f2, data = shva)
plot(fit)
```




```
print(fit)
```

```
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  vowel
## Inputs:  f1, f2
## Number of observations:  160
##
## 1) f1 <= 522; criterion = 1, statistic = 127.646
##   2) f1 <= 474; criterion = 0.985, statistic = 7.134
##     3)*  weights = 37
##   2) f1 > 474
##     4)*  weights = 13
## 1) f1 > 522
##   5) f2 <= 1378; criterion = 1, statistic = 66.584
##     6) f1 <= 597; criterion = 1, statistic = 21.42
##       7)*  weights = 7
##     6) f1 > 597
##       8)*  weights = 65
##   5) f2 > 1378
##     9)*  weights = 38
```

```
predict(fit, newdata = data.frame(f1 = as.integer(600),
                        f2 = as.integer(1300)), response = TRUE)
```

```
## [1] A
## Levels: a A y
```

```r
fit2 <- cforest(vowel~f1+f2, data = shva, controls=cforest_unbiased(ntree=100, mtry=2))
varimp(fit2)
```

```
##        f1        f2
## 0.4143103 0.1417241
```

```r
predict(fit2, newdata = data.frame(f1 = as.integer(600),
                        f2 = as.integer(1300)),OOB=TRUE)
```

```
## [1] A
## Levels: a A y
```

```r
fit
```

```
##
##   Conditional inference tree with 5 terminal nodes
##
## Response:  vowel
## Inputs:  f1, f2
## Number of observations:  160
##
## 1) f1 <= 522; criterion = 1, statistic = 127.646
##   2) f1 <= 474; criterion = 0.985, statistic = 7.134
##     3)*  weights = 37
##   2) f1 > 474
##     4)*  weights = 13
## 1) f1 > 522
##   5) f2 <= 1378; criterion = 1, statistic = 66.584
##     6) f1 <= 597; criterion = 1, statistic = 21.42
##       7)*  weights = 7
##     6) f1 > 597
##       8)*  weights = 65
##   5) f2 > 1378
##     9)*  weights = 38
```