Variable importances for model-based trees

Marjolein Fokkema

9-9-2022

In general, I think two possible approaches:

1) Importances based on improvement in likelihood (i.e., an impurity-based measure).
2) Effect-size driven importances (based on coefficients in terminal nodes; more similar to PRE importances).

First approach is easiest to implement. Second approach seems more interesting.

Importances based on improvement in likelihood for model-based trees

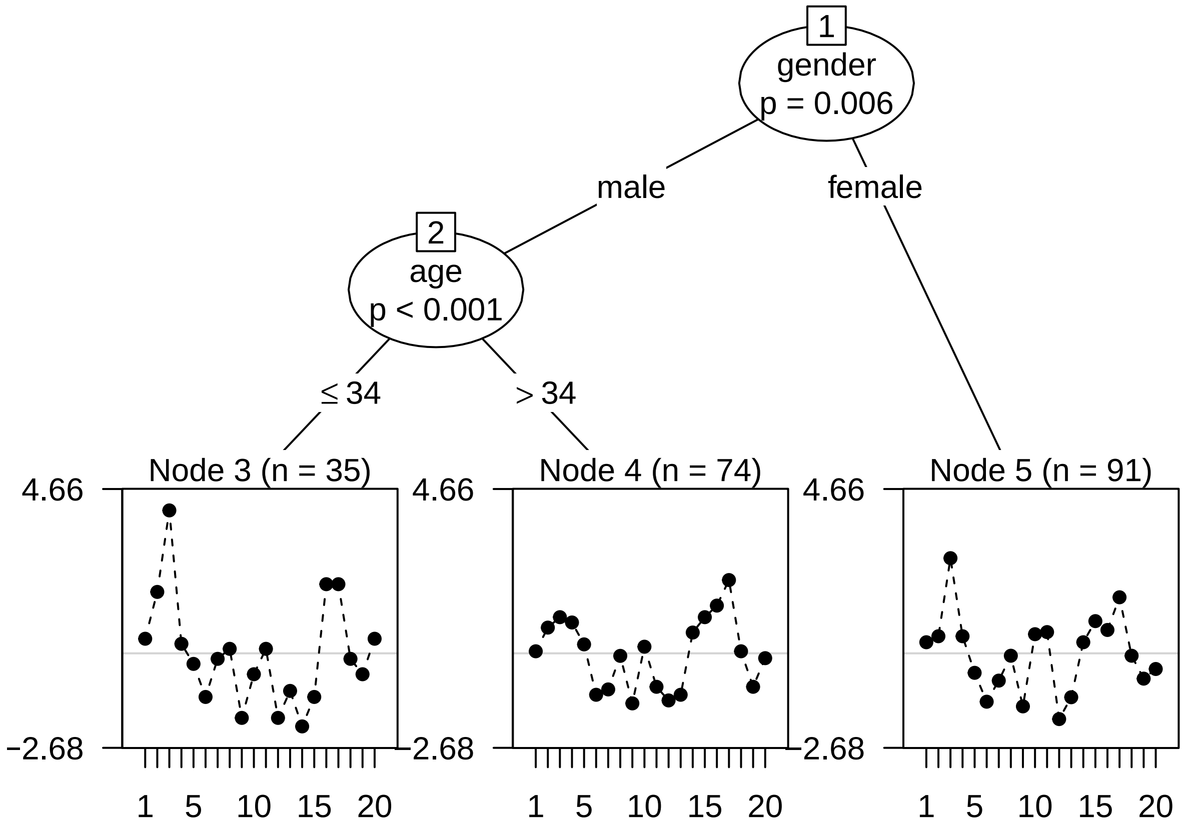
```r
##
## Function that takes any model-based tree as input, and returns the variable
## importances for all partitioning variables specified.
##
## The function checks for every split (inner) node that is not the root:
## How much the split improves the logLik
## Then adds this reduction to the varimp for that variable
##
varimp_mobtree <- function(object) {
  imps <- numeric(ncol(object[[1]]$node$info$test))
  names(imps) <- colnames(object[[1]]$node$info$test)
  for (i in 1:length(object)) {
    if (!is.null(object[[i]]$node$split)) {
      ## Sum logLik of the two daughter nodes, substract logLik of the mother node
      LL_redux <-  (logLik(object[[i]]$node$kids[[1]]$info$object) +
                      logLik(object[[i]]$node$kids[[2]]$info$object)) -
        logLik(object[[i]]$node$info$object)
      ## Add logLik reduction to current varimps
      which_var <-
        rownames(attr(terms(object), "factors"))[object[[i]]$node$split$varid]
      imps[which_var] <- imps[which_var] + LL_redux
    }
  }
  return(imps)
}
```

Illustrate use with a `raschtree`:

```r
library("psychotree")
```

```
## Loading required package: partykit

## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm

## Loading required package: psychotools
```

```r
data("DIFSim", package = "psychotree")
rt <- raschtree(resp ~ age + gender + motivation, data = DIFSim)
plot(rt)
```



```r
## print logLik values of each node
for (i in 1:length(rt)) {
  print(paste("node", i, logLik(rt[[i]]$node$info$object)))
}
```

```
## [1] "node 1 -1764.32316282135"
## [1] "node 2 -967.225924957507"
## [1] "node 3 -273.247841215574"
## [1] "node 4 -649.935172289657"
## [1] "node 5 -773.730416739513"
```

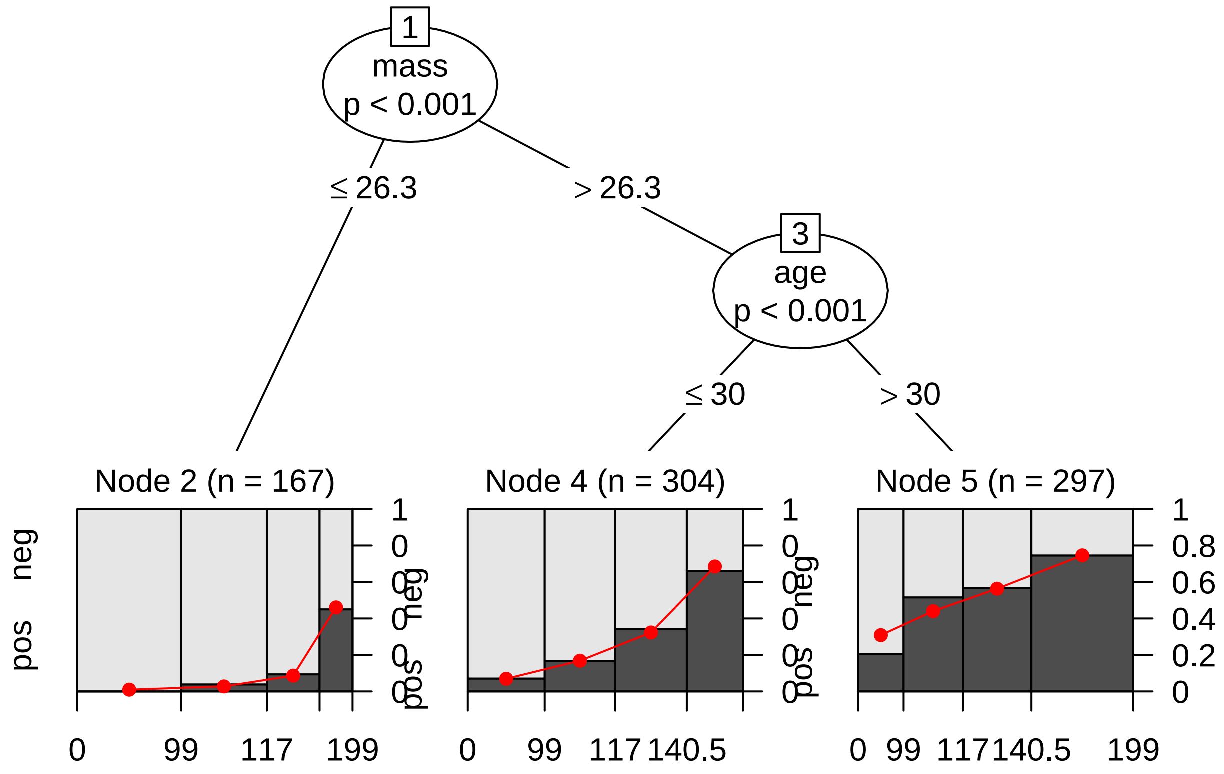
```
varimp_mobtree(rt)
```

```
##        age     gender motivation
##   44.04291   23.36682    0.00000
```

Illustrate use with a `glmtree`:

```
data("PimaIndiansDiabetes", package = "mlbench")
gt <- glmtree(diabetes ~ glucose | pregnant + pressure + triceps + insulin +
                mass + pedigree + age, data = PimaIndiansDiabetes,
              family = binomial)
plot(gt)
```

```
## Loading required namespace: vcd
```



```
## print logLik values of each node
for (i in 1:length(gt)) {
  print(paste("node", i, logLik(gt[[i]]$node$info$object)))
}
```

```
## [1] "node 1 -404.359818675696"
## [1] "node 2 -30.2511982895619"
## [1] "node 3 -344.224956244593"
## [1] "node 4 -140.490542092715"
## [1] "node 5 -184.716103886093"
```

```
varimp_mobtree(gt)
```

```
## pregnant pressure  triceps  insulin     mass pedigree      age
##  0.00000  0.00000  0.00000  0.00000 29.88366  0.00000 19.01831
```

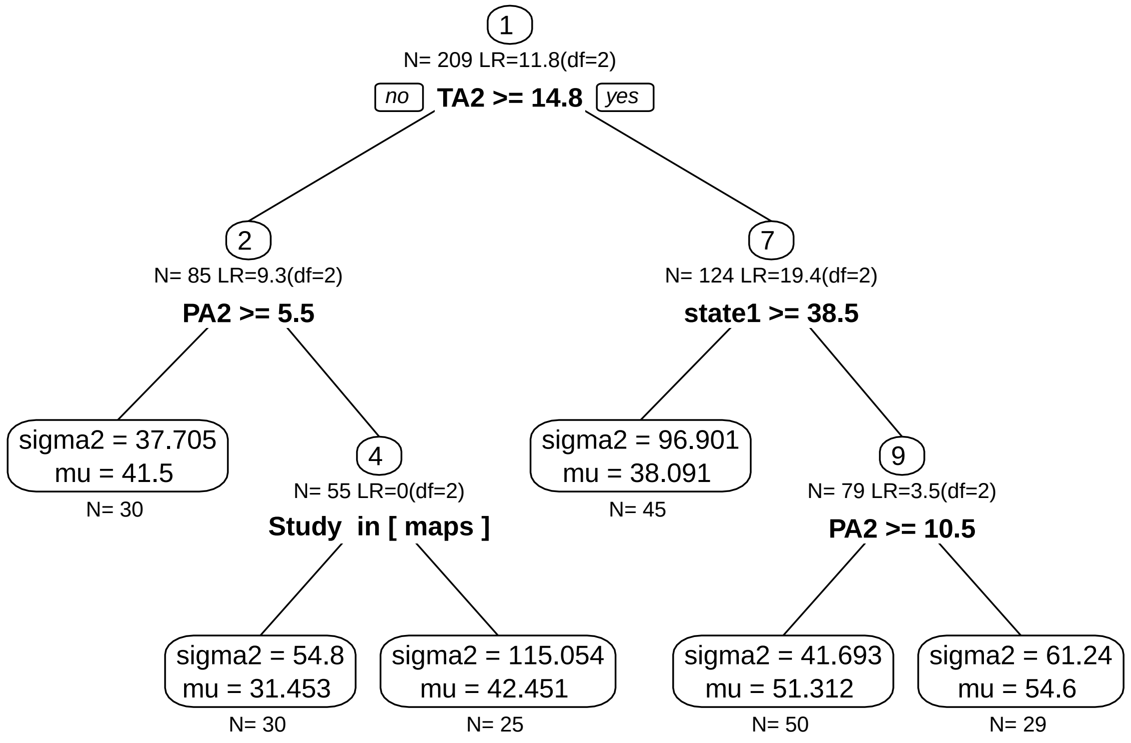
How does semforest do this?

According to `?semtree::varimp`: The value of the -2LL of the leaf nodes is compared to baseline overall model. Not sure how one gets an importance for a variable in this way. Let's fit a single-tree semforest, plot the resulting tree and compute importances:

```
library("semtree")
library("psychTools")
data(affect)
affect$Film <- as.factor(affect$Film)
affect$lie <- as.ordered(affect$lie)
affect$imp <- as.ordered(affect$imp)

library("OpenMx")
manifests <- c("state2")
latents <- c()
model <- mxModel("Univariate Normal Model",
                 type="RAM",
                 manifestVars = manifests,
                 latentVars = latents,
                 mxPath(from="one",to=manifests, free=c(TRUE),
                        value=c(50.0) , arrows=1, label=c("mu") ),
                 mxPath(from=manifests,to=manifests, free=c(TRUE),
                        value=c(100.0) , arrows=2, label=c("sigma2") ),
                 mxData(affect, type = "raw")
)
control <- semforest.control(num.trees = 1)
forest <- semforest(model=model,
                    data = affect,
                    control = control,
                    covariates = c("Study","Film", "state1", "PA2","NA2","TA2"))
```

```
## > Model was not run. Estimating parameters now before running the forest.
## <U+2714> Tree construction finished [took 8s].
## <U+2714> Forest completed [took 8s]
```

```
vim <- semtree::varimp(forest)
plot(forest$forest[[1]])
```



```
print(vim, sort.values=TRUE)
```

```
## Variable Importance
##     Study      Film       NA2    state1       PA2       TA2
## -1.869657  0.000000  0.000000 27.184811 27.600267 73.146086
```

How does mobforest do this?

mobForest returns OOB permutation importances (which I guess assumes a supervised learning problem). It has a function `varimp.output`, which according to the documentation returns: "Variable importance matrix containing the decrease in predictive accuracy after permuting the variables across all trees."

"Values of variable 'm' in the oob cases are randomly permuted and R2 obtained through variable-m-permuted oob data is subtracted from R2 obtained on untouched oob data. The average of this number over all the trees in the forest is the raw importance score for variable m."