MSA220/MVE440 Statistical Learning for Big Data

Lecture 1

Rebecka Jörnsten

Mathematical Sciences
University of Gothenburg and Chalmers University of Technology

Statistical Learning for Big Data

Image from "Statistical Truisms", Kirk Borne, 2013

Larry Wasserman, 2013 (partial quote):
"Data Science: The End of Statistics?
As I see newspapers and blogs filled with talk of Data Science and Big Data I find myself filled with a mixture of optimism and dread. Optimism, because it means statistics is finally a sexy field. Dread, because statistics is being left on the sidelines.
The very fact that people can talk about data science without even realizing there is a field already devoted to the analysis of data - a field called statistics - is alarming.

I like what Karl Broman says:

> *"When physicists do mathematics, they don't say they're doing number science. They're doing math.*
>
> *If you're analyzing data, you're doing statistics. You can call it data science or informatics or analytics or whatever, but it's still statistics."*

Well put."

Big Data

- BIG DATA: can't fit on a HD
- Big Data: 10Gb+1Tb
- big data: ¡10Gb, can work on a laptop.

MSA220/MVE440

- We will work with "big data" in this class, that is data that you can work with on a "PC".
- Emphasis will be on statistical methods that scale to at least Big Data (Tb).

Myth about Big Data

- Makes statistics obsolete since uncertainties of estimation are reduced to 0.

Reality

- When the sample size grows, we often see the complexity and variety of data grow also (number of variables p grows with n, and maybe much larger than n).
- Inference is not as simple as removing estimation uncertainty - draw conclusions from the data and interpret
- Just exploration, visualization and model building for Big Data requires statistics
- With Big Data comes also PEV (probability of extreme values) - spurious correlations
- COD (curse of dimensionality) can be overcome with parametric modeling, but then we need to understand the assumptions that lie behind these models.

The "V"s of Big Data

4 Vs

- VOLUME: exploratory data analysis (EDA), low-rank or low-dimensional representations of data, visualization. Different infrastructure. `Big...` `R` projects + distributed computing resources.
- VARIETY: many variables (more than samples, $p > n$), different types, sources. Statistics looks a bit different when $p = p(n)$!
- VERACITY: convenience sampling, missing values, different quality, data cleaning and preparation
- VELOCITY: online estimation, parallelization, subsampling

Beyond Vs

- Visualization: another V? Bandwidth limit to what the human eye can process. Efficient and informative dimension reduction.
- Extracting information from large models, interpretation.

Overview of the Course

Themes

- Model building and prediction (regression and classification)
- Data representation (low-rank and low-dimensional approximations)
- Clustering
- Large-sample methods

Settings

- Big p: high-dimensional data
- Big n and p.
- Big n: subsampling, divide and conquer

We will explore classical methods within each theme and then look at how these methods have been adjusted to work in big data settings.

Regularization

- High-dimensional modeling has been an active research field in statistics for the past 10-15 years
- First principles: to build a model with p parameters we need $n > p$ samples
- Assumptions: IF we assume that the model is *sparse*, i.e. only few of the p variables actually relate to our outcome then we can overcome the high-dimensional problem and fit models even with $p > n$!
  - How do we fit sparse models?
  - Tractable sparse models solutions come at the price of *bias* but research within the last couple of years has provided *de-biasing* methods
- Theory: often assumptions about p growing with n but "not too fast", often difficult to verify assumptions about the sparse structure.

Big p! - regularization, sparse modeling

- Key concept in statistics is the BIAS-VARIANCE trade-off
- When $p > n$ variance goes to infinity (models cannot be uniquely identified).
- If we allow for some bias, by assuming some restricting structure like sparsity, we can reduce variance.
- Recent work on de-biasing such sparse estimates has improved performance of these methods.

BUT.... is the world sparse???

Data representations

- Matrix representations to capture and summarize structure in big data.
- Traditional methods like PCA, SVD - Problems with consistency, spurious correlations,...
- PCA: as p grows, not necessarily the case that a few components explain most of the variance. Theory for high-dim PCA relies on sparse (spiked) assumptions on the data structure.
- How even compute? Fast and scalable solutions using e.g. random projections.
- Clustering and pattern discovery for big p and n.

Big n

Large-sample analysis

- When n is large, p-values are essentially meaningless.
- Why? "Every model is wrong" so the p-values reflect model approximation errors.
- Be Bayesian instead? (No one true model but a distribution of models).

Sub-sampling

- Methods for dealing with large n: blocking or subsampling of data
- Basis for this: each subsample big enough for asymptotics to have kicked in
- Then we can show that aggregating estimates from each block is as good as running on all data (which we can't do when n is massive).

Overview

Topics

- Model building: regression and classification
- Dimension reduction/Clustering
- Subsampling, blocking procedures.

Classification:

Goals and setup:

- Build a predictive model/algorithm to discriminate between known groups
- Need a data set with know group labels on which we can construct/learn our model.

Combine with dimension reduction/feature selection:

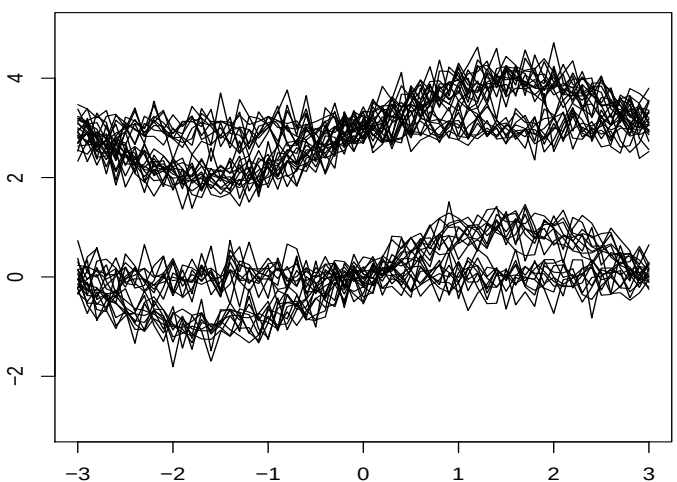
- increased interpretability - which features discriminate between the groups?
- improved performance - don't train your model on features that are unrelated to the groups.
- some methods can't be applied in a high-dimensional setting so you need to reduce the number of features first.

Clustering:

Goals and setup:

- Finding groups in data
- Summarize data this is otherwise difficult to "get a feeling for".
- Data exploration

This is a more difficult task than classification in the sense that the goal is *subjective*. What is a group? A set of observations separated from the rest? A set of observations close together? What is meant by "close"?



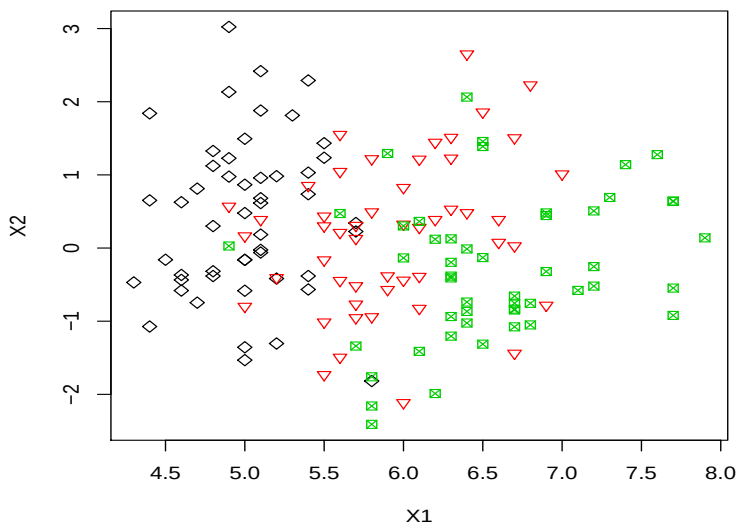
Example: Curve data. Each observation is a feature measured over time.

2 groups? 4 groups? Depends what you think close means. Dynamic behaviour of interest or only mean offset value?

Clustering:

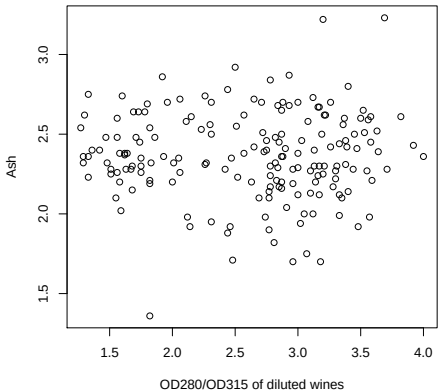
Combine with dimension reduction/feature selection:

- Which features are "responsible" for group separation?
- Some methods can't be applied in a high-dimensional setting so you need to reduce the number of features first. In a very high-dimensional setting "closeness" looses meaning (curse of dimensionality).

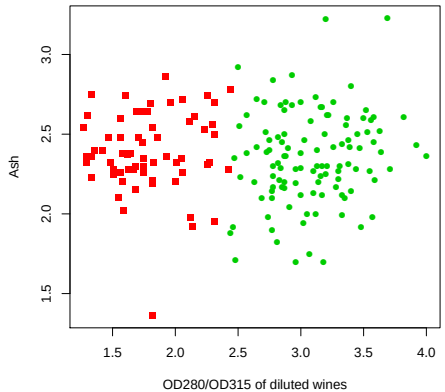


Group separation in feature X1 but not feature X2.

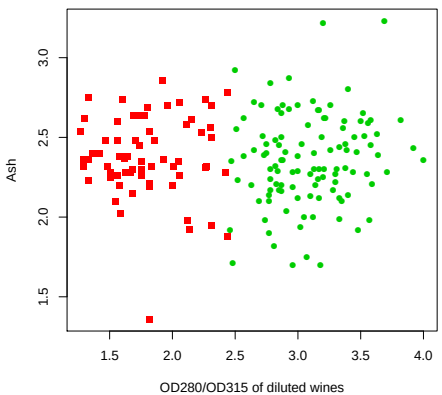
Clustering



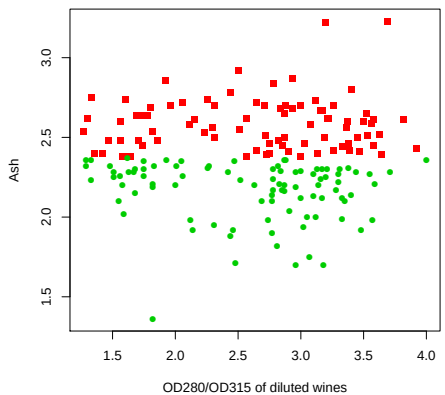
(a)



(b)



(c)



(d)

(a) is the original data which hints at a group presence along the first coordinate but not the second. (b) is a clustering result when we use both features and (c) is when we use only the first feature - these agree. (d) is the clustering we get if we use only the second feature - groups are formed but they are not well separated.

Discussion Problems

First Trimester Screening: Measures of hCG (human chronic gonadotropin) levels in blood serum as well as NT (nuchal translucency, a measure of the amount of fluid around the neck of the fetus) are indicators of chromosomal abnormalities. Can we learn from data how to identify fetuses with chromosomal abnormalities?

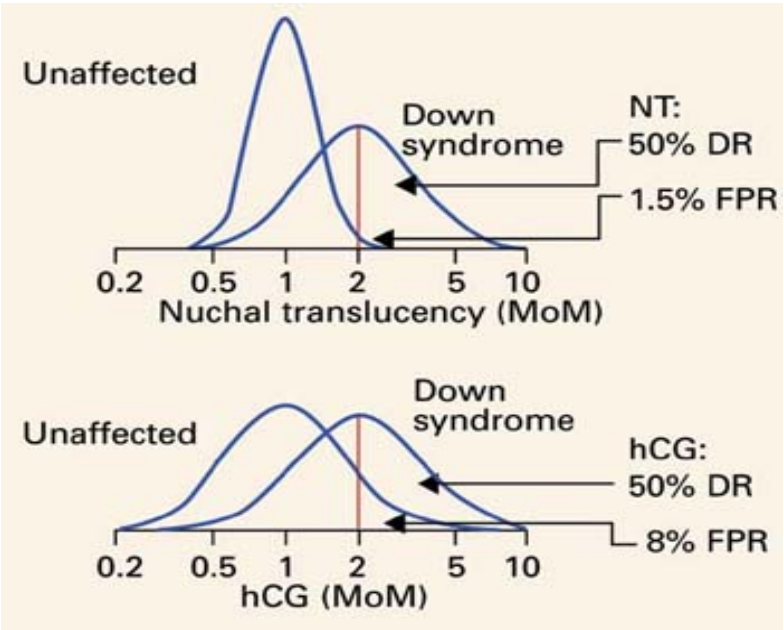
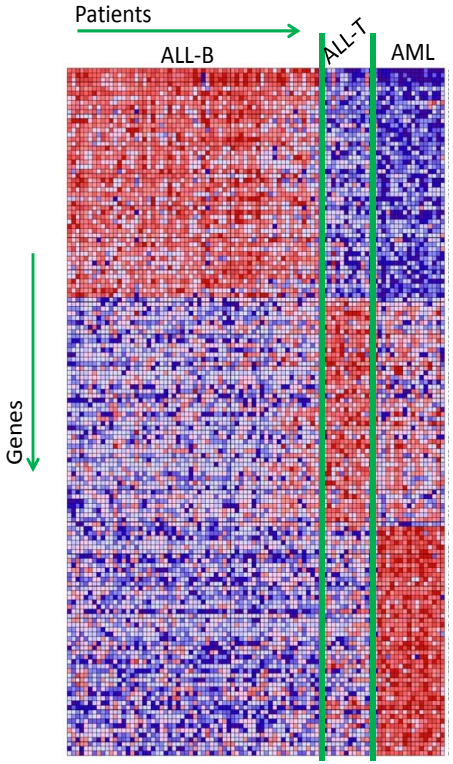


Figure from "New choices in prenatal screening for Down syndrome" by J.A. Canick appearing in OBG Management, Clinical Reviews, Dec 2005, Vol 17, 12

Challenges? Clearly a lot of variability in the affected population. How does one pick a decision threshold? FPR = false positive rate, the proportion of normal fetuses that our chosen threshold identifies as abnormal.

Genomic diagnostic tool for leukemia: Measures of activity levels of 1000s of genes for patients with three different kinds of leukemia.



Colors indicate activity level for each gene in each patient.

Challenges? It's easy to spot the differences between the three different kinds of leukemia. We could practically choose any of the genes (or a pair) to discriminate between all three cancers. Not so easy to identify *the key* genes that are responsible for discriminating between the cancers - is there such a key set? redundant information? finding groups of genes that discriminate between the cancers in a similar way.

Classification - some notation

Data $\{y_i, x_i\}_{i=1}^n$. Index i is the observation number, there are n observations in this data set.

$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ . \\ x_{ip} \end{pmatrix}$ is a p-dimensional feature vector. That is, we have p variables in our data set.

y_i is the class label (group membership) of observation i.

$y_i \in \{1, 2, \cdots, C\}$ if there are C distinct groups or classes in the data set. We can of course also use other kinds of labeling formats, e.g. "Cancer", "Not Cancer" etc.

Classification - some notation

Goal

- From n observations, construct a *rule* $c()$ that maps X (feature space) to $\{1, 2, \cdots, C\}$ (group labels).
- We want the rule to *generalize* to future data (prediction)

Example: You have data from 250 male patients; age, prostate volume and level of PSA (a protein that, if detected at increased levels in blood, may be indicate of cancer). In general, older men are at higher risk for cancer. Increased prostate volume is also a risk factor.
We look at the data and come up with a rule:
If
$age > 70$ *and* *volume* $> 10\%$ *increase* *andPSA* $> 3 * normal$ $\rightarrow$ predict cancer.

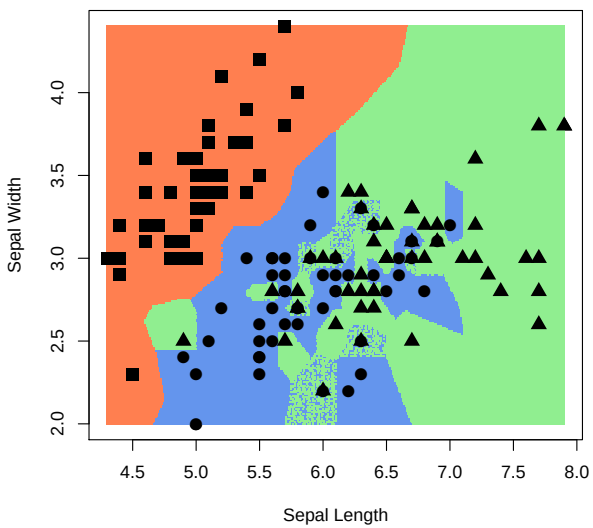
Classification - some notation

The explicit numbers (70,10%,3*) are based on our data $\{y_i, x_i\}_{i=1}^n$. We emphasize this by denoting the resulting rule as $\hat{c}()$ where the "hat" on the c is used to denote that parameters/rule thresholds are estimated from data.
We predict the class label of a new patient based on his feature data x_{new} as $\hat{y}_{new} = \hat{c}(x_{new})$.
If we know the true label of this observation, y_{new}, we can compare the outcome of our trained rule to this:
Prediction error or misclassification: $y_{new} \neq \hat{y}_{new}$

Loss function:
It is common to simply *count* the number of mistakes a rule makes. This measure of performance is called 0-1 loss.
Error rate $= \frac{1}{n}\sum_{i=1}^{n} 1\{y_i \neq \hat{c}(x_i)\}$, where $1\{.\}$ is an indicator function that takes on value 1 if what is inside the curly brackets is true.
We can also write it on a more general form:
Error rate $= \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{c}(x_i))$, where $L(.,.)$ is called the *loss function*. In regression we usually use the L2-loss, $(y_i - \hat{c}(x_i))^2$.

An example - kNN

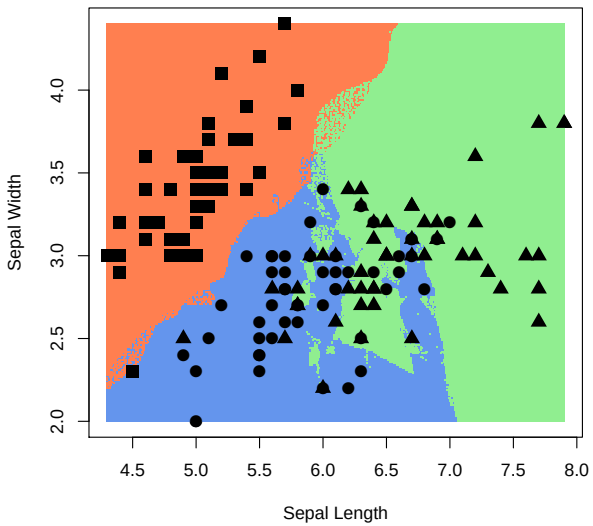
kNN - k nearest neighbor is a very intuitive classification method. For observation x_i we identify the closest neighbors in x-space. We use a majority rule based on these neighbors to make a class prediction for x.



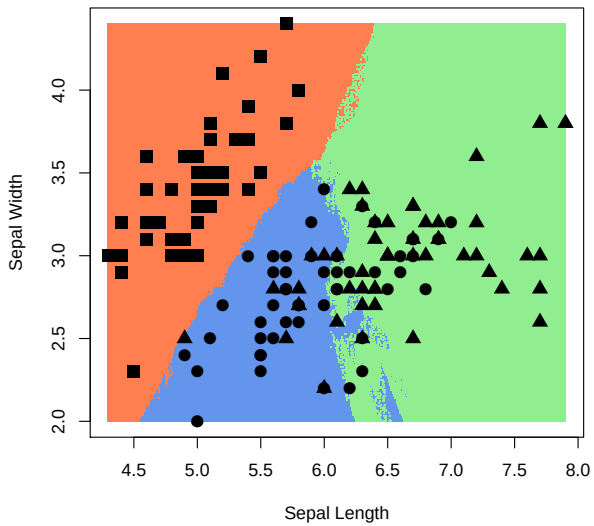
Here we use $k = 1$ neighbors. Each observation is then predicting the label of a region around it comprising points in x-space closer to this observation than any other observation. As you can see, this may results in a very irregular rule with isolated "islands" of a particular class.

An example - kNN

The kNN rule can be written as
$\hat{c}(x_i) = \arg\max_{c \in \{1,2,..,C\}} \sum_{j \in N(x_i)} 1\{y_j = c\}$
$N(x_i)$ is the neighborhood of observation x_i and we simply count the number of observation with a certain label c we see in this neighborhood. We vote for the class c with the largest count.



$k = 5$



$k = 20$

The larger the neighborhood, the smoother the classification rule is - more cohesive regions that are predicted to correspond to a certain class.

An example - kNN

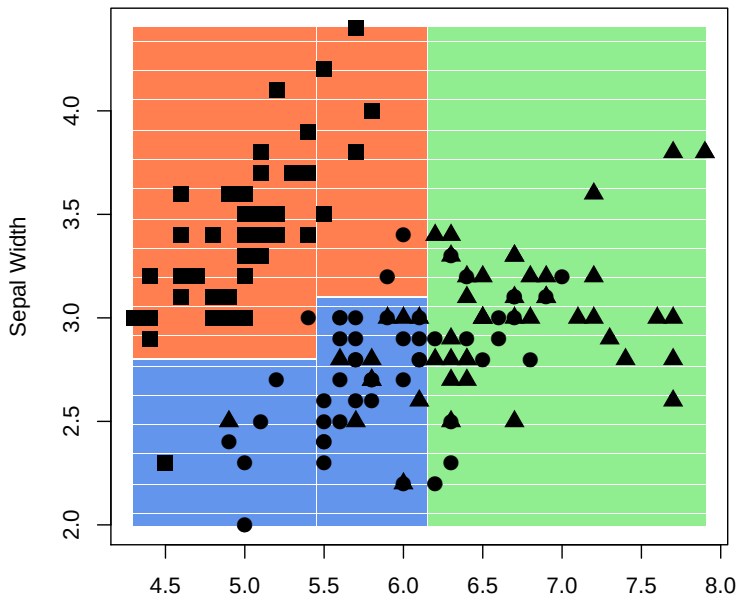
How to pick the size of the neighborhood? Small neighborhood = irregular class boundaries. Large neighborhood = smooth boundaries. How do we know which is best?
Well, we can always check to see how well the rule performs!
However, you can't both train the rule and evaluate/test it on the same data set. If you do that for kNN you would always choose k=1 since each observation would be predicting itself and the rule would never make a mistake.
We are interested in rules that *generalize* to future data!

Another example - CART

CART stands for Classification and Regression Trees.
CART is a rule-type that constructs rectangular regions in x-space and allocates a class label to each region. Rectangular regions are defined by thresholds on features, e.g.
$\{40 < age < 55\} \cap \{PSA < 3 * normal\}$. To this region we allocate the label "Not Cancer".

Another example - CART

The CART rule can be written as
$\hat{c}(x) = \arg\max_{c \in \{1,2,..,C\}} \sum_{m=1}^{M} 1\{x \in R_m\} \sum_{i \in R_m} 1\{y_i = c\}$
R_m denotes a rectangular region in x-space and there are a total of M such regions in the rule. The indicator $1\{x \in R_m\}$ states that to predict the label at location x you only need to consider the rectangle it falls in. The sum $\sum_{i \in R_m} 1\{y_i = c\}$ counts up the observations of label c in this rectangle. We vote for the class c that has the maximum count.



Another example - CART

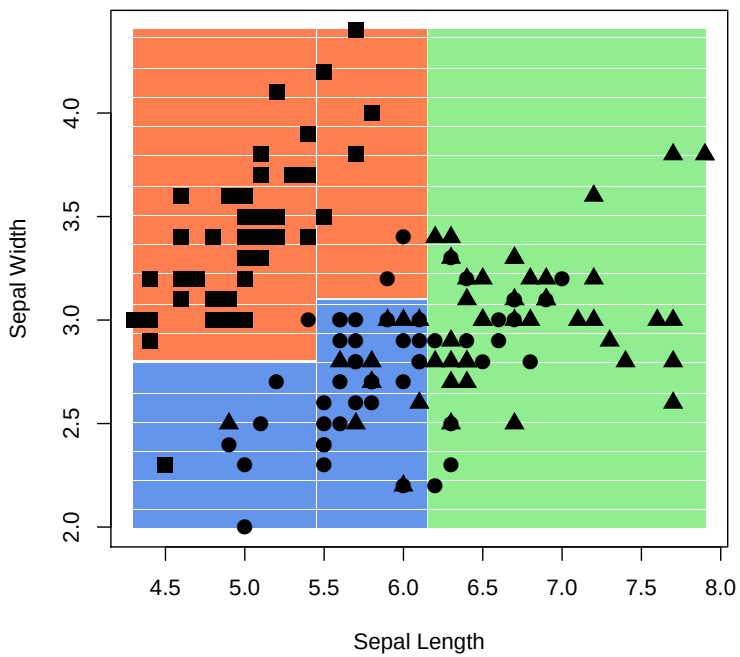
We don't allow for any shape R_m because then the training process would be too difficult (too many combinations of features and thresholds to consider).
Instead we use *binary splits* to form the rectangles.

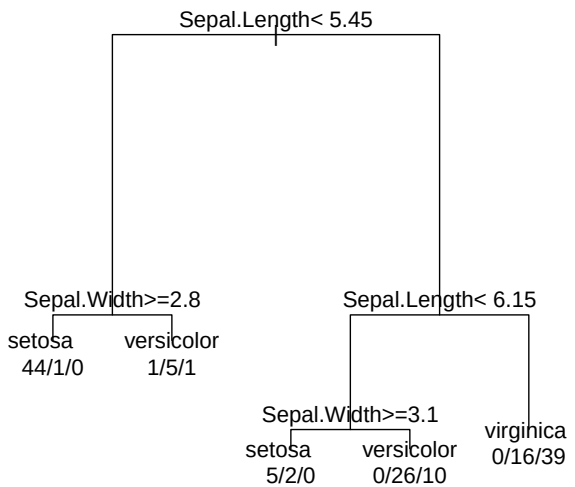
1. Choose a feature x_j and split the data in two parts: $R = \{i : x_{ij} > t_j\}$ and $L = \{i : x_{ij} \leq t_j\}$.
2. The feature j and the threshold value t_j are chosen to make the right (R) and left (L) parts of the data as "pure" as possible, i.e. dominated by one of the classes.
3. Repeat steps 1-2 separately for the R and L data sets.

If you keep on iterating this scheme you build smaller and smaller rectangular regions containing smaller sets of data with as similar class labels as possible.

Another example - CART

One of the reasons CART is so popular is because the rectangular region representation of the rule can also be depicted as a decision tree. Each data split (steps 1-2 above) can be drawn in order as a tree. At each node in the tree we perform one of data splits. The "leaves" of the tree represent the rectangles formed by the final rule.





A little bit of everything!!!

CART is notoriously unstable, meaning small perturbations of the data can alter the tree substantially.

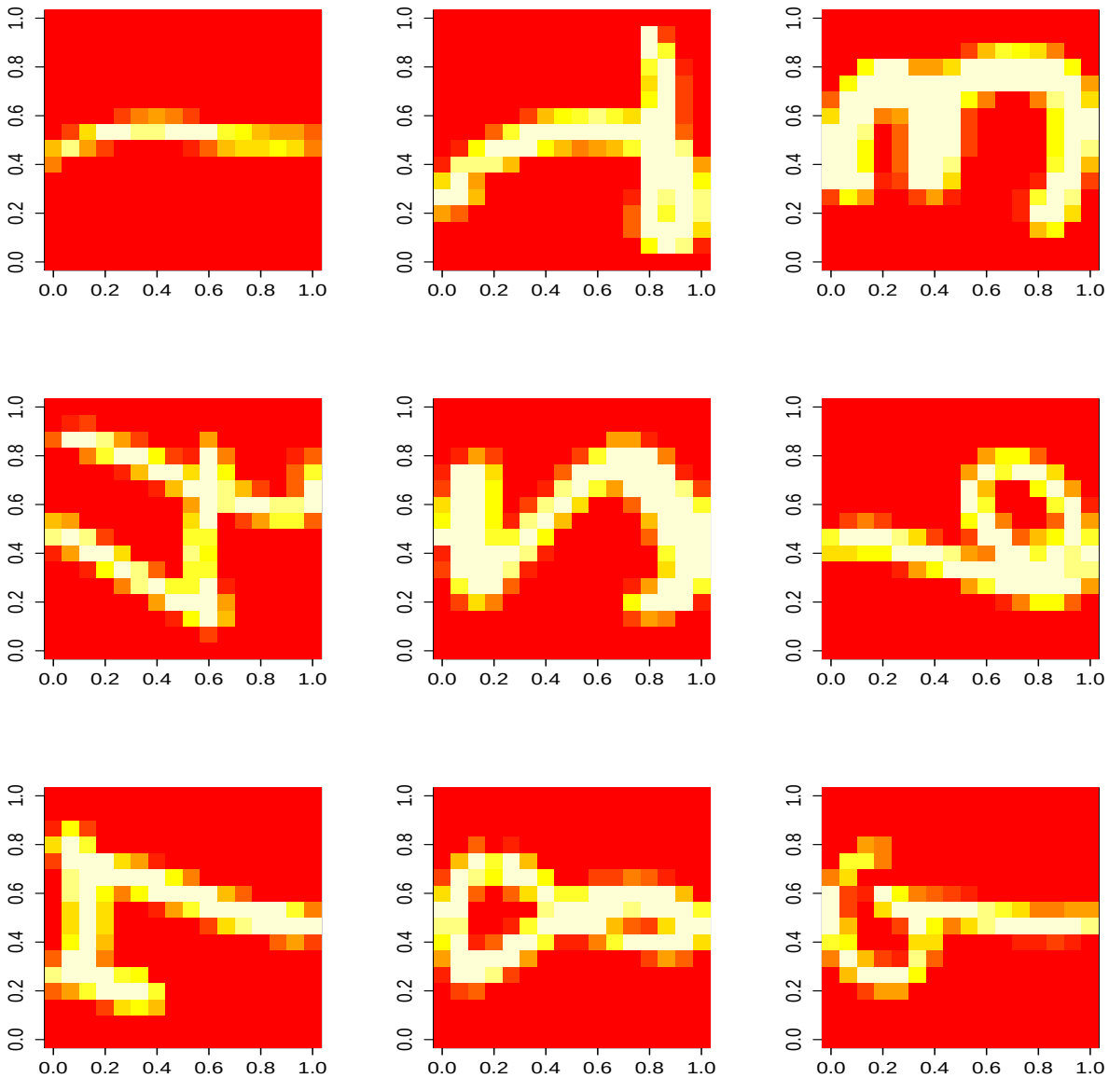
Random Forests

- Sub-sampling: use blocks of data to build many trees and combine
- Random sets of variables: to further enrich the models generated, we choose a random set of variables to split on inside each node of the tree
- Aggregate: majority decision or mean of predictions from each tree

Random forests is a procedure that captures many aspects of Big Data statistics: Project 1!!!

Handwritten digits

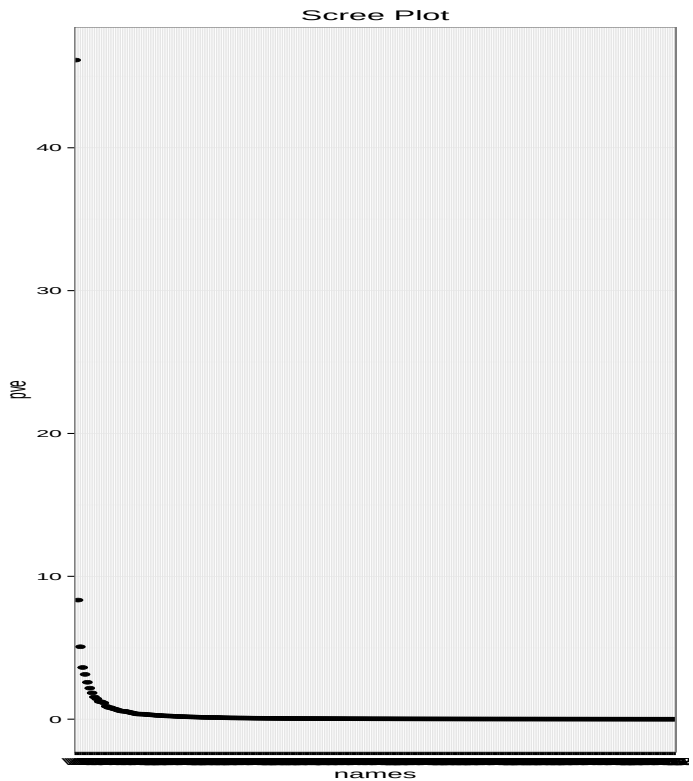
7291 handwritten digits. Data: images 16x16 pixels.

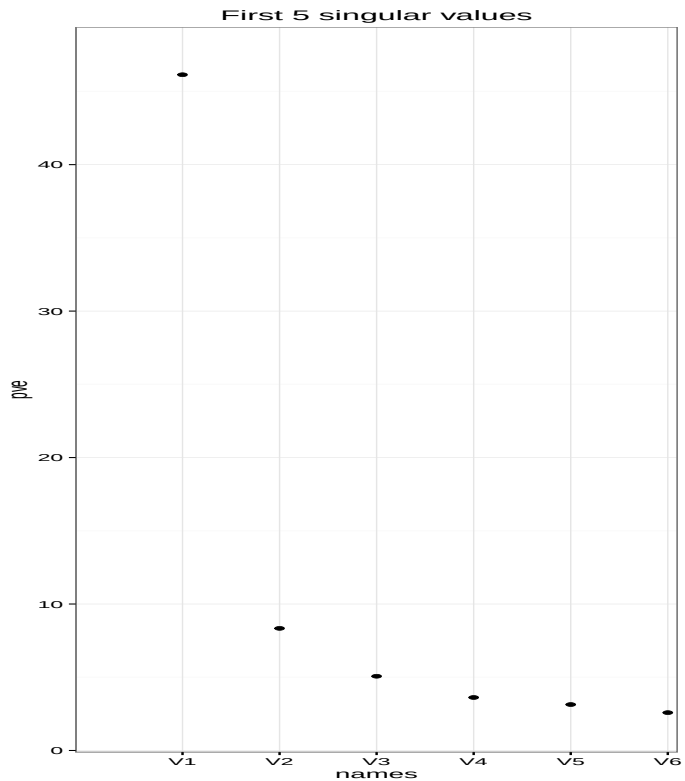


Handwritten digits

7291 handwritten digits. Data: images 16x16 pixels. `svd.obj <-`

```
svd(zip.train[,-1])
colnames(svd.obj$v) = paste0("V",1:7291)
rownames(svd.obj$v) = paste0("Sample",1:256)
svd.scree(svd.obj, subr=5,
            axis.title.x="Full scree plot", axis.title.y="Var
Explained")
```

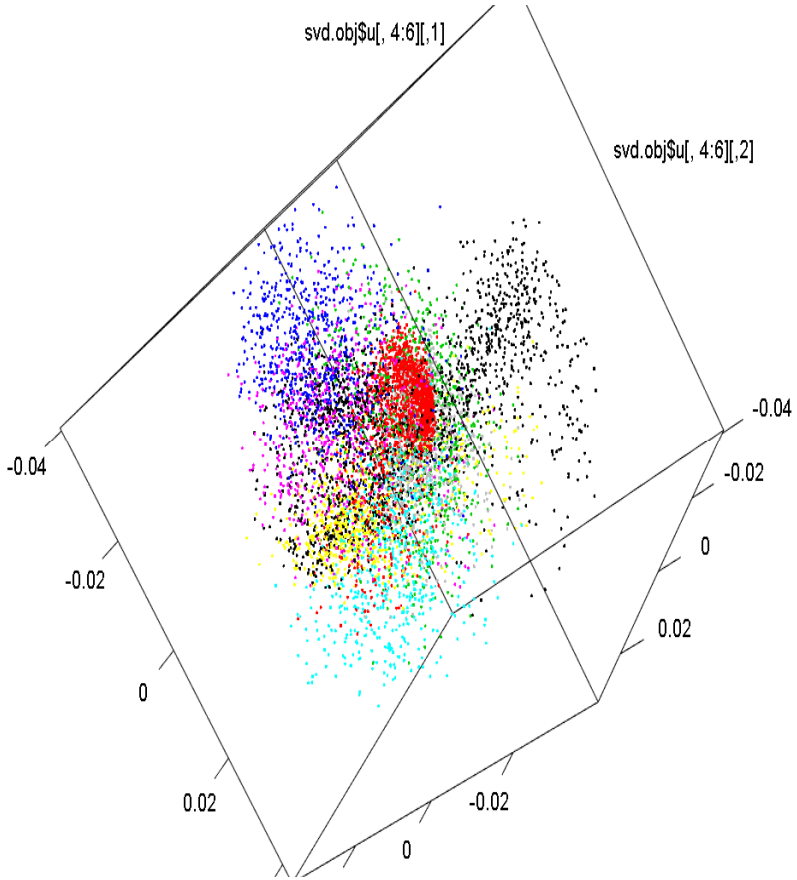




Handwritten digits

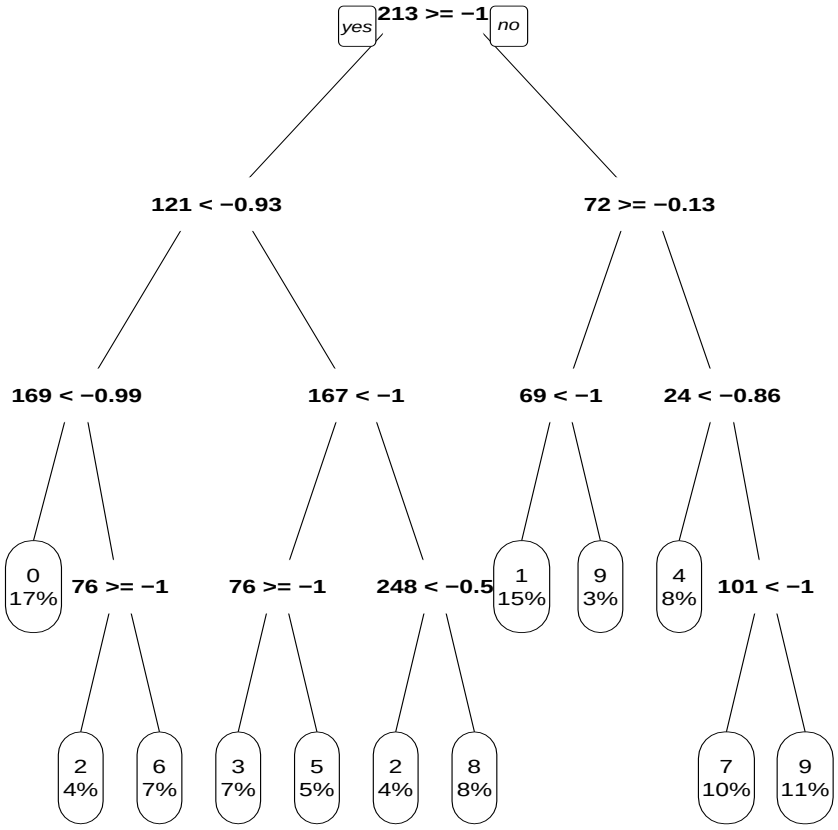
7291 handwritten digits. Data: images 16x16 pixels.

```
plot3d(svd.obj$u[,1:3],col=zip.train[,1]+1)
legend3d("topright", legend =
             paste('Type', c(unique(zip.train[,1]))),
             pch = 5, col=seq(1,10), cex=1, inset=c(0.02))
```



Handwritten digits

7291 handwritten digits. Data: images 16x16 pixels.




```
tree1<-rpart(number .,data=Numbers,
             control=rpart.control(cp=0.01, minsplit=10,xval=3)
             ,method="class")
prp(tree1,extra=100)
```

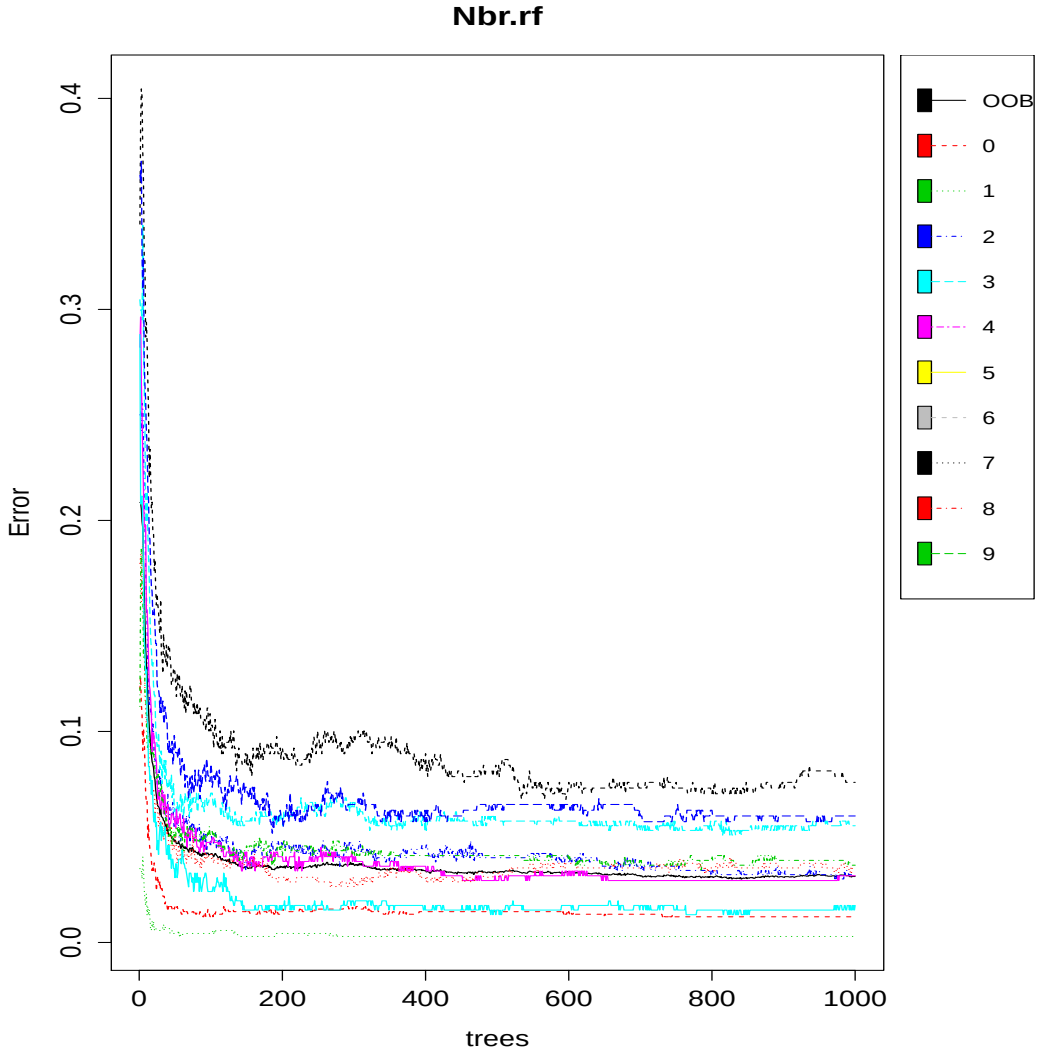
Random Forest

- Lots of R-packages!
- CART with subsampling and random variable selection for each node

```
library(randomForest)
Nbr.rf<-randomForest(y=Numbers[ii,1],x=Numbers[ii,-1],
             ytest=Numbers[-ii,1],xtest=Numbers[-ii,-1],
             ntree=1000, proximity=T, keep.forest=TRUE,
             importance=TRUE)
```

Handwritten digits

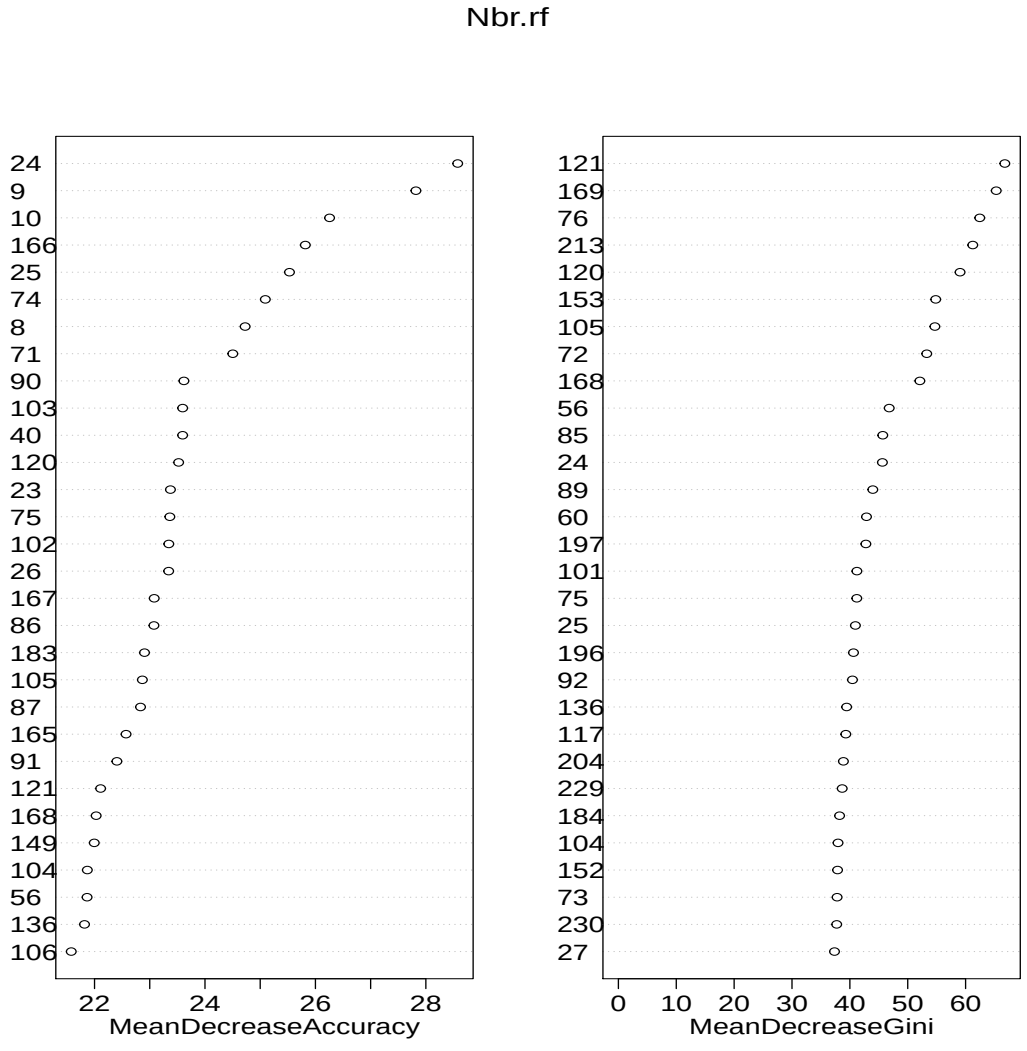
7291 handwritten digits. Data: images 16x16 pixels.




```
plot(Nbr.rf)
```

Handwritten digits

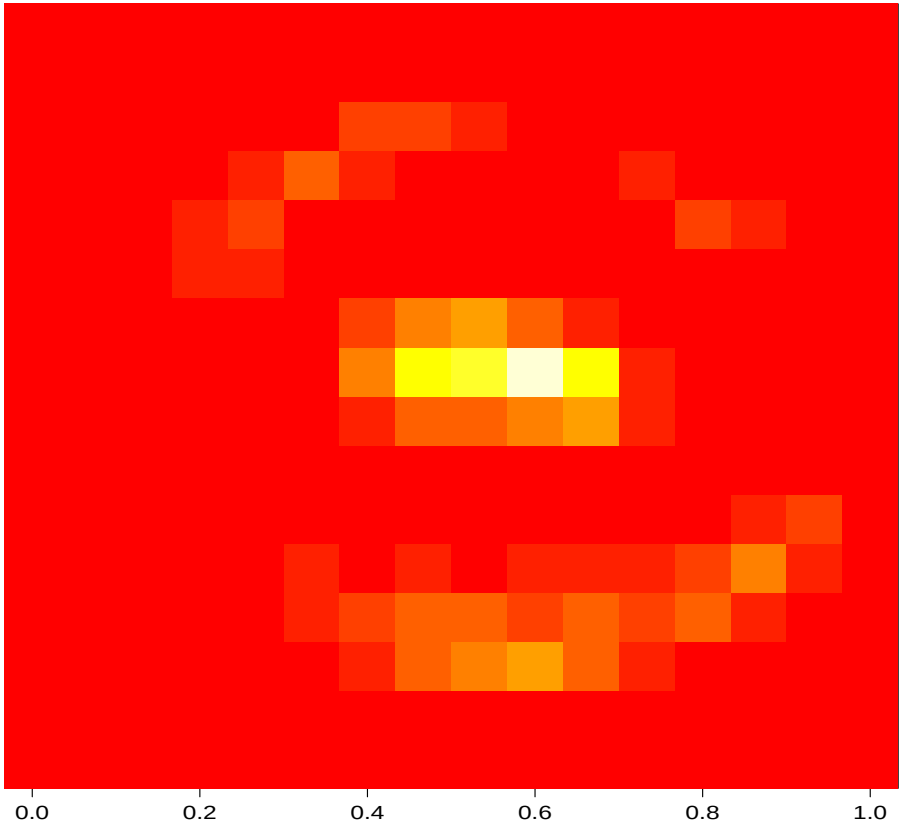
7291 handwritten digits. Data: images 16x16 pixels.



`varImpPlot(Nbr.rf)`

Handwritten digits

7291 handwritten digits. Data: images 16x16 pixels.




```
image(matrix(Nbr.rf$importance,16,16,byrow=T))
```