Pattern recognition on spatial data

Predicting structure : other classification methods

Yves Brostaux

June 2017

Contents

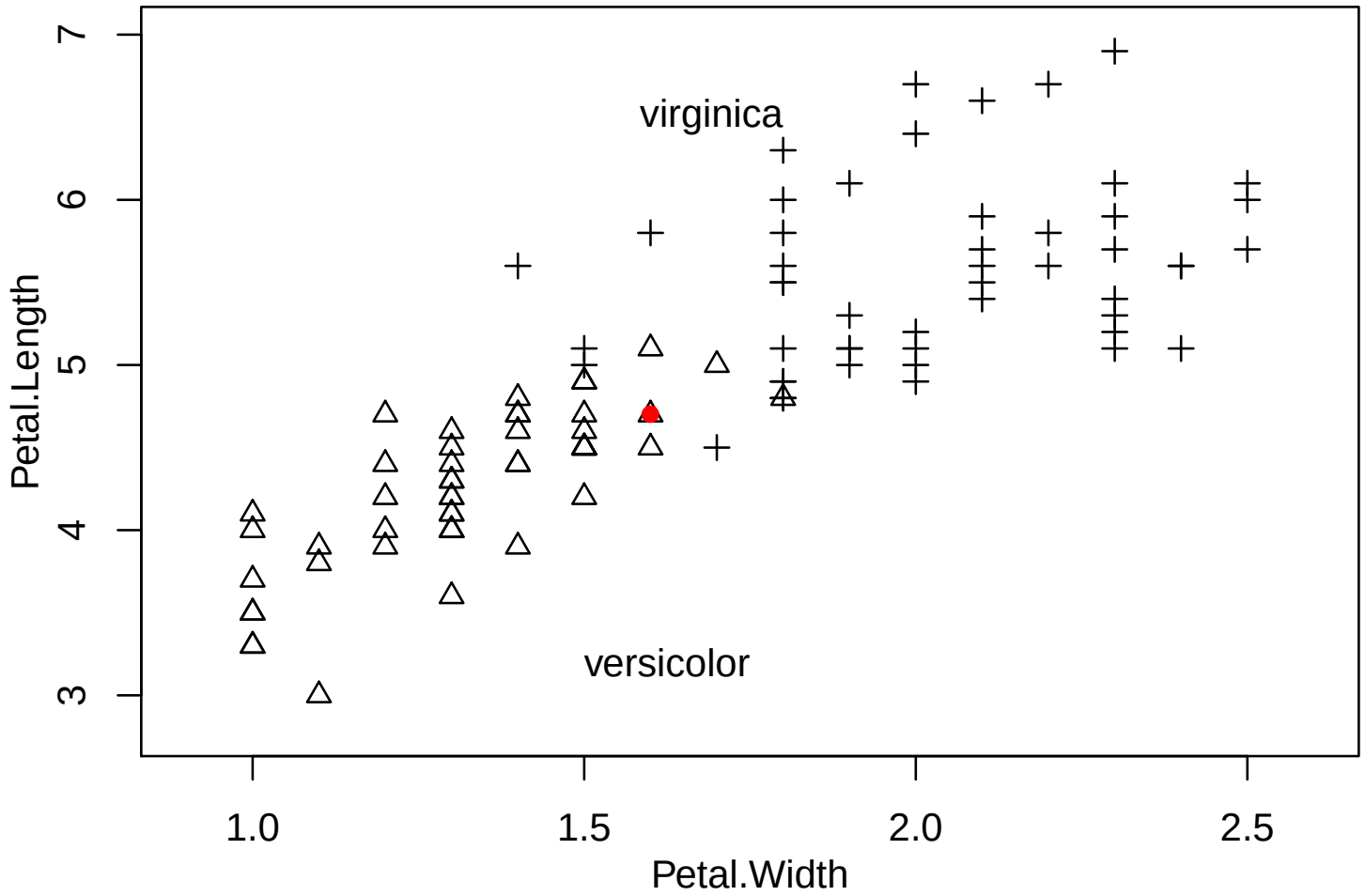
Basics

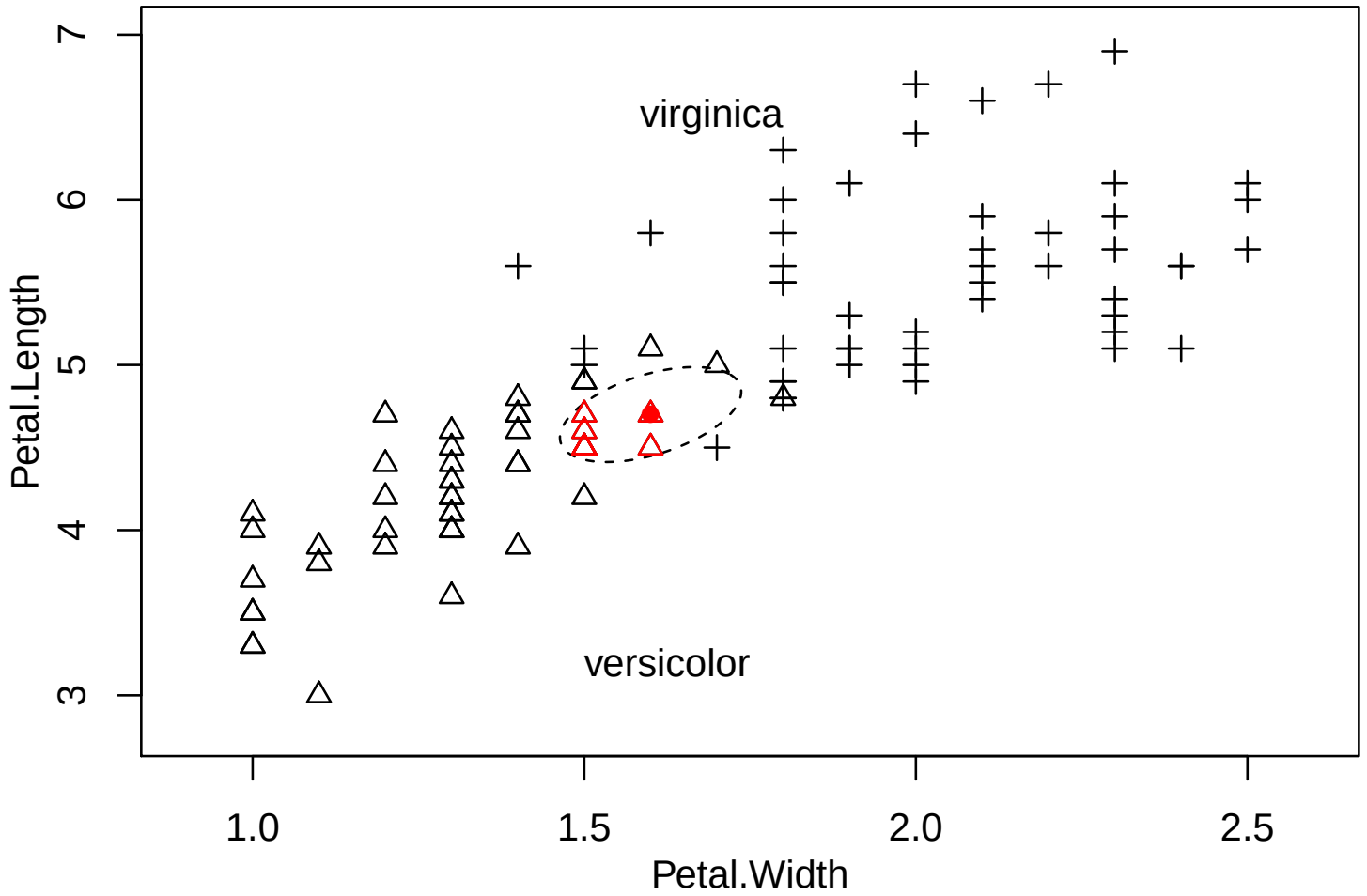
Classification rule

Affect observation i to the population h which is the most represented amongst the r nearest neighbours of this observation

Distance

- euclidian
- Mahalanobis'


virginica
versicolor
Petal.Length
Petal.Width
3
4
5
6
7
1.0
1.5
2.0
2.5


virginica
versicolor
Petal.Length
Petal.Width
3
4
5
6
7
1.0
1.5
2.0
2.5

Pros/cons

- **Pros**
  - no assumptions about distributions
  - easily adapt to complex concepts
- **Cons**
  - no model $\Rightarrow$ need a new computation for each new prediction
  - can be slow for very high number of individuals

Example with R

```
library(class)

# separate data set into training and test sets
set.seed(123)
train <- sample(1:150, 125, replace=FALSE)
iris.trn <- iris4[train,]
iris.tst <- iris4[-train,]

# predict class for test set
cltest <- knn(train=iris.trn[,1:2],
              test=iris.tst[,1:2],
              cl=iris.trn$Species, k = 5)
```

Example with R

```
# confusion matrix
knn.cm <- table(iris.tst$Species, cltest)
knn.cm
```

```
##                cltest
##                 setosa versicolor virginica
##    setosa            7          0         0
##    versicolor        0          8         0
##    virginica         0          1         9
```

```
# actual error rate
1 - sum(diag(knn.cm))/sum(knn.cm)
```

```
## [1] 0.04
```

Contents

Basics

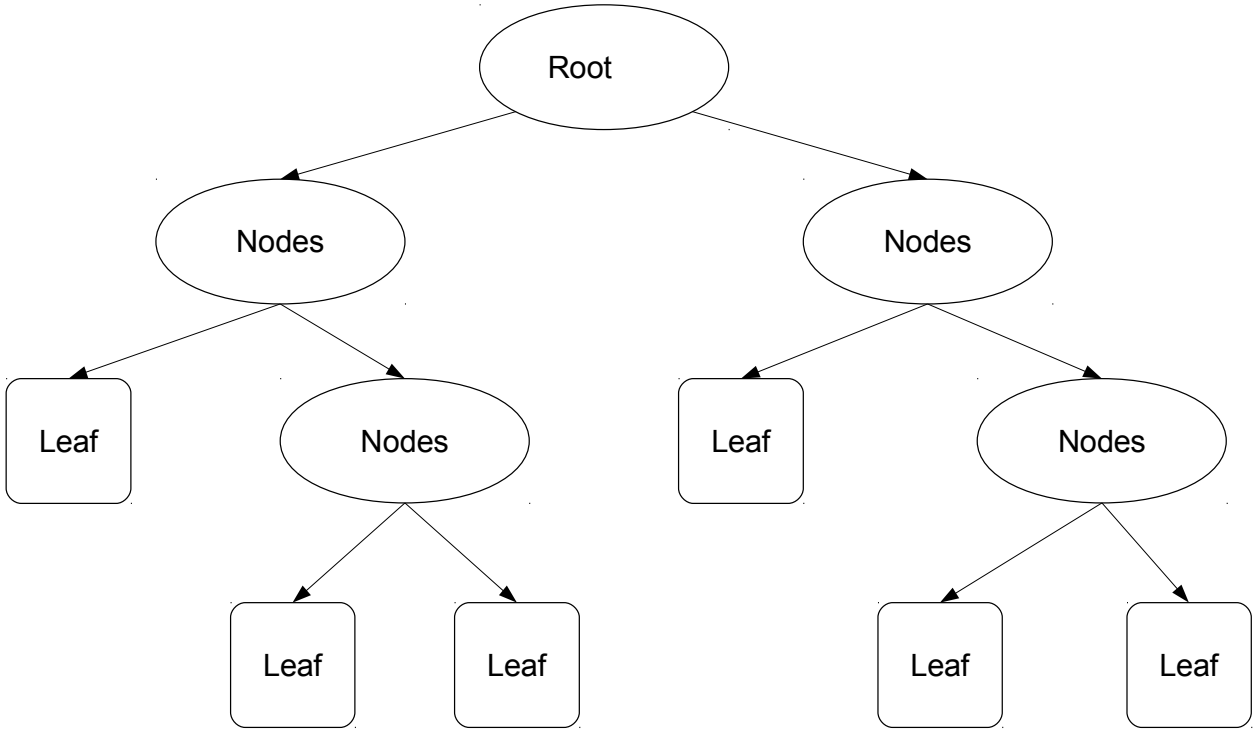
Construction of the classifier

- Recursively partition the observations into groups of increasing homogeneity regarding their populations' distribution.

= decision tree

Classification rules

- Follow the decision path from the *root* of the tree to its *leaves*
- Assign observation to the population with the highest estimated posterior probability


Root
Nodes
Nodes
Leaf
Nodes
Leaf
Nodes
Leaf
Leaf
Leaf
Leaf

Methods

- Number of choices at each node
- Type of decision (univariate, multivariate)
- Homogeneity criteria
- Stopping rule

CART method

- Dichotomic univariate choices
- Homogeneity/splitting criteria : Entropy/Shannon's Index or Gini's index

Spliting criteria

Entropy or Shannon's index

$$I_{Shannon}(E) = \sum_{j=1}^{g} -\frac{n_{j.}}{n_{..}} \log_2 \frac{n_{j.}}{n_{..}}$$

Gini's index

$$I_{Gini}(E) = \sum_{j=1}^{g} \frac{n_{j.}}{n_{..}}(1 - \frac{n_{j.}}{n_{..}})$$

Choose the split which maximize the information gain :

$$gain(E, A) = \Delta I = I(E) - \sum_{i=1}^{p} \frac{n_{.i}}{n_{..}} I(E_i)$$

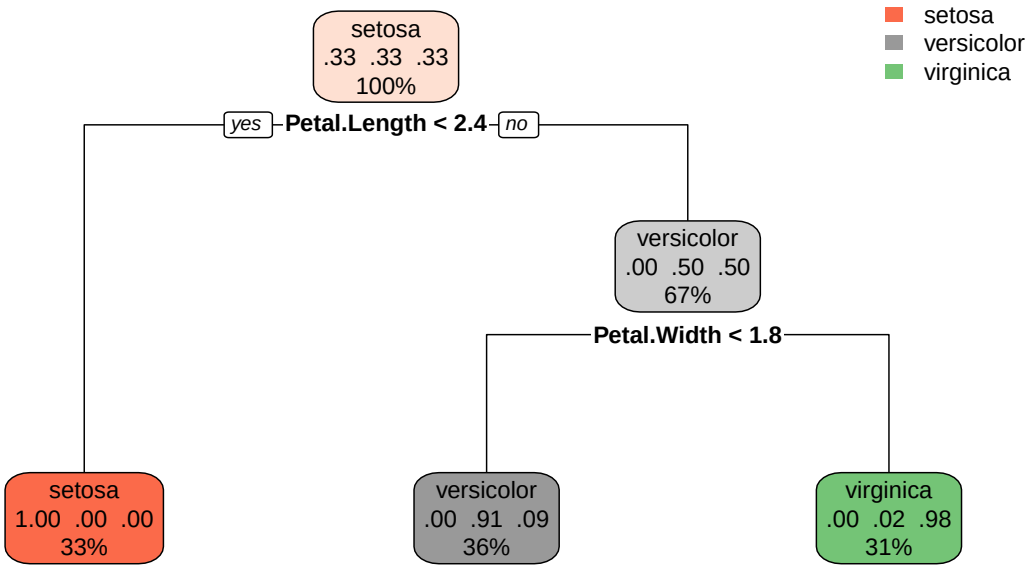
Example with R

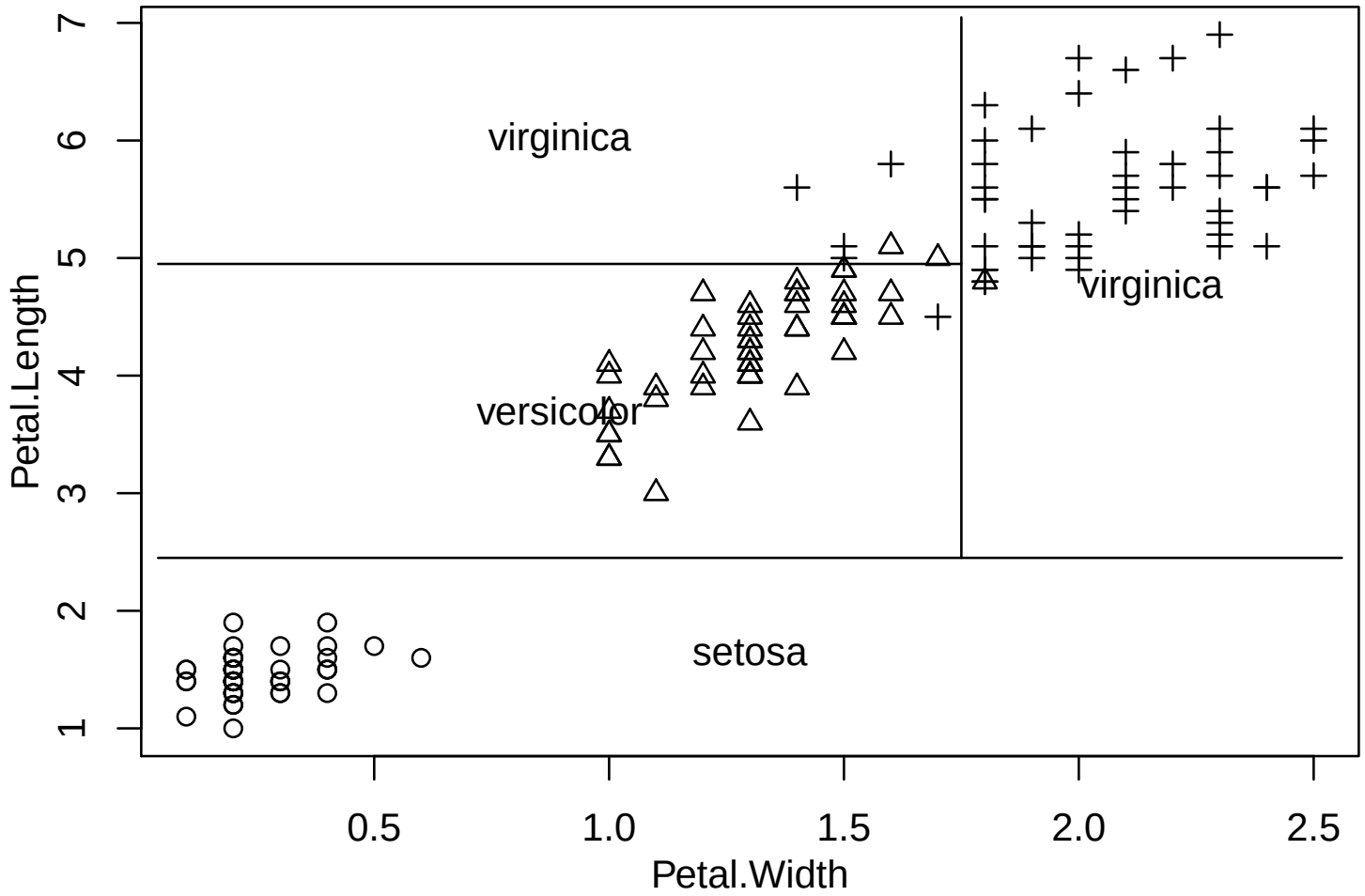
```
library(rpart)
iris.rp <- rpart(Species~., data=iris)
iris.rp
```

```
## n= 150
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)
##   2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000
##   3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.5000
##     6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.9074
##     7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173
```

Plotting the tree

```
library(rpart.plot)
rpart.plot(iris.rp)
```




virginica
versicolor
setosa
virginica
Petal.Length
Petal.Width
1
2
3
4
5
6
7
0.5
1.0
1.5
2.0
2.5

Optimal size

Fully grown trees tend to overfit data (produce non significant splits), which lower global the prediction performance.

- Stopping rules
  - minimal size
  - minimal gain
  - significance test (χ^2)
- Pruning
  - cut parts of a full grown tree to improve expected error
  - based on penalties on error

rpart R command

```
rpart(formula, data, parms, control, ...)
```

formula : $Y \sim X_1 + X_2 + \ldots + X_p$
parms : list with

- prior component (vector of prior probabilities)
- split component (splitting criteria, gini or information)

control : list with

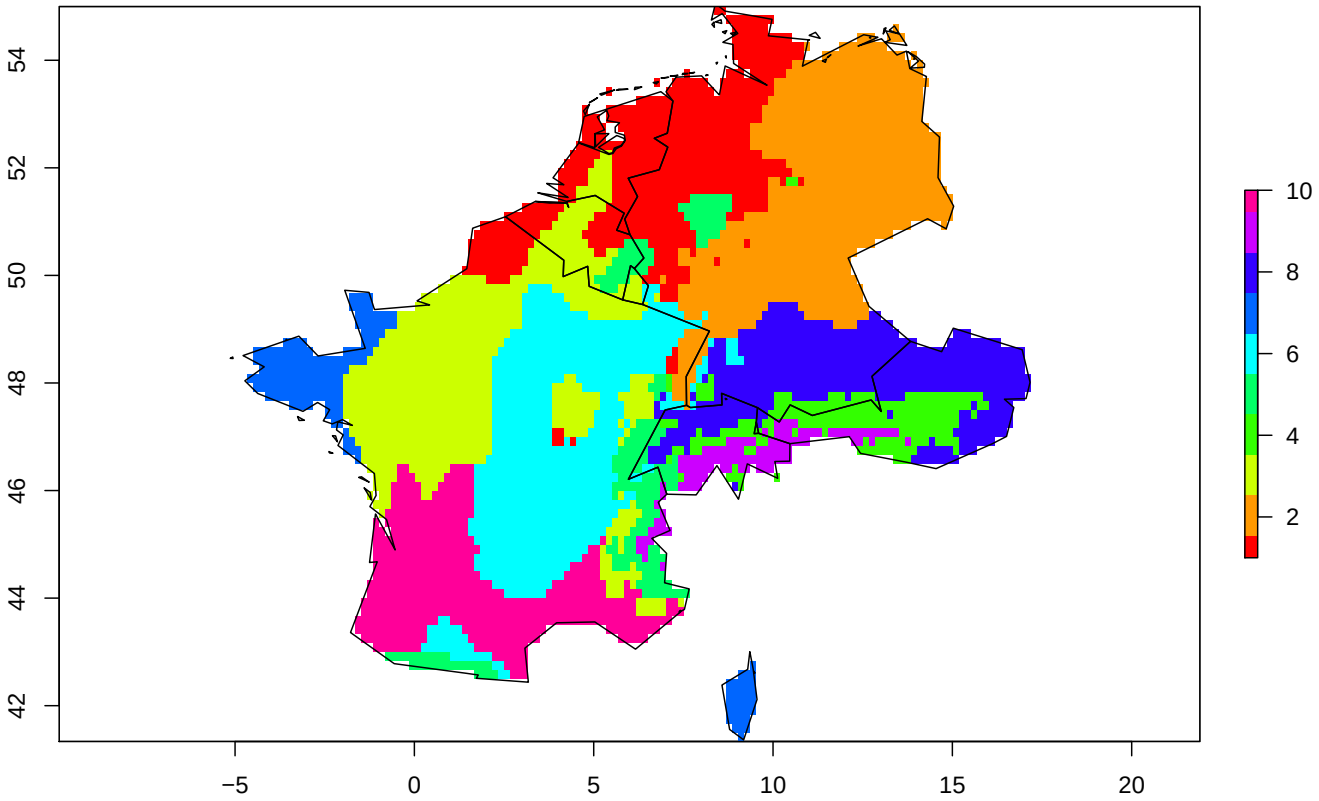
- minsplit minimum number of observations in a node to split it
- minbucket minimum number of observations in any terminal node
- cp complexity parameter. minimum improvement of the splitting criteria to attempt a split

Bigger example : back to climatic typology

```
# get worldwide bioclimatic data
library(raster)
wclim <- getData('worldclim', res=10, var='bio', path="../")

# loading countries borders
library(maptools)
data(wrld_simpl) # simplified world contries borders
# selection of west european region
eur_simpl <- wrld_simpl[wrld_simpl@data$SUBREGION==155,]

# crop climatic data for western europe
climEur <- crop(wclim, bbox(eur_simpl))
climEur <- mask(climEur, eur_simpl)

# extract data for further analysis
climdat <- getValues(climEur)
climdat <- na.omit(as.data.frame(climdat))
```

Bigger example : back to climatic typology

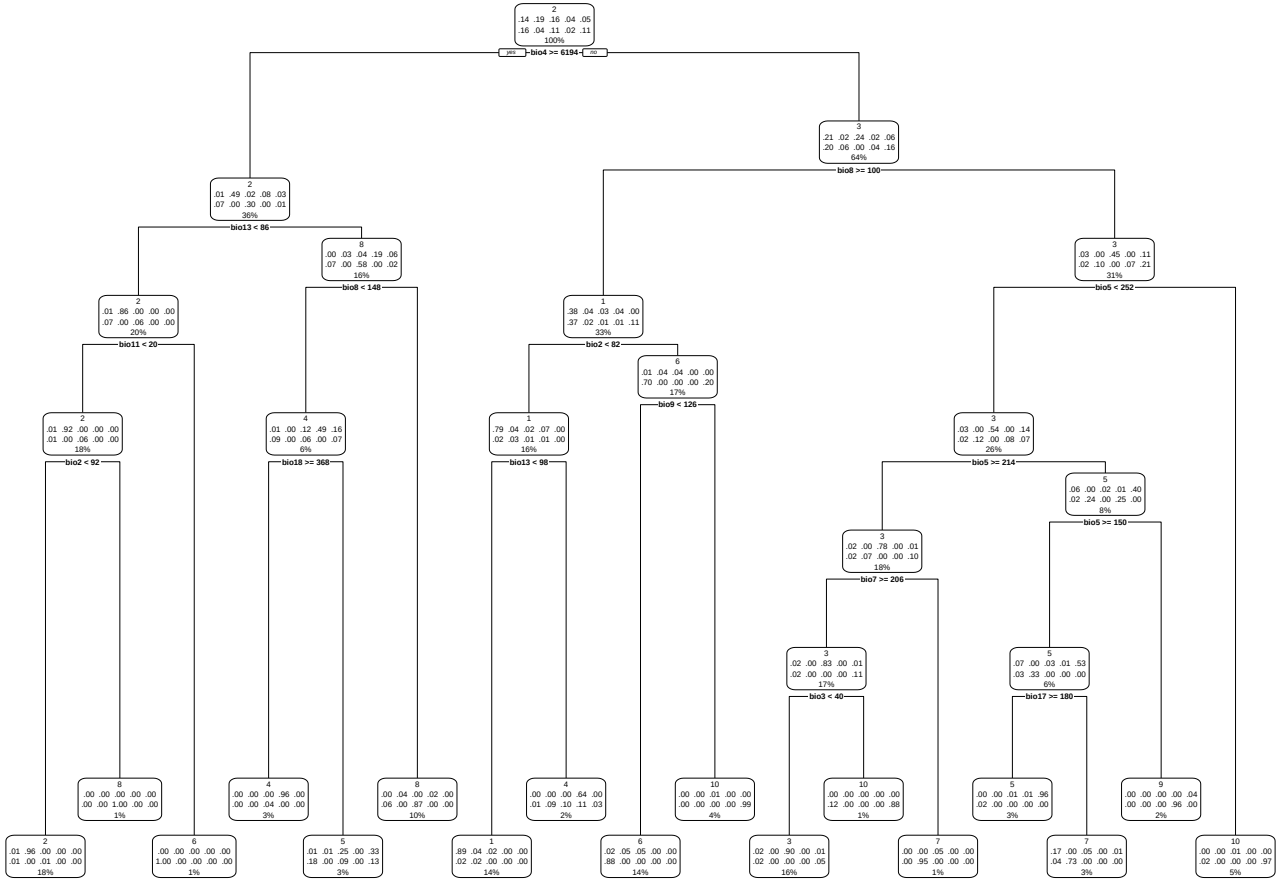
```
climClust <- raster("climTypo10.tif")

# extract groups data
groups <- getValues(climClust)
groups <- na.omit(groups)

plot(climClust, col=rainbow(length(unique(groups))))
plot(eur_simpl, add=TRUE)
```


54
52
50
48
46
44
42
−5
0
5
10
15
20
10
8
6
4
2

```
clim.rp <- rpart(as.factor(groups)~., data=climdat)

rpart.plot(clim.rp)
```

```
# resubstitution confusion matrix
clim.pred <- predict(clim.rp, type="class")
table(groups, clim.pred)
```

```
##        clim.pred
## groups   1   2   3   4   5   6   7   8   9  10
##      1 595   9  17   0   2  12  22   0   0   0
##      2  29 813   0   0   1  31   0  22   0   0
##      3  15   1 666   0  35  32  10   0   0   3
##      4   0   0   0 190   2   2   0  11   0   0
##      5   0   0   9   0 201   2   1   0   4   0
##      6  13  11  12   1  27 636   5  32   0  12
##      7  16   0   2   8   0   0 151   0   0   0
##      8   0   9   0  14  12   1   0 482   0   0
##      9   0   0   0  10   0   0   0   0 100   0
##      10  0   0  38   3  18   0   0   0   0 449
```


bio4 >= 6194
yes
no
bio13 < 86
bio8 >= 100
bio8 < 148
bio11 < 20
bio2 < 82
bio5 < 252
bio2 < 92
bio18 >= 368
bio13 < 98
bio9 < 126
bio5 >= 214
bio5 >= 150
bio7 >= 206
bio3 < 40
bio17 >= 180

Pruning the tree

Cost-complexity pruning evaluate the performance of sequences of (sub)trees, defined by an increasing cost-complexity penalty α.

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|$$

with $R(T)$ the resubstitution error of the tree T and $|\tilde{T}|$ the number of terminal nodes.

For each α the algorithme search the subtree with the minimum $R_\alpha(T)$ and estimate by cross validation the associated error.

The cost-complexity value associated with the **minimal CV error** is then used to prune the initial tree to a smaller one with a better expected error rate.

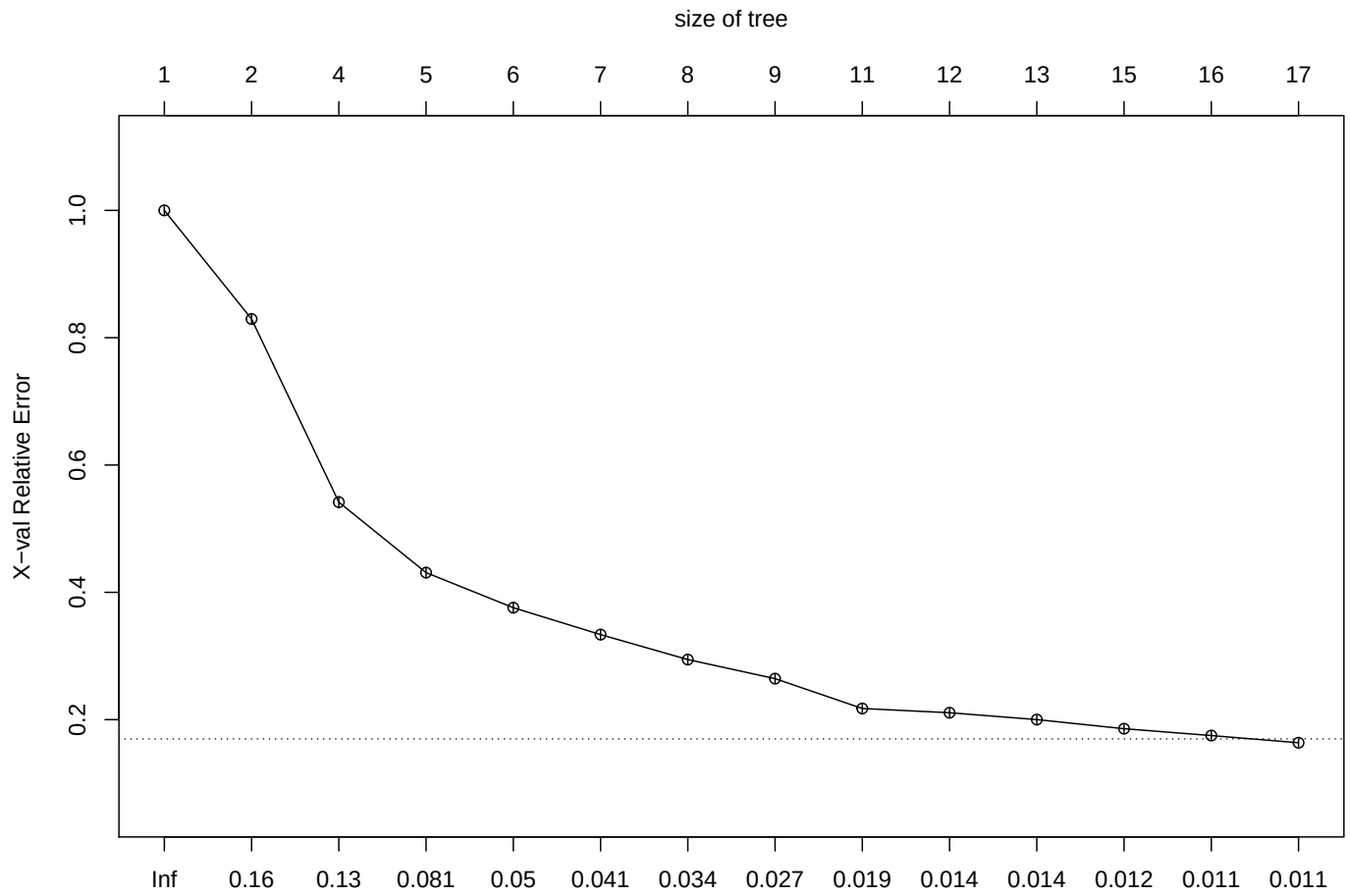
Cost-complexity pruning

```
printcp(clim.rp)
```

```
##
## Classification tree:
## rpart(formula = as.factor(groups) ~ ., data = climdat)
##
## Variables actually used in tree construction:
##  [1] bio11 bio13 bio17 bio18 bio2  bio3  bio4  bio5  bio7  bio8  bio
##
## Root node error: 3903/4799 = 0.81329
##
## n= 4799
##
##          CP nsplit rel error  xerror      xstd
## 1  0.170894      0   1.00000 1.00000 0.0069164
## 2  0.144248      1   0.82911 0.82936 0.0083164
## 3  0.110684      3   0.54061 0.54163 0.0088115
## 4  0.058673      4   0.42993 0.43095 0.0084685
```

Cost-complexity pruning

```
plotcp(clim.rp)
```



Pros & cons

Pros

- Quick even on big data sets
- Easily readable and interpretable
- Classification rules $\rightarrow$ simple logical rules

Cons

- Instability $\Rightarrow$ Random forests

Contents

Basics

CART tree node selection is sensitive to variation in the training data.

Instead of pruning the tree to remove this instability, Random Forests promote this instability by using **resampling of the sample and of the attributes** to build multiple tree predictors (a forest) which are **pooled to increase the robustness** of the prediction.

Algorithm

Starting with a training sample with n observations, p descriptive attributes and a class variable Y

1. For each tree, a bootstrap sample of the original data is used
2. At each node, the attributes selection starts with a random choice of *mtry* attributes, followed by a classical selection based on the partition performance
3. All this is repeated to build *ntree* trees

Prediction is done by aggregating the results of the *ntree* trees (mean for quantitative Y, mode for qualitative Y).

Example with R

```
library(randomForest)
clim.rf <- randomForest(as.factor(groups)~.,
                        importance=TRUE,
                        ntree=1000,
                        data=climdat)

clim.rf
```

```
##
## Call:
##  randomForest(formula = as.factor(groups) ~ ., data = climdat,
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 1.31%
## Confusion matrix:
##      1   2   3   4   5   6   7   8   9  10 class.error
## 1  645   7   2   0   0   3   0   0   0   0 0.018264840
## 2    1 891   0   0   0   1   0   3   0   0 0.005580357
## 3    2   0 752   0   4   1   1   0   0   2 0.013123360
## 4    0   0   0 200   0   0   0   3   2   0 0.024390244
## 5    0   0   3   0 212   0   0   0   2   0 0.023041475
## 6    4   2   0   0   1 739   0   2   0   1 0.013351135
## 7    0   0   1   0   0   0 176   0   0   0 0.005649718
## 8    0   4   0   3   0   1   0 510   0   0 0.015444015
## 9    0   0   0   1   0   0   0   0 109   0 0.009090909
## 10   0   0   6   0   0   0   0   0   0 502 0.011811024
```

Pros & cons

Pros

- Lose the simplicity and readability of single trees
- can be computer intensive for big datasets

Cons

- better robustness and prediction performance
- new information available
  - out-of-bag (OOB) error
  - variable importance

Out-of-bag (OOB) error

Each tree of the random forest is build with a training set obtained by bootstrapping (sample of the same size with replacement) the originial dataset.

In such sample, some observations are present more than one time, and conversely some observations are absent (out-of-bag) from the bootstrap sample (about 33% on average)

Those OOB observations are used as a test set for the corresponding tree to estimate the OOB error rate.

Variable importance

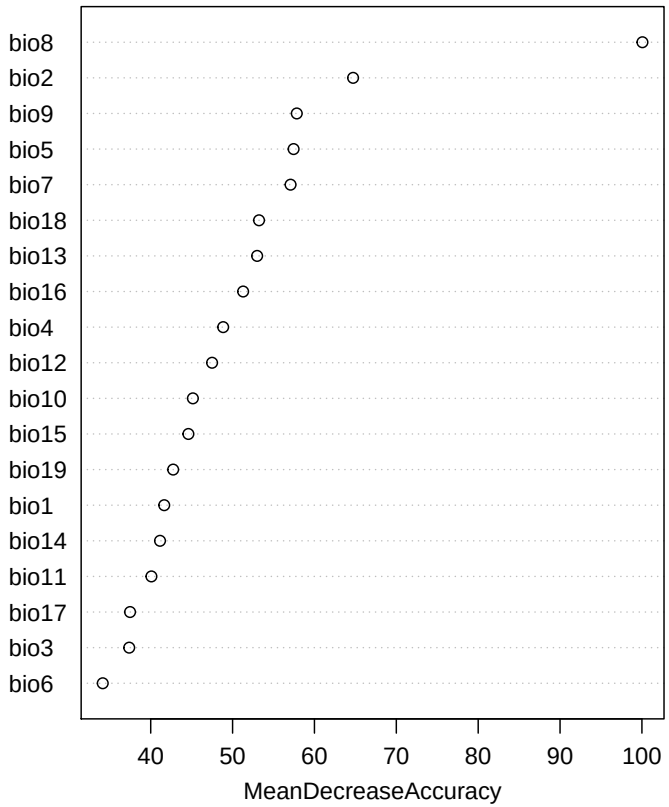
The importance (influence) of a variable in the RF prediction process is estimated by two different methods

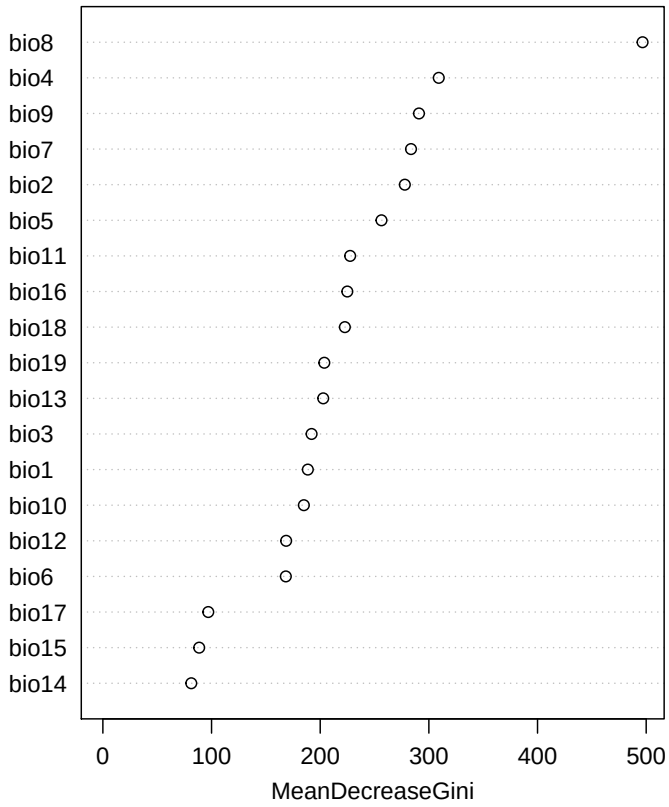
- by comparing the OOB performance on the original data and a sample where the value of the variable are randomly permutated (*MeanDeacreaseAccuracy*);
- by summing all the contribution of the variable to the decrease of the splitting criteria (*MeanDecreaseGini*)

Example with R

```
# need to use to argument importance=TRUE
# while building the random forest
varImpPlot(clim.rf)
```

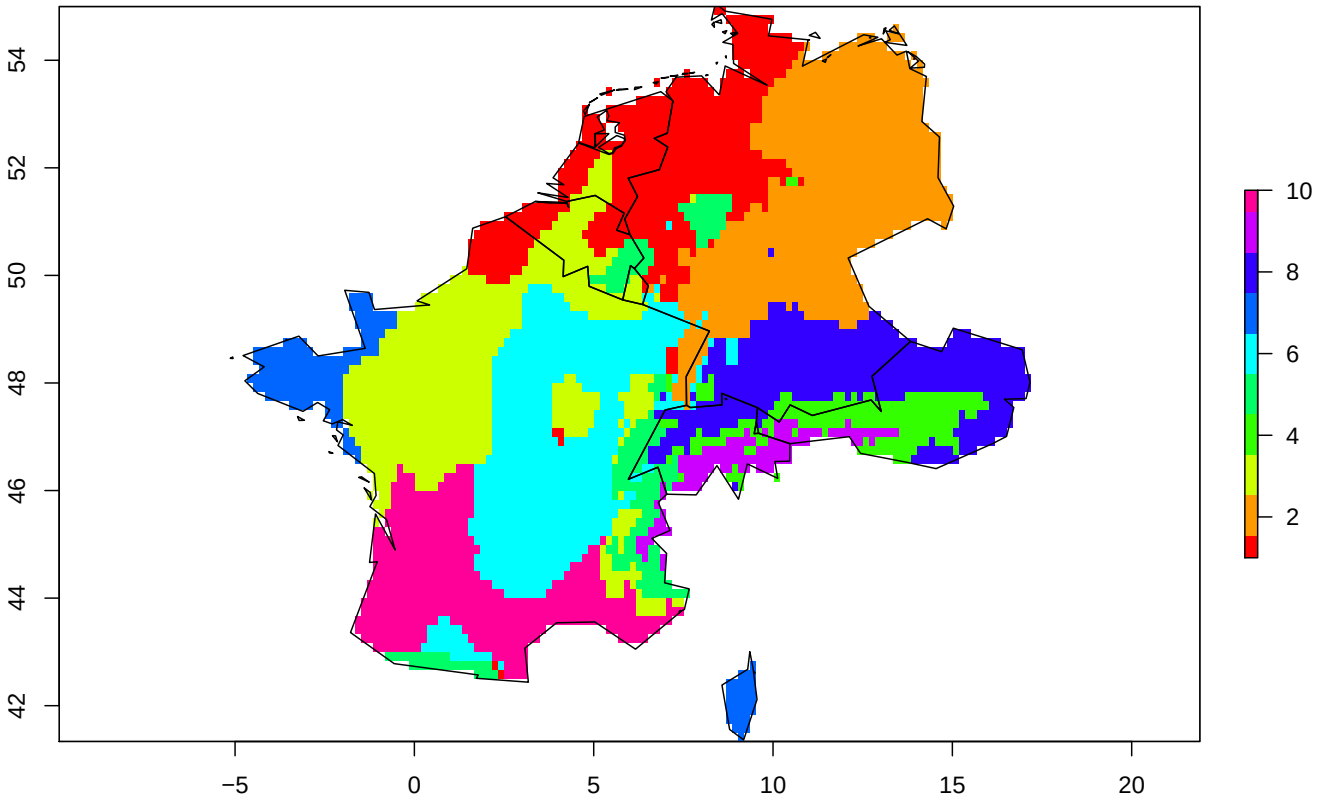
clim.rf


bio8
bio2
bio9
bio5
bio7
bio18
bio13
bio16
bio4
bio12
bio10
bio15
bio19
bio1
bio14
bio11
bio17
bio3
bio6
40
50
60
70
80
90
100
MeanDecreaseAccuracy


bio8
bio4
bio9
bio7
bio2
bio5
bio11
bio16
bio18
bio19
bio13
bio3
bio1
bio10
bio12
bio6
bio17
bio15
bio14
0
100
200
300
400
500
MeanDecreaseGini

Back to spatial

```
climRF <- raster(climEur)
values(climRF)[-nai] <- predict(clim.rf)
plot(climRF, col=rainbow(length(unique(groups))))
plot(eur_simpl, add=TRUE)
```


54
52
50
48
46
44
42
−5
0
5
10
15
20
10
8
6
4
2

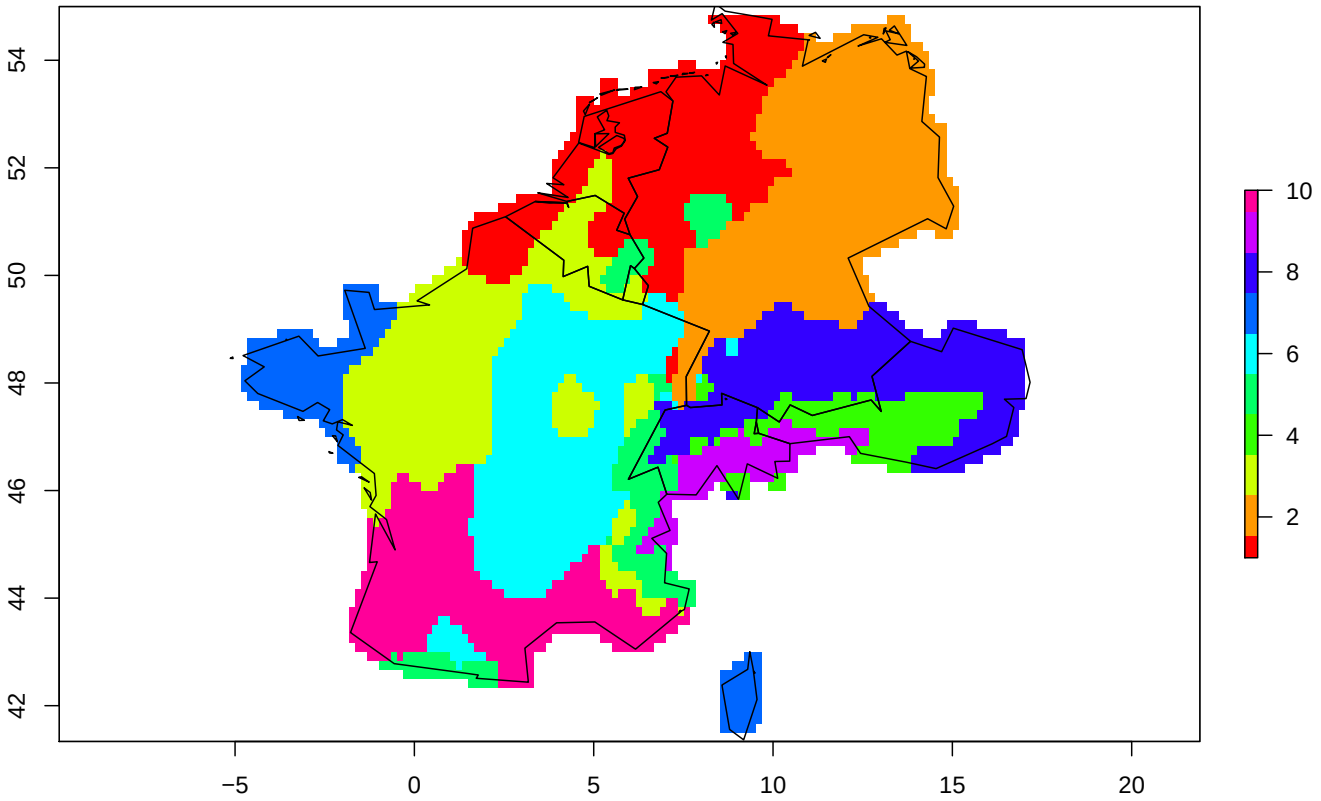
Contents

Spatial Smoothing of the prediction

As classical classification methods don't use spatial information to predict the classes, you can get local artefacts (lone pixel of one class surrounded by another class), especially in noisy environment.

You can process the raw prediction a posteriori to limit those artifacts by a spatial smoothing. A spatial smoothing apply a function to a defined neighborhood of a point to restimate the value of the point

```
# definition of the smoothing window
# 3 x 3 square, constant weight
windw <- matrix(rep(1, 9), nrow=3)

# take the majority class in the window for each pixel
climRF.sm <- focal(climRF, w=windw, fun=modal,
                   na.rm=T, pad=F)
plot(climRF.sm, col=rainbow(length(unique(groups))))
plot(eur_simpl, add=TRUE)
```


54
52
50
48
46
44
42
−5
0
5
10
15
20
10
8
6
4
2