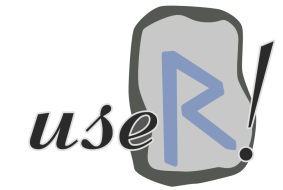
# PPforest package

Iowa State University
Natalia da Silva

July 2



```
> sessionInfo()
[1] "June 30 - July 3, 2015"
[2] "Aalborg, Denmark"
```

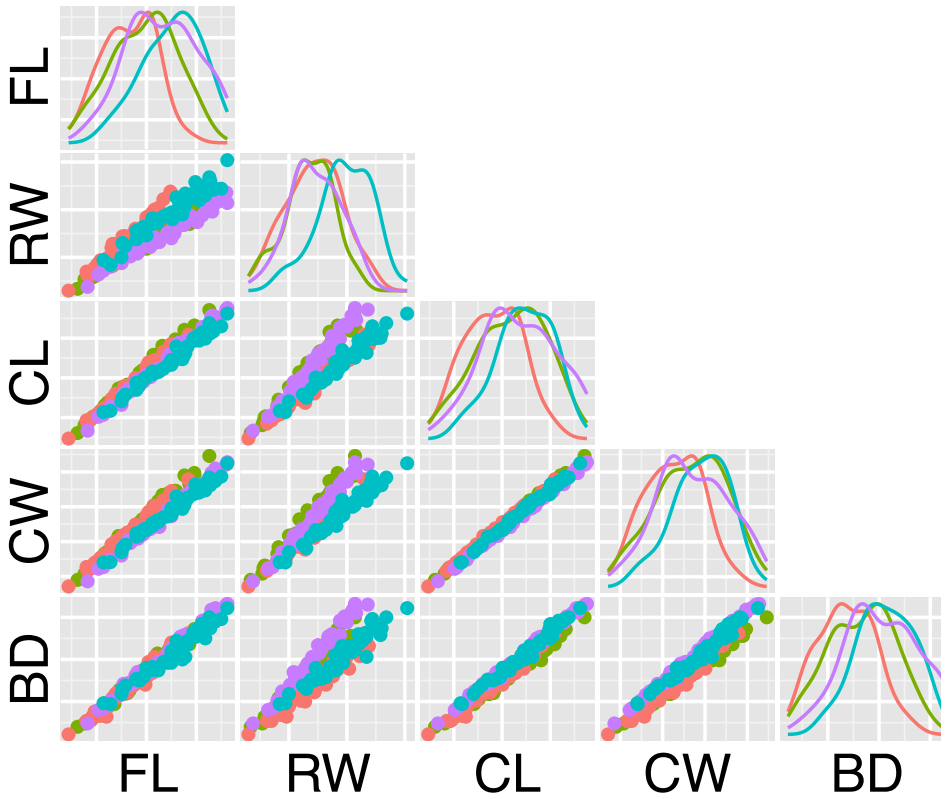
# Motivation

The most popular random forest uses univariate decision trees (CART or C4.5). Separate feature space by hyperplanes that are orthogonal to single feature axes.
When the data are collinear with correlated features, hyperplanes that are oblique to the axis maybe do better to separate the feature space.

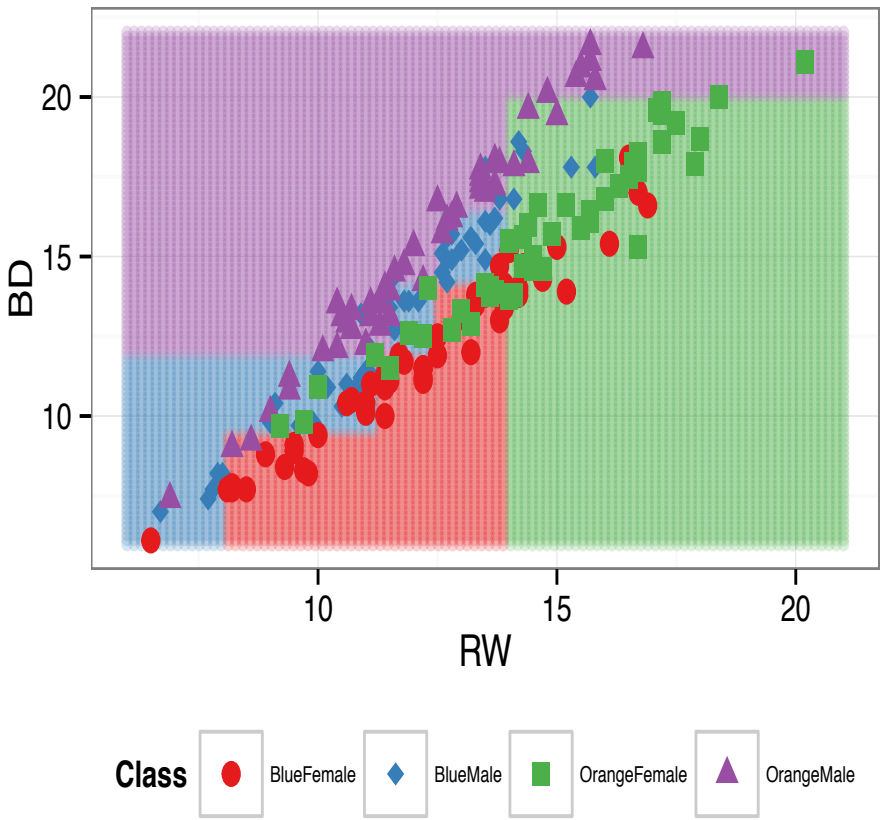
# Crab data

Measurements on rock crabs, 200 observations. 4 classes species-sex.

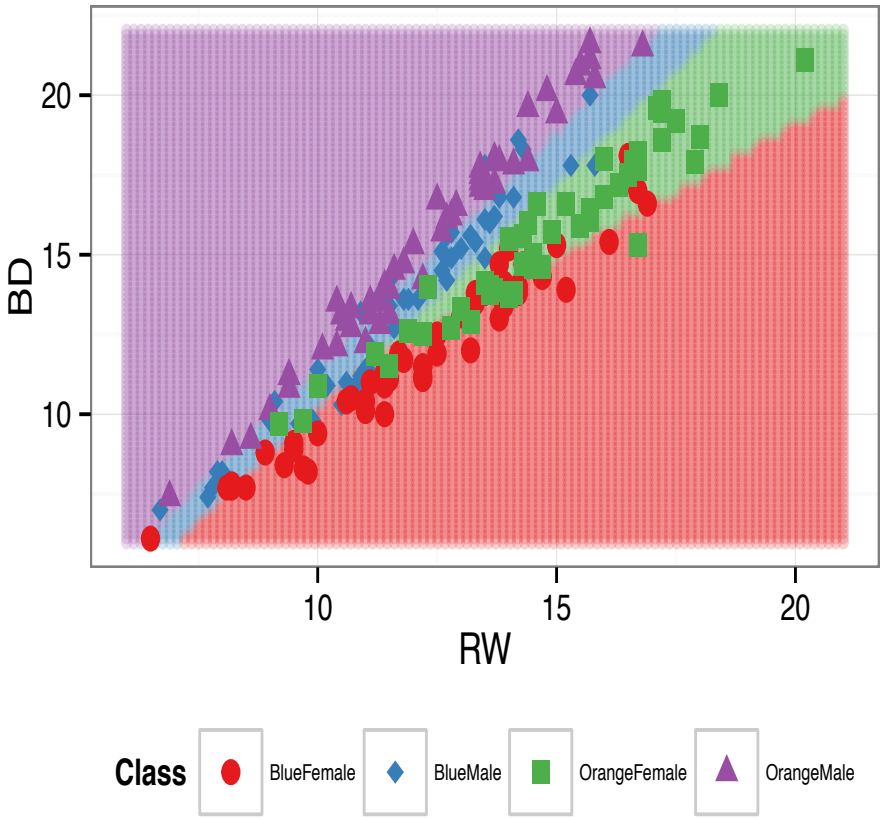
1. FL the size of the frontal lobe length, in mm
2. RW rear width, in mm
3. CL length of mid-line of the carapace, in mm
4. CW maximum width of carapace, in mm
5. BD depth of the body; for females, measured after displacement of the abdomen, in mm



## Decision boundaries from rpart




## Decision boundaries form PPtree



# Previous work and objective

1. Previous work
   - Oblique decision trees. Kim and Loh (2001), Brodley and Utgoff (1995), Tan and Dowe (2005), Truong (2009). In R `oblique.tree`.
   - Oblique random forest. Tan and Dowe (2006), Menze et al. (2011). In R `obliqueRF`.
2. Objective
   - `PPforest` implements a projection pursuit classification random forest
   - Adapts random forest to utilize combinations of variables in the tree construction
   - Projection pursuit classification trees are used to build the forest, ( from `PPtreeViz` package )

# Projection pursuit

Find interesting low-dimensional linear projections of high-dimensional data optimizing some specified function called the projection pursuit index.
Advantages:

- Able to bypass the curse of dimensionality, because work in low-dimensional linear projections.
- Relevant projection pursuit indexes are able to ignore irrelevant variables

We use LDA and PDA index in our PPforest.

# PPtree

Combines tree structure methods with projection pursuit dimension reduction.
`PPtree` projection pursuit classification tree:

1. In each node a PP index is maximized to find the optimal $1-D$ projection, $\alpha^*$, for separating all classes in the current data.
2. Reduce the number of classes to two, by comparing means and assign new labels, $G_1$ or $G_2$ ($y_i^*$) to each observation.
3. Re-do PP with these new group labels finding the $1-D$ projection, $\alpha$ using $(x_i, y^*)$.
4. Calculate the decision boundary c, keep $\alpha$ and $c$.
5. Separate data into two groups using new group labels $G_1$ and $G_2$.
6. For each group, stop if there is only one class else repeat the procedure, the splitting steps are iterated until the last two classes are separated.

# rpart vs PPtree

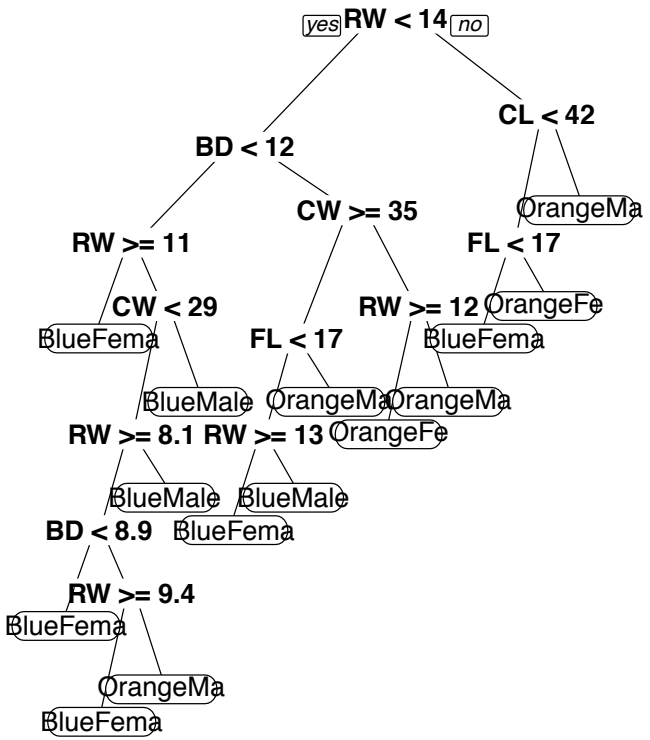



Figure : rpart tree using crab data

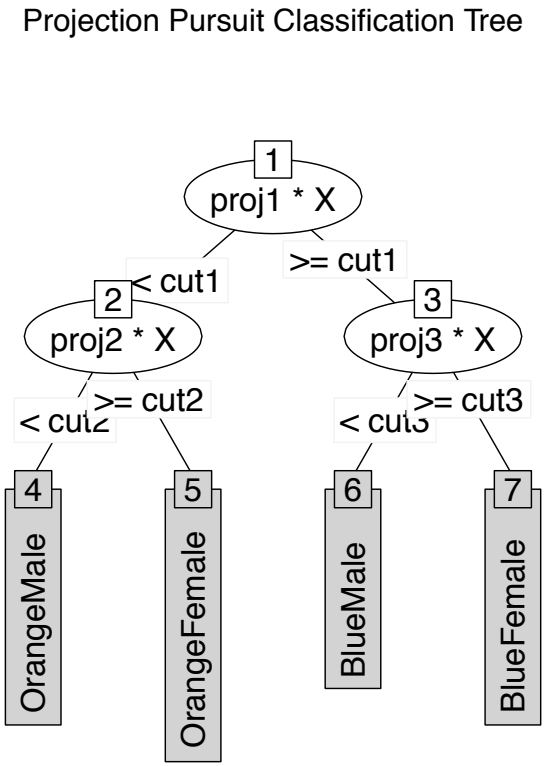



Figure : PPtree using crab data

# Features of PPtee

- Produces a simple tree.
- Uses association between variables to find separation.
- If a linear separation exists, `PPtree` produces a tree without misclassification.
- One class is assigned only to one final node, depth of the tree is at most the number of classes.
- The projection coefficients used to obtain the dimension reduction at each node can be used to determine the variable importance (variables are standardized).

# PPforest

A random forest is an ensemble learning method, built on bagged trees. `PPforest` conducts a supervised classification using projection pursuit trees and random forest ideas. Using this combination we are take into account the association between variables to find separation that is not considered in a classic random forest.

# PPforest algorithm

1. Input: $L = \{(x_i, y_i), i = 1, \ldots n\}, \quad y_i \in \{1, \ldots g\}$ where $y_i$ is the class information
2. Draw $b = 1 \ldots B$ bootstrap samples, $L^{*b}$ of size $n$ from $L$
3. For each bootstrap sample grow a `PPtree` classifier $T^{*b}$ and for every node a sample of m variables without replacement is drawn.
4. Predict the classes of each case not included in $L^*$ and compute the oob error.
5. Based on majority vote predict the class in a new data set.

# PPforest package

| | |
|---|---|
| `PPforest`: | Run a Projection pursuit random forest |
| `predict.PPforest`: | Vector with predicted values from a PPforest object |
| `ppf_importance`: | Plot a global measure of variable importance |
| `pproxy_plot`: | Proximity matrix visualization |
| `ppf_oob_error` | OOB error summary and visualization |
| `var_selec`: | Index id for variables set, sample variables without replacement with constant sample proportion. |
| `train_fn`: | Index id for training set, sample in each class with constant sample proportion. |
| `PPtree_split`: | Projection pursuit classification tree with random variable selection in each split |
| `trees_pp`: | Projection pursuit trees for bootstrap samples. |
| `ppf_bootstrap`: | Draws bootstrap samples with strata option. |
| `print.PPforest`: | Print PPforest object |

# Crab data

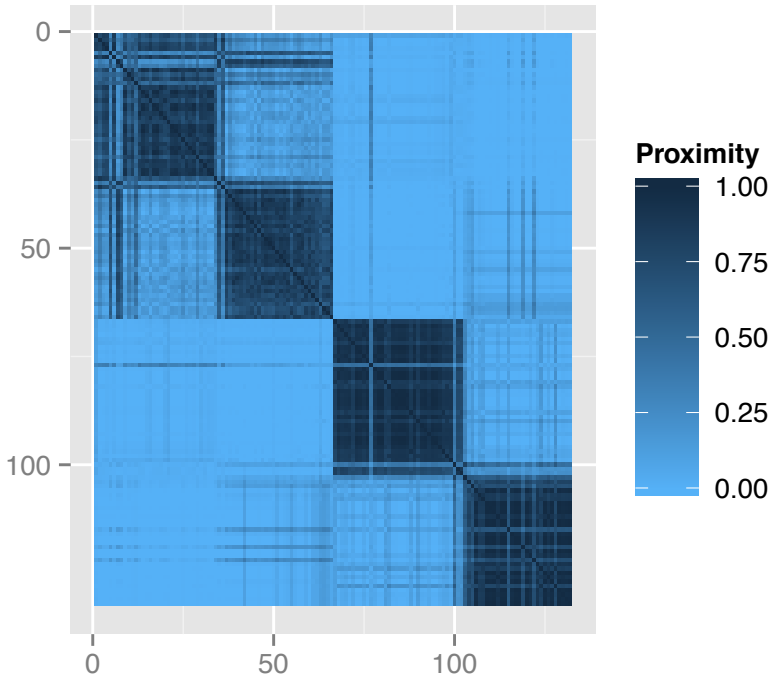
```
pprf.crab <- PPforest::PPforest(data = crab, size.tr = 2/3, m = 500, size.p = .7,
             PPmethod = 'LDA', strata = TRUE)
             str(pprf.crab, max.level = 1)
List of 19
 $ prediction.training: chr [1:132] "BlueMale" "BlueMale" "BlueMale" "BlueMale" ...
 $ training.error     : num 0.0303
 $ prediction.test    : chr [1:68] "BlueFemale" "BlueMale" "BlueMale" "BlueFemale" ...
 $ error.test         : num 0.0882
 $ oob.error.forest   : num 0.0303
 $ oob.error.tree     : num [1:500] 0.1591 0.3265 0.0204 0.1707 0.0536 ...
 $ boot.samp          :Classes grouped_df, tbl_df, tbl and 'data.frame':  132 obs. of  6
 $ output.trees       :Classes rowwise_df, tbl_df and 'data.frame': 500 obs. of  2 variables:
 $ proximity          :'data.frame': 8778 obs. of  3 variables:
 $ votes              : num [1:132, 1:4] 0.428 0.351 0.291 0.358 0.271 ...
 $ prediction.oob     : chr [1:132] "BlueMale" "BlueMale" "BlueMale" "BlueMale" ...
 $ n.tree             : num 500
 $ n.var              : num 3
 $ type               : chr "Classification"
 $ confusion          : num [1:4, 1:5] 32 1 0 0 1 32 0 0 0 0 ...
 $ call               : language PPforest::PPforest(data = crab, size.tr = 2/3, m = 500, PPmethod = "LDA"
 $ train              :'data.frame': 132 obs. of  6 variables:
 $ test               :'data.frame': 68 obs. of  5 variables:
 $ vote.mat           : chr [1:500, 1:132] "BlueFemale" "BlueFemale" "BlueMale" "BlueMale" ...
 - attr(*, "class")= chr "PPforest"
```

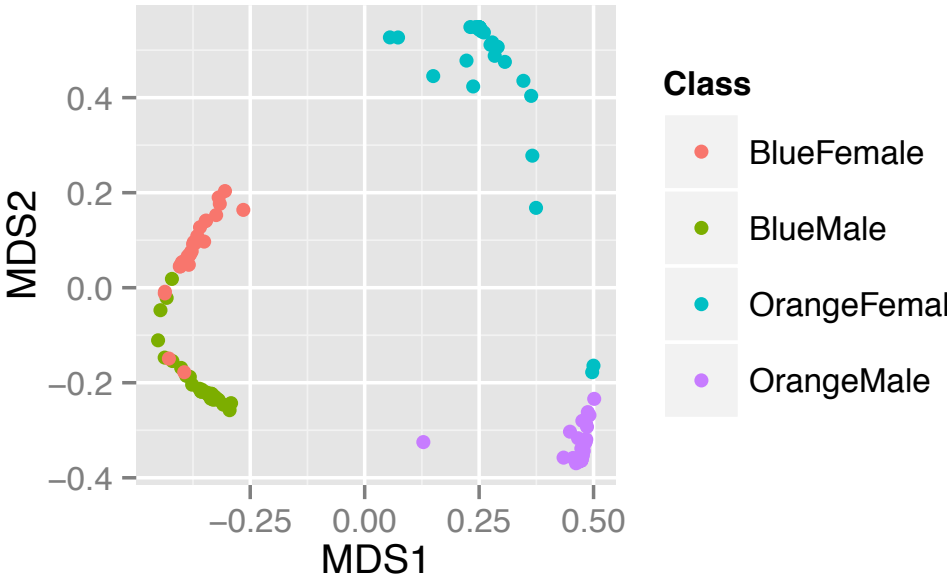
# Comparison to randomForest

```
rf.crab
Call:
 randomForest(formula = Type ~ ., data = crab,
              proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 20.5%
Confusion matrix:
             BF  BM  OF  OM class.error
BF           41   3   6   0        0.18
BM            3  39   0   8        0.22
OF            4   0  41   5        0.18
OM            2   5   5  38        0.24
```

```
pprf.crab
Call:
 PPforest::PPforest(data = crab, size.tr = 1, m = 500,
 PPmethod = "LDA", size.p = 0.5, strata = TRUE)
               Type of random forest: Classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 6%
Confusion matrix:
             BF  BM  OF  OM class.error
BF           48   2   0   0        0.04
BM            5  45   0   0        0.10
OF            0   0  45   3        0.10
OM            0   1   0  50        0.00
```

# PPforest visualization



# Importance measure

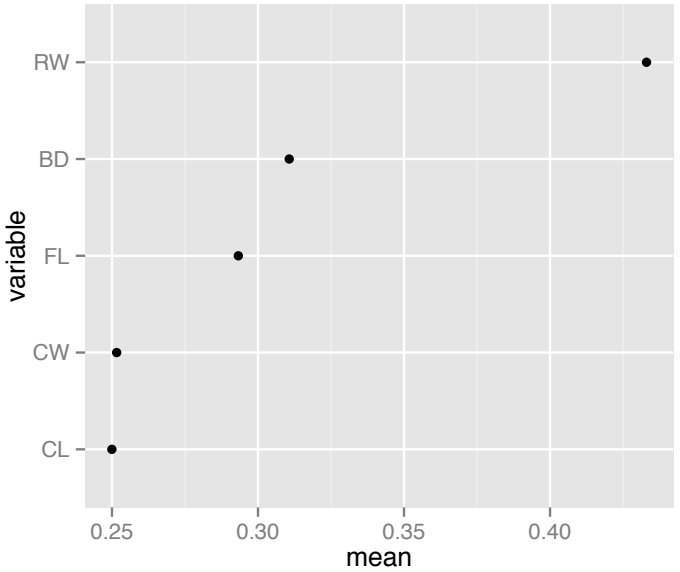
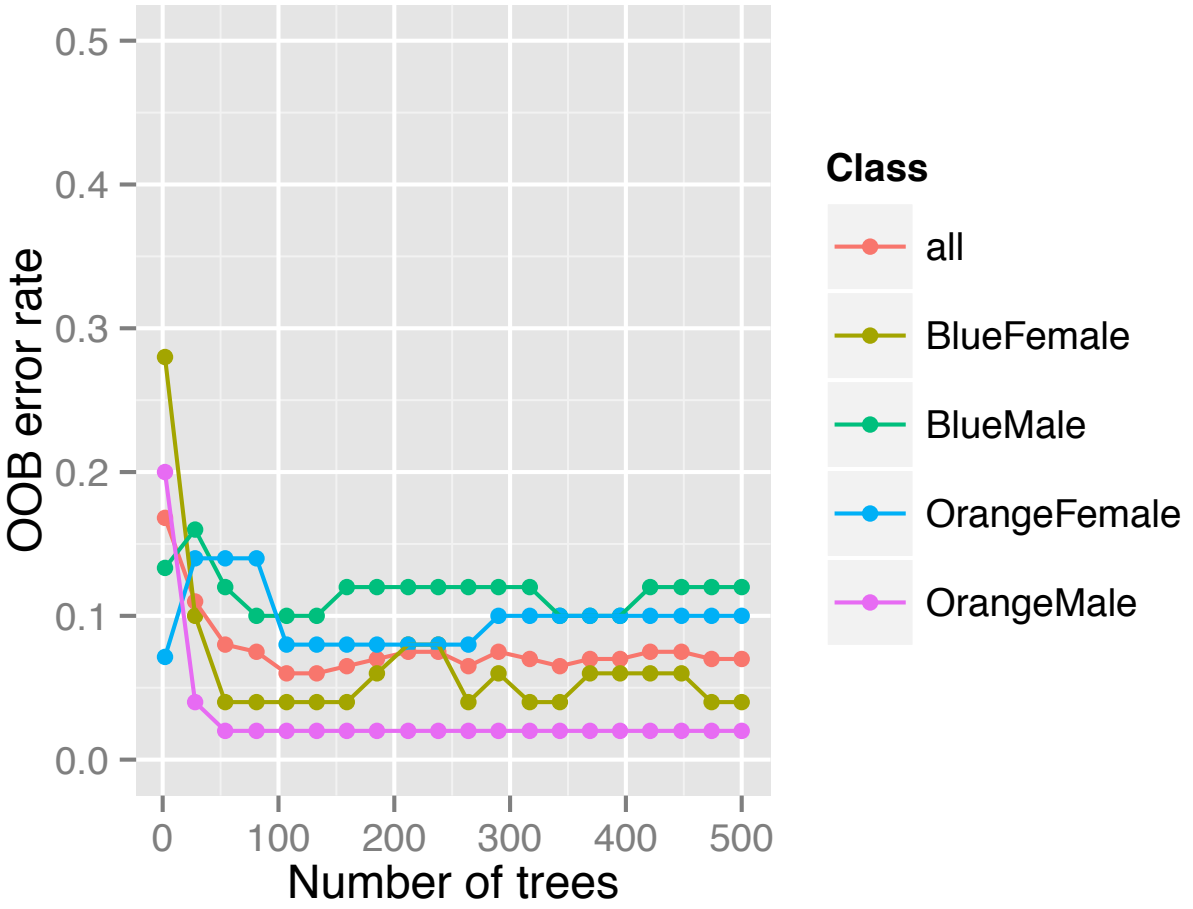



Figure : PPforest global importance

Weighted mean of the absolute value of the projection coefficients across all nodes in every tree. The weights are the projection pursuit index in each node, and 1-(the out of bag error of each tree).

# Cumulative oob error visualization



# Prediction accuracy comparison

Table : Comparison of PPtree, CART, random forest and PPforest results with various data sets. The mean of training and test error rates from 200 re-samples is shown

| | TRAIN | | | | TEST | | | |
|---|---|---|---|---|---|---|---|---|
| | PPtree | Cart | RF | PPforest | PPtree | Cart | RF | PPforest |
| Crab | 0.04 | 0.27 | 0.21 | 0.05 | 0.06 | 0.45 | 0.31 | 0.04 |
| Leuke. | 0.01 | 0.04 | 0.06 | 0.00 | 0.05 | 0.15 | 0.00 | 0.00 |
| Lymph. | 0.03 | 0.05 | 0.09 | 0.00 | 0.07 | 0.17 | 0.04 | 0.04 |
| NCI60 | 0.06 | 0.46 | 0.45 | 0.00 | 0.48 | 0.75 | 0.33 | 0.19 |
| Wine | 0.00 | 0.05 | 0.03 | 0.01 | 0.02 | 0.12 | 0.00 | 0.00 |
| Glass | 0.31 | 0.24 | 0.25 | 0.27 | 0.42 | 0.34 | 0.18 | 0.32 |
| Fish. | 0.00 | 0.14 | 0.20 | 0.00 | 0.02 | 0.23 | 0.26 | 0.02 |
| Image | 0.07 | 0.07 | 0.02 | 0.06 | 0.07 | 0.08 | 0.03 | 0.07 |
| Parki. | 0.12 | 0.08 | 0.11 | 0.12 | 0.18 | 0.16 | 0.09 | 0.18 |

# Comments

1. `PPforest` uses the association between variables to find separation.
2. The strength of each individual tree in the forest increases when classes are linearly separable, smaller error rate in the forest.
3. The predictive performance is better than other classifiers for some of the analyzed data.

# Further work

1. Improve the performance of the algorithm.
2. Incorporate additional project pursuit indexes.
3. Improve the visualization.
4. Work in regression PPforest.

# References I

Brodley, C. E. and Utgoff, P. E. (1995), "Multivariate decision trees," *Machine learning*, 19, 45–77.

Kim, H. and Loh, W.-Y. (2001), "Classification trees with unbiased multiway splits," *Journal of the American Statistical Association*, 96.

Menze, B. H., Kelm, B. M., Splitthoff, D. N., Koethe, U., and Hamprecht, F. A. (2011), "On oblique random forests," in *Machine Learning and Knowledge Discovery in Databases*, Springer, pp. 453–469.

Tan, P. J. and Dowe, D. L. (2005), "MML inference of oblique decision trees," in *AI 2004: Advances in Artificial Intelligence*, Springer, pp. 1082–1088.

— (2006), "Decision forests with oblique decision trees," in *MICAI 2006: Advances in Artificial Intelligence*, Springer, pp. 593–603.

Truong, A. (2009), "Fast Growing and Interpretable Oblique Trees via Probabilistic Models," *Univ. of Oxford, A thesis submitted for the degree of Doctor of Philosophy, Trinity term*, 1–119.